FUJITSU

Fujitsu Software
Technical Computing Suite V4.0L20

Job Operation Software
Administrator's Guide
for Job Management

J2UL-2456-02ENZ0(09)
September 2023

Preface

Purpose of This Manual

This manual describes the Job Operation management function settings and operation methods of the Job Operation Software included in Technical Computing Suite.

Intended Readers

This manual is intended for the administrator who operates and manages the system with the Job Operation Software.

The manual assumes that readers as follows.

- Linux basic knowledge is required.
- Understand the overview of Job Operation Software in the "Job Operation Software Overview."
- Knowledge of container virtualization technology and Docker in Linux.
- Knowledge of Linux virtualization technology (KVM) and virtualization management utilities (libvirt and virsh)

System management administrators are requested to read "Job Operation Software Administrator's Guide for System Management."

For details on maintenance and troubleshooting, see "Job Operation Software Administrator's Guide for Maintenance" and "Job Operation Software Troubleshooting."

Organization of This Manual

This manual is organized as follows.

Chapter 1 Overview of the Job Operation Management Function

This chapter provides an overview of the job operation management function.

Chapter 2 Details of the Job Operation Management Function

This chapter describes in detail the functions provided by the job operation management function.

Chapter 3 Job Operation Management Function Settings

This chapter describes the necessary setting items for using the job operation management function.

Chapter 4 Operation with the Job Operation Management Function

This chapter describes specific kinds of operation methods for using the job operation management function.

Appendix A Invalid Values for Job Statistical Information at State Transition

This appendix describes job statistical information that is invalid during state transition.

Appendix B Settings Related to Execution in MPI Processing Systems Other Than Development Studio

This appendix describes the settings for executing MPI programs in MPI processing systems other than Development Studio on the Job Operation Software.

Appendix C Settings for Using GPUs [PG]

This appendix describes the settings for job execution using GPUs.

Appendix D Settings for Using Singularity [PG]

This appendix describes the settings for executing Singularity in Docker mode in the job execution environment. Singularity is container virtualization software for HPC.

Appendix E How to Use Dynamic Parameters in Startup Configuration Files (Docker Mode) [PG]

This appendix describes how to configure the startup configuration files for the job execution environment in order to apply values appropriate to the environment at the job runtime.

Appendix F Defined items of the job ACL function

This appendix describes the defined items of the job ACL function.

Notation Used in This Manual

Representation of units

The following table lists the prefixes used to represent units in this manual. Basically, disk size is represented as a power of 10, and memory size is represented as a power of 2. Be careful about specifying them when displaying or entering commands.

Prefix	Value	Prefix	Value
K (kilo)	10^{3}	Ki (kibi)	2^{10}
M (mega)	10^{6}	Mi (mebi)	2^{20}
G (giga)	10^{9}	Gi (gibi)	2^{30}
T (tera)	10^{12}	Ti (tebi)	2^{40}
P (peta)	10^{15}	Pi (pebi)	2^{50}

Notation of model names

In this manual, the computer that based on Fujitsu A64FX CPU is abbreviated as "FX server", and FUJITSU server PRIMERGY as "PRIMERGY server" (or simply "PRIMERGY").
Also, specifications of some of the functions described in the manual are different depending on the target model. In the description of such a function, the target model is represented by its abbreviation as follows:
[FX]: The description applies to FX servers.
[PG]: The description applies to PRIMERGY servers.

Administrators

The Job Operation Software has different types of administrator: system administrator, cluster administrator, and job operation administrator. Unless otherwise noted, the descriptions in this manual apply to functions for system administrators and cluster administrators. The term "administrator" in the text usually means a system administrator and cluster administrator.

Path names of the commands

In the examples of the operations, the path names of the commands in the directory /bin, /usr/bin, /sbin or /usr/sbin might not be represented by absolute path.

Symbols in This Manual

This manual uses the following symbols.

Note

The Note symbol indicates an item requiring special care. Be sure to read these items.

See

The See symbol indicates the written reference source of detailed information.

Information

The Information symbol indicates a reference note related to Job Operation Software.

Export Controls

Exportation/release of this document may require necessary procedures in accordance with the regulations of your resident country and/or US export control laws.

Trademarks

- Linux(R) is the registered trademark of Linus Torvalds in the U.S. and other countries.
- Red Hat and Red Hat Enterprise Linux are registered trademarks of Red Hat, Inc. in the U.S. and other countries.
- Intel is a trademark of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
- All other trademarks are the property of their respective owners.

Date of Publication and Version

Version	Manual code
September 2023, Version 2.9	J2UL-2456-02ENZ0(09)
March 2022, Version 2.8	J2UL-2456-02ENZ0(08)
November 2021, Version 2.7	J2UL-2456-02ENZ0(07)
August 2021, Version 2.6	J2UL-2456-02ENZ0(06)
March 2021, Version 2.5	J2UL-2456-02ENZ0(05)
January 2021, Version 2.4	J2UL-2456-02ENZ0(04)
December 2020, Version 2.3	J2UL-2456-02ENZ0(03)
September 2020, Version 2.2	J2UL-2456-02ENZ0(02)
June 2020, Version 2.1	J2UL-2456-02ENZ0(01)
March 2020, Second version	J2UL-2456-02ENZ0(00)
January 2020, First version	J2UL-2456-01ENZ0(00)

Copyright

Update history

Changes	Location	Version
Fixed errata.	-	2.9
Added an item of power information acquisition status to the periodic collection of job statistical information.	2.6.3	2.8
Added description of setting files for using GPU.	C.3	
Added description of setting files for using Singularity.	D.2	
Added notes on using Docker mode script on RHEL8 based PRIMERGY compute nodes.	Appendix E.	
Improved description of resource units. Improved description of ResourceUnitName in the pmpjm.conf and the pmrsc.conf files.	2.2 3.5.1.1 3.5.2.1	2.7
Improved procedure for changing JobMem in the parsc.conf and the pmrsc.conf files.	3.4.3.2 3.5.2.2	
Changed the default values for the setting items JobSchedulingTargetLimit and JobSchedulingTargetMode in the papjm.conf and pmpjm.conf files.	3.4.1 3.4.1.2 3.4.1.5 3.5.1 3.5.1.9	2.6
Fixed description of Image specification when Docker mode and NeedCustomImage specification is true.	3.5.7.3	

Changes	Location	Version
Improved description about overtaking the order in which jobs run by the backfill function.	2.5.4.1 2.5.4.2	2.5
Added notes for UDI specification in Docker mode. - Only one job resource manager exit script can be registered per resource unit. - About the changing the path of the commands in the sample script.	3.5.7.7	
Changed NVIDIA Tesla to NVIDIA V100 and A100.	Appendix C	
Added notes for the systems with a single node serving as all of the system management node, compute cluster management node, and login node.	3.4.5 3.5.7 3.5.7.3	2.4
Added setting for RHEL 8 in the ODBC configuration (.odbc.ini file).	3.3	2.3
Added a description of the settings required to use Intel MPI 2019.	B.4	
Added a note about job statistical information that require to set multiple items for its definition.	3.4.2.2	2.2
Changed the description of StepJobAcceptDate and StepSchedUnit in the pjs.conf file.	3.4.5	
Improved description of creating startup configuration file for the Docker mode.	3.5.7.4	
Added description about the disturbance to the job performance due to the memory recovery process of the OS.	3.7	
Modified the output example due to the specification change of the output of the pmpjmopt command.	4.2.7	
Added a note about the tcs-bare.conf startup configuration file for the normal mode.	C.4	
Improved the description of the range specification for the item NodeID set in the CustomResource section of the pmpjm.conf file.	3.5.1.6	2.1
Fixed the procedure for applying the contents of the job execution environment information file to the system.	3.5.7.3	
Added procedure for avoiding disturbance to job execution performance.	3.7	
Added notes on changing resource units and resource groups running jobs with the pmalter command.	4.2.3	
The communication path of a job can now be dynamically changed when a Tofu interconnect link goes down.	2.4.3 4.1.2 Appendix F	2
The elapsed time limit value of a running job can be changed.	2.5.2.2 3.4.1 3.4.1.1 4.2.3	
The behavior of job information output at deleting the job in a QUEUED state can be specified.	3.4.1 3.4.1.1 Appendix F	
Fixed the description of conditions under which job history information displayed by the -H option of the pjstat command is removed by changing job statistics or custom resource settings.	3.4.2.5 3.5.1.9	
KVM mode in job execution environments is supported.	3.5.7	
Use of GPUs by jobs is now supported.	Appendix C	
Execution of Singularity in job execution environments is now supported.	Appendix D	
File system mount points can now have parameters that changes dynamically to match the environments in Docker mode in job execution environments.	3.5.7.3 Appendix E	
Changed the look according to product upgrades.	-	

Contents

Chapter 1 Overview of the Job Operation Management Function

This chapter describes the purpose of the job operation management function and provides a functional overview.

In a computer system that performs scientific computations, an enormous number of nodes execute parallel processing to improve computational performance. The following conditions must be met for this kind of ultra-large-scale system.

- The system can efficiently process an enormous number of jobs submitted by many end users.
- There is a capability for control in accordance with the operation policy defined by the job operation administrator.
- The job operation administrator can easily control and monitor job operations.
- Though consisting of many nodes, the system can be seen as one virtual parallel computer.

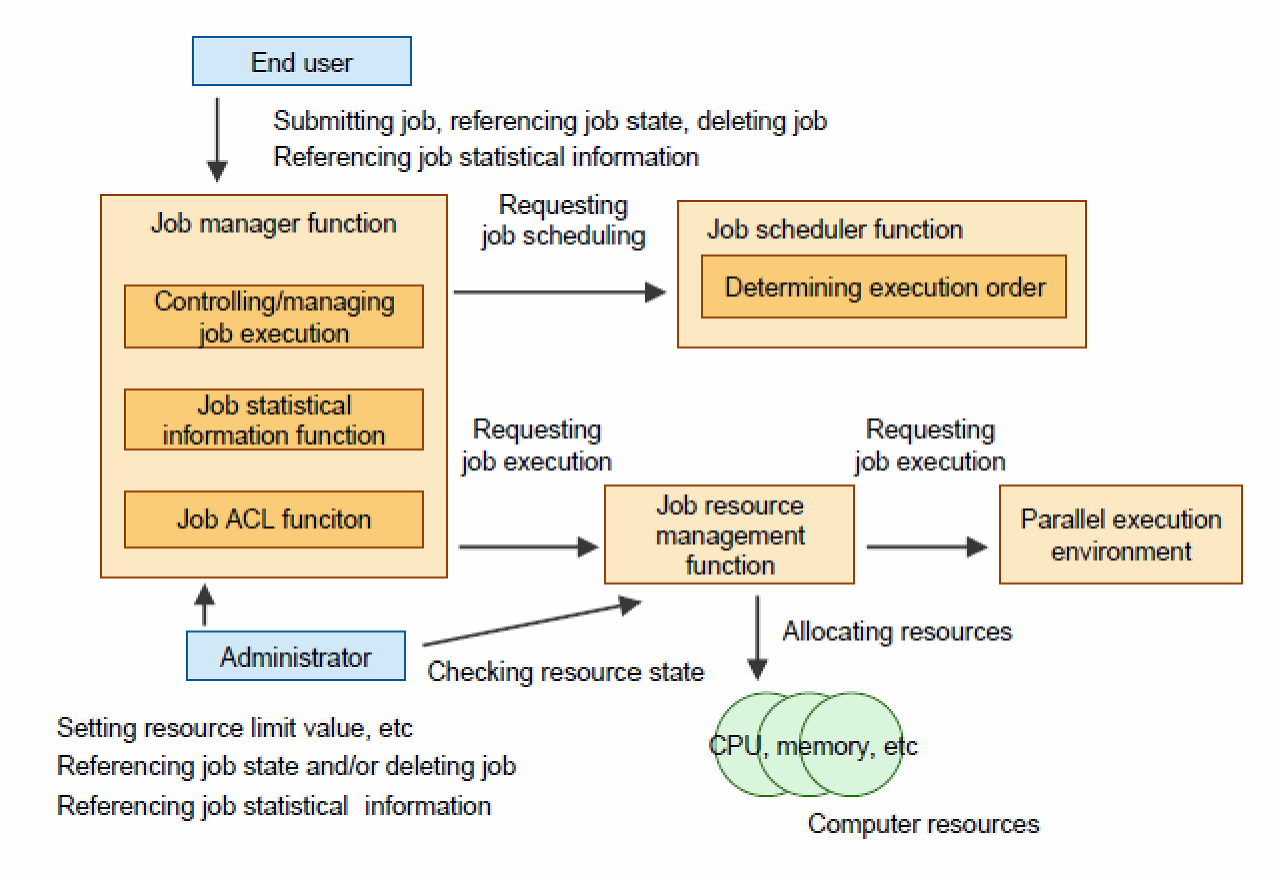
The Job Operation Software provides a "job operation management function" that satisfies these requirements.

The job operation management function roughly consists of the following functions.

Table 1.1 Features of the job operation management function

Name	Description
Job manager function	Function for job execution control and job management. - Job submission acceptance, information display, deletion, and execution request to the job resource management function - Job access control list (ACL) function, which controls access to the system by jobs - Job statistical information function, which records information such as the actual quantity of resources used by jobs
Job scheduler function	Function that allows multiple submitted jobs to be executed efficiently. - Selecting the resources to be allocated to jobs - Determining the execution order
Job resource management function	Function for managing the resources required by jobs. - Resource allocation - Resource release It requests job execution in the parallel execution environment.
Parallel execution environment	Mechanism of job process control for many nodes. With this mechanism, jobs treat the system as though they were running on one parallel computer.

Figure 1.1 job operation management function


End user
Submitting job, referencing job state, deleting job
Referencing job statistical information
Job manager function
Controlling/managing job execution
Job statistical information function
Job ACL funciton
Requesting job scheduling
Job scheduler function
Determining execution order
Requesting job execution
Job resource management function
Requesting job execution
Parallel execution environment
Administrator
Checking resource state
Allocating resources
Setting resource limit value, etc
Referencing job state and/or deleting job
Referencing job statistical information
CPU, memory, etc
Computer resources

Administrators can also take advantage of the following functions for flexible job operations.

Table 1.2 The functions for flexible job operations

Name	Description
Job execution environment customization function	The function that switches the software environment (job execution environment) for executing job programs according to user specifications.
Command API	The APIs for calling functions (job operation and information acquisition) from programs, equivalent to the commands provided by the job operation management function
Job information notification API	The APIs for job-related information notification of the programs used for processing specific to job operations. Examples of program include charge processing and job trace processing created by the job operation administrator. The programs are notified at the timing of job state transition.
Scheduler plugin function	The function that incorporates an original scheduling algorithm created by the job operation administrator into the job scheduler to replace the scheduling algorithm of the job operation management function.

Chapter 2 Details of the Job Operation Management Function

This chapter describes the job operation management function in detail.

2.1 Jobs

The job operation management function batch processes programs, created by end users, in units called "jobs." For details on jobs, see "What Are Jobs?" in "Chapter 1 Mechanism of Jobs" in "Job Operation Software End-user's Guide."

2.2 Resource Units and Resource Groups

For efficient job operation and management, the job operation management function adopts internal cluster structures called resource units and resource groups.

This section describes clusters, resource units, and resource groups.

Cluster

A cluster is a range of nodes grouped together by role in a functional hierarchy for efficient management of system components.

Resource unit

A job selection policy, such as on how to allocate computer resources, is applied to a range of nodes. This range of nodes is called a "resource unit." Job operations are performed in units of resource units.

Resource units are determined by logically dividing the nodes in a cluster.

Resource group

A unit of computer resources gathered in a resource unit that can execute a job is called a "resource group." The resource group is the smallest unit for a job operation. It is used to divide a job operation in further detail within a resource unit.

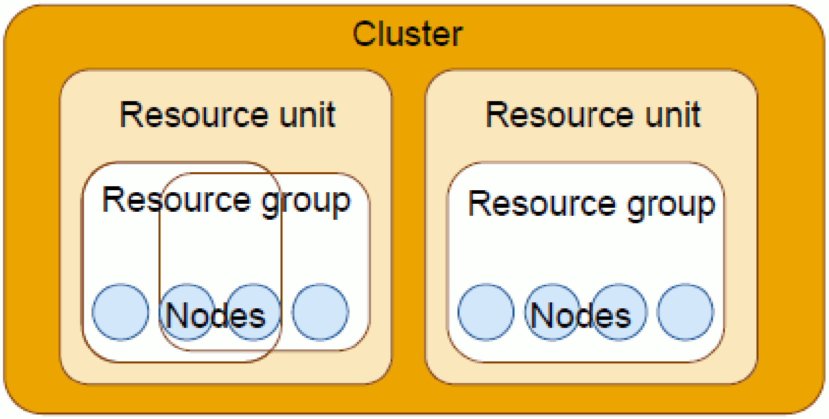
Each job is executed within a single resource group.

You can set a resource group range within any resource unit regardless of its structure for system configuration management, such as the node group structure. You can also use a configuration that shares computer resources among multiple resource groups.

From the perspective of information confidentiality, you can create a resource group to restrict users who do not have the privileges for this specific resource group from referencing the job execution status, etc.

The following figure shows the relationship between clusters, resource units, and resource groups.

Figure 2.1 Relationship between clusters, resource units, and resource groups



2.3 Roles of Users with Operation Administrator Privileges

The cluster administrator and job operation administrator play different roles in the job operation management function. The following table describes each administrator role.

Table 2.1 Role of each administrator privilege

Administrator privileges	Role
Cluster administrator	This administrator manages and operates an entire cluster. On behalf of the job operation administrator, the cluster administrator can manage and operate resource units and resource groups in the cluster.
Job operation administrator	This administrator manages and operates the resource units in a cluster and the resource groups in the resource units. The administrator can apply different operation policies to different resource units for operations.

2.4 Job Manager Function

The job manager function has the following functions.

- Job execution control

  This function controls and manages the workflow from job acceptance to execution to completion.

- Job operation support

  This function supports job operations for smooth job management.

- High availability of job operations

  This function allows job operation to continue even if an error occurs on a redundantly configured compute cluster management node or on the compute node running the job.

2.4.1 Job execution control

The job manager function controls the workflow related to job execution. The workflow includes job submission, execution, display, and deletion.

A job undergoes state transitions according to the processing by the job manager function. The following table lists the job states.

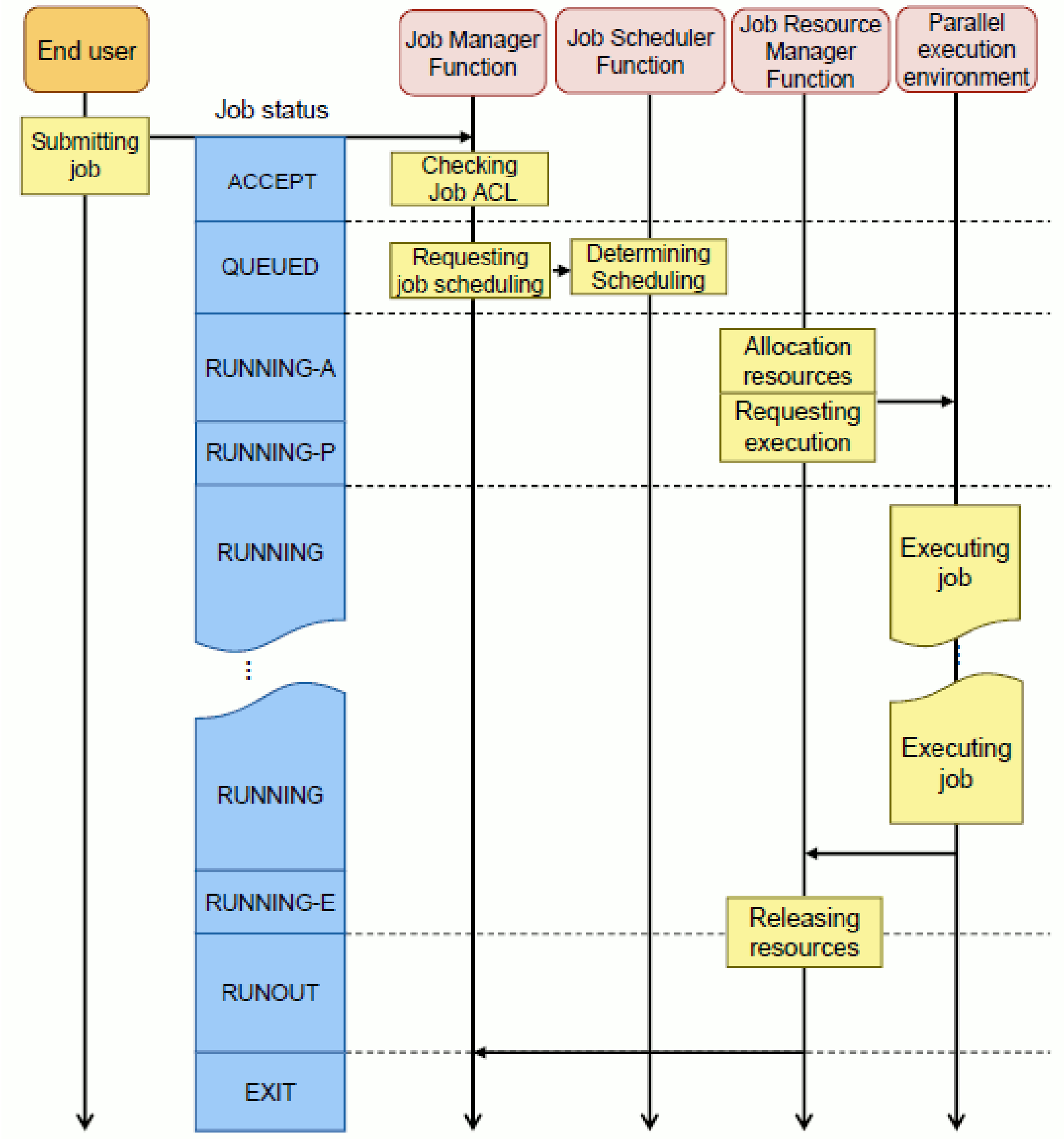
Table 2.2 Job states

State name	Description
ACCEPT	Checking whether the job satisfies the acceptance conditions (items restricted with the job ACL function)
QUEUED	Waiting for a turn to execute the job, which has been accepted
RUNNING-A	Reserving the resources required for job execution
RUNNING-P	Executing the prologue process
RUNNING	Executing the job
RUNNING-E	Executing the epilogue process
RUNOUT	Job end being processed
EXIT	Job end
REJECT	Job acceptance rejected
CANCEL	Job stopped by an instruction from the job submitter or the administrator
HOLD	Job execution canceled, with the job held in the submitted state

State name	Description
ERROR	Submitted job stopped, but kept in the submitted state, because of an error detected by the job operation management function

The following shows the basic actions of the job operation management function from job submission to execution to completion.

Figure 2.2 Processes and job states of the job operation management function



2.4.1.1 Job states and operations

This section describes the actions of the job operation management function from job submission to job completion.

1. Submitting a job

   When an end user submits a job, the job enters the ACCEPT state.

   Then, the job manager function checks whether the job satisfies the conditions defined by the job ACL function. If the conditions are satisfied, the job enters the QUEUED state. If the conditions are not satisfied, the job is rejected and enters the REJECT state.

2. Job scheduling

   When a job enters the QUEUED state, the job manager function requests the job scheduler function to schedule the job. The scheduling process evaluates the submitted job based on limit values and determines the job execution order and the selection of resources for jobs.

3. Job execution

   When determining the scheduling of a job, the job manager function or job scheduler function requests the job resource management function to execute the job. The job resource management function allocates resources to the job and returns the results to the requester. In this process, the job is in the RUNNING-A state, where it waits for resources to be reserved.

   When the resources are successfully allocated, the job resource management function requests the parallel execution environment to execute the job. At this time, the job enters the RUNNING state.

   When the prologue and epilogue functions are set, the job is in the RUNNING-P and RUNNING when prologue script completed.

4. Job end

   Once a job receives the end notification from the job resource management function, the job end process starts for the job. When the prologue and epilogue functions are set, the job is in the RUNNING-E, and receives the end notification. The job is in the RUNOUT state while waiting for the end process to complete. When the end process completes, the job enters the EXIT state.

You can perform the following operations for a job.

Deleting a job

When requested by the job operation administrator or an end user to delete a job, the job manager function deletes the job, which then enters the CANCEL state. This is called job deletion. If the job is in the RUNNING state, it goes through the RUNOUT state and is then deleted. The job when deleted enters the CANCEL state. If the job deletion request is for a job in the RUNOUT state, the job when deleted enters the EXIT state.

Holding a job and canceling the hold

Interrupting a job but keeping the state of the submitted job from changing is called holding a job. The job at this time is in the HOLD state.

When the hold on a job is canceled, the job is rescheduled and then executed.

Also, for a job in the ERROR state because it could not be executed normally, canceling this state reschedules the job, which is then executed.

Changing job parameters

You can change the following parameters for a submitted job:

- Elapsed time limit value
- Resource unit on which the job is executed
- Resource group on which the job is executed
- Job priority of the same user
- Priority in resource unit

Recovery from the job error state

After job submission, the job that was temporarily in the ERROR state can be canceled, and queuing can be done again.

Sending a signal to a job

You can send a signal to a running job.

For details on these operations, see "4.2.3 Job Operations" in this manual, or "Checking the Job Status" in "Chapter 2 Job Operation Procedures" in "Job Operation Software End-user's Guide."

2.4.1.2 Job ACL function

Before submitting a job, the end user needs to specify items such as the amount of resources to allocate to the job and the executable times. To prevent these resources from being allocated to individual jobs without limit, the cluster administrator and job operation administrator need to set limits in accordance with the job selection policy of the system they are operating. The job ACL function controls the limits.

The job ACL function manages the upper and lower limit values for specifiable values and the default values used when nothing is specified. Moreover, you can change these limit values in management units of job operations and by group or user.

The job ACL function retains all this aggregated information relating to various limit values. The retained information is called a "job ACL database."

The job ACL function can control the following items:

- Definitions on the simultaneous job acceptance limit and simultaneous job execution limit
- Definitions on the upper limit values, lower limit values, and default values for resources and custom resources at job submission
- Definitions involving job control setting values
- Definitions specifying whether a job operation command is executable
- Definitions specifying permission for the operated objects of a job

The job ACL database is operated and managed by two types of administrator: cluster administrator and job operation administrator.

The following describes the details of management by each administrator.

Cluster administrator

Basically, the job operation administrator who manages resource units configures the operation of the job ACL database. However, the cluster administrator sets the following items:

- Exclusive defined items of a cluster
- Common definition values in a cluster

Information

- Basically, job operations are performed in units of resource units, but some items are managed in units of clusters. Since this setting has an influence on the cluster range, the cluster administrator and not the job operation administrator makes the setting.
- Cluster settings are supposed to be made when the job operation management function is installed or when the resource unit configuration is changed. Users seldom need to make these settings in normal operation.

Job operation administrator

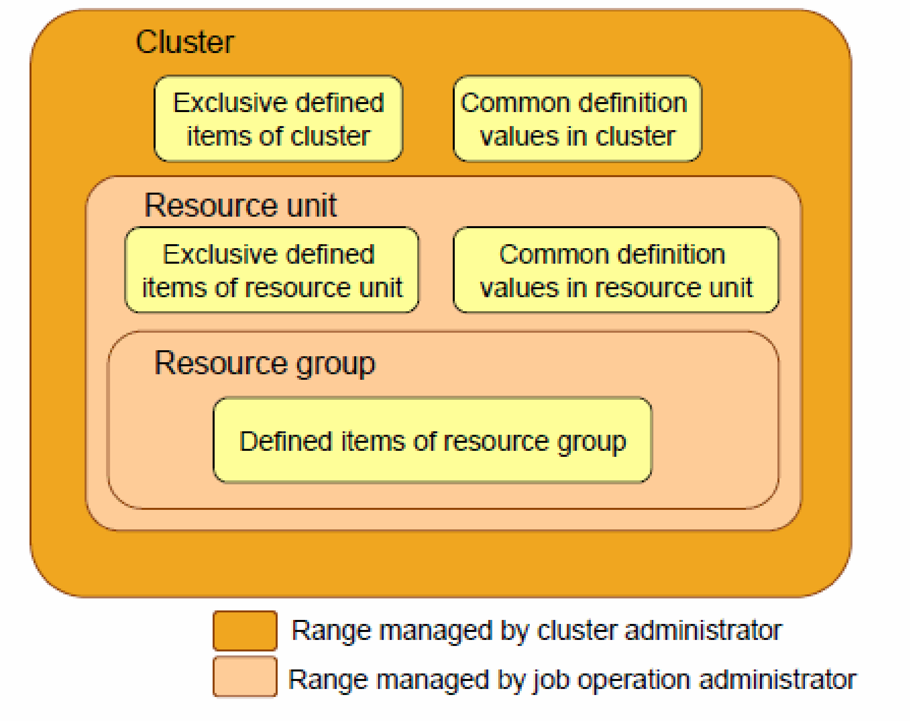
The job operation administrator sets and manages the following defined items for the target resource unit:

- Exclusive defined items of resource units
- Common definition values in resource units
- Defined items for each resource group in the target resource unit

Information

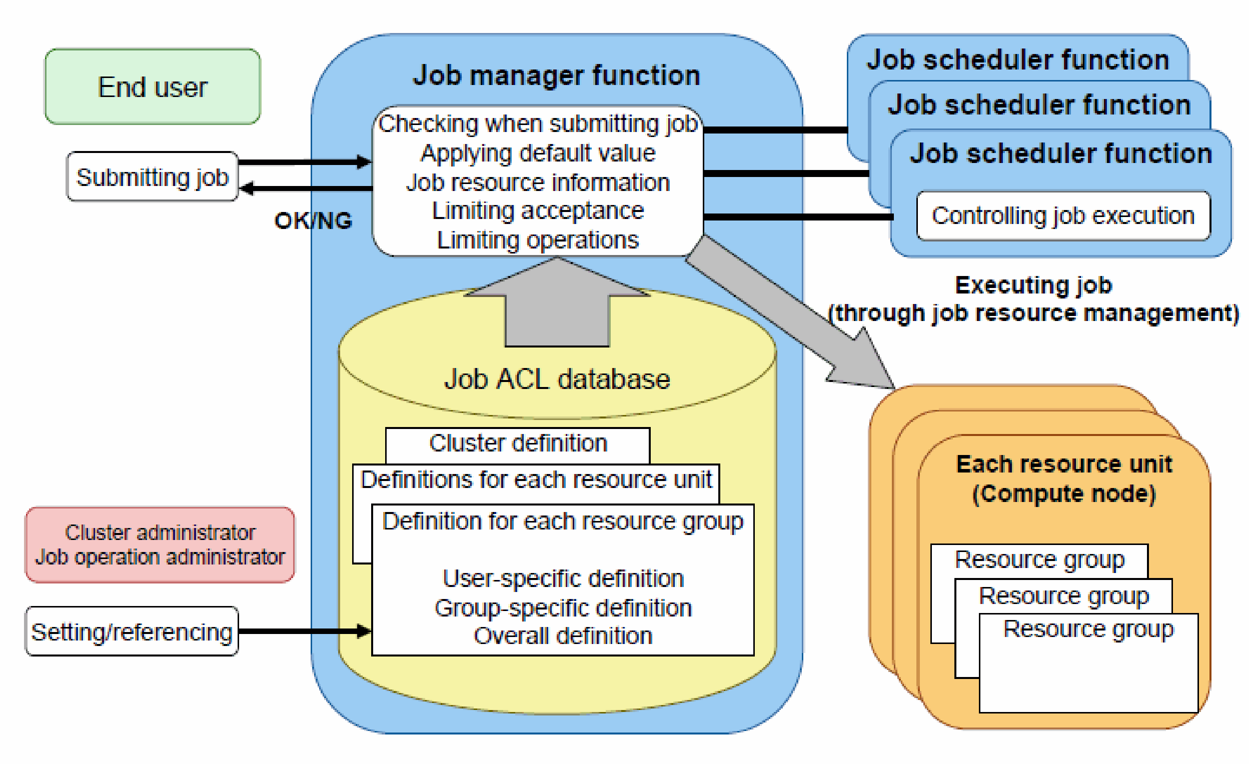
Resource groups have been prepared for the job operation administrator as a means to performing more detailed operations within a resource unit. The job ACL database allows definitions by resource group for some items.

Figure 2.3 Job ACL and administrator privileges


Cluster
Exclusive defined items of cluster
Common definition values in cluster
Resource unit
Exclusive defined items of resource unit
Common definition values in resource unit
Resource group
Defined items of resource group
Range managed by cluster administrator
Range managed by job operation administrator

The following shows the relationship between the job manager function and job ACL database.

Figure 2.4 Job manager function and job ACL database



You can define the above definition units within a cluster, within a resource unit, and within a resource group.

The job ACL function can change settings by user, by group, and by layer in a cluster.

The job ACL function manages information in the following units of definitions.

Table 2.3 Units of definitions of the job ACL function

Definition unit	Meaning
By user	These definitions can store information for individual users, such as setting values and limit values for jobs. Without items defined for individual users, these definitions are valid for application of the same definition to the users in a group or to all the users who submit jobs. You can specify the following definition targets: - Default definitions for users - Default definitions for the users in a group - Specific user definitions - Specific group definitions among specific users
By group	These definitions store information for controlling jobs by group. The definitions include the following definitions, which target all the users in a group: - Default definitions for groups - Specific group definitions
Overall	These definitions store information for controlling jobs as a whole. The items under control as a whole are limited to special definitions such as the limit value for simultaneous acceptance.

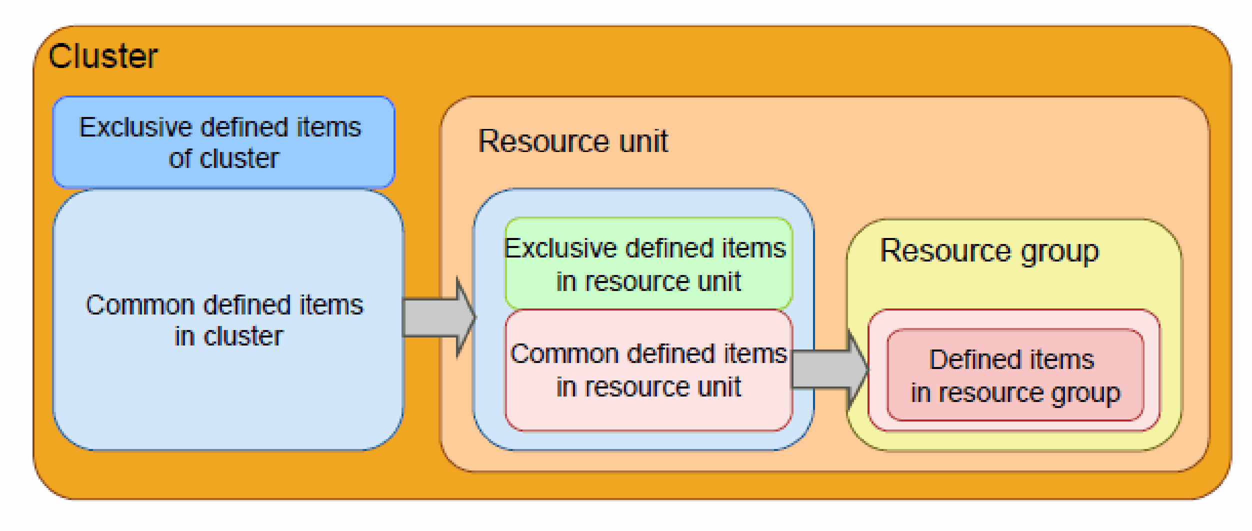
The following table lists the exclusive defined items and common defined items in the ranges of the job ACL function by layer.

Table 2.4 Job ACL function ranges by layer

Application range	Item	Description
Inside a cluster	Exclusive defined items of the cluster	These items can be defined only for the cluster. They cannot be defined for resource units and resource groups. These items do not affect the definitions within resource units, such as the default name for a submitting resource unit.
	Common definition values in the cluster	These defined items can be applied to resource units and resource groups. Basically, they serve as default values to supplement undefined items in the resource units in the cluster. Even in resource units that have the corresponding definitions, these values are applied according to application rules.
Inside a resource unit	Exclusive defined items of the resource unit	These items can be defined only for resource units. They cannot be defined for resource groups. The items include items that do not affect the definitions within resource groups, such as the default name for a submitting resource group. They also include control items for a resource unit range, such as the limit on accepted jobs within a resource unit.
	Common definition values in the resource unit	These defined items can be applied to resource groups. Basically, they serve as default values to supplement undefined items in the resource groups in the resource unit. Even in resource groups that have the corresponding definitions, these values are applied according to application rules.
Inside a resource group	Defined items of the resource group	These defined items can be applied in a resource group range.

The following shows units of definitions and ranges by layer of the job ACL function.

Figure 2.5 Units of definitions and ranges by layer of the job ACL function


Cluster
Exclusive defined items of cluster
Common defined items in cluster
Resource unit
Exclusive defined items in resource unit
Common defined items in resource unit
Resource group
Defined items in resource group

The following describes conceptual images of application within each application range.

Conceptual image of application within a cluster

For each resource unit or resource group that has no definitions, common definition values in the cluster from among the definitions made in the cluster are applied to its individual resource units. Among the definitions applied to resource units, the common definition values in a resource unit are also applied as the definition values of its individual resource groups. So even if there are no cluster definitions, default values will apply.

Figure 2.6 Conceptual image of application within a cluster (before application)

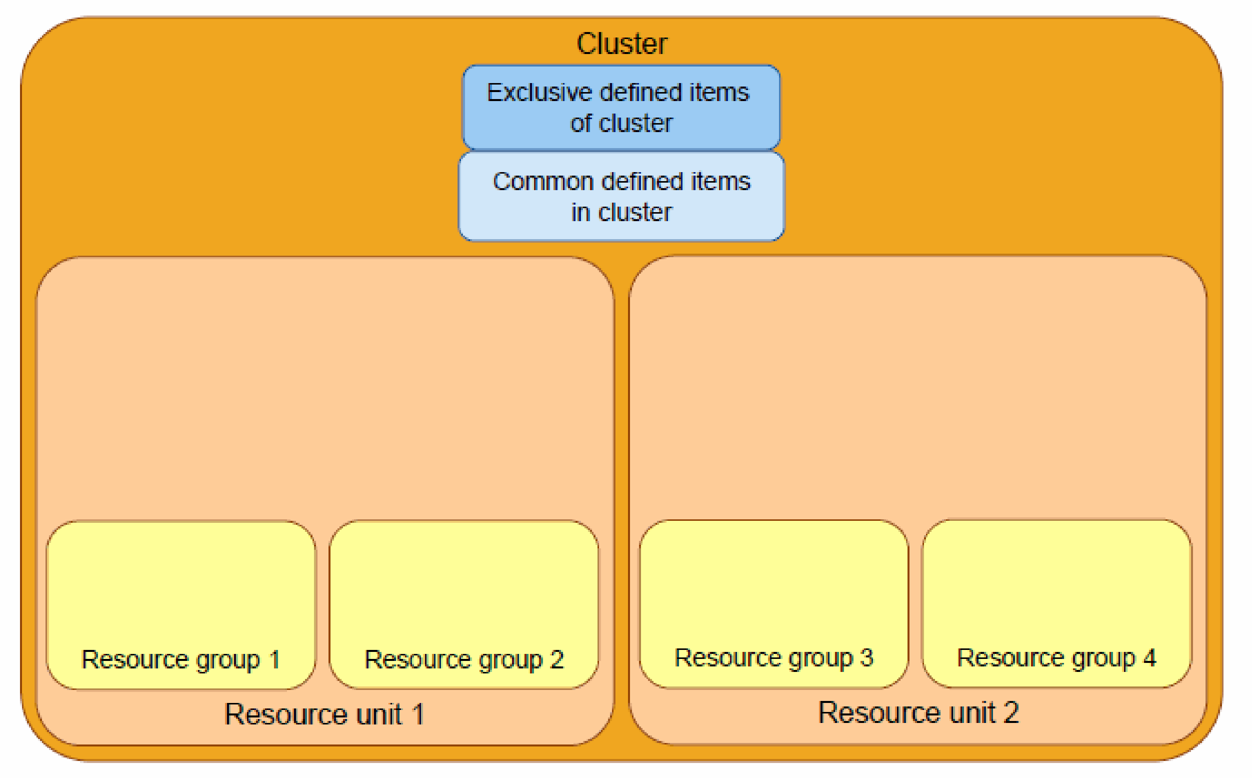
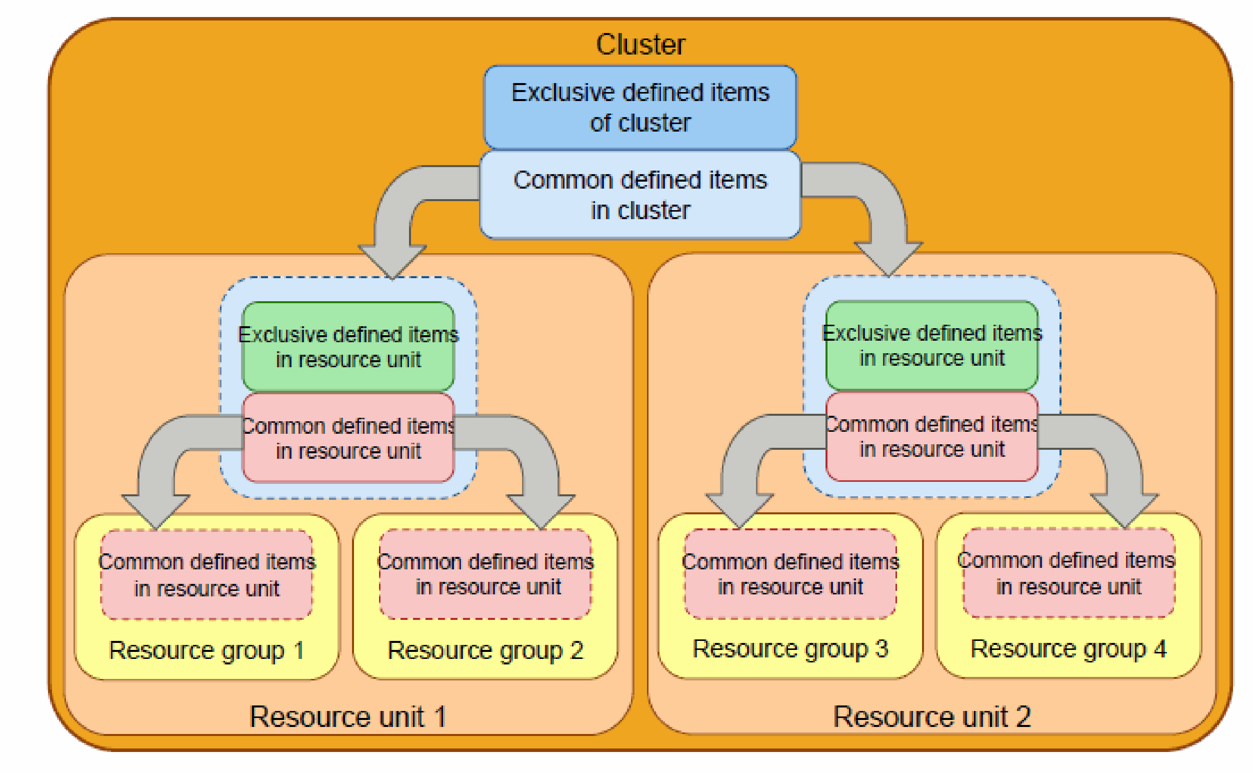


Figure 2.7 Conceptual image of application within a cluster (after application)


Cluster
Exclusive defined items of cluster
Common defined items in cluster
Exclusive defined items in resource unit
Common defined items in resource unit
Exclusive defined items in resource unit
Common defined items in resource unit
Common defined items in resource unit
Resource group 1
Common defined items in resource unit
Resource group 2
Common defined items in resource unit
Resource group 3
Common defined items in resource unit
Resource group 4
Resource unit 1
Resource unit 2

Conceptual image of application within a resource unit

Suppose each resource group has no definitions. If resource unit 2 has definitions, the contents of the resource unit 2 definitions are applied to the resource group3, 4.

Figure 2.8 Conceptual image of application within a resource unit (before application)

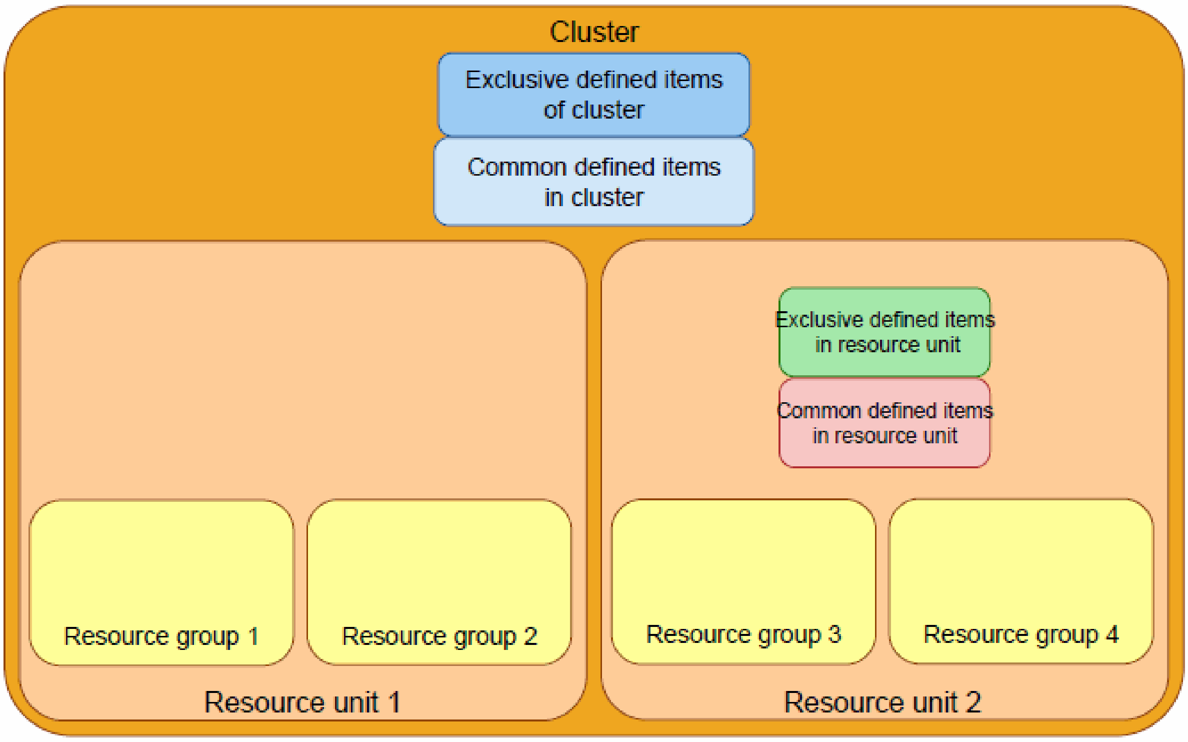
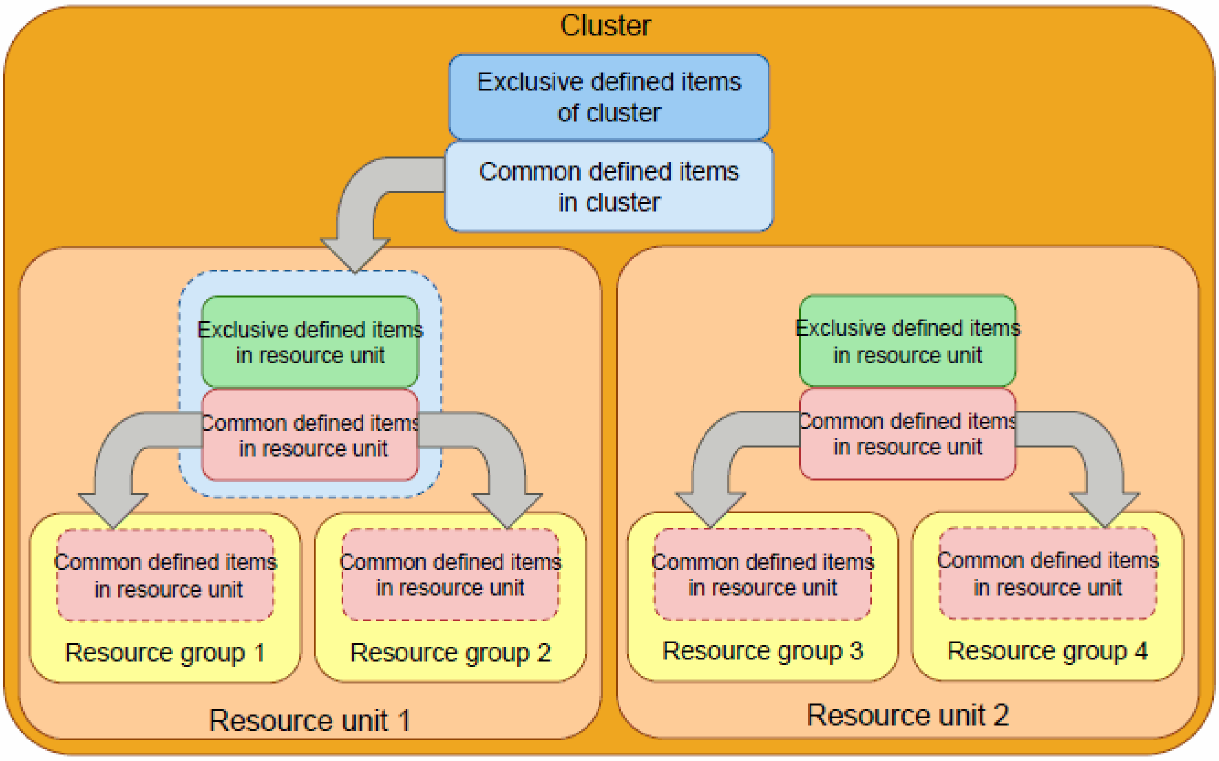

Cluster
Exclusive defined items of cluster
Common defined items in cluster
Exclusive defined items in resource unit
Common defined items in resource unit
Resource group 1
Resource group 2
Resource group 3
Resource group 4
Resource unit 1
Resource unit 2

Figure 2.9 Conceptual image of application within a resource unit (after application)


Cluster
Exclusive defined items of cluster
Common defined items in cluster
Exclusive defined items in resource unit
Common defined items in resource unit
Common defined items in resource unit
Resource group 1
Common defined items in resource unit
Resource group 2
Resource unit 1
Exclusive defined items in resource unit
Common defined items in resource unit
Common defined items in resource unit
Resource group 3
Common defined items in resource unit
Resource group 4
Resource unit 2

Conceptual image of application within a resource group

You can make definitions by resource group. For resource units and resource groups that have no definitions, higher-layer definitions will apply.

Figure 2.10 Conceptual image of application within a resource group (before application)

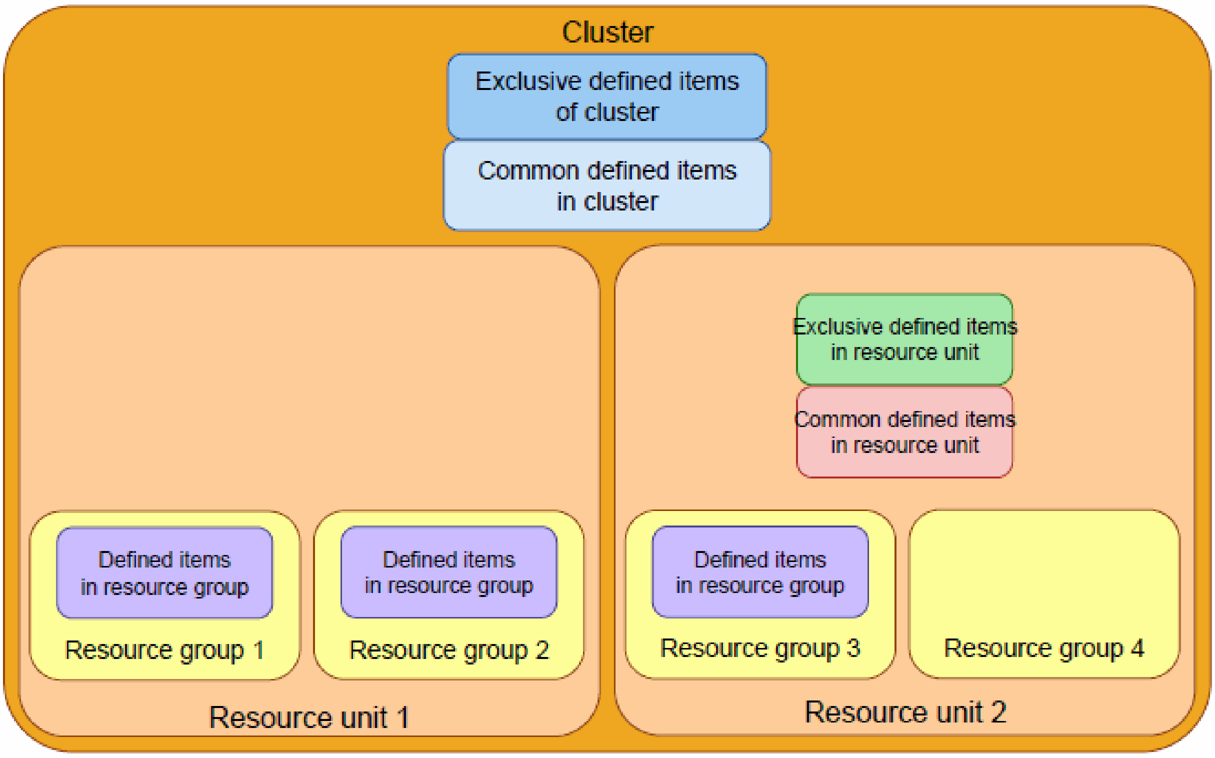
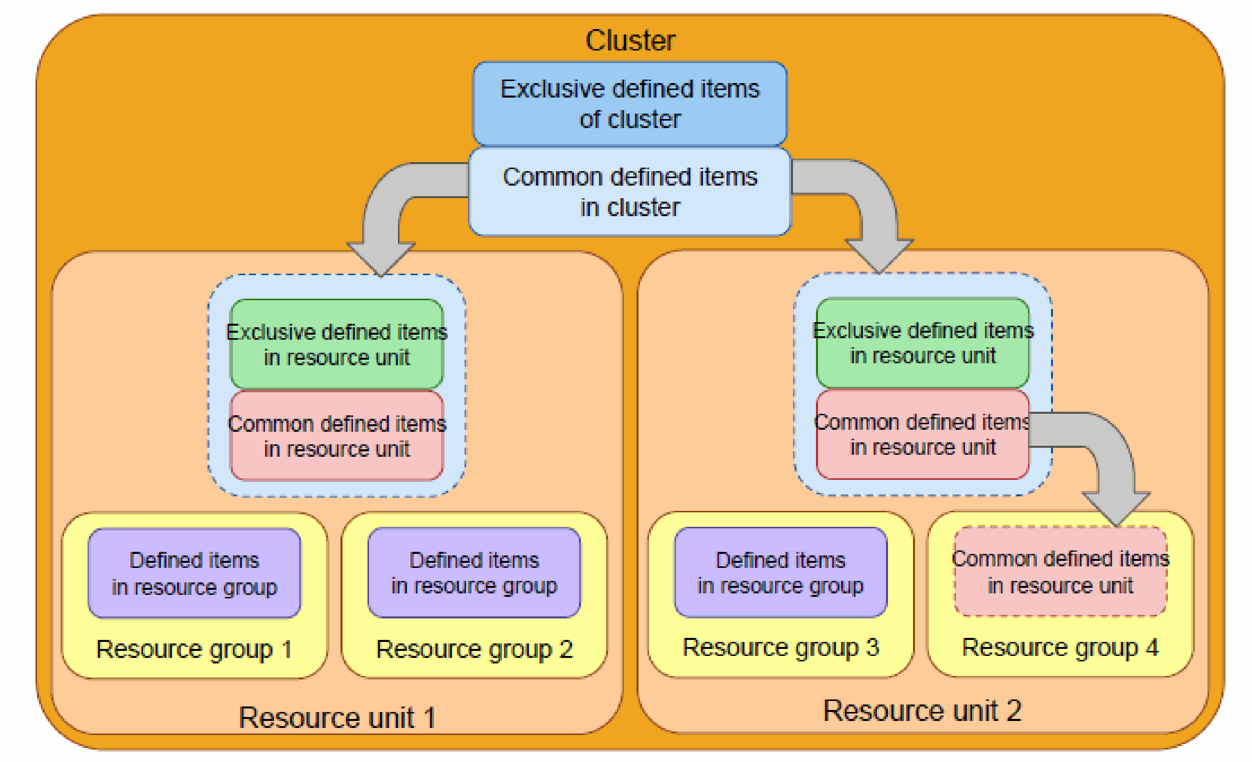

Cluster
Exclusive defined items of cluster
Common defined items in cluster
Defined items in resource group
Resource group 1
Defined items in resource group
Resource group 2
Resource unit 1
Exclusive defined items in resource unit
Common defined items in resource unit
Defined items in resource group
Resource group 3
Resource group 4
Resource unit 2

Figure 2.11 Conceptual image of application within a resource group (after application)


Cluster
Exclusive defined items of cluster
Common defined items in cluster
Exclusive defined items in resource unit
Common defined items in resource unit
Defined items in resource group
Resource group 1
Defined items in resource group
Resource group 2
Resource unit 1
Exclusive defined items in resource unit
Common defined items in resource unit
Defined items in resource group
Resource group 3
Common defined items in resource unit
Resource group 4
Resource unit 2

2.4.2 Job operation support

The job manager function has various auxiliary functions for smooth job operations as described below.

This section describes these functions.

2.4.2.1 Saving ended job script files

The job manager function has a function for saving the job script files of ended jobs for the job operation administrator. For details on the setting for saving ended job script files, see "3.4.1 Settings for job operation management function in a cluster (papjm.conf file)". For the save destination of ended job script files, see "4.1.4 Saving ended job script files ".

2.4.2.2 Prologue and epilogue function

Before and after every job script, you can execute any process specified by the job operation administrator. This is called the prologue and epilogue function.

Use this function to apply settings and processes conforming to the job selection policy to all job scripts. For example, the function is used in the following ways.

- Set system-specific environment variables for use in job scripts.
- Create a work directory at a specific location where a job script can use it, and delete it when the job ends.

See

The prologue and epilogue function is one of the mechanisms called "hooks" in the job operation management function. For details, see "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

2.4.2.3 Job Manager Exit Function

The job manager function has exit functions for calling an arbitrary process, which has been prepared by the administrator, at the specific timing for the execution of each job.

You can use the job manager exit function to, for example, achieve ticket control that allows job submission and execution within the budget allocated to each user who submits a job.

See

The job manager exit function is one of the mechanisms called "hooks" in the job operation management function. For details, see "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

2.4.2.4 Job statistical information function

The job manager function has the job statistical information function, which can record information (job statistical information) such as the amount of resources used by an executed job and various limit values.

The function outputs job statistical information to a job statistical information file. After steps have been taken in job execution, the job operation administrator can analyze the steps by referencing job statistical information from this file.

Information

- Note that, besides the CPU core allocated to jobs, the job information enabled with the FX server has a CPU core called the assistant core. The assistant core is included in the use status calculation. The assistant core handles OS interrupt processing and daemon processing that deteriorate job execution performance. Also, in jobs, MPI asynchronous communication processing is executed on this assistant core. Therefore, the job statistical information of jobs being executed with the FX server includes, besides the resources allocated to jobs, the assistant core use information related to MPI asynchronous communication processing.
- The job statistical information file is backed up by function (logrotate) of OS log rotation. For details, see "3.6 Setting Log Rotation."

To display the contents of the job statistical information file, use the pmdumpjobinfo command on the system management node or compute cluster management node. The pmdumpjobinfo command converts the job statistical information file, which is a binary file, into CSV text format form delimited by comma before displaying the contents.

Note

The pmdumpjobinfo command outputs just character strings and numerical values to facilitate automatic processing of the output. The output is different from the job statistical information displayed by the pjsub command.

The job statistical information file has job statistical information arranged in units called records.

Figure 2.12 Structure of the job statistical information file

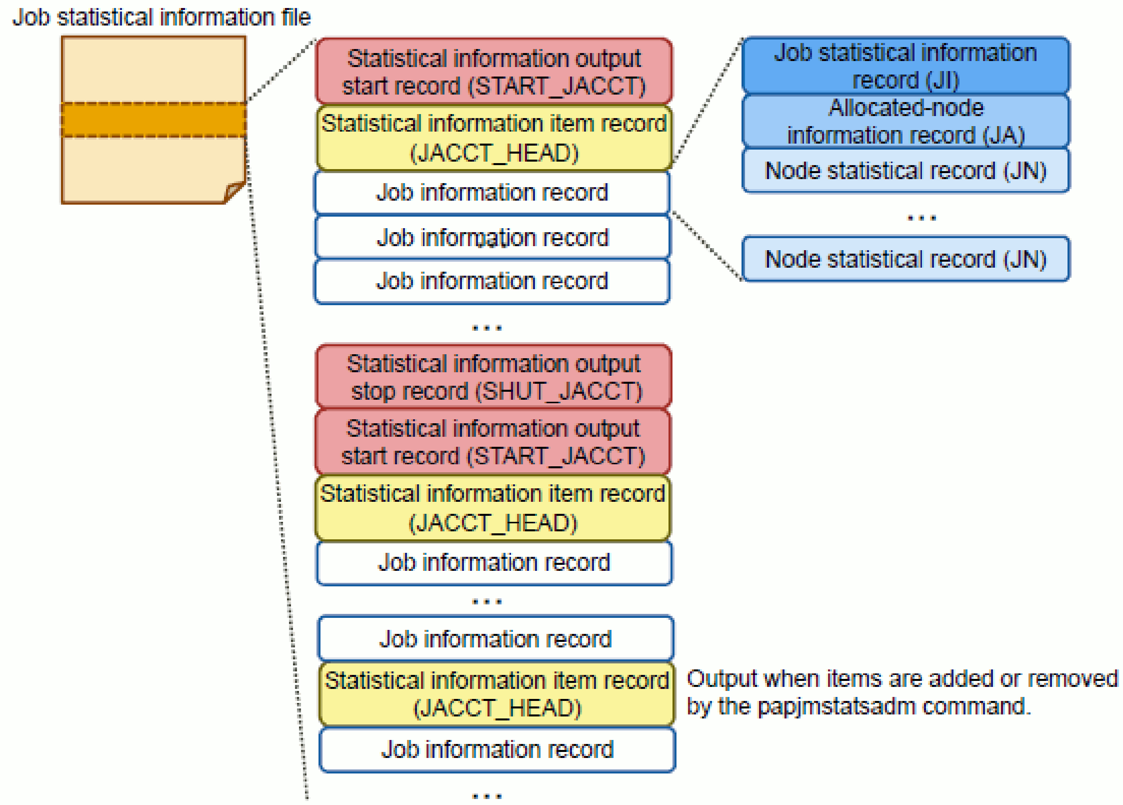

Job statistical information file
Statistical information output start record (START_JACCT)
Statistical information item record (JACCT_HEAD)
Job information record
Job information record
Job information record
...
Statistical information output stop record (SHUT_JACCT)
Statistical information output start record (START_JACCT)
Statistical information item record (JACCT_HEAD)
Job information record
...
Job information record
Statistical information item record (JACCT_HEAD)
Job information record
...
Job statistical information record (JI)
Allocated-node information record (JA)
Node statistical record (JN)
...
Node statistical record (JN)
Output when items are added or removed by the papjmstatsadm command.

Table 2.5 Job statistical information file and records

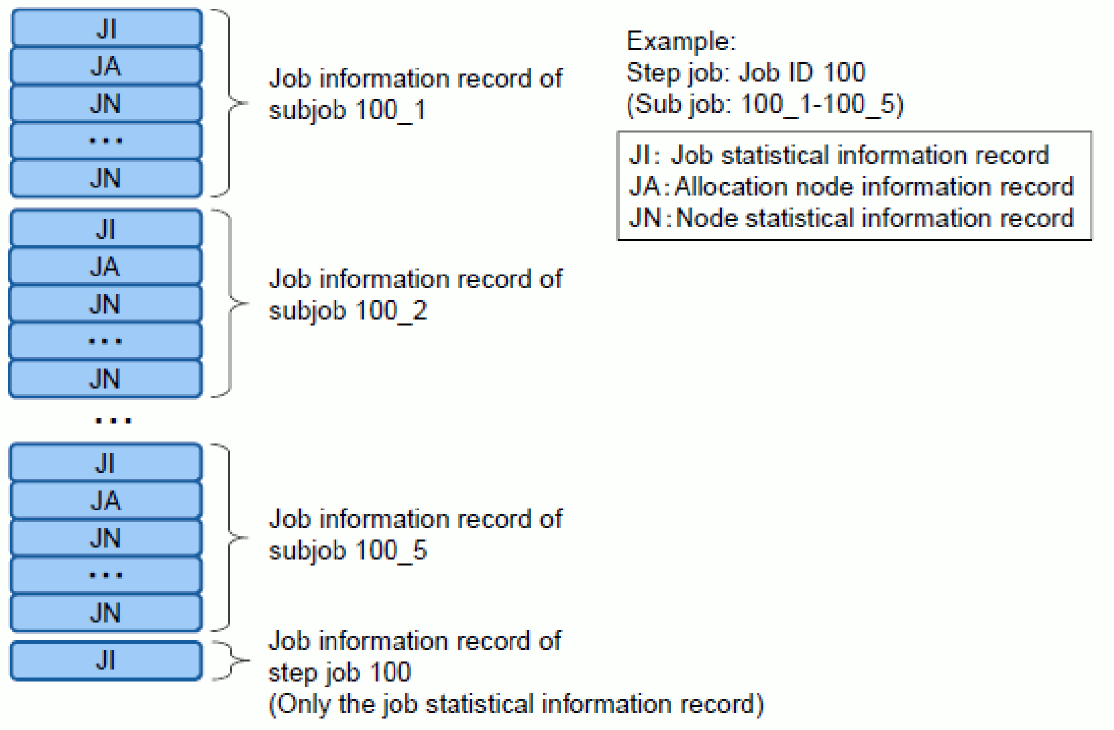
Record name	Description	
Statistical information output start record (START_JACCT)	This record contains time information about when the job statistical information function starts. When the job operation software starts, this record is output. Moreover, when the job statistical information file is switched by the function (logrotate) of OS log rotation, it is output to the head of the file newly switched.	
Statistical information item record (JACCT_HEAD)	This record contains the item names of statistical information. This record follows the statistical information output record. It is also output when the items of statistical information are added or removed by the papjmstatsadm command.	
Job information record	Job information record is a general term for the following records containing statistical information on one job or sub job: Job statistical information record, allocation node information record, and node statistical information record.	
	Job statistical information record (JI)	This record contains statistical information about one job or sub job.
	Allocation node information record (JA)	This record contains statistical information about each node allocated to job. Only one of this record is output after the job statistical information record.
	Node statistical information record (JN)	This record contains statistical information about each node allocated to the job. If virtual nodes are allocated to the job, the record contains information about each virtual node. Only the records for the number of nodes (for node allocated jobs on the FX servers nodes and PRIMERGY servers) or virtual nodes (for virtual node allocated jobs on the PRIMERGY

Record name	Description	
		servers) are output after the allocation node information record.
Statistical information output stop record (SHUT_JACCT)	This record contains time information about when the job statistical information function stops. When the job operation software stops, this record is output. Moreover, when the job statistical information file is switched by the function (logrotate) of OS log rotation, it is output to the tail of the old file.	

In a bulk job or step job, a job information record is output for each sub job assigned a sub job ID. When all the sub jobs end, only the job statistical information record assigned the job ID is output as a job summary.

The pmdumpjobinfo command outputs the above records (statistical information output start record, statistical information item record, job statistical information record, allocation node information record, node statistical information record, and statistical information output stop record), with one record per line.

Figure 2.13 Job information record of a bulk job or a step job


JI
JA
JN
• • •
JN
Job information record of subjob 100_1
JI
JA
JN
• • •
JN
Job information record of subjob 100_2
• • •
JI
JA
JN
• • •
JN
Job information record of subjob 100_5
JI
Job information record of step job 100 (Only the job statistical information record)
Example:
Step job: Job ID 100
(Sub job: 100_1-100_5)
JI: Job statistical information record
JA: Allocation node information record
JN: Node statistical information record

A job information record is output when a job or sub job ends or is deleted. Specifically, the record is output when the job or sub job state changes as shown below.

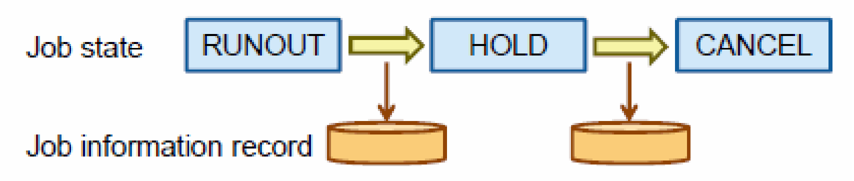
Table 2.6 Job information record output conditions

	Before the state changes	After the state changes
Pattern A	RUNNING-A RUNNING RUNOUT	CANCEL EXIT HOLD ERROR QUEUED
Pattern B	QUEUED HOLD ERROR	CANCEL

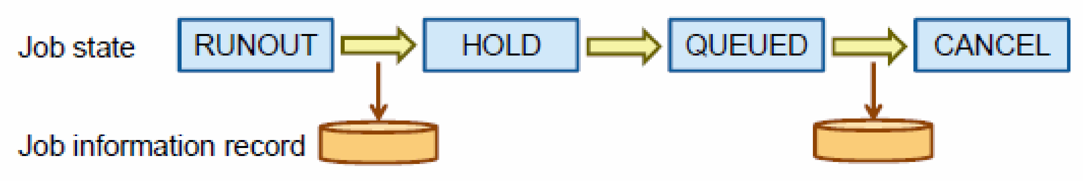
There are pattern A and pattern B at the output opportunity. In the state transition of pattern B, the job is not executed, so that the value of a part of job statistical information in the job information record is not set (see "Appendix A Invalid Values for Job Statistical Information at State Transition"). Therefore, please total the job information record output at the opportunity of pattern A when you total results of the job execution by the pmdumpjobinfo command. You can confirm if the transition corresponds to pattern A by confirming the value output to "State of job" and "Prior state of job" in the job information record.

Note

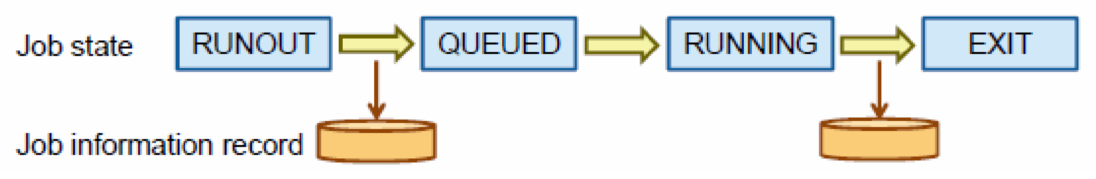
- Accordingly, multiple job information records may be output for one job ID. The following cases are examples.
  - The running job is held by the pjhold command and then deleted by the pjdel command.



  - The running job is held by the pjhold command. After the hold is released by the pjrls command, the pjdel command deletes the job while it is in the QUEUED state.



  - The system error (the compute node down etc.) occurred while executing the job and the job is re-executed.



- Job statistical information is not output for jobs rejected at acceptance.

The administrator can set the job statistical information function as follows.

Enable/disable output of job statistical information

The cluster administrator can set whether to output job statistical information. For details on the setting method, see "3.4.1 Settings for job operation management function in a cluster (papjm.conf file)."

Add a new job statistical information item

The cluster administrator can add an original item for output in job statistical information. For details on how to add the item, see "3.4.2.1 Settings of administrator-defined items in job statistical information."

Specify the items to output from job statistical information

The cluster administrator can specify the items to output from job statistical information. Output items can be specified for each of the following output destinations:

- Job statistical information file (file specified in the pmdumpjobinfo command)
- .stats file (file output when the job was submitted with the -s or -S option specified in the pjsub command)
- Output for display when the -s or -S option in the pjstat command is specified

The administrator can also specify a path name for the job statistical information file.

For details on the setting method, see "3.4.2.2 Definitions of output items in job statistical information."

Set a value for an added job statistical information item

From the job manager exit function or a resource manager exit script, the job operation administrator can set a value for a job statistical information item added by the cluster administrator. For details, see "[Settings for Job Statistical Information]" in "Creating an Exit Function Source File" and "Creating an Exit Script" in "Chapter 2 Creating and Incorporating Hooks" in "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

2.4.3 High availability of job operations

The job manager function has the high availability function, which allows job operation to continue even if an error occurs on the active or standby node of the redundantly configured compute cluster management node or on the compute node running the job.

Among the redundantly configured compute cluster management nodes, if an error occurs on the active compute cluster management node, the standby compute cluster management node is automatically switched with the active node. Even though the active compute cluster management node is switched with the standby node, the jobs running on compute nodes continue to run without interruption. If an error occurs on the compute node running a job, the running job will be automatically re-executed provided that re-execution is specified.

Operation when a Tofu interconnect link is down [FX]

If a Tofu interconnect link goes down due to a hardware failure or other cause during job execution on an FX server, communication between job processes is affected. However, this may be avoided by dynamically changing the communication path.

When submitting a job, the end-user can specify (--net-route option of the pjsub command) whether to change the communication path at the time that a Tofu interconnect link is down.

For this specification, the administrator can set the following with the job ACL function:

- Default setting for the job submission time (define net-route)
- Permission to specify the --net-route option (execute pjsub-net-route)

These settings are valid only for node-exclusive jobs on an FX server and MPI jobs (only for MPI processing systems using Development Studio).

When the link goes down during job execution and the Tofu interconnect communication path is changed, communication is retransmitted. Then, a message is output to the /var/log/messages file on the compute node where the communication was retransmitted.

See

- For details on job ACL function settings related to the --net-route option of the pjsub command, see "3.4.4 Job ACL function settings in a cluster." For details on the --net-route option of the pjsub command, see "Specifying the operation when a Tofu interconnect link is down" in "Chapter 2 Job Operation Procedures" in the "Job Operation Software End-user's Guide."
- For details on the message, see "Tofu Library (TOF) Messages" in "Appendix B System Log Messages" in the "Job Operation Software Troubleshooting."

Note

- Enabling the setting that dynamically changes a Tofu interconnect communication path may cause job execution performance to deteriorate. Be careful in environments for executing jobs where execution performance is critical.
- Depending on the location of the link that went down, there may be no alternative communication path, in which case the job is aborted.
- Enabling the setting that dynamically changes a Tofu interconnect communication path extends the range of nodes that can be used as a communication path. Therefore, even when deadline scheduling is set so that a node should not be the communication path of a job (padeadline --ic), the node may fall in the range of nodes for a running job. So the node might be used as a communication path of the job. In that case, the padeadline command causes an error, and the deadline schedule cannot be set.

- In cases where a Tofu interconnect link goes down, the communication path is changed, and the job continues, the -v option of the pjshowclst command displays the following for the node where the link went down:

  - REASON field: PortRouterFatal

  - ARCH_STATUS field: ICC_PreDisable

  For details on what the pashowclst command displays, see "Displaying Operation Status of the System" in "Chapter 3 Details of the System Management Function" in the "Job Operation Software Administrator's Guide for System Management."
  New jobs are not allocated to this node until link down recovery.

2.5 Job Scheduler Function

The job scheduler function has the following functions:

- Job resource selection function

  The function selects the resources required for a job from computer resources.

- Job execution selection function

  The function determines the execution order of multiple jobs.

- Deadline scheduling function

  The function schedules job execution to prevent jobs from being executed within the scope and period of stoppage.

- Job rescheduling function

  The function determines when the job runs.

- Job scheduling function using custom resources
  The function defines and manages custom resources, selects the required custom resources for jobs, and determines the execution order.

- Job scheduler exit function

  The function calls an arbitrary process prepared by the administrator at a specific time related to the scheduling of each job.

2.5.1 Job resource selection function

The job resource selection function selects a node according to the requirements of a job and allocates it to the job.
The concept varies depending on the unit of allocation (node or virtual node) of node resources.

2.5.1.1 Allocation in units of nodes

For allocation of node resources in units of nodes, specify a shape for FX servers, or specify a number of nodes for PRIMERGY servers.

The concept of the shape of FX servers is described below.

Nodes are assumed located in a virtual space, and end users can specify the number and shape (one-dimensional to three-dimensional shapes) of nodes to allocate. The job scheduler function selects nodes according to those specifications.

Regardless of their shape, if nodes cannot be allocated at the present time, they will be allocated when the amount of free space required by the job can be reserved.

The job administrator sets the maximum node dimensions that allow nodes to be allocated to jobs. The shape of nodes allocated to a job must fit within the maximum node dimensions.

Suppose that the maximum values of the Tofu coordinates (which are along the X, Y, and Z axes) in the system are Lx, Ly, and Lz in the case of nodes allocated in torus mode or mesh mode (described below). In this case, the maximum dimensions (Xmax, Ymax, and Zmax) for nodes that can be allocated to a job are calculated as follows.

Table 2.7 Maximum dimensions for nodes

Number of dimensions	Maximum node dimensions
One-dimensional	Xmax = Lx * Ly * Lz * 12

Number of dimensions	Maximum node dimensions
Two-dimensional	Xmax = Lx * 6, Ymax = Ly * Lz * 2
Three-dimensional	Xmax = Lx * 2, Ymax = Ly * 3, Zmax = Lz * 2

If a specified two-dimensional or three dimensional shape does not fit within the maximum dimensions as is, it may fit when rotated as an adjustment. You can specify such a shape. In that case, the job scheduler function automatically adjusts it in that way.

For example, suppose the maximum node dimensions are 2 x 3 x 32. You can specify a node shape of 6 x 3 x 2 even though it does not fit as specified. This is because the node shape fits within the maximum dimensions of 2 x 3 x 32 when the job scheduler function automatically rotates it to 2 x 3 x 6. In contrast, a node shape of 4 x 4 x 4 does not fit within the maximum dimensions of 2 x 3 x 32 no matter how it is rotated. For this shape, the job would not be submitted.

For non-contiguously allocated jobs, nodes can be allocated to the jobs regardless of the node shape of the jobs if the number of free nodes fulfills the requests from the jobs.

Node placement methods

When allocating FX servers, you can give an instruction for a node placement method on Tofu coordinates.

- Torus mode
  The minimum node allocation unit is one Tofu (12 nodes).
  Nodes are placed so that they are mutually adjacent in the Tofu coordinate space.

- Mesh mode
  The minimum node allocation unit is one node.
  Nodes are placed so that they are mutually adjacent in the Tofu coordinate space.

- Non-contiguous mode
  The minimum node allocation unit is one node.
  Nodes are allocated so that they are mutually adjacent in the Tofu coordinate space as far as possible. However, this may not always be possible.

Note

- Even in torus mode, if the job uses only the single node of a one-dimensional shape, resources are allocated in units of nodes. However, if the specified shape is two dimensional (1 x 1) or three dimensional (1 x 1 x 1), the unit for node allocation is rounded up to the Tofu unit.

- In torus mode, multiple jobs, including multi-node jobs, can share the nodes contained within the same Tofu unit under the following conditions. First, the jobs must be in a one-dimensional shape. Also, the number of nodes requested by the jobs is in a range from 1 to 11. To enable this function, use resource group settings. For details on the setting method, see the "ResourceGroupTsha" definition item in "3.5.1.2 Resource group settings."

- In the following cases, job execution may fail due to the node resources being judged insufficient:

  - In torus mode or mesh mode, the specified shape fits within the maximum shape and the nodes are specified within the range of the number of nodes that can be used. However, since one or more nodes are faulty, a torus cannot be formed.

  - In non-contiguous mode, inter-node all-to-all communication is not guaranteed.

- Suppose that both a job in torus mode or mesh mode and a job in non-contiguous mode are mixed and executed in a resource unit. In this situation, the job in non-contiguous mode may deteriorate the communication performance of the job in torus mode or mesh mode.

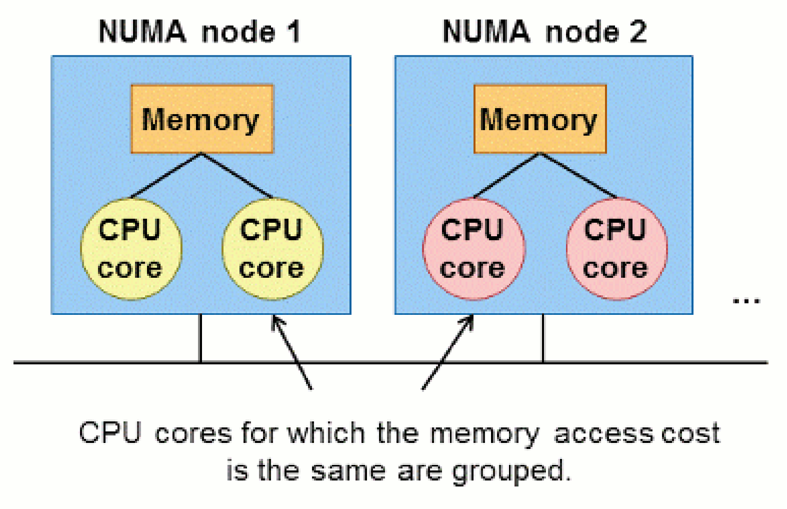
2.5.1.2 Allocation in units of virtual nodes

Job resources are handled in the unit of a set of a CPU core and memory on a virtual node. End users can specify the number of virtual nodes to allocate to their jobs, the number of CPU cores and amount of memory per virtual node, and the amount of memory per CPU core. For details, see "Node Resource Allocation" in "Chapter 1 Job Mechanism" in the "Job Operation Software End-user's Guide."

2.5.1.3 NUMA allocation policy

For NUMA (Non Uniformed Memory Access) architecture compute nodes, the cost of accessing memory shared among multiple CPU cores is not uniform. Depending on the CPU core allocated to the job, this may cause variation in or deterioration of the execution performance. Therefore, the CPU cores for which the cost of accessing memory is the same are logically paired with memory and divided into groups. These groups are called "NUMA nodes."

Figure 2.14 NUMA nodes



On virtual nodes, the job operation management function can be set so that a job is allocated so that it can fit within the NUMA node or so that a job spans NUMA nodes (NUMA allocation policy). For details on the NUMA allocation priority setting method, see "3.4.4.2 Defined items of the job ACL function."

2.5.2 Job execution selection function

Numerous jobs are submitted to the system. The job execution selection function efficiently allocates resources to these jobs and determines the order in which to execute these jobs. The concept of determining the job execution order is called "job selection policy."

2.5.2.1 Job selection policy

The job selection policy is an evaluation criterion used to determine job execution priority. It can be defined by combining multiple evaluation items. Jobs are executed in descending order of execution priority when the backfill function, which is described in a subsequent section, is disabled.
Job operation administrators apply the job selection policy defined by the cluster administrator to the resource units that they manage. They can change those definitions for each resource unit and resource group as required.

The following table lists evaluation items.

Table 2.8 Evaluation items for determining the job execution order

Item name	Description
fcfs	A job that is submitted earlier has a higher execution priority. This is called the fcfs method (First-Come First-Serve).
node (*)	The evaluation is based on size, that is, the number of nodes used. It can be specified only for the FX server.
group_prio (*)	The evaluation is based on the priority of the group to which the user (job submitter) belongs.
user_prio (*)	The evaluation is based on the user (job submitter) priority.
usr_in_grp_prio (*)	The evaluation is based on the user priority within the group.
job_prio (*)	The evaluation is based on the job priority.
job_aprio (*)	The evaluation is based on the job priority within the resource unit.

Item name	Description
rscgrp_prio (*)	The evaluation is based on the resource group priority.
group_fairshare (*)	The evaluation is based on the group's fair share value. The fair share value is an evaluation value used by the fair share function to ensure fair use of computer resources. For details, see "2.5.2.2 Fair share function."
user_fairshare (*)	The evaluation is based on the user's fair share value.
usr_in_grp_fairshare (*)	The evaluation is based on the user's fair share value within the group.
elapse_limit (*)	The evaluation is based on the limit value on the elapsed execution time for the job.
node_times_elapse (*)	The evaluation is based on the product of the number of nodes (node) multiplied by the limit value on the elapsed execution time for the job (elapse_limit). It can be specified only for the FX server.
at	The evaluation is based on whether the execution start time is specified.
interact	The evaluation is based on whether the job is an interactive job.
job_epoint (*)	The evaluation is based on job evaluation points calculated from the length of the job execution wait time. The method of calculating the job evaluation points is defined by the administrator.

Information

For the marked (*) items in the above table, you can specify whether to evaluate the value in ascending or descending order in size.

The cluster administrator or job operation administrator can define the job selection policy by choosing multiple items to be used for evaluation. The administrator can specify an evaluation order for the chosen items. Until the job execution priority is determined, lower-order items are evaluated.

The settings by the cluster administrator are written in the papjm.conf file. The settings by the job operation administrator are written in the pmpjm.conf file. For details on how to code the papjm.conf file, see "3.4.1.3 Job selection policy settings." The method of coding the pmpjm.conf file is the same as for the papjm.conf file. For details on how to reflect the papjm.conf file contents, see "3.4.1.5 Reflecting and referencing the papjm.conf file." For details on how to reflect the pmpjm.conf file contents, see "3.5.1.9 Reflecting and referencing the pmpjm.conf file."

The execution priority of jobs in a resource unit is determined as follows based on the job selection policies defined the resource unit and the resource group.

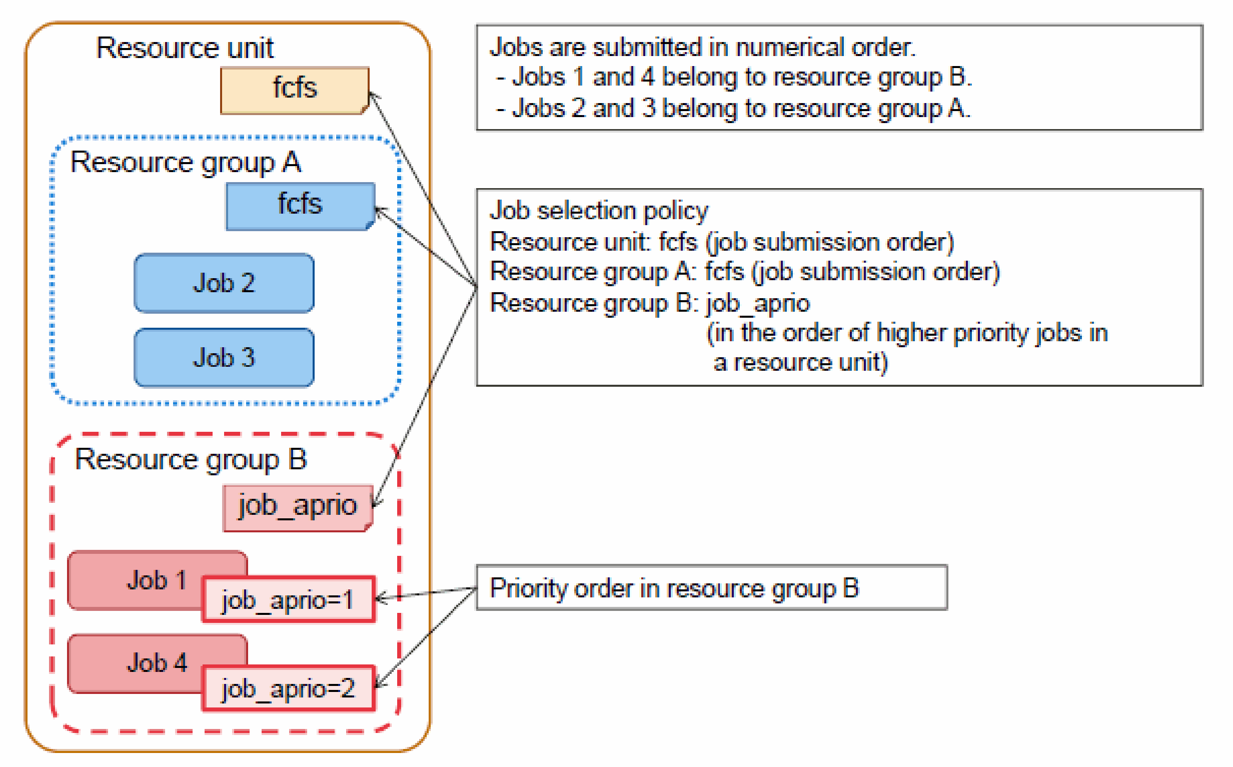
1. For each resource group, the job representing each resource group is determined as the highest priority job based on the job selection policy on each resource group.
2. The highest priority job in the resource unit is determined by the job selection policy on the resource unit, from among the jobs representing each resource group.
3. For the resource group that has the highest priority job in the resource unit, the second highest priority job in the resource group is chosen by the job selection policy on the resource group, and becomes next representative job in the resource group.
4. Then, steps 2 and 3 are repeated to determine the job execution priority in the resource unit.

Note

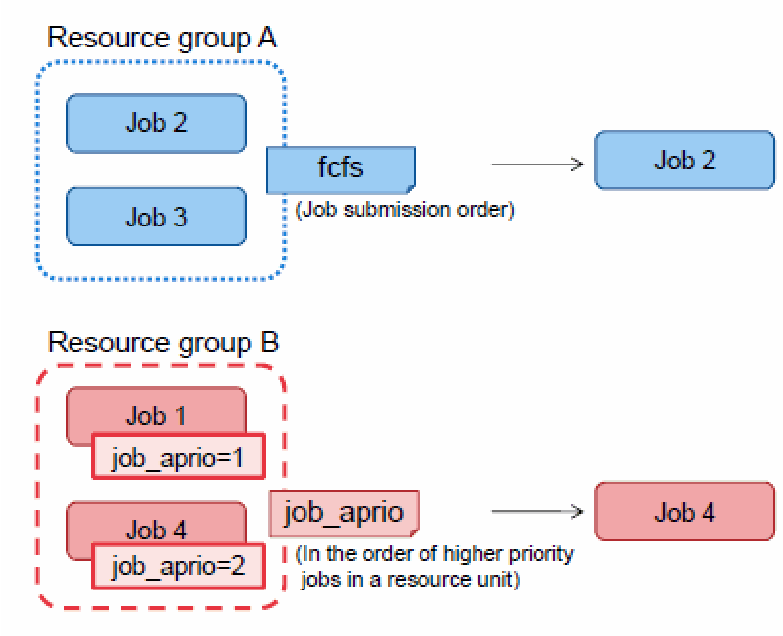
If no job selection policy is set for a resource group of the job, the resource unit policy is applied.

The steps in determining the execution priority of jobs 1 to 4 submitted to resource groups A and B are described below.

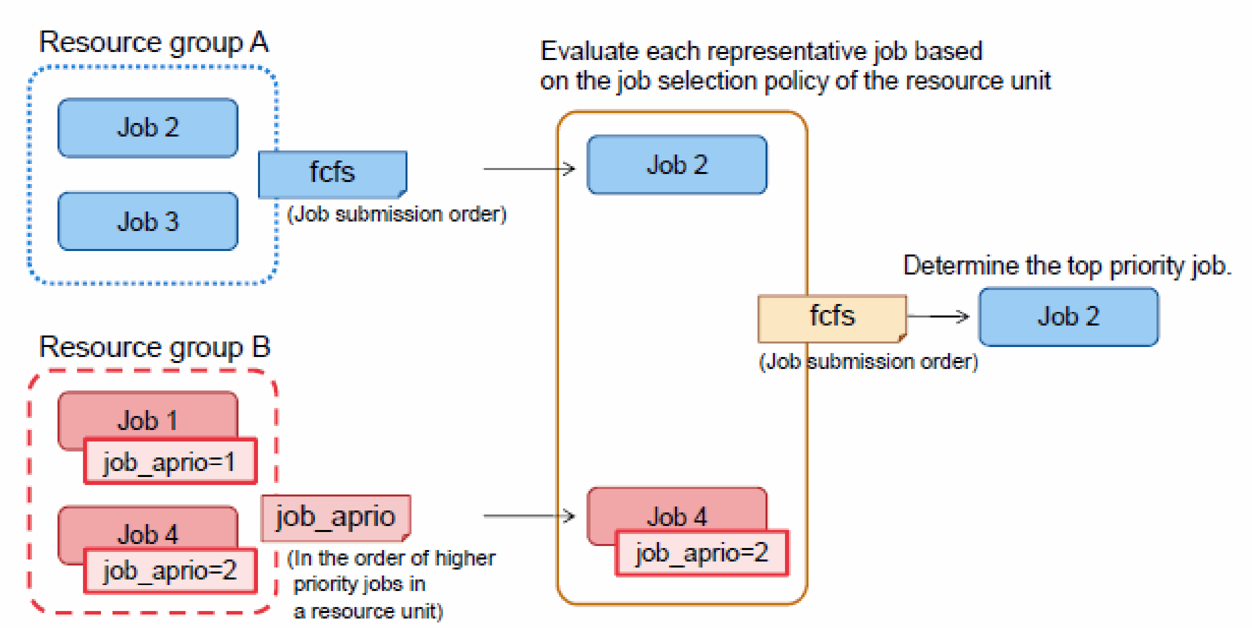
Figure 2.15 Steps in determining the execution priority of jobs submitted to resource groups


Resource unit
fcfs
Resource group A
fcfs
Job 2
Job 3
Resource group B
job_aprio
Job 1
job_aprio=1
Job 4
job_aprio=2
Jobs are submitted in numerical order.
- Jobs 1 and 4 belong to resource group B.
- Jobs 2 and 3 belong to resource group A.
Job selection policy
Resource unit: fcfs (job submission order)
Resource group A: fcfs (job submission order)
Resource group B: job_aprio
(in the order of higher priority jobs in a resource unit)
Priority order in resource group B

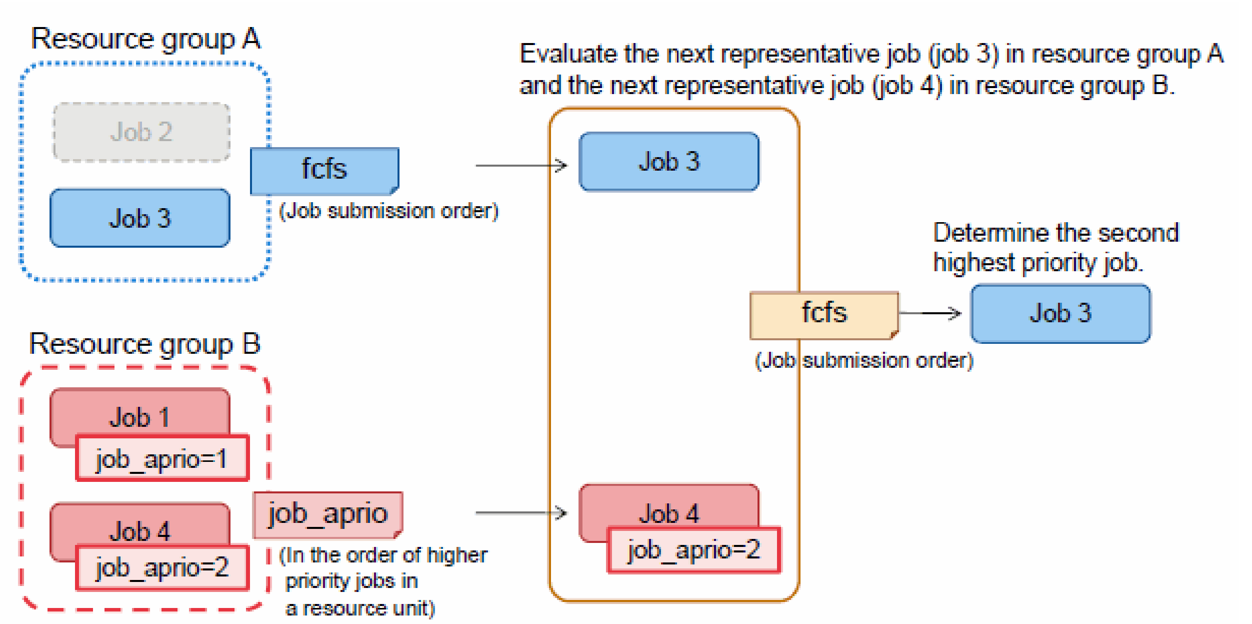
1. Select the representative jobs (of job 2 and job 4) based on the job selection policy of each resource group.



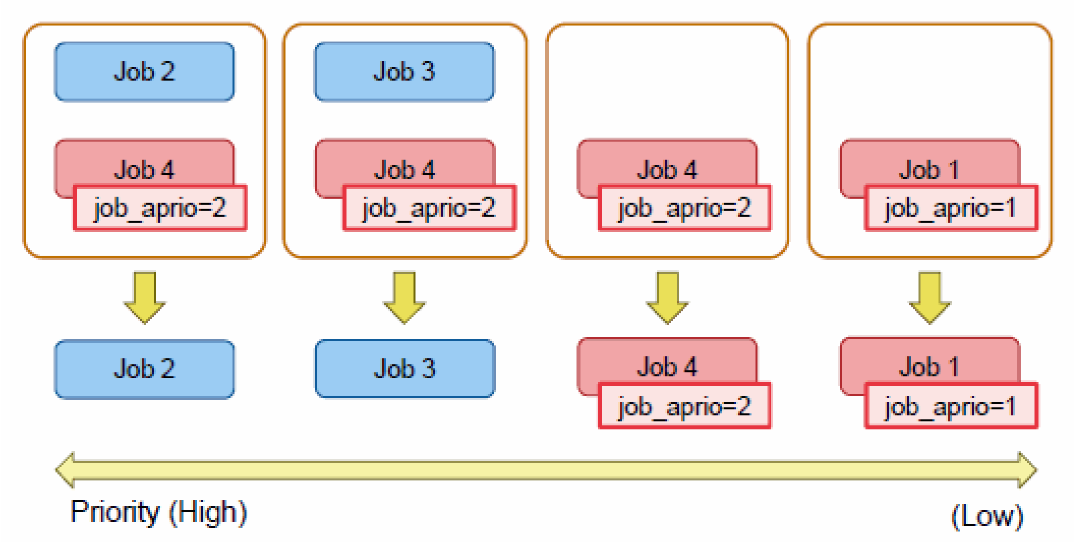
2. Based on the job selection policy (fcfs) of the resource unit, compare the representative jobs of the respective resource groups. Job 2 was submitted earlier, so select it.


Resource group A
Job 2
Job 3
fcfs
(Job submission order)
Evaluate each representative job based on the job selection policy of the resource unit
Job 2
Determine the top priority job.
fcfs
(Job submission order)
Job 2
Resource group B
Job 1
job_aprio=1
Job 4
job_aprio=2
job_aprio
(In the order of higher priority jobs in a resource unit)
Job 4
job_aprio=2

3. Since job 2 has been selected from resource group A, the next representative job becomes job 3. Evaluate job 3 and job 4 as the representative jobs of the respective resource groups. Job 3 was submitted earlier, so select it.


Resource group A
Job 2
Job 3
fcfs
(Job submission order)
Evaluate the next representative job (job 3) in resource group A and the next representative job (job 4) in resource group B.
Job 3
Determine the second highest priority job.
fcfs
(Job submission order)
Job 3
Resource group B
Job 1
job_aprio=1
Job 4
job_aprio=2
job_aprio
(In the order of higher priority jobs in a resource unit)
Job 4
job_aprio=2

4. In this way, jobs are evaluated and the execution priority is finally determined, as shown below.

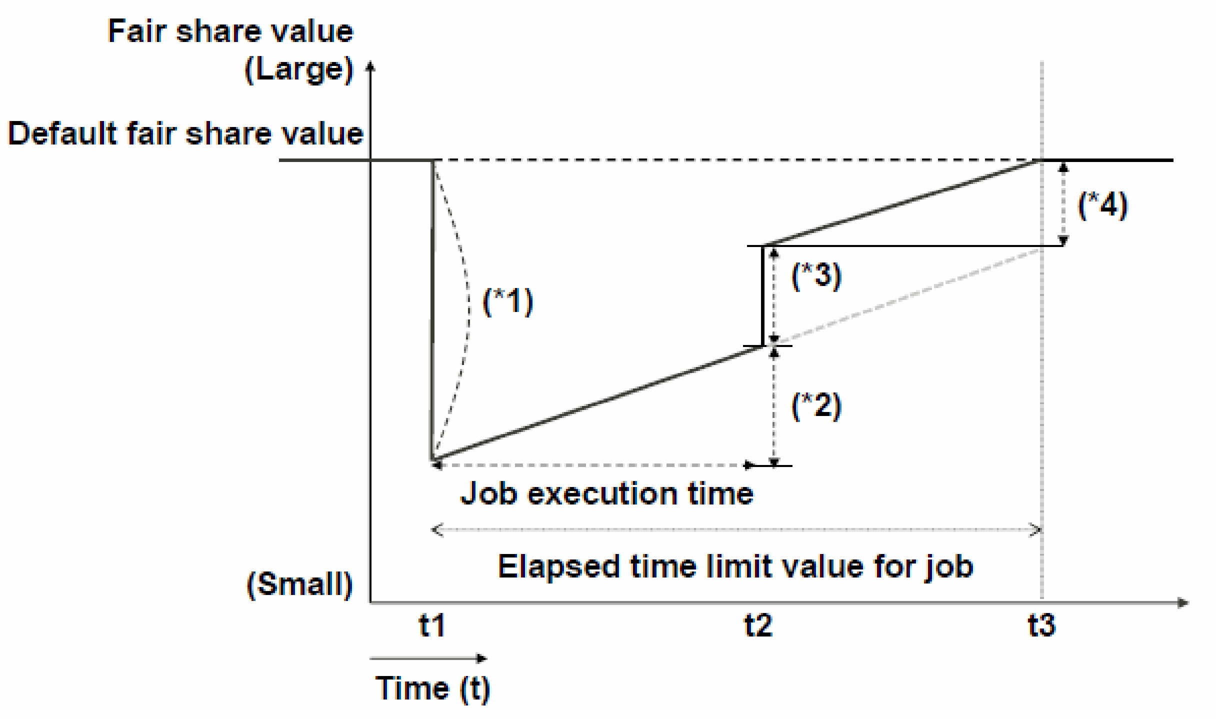


2.5.2.2 Fair share function

To prevent the jobs from being executed only by specific groups and users, the fair share function determines the execution priority of jobs based on past resource use results (nodes for FX server, CPU cores for PRIMERGY server) by group, user and user in group.
In the fair share function, the job execution priority is determined by the fair share value. The fair share value reduces when the job is executed, and it recovers as time passes.

The following figure shows how the fair share value changes from "job execution starts" to "after the job ends."

Figure 2.16 Change of fair share value after end from job execution start



When job execution starts

The fair share value is reduced according to the elapsed time limit value for the job. This reduction at the job execution start time is called fair share usage.
The start of job execution reduces the fair share value by the fair share usage (*1). The fair share usage is determined as follows.

```
Fair share usage (*1) = amount of resource for job x elapsed time limit value for job (seconds)
```

For an FX server, the amount of resources for a job may be either the number of nodes requested by the job or the requested number of CPU cores. As for the PRIMERGY server, it is all the numbers of CPU cores allocated to the job.

Information

- By default, for an FX server, the number of nodes is used to calculate the amount of resources for a job. For details on changing this calculation to use the number of CPU cores instead, see "3.4.5 Settings for advanced job scheduling."
- For the FX server, though the number of allocated nodes is rounded up to the nearest Tofu unit, the fair share usage is calculated based on the requested number of nodes.

During job execution

The reduced fair share value recovers over time to an upper limit, which is the initial fair share value.
The fair share value that was reduced at job execution start recovers with the passage of time during job execution.

```
Amount recovered by job end (*2) = fair share recovery rate x job execution time (t2 - t1) (seconds)
```

The fair share recovery value is the product of the base for recovery of a fair share value and recovery factor.

```
Fair share value recovery rate = fair share recovery value x recovery factor
```

Information

If you want to change the fair share recovery rate for each user, group, or user in a group, adjust it with the recovery factor. The fair share recovery value is constant within a resource unit.

When the job ends

When the job ends before reaching the elapsed time limit value, the fair share value recovers by an amount equal to the time not used by job (elapsed time limit value - actual execution time) (*3).

```
Amount recovered immediately after job end (*3) = amount of resource for job x (t3 - t2) (seconds)
```

Information

This fair share value recovery can be disabled. For details on the setting method, see "3.4.5 Settings for advanced job scheduling."

Note

If the elapsed time limit value of a running job decreases, the fair share value recovers by the amount of the reduction.

```
Recovery amount = amount of job resources x (elapsed time limit value before change - elapsed time
limit value after change)
```

In contrast, if the elapsed time limit value of a running job increases, the fair share value is reduced by the amount of the increase.

```
Reduction amount = amount of job resources x (elapsed time limit value after change - elapsed time
limit value before change)
```

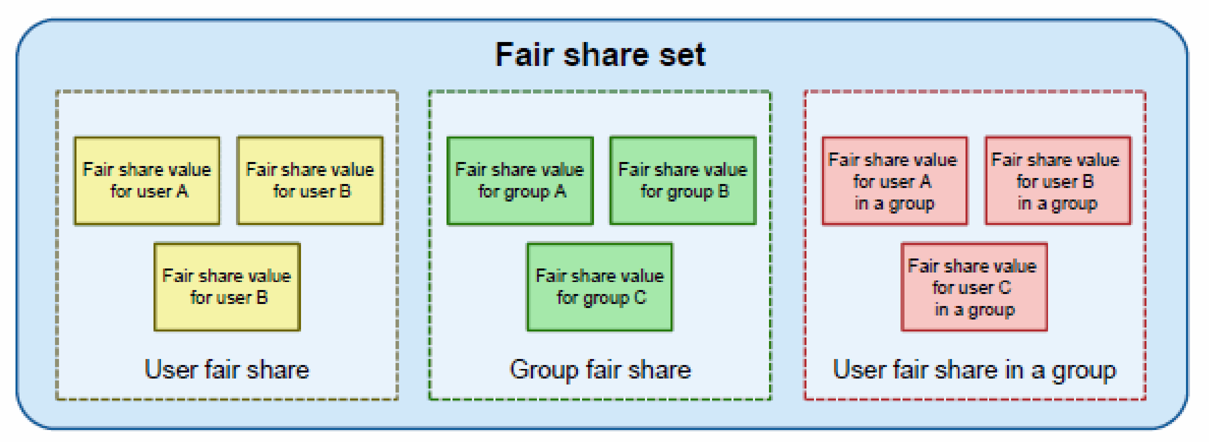
After the job ends

Even after the job ends, the fair share value continues to recover until it reaches the initial fair share value (*4). In this case too, the recovery rate is the fair share recovery value x recovery factor.

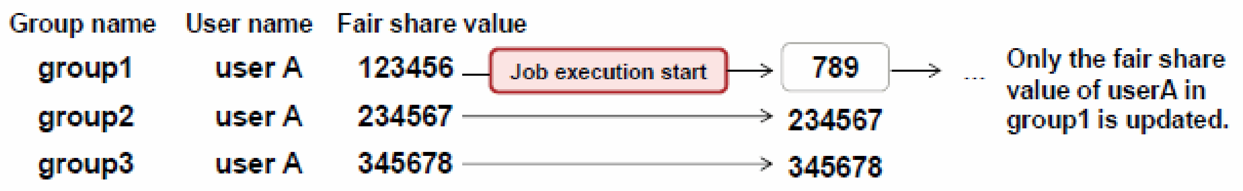
Information

The fair share function has fair share values for each group, user, and user in a group. A set of these values is called a fair share set. By defining a fair share set for every resource unit and resource group, you are determining the job execution priority for every resource unit and resource group. Also, by defining a fair share set shared by multiple resource groups, you are determining the job execution priority in these groups.

Figure 2.17 Fair share set

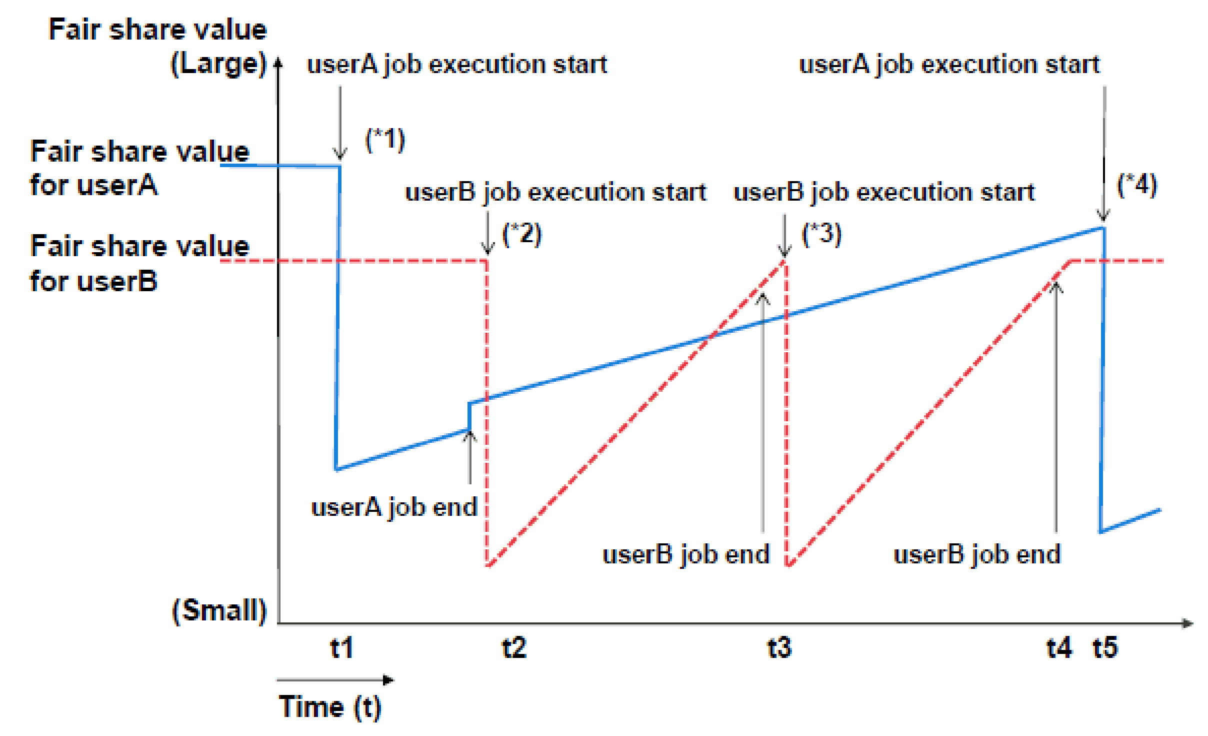

Fair share set
Fair share value for user A
Fair share value for user B
Fair share value for user B
User fair share
Fair share value for group A
Fair share value for group B
Fair share value for group C
Group fair share
Fair share value for user A in a group
Fair share value for user B in a group
Fair share value for user C in a group
User fair share in a group

The difference between the fair share value of a user and the fair share value of a user in a group is as follows.
The fair share value determined for each user is a unique value for the user within a single fair share set. The fair share value is updated each time that a job is executed in any group to which the user belongs.
In contrast, each user has as many fair share values of the user in a group as the number of groups to which the user belongs in the fair share set. The fair share value of the user is updated only when a job is executed by a corresponding group or a user belonging to the group. At this time in the corresponding groups, the fair share values of this user in the other groups to which the user belongs are not updated.
In the following example, the user userA has the respective fair share values for the groups group1, group2, and group3 in a resource unit, and the userA job is executed by group1.



The appearance from which the job is executed without biasing to one user by using the fair share value when two or more users exist is shown as follows.

Figure 2.18 Job execution priority control by difference of users' fair share values



See

To use the fair share function, configure the papjm.conf or pmpjm.conf file to subtract the fair share value when job execution starts and to use the fair share value as a job selection policy. For details on the setting method, see "3.4.1.4 Settings for the fair share function ."

For details on the Initial fair share value settings and the recovery factor settings, see "3.4.4.2 Defined items of the job ACL function. For details on the fair share recovery value settings, see "3.4.1.2 Default value settings for resource units."

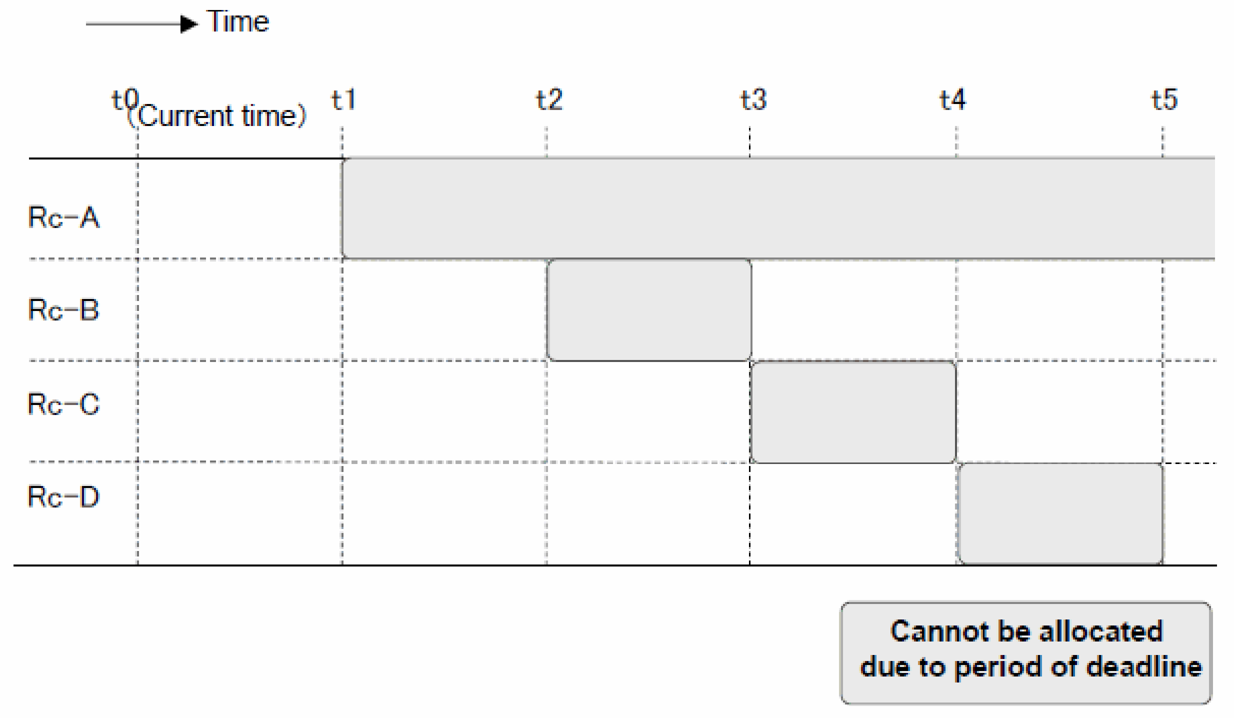
You can also change the fair share value and the initial fair share value with the pmpjmopt command. For details on how to change the value, see "4.2.5 Monitoring and changing a fair share value and initial fair share value ."

2.5.3 Deadline scheduling function

If you want to stop operation in all or part of the system for planned maintenance or a power outage, you must not do so until every running job has been aborted or terminated. The deadline scheduling function schedules job execution to prevent jobs from being executed within the scope and period of stoppage.

The following example shows the deadline scheduling function settings for system maintenance.

Figure 2.19 Example of deadline scheduling function settings (when the maintenance target does not have a faulty node)

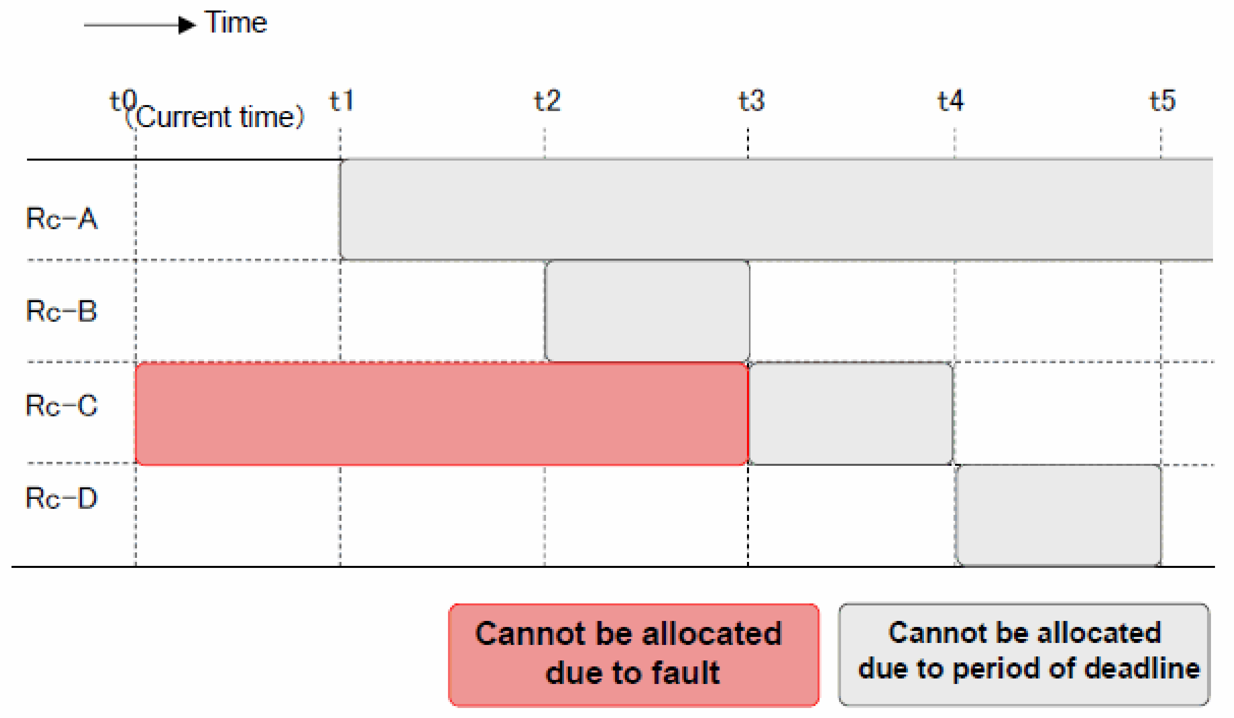


In the above example, all the normal resources Rc-A, Rc-B, Rc-C, and Rc-D are available for the period from the current time, t0, to the deadline t1.

Resource Rc-A is excluded from the allocation targets for an indefinite time after deadline t1, because the end of the allocation suppression period is not specified.

In contrast, resources Rc-B, Rc-C, and Rc-D are excluded from the allocation targets for limited periods. They can be allocated after their respective periods end.

Figure 2.20 Example of deadline scheduling function settings (when the maintenance target has a faulty node)


Time
t0(Current time)
t1
t2
t3
t4
t5
Rc-A
Rc-B
Rc-C
Rc-D
Cannot be allocated due to fault
Cannot be allocated due to period of deadline

In the above example, Rc-C is faulty. The normal resources Rc-A, Rc-B, and Rc-D are available for the same period of time as in the example where the maintenance target does not have a faulty node (Figure 2.19).

Since Rc-C is faulty, it cannot be allocated after the current time, t0. However, Rc-C is expected to recover from the fault through maintenance work during the deadline period, so it can be allocated for a job after t4, the end time of the deadline period. For example, suppose that job scheduling is done at the time t1. Rc-C is considered as a resource that cannot be allocated during the period of time from t1 to t4 but can be allocated after t4, the end time of the deadline period. If no deadline is set for the faulty Rc-C, the resource is considered as not allocatable in any period of time.

For details on the deadline scheduling function, see "4.1.2 Cluster deadline scheduling management".

2.5.4 Job scheduling function

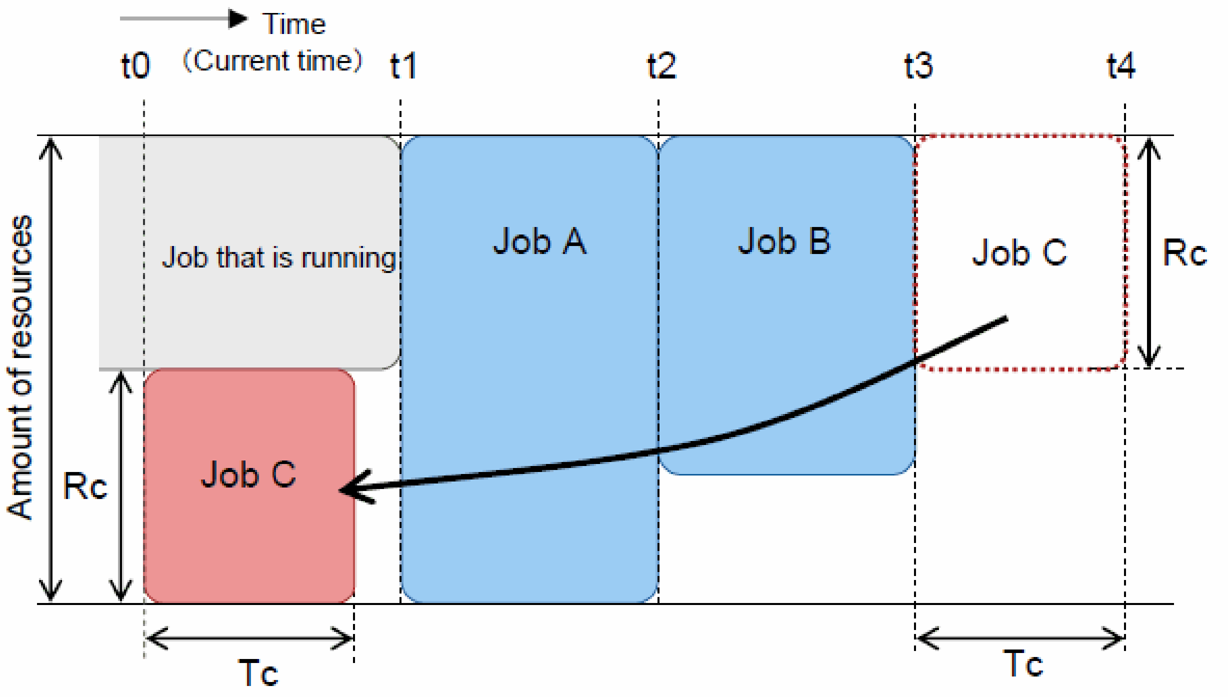
2.5.4.1 Backfill function

As a rule, jobs are allocated free resources and executed in descending order of execution priority. Suppose that job execution priorities are strictly observed to determine the job execution order and that a job requesting a lot of resources is waiting to be executed. Even though resources are free, subsequent jobs cannot be executed, so this may be a waste of resources.

If a job with a lower execution priority can be executed with the available resources, it is executed before a job with a higher execution priority. This function is called the backfill function.
The administrator can set to enable or disable the backfill function for each resource group or resource unit. Set it with the Backfill setting item of the papjm.conf or pmpjm.conf file. For details of settings, see "3.4.1.2 Default value settings for resource units" and "3.5.1.2 Resource group settings".

The following example determines the order of execution priority of jobs by using the backfill function.

Figure 2.21 Moving a job past other jobs in the execution order by using the backfill function



- At the current time t0 in the above figure, one job is running, and three jobs are waiting to be executed. The three jobs are A, B, and C, which were submitted in this order.

- With the fcfs method used to determine the execution order, jobs A, B, and C will be executed at times t1, t2, and t3, respectively. In this case, resource amount Rc is available for period t1, which are what is needed to execute job C, between the current time t0 and time t1, but job C is not executed until time t3.

- If the backfill function is enabled in these circumstances, it schedules job C to overtake jobs A and B, which have higher execution priority, and be executed first.
  However, overtaking in scheduling is done within a range that does not interfere with the start of a higher priority job.

The job scheduler function manages the allocation schedule of the resource for the future from the current time, and judges whether backfill is possible based on the allocation schedule of the resource.

See

Because the occurrence of job overtaking with the backfill function is also related to the scheduling period described in "2.5.4.2 Job scheduling parameters", see also "Scheduling period (SchedulePeriod, DynamicSchedulePeriod)".

2.5.4.2 Job scheduling parameters

The job scheduler function manages plans involving changes in the amounts of available resources from the present time to the future. The allocation schedule of the resource at a certain time is called "Resource map", and the resource map to two or more time of the future is created. The job scheduler function schedules jobs waiting to be executed (in a node resource allocation plan) based on these resource maps. When the rescheduling occurs due to an event (ex. job termination and job submitting), the resource map during the future is recreated.

The cluster administrator and job operation administrator can adjust the following parameter for efficient job scheduling.

See

For details on how to set scheduling time-related parameters, see "3.4.1 Settings for job operation management function in a cluster (papjm.conf file)" and "3.4.5 Settings for advanced job scheduling".

Scheduling period (SchedulePeriod, DynamicSchedulePeriod)

Scheduling period is a term for the job scheduling to the future.
If the period is too long, the resource maps to be created increase. Therefore, the load of the job scheduler function goes up, and the scheduling processing may be delayed.
If a job cannot start during the schedule period, the scheduling for the job is suspended. The scheduling for other job whose priority is lower than the job is also suspended.

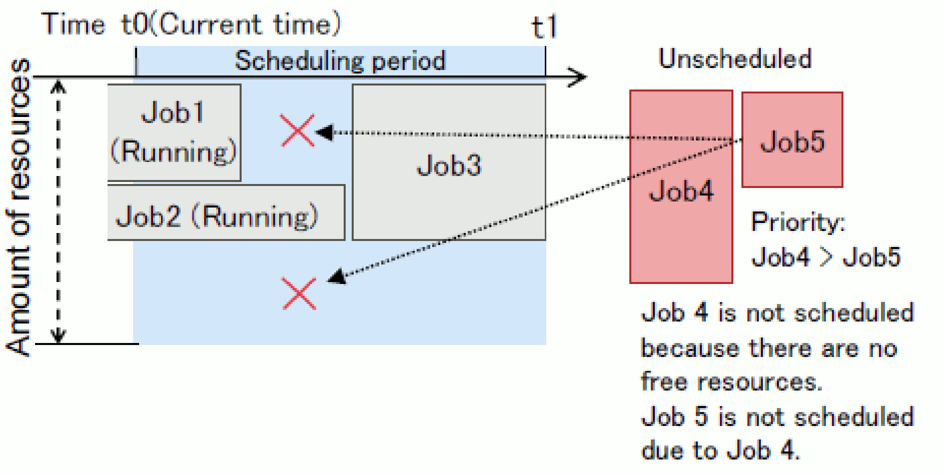
Information

The SchedulePeriod parameter represents a fixed-length scheduling period. The DynamicSchedulePeriod parameter is a scheduling period variable that depends on the maximum elapsed time limit value of the submitted job.

The following describes the relationship between the scheduling period and the backfill function when overtaking an unscheduled job.

In the following example, the scheduling for Job4 is suspended because there are no free resources in the scheduling period. If Job5, which has a lower priority than Job4, is submitted, and the backfill function is disabled, then Job5 is also suspended due to the impact of Job4.

Figure 2.22 Example of suspending the scheduling (When the backfill function is disabled)



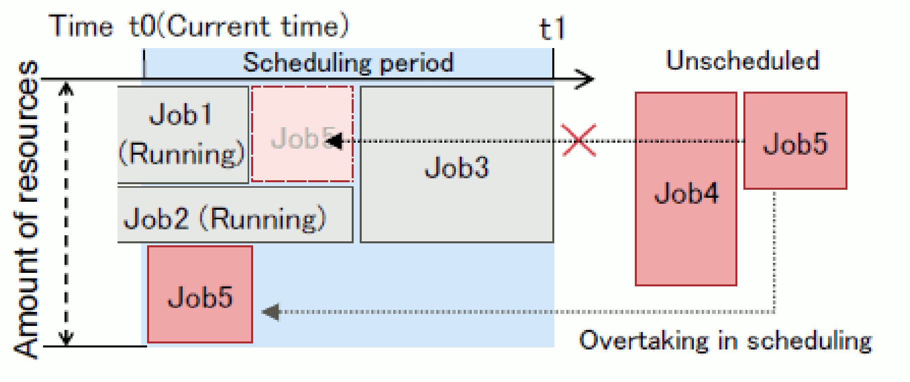
When there are jobs pending scheduling, the backfill function allows a low-priority job to be scheduled first if the following conditions are met:

a. There are free resources available at the current time (Time the schedule started) for the job being scheduled.

b. The expected end time of the job to be scheduled does not exceed the latest expected end time of the running job in all resource groups in the resource unit and does not exceed the scheduling period.

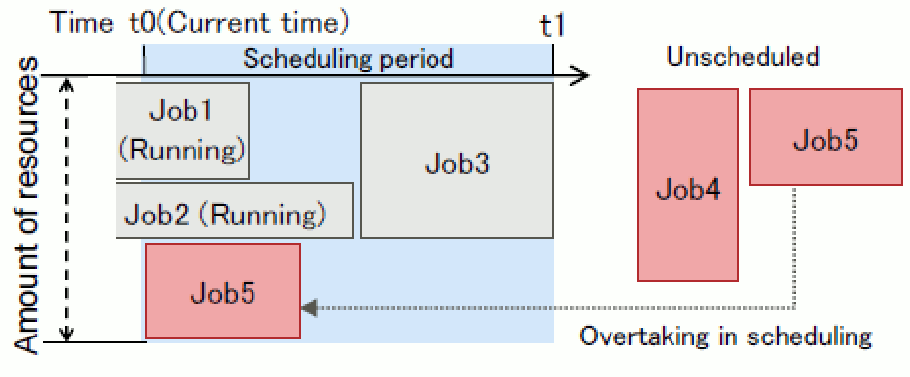
If the backfill function is enabled in above figure, Job5 is scheduled to overtake Job4 and run from time t0 as shown below (Condition 'a'). The backfill function does not schedule Job5 to run after Job1, rather than at the current time t0.

Figure 2.23 Example of overtaking an unscheduled job (1)



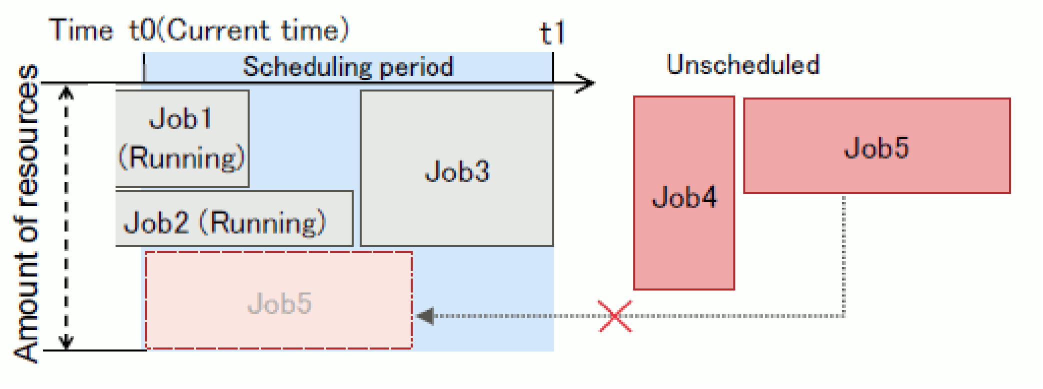
In the following example, if Job5 is scheduled to run from time t0, its expected end time exceeds the expected end time of the running Job1, but it does not exceed the latest expected end time of Job2(Condition 'b'), allowing Job5 to overtake Job4.

Figure 2.24 Example of overtaking an unscheduled job (2)



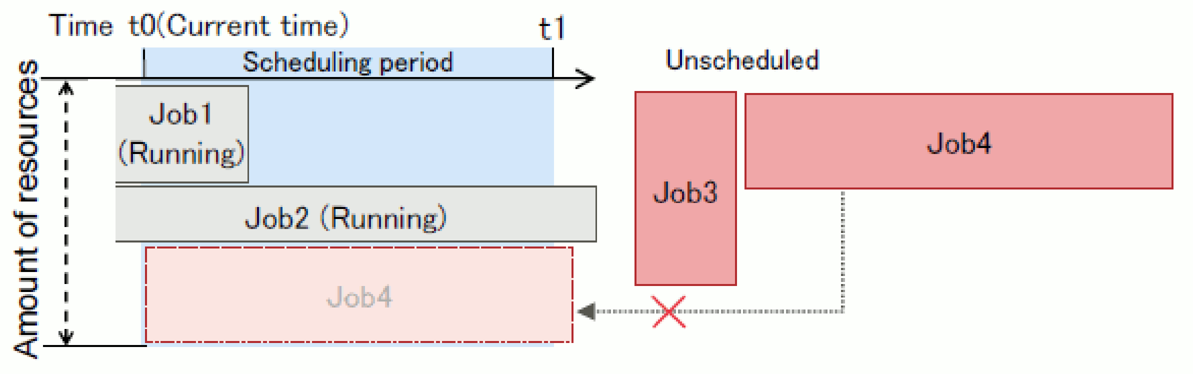
In the following example, Job5 does not be allowed overtaking Job4. Because if Job5 is scheduled to run from time t0, its expected end time exceeds the expected end time of Job2, which is the latest.

Figure 2.25 Example of not allowed to overtake the unscheduled job (1)



In the following example, Job4 does not be allowed overtaking Job3. Because if Job4 is scheduled to run from time t0, its expected end time does not exceed the latest expected end time of Job2, but exceeds the scheduling period.

Figure 2.26 Example of not allowed to overtake the unscheduled job (2)

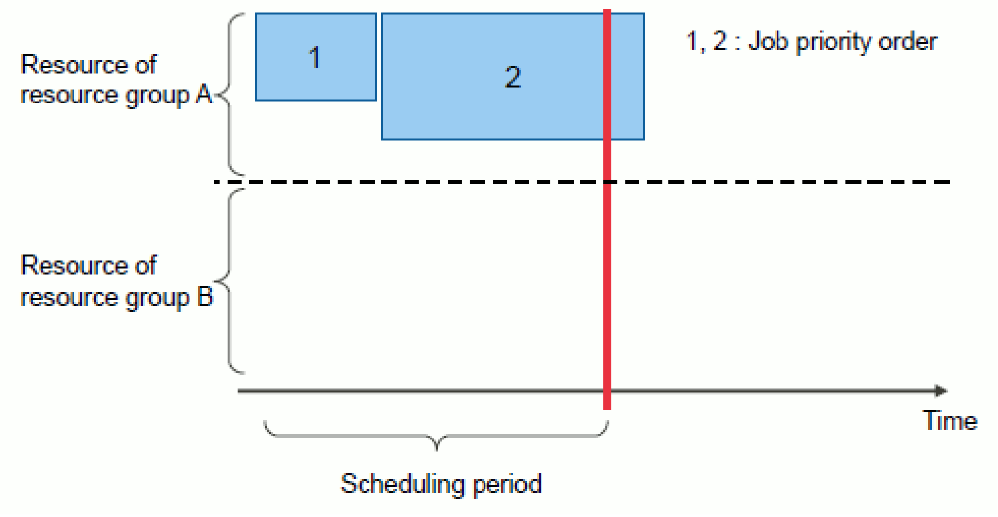


Note

If the elapse time limit of the job is longer than the scheduling period, the expected end time always exceeds the end of the scheduling period regardless of resource usage.
A value that is longer than the elapse time limit of job is recommended to be set to the scheduling period. The administrator should examine the tendency of the elapse time limits of the job.

If the backfill function is enabled, the overtaking target range of the job is, by default, all jobs in the same resource group.
The following figure shows an example. In this example, the job with priority order *N* is called Job*N*, and there are no jobs running for the sake of simplicity.
This example assumes that job 1 and job 2 are already scheduled.

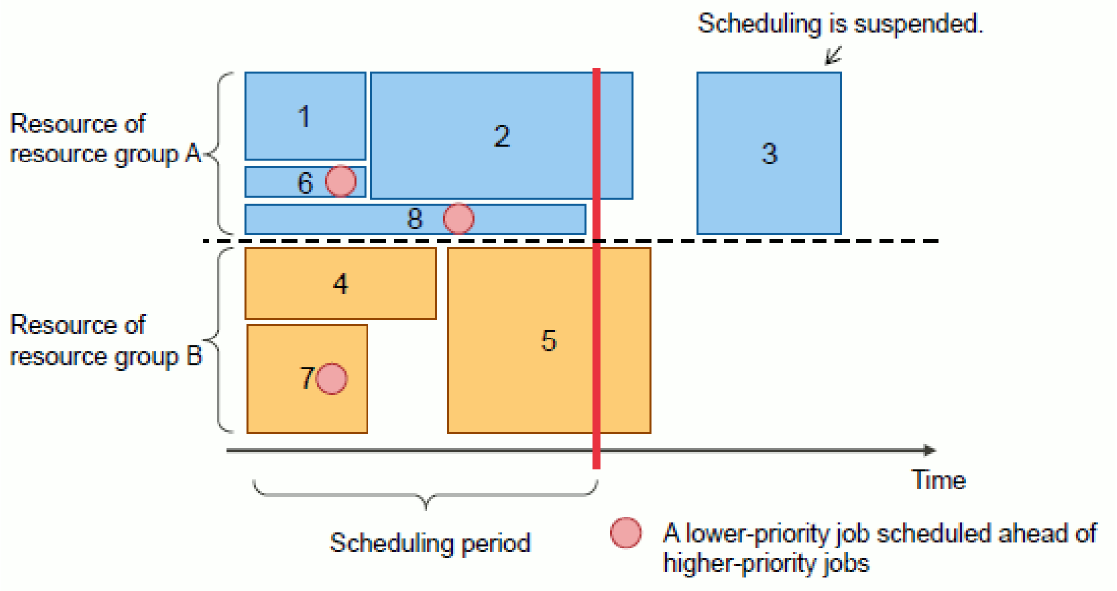
Figure 2.27 Example of overtaking the job in scheduling on multiple resource groups (1)


Resource of resource group A
1
2
1, 2 : Job priority order
Resource of resource group B
Time
Scheduling period

When Job3 to job8 are submitted as shown below, the situation becomes "Figure 2.28 Example of overtaking the job in scheduling on multiple resource groups (2)".

- Job 3 (Submitted to resource group A)
  Since no resources are free in resource group A, the scheduling of job 3 is suspended.
- Job 4 (Submitted to resource group B)
  Since resources are free in resource group B, job 4 is scheduled. The comparison on whether a job has a lower priority is limited to jobs in resource group B, where job 4 belongs. Therefore, jobs 1, 2, and 3 in resource group A are not considered as jobs with lower priority. In this case, job 4 is executed before job 1, 2, and 3. But this behavior is not due to the backfill function.
- Job 5 (Submitted to resource group B)
  The scheduled end time of job 5 is out of the scheduled period. However, job 5 is scheduled because there are no higher-priority jobs in the same resource group. This does not affect the execution start time of job 3.
- Job 6 (Submitted to resource group A)
  Job 7 (Submitted to resource group B)
  Job 8 (Submitted to resource group A)
  For these jobs, the resource group to which they were submitted has available resources. In addition, the execution of lower-priority jobs does not affect the execution start time of the higher-priority jobs in their resource groups. Therefore, in jobs 6 to 8, lower-priority jobs can be executed ahead of higher-priority jobs.

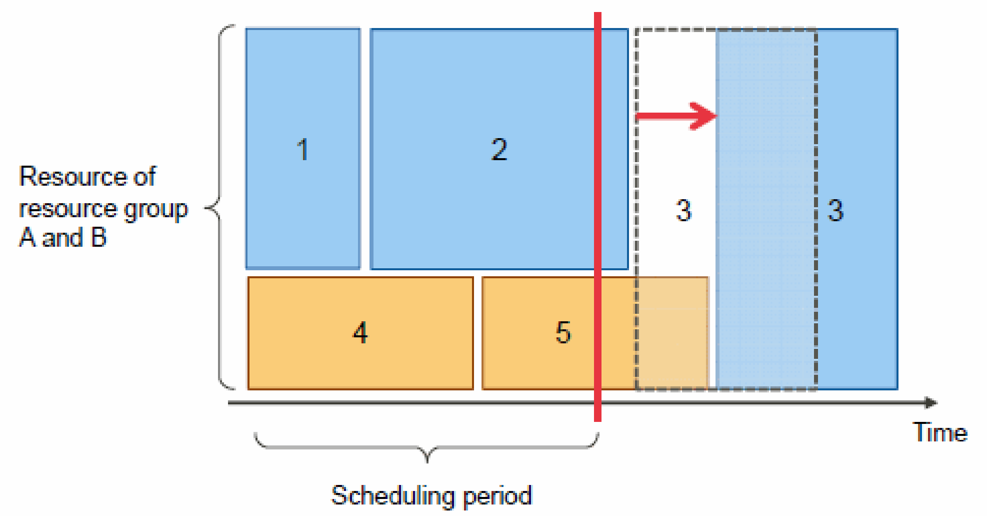
Figure 2.28 Example of overtaking the job in scheduling on multiple resource groups (2)


Scheduling is suspended.
Resource of resource group A
1
2
3
6
8
Resource of resource group B
4
5
7
Time
Scheduling period
A lower-priority job scheduled ahead of higher-priority jobs

If multiple resource groups share the same resources, there may be an influence on the higher-priority jobs in another resource group when lower-priority jobs are executed before higher-priority jobs in a resource group.

For example, you can see this situation below where job 5 with a lower priority in resource group B is executed ahead of job 3, which has a higher priority, in resource group A. In this case, since both resource groups share the same resources, the execution start time of job 3 is delayed due to the prioritized execution of job 5.

Figure 2.29 Example of overtaking the job in scheduling on multiple resource groups (3)



As a way to prevent such a disadvantage, you can set the backfill function for a job that has a lower priority in a resource unit. To do so, set the BackfillTarget setting item in the pmpjm.conf or papjm.conf file for rscunit.

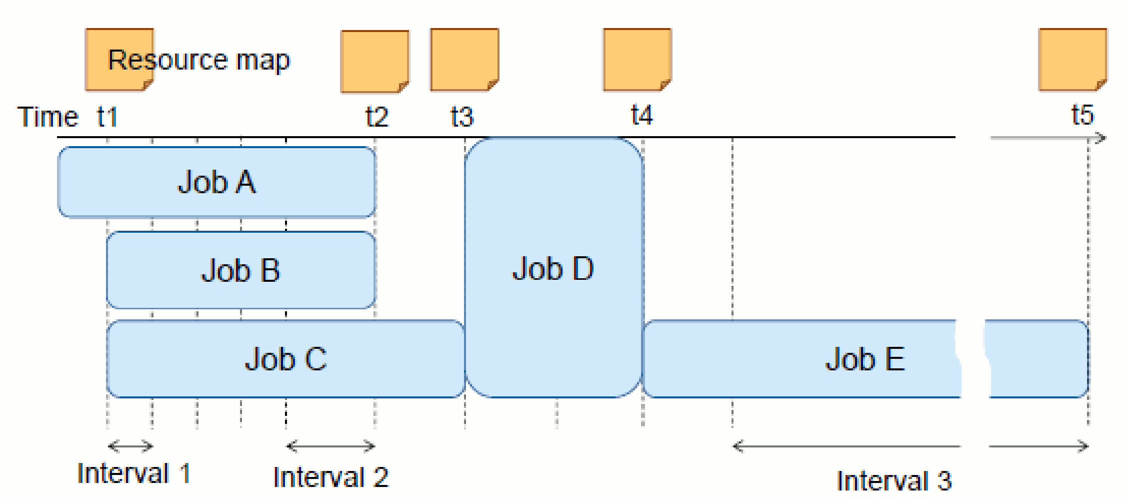
Minimum interval for creating the resource map (CreateRscMap)

The resource maps are created for the expected job start and end time at which the resource allocation changes. The expected job start and end time are rounded in the minimum interval of creating the resource map.

If the interval is short, the accuracy of the expected job start and end time improves. However, the number of resource maps increases. It might cause to increase the load of the scheduling function.

Therefore, the scheduling period is divided at some time zones, and the minimum interval of creating the resource map can be changed every time zone (ex. the value is set as short to the near future, and long to the far future).

Figure 2.30 Creating the resource maps for future at a certain time



Buffer time for the job execution interval (DecidedGap)

A job may end later than the planned time because job resource allocation or release took a longer time. Consequently, if jobs are scheduled without time gaps between them, a resource shortage may occur at the time between one job and the next, so the later job cannot be executed.

When scheduling jobs with the job scheduler function, you can set a time margin that considers possible delays at job end. These time margins are called "buffer periods."

If a buffer period is too long, the planned execution start time of a job following another job may be unnecessarily delayed.

Figure 2.31 Buffer periods not taken into consideration

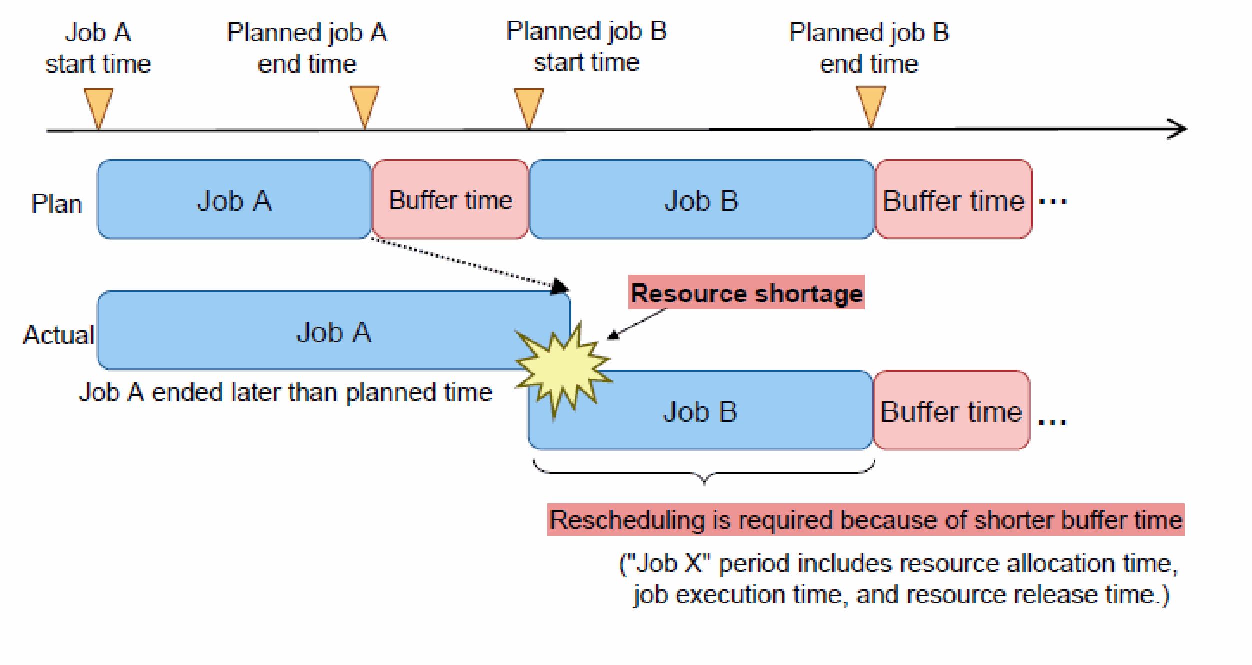
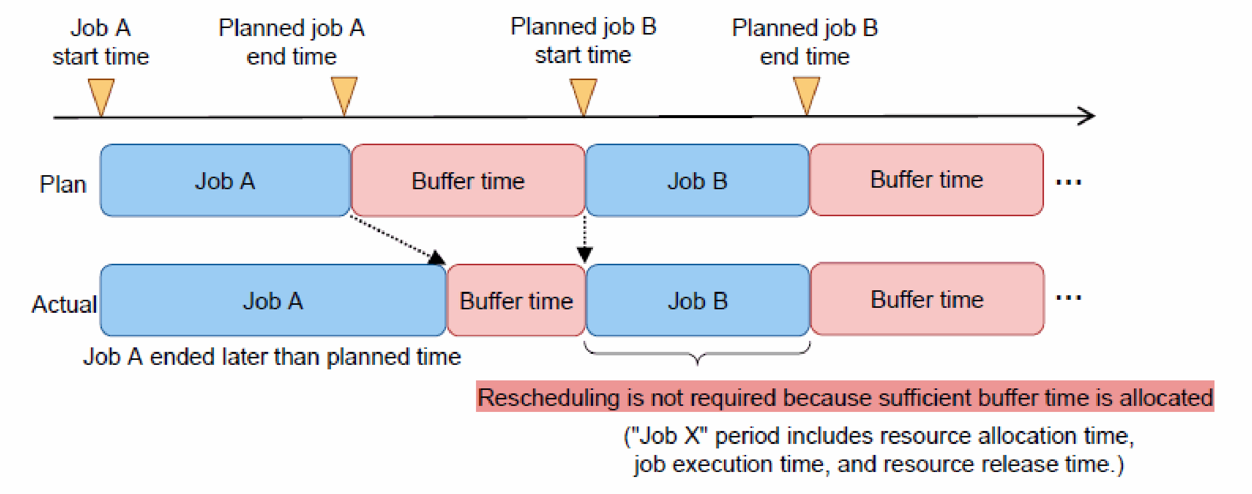

Job A start time
Planned job A end time
Planned job B start time
Planned job B end time
Plan
Job A
Buffer time
Job B
Buffer time
…
Resource shortage
Actual
Job A
Job A ended later than planned time
Job B
Buffer time
…
Rescheduling is required because of shorter buffer time
("Job X" period includes resource allocation time, job execution time, and resource release time.)

Figure 2.32 Buffer periods taken into consideration


Job A start time
Planned job A end time
Planned job B start time
Planned job B end time
Plan
Job A
Buffer time
Job B
Buffer time
…
Actual
Job A
Buffer time
Job B
Buffer time
…
Job A ended later than planned time
Rescheduling is not required because sufficient buffer time is allocated
("Job X" period includes resource allocation time, job execution time, and resource release time.)

Grace time for rescheduling (Grace)

A job may end earlier than expected, before the planned end time, making resources available but also lowering the efficiency of resource use. In such a case, it may be better to review the subsequent schedule.
However, frequent rescheduling increases the load on the job scheduler function, possibly delaying the scheduling process.

Therefore, with the job scheduler function, you can specify a "grace time" for rescheduling. The grace time serves as a threshold value for the difference between the planned job end time and actual job end time. It is used to determine whether to reschedule jobs.

Figure 2.33 Relationship between the grace time and rescheduling

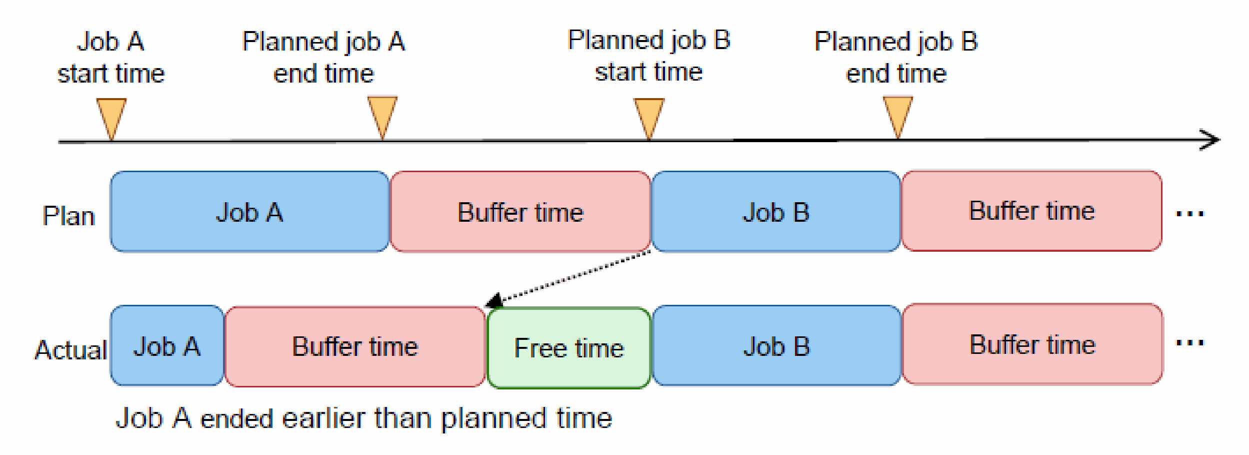


Figure 2.34 Free time longer than the grace time

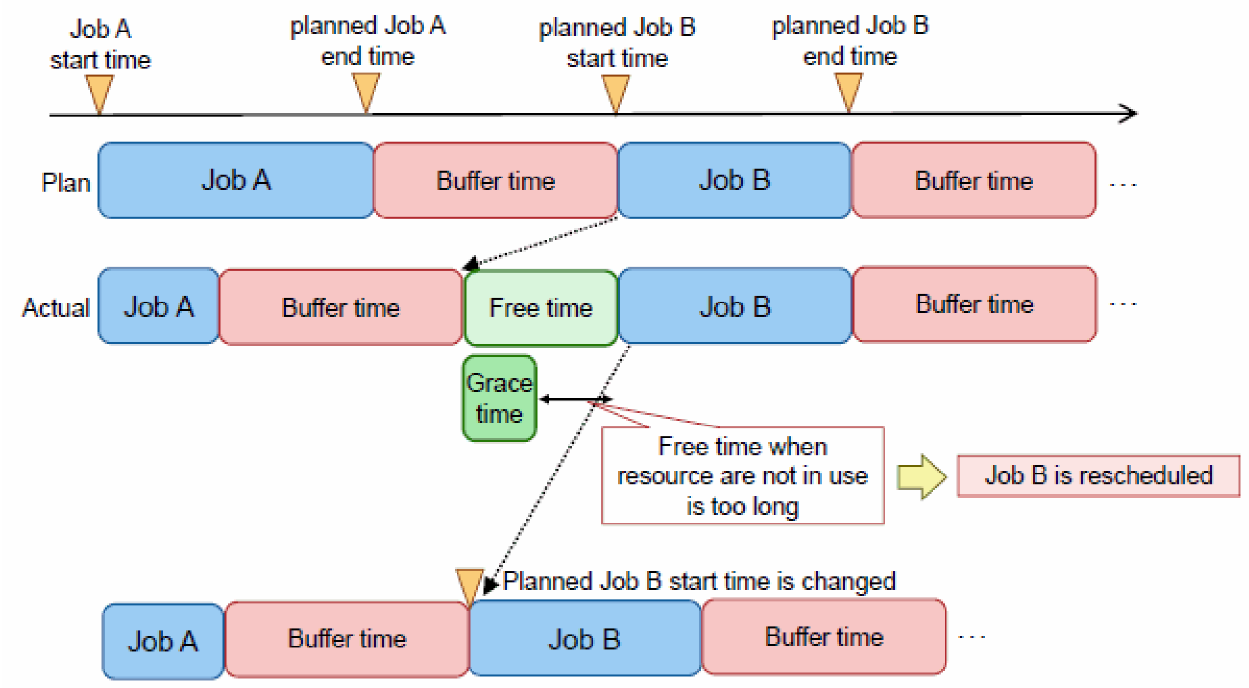
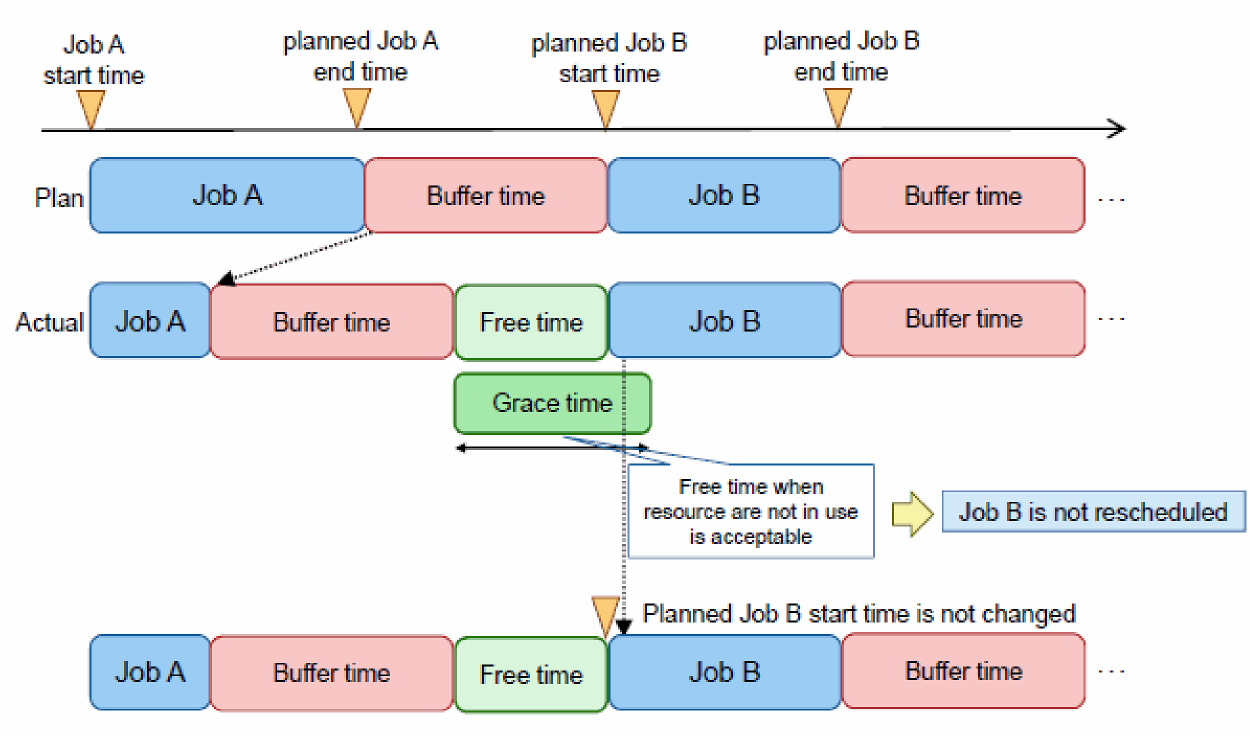


Figure 2.35 Free time shorter than the grace time



Information

The planned end time of a job is calculated as the sum of the planned start time (resource allocation start time), the upper limit of the executable time, and the above-described buffer period.

However, the planned start time and planned end time may change according to the schedule, and they do not need to be calculated exactly. The actual scheduling uses the time values obtained by rounding up time values by a certain unit of time. Thus, multiple jobs can be scheduled together, decreasing the load on processing. In addition, the unit of time for rounding up increases for plans further in the future as the plans become less certain.

The job scheduler function provides parameters, described below, for adjusting rescheduling triggers in order to perform more advanced job scheduling.

Rescheduling start wait time

The jobs waiting to be executed are rescheduled by the job scheduler function triggered by an event such as job submission or end. If job submission or end is a frequent occurrence when numerous jobs are waiting to be executed, rescheduling is repetitious. In the resulting situation, lower priority jobs waiting to be executed may remain unexecuted indefinitely. To prevent this situation, you can adjust the rescheduling start wait time for when a job is submitted or ends.

Note

The following two settings adjust the start time for rescheduling: "rescheduling grace time setting," which is mentioned above, and "rescheduling start wait time setting."
The "rescheduling grace time setting" can adjust the rescheduling frequency only when a job ends. On the other hand, the "rescheduling start wait time setting" can adjust the rescheduling frequency not only when a job ends but also when a new job is submitted. Thus, since the "rescheduling start wait time setting" can deal with a greater variety of situations, we recommend adjusting rescheduling using the setting in normal occasions.

The following parameters are used with the rescheduling start wait time setting.

- Time to wait until rescheduling starts if a job ends earlier than the upper limit of the elapsed execution time (GraceOverReschedAggregateTime, AggregateReschedGapFactor)
  A shorter wait time is selected in a comparison of GraceOverReschedAggregateTime (seconds) and the value obtained by multiplying AggregateReschedGapFactor (milliseconds) by the number of queued jobs.

- Time to wait until rescheduling starts if a batch job is submitted (NewJobBatchReschedAggregateTime, AggregateReschedNewJobFactor)
  A shorter wait time is selected in a comparison of NewJobBatchReschedAggregateTime (seconds) and the value obtained by multiplying AggregateReschedNewJobFactor (milliseconds) by the number of queued jobs.

- Time to wait until rescheduling starts if an interactive job is submitted (NewJobInterReschedAggregateTime)
  A shorter wait time is selected in a comparison of NewJobInterReschedAggregateTime (seconds) and the value obtained by multiplying AggregateReschedNewJobFactor (milliseconds) by the number of queued jobs.

- Time to wait until rescheduling starts if a job parameter is changed with the pjalter command or pmalter command (PmpjmAlterReschedAggregateTime)

- Rescheduling start wait time when there are jobs waiting for scheduling (ExtendSchedPeriod)
  The submission of many jobs extends the scheduling period per job. Consequently, rescheduling during the scheduling process is more likely to occur, resulting in a situation with incomplete scheduling. This parameter specifies the wait time for scheduling to complete when rescheduling occurs during the scheduling process.

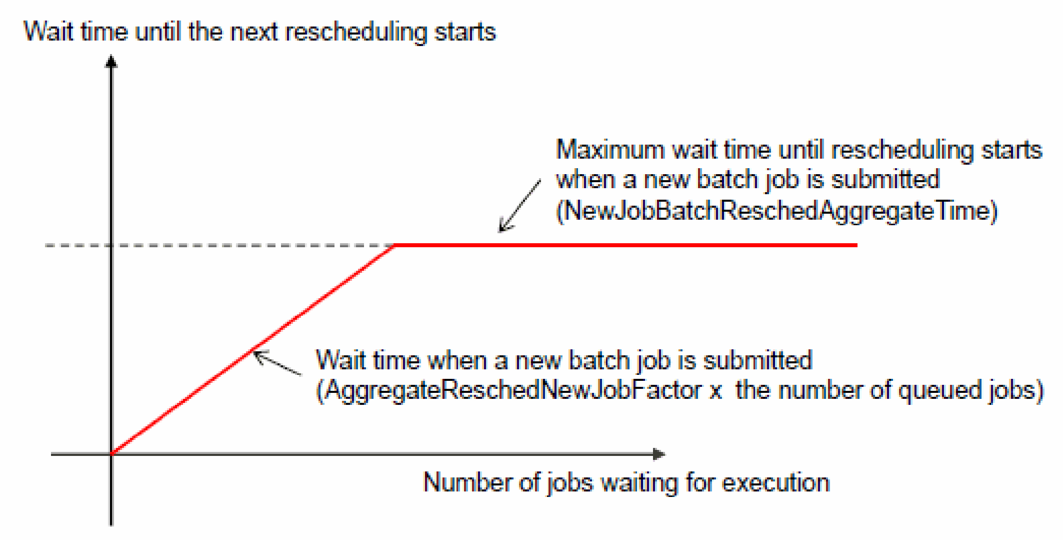
The following are examples of the above cases.

Example 1: When a new batch job is submitted

The time to wait until rescheduling starts when a new batch job is submitted is proportional to the number of jobs waiting for execution.

```
AggregateReschedNewJobFactor x number of jobs waiting for execution
```

Figure 2.36 Time to wait until rescheduling starts when a new batch job is submitted



Increase the NewJobBatchReschedAggregateTime value to extend the time to wait until rescheduling starts.

Example 2: When a job ends

The time to wait until rescheduling starts when a job ends is proportional to the number of jobs waiting for execution.

```
AggregateReschedGapFactor x number of jobs waiting for execution
```

Increase the GraceOverReschedAggregateTime value to extend the time to wait until rescheduling starts.

Note

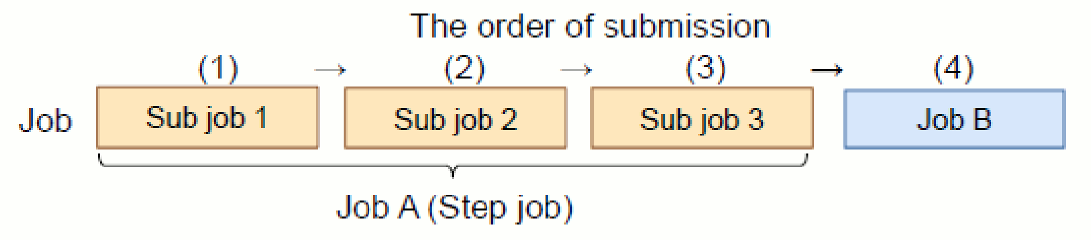
- You can reduce the time until resources are available and effectively use resources by reducing the job rescheduling start wait time. However, if the wait time is too short, rescheduling frequently occurs, which may lead to a delay in the scheduling process.
- If the wait time until rescheduling starts needs to be determined because a new event occurs after the wait time until the next scheduling starts is determined due to the occurrence of an event such as new job submission or job end, the wait time determined previously and the wait time determined at this event are compared. Then, the shorter wait time is selected. The longer wait time setting is discarded.

2.5.4.3 Scheduling of sub jobs of a step job

Job scheduling is performed each time that a job is submitted. However, the scheduling of a sub job of a step job is performed after the preceding sub job ends.
Consequently, multiple sub jobs of a step job may remain unscheduled when another job is submitted. If the jobs in this situation are scheduled based on the submission time, the multiple sub jobs of the step job submitted earlier are executed with priority over the newly submitted job. To the end user, the newly submitted job seems to have been left unexecuted because it was passed by the sub jobs of the step job.

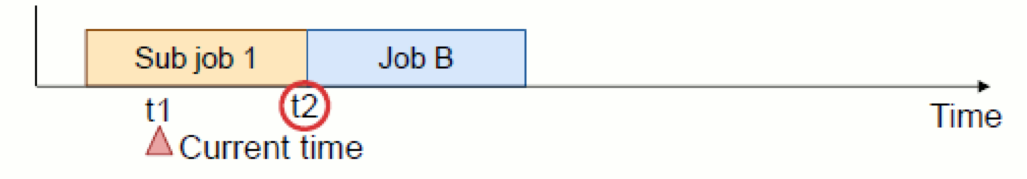
The following example describes the submission of job A (step job) and job B (Figure 2.37).

Figure 2.37 How the priority of a job becomes lower than the priority of a sub job of a step job (1)



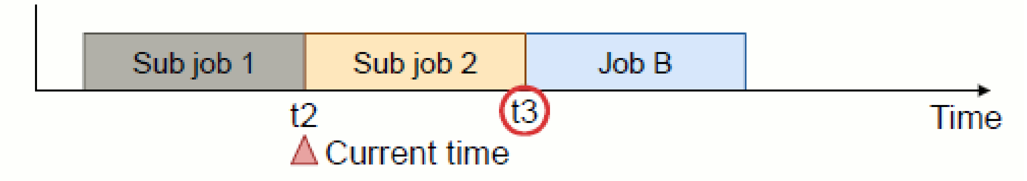
First, job B is submitted during execution of sub job 1 of the step job. The scheduled execution time of job B is t2 (Figure 2.38). Sub jobs 2 and 3 have yet to be scheduled.

Figure 2.38 How the priority of a job becomes lower than the priority of a sub job of a step job (2)



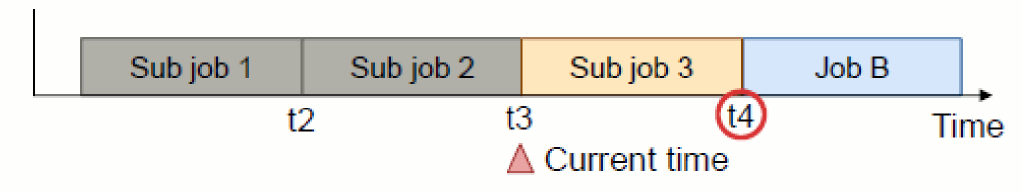
Rescheduling takes place when sub job 1 ends at time t2. The submission times of sub job 2 and job B are compared. The submission time of job B is later than that of sub job 2, so the execution of job B is postponed until time t3, after sub job 2 ends (Figure 2.39).

Figure 2.39 How the priority of a job becomes lower than the priority of a sub job of a step job (3)



When sub job 2 ends, sub job 3 is executed with priority over job B, like in the above step. The execution start time of job B is further postponed to time t4 (Figure 2.40).

Figure 2.40 How the priority of a job becomes lower than the priority of a sub job of a step job (4)



As shown above, after being scheduled once, job B is passed by a sub job of the step job each time that rescheduling takes place. The execution start time of job B is postponed accordingly.

To prevent this, you can configure the scheduling times of sub jobs to be considered as their submission times.

See

For details on the setting method, see "3.4.5 Settings for advanced job scheduling."

2.5.4.4 Guarantee of planned job execution start time (setting that prevents a delay in the job execution start time)

The planned execution start time of a job may change when a new job with a higher priority is submitted. To prevent this, the administrator can configure a setting for Guarantee of planned job execution start times (setting that prevents a delay in the job execution start time). Enabling this setting prevents a delay in the planned execution start time by prohibiting rescheduling of jobs scheduled to be executed within a certain period from that point in time, even if new jobs are submitted. However, the execution start time automatically becomes earlier if resources are free.

Note

- The planned execution start time of a job belonging to the relevant resource unit or resource group is reset by the pmpjmopt command stopping and restarting the submission of the job to the resource unit or resource group.
- If the compute cluster management node starts or a failover occurs, the planned execution start time of all jobs are reset even when the guarantee of planned execution start times is in effect.
- If the following event occurs, the planned job execution start time may be reset even when the guarantee of planned execution start times is in effect.
    - The compute node to be allocated cannot be used.
      Since the planned execution start time of the job to be allocated the compute node is reset, the execution start time of the job is later than that of other jobs.
      The following cases are examples:
        - An error occurs, such as the node going down.
        - The range of a node belonging to the resource group is changed.
    - A job scheduler function-related setting, such as the end time for creating a resource map or the time interval for creating a resource map, is changed by the papjmadm or pmpjmadm command.
    - A job parameter is changed by the pmalter or pjalter command.
    - A new deadline schedule is set by the --enforce option of the padeadline command.
    - The quantitative limit on simultaneously executed jobs, quantitative limit on simultaneously used nodes, or quantitative limit on simultaneously used CPU cores, all of which are setting items of the job ACL function, is changed by the pmjacladm command.
- If the setting for guarantee of planned execution start times is enabled, the planned execution start time of the job is guaranteed even if another job is aborted because of a system problem, such as a node failure, and will be re-executed.
  Therefore, the aborted job may be re-executed later than the job with the planned execution start time that is guaranteed.

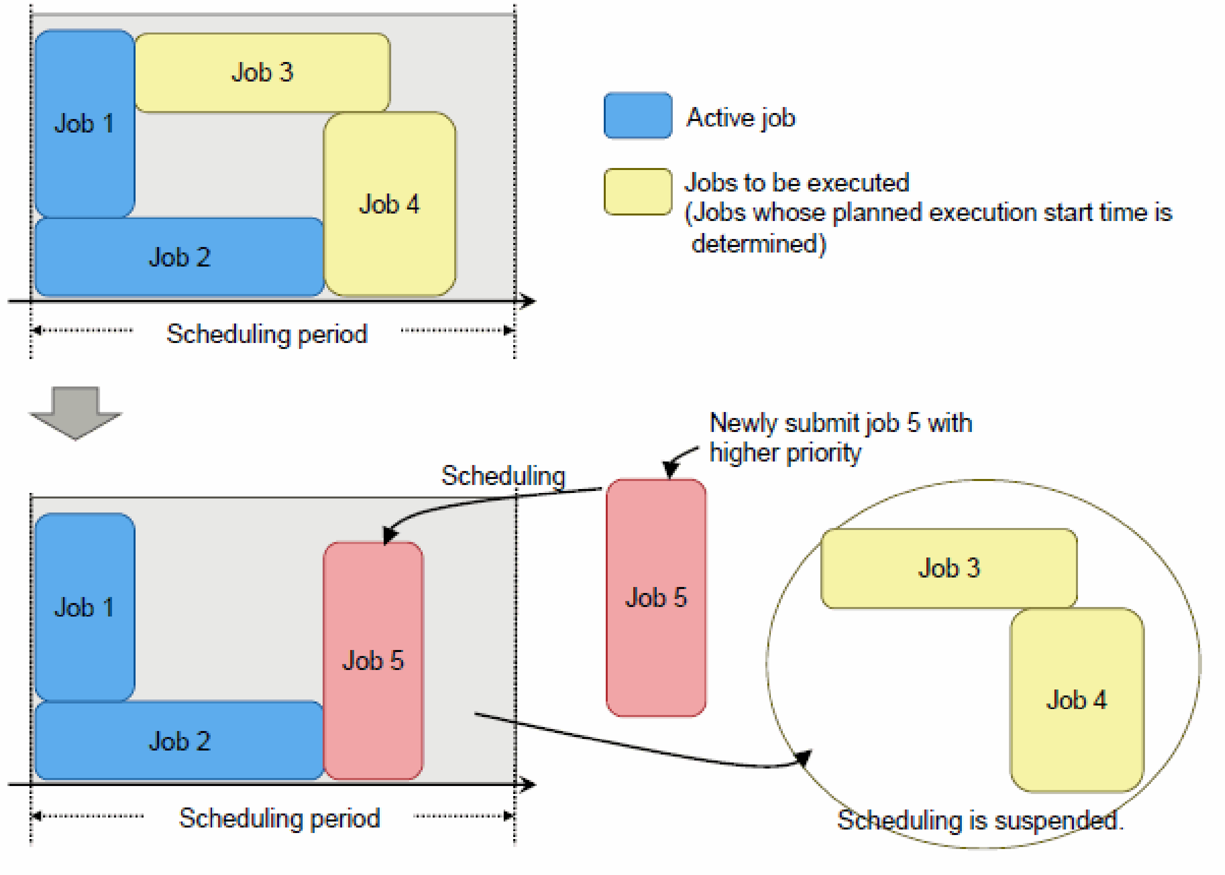
- Suppose that you are setting a deadline schedule. Then, the set nodes and time zone for the deadline schedule may overlap the nodes and time zone to be used by a job that guarantees the planned execution start time. In this case, set the deadline schedule by specifying the --enforce option in the padeadline command.

The following example shows the operation when a new higher-priority job is submitted.

When the setting for guarantee of planned execution start times is disabled

If a new higher-priority job is submitted when the setting for guarantee of planned execution start times is disabled, the scheduling of the higher-priority job has priority. Therefore, the start time of the job with the fixed schedule is rescheduled.

Figure 2.41 When the setting for guarantee of planned execution start times is disabled

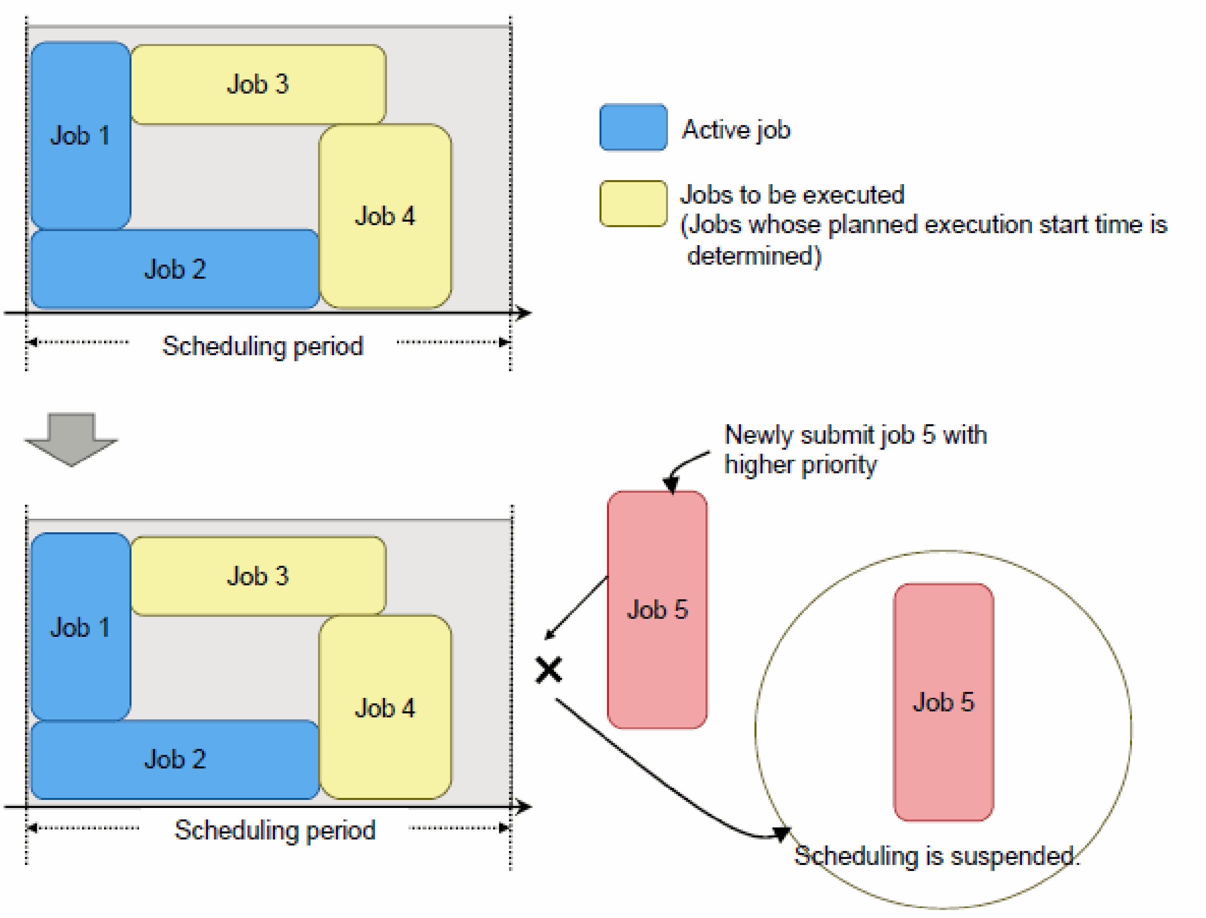


In the above example, the planned execution start times of jobs 3 and 4 are rescheduled since job 5 with a higher priority is submitted. However, the scheduling of jobs 3 and 4 is suspended because the rescheduled execution start times cannot be assigned in the scheduling period (SchedulePeriod or DynamicSchedulePeriod).

When the setting for guarantee of planned execution start times is enabled

If the setting for guaranteeing planned execution start times is enabled, a job with a scheduled execution start time is assigned within the scheduling period even after rescheduling.

Figure 2.42 When the setting for guarantee of planned execution start times is enabled



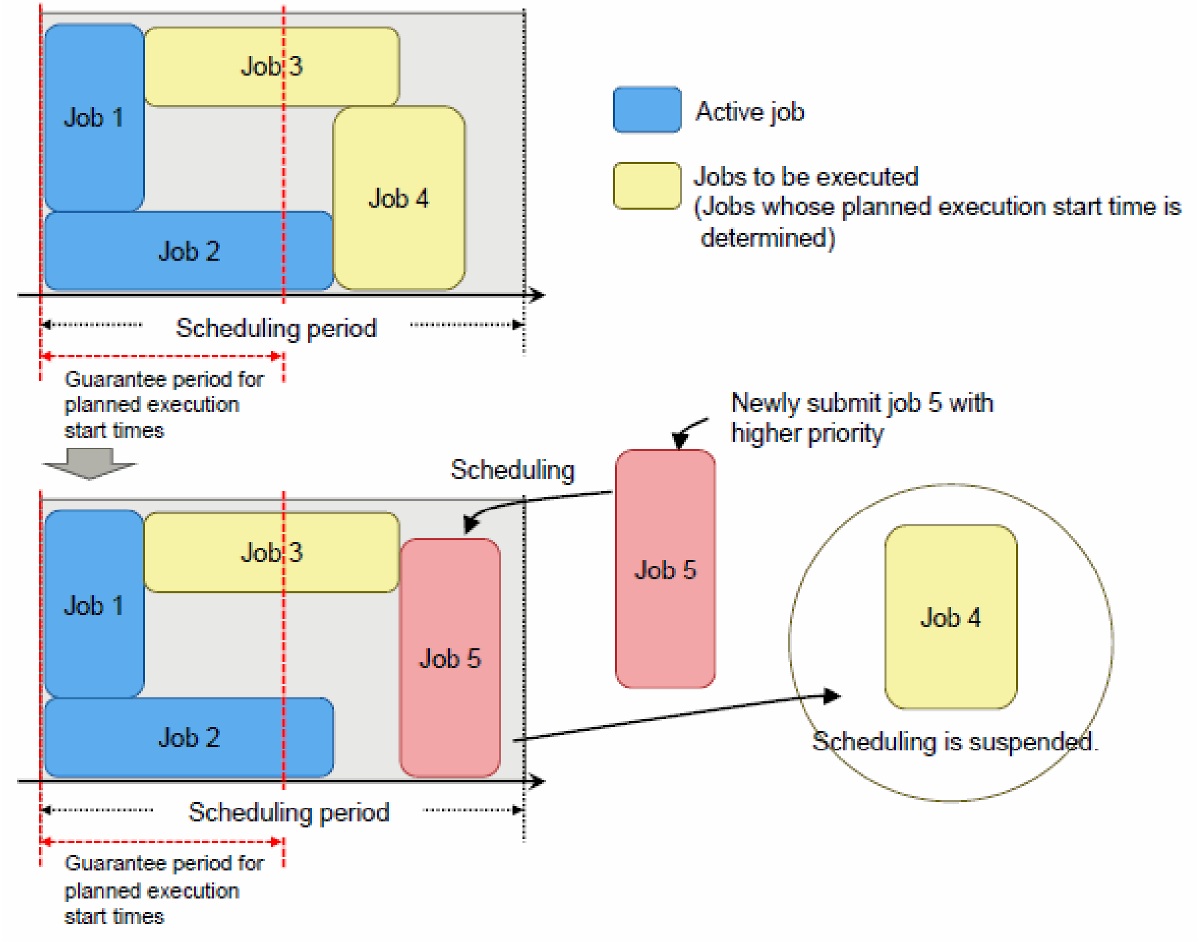
In the above example, job 5 cannot be assigned in the scheduling period since the planned execution start times of jobs 3 and 4 are not extended even when job 5 with higher priority is submitted. Therefore, even if a higher-priority job is submitted, the scheduling of the job may be suspended.

To prevent this situation, change the guarantee period of planned execution start times (NotReschedPeriod).

When the setting for guarantee of planned execution start times is enabled and when the guarantee period of planned execution start times is reduced to half the time of the scheduling period

If the guarantee period of planned execution start times is set to less than the scheduling period, the planned execution start time that exceeds the guarantee period of planned execution start times is reset at rescheduling. Accordingly, a newly submitted job with a higher priority can be easily assigned within the scheduling period.

Figure 2.43 When the setting for guarantee of planned execution start times is enabled and when the guarantee period of planned execution start times is reduced to half the time of the scheduling period


Job 1
Job 3
Job 4
Job 2
Active job
Jobs to be executed
(Jobs whose planned execution start time is determined)
Scheduling period
Guarantee period for planned execution start times
Newly submit job 5 with higher priority
Scheduling
Job 5
Job 4
Scheduling is suspended.
Job 1
Job 3
Job 5
Job 2
Scheduling period
Guarantee period for planned execution start times

In the above example, the planned execution start times of jobs 3 and 4 are rescheduled since job 5 with a higher priority is submitted. Since the planned execution start time of job 3 is within the guarantee period of planned execution start times, it is assigned in the planned execution start time. However, the planned execution start time of job 4 is set outside the guarantee period of planned execution start times. Therefore, the scheduling of job 4 is suspended. This creates an available time in the scheduling period, where job 5 can be assigned.

See

For details on how to set, see "3.4.1.2 Default value settings for resource units", "3.5.1.2 Resource group settings", and "3.4.5 Settings for advanced job scheduling".

2.5.4.5 Limit on the number of jobs to schedule

Set the ExtendSchedPeriod parameter (one of the parameters in "Rescheduling start wait time"), which is described in "2.5.4.2 Job scheduling parameters," to have rescheduling done only after waiting for the completion of scheduling. However, an increase in the number of jobs to schedule will extend the scheduling period. Consequently, newly submitted jobs may not be promptly scheduled.

To prevent this, you can limit the number of jobs to be scheduled each time. When the number of jobs to schedule reaches the specified value, the remaining jobs are processed at the next scheduling time. In step jobs and bulk jobs, the number of sub jobs is counted as the number of jobs.

To limit the number of jobs to schedule, you can specify the following methods of selecting the jobs.

Table 2.9 Methods of selecting jobs to schedule

Selection Method	Description
In order by job priority and job submission time	Sort the jobs in a resource unit by priority and by submission time. Select jobs in order from the top. For details on job priority within a resource unit, see "2.5.2.1 Job selection policy."
In line with job selection policy	Select jobs according to the job selection policy settings described in "2.5.2.1 Job selection policy." However, the following selection policies are ignored even if set: usr_in_grp_prio (User priority within the group) job_prio (Job priority) group_fairshare (Priority based on the group's fair share value) usr_in_grp_fairshare (Priority based on the user's fair share value within the group) user_fairshare (Priority based on the user's fair share value)

See

To limit the number of jobs to schedule, set the JobSchedulingTargetLimit and JobSchedulingTargetMode items in the papjm.conf file. For details, see "3.4.1.2 Default value settings for resource units."

2.5.4.6 Elapsed time limit for a job

When submitting a job, you can specify the elapsed time limit (executable time) for the job in the following format.

Specifying an elapsed time limit value

```
{-L | --rsc-list} elapse=limit
```

When the elapsed time of a job reaches limit, the SIGXCPU signal is sent to the job. The job is forcibly terminated 10 seconds later.

Specifying a range of elapsed time limit values [FX]

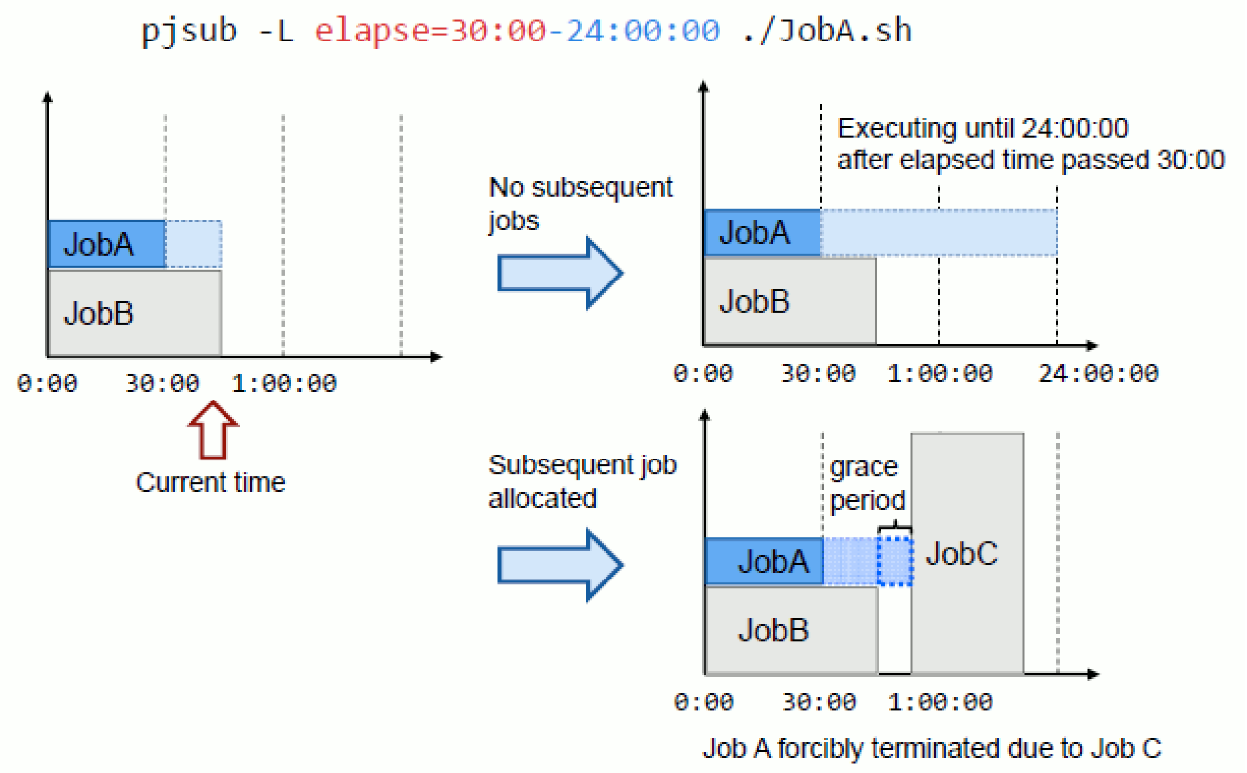
```
{-L | --rsc-list} elapse=min_limit-                (*) Hyphen needed after min_limit
{-L | --rsc-list} elapse=min_limit-max_limit
```

Even after the elapsed time reaches min_limit, execution can continue until the elapsed time reaches max_limit. However, when the elapsed time has passed min_limit and the job is still running, a node may be required for a subsequent job or processing may enter a period where nodes are reserved by deadline scheduling. In either case, the job is forcibly terminated even if the elapsed time has not yet reached max_limit.
To not limit the maximum time that execution can continue, specify "unlimited" in max_limit (e.g., elapse=600-unlimited). If max_limit is not specified (e.g., elapse=600-), the set value in the job ACL function is applied.
By specifying a range of elapsed time limit values for a job in this submission method, you can effectively utilize available compute nodes and expect improvement in system availability (job fill rate).
When a job is forcibly terminated, a grace time is given for termination processing of the job. By default, the SIGTERM signal is sent to the job 10 seconds before termination.
When the elapsed time reaches max_limit, the SIGXCPU signal is sent to the job. The job is forcibly terminated 10 seconds later.

Information

The administrator can change the period between the sending of the SIGTERM signal and the forced termination by using the AdaptiveElapsedTimeJobTerminateGrace item in the papjm.conf or pmpjm.conf file. For details on how to change it, see "3.4.1 Settings for job operation management function in a cluster (papjm.conf file)."

Figure 2.44 Specifying a range of elapsed time limit values



The following definition items of the job ACL function relate to the elapsed time limit values for jobs:

- Specifying which format, elapse=limit or elapse=min_limit[-max_limit], to apply as the operation when the elapse parameter is omitted at the job submission time

  ```
  define elapsed-time-mode
  ```

- Availability of the methods for specifying elapsed time limit values

  ```
  execute pjsub-fixed-elapsed-time
  execute pjsub-adaptive-elapsed-time
  ```

- Default value for the elapsed time limit and upper and lower limits on the values that can be specified

  ```
  joblimit elapse
  joblimit adaptive-elapsed-time-min
  joblimit adaptive-elapsed-time-max
  joblimit interact-adaptive-elapsed-time-min
  joblimit interact-adaptive-elapsed-time-max
  joblimit adaptive-node-elapse-min
  joblimit adaptive-node-elapse-max
  joblimit interact-adaptive-node-elapse-min
  joblimit interact-adaptive-node-elapse-max
  ```

See

For details on the items and setting method, see "3.4.4 Job ACL function settings in a cluster.

The following descriptions provide notes on job scheduling when specifying a range of elapsed time limit values for a job (elapse=*min_limit-max_limit*).

Step job

Each sub job of a step job is scheduled to be executed after "planned start time of previously executed sub job + maximum elapsed time limit value."

Deadline schedule

When you set a deadline schedule with the padeadline command, there may be a forecast that an existing job will be running at the start time of the deadline schedule period. If so, the operation for the job is one of the following, depending on the expected elapsed time of the job at that time.

Information

The forecast on whether a job is running at the start time of a deadline schedule period is based on the planned execution start time and elapsed time limit value for the job.

Job has not reached the minimum elapsed time limit value

The deadline schedule is not set because the job execution time is guaranteed until the job reaches the minimum elapsed time limit value.

Job has exceeded the minimum elapsed time limit value

The job is forcibly terminated before the start of the deadline schedule period. If it is expected that the forced termination will not be able to be completed before the start of the deadline schedule period, the deadline schedule is not set.

Specifically, the deadline schedule is not set if the time from the job reaching the minimum elapsed time limit value to the start of the deadline schedule period is shorter than the sum of the following times:

- Grace time before forced termination (Set value of AdaptiveElapsedTimeJobTerminateGrace)
- Approximate execution time of the epilogue script (Set value of EpilogueTime)
- Buffer time between job executions (DecidedGap)

Fair share value

The elapsed time limit value for a job is used to calculate fair share usage and the recovery rate of the fair share value (see "2.5.2.2 Fair share function"). For a job specified with a range of elapsed time limit values, the maximum elapsed time limit value is used to calculate fair share usage and the fair share value.

Job selection policy

The job selection policy determines the job execution order (see "2.5.2.1 Job selection policy"). Some policy items are based on the elapsed time limit values for jobs. For a job specified with a range of elapsed time limit values, the job selection policy handles the order as follows.

Table 2.10 Job execution order when a range of elapsed time limit values is specified

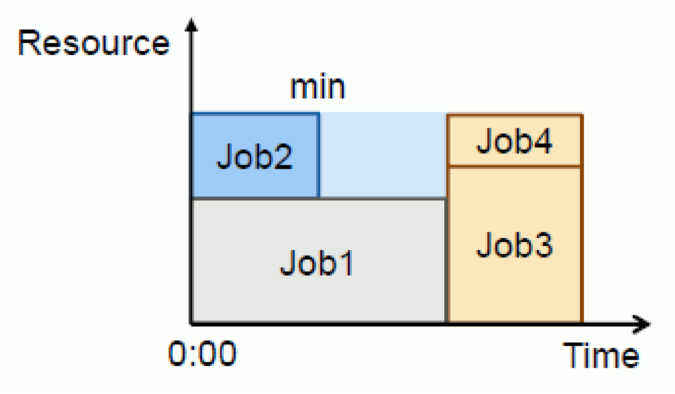
Element	Description
elapse_limit	Evaluates based on the maximum elapsed time limit value.
node_times_elapse	Evaluates based on the product of the number of requested nodes multiplied by the maximum elapsed time limit value.
group_fairshare user_fairshare usr_in_grp_fairshare	Calculates the fair share value, as described above, by using the maximum elapsed time limit value.

Quantitative limits on simultaneously executed jobs and simultaneously used nodes

Jobs are subject to limits on the number of simultaneously executed jobs and the number of simultaneously used nodes even when still running after exceeding the minimum elapsed time limit value. Subsequent jobs are scheduled for execution at times when these limits will not be exceeded. In other words, a job exceeding the minimum elapsed time limit value may be running when a subsequent job is executed, but the job is not forcibly terminated even if the number of simultaneously executed jobs or used nodes at this time exceeds the limit.

The following figure shows an example where the number of simultaneously executed jobs is limited to two.

Figure 2.45 Example of scheduling with a limit on the number of simultaneously executed jobs



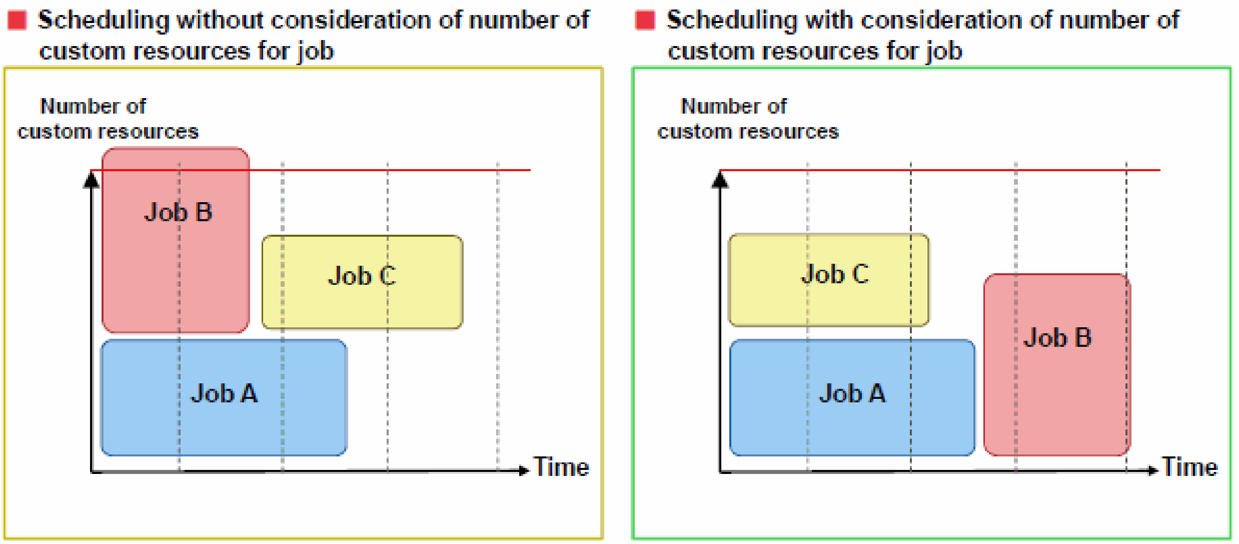
Job 2 is running after exceeding the minimum elapsed time limit value, min. During this time, the number of simultaneously executed jobs reaches the limit of 2. Consequently, subsequent jobs 3 and 4 are not executed. This means that job 2 is not forcibly terminated in order to execute job 4.
The end of job 1 allows for a job to be added to the number of simultaneously executed jobs. As a result, running job 2, which has exceeded the minimum elapsed time limit value, is forcibly terminated in order to execute job 3. This results in an available node, so job 4 is also executed.

2.5.5 Job scheduling function using custom resources

The Job Operation Software allows you to define any resource, such as a software license and power. This arbitrary resource is called "custom resource."
For example, you define a software license as a custom resource and request the necessary quantity of software licenses when submitting a job. Then, the job is scheduled to be executed at a time when there is a sufficient quantity of software licenses for use in the job.
By using custom resources in this way, you can schedule jobs according to various purposes with not only a hardware resource, such as the CPU or memory of a compute node, but also a general resource.

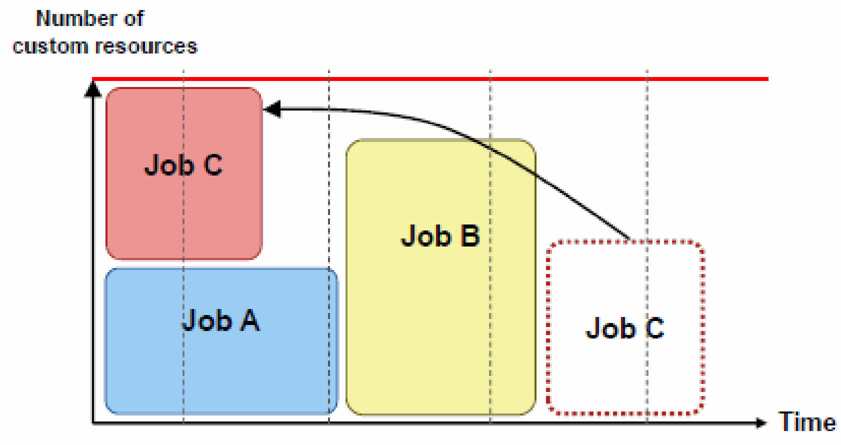
The following figure shows job scheduling with and without consideration of the number of custom resources.

Figure 2.46 Comparison of job scheduling with and without consideration of the number of custom resources


Scheduling without consideration of number of custom resources for job
Number of custom resources
Job B
Job C
Job A
Time
Scheduling with consideration of number of custom resources for job
Number of custom resources
Job C
Job B
Job A
Time

When you have specified a custom resource for a job, the backfill function can be used for the job. (For details on the backfill function, see "2.5.4.1 Backfill function.") In the following example, a custom resource is specified for a job, and the backfill function moves the job past other jobs in the execution order.

Figure 2.47 Scheduling of jobs with custom resources by using the backfill function



Custom resources are defined per resource unit or resource group.
To use a custom resource, the resource can be defined with a numerical value or a type. In addition to these two types of custom resources, custom resources can be defined in the following units.

- Custom resources per resource unit or resource group

  You can define the numerical number and type of resources for all the compute nodes in a resource unit or resource group.

  For example, if the defined numerical number of this resource is eight, the compute nodes in the resource unit or resource group can altogether use eight resources. Also, if the custom resource types a and b are set, all of the compute nodes in the resource unit or resource group can use these two types.

- Custom resources per node

  You can define the numerical number and type of resources for the specified compute node in a resource unit or resource group.

  If the set numerical number of custom resources is eight, each of the specified nodes in a resource unit or resource group can use eight resources. Also, if the custom resource types a and b are defined, each of the specified nodes can use the types a and b.

For details on how to define custom resources, see "3.5.1.6 Custom resource settings."

Custom resources are resources defined by the job administrator. The administrator needs to create a process for determining whether there are enough resources to handle requests from users and whether the requested types are available as custom resources.
The administrator is requested to create programs using the following mechanisms called "hooks" to check the amount of requested resources and determine request types at a specific timing during job submission or execution.

To determine whether a job can be executed, the job operation management function checks the total quantity of custom resources and the quantities or types of custom resources requested by the job. However, the function does not substantially manage custom resources, so it cannot determine the validity of the custom resources specified for a job. In other words, when the job is submitted, the function cannot determine whether enough resources were requested by the job or whether the types requested by the job can be used as custom resources. Therefore, the job operation management function provides the multiple exit functions described below. They can check the requested quantities, requested types, etc. of custom resources. This document generally refers to these functions as "hooks."

- Job information acquisition function and job information setting function APIs that can be used in an exit function (job manager exit function and job scheduler exit function)

- Environment variables that can be referenced in prologue and epilogue scripts (prologue and epilogue function)

- Environment variables that can be referenced in an exit script (resource management exit function)

By using these hooks, you can determine whether the custom resources specified at job submission or at a specific timing during job execution can be used.

Information

On PRIMERGY server, by defining GPUs as custom resources, the Job Operation Software can manage them as computer resources and schedule their allocation to jobs. To define GPUs as custom resources, settings must be configured as described in "Appendix C Settings for Using GPUs [PG]."

See

For details on hooks in the job operation management function, see "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

2.5.5.1 Power cap scheduling function

One function defines system power consumption as a custom resource. When a job is submitted, the function estimates the power consumption of the job and schedules the job with this estimate as the requested quantities of the custom resource. This function is called the power cap scheduling function.

This function enables capping of system power consumption.

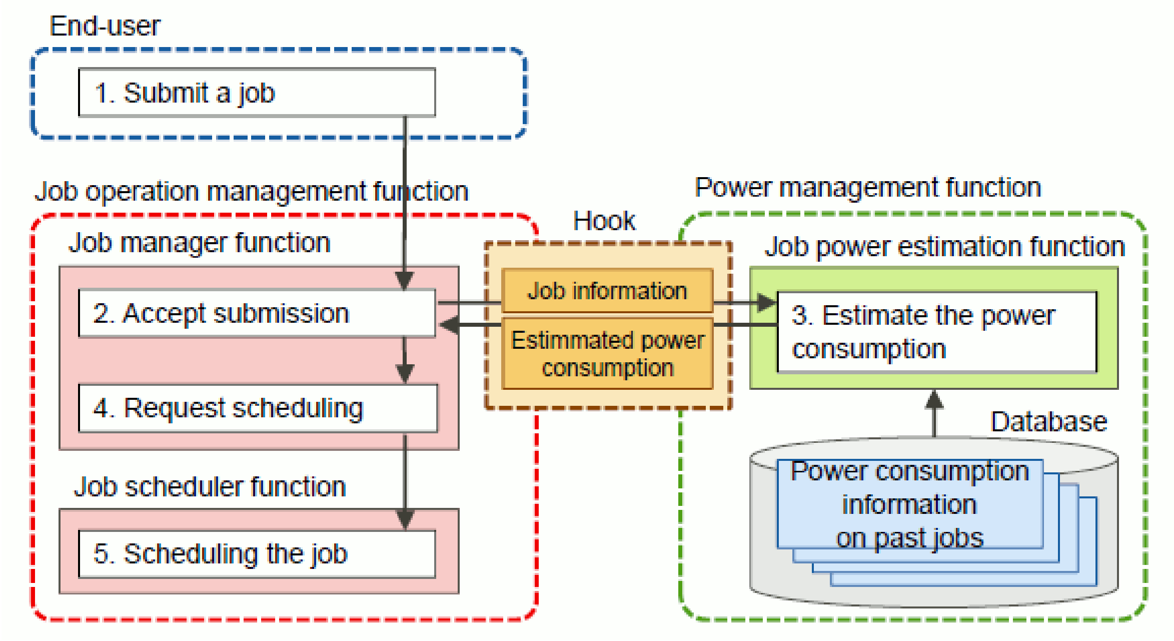
The power cap scheduling function is implemented in linkage with the job operation management function (job manager function, job scheduler function, and job resource management function) and the power management function. For the power cap scheduling function, the job operation management function supports functions for allocating resources to jobs based on the estimated power consumption of the jobs and determining the job execution order.

See

For the power cap scheduling function, a function estimates the power consumption of a submitted job is called "job power estimate function." The power management function supports the job power estimate function. For details on the job power estimate function, see "Job power estimate function" in "Chapter 2 Details of the Power Management Function" in "Job Operation Software Administrator's Guide for Power Management."

The following figure shows the behavior of the power cap scheduling function at the submission of a job with power as a resource.

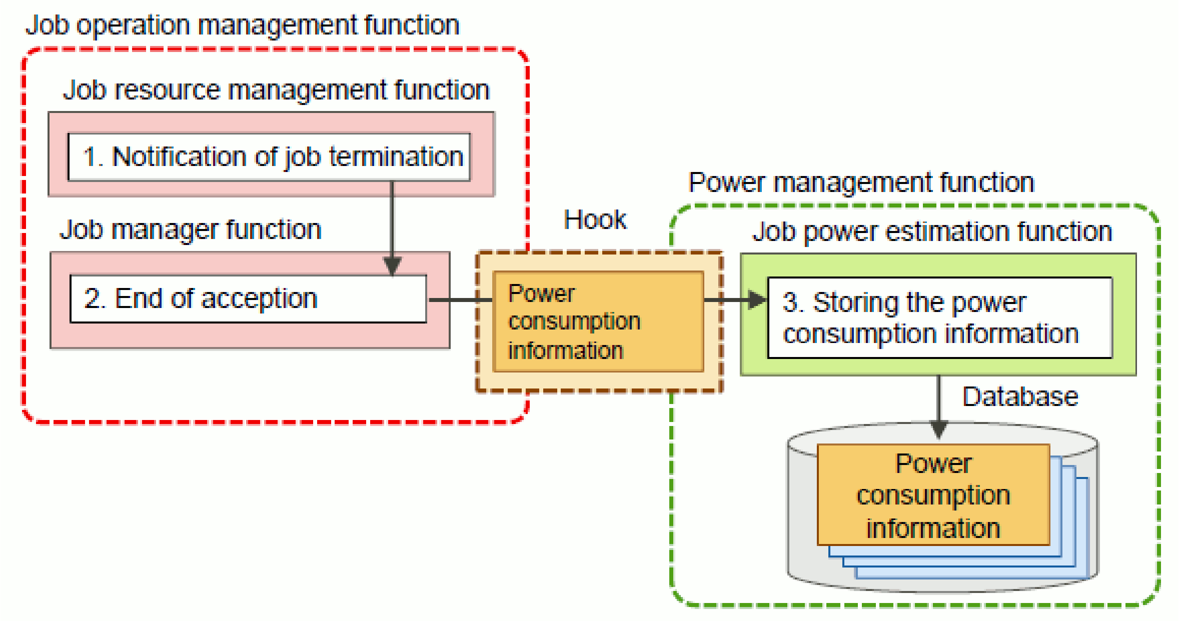
Figure 2.48 Behavior of the power cap scheduling function at the submission of a job with power as a resource (At job submission)


End-user
1. Submit a job
Job operation management function
Job manager function
2. Accept submission
4. Request scheduling
Job scheduler function
5. Scheduling the job
Hook
Job information
Estimmated power consumption
Power management function
Job power estimation function
3. Estimate the power consumption
Database
Power consumption information on past jobs

At job submission

1. The user submits a job.
2. The job manager function accepts the submission of this job.
3. The job manager function uses a hook of the job operation management function to hand over information on the submitted job to the job power estimate function.
   Based on past job information saved in the job power database, the job power estimate function estimates the power consumption of the submitted job. Then, it uses a hook to hand over the estimated value to the job manager function.
4. The job manager function requests the job scheduler function to schedule the job.
5. The job scheduler function schedules this job based on the power consumption estimate for the job, without exceeding the preset number of custom resources (see the description below).

Figure 2.49 Behavior of the power cap scheduling function at the end of a job with power as a resource (At job end)


Job operation management function
Job resource management function
1. Notification of job termination
Job manager function
2. End of acception
Hook
Power consumption information
Power management function
Job power estimation function
3. Storing the power consumption information
Database
Power consumption information

At job end

1. The job resource management function measures the power consumption of the executed job when the job ends. This function hands over the measured power consumption value (power consumption information on the job) to the job manager function.
2. Upon accepting the end of the job, the job manager function uses a hook to hand over the power consumption information on the job to the job power estimate function.
3. The job power estimate function saves the power consumption information on this job in the job power database of the job power estimate function. The saved information will be used to estimate the power consumption of the next job.

Information

As described above, hooks are configured to call the power management function to estimate the power consumption of jobs. However, you can change them to call an original power estimate module created by the administrator.

The job manager exit function, which is one of the hooks, is used for this purpose. The job manager exit function works by making the job operation management function call a function (exit function) with a specific name at a specific timing during job submission or execution.

For power consumption estimates of a job, the administrator creates the following exit function and incorporates it into the job operation management function. The exit function is called at job submission or end.

At job submission: Job manager exit function pjmx_quejob()

1. Acquire information on the submitted job by using the job information acquisition function provided by the job manager exit function.
2. Hand over the acquired job information to the power estimate module created by the administrator.
3. The power estimate module estimates power consumption based on the information on the submitted job and the accumulated power consumption information on past jobs in the database.
4. Hand over the estimated power consumption to the job manager function by using the job information setting function pjmx_setinfo_power_estimation() provided by the job manager exit function.

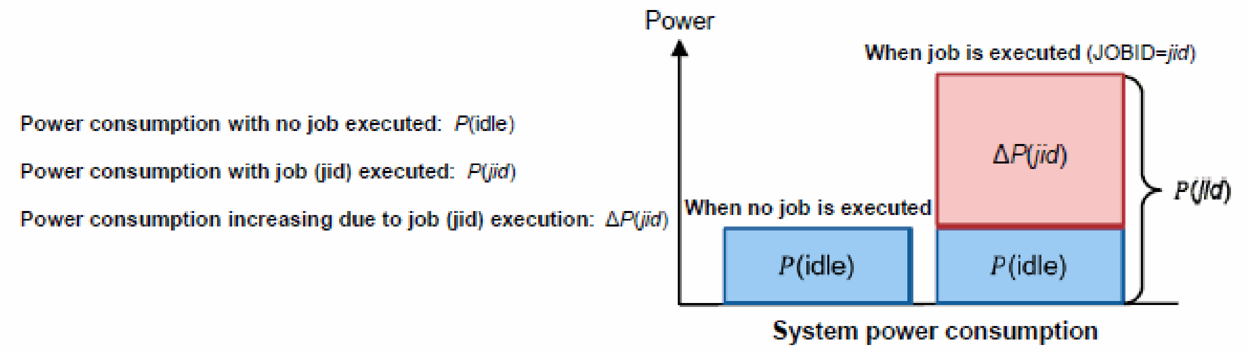
At job end: Job manager exit function pjmx_endjob()

1. Acquire the job information and power consumption information handed over from the job manager function. Use the job information acquisition function pjmx_getinfo_power_consumption() to acquire the power consumption information.
2. Save the acquired job information and power consumption information in the database for subsequent use in estimating the power consumption of a job.

For details on the job manager exit function (exit function and job information acquisition function), see "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

The scheduling by the power cap scheduling function takes into consideration the compute node power consumption increasing due to job execution.

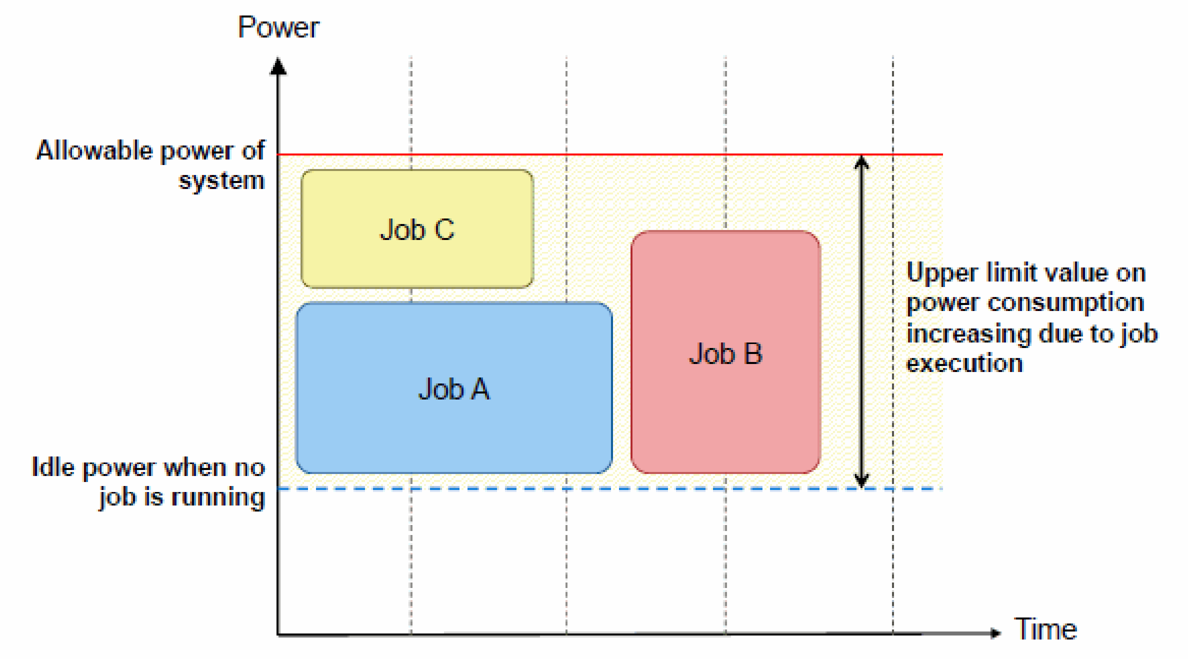
Figure 2.50 Compute node power consumption increasing due to job execution


Power
When job is executed (JOBID=jid)
Power consumption with no job executed: P(idle)
Power consumption with job (jid) executed: P(jid)
Power consumption increasing due to job (jid) execution: ΔP(jid)
ΔP(jid)
P(jid)
When no job is executed
P(idle)
P(idle)
System power consumption

Therefore, it is necessary to set the "upper limit value on power consumption increasing due to job execution" for the values where system power consumption has been defined as a custom resource. The idle power when no job is running is subtracted from the allowable power of the system to calculate the upper limit value on power consumption increasing due to job execution. For details on how to set this upper limit value, see "3.5.1.6 Custom resource settings."

The following figure shows job scheduling by the power cap scheduling function.

Figure 2.51 Job scheduling by the power cap scheduling function


Power
Allowable power of system
Job C
Job A
Job B
Upper limit value on power consumption increasing due to job execution
Idle power when no job is running
Time

Jobs A and C require less power consumption, and job B requires greater power consumption. The power cap scheduling function schedules jobs A and C to be executed first, and job B to be executed after them. In this way, it controls power consumption so that it does not exceed the upper limit on power consumption.

Note

The power cap scheduling function uses the power consumption estimates of jobs to schedule the jobs. Therefore, if an estimate is wrong, the upper limit on power consumption may be exceeded. In such cases, the administrator needs to stop the job manually. Check system power consumption with the pasyspwr command. For details on how to use the pasyspwr command, see Chapter 4 in "Job Operation Software Administrator's Guide for Power Management."

2.5.6 Job scheduler exit function

The job scheduler function has an exit function for calling any process prepared by the administrator. The function is called at a specific timing during the execution of each job. The exit function allows the job operation administrator to configure a process for each resource unit or resource group.
Using the job scheduler exit function, the administrator can implement ticket control that, for example, permits job submission and execution within the budget allocated to each job submitter.

See

The job scheduler exit function is one of the mechanisms called "hooks" in the job operation management function. For details, see "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

2.6 Job Resource Management Function

The job resource management function ensures that resources are not repeatedly allocated to different jobs and resource shortages do not occur, by exclusively allocating computer resources such as CPUs and memory to jobs. The function manages the usage amounts of computer resources, such as CPUs and memory, contained in the system.

According to instructions from the job manager function, the job resource management function controls the start of jobs and allocates and releases the resources required by jobs.

2.6.1 Job Resource Management

The job resource management function runs on the compute cluster management node to centrally manage the total amount of job resources on compute clusters.

The managed job resources are as follows.

CPU resources

All the cores in the CPU chips installed on the compute nodes are managed as CPU resources for jobs. The CPU resource management does not consider chips. It manages only the number of cores.

Memory resources

Normally, the job management function manages 90% of the amount of mounted memory on the compute nodes as the memory resources for jobs.
You can change the percentage of the amount of memory for jobs to any value from 1 to 100%. Set this percentage of the memory amount in the parsc.conf file or pmrsc.conf file. For details on parsc.conf file settings, see "3.4.3 Settings for job resource management function in a cluster (parsc.conf file)". For details on pmrsc.conf file settings, see "3.5.2 Job resource management function settings in a resource unit (pmrsc.conf file)".

For cases where a job requires excessive memory resources, you can select between a memory allocation failure for the job and forcible end of the job. Make this setting in the parsc.conf file.

Note

If you set the percentage of the amount of memory for jobs to a value exceeding 90%, the likelihood of an OS hang-up will increase.

2.6.2 Job Resource management exit function

The job resource management function makes it possible to perform unique processing (a script) specified by the job operation administrator, for example, before the allocation of job resources or after the release of job resources. This is called the resource management exit function. Using the job resource manager exit function, the administrator can execute an original process with an awareness of the resource allocation and release timing before and after the execution of a single job.

See

The job resource manager exit function is one of the mechanisms called "hooks" in the job operation management function. For details, see "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

2.6.3 Periodic collection of job statistical information

The Job Operation Software allows the administrator to periodically collect and accumulate job statistical information of running jobs as time-series information to find out the job operation status and performance in the system.

The collected information is output to the following file on the compute cluster management node.

```
/var/opt/FJSVtcs/prm/jobrscusage
```

Note

The file system requires sufficient free space because a lot of information is output. For details on how to periodically back up (rotate) the jobrscusage file and guidelines on the file size, see "3.6 Setting Log Rotation."

The output format of the information is the CSV format shown below.

```
Time,JOBID,BulkNumber,StepNumber,RetryCount,JobSubmissionTime,InformationTag,InformationBody
```

Table 2.11 Job statistical information periodically collected by the job resource manager

Item	Description
Time	Acquisition time of information: *YYYYMMDD.hhmmss* *YYYY*: year, *MM*: month, *DD*: day, *hh*: hour, *mm*: minute, *ss*: second
JOBID	Job ID
Bulk Number	Bulk number For jobs other than bulk jobs, this item is blank.
Step Number	Step number For jobs other than step jobs, this item is blank.
Retry count	Job retry count
Job submission time	Submission time of the job: *YYYYMMDD.hhmmss*
Information tag	JIS: Indicates a snapshot of information by job. JNS: Indicates a snapshot of information by node used by 1 job.
Information body	Information is output in the order shown in the following table.

Table 2.12 Job statistical information periodically collected by the job resource manager (information body)

Information	Corresponding information name displayed by pjstat -s/-S (*1)
Node ID	NODE ID
Virtual node ID (*2)	VNODE ID
Number of used CPUs	CPU NUM (USE)
CPU time used by user	USER CPU TIME (USE)
CPU time used by system	SYSTEM CPU TIME (USE)
Maximum memory usage	MAX MEMORY SIZE (USE)
Number of Fujitsu profiler use times	FJ PROFILER
Number of starts of a program using the sector cache	SECTOR CACHE
Number of starts of a program using the inter-core hardware barrier	INTRA NODE BARRIER
Power information acquisition status	POWER CONSUMPTION STATE
Average node power consumption (Estimate)	AVG POWER CONSUMPTION OF NODE (IDEAL)
Maximum node power consumption (Estimate)	MAX POWER CONSUMPTION OF NODE (IDEAL)
Minimum node power consumption (Estimate)	MIN POWER CONSUMPTION OF NODE (IDEAL)
Node power consumption (Estimate)	ENERGY CONSUMPTION OF NODE (IDEAL)
Average node power consumption (actual measurement)	AVG POWER CONSUMPTION OF NODE (MEASURED)
Maximum node power consumption (actual measurement)	MAX POWER CONSUMPTION OF NODE (MEASURED)
Minimum node power consumption (actual measurement)	MIN POWER CONSUMPTION OF NODE (MEASURED)
Node power consumption (actual measurement)	ENERGY CONSUMPTION OF NODE (MEASURED)
Power knob usage information	UTILIZATION INFO OF POWER API
Number of processes (*2)	PROC NUM
Number of execution cycles (*2)	PERF COUNT 1
Number of floating-point instruction operations 1 (*2)	PERF COUNT 2

Information	Corresponding information name displayed by pjstat -s/-S (*1)
Number of floating-point instruction operations 2 (*2)	PERF COUNT 3
Number of memory read requests (*2)	PERF COUNT 4
Number of memory write requests (*2)	PERF COUNT 5
Sleep cycle (*2)	PERF COUNT 6
Reserved item for future expansion (*2)	PERF COUNT 7
Reserved item for future expansion (*2)	PERF COUNT 8
Reserved item for future expansion (*2)	PERF COUNT 9

(*1) The output information corresponds to the information items displayed by the -s/-S option of the pjstat command.

(*2) If the information tag is JIS, this field is blank.

See

Settings for periodically collecting job statistical information are set in the parsc.conf ("3.4.3 Settings for job resource management function in a cluster (parsc.conf file)") or pmrsc.conf file ("3.5.2 Job resource management function settings in a resource unit (pmrsc.conf file)".)

2.7 Parallel Execution Environment

In the parallel execution environment, in accordance with requests issued by the process parallel processing program, parallel processes in units called tasks are created and controlled. The number of parallel processes that can be created at one time is restricted to 128, which is the maximum number of tasks that can be created for one job. For this reason, the number of MPI programs (mpiexec command) that can be simultaneously executed is limited to 128. Similarly, the number of sequential programs (pjexe command) that can be simultaneously executed on multiple virtual nodes is limited to 128.

2.8 Job Execution Environment Customization Function

The Job Operation Software has the job execution environment customization function, which switches the software environment for executing job programs (job execution environment) according to user specifications. With this function, users can select a suitable environment for executing their own jobs. From the job execution environments deployed on the system, they can use a specialized environment for the jobs. They can also use execution environments prepared by the users themselves.

See

For details on the mechanism and operation of the job execution environment and how to submit a job using the job execution environment, see "Job Operation Software End-user's Guide."

For details on how to build and configure a job execution environment, see "3.5.7 Configuring a Job Execution Environment."

2.9 Command API

The user interface required for job operations varies depending on the operating system. To support users creating commands that have the user interfaces they want, the job operation management function provides interfaces for calling functions (job operation and information acquisition) equivalent to these commands from programs. These interfaces are called command APIs (Application Programming Interfaces).

The command APIs provide the following functions.

Table 2.13 Functions provided by the command APIs

Target user	Classification	API type
End user and administrator	Job operation API	Job submission (equivalent to pjsub command)
		Job deletion (equivalent to pjdel command)
		Job hold (equivalent to pjhold command)
		Job release (equivalent to pjrls command)
		Signal sending to job (equivalent to pjsig command)
		Job end wait (equivalent to pjwait command)
		Job parameter change (equivalent to pjalter command)
	Information acquisition API	Job information acquisition (equivalent to -s/-S option in pjstat and pjstat commands)
		Resource information acquisition (equivalent to --rsc option in pjstat command)
		Limit value information acquisition (equivalent to --limit option in pjstat command)
		Job ACL information acquisition (equivalent to pjacl command)
		Resource status acquisition (equivalent to pjshowrsc command)
Administrator	System operation API	Job submission and execution permission change (equivalent to --set-rsc-ug and --show-rsc-u options in pmpjmopt command)
	Job operation API	Job parameter change (equivalent to pmalter command)

For details on the command APIs, see "Job Operation Software API user's Guide for Command API."

2.10 Job Information Notification API

The job manager function of the job operation management function provides an interface for job-related information notification of the programs used for processing specific to job operations. Examples include charge processing and job trace processing created by the job operation administrator. The programs are notified at the timing of job state transition. This interface is called the job information notification API.

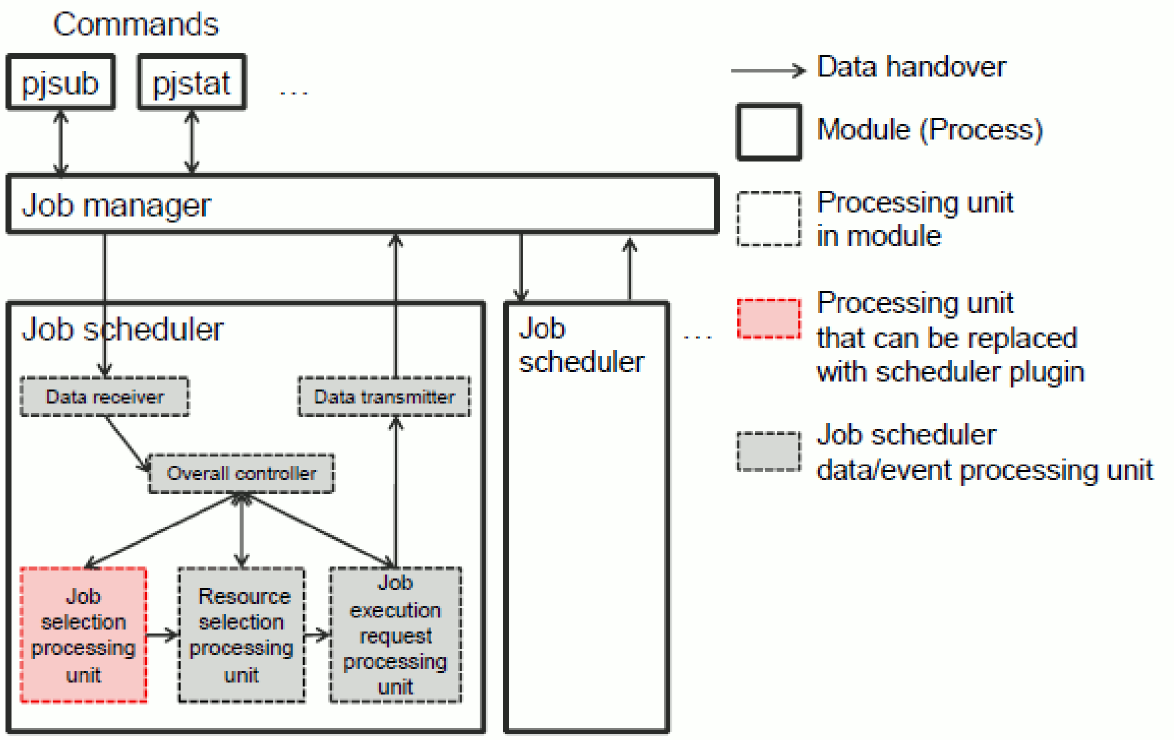
For details on the job information notification API below, see "Job Operation Software API user's Guide for Job Information Notification API."

2.11 Scheduler Plugin Function

The Job Operation Software has a function that incorporates an original scheduling algorithm created by the job operation administrator into the job scheduler to replace the scheduling algorithm of the job operation management function. This function is called the scheduler plugin function. Using the scheduler plugin function, you can apply the optimal scheduling algorithm to job operations.

The incorporation of an original scheduling algorithm is achieved by replacing processing units composing the job scheduler with a module (shared library) created by the job operation administrator. The following figure shows the internal structure of the job scheduler and the relationship with related (job manager and command) modules.

Figure 2.52 Internal structure of the job scheduler and areas that can be replaced with a plugin


Commands
pjsub
pjstat
...
Job manager
Job scheduler
Data receiver
Data transmitter
Overall controller
Job selection processing unit
Resource selection processing unit
Job execution request processing unit
Job scheduler
...
Data handover
Module (Process)
Processing unit in module
Processing unit that can be replaced with scheduler plugin
Job scheduler data/event processing unit

The job scheduler is a module that allocates resources to jobs in descending order of priority and determines the job execution order (planned execution start times of jobs). The scheduler operates together with the job manager, which has an interface with commands. The job scheduler consists of the following processing units.

Data receiver

This processing unit receives data sent from the job manager. The types of received data include job submission, node state change, and system setting change data. It hands over the received data to the overall controller.

Overall controller

This processing unit controls the start and end of job scheduling. When triggered by data received from other processing units, this unit interrupts scheduling in progress and starts new scheduling. When scheduling in progress is interrupted, each processing unit takes back jobs being processed.

Job selection processing unit

This processing unit prioritizes submitted jobs according to the job selection policy and sorts them in descending order of priority. It hands over jobs to the resource selection processing unit in descending order of priority. The job scheduler function is made from this processing unit working together with the resource selection processing unit.

Resource processing unit

This processing unit allocates job-dedicated resources (compute nodes, custom resources, etc.) to jobs in descending order of priority, determined by the job selection processing unit, with consideration of the number of requested nodes, elapsed time limits, etc. Then, it determines the planned execution start times of the jobs. It hands over the jobs for the selected resources to the job execution request processing unit. The job scheduler function is made from this processing unit working together with the job selection processing unit.

Job execution request processing unit

This processing unit requests the job manager to execute a job when the planned execution start time of the job arrives. The time is determined by the resource selection processing unit.

The scheduler plugin function provides an API to replace the job selection processing units, which are in these processing units, with an original scheduling algorithm. This API called the scheduler API.

The job operation administrator uses this API to implement an original scheduling algorithm as a shared library and incorporate the library into the job scheduler. This shared library is called "plugin library" or "scheduler plugin."

For details on the scheduler plugin function, see "Job Operation Software API user's Guide for Scheduler API."

Chapter 3 Job Operation Management Function Settings

This chapter describes the settings of the job operation management function.

The following tables list the configuration files and related commands for the job operation management function. The tables are divided into settings by the cluster administrator and settings by the job operation administrator. The cluster administrator manages job operations for a cluster, and the job operation administrator manages job operations for resource units.

Table 3.1 Files set by the cluster administrator and related commands

Setting contents	Related command	Configuration file name	Node and path where created
Work environment definitions relating to the job operations of an entire cluster	papjmadm command	papjm.conf	System management node /etc/opt/FJSVtcs/
Work environment definitions such as computer resource allocation amounts for the jobs of an entire cluster	parscadm command	parsc.conf	System management node /etc/opt/FJSVtcs/
Job ACL function settings of an entire cluster	pmjacladm command	Optional	System management node or compute cluster management node
Job statistical information settings of an entire cluster	papjmstatsadm command	papjmstats.conf	System management node /etc/opt/FJSVtcs/
Advanced job scheduling settings of an entire cluster	None	pjs.conf	Active compute cluster management node /var/opt/FJSVtcs/shared_disk/pjm/.private/

Table 3.2 Files set by the job operation administrator and related commands

Setting contents	Related command	Configuration file name	Node and path where created
Work environment definitions relating to job operations by resource unit	pmpjmadm command	pmpjm.conf	System management node /etc/opt/FJSVpnavi/Rscunit.d/resource-unit-name/
Work environment definitions such as computer resource allocation amounts for jobs by resource unit	pmrscadm command	pmrsc.conf	System management node /etc/opt/FJSVpnavi/Rscunit.d/resource-unit-name/
Job ACL function settings by resource unit	pmjacladm command	Optional	System management node or compute cluster management node

Note

The job operation management function needs to be set only during initial installation of the Job Operation Software and when changing operations. The job resource management function needs to be set too when adding a new compute node because the function has settings (parsc.conf and pmrsc.conf files) configured for each compute node. For details on how to have only particular compute nodes reflect the job resource management function settings, see "Information" in "3.4.3.2 Reflecting and referencing the parsc.conf file" and "3.5.2.2 Reflecting and referencing the pmrsc.conf file."

3.1 Checking the System Configuration

To set the job operation management function, the administrator needs to know the system configuration. Such information includes the cluster name, resource unit names, and the number of nodes.

System configuration information relating to job operations is managed for each resource unit. You can check that information by using the pashowclst command.

The following operation examples check the configuration information by using the pashowclst command.

Displaying information on a specific cluster

To check cluster information, specify a cluster name in the -c option of the pashowclst command.

```
[System management node]
# pashowclst -c cluster1 --rscunit
[ CLST: cluster1 ]
RSCUNIT  RUNNING  STOPPED  ERROR  DISABLE
unit1    393      10       2      3
unit2    816      0        0      0
```

You can omit specifying the -c option by specifying the cluster name in the environment variable PXMYCLST.

Displaying information on a specific resource unit

To check resource unit information, specify a resource unit name in the --rscunit option of the pashowclst command.

```
[System management node]
# pashowclst -c cluster1 --rscunit unit1
[ CLST: cluster1 ]
[ RSCUNIT: unit1 ]
RSCUNIT  RUNNING  STOPPED  ERROR  DISABLE
unit1    393      10       2      3
```

Displaying a list by node type

To display the number of nodes by node type, specify the -l option in the pashowclst command.

```
[System management node]
# pashowclst -c cluster1 --rscunit -l
[ CLST: cluster1 ]
RSCUNIT           TOTAL   SMM  CCM  LN   CCS  BIO    SIO    GIO    CN      SCM  MGS  MDS  OSS
unit1             408     -    -    -    -    8      12     4      384     -    -    -    -
unit2             816     -    -    -    -    16     24     8      768     -    -    -    -
# pashowclst -c cluster1 --rscunit unit1 -l
[ CLST: cluster1 ]
[ RSCUNIT: unit1 ]
RSCUNIT           TOTAL   SMM  CCM  LN   CCS  BIO    SIO    GIO    CN      SCM  MGS  MDS  OSS
unit1             408     -    -    -    -    8      12     4      384     -    -    -    -
```

See

For details on using the pashowclst command to check the system configuration, also see "Displaying System Configuration Information" in "Chapter 3 Details of the System Management Function" in "Job Operation Software Administrator's Guide for System Management."

3.2 How to Code a Configuration File

The configuration files for the job operation management function have the following basic structure of coding.

```
section name {
    item1 = value
    item2 = value
    subsection name {
        item3 = value
        item4 = value
    }
}
```

The section name is a keyword indicating what is covered by the structure definition, which is the part enclosed by curly brackets ("{ }"). The name is determined according to the definition contents. Some defined items contain subsections in their sections.

3.3 MariaDB Settings

The job manager function has an internal database for managing jobs. MariaDB must be configured in preparation for this database. Perform the following work during installation only once.

See

MariaDB must be installed on the compute cluster management node in advance.
For the procedure to install MariaDB, see "Performing MariaDB-related Work for Job Operations" in "Chapter 2 New System Installation" in "Job Operation Software Setup Guide."

1. Work on the active compute cluster management node

   Confirm in advance that /var/opt/FJSVtcs/shared_disk is mounted. Then, perform the following work.

   1. Creating a database

      Create a database for the job manager function. The database name is pjmdb.

      ```
      # mysql -u root -p
      Enter password: password of root
      MariaDB [(none)]> create database pjmdb;
      MariaDB [(none)]> exit
      Bye
      # mysql -u root -p pjmdb < /var/opt/FJSVtcs/shared_disk/pjm/.private/pjmdb.sql
      Enter password: password of root
      ```

   2. Creating a user for accessing the database

      Create a user for accessing the pjmdb database.
      In the following example, the user name is pjmdbadmin and the password is password.

      ```
      # mysql -u root -p
      Enter password: password of root
      MariaDB [(none)]> grant all on pjmdb.* to pjmdbadmin identified by 'password';
      MariaDB [(none)]> exit
      Bye
      ```

2. Work on the nodes used to execute the pjstat command (system management, compute cluster management, and login nodes)

   Create a new /root/.odbc.ini file as the ODBC environment settings, and code the following contents.

   ```
   # vi /root/.odbc.ini
   [pjmdb]
   Driver=ODBC_Driver   <- Specify MySQL for RHEL 7 and MariaDB for RHEL 8
   Description=pjm DB.
   SERVER=IP_Address    <- Representative IP address of compute cluster management node
   USER=UserName        <- Name of user created in step 1-2 (pjmdbadmin in above example)
   Password=password    <- Password of user created in step 1-2 (password in above example)
   Database=pjmdb       <- Job manager database name
   OPTION=1048576
   ```

Note

Since passwords are written in plain text in the /root/.odbc.ini file, set the permissions of the file to the owner root, the group root, and mode 0600 to prevent unauthorized users from viewing the file.

3.4 Settings for the Cluster Administrator

This section describes the contents of settings by the cluster administrator, among the settings of the job operation management function.

The cluster administrator handles the following settings:

- Settings for job operations in a cluster
- Settings for job statistical information in a cluster
- Settings for job resource management in a cluster
- Job ACL function settings in a cluster
- Advanced job scheduling settings

Information

- The setting and operation work of the job operation management function requires cluster administrator privileges or higher.
- The examples in this section contain commands specifying a cluster name in the -c option for operation on the system management node. Write the name in this way. However, if the environment variable PXMYCLST specifies the cluster name, you can omit specification of the cluster name in the -c option during actual operation.

3.4.1 Settings for job operation management function in a cluster (papjm.conf file)

The cluster administrator can make the following settings in the papjm.conf file (system management node: /etc/opt/FJSVtcs/papjm.conf) by cluster:

- Job manager function settings
  - Log output level setting for the job manager function
  - Start and stop settings for job statistical information output
  - Setting of whether to save information on jobs that were submitted and rejected
  - Setting of the job state storage period
  - Ended job script file storage setting
  - Setting of jobs to be limited on the total number of CPU cores used per job multiplied by the total CPU core time used per job.
  - Setting of whether to allow changes of the elapsed time limit value of a running job
  - Setting that suppresses the output of the job information to the history information which is output with the -H option of the pjstat command for QUEUED jobs when they are deleted.
  - Setting that suppresses the output of the job statistics file (.stats file) for QUEUED jobs when they are deleted.
- Default value settings for resource units
  - Log output level setting for the job scheduler function
  - Setting for enabling/disabling the backfill function
  - Setting the target range of the backfill function (executing lower-priority jobs ahead of higher-priority jobs)
  - Buffer time setting for the job execution interval

    - Rescheduling grace time setting
    - Setting of the grace time before the forced termination of a job that continued running after exceeding the minimum elapsed time limit value
    - Interval setting for job resource map creation
    - Scheduling period setting
    - Setting of a dynamic scheduling period
    - Settings of whether job models (normal, step, and bulk job) automatic re-execution
    - E-mail transfer function setting
    - Setting of whether to subtract the fair share value when job execution starts
    - Setting of the fair share recovery value
    - Setting the jobs to be limited on the number of CPU cores used simultaneously
    - Setting for guarantee of planned execution start times
    - Interpretation of the submission time when scheduling a held job that was released
    - Setting of the maximum number of jobs to schedule
    - Setting of the method for limiting the jobs to schedule
- Job selection policy settings
    - Setting of the job selection policy name
    - Priority setting for the job submission time
    - Priority setting for the number of required nodes
    - Group priority setting
    - User priority setting
    - Priority setting for the users in a group
    - Job priority setting
    - Priority setting for the fair share value for groups
    - Priority setting for the fair share value for the users in a group
    - Priority setting of the fair share value for users
    - Priority setting for the elapsed time limit value
    - Priority setting for the node time product
    - Priority setting for the job evaluation point
    - Priority setting for execution start time specification
    - Priority setting for interactive job specification

The following example shows settings of the papjm.conf file.

```
[System management node]
# cat /etc/opt/FJSVtcs/papjm.conf
Cluster {
    ClusterName = clusterA
    LogLevel = 1
    JstiOutData = yes
    JstiRejectData = no
    KeepJobData = 10
    SaveScript = off
    RunJobAlterElapse = off
    PjdelNoHistory = no
    PjdelNoStats = no
    ResourceUnit {
        LogLevel = 1
        Backfill = yes
        BackfillTarget = rscgrp
        DecidedGap = 00:01:00
        Grace = 00:02:00
        CreateRscMap = "01:00:00, 00:10:00"
        CreateRscMap = "24:00:00, 01:00:00"
        CreateRscMap = "*, 24:00:00"
        SchedulePeriod = 25:00:00
        DynamicSchedulePeriod = 2,24:00:00
        MailSend = yes
        Fairshare = off
        FshareRecoveryValue = 236
        StartTimeGuarantee = on
        HoldAcceptDate = release
        JobSchedulingTargetLimit = 10000
        JobSchedulingTargetMode = jobselectpolicy
    }
    # default Job Select Policy
    JobSelectPolicy {
        job_aprio =            2,desc
#        group_fairshare =      3,desc
#        user_fairshare =       4,desc
#        group_prio =           5,desc
#        usr_in_grp_prio =      6,desc
#        usr_in_grp_fairshare = 7,desc
#        interact =             8,true
#        node_times_elapse =    9,asc
#        node =                 10,asc
#        elapse_limit =         11,asc
#        user_prio =            12,desc
#        job_prio =             13,desc
#        at =                   14,true
        fcfs =                  256,asc
    }
# Job Select Policy 1
    JobSelectPolicy {
        name = policy1
        group_fairshare =       3,desc
        fcfs =                  256,asc
    }
    # Job Select Policy 2
    JobSelectPolicy {
        name = policy2
        fcfs =                  256,asc
        job_aprio = 2,desc
    }
# Job Select Policy 3
    JobSelectPolicy {
        name = policy3
        group_fairshare =       3,desc
        group_prio =            4,desc
        usr_in_grp_fairshare =  5,desc
        fcfs =                  256,asc
    }
# Job Select Policy 4
    JobSelectPolicy {
        name = policy4
        job_epoint =            1,desc
        fcfs =                  255,asc
    }
    JobEvaluation {
        name = jobeval1
        waittime = 1,1800@1:*@2
    }
}
```

For details on the settings, see the man page for the papjm.conf file.

Information

The papjm.conf file is installed on the system management node. If you want to change the default settings, edit the file.

Note

- You cannot change settings related to job operations simply by creating or editing the papjm.conf file. As described below, the papjmadm command reflects the contents of the papjm.conf file in job operations.
- The maximum length of a single line in the papjm.conf file is 511 characters.

3.4.1.1 Job manager function settings

The following table lists the items set in the Cluster section.

Table 3.3 Setting items of the job manager function (Cluster section)

Item name	Definition contents	Specifiable value	Default value
ClusterName	Cluster name	Character string with 1 to 63 characters, consisting of single-byte alphanumeric characters, hyphen, and underscore	Not omissible
LogLevel	Log output level (Normally, 1 is used for operation.) 1: Normal log level 2: Detailed level for debugging 3: More detailed level for debugging	1-3	1
JstiOutData	Specifying job statistical information output	yes/no yes: Output no: Do not output	yes
JstiRejectData	Setting of whether to save information on jobs that were submitted and rejected	yes/no yes: Save no: Do not save	no
KeepJobData	Job information retention period (unit: day) for jobs in the EXIT, CANCEL, or REJECT state	0-365	10
SaveScript	Save ended job script files	on/off on: Save off: Do not save	off
TotalCoresUpper	Setting the target range of the job that limits the following job ACL setting item. The target range of the job that limits the following items specify "only Virtual node allocated job (vnode)" or "all jobs (all)." - joblimit total-cores - joblimit interact-total-cores	vnode: virtual node allocated jobs. all: all jobs (*)	all

Item name	Definition contents	Specifiable value	Default value
	- joblimit total-cores-elapse - joblimit interact-total-cores-elapse		
RunJobAlterElapse	Setting of whether to allow changes of the elapsed time limit value of a running job	on/off on: Allow changes off: Do not allow changes	off
PjdelNoHistory	Setting that suppresses the output of the job information to the history information which is output with the -H option of the pjstat command for QUEUED jobs when they are deleted.	yes/no yes : Suppress no : Do not suppress. However, it is suppressed if the --no-history option of the pjdel command is specified.	no
PjdelNoStats	Setting that suppresses the output of the job statistics file (.stats file) for QUEUED jobs when they are deleted.	yes/no yes : Suppress no : Do not suppress. However, it is suppressed if the --no-stats option of the pjdel command is specified.	no

(*) The total number of CPU cores used for node allocated jobs is a value obtained by multiplying the number of CPU cores on a node by the number of request nodes.

3.4.1.2 Default value settings for resource units

The following table lists the items set in the ResourceUnit section.

Table 3.4 Default value setting items for resource units

Item name	Definition contents	Specifiable value	Default value
LogLevel	Log output level (Normally, 1 is used for operation.) 1: Normal log level 2: Detailed level for debugging 3: More detailed level for debugging	1-3	1
Backfill	Backfill function	yes/no yes: Enabled no: Disabled	yes
BackfillTarget	Setting the target range of the backfill function (execution of lower-priority jobs ahead of higher-priority jobs)	rscunit: All jobs in a resource unit rscgrp: All jobs in the resource group where the job belongs	rscgrp
DecidedGap	Buffer time setting	Hour:minute:second 00:00:00 to 99:59:59	00:01:00
Grace	Grace time setting	Hour:minute:second 00:00:00 to 99:59:59	00:02:00
AdaptiveElapsedTimeJobTerminateGrace	Setting of the grace time before the forced termination of a job Jobs that continue running after exceeding the minimum elapsed time limit value receive the SIGTERM signal, and then are forcibly terminated after a certain time has elapsed. The signal is sent at the start of execution of a subsequent job, This item sets the period	Hour:minute:second 00:00:00 to 99:59:59	00:00:10

Item name	Definition contents	Specifiable value	Default value
	from the signal sending to the forced termination.		
CreateRscMap	Time zone and time interval setting for job resource map creation The format is "time_zone, time_interval". (Resource map making interval can be specified in several numbers. If it is specified multiple, it must begin from the closest time zone apart from now.)	time_zone: Hour:minute:second 00:01:00 to 9999:59:59,* (* will be set in the last line) time_interval: Hour:minute:second 00:01:00 to 9999:59:59	(It is set as the following 4 lines) 01:00:00,00:10:00 24:00:00,00:30:00 240:00:00,01:00:00 *,24:00:00
SchedulePeriod	Scheduling period setting The time in the future which is later than current time added with this period will not be assigned.	Hour:minute:second 00:01:00 to 9999:59:59	240:00:00
DynamicSchedulePeriod	Setting of a dynamic scheduling period The job scheduling period is determined based on the maximum elapsed time limit value of the submitted job, not the fixed value shown in the SchedulePeriod item. (*1)	multiplicative factor,minimum scheduling period Multiplicative factor: 1 to 2147484347 Minimum scheduling period: Hour:minute:second (*1)	2,24:00:00
RestartNormal	Setting of whether a normal job automatic re-execution	yes: Automatic re-execution is enabled. no: Automatic re-execution is disabled.	yes
RestartStep	Setting of whether a step job automatic re-execution	yes: Automatic re-execution is enabled. no: Automatic re-execution is disabled.	yes
RestartBulk	Setting of whether a bulk job automatic re-execution	yes: Automatic re-execution is enabled. no: Automatic re-execution is disabled.	no
MailSend	E-mail transfer setting All related mails can be sent to jobs in the current resource unit is set.	yes: Send no: Do not send	yes
Fairshare	Setting of whether to subtract the fair share value when job execution starts	on: Subtract off: Do not subtract	off
FshareRecoveryValue	Fair share recovery value Job using resource (the number of nodes when referring to FX servers, or the number of CPU cores when referring to PRIMERGY servers) which is treated as fair share * elapse time is recovery quantity in the period. In the operation, after assuming scale job has been executed, the period which lasts until execution priority having completely recovered will be took as standard. This item is valid for all the fair share sets in the resource unit range.	0-18446744073709551615	236 (*2)

Item name	Definition contents	Specifiable value	Default value
UseCoreLimit	Setting of the jobs subject to the limit on the number of concurrently used CPU cores Specify the jobs to be limited by the following items in the job ACL: ru-use-core rg-use-core ru-interact-use-core rg-interact-use-core	vnode: Targets only a virtual node allocated job. all: Targets all jobs. (*3)	vnode
StartTimeGuarantee	Setting for guarantee of planned execution start times	on: planned execution start times are guaranteed. off: planned execution start times may change due to rescheduling.	off
HoldAcceptDate	How to interpret the submission time in job scheduling when a job is released from hold status.	release: Schedule the job based on when it was released. accept: Schedule the job based on the time it was submitted.	release
JobSchedulingTargetLimit	Setting of the maximum number of jobs to schedule However, regardless of this setting, all interactive jobs have to be scheduled. In step and bulk jobs, the number of sub jobs is counted as the number of jobs.	0: Do not limit 1 to 2147483647: Maximum number of jobs to schedule	10000
JobSchedulingTargetMode	Method for limiting the jobs to schedule	aprio: Sort the jobs in a resource unit by priority and by submission time. Select the target jobs in ascending order. jobselectpolicy: Sort jobs by using the JobSelectPolicy specified for each resource group/resource unit. (*4)	jobselectpolicy

(*1) The scheduling period is calculated with the following formula. Jobs are not scheduled at or after the time at "current time + scheduling period."

```
Scheduling period = maximum elapsed time limit value of submitted job x multiplicative factor
```

If the calculated scheduling period is shorter than the minimum scheduling period, the scheduling period is the minimum scheduling period. If the calculated scheduling period is longer than the value specified by the SchedulePeriod item, the scheduling period is the value of the SchedulePeriod item. If you do not want the scheduling period to change dynamically, set the minimum scheduling period to a greater value than the value of the SchedulePeriod item.
If the minimum scheduling period is omitted as shown below, the applied minimum scheduling period is 24:00:00.

```
DynamicSchedulePeriod=multiplicative factor
```

If the multiplicative factor is omitted as shown below, the applied multiplicative factor is 2.

```
DynamicSchedulePeriod=,minimum scheduling period
```

(*2) The default value of 236 assumes the following:

- This value assumes that the requested number of nodes for the job is 165888 and the elapsed time limit for the job is 24 hours. Under these circumstances, the fair share value that is reduced at the job start time recovers in a period of one week at the recovery factor of 100.

```
(165888 x 24 hours x 3600 seconds) / (7 days x 24 hours x 3600 seconds) / recovery factor of 100 =
236
```

You can define the recovery factor for each user and group by using the define fshare-recovery setting item of the job ACL function (see "3.4.4.2 Defined items of the job ACL function").

(*3) The number of CPU cores that can be used concurrently by a node allocated job is the number of mounted CPU cores in a single node multiplied by the requested number of nodes.

(*4) The following JobSelectPolicy setting items (see "3.4.1.3 Job selection policy settings") are not used to determine the target jobs:

```
usr_in_grp_prio, job_prio, group_fairshare, user_fairshare, usr_in_grp_fairshare
```

3.4.1.3 Job selection policy settings

You can set the job selection policy for a cluster by using the JobSelectPolicy subsection in the Cluster section.

Table 3.5 Setting items of a job selection policy (JobSelectPolicy subsection)

Item name	Definition contents	Specifiable value	Default value
name	Job selection policy name	Character string with 1 to 15 characters, consisting of single-byte alphanumeric characters, hyphen, and underscore	The subsection is used as the default setting for the job selection policy.
fcfs	Job submission time	*order*[,asc|,desc]|off *order*: (1 to 256) Evaluated in the order of *order* asc: Ascending order desc: Descending order off: Not evaluated	asc
node	Number of required nodes [FX]		asc
group_prio	Group priority		desc
user_prio	User priority		desc
usr_in_grp_prio (*1)	Priority of the users in a group		desc
job_prio (*2)	Job priority		desc
job_aprio	Job priority within a resource unit		desc
rscgrp_prio	Resource group priority		desc
elapse_limit	Elapsed time limit value		asc
node_times_elapse	Node time product [FX]		asc
job_epoint	Job evaluation point		desc
group_fairshare	Fair share value for groups		desc
usr_in_grp_fairshare (*1)	Fair share value for the users in a group		desc
user_fairshare	Fair share value for users		desc
at	Execution start time specification	*order*[,true|,false]|off *order*: (1 to 256) Evaluated in the order of *order* true: has a preference for interact jobs(default) false: has a preference for non-interact jobs off: Not evaluated	true
interact	Interactive job specification		true

(*1)
Be sure to set group_prio as the policy to be evaluated before the evaluation order set in usr_in_grp_prio. For example, if the evaluation order of usr_in_grp_prio is 5, set the evaluation order of group_prio to 4.

Likewise for usr_in_grp_fairshare, set group_prio as the policy to be evaluated before the evaluation order set in usr_in_grp_fairshare. When setting usr_in_grp_prio and usr_in_grp_faishare, be sure to evaluate usr_in_grp_prio and usr_in_grp_faishare consecutively, and be sure to set group_prio as the policy to be evaluated before the previous one. For example, set the evaluation order of usr_in_grp_prio and usr_in_grp_faishare to 5 and 6, and set the evaluation order of group_prio to 4.
For all groups that run the job, set a different value for the group priority within the resource unit (define pri-g) of the job ACL function.

(*2)
Be sure to set user_prio as the policy to be evaluated before the evaluation order set in job_prio. For example, if the evaluation order of job_prio is 5, set the evaluation order of user_prio to 4.
For all users that run the job, set a different value for the user priority within the resource unit (define pri) of the job ACL function.

Information

In the job selection policy, fcfs is not defined, and when the job execution order is not determined in the defined items, the jobs are executed in the order of submission.

As shown below, write a job evaluation definition as the subsection JobEvaluation for determining a job evaluation point (job_epoint).

```
Cluster {
    ...
    JobEvaluation {
        [name = job evaluation definition name]
        waittime = Weight[,Value1@Point1:Value2@Point2:...:ValueN@PointN]
        ...
    }
    ...
}
```

The following table describes details of the coding.

Table 3.6 Format of job evaluation definition coding

Item	Description
JobEvaluation { ... }	This indicates one job evaluation definition. You can write up to 32 job evaluation definitions in the papjm.conf file.
name= *job evaluation definition name*	This is the name assigned to a job evaluation definition. The job evaluation definition name must have a length ranging from 1 to 127 characters. The name must also be unique within the cluster. Only one job evaluation definition without the setting of a job evaluation definition name can be set in the papjm.conf file. This definition is applied as a common definition within the cluster. The papjm.conf file defines the job evaluation definitions having names. A job evaluation definition can be used when its name is specified in the pmpjm.conf file, which is configured for each resource unit.
waittime	This indicates that the point is based on the time (in seconds) from job acceptance to the start of job scheduling or rescheduling.
Weight	You can specify a value from -2147483648 to 2147483647. Normally, specify 1.

Value1@Point1: *Value2@Point2*: ...: *ValueN@PointN*	This defines a point based on the job execution wait time (point definition).
	Write the point definition in the "*value@point*" format. When writing multiple point definitions, separate them with a colon (:). You can specify up to 64 point definitions.
	The point is determined as follows:
	`Minimum value of item <= Value <= Value1   ...  Point1`
	`Value1 < Value <= Value2                   ...  Point2`
	`...`
	`ValueN-1 < Value <= ValueN                 ...  PointN`
	You can specify a value from 1 to 2147483647 as the job execution wait time.
	You can specify a value from 1 to 1024 as a point.

Item	Description
	When writing multiple point definitions, enumerate them in order such that "*value n-1* < *value n*" holds true. The multiple point definitions must each be a unique value. The "*" (asterisk) specified as a value means a value greater than *value n-1*. [Example] 10@1:20@2:*@3 If the value is greater than 20, the point is 3.

The job evaluation point is the sum of individual points multiplied by *Weight*.
In the process of calculating the job evaluation point, the calculated value may be smaller than the lower limit value, -9223372036854775807, or larger than the upper limit value, 9223372036854775807. If so, the job evaluation point is set to the lower limit value or upper limit value, respectively.

The following example shows job evaluation definition coding in the papjm.conf file.

```
# cat /etc/opt/FJSVtcs/papjm.conf
Cluster {
  LogLevel = 1
  ...
  JobSelectPolicy {
    job_epoint  = 1,desc
    fcfs        = 255,asc
  }
  JobEvaluation {
    waittime     = 1
  }
  JobEvaluation {
    name        = jobeval1
    waittime    = 1,1800@1:*@2
  }
}
```

The job evaluation definition with the defined name (jobeval1 in the above example) can be used in the pmpjm.conf file used by the job operation administrator to configure a resource unit. For examples of coding in the pmpjm.conf file, see "3.5.1.4 Job selection policy settings."

3.4.1.4 Settings for the fair share function

To control job execution priorities with fair share values within a single resource unit, configure the papjm.conf file as follows.

1. Configure the fair share function.

   Set the Fairshare item in the papjm.conf file to on so that the fair share value is subtracted when job execution starts.

2. Configure the job section policy.

   To use fair share values as the job selection policy, add policies as follows to the job selection policy JobSelectPolicy in the papjm.conf file.

   - To include user fair share values in priority control, add user_fairshare.

   - To include group fair share values in priority control, add group_fairshare.

   - To include the fair share values of users in a group in priority control, add usr_in_grp_fairshare.

In addition to the above settings, change the setting of the fair share value, initial fair share value, fair share recovery value, or recovery factor as required. For details on the setting method, see "4.2.5 Monitoring and changing a fair share value and initial fair share value ."

3.4.1.5 Reflecting and referencing the papjm.conf file

After setting the papjm.conf file, you need to execute the papjmadm command on the system management node so that the system reflects the setting contents.

```
[System management node]
# papjmadm -c clstname --set
```

The above operation incorporates settings immediately to reflect them in operation. Neither a node restart nor a complete system restart is required. To display the current settings, use the papjmadm command with the --show option.

```
[System management node]
# papjmadm -c clusterA --show
Cluster {
    ClusterName = clusterA
    LogLevel = 1
    JstiOutData = yes
    KeepJobData = 10
    SaveScript = off
    TotalCoresUpper = all
    JstiRejectData = no
    RunJobAlterElapse = off
    PjdelNoHistory = no
    PjdelNoStats = no
    ResourceUnit {
        LogLevel = 1
        Backfill = yes
        DecidedGap = 00:01:00
        Grace = 00:00:10
        CreateRscMap = 01:00:00, 00:10:00
        CreateRscMap = 24:00:00, 00:30:00
        CreateRscMap = 240:00:00, 01:00:00
        CreateRscMap = *, 24:00:00
        SchedulePeriod = 240:00:00
        RestartNormal = yes
        RestartStep = yes
        RestartBulk = no
        MailSend = yes
        Fairshare = off
        FshareRecoveryValue = 236
        UseCoreLimit = vnode
        BackfillTarget = rscgrp
        StartTimeGuarantee = on
        HoldAcceptDate = release
        AdaptiveElapsedTimeJobTerminateGrace = 00:00:10
        DynamicSchedulePeriod = 2,24:00:00
        JobSchedulingTargetLimit = 10000
        JobSchedulingTargetMode = jobselectpolicy
    }
    JobSelectPolicy {
        fcfs = 256,asc
        job_aprio = 2,desc
    }
    JobSelectPolicy {
        name = policy1
        fcfs = 256,asc
        group_fairshare = 3,desc
    }
    JobSelectPolicy {
        name = policy2
        fcfs = 256,asc
        job_aprio = 2,desc
    }
    JobSelectPolicy {
        name = policy3
        fcfs = 256,asc
        group_prio = 4,desc
        group_fairshare = 3,desc
        usr_in_grp_fairshare = 5,desc
    }
    JobSelectPolicy {
        name = policy4
        fcfs = 255,asc
        job_epoint = 1,desc
    }
    JobEvaluation {
        name = jobeval1
        waittime = 1,1800@1:*@2
    }
}
```

3.4.2 Settings for job statistical information in a cluster (papjmstats.conf file)

The cluster administrator can make settings related to the following job statistical information in the papjmstats.conf file (system management node: /etc/opt/FJSVtcs/papjmstats.conf):

- Administrator-defined items in job statistical information
- Definitions of output items in job statistical information
- Path to a job statistical information file

The following example shows settings in the papjmstats.conf file.

```
Cluster {
    Item {    <- Defines ITEM_A, administrator-defined item in job statistical information
        ItemName = ITEM_A
        RecordNameList = JI,JN
    }
    Item {    <- Defines ITEM_B, administrator-defined item in job statistical information
        ItemName = ITEM_B
        RecordNameList = JI
    }
    Record {  <- Defines what is output to job statistical information file
        PATH = /xxxx/xxxxx/jobinfo_acct
        JI {
            ITEM = jid,jnam,elpl,nnuma
            ITEM = elp,mmszu
        }
    }
    File {    <- Defines what is output to .stats file
        JI {
            ITEM = jid,jnam,elpl,nnuma
        }
    }
    Command { <- Defines what is output for display by pjstat command
        JI {
            ITEM = jid,jnam,elpl,nnuma
        }
    }
}
```

Set job statistical information in subsections in the Cluster section. This section describes the setting items in each subsection.

3.4.2.1 Settings of administrator-defined items in job statistical information

The Item subsection defines administrator-defined items in job statistical information.

```
Cluster {
    Item {
        ItemName=Value
    }
}
```

Table 3.7 Job statistical information settings

Item name	Description	Specifiable value	Default value
ItemName	Administrator-defined item name (identifier) This item is used to specify the item name for the set value of an administrator-defined item by the job manager exit function and for a set output item in job statistical information.	Character string with up to 79 characters The available characters are single-byte alphanumeric characters and a hyphen or underscore. The string cannot begin with a hyphen. Names beginning with "CR-" or "pjm-" are reserved words, so they cannot be used as administrator-defined item names. The same name as a job statistical information item name defined by the Job Operation Software can be used for ItemName. In this case, when output items in job statistical information are defined (see "3.4.2.2 Definitions of output items in job statistical information"), "pjm-" must be prefixed to the item name to distinguish the job statistical information item defined by the Job Operation Software from the administrator-defined item. For the names defined by the Job Operation Software, see "Item names" in the man page for pjstatsinfo(7).	Not omissible
ItemNameDisp	Display name for job statistical information that is output to the .stats file or output when the -s/-S option of the pjstat command is specified	Character string with up to 79 characters The available characters are single-byte alphanumeric characters and a hyphen or underscore.	Character string specified for ItemName
RecordNameList	Record type of a defined item A record is the unit of output of items. Records can also be defined by the administrator as described in "3.4.2.2 Definitions of output items in job statistical information." This setting specifies which record includes the item by default.	JI: Job statistical information record JN: Node statistical information record You can specify multiple values by delimiting them with a comma. For node statistical information, information by virtual node is output in cases where jobs have virtual nodes allocated.	JI, JN

Item name	Description	Specifiable value	Default value
DataType	Type of item value	Select a value from "Table 3.8 Specifiable values for DataType" according to the type.	PJMX_DATATYPE_UINT64
DispFormat	Display format for the item value This setting specifies the format for values that are output to the .stats file or output when the -s/-S option in the pjstat command is specified.	Select a value from "Table 3.9 Specifiable values for DispFormat."	Default format according to DataType
DispFormatOption	Character string appended to the displayed value Specify this item to append a unit such as B (bytes) or W (watts). However, the setting is valid only when DispFormat is one of the following: dec, prefix_kilo, prefix_kibi, prefix_mega, prefix_mebi, prefix_giga, prefix_gibi, prefix_tera, prefix_tebi	Character string with up to 15 characters	No unit is appended.

Note

- Multiple definitions of the same item cause an error when the settings are reflected.
- The administrator needs to set values for the administrator-defined items by using the job manager exit function or job resource manager exit function. For details, see "Chapter 2 Creating and Incorporating Hooks" in "Job Operation Software Administrator's Guide for Job Operation Manager Hook.".

Table 3.8 Specifiable values for DataType

Value	Type	Remarks
PJMX_DATATYPE_CHAR	char	Displays the specified character. DispFormat will be invalid.
PJMX_DATATYPE_INT8	int8	Displays a decimal number if DispFormat is not specified when the pmdumpjobinfo command is used.
PJMX_DATATYPE_UINT8	uint8	Displays a decimal number if DispFormat is not specified when the pmdumpjobinfo command is used.
PJMX_DATATYPE_INT16	int16	Displays a decimal number if DispFormat is not specified when the pmdumpjobinfo command is used.
PJMX_DATATYPE_UINT16	uint16	Displays a decimal number if DispFormat is not specified when the pmdumpjobinfo command is used.
PJMX_DATATYPE_INT32	int32	Displays a decimal number if DispFormat is not specified when the pmdumpjobinfo command is used.
PJMX_DATATYPE_UINT32	uint32	Displays a decimal number if DispFormat is not specified when the pmdumpjobinfo command is used.
PJMX_DATATYPE_INT64	int64	Displays a decimal number if DispFormat is not specified when the pmdumpjobinfo command is used.
PJMX_DATATYPE_UINT64	uint64	Displays a decimal number if DispFormat is not specified when the pmdumpjobinfo command is used.

Value	Type	Remarks
PJMX_DATATYPE_FLOAT	float	Displays a value up to 6 decimal places if DispFormat is not specified when the pmdumpjobinfo command is used.
PJMX_DATATYPE_DOUBLE	double	Displays a value up to 6 decimal places if DispFormat is not specified when the pmdumpjobinfo command is used.
PJMX_DATATYPE_TIMESPEC	struct timespec	Displays a value in nanoseconds if DispFormat is not specified when the pmdumpjobinfo command is used.
PJMX_DATATYPE_TIME	time_t	Displays a decimal number if DispFormat is not specified when the pmdumpjobinfo command is used.
PJMX_DATATYPE_SIZE	size_t	Displays a decimal number if DispFormat is not specified when the pmdumpjobinfo command is used.
PJMX_DATATYPE_STRING	char *	Displays the specified character string. DispFormat will be invalid.

Table 3.9 Specifiable values for DispFormat

Value	Display format	Remarks
dec	Decimal	Valid only when the DataType value is one of the following: PJMX_DATATYPE_INT8 PJMX_DATATYPE_UINT8 PJMX_DATATYPE_INT16 PJMX_DATATYPE_UINT16 PJMX_DATATYPE_INT32 PJMX_DATATYPE_UINT32 PJMX_DATATYPE_INT64 PJMX_DATATYPE_UINT64 PJMX_DATATYPE_FLOAT PJMX_DATATYPE_DOUBLE PJMX_DATATYPE_TIME PJMX_DATATYPE_SIZE
bin	Binary (without padding of high-order digits)	Valid only when the DataType value is one of the following: PJMX_DATATYPE_UINT8 PJMX_DATATYPE_UINT16 PJMX_DATATYPE_UINT32 PJMX_DATATYPE_UINT64 Do not pad high-order digits with zero.

bin_padding	Binary (with padding of high-order digits)	Valid only when the DataType value is one of the following:
		PJMX_DATATYPE_UINT8
		PJMX_DATATYPE_UINT16
		PJMX_DATATYPE_UINT32
		PJMX_DATATYPE_UINT64
		High-order digits are padded with 0 according to the bit length of the value when displayed.
		Example: To display the value of 255
		`8bits: 11111111`
		`16bits: 0000000011111111`
		`32bits: 00000000000000000000000011111111`
		`64bits:`
		`00000000000000000000000000000000000000000000000000000000`
		`11111111`

Value	Display format	Remarks
oct	Octal (without padding of high-order digits)	Valid only when the DataType value is one of the following: PJMX_DATATYPE_UINT8 PJMX_DATATYPE_UINT16 PJMX_DATATYPE_UINT32 PJMX_DATATYPE_UINT64 Do not pad high-order digits with zero.

oct_padding	Octal (with padding of high-order digits)	Valid only when the DataType value is one of the following:
		PJMX_DATATYPE_UINT8
		PJMX_DATATYPE_UINT16
		PJMX_DATATYPE_UINT32
		PJMX_DATATYPE_UINT64
		High-order digits are padded with 0 according to the bit length of the value when displayed.
		Example: To display the value of 255
		`8bits: 0377`
		`16bits: 0000377`
		`32bits: 000000000377`
		`64bits: 0000000000000377`

hex	Hexadecimal (without padding of high-order digits)	Valid only when the DataType value is one of the following: PJMX_DATATYPE_UINT8 PJMX_DATATYPE_UINT16 PJMX_DATATYPE_UINT32 PJMX_DATATYPE_UINT64 Do not pad high-order digits with zero.

hex_padding	Hexadecimal (with padding of high-order digits)	Valid only when the DataType value is one of the following:
		PJMX_DATATYPE_UINT8
		PJMX_DATATYPE_UINT16
		PJMX_DATATYPE_UINT32
		PJMX_DATATYPE_UINT64
		High-order digits are padded with 0 according to the bit length of the value when displayed.
		Example: To display the value of 255
		`8bits: 0xFF`
		`16bits: 0x00FF`
		`32bits: 0x000000FF`
		`64bits: 0x00000000000000FF`

prefix_kilo	*m* K (*n*)	*m*: Displays a value in units of kilo (K) *n*: Value The displayed value is rounded up to the nearest tenth. Example: To display the value of 1024 `1.1 K (1024)` Valid only when the DataType value is one of the following: PJMX_DATATYPE_UINT8 PJMX_DATATYPE_UINT16 PJMX_DATATYPE_UINT32 PJMX_DATATYPE_UINT64

Value	Display format	Remarks
prefix_kibi	*m* Ki (*n*)	*m*: Displays a value in units of kibi (Ki) *n*: Value The displayed value is rounded up to the nearest tenth. Example: To display the value of 1024 `1.0 Ki (1024)` Valid only when the DataType value is one of the following: PJMX_DATATYPE_UINT8 PJMX_DATATYPE_UINT16 PJMX_DATATYPE_UINT32 PJMX_DATATYPE_UINT64
prefix_mega	*m* M (*n*)	*m*: Displays a value in units of mega (M) *n*: Value The displayed value is rounded up to the nearest tenth. Example: To display the value of 1048576 `1.1 M (1048576)` Valid only when the DataType value is one of the following: PJMX_DATATYPE_UINT8 PJMX_DATATYPE_UINT16 PJMX_DATATYPE_UINT32 PJMX_DATATYPE_UINT64
prefix_mebi	*m* Mi (*n*)	*m*: Displays a value in units of mebi (Mi) *n*: Value The displayed value is rounded up to the nearest tenth. Example: To display the value of 1048576 `1.0 Mi (1048576)` Valid only when the DataType value is one of the following: PJMX_DATATYPE_UINT8 PJMX_DATATYPE_UINT16 PJMX_DATATYPE_UINT32 PJMX_DATATYPE_UINT64
prefix_giga	*m* G (*n*)	*m*: Displays a value in units of giga (G) *n*: Value The displayed value is rounded up to the nearest tenth. Example: To display the value of 1073741824 `1.1 G (1073741824)` Valid only when the DataType value is one of the following: PJMX_DATATYPE_UINT8 PJMX_DATATYPE_UINT16 PJMX_DATATYPE_UINT32 PJMX_DATATYPE_UINT64
prefix_gibi	*m* Gi (*n*)	*m*: Displays a value in units of gibi (Gi) *n*: Value The displayed value is rounded up to the nearest tenth.

Value	Display format	Remarks
		Example: To display the value of 1073741824 `1.0 Gi (1073741824)` Valid only when the DataType value is one of the following: PJMX_DATATYPE_UINT8 PJMX_DATATYPE_UINT16 PJMX_DATATYPE_UINT32 PJMX_DATATYPE_UINT64
prefix_tera	*m* T (*n*)	*m*: Displays a value in units of tera (T) *n*: Value The displayed value is rounded up to the nearest tenth. Example: To display the value of 1099511627776 `1.1 T (1099511627776)` Valid only when the DataType value is one of the following: PJMX_DATATYPE_UINT8 PJMX_DATATYPE_UINT16 PJMX_DATATYPE_UINT32 PJMX_DATATYPE_UINT64
prefix_tebi	*m* Ti (*n*)	*m*: Displays a value in units of tebi (Ti) *n*: Value The displayed value is rounded up to the nearest tenth. Example: To display the value of 1099511627776 `1.0 Ti (1099511627776)` Valid only when the DataType value is one of the following: PJMX_DATATYPE_UINT8 PJMX_DATATYPE_UINT16 PJMX_DATATYPE_UINT32 PJMX_DATATYPE_UINT64
date	*YYYY*/*MM*/*DD* *hh*:*mm*:*ss*	Displays the date and time. *YYYY*:Year, *MM*:Month, *DD*:Day, *hh*:Hour, *mm*:Minute, *ss*:Second Valid only when the DataType value is one of the following: PJMX_DATATYPE_TIMESPEC PJMX_DATATYPE_TIME
time	[*DD*] *hh*:*mm*:*ss*	Displays the time. *DD* is displayed for values greater than 24 hours. *DD*:Day, *hh*:Hour, *mm*:Minute, *ss*:Second Values less than one second are rounded up. Example: Value of tv_sec=86400, tv_nsec=1 `01 00:00:01` Valid only when the DataType value is one of the following: PJMX_DATATYPE_TIMESPEC PJMX_DATATYPE_TIME
time_nsec	[*DD*] *hh*:*mm*:*ss*.*nnnnnnnnn*	Displays the time. *DD* is displayed for values greater than 24 hours. *DD*:Day, *hh*:Hour, *mm*:Minute, *ss*:Second, *nnnnnnnnn*:Nanosecond

Value	Display format	Remarks
		Example: Value (struct timespec val) of val.tv_sec=86400 (seconds), val.tv_nsec=1 (nanosecond) `01 00:00:00.000000001` Valid only when the DataType value is one of the following: PJMX_DATATYPE_TIMESPEC
sec	*n* sec	Displays the time in seconds. Example: Value of 2 `2 sec` Valid only when the DataType value is one of the following: PJMX_DATATYPE_UINT8 PJMX_DATATYPE_UINT16 PJMX_DATATYPE_UINT32 PJMX_DATATYPE_UINT64
msec	*n* msec	Displays the time in milliseconds. Example: Value of 100 `100 msec` Valid only when the DataType value is one of the following: PJMX_DATATYPE_UINT8 PJMX_DATATYPE_UINT16 PJMX_DATATYPE_UINT32 PJMX_DATATYPE_UINT64

3.4.2.2 Definitions of output items in job statistical information

The cluster administrator can define which items are output according to the output destination of job statistical information.

The following output destinations are available for job statistical information:

- Job statistical information file (file displayed by the pmdumpjobinfo command)
- .stats file (output file when the -s/-S option in the pjsub command is specified)
- Display when the -s/-S option in the pjstat command is specified

The Record, File, and Command subsections define the output items for each of the output destinations.

```
Cluster {
    Record {
        RecordType {
            DefinitionName=ItemName,ItemName,...
        }
    }
    File {
        RecordType {
            DefinitionName=ItemName,ItemName,...
        }
    }
    Command {
        RecordType  {
            DefinitionName=ItemName,ItemName,...
        }
    }
}
```

The Record subsection defines items for output to the job statistical information file. The File subsection defines items for output to the .stats file that is output when the -s or -S option in the pjsub command is specified. The Command subsection defines items for display when the -s/-S option in the pjstat command is specified.

What is defined in the "*RecordType*" subsection is called a record. Data is output in units of records.
The two types of records are job statistical information records and node statistical information records. Their respective character strings for "*RecordType*" are as follows.

Table 3.10 Record types

Record type	Description
JI	Definition of a job statistical information record One record is output per job.
JN	Definition of a node statistical information record As many records as the number of nodes used by one job are output. In cases where jobs have virtual nodes allocated, as many records as the number of virtual nodes are output.

The "*RecordType*" subsection lists items that are output or are not output to one record. The respective definition names are as follows.

Table 3.11 Definitions of output items for a record

Definition name	Value
ITEM	Item name for job statistical information that is output You can specify the item name of an item defined by the Job Operation Software (see pjstatsinfo(7)) or administrator-defined item (see "ItemName" in "3.4.2.1 Settings of administrator-defined items in job statistical information"). However, items defined by the Job Operation Software may or may not be output, depending on the output destination (see pjstatsinfo(7)). If the specified item cannot be output, it is ignored. To specify multiple items, delimit them with a comma. Their order of output is the order that they are specified in ITEM.
NOT_SAVE_ITEM	Item name for job statistical information that is not output Out of the items defined by the Job Operation Software and the administrator-defined items, only the item specified here is not output. To specify multiple items, delimit them with a comma.

Note

An administrator-defined item may have the same name as an item name for job statistical information defined by the Job Operation Software (see "3.4.2.1 Settings of administrator-defined items in job statistical information"). To use both items, prefix "pjm-" to the item name defined by the Job Operation Software to distinguish between them.

```
[Setting example]
Cluster {
    Record {              <- Defines item for output to job statistical information file
        JI {                             <- Defines job statistical information record
            ITEM=jid,jnam,elpl,nnuma,....   <- Defines item that is output
        }
        JN {                             <- Defines node statistical information record
            NOT_SAVE_ITEM=vnid,...          <- Defines item that is not output
        }
    }
    File {                               <- Defines item for output to .stats file
        JI {
            ITEM=jid,jnam,elpl,nnuma,...
        }
        JN {
            ITEM=nid,....
        }
    }
    Command {             <- Defines item for output by -s or -S option of pjstat command
        JI {
            ITEM=jid,jnam,elpl,nnuma,...
        }
        JN {
            ITEM=nid,....
        }
    }
}
```

Note

- You can write either ITEM or NOT_SAVE_ITEM but not both in a single record definition.
- If there are many items to write, you can divide and write ITEM or NOT_SAVE_ITEM across several lines within the definition of one record.

```
Cluster {
    Record {
        JI {
            ITEM = jid,jnam,elpl,nnuma
            ITEM = elp,mmszu
        }
    }
    ...
}
```

- If the same item of job statistical information is specified multiple times in the definition of one record, an error occurs when the settings are reflected.
- For details on custom resource item names, see "3.4.2.4 Custom resource item name."
- The following job statistical information that output to the .stats file or output by the pjstat command are generated from multiple items. To output these job statistics correctly, define the records so that all required items are output.

Table 3.12 Items required to output job statistical information

Job statistical information	Required items
JOB ID	jid,stepno,bulkno
NODE NUM (REQUIRE)	nnumr,nnumr_x,nnumr_y,nnumr_z,alloctype,nalimit
NODE NUM (ALLOC)	nnuma,nnuma_x,nnuma_y,nnuma_z,alloctype,nalimit
JOB START DATE	sdt,backfill,specifiedsdt (*) For the .stats file, only sdt is required.
ELAPSE TIME (LIMIT)	elpl,maxelpl,minelpl
SUB JOB NUM	snum,esnum (*) For the .stats file, only snum is required.
AVG POWER CONSUMPTION OF CORES/CMG(*num*) (IDEAL)	cmgno,avgpcocc
MAX POWER CONSUMPTION OF CORES/CMG(*num*) (IDEAL)	cmgno,maxpcocc
MIN POWER CONSUMPTION OF CORES/CMG(*num*) (IDEAL)	cmgno,minpcocc
ENERGY CONSUMPTION OF CORES/CMG(*num*) (IDEAL)	cmgno,ecocc

Job statistical information	Required items
AVG POWER CONSUMPTION OF L2CACHE/ CMG(*num*) (IDEAL)	cmgno,avgpcolc
MAX POWER CONSUMPTION OF L2CACHE/ CMG(*num*) (IDEAL)	cmgno,maxpcolc
MIN POWER CONSUMPTION OF L2CACHE/ CMG(*num*) (IDEAL)	cmgno,minpcolc
ENERGY CONSUMPTION OF L2CACHE/CMG(*num*) (IDEAL)	cmgno,ecolc
AVG POWER CONSUMPTION OF MEM/CMG(*num*) (IDEAL)	cmgno,avgpcomc
MAX POWER CONSUMPTION OF MEM/CMG(*num*) (IDEAL)	cmgno,maxpcomc
MIN POWER CONSUMPTION OF MEM/CMG(*num*) (IDEAL)	cmgno,minpcomc
ENERGY CONSUMPTION OF MEM/CMG(*num*) (IDEAL)	cmgno,ecomc
AVG POWER CONSUMPTION OF CPU/PKG(*num*)	pkgno,avgpcocpkg
MAX POWER CONSUMPTION OF CPC/PKG(*num*)	pkgno,maxpcocpkg
MIN POWER CONSUMPTION OF CPU/PKG(*num*)	pkgno,minpcocpkg
ENERGY CONSUMPTION OF CPU/PKG(*num*)	pkgno,ecocpkg
AVG POWER CONSUMPTION OF MEM/PKG(*num*)	pkgno,avgpcompkg
MAX POWER CONSUMPTION OF MEM/PKG(*num*)	pkgno,maxpcompkg
MIN POWER CONSUMPTION OF MEM/PKG(*num*)	pkgno,minpcompkg
ENERGY CONSUMPTION OF MEM/PKG(*num*)	pkgno,ecompkg
AVG POWER CONSUMPTION OF PP0/PKG(*num*)	pkgno,avgpcop0pkg
MAX POWER CONSUMPTION OF PP0/PKG(*num*)	pkgno,maxpcop0pkg
MIN POWER CONSUMPTION OF PP0/PKG(*num*)	pkgno,minpcop0pkg
ENERGY CONSUMPTION OF PP0/PKG(*num*)	pkgno,ecop0pkg
TOFU COORDINATE	tofu_x,tofu_y,tofu_z
NODE COORDINATE	node_x,node_y,node_z

3.4.2.3 Path to a job statistical information file

You can specify the path to a job statistical information file in the Record subsection. In addition, different records can be output to multiple job statistical information files when multiple Record subsections are defined.

The following example outputs a job ID and job name to the job statistical information file jobinfo_acct_summary. The limit value on the elapsed execution time, number of nodes used, elapsed execution time, and memory usage by the job statistical information file jobinfo_acct_detail are also output.

```
Cluster {
    Record {
        PATH=/xxxx/xxxxx/jobinfo_acct_summary
        JI {
            ITEM = jid,jnam
        }
    }
    Record {
        PATH=/xxxx/xxxxx/jobinfo_acct_detail
        JI {
            ITEM = elpl,nnuma,elp,mmszu
        }
    }
}
```

Table 3.13 Specification of the path to a job statistical information file

Item name	Description
PATH	Path to a job statistical information file If one already exists, this is added to it. In cases of output to multiple files, the same file name cannot be specified. If the path to a job statistical information file is changed to a path other than the default path (/var/opt/FJSVtcs/shared_disk/pjm/jsti/jobinfo) or if a new path is added for the file, the file is excluded from log rotation files. To include a job statistical information file in log rotation files, see "3.6 Setting Log Rotation."

3.4.2.4 Custom resource item name

Output job statistical information can also include information on a custom resource. When you define a custom resource, it is automatically given an item name as follows and output in job statistical information by default. However, if the output job statistical information is specified by the papjmstats.conf file, it is output accordingly.

Table 3.14 Custom resource item names for job statistical information

Custom resource information	Item name
Requested amount	CR-STR-*CustomRscName*-req (String type) CR-NUM-*CustomRscName*-req (Numeric type)
Quota	CR-STR-*CustomRscName*-alloc (String type) CR-NUM-*CustomRscName*-alloc (Numeric type)
Usage	CR-STR-*CustomRscName*-use (String type) CR-NUM-*CustomRscName*-use (Numeric type)

3.4.2.5 Reflecting and viewing the papjmstats.conf file

After configuring the papjmstats.conf file, you will need to execute the papjmstatsadm command from the system management node so that the system reflects the settings in the file.

```
[System management node]
# papjmstatsadm -c clstname --set
```

Immediately after the above operation, the settings are reflected in operation. Restarting a node or the entire system is not necessary.

Note

If you modify the Command or Item subsection that affect what pjstat commands display, jobs previously in the EXIT/CANCEL/REJECT state are deleted from the list of jobs output by the -H option of the pjstat command. The information output by the pmdumpjobinfo command is not affected.

To display the current settings, use the --show option of the papjmstatsadm command.

```
[System management node]
# papjmstatsadm -c clstname --show
```

To return settings to their initial state, configure the papjmstats.conf file as follows, and apply the file.
This sets the PATH item to the initial value. It is set to the initial value because no record definition (JI/JN subsection) is written.

```
Cluster {
    Record {
        PATH=/var/opt/FJSVtcs/shared_disk/pjm/jsti/jobinfo
    }
}
```

3.4.2.6 Example of job statistical information settings

The following example shows settings in the papjmstats.conf file.

```
Cluster {
    Item {
        ItemName=MyItem
        ItemNameDisp=MY_ITEM
        RecordNameList=JI,JN
    }
    Record {
        PATH = /xxxx/xxxxx/jobinfo_acct
        JI {
            ITEM = jid,jnam,elpl,nnuma
            ITEM = elp,mmszu,MyItem
        }
    }
    File {
        JI {
            ITEM = jid,jnam,elpl,nnuma,MyItem
        }
    }
    Command {
        JI {
            ITEM = jid,jnam,elpl,nnuma
        }
    }
}
```

The output of job statistical information with the above settings looks like the following.

Display by the pmdumpjobinfo command

The output job statistical information records (seven items) are defined in the Record subsection.

```
# pmdumpjobinfo /xxxx/xxxxx/jobinfo_acct
1,job1,3600,20,120,10,MyVal1
2,job2,7200,10,200,20,MyVal2l
```

Output to the .stats file

The output job statistical information record (five items) is defined in the File subsection.

```
Job Statistical Information

 JOB ID                  : 1
 JOB NAME                : job1
 ELAPSE TIME (LIMIT)     : 00:01:00 (3600)
 NODE NUM (ALLOC)        : 20
 MY_ITEM                 : MyVal1
```

Display by the -s option of the pjstat command

The output job statistical information records (four items) are defined in the Command subsection.

```
$ pjstat -s
```

```
JOB ID                  : 1
JOB NAME                : job1
ELAPSE TIME (USE)       : 00:01:00 (3600)
NODE NUM (ALLOC)        : 20

JOB ID                  : 2
JOB NAME                : job2
ELAPSE TIME (USE)       : 00:03:00 (7200)
NODE NUM (ALLOC)        : 10
```

3.4.3 Settings for job resource management function in a cluster (parsc.conf file)

The cluster administrator can make the following settings in the parsc.conf file (system management node: /etc/opt/FJSVtcs/parsc.conf) by cluster:

- Setting of a percentage of memory for jobs
- Log output level setting for the job resource management function
- Setting for periodically collecting interval of the job statistical information
- Setting to forcibly terminate jobs when the allocated memory exceeds the limit or when OOM-Killer is run
- Setting of whether to periodically collect job statistical information

The following example shows settings of the parsc.conf file.

```
[System management node]
# cat /etc/opt/FJSVtcs/parsc.conf
Cluster {
    ClusterName = clusterA          <- (1)
    LogLevel = 1                    <- (2)
    JobMem = 90                     <- (3)
    RscWatchInterval = 10           <- (4)
    MemFailJobDel = on              <- (5)
}
```

(1) To set the cluster name

(2) To set the log level of the job resource management function to 1

(3) To set the percentage of memory for jobs to 90%

(4) To set the periodically collecting interval of job statistical information to 10 minutes

(5) Setting to forcibly terminate jobs when the allocated memory exceeds the limit or when OOM-Killer is run

For details on the settings, see the man page for the parsc.conf file.

Note

You cannot change settings related to job operations simply by creating or editing the parsc.conf file. As described below, the parscadm command reflects the contents of the parsc.conf file in job operations.

Information

There is no parsc.conf file when the Job Operation Software is installed. If you want to change the default settings, create a new file or copy and edit the sample file (/etc/opt/FJSVtcs/sample/parsc.conf), and place the file in the above path.

3.4.3.1 Settings for job resource management function in a cluster

The following table lists the items set in the Cluster section.

Table 3.15 Setting items of the job resource management function (Cluster section)

Item name	Definition contents	Specifiable value	Default value
ClusterName	Cluster name	Character string with 1 to 63 characters, consisting of single-byte alphanumeric characters, hyphen, and underscore	Not omissible
LogLevel	Log output level of the job resource management function (Normally, 1 is used for operation.) 1: Normal log level 2: Detailed level for debugging 3: More detailed level for debugging	1-3	1
JobMem	Setting of the percentage of memory for jobs that is available to compute nodes (%)	1-100 (Can be specified to the first decimal place.) The OS may hang up if the JobMem value is greater than 90.	90
RscWatchInterval	The periodically collecting interval (unit: minutes) of the job statistical information The time of collection is on a per-job basis, and not all jobs are collected at the same time.	0-1440 (If it is 0, do not collect.)	10
MemFailJobDel	Setting to forcibly terminate jobs when the memory used by jobs (include prologue and epilogue process) exceeds the limit value or when OOM-Killer is run	on: Forcibly terminate off: Continue execution	off
JobRscUsage	Setting of whether to periodically collect job statistical information	on: Collect off: Do not collect	off

Note

- If the value of the JobMem item is too large, the setting may fail. In this case, jobs cannot be executed until a value could be set normally, so job operations are affected. Exercise caution when changing this value. Normally, the JobMem item does not need to be changed.
- It is not usually necessary to change the value of the JobMem and RscWatchInterval. Consult with a Fujitsu systems engineer (SE) or Fujitsu Support Desk to tune up job employment according to the use situation of a system.
- If RscWatchInterval, the periodically collecting interval of the job statistical information is shortened, the load accompanying collection may increase and it may affect job execution.
- If the MemFailJobDel item is off, jobs continue even after reaching the upper limit on memory usage. However, the OS forcibly terminates processes in the jobs.
- If the MemFailJobDel item is on and OOM-Killer starts operating due to OS memory depletion caused by a job, the OS forcibly terminates all the jobs that have the same user ID as the depletion-causing job.
  Jobs whose user ID is 0 (root) are not forcibly terminated even when the MemFailJobDel item is on. For example, suppose that settings are made so that prologue and epilogue processing is executed with root privileges. Then, neither the prologue and epilogue processing nor jobs are forcibly terminated even when memory usage exceeds the limit during the processing or when OOM-Killer starts operating.
- If the JobRscUsage item is on, the size of the file output to the compute cluster management node increases, so the file system must have sufficient free space. For size guidelines, see "3.6 Setting Log Rotation."

- The Job Operation Software sets an upper limit on memory usage, but usage exceeding that limit by any job executed in KVM mode in the job execution environment will not be detected. The behavior when excessive memory is used in a virtual machine depends on the settings of the virtual machine image file used.

3.4.3.2 Reflecting and referencing the parsc.conf file

After setting the parsc.conf file, you need to execute the parscadm command on the system management node so that the system reflects the setting contents.

Note

When changing the setting of the JobMem or MemFailJobDel item, be sure to perform the following operations before reflection by the parscadm command, so that no jobs are running in the target cluster of the settings:

- Stopping the job schedulers and executing the jobs by pmpjmiot command.
- Deleting a job by pjdel command or holding a job on hold by pjhold command.

```
[System management node]
# parscadm --set
```

The above operation incorporates settings immediately to reflect them in operation. Neither a node restart nor a complete system restart is required (except the changes of the JobMem).
If the JobMem item has changed, execute the parscadm command. Then, restart the compute cluster management nodes, and all compute nodes in the target cluster of the settings. Each node then reflects the value of the JobMem item.

Information

If some compute nodes fail to reflect the settings or if new compute nodes are added, you can apply the settings again to only these compute nodes by specifying them in the -f option of the parscadm command. However, all the compute nodes in a cluster must have the same settings, so do not use the -f option to set different values for individual compute nodes.
When changing the setting of the JobMem item with the -f option in the parscadm command, you will need to restart only the target compute nodes.

To display the current settings, use the parscadm command with the --show option.

```
[System management node]
# parscadm --show
Cluster {
    ClusterName = "clusterA"
    JobMem = "90"
    LogLevel = "1"
    RscWatchInterval = "10"
    MemFailJobDel = "on"
}
```

3.4.4 Job ACL function settings in a cluster

The cluster administrator and job operation administrator make job ACL function settings by using pmjacladm command.

Information

- The configuration of the job ACL function is set on the system management node or the compute cluster manegement node. To reflect the contents of the configuration, specify the configuration file or the contents directly to the pmjacladm command as its arguments.

- If you do not set the job ACL function, the initial values set when the job operation management function was installed are applied. For information about the initial values of the job ACL feature, see "Appendix F Defined items of the job ACL function."

The application ranges of the job ACL function are clusters, resource units, and resource groups. You can set the function for each of the following definition targets:

- User
- Group
- Entire range

In this section, a definition for each group means a definition of limits for the entire group. For example, if the limit value of a certain resource is defined for a group, the value is the limit value for the resource used by the entire group. Therefore, even if the amount of resources used by individual users is under the limit value for each user, the group as a whole may in some cases reach the limit value.

Similarly, a definition for an entire range means a definition of a limit value for all resource units or resource groups. Even if individual users or groups do not reach the limit value, all of them as a whole may in some cases reach the limit value.

Note

- Normally, the settings for resource units and resource groups are made by the job operation administrator who manages the units and groups.
  Therefore, when making settings for resource units and resource groups, the cluster administrator needs to confirm the settings beforehand so that they do not conflict with the settings made by the job operation administrator.
- If the resource limit value is not set at the job submission time, the default value in the job ACL function definitions applies. Any users who are not aware of the resource limit value when they submit jobs may not know that the default value is applied. Consequently, they may end up consulting the administrator to determine the cause of operation due to the resource limit value being reached. Therefore, we recommend paying attention to the selection of the default value as the resource limit value and disseminating the default value to all users.

The following example shows job ACL function settings.

```
USER: CL {                                             <- User definitions in a cluster
    user=<def> {                                       <- Default user definitions
        define rscunit           rscunit1
        define rscgroup          rscgroup1

        limit  ru-accept         unlimited
        limit  ru-run-job        unlimited
        limit  ru-use-node       unlimited

        joblimit  elapse         1:00:00    24:00:00   24:00:00
        joblimit  node           1          2147483647 1

        execute pjsub        enable
        execute pjstat       enable
        execute pjdel        enable
        execute pjhold       disable
        execute pjacl        enable

        permit pjsub         allow own
        permit pjstat        allow own
        permit pjdel         allow own
        permit pjhold        deny all
        permit pjacl         allow own
    }
}
GROUP: CL {                                            <- Group definitions in a cluster
    group=<def> {                                      <- Default group unit definitions
        limit  ru-accept         unlimited
        limit  ru-run-job        unlimited
        limit  ru-use-node       unlimited

        define pri-g           128
        define fshare-init-g   100
        define fshare-recovery-g   100
    }
    group=group1 {                    <- Group unit definitions for the target group group1
        define pri-g 130
        define fshare-init-g 200
        define fshare-recovery-g 100
    }
}
ALL: CL  {                                             <- Cluster unit definition
         limit  ru-accept         unlimited
}
```

3.4.4.1 Format of job ACL function definitions

Job ACL function settings are written in a file on the system management node or compute cluster management node. Its path name is an arbitrary name.

Use the following format for the coding in the configuration file according to the definition target, which is a user, a group, or the entire range.

Definitions by user (USER definitions)

```
USER: application_range {
   definition_target {
       defined_item_name value
   }
}
```

Definitions by group (GROUP definitions)

```
GROUP: application_range {
   definition_target {
       defined_item_name value
   }
}
```

Definitions for the entire range (ALL definitions)

```
ALL: application_range {
   defined_item_name value
}
```

In the above format, the application range is a cluster, resource unit, or resource group, and it is coded as follows according to the range.

Table 3.16 Application range format

Format	Description
CL	The application range is a cluster.
CL, RU=*runame*	The application range is the resource unit *runame*.
CL, RU=*runame*, RG=*rgname*	The application range is the resource group *rgname* in the resource unit *runame*.

Information

The coding of the application range does not include a cluster name. Instead, specify the target cluster name when reflecting job ACL function settings with the pmjacladm command.

For a definition target name, write the user or group that is the definition target in the following format.

Table 3.17 Definition target format in the USER definitions

Format	Description
user=<def>	This indicates the default definition for all users.
user=<def>:*gname*	This indicates the default definition for the user whose group is *gname*.
user=*uname*	This indicates the definition for the user *uname*.
user=*uname*:*gname*	This indicates the definition for the user *uname* whose group is *gname*.

Table 3.18 Definition target format in the GROUP definitions

Format	Description
group=<def>	This indicates the default definition for all groups.
group=*gname*	This indicates the definition for the group *gname*.

The next section describes the defined items.

See

For details on the definition formats of the job ACL function, see the man page for pmjacladm(8).

3.4.4.2 Defined items of the job ACL function

You can define the following as defined items of the job ACL function.

Table 3.19 Defined items of the job ACL function

Definition type	Description
limit	This item limits the numbers of submitted jobs, executed jobs, custom resources, etc.
joblimit	This item defines the upper limit values, lower limit values, and default values for limits on resources (CPU resources, memory resources, custom resources, etc.) that can be specified at job submission.
define	This item defines various setting values used for job control.
execute	This item defines whether job-related commands can be executed.
permit	This item defines permission for operation with job-related commands.
select	This item defines the types of custom resources that can be specified at job submission.

Note

The permit and execute items define permissions for command execution and operation. The permissions apply to the user and group privileges at the command execution time. Command execution may consequently encounter an error in cases like the following. Keeping this in mind, set the items appropriately.

Suppose that the primary group group1 and secondary group group2 are set for the user userA. In the following example, permission for execution and operation are set for the secondary group of this user.

```
user=<def> {         # pjsub command execution is not permitted by default.
  execute pjsub        disable
  permit pjsub         deny all
}

user=userA:group2 {  # Permission is granted to the user userA and group group2.
  execute pjsub        enable
  permit pjsub         allow own
}
```

With the above permission settings, an error occurs when the user userA submits a job as shown below.

```
$ id
uid=10001(userA) gid=20001(group1) groups=20001(group1),20002(group2)
$ pjsub job.sh
[ERR.] PJM 0070 pjsub No execute permission.
```

The primary group group1 executed the pjsub command with its group privileges, so pjsub command execution was denied.

Similarly, the error occurs even when the secondary group group2 is specified in the -g option of the pjsub command.

```
$ pjsub -g group2 job.sh
[ERR.] PJM 0070 pjsub No execute permission.
```

This does not mean that the secondary group group2 specified in the -g option was denied. Rather, pjsub command execution was denied because the primary group group1 executed the pjsub command with its privileges.

To prevent the above error, grant pjsub command execution permission to the group group1 at the execution time.

```
user=userA:group1 {  # Permission is granted to the user userA and group group1.
  execute pjsub        enable
  permit pjsub         allow own
}
```

See

For details on the definition items of the job ACL function, see "Appendix F Defined items of the job ACL function" or the man page pmjacladm(8).

3.4.4.3 Priority control of allocated nodes [PG]

Priority control of allocated nodes is a method of selecting nodes according to the priority set for the nodes. It is specified in job ACL function.

A node priority is set up according to node ID. It means that 31 was set up as for the node that has not been set up.

A specification format is shown below.

```
define node-priority priority: nodeID[,nodeID,...]
```

Table 3.20 Setting items of priority control of allocated nodes

Item name	Definition contents
Priority	Specify the priority number (0 -63)
node ID	Specify the node IDs. Delimit multiple node IDs with ",". You can specify a range for multiple node IDs. Specify "-" to indicate a range.

511 characters can be described to a one sentence(include item name).

The following is equivalent to specifying the node ID as 0x41FF0020, 0x41FF0021, 0x41FF0022.

```
define node-priority 16 : 0x41FF0020-0x41FF0022
```

Selection order of compute node

A compute node is chosen in following order by definition.

- For a node allocated job

    1. Exclude the compute nodes used by other jobs.

    2. Sort the compute nodes excluded in step "1" by allocated node priority.

3. Sort the nodes sorted in step "2" by node ID.
4. Add a rank map in the sorted order.

- For a virtual node allocated jobs

1. An execution mode policy excepts the compute node which the job which is SIMPLEX is performing.
2. While Excepting by "1.", Assign [from] and Sort compute Node with Node Priority.
3. If Priority of "2." is the Same compute Node, Sort compute Node with Node Alternative Form.
4. By being Vacant in "3.", if Situation is the Same compute Node, Sort compute Node by Node ID.
5. In order of the sorted compute node, according to a virtual node arrangement policy, choose a compute node and arrange a virtual node.
6. Give virtual node ID of the arranged virtual node according to a rank map.

3.4.4.4 How to define a fair share set

By specifying a fair share set, you can define the initial fair share value and recovery factor for each fair share set by specifying "@*fair_share_set_name*" at the end of the setting value.

The following rules apply to how an initial fair share value and a recovery factor are defined when a fair share set is specified.

- When defining multiple setting values, delimit them with a comma (",").
- To indicate a setting value that is for all the fair share sets, mark it with an asterisk ("*").
- You can define multiple values for the same fair share set. However, in that case, the setting value written in the last position is valid.
- You can omit specifying "@*fair_share_set_name*" at the end of the setting value. In that case, "@def_fs" is assumed specified.

The following examples show coding.

Example 1: Defines the initial user fair share values for the fair share sets fs1 and fs2.

```
define fshare-init 120@fs1,90@fs2
```

Initial user fair share value for fair share set fs1: 120
Initial user fair share value for fair share set fs2: 90

Example 2: Uses an asterisk ("*") to define the initial user fair share value for the fair share sets def_fs, fs1, and fs2.

```
define fshare-init 120@*
```

Initial user fair share value for fair share set def_fs: 120
Initial user fair share value for fair share set fs1: 120
Initial user fair share value for fair share set fs2: 120

Example 3: Changes the initial user fair share value for the fair share set fs1 from the definition in example 2.

```
define fshare-init 120@*,90@fs1
```

Initial user fair share value for fair share set def_fs: 120
Initial user fair share value for fair share set fs1: 90
Initial user fair share value for fair share set fs2: 120

3.4.4.5 Changing the display format of planned job execution start times

Using the job ACL function settings, the administrator can change the display format of planned job execution start times displayed by the pjstat command. The administrator can also change the precision for displaying planned execution start times and the scheduling symbols appended to the displayed times (underlined parts in the following example).

```
$ pjstat
JOB_ID     JOB_NAME   MD ST  USER     START_DATE      ELAPSE_LIM            NODE_REQUIRE
50         job.sh     NM RUN user1    12/08 08:31:22  0000:02:00            12
57         job.sh     NM QUE user1    (12/08 08:50)<  0000:02:00            12
```

```
58         job.sh     NM QUE user1    (12/10 00:00)   0000:02:00           12
59         job.sh     NM QUE user1    (12/15 00:00)#  0000:02:00           12
```

The method of setting the precision for displaying planned execution start times is as follows.

```
define pjstat-sdt-format <DisplayFormat>
```

The following two types of display formats can be specified.

Table 3.21 Display formats of planned job execution start times

Display format	Description
fine	Displays the planned job execution start time in minutes.
custom=*<time_zone>*,*<time_interval>*	Allows the precision of time displayed to be specified according to planned execution start time. To specify multiple formats, delimit them with a semicolon (";"). However, when specifying them, set them in order from the time zone closest to the current time. Example: custom=01:00:00,00:10;08:00:00,01:00;*,24:00

Specify *<time_zone>* and *<time_interval>* as follows.

<time_zone>

Specify a time from the current time as the time zone covered by this setting. You can specify a range of values from 00:01:00 to 9999:59:59.
If "*" is specified, all the time zones after the time covered by this setting are assumed specified.
If "over" is specified, this setting covers the time zones beyond the scheduling period.

<time_interval>

Specify the precision for displaying any planned execution start time that falls into *<time_zone>*. Specify the precision in any of the following formats.

Table 3.22 Specifying the precision for displaying a planned execution start time

Format	Description
hh:mm	Displays the planned execution start time by rounding up to nearest hour (*hh*) and minute (*mm*).
DD	Displays the time by rounding up to nearest day (*DD*).
"*String*"	Displays the specified character string instead of the time. The character string has up to 7 characters, consisting of single-byte alphanumeric characters, a single-byte space, and the symbols shown in the table below. To include a single-byte space in the character string, enclose the character string in double quotation marks escaped by a backslash, and further enclose *<display format>* in double quotation marks (e.g., "custom=*,\"A B\"")

<table>
<tr><td>!</td><td>"</td><td>#</td><td>$</td><td>%</td><td>&</td><td>(</td><td>)</td><td>*</td><td>+</td><td>,</td><td>-</td><td>.</td><td>/</td></tr>
<tr><td>:</td><td><</td><td>=</td><td>></td><td>?</td><td>@</td><td>[</td><td>]</td><td>^</td><td>_</td><td>`</td><td>{</td><td>|</td><td>~</td></tr>
</table>

The following examples show settings and the display.

Example 1: Display format is fine

```
define pjstat-sdt-format fine
```

This format displays planned execution start times in minutes. The following example displays the planned execution start times of jobs 57, 58, 68, and 79 in minutes.

```
$ pjstat
JOB_ID     JOB_NAME   MD ST  USER     START_DATE      ELAPSE_LIM           NODE_REQUIRE
50         job.sh     NM RUN user1    12/08 08:31:22  0000:02:00           12
```

```
57         job.sh     NM QUE user1    (12/08 09:10)<  0000:02:00           12
58         job.sh     NM QUE user1    (12/08 09:20)<  0000:02:00           12
68         job.sh     NM QUE user1    (12/08 10:20)   0000:02:00           12
79         job.sh     NM QUE user1    (12/09 09:10)   0000:02:00           12

$ pjstat -s --choose jid,std    * Displays only job IDs and planned execution start times
JOB ID                      : 50
JOB START DATE              : 2018/12/08 08:31:22
JOB ID                      : 57
JOB START DATE              : (2018/12/08 09:10:00)<
JOB ID                      : 58
JOB START DATE              : (2018/12/08 09:20:00)<
JOB ID                      : 68
JOB START DATE              : (2018/12/08 10:20:00)
JOB ID                      : 79
JOB START DATE              : (2018/12/09 09:10:00)
```

Example 2: Display format is custom

```
define pjstat-sdt-format custom=01:00:00,00:10;08:00:00,01:00;*,24:00
```

This format displays planned job execution start times in units of 10 minutes for times up to 1 hour later, in units of hours for times up to 8 hours later, and in units of 24 hours for even later times. The displayed times for jobs 68 and 79 are different from those in example 1.

```
$ pjstat
JOB_ID     JOB_NAME   MD ST  USER     START_DATE      ELAPSE_LIM           NODE_REQUIRE
50         job.sh     NM RUN user1    12/08 08:31:22  0000:02:00           12
57         job.sh     NM QUE user1    (12/08 09:10)<  0000:02:00           12
58         job.sh     NM QUE user1    (12/08 09:20)<  0000:02:00           12
68         job.sh     NM QUE user1    (12/08 11:00)   0000:02:00           12
79         job.sh     NM QUE user1    (12/10 00:00)   0000:02:00           12

$ pjstat -s --choose jid,sdt
JOB ID                      : 50
JOB START DATE              : 2018/12/08 08:31:22
JOB ID                      : 57
JOB START DATE              : (12/08 09:10)<
JOB ID                      : 58
JOB START DATE              : (12/08 09:20)<
JOB ID                      : 68
JOB START DATE              : (12/08 11:00)
JOB ID                      : 79
JOB START DATE              : (12/10 00:00)
```

Example 3: Display when the scheduling period is exceeded

The display format fine appends the # symbol to the planned execution start times of jobs that have exceeded the scheduling period.

```
$ pjstat
JOB_ID     JOB_NAME   MD ST  USER     START_DATE      ELAPSE_LIM           NODE_REQUIRE
50         job.sh     NM RUN user1    12/08 08:31:22  0000:02:00           12
57         job.sh     NM QUE user1    (12/08 08:50)<  0000:02:00           12
58         job.sh     NM QUE user1    (12/10 00:00)   0000:02:00           12
59         job.sh     NM QUE user1    (12/15 00:00)#  0000:02:00           12
```

By using the display format custom, you can display an arbitrary character string instead of the time of a job that has exceeded the scheduling period.

```
define pjstat-sdt-format custom=over,"(-)"    * Displays character string "(-)"
```

With the above setting, the command displays the planned execution start time of job 59 as follows.

```
$ pjstat
JOB_ID     JOB_NAME   MD ST  USER     START_DATE      ELAPSE_LIM           NODE_REQUIRE
```

```
50          hoge.sh    NM RUN user1    12/08 08:31:22  0000:02:00          12
57          hoge.sh    NM QUE user1    (12/08 08:50)<  0000:02:00          12
58          hoge.sh    NM QUE user1    (12/10 00:00)   0000:02:00          12
59          hoge.sh    NM QUE user1    (-)#            0000:02:00          12
```

The method of changing the scheduling symbol appended to the displayed planned execution start time is as follows.

```
define pjstat-sdt-mark <form>
```

You can specify the following values in <form>.

Table 3.23 Value that can be specified in <form>

<form>	Description
@	Displays "@" for jobs with a specified start time.
<	Displays "<" for jobs that are backfill scheduled.
#	Displays # for jobs that have exceeded the scheduling period.
all	Same as specifying "@","<", and "#" altogether
nothing	Same as specifying none of "@", "<", and "#". Therefore, specify this value when you do not want to display any scheduling symbol.

Specify @, <, and # by delimiting them with a comma.

Example 1: define pjstat-sdt-mark all

This example displays all the scheduling symbols: @, <, and #.

```
$ pjstat
JOB_ID      JOB_NAME   MD ST  USER     START_DATE      ELAPSE_LIM          NODE_REQUIRE
50          job.sh     NM RUN user1    12/08 08:31:22  0000:02:00          12
57          job.sh     NM QUE user1    (12/08 08:50)<  0000:02:00          12
58          job.sh     NM QUE user1    (12/10 00:00)@  0000:02:00          12
59          job.sh     NM QUE user1    (12/15 00:00)#  0000:02:00          12
```

Example 2: define pjstat-sdt-mark @,<

This example displays only the scheduling symbols @ and <. Unlike example 1, # is not displayed after the planned execution start time of job 59.

```
$ pjstat
JOB_ID      JOB_NAME   MD ST  USER     START_DATE      ELAPSE_LIM          NODE_REQUIRE
50          job.sh     NM RUN user1    12/08 08:31:22  0000:02:00          12
57          job.sh     NM QUE user1    (12/08 08:50)<  0000:02:00          12
58          job.sh     NM QUE user1    (12/10 00:00)@  0000:02:00          12
59          job.sh     NM QUE user1    (12/15 00:00)   0000:02:00          12
```

Example 3: define pjstat-sdt-mark nothing

This example displays none of the scheduling symbols.

```
$ pjstat
JOB_ID      JOB_NAME   MD ST  USER     START_DATE      ELAPSE_LIM          NODE_REQUIRE
50          job.sh     NM RUN user1    12/08 08:31:22  0000:02:00          12
57          job.sh     NM QUE user1    (12/08 08:50)   0000:02:00          12
58          job.sh     NM QUE user1    (12/10 00:00)   0000:02:00          12
59          job.sh     NM QUE user1    (12/15 00:00)   0000:02:00          12
```

3.4.4.6 Settings for limiting access to job information

The pjstat command displays job information. When the -A option is specified in the command, other users' job information is also displayed. However, you may want to prevent unauthorized users from viewing the job information.

With the job ACL function, you can set access privileges in the permit pjstat parameter to determine whether a user or group can view other users' or groups' job information with the pjstat command.

Furthermore, you can set a display mode in the define pjstat-display-mode parameter of the job ACL function to determine how jobs are displayed by the pjstat command without access privileges.

The following table shows the three types of display modes for the pjstat command. You can set them in the define pjstat-display-mode parameter of the job ACL function.

Table 3.24 Job information display modes for the pjstat command (define pjstat-display-mode)

Display mode	Description
anonymous	The -A option of the pjstat command allows a user to display other users' job information, even without viewing privileges. However, job, submitter, and group names are masked with the "*"character in the job information displayed without viewing privileges.
nothing	The -A option of the pjstat command does not display other users' job information without viewing privileges. Without viewing privileges, the pjstat command with the --with-summary or --summary option specified also does not include these jobs in summary information on the displayed number of jobs.
summary	The -A option of the pjstat command does not display other users' job information without viewing privileges. However, even without viewing privileges, the pjstat command with the --with-summary or --summary option specified includes these jobs in summary information on the displayed number of jobs.

The following examples show differences in display among the display modes.

anonymous

```
$ pjstat -Av --with-summary

  ACCEPT QUEUED RUNING RUNOUT   HOLD  ERROR REJECT   EXIT CANCEL   TOTAL
       0      0      4      0      0      0      0      0      0       4
s      0      0      4      0      0      0      0      0      0       4

JOB_ID     JOB_NAME   MD ST  USER     GROUP    START_DATE      ELAPSE_TIM ELAPSE_LIM ...
238        job1.sh    NM RUN user1    group1   11/17 09:01:41  0000:09:11 0001:00:00 ...
239        job2.sh    NM RUN user1    group1   11/17 09:05:01  -          0001:00:00 ...
240        job3.sh    NM RUN user1    group1   11/17 09:08:33  -          -          ...
241        ********** NM RUN ******** ******** 11/17 10:03:21  0000:09:10 0001:00:00 ...
```

In this example, the user does not have viewing privileges for the job with job ID 241, so job, user, and group names are masked with the "*" character.

nothing

```
$ pjstat -Av --with-summary

  ACCEPT QUEUED RUNING RUNOUT   HOLD  ERROR REJECT   EXIT CANCEL   TOTAL
       0      0      3      0      0      0      0      0      0       3
s      0      0      3      0      0      0      0      0      0       3

JOB_ID     JOB_NAME   MD ST  USER     GROUP    START_DATE      ELAPSE_TIM ELAPSE_LIM ...
238        job1.sh    NM RUN user1    group1   11/17 09:01:41  0000:09:11 0001:00:00 ...
239        job2.sh    NM RUN user1    group1   11/17 09:05:01  -          0001:00:00 ...
240        job3.sh    NM RUN user1    group1   11/17 09:08:33  -          -          ...
```

In this example, the display mode is nothing, and the user does not have viewing privileges for job ID 241. The job is not included in the summary on the number of jobs and job information.

summary

```
$ pjstat -Av --with-summary
```

```
  ACCEPT QUEUED RUNING RUNOUT   HOLD  ERROR REJECT   EXIT CANCEL   TOTAL
       0      0      4      0      0      0      0      0      0       4
s      0      0      4      0      0      0      0      0      0       4

JOB_ID     JOB_NAME   MD ST  USER     GROUP    START_DATE      ELAPSE_TIM ELAPSE_LIM ...
238        job1.sh    NM RUN user1    group1   11/17 09:01:41  0000:09:11 0001:00:00 ...
239        job2.sh    NM RUN user1    group1   11/17 09:05:01  -          0001:00:00 ...
240        job3.sh    NM RUN user1    group1   11/17 09:08:33  -          -          ...
```

In this example, the display mode is summary, and the user does not have viewing privileges for job ID 241. The job is included in the summary on the number of jobs but is not displayed in job information.

Note

The pjstat command displays two types of step job information: information on the job as a whole (summary job), and information on each sub job.

Summary job information on step jobs is displayed according to the display mode of the resource groups where sub jobs are running.

For example, if the display mode of only some resource groups is nothing, summary job information may or may not be displayed depending on the running sub job.

Such display may confuse users. Therefore, in a system where users can submit the sub jobs of a single step job to different resource groups, we recommend setting the same job information display mode for the resource groups.

The pjshowrsc command displays resource states. With the -v option, the command can display the jobs that use compute nodes.

```
$ pjshowrsc -v 1
[ CLST: clst000 ]
[ RSCUNIT: unit1 ]
[ NODE: 0x01FF0001 ]
     RSC     TOTAL    FREE   ALLOC
     cpu         2       1       1
     mem     499Mi   499Mi       0
RUNNING_JOBS:4243                     <- Job using node 0x01FF0001
```

For this information too, you can set the permit pjshowrsc parameter of the job ACL function to prevent unauthorized users from displaying job information.

3.4.4.7 Application rules for job ACL function definitions

The following USER definition and GROUP definition tables list the order of priority for application of the contents of multiple definitions.

Table 3.25 USER definitions

Priority order	Definition target
1	user=*uname*:*gname*
2	user=*uname*
3	user=<def>:*gname*
4	user=<def>

Table 3.26 GROUP definitions

Priority order	Definition target
1	group=*gname*
2	group=<def>

If there are multiple application ranges, the following order of priority applies to each defined item.

Table 3.27 Application rules between definition type layers

Defined item	Description
limit	Application priority is in the order of resource group, resource unit, cluster, and default value.
define	Application priority is in the order of resource group, resource unit, cluster, and default value.
joblimit	Application priority is in the order of resource group, resource unit, cluster, and default value.
execute	With the pjsub command, application priority is in the order of resource group, resource unit, cluster, and default value. With other commands, there is no concept of resource units and resource groups, so no definition can be made for these ranges. Therefore, the definition for the cluster applies. If the definition item execute is not defined for the root user, the default value is 'enable', regardless of what the pmjacladm or pjacl command displays.
permit	Application priority is in the order of resource group, resource unit, cluster, and default value. If the definition item permit is not defined for the root user, the default value is 'allow all', regardless of what the pmjacladm or pjacl command displays.
select	Application priority is in the order of resource group, resource unit, cluster, and default value.

3.4.4.8 Examples of job ACL function settings

The following examples show job ACL function settings.

- For the elapsed time limit value of userD in the USER definition, change the lower limit to 1 second, the upper limit to 3600 seconds and the default value to 1800 seconds. Also, change the limit on the number of concurrent execution of jobs to undefine in resource unit rscunit1.

```
[System management node or compute cluster management node]
# pmjacladm -c clstname  --set \
'USER: CL, RU=rscunit1 { user=userD { joblimit elapse 1 3600 1800; limit ru-run-job <undef> } }'
```

- Add a new entry for newuserB to the USER definition of the resource unit rscunit1. After the addition, job ACL control remains unchanged since the defined items are all undefined.

```
[System management node or compute cluster management node]
# pmjacladm -c clstname --set 'USER: CL, RU=rscunit1 { user=newuserB }'
```

- The cluster administrator or job operation administrator who has the required privileges for the resource unit rscunit1 in the cluster edits the USER definition of the resource unit and then reflects the change to the job ACL database.

```
[System management node or compute cluster management node]
# pmjacladm -c clstname --show -o jobacl.txt 'USER: CL, RU=rscunit1'   <- (*1)
# vi jobacl.txt
       (Editing of defined items as required)
# pmjacladm -c clstname --set -f jobacl.txt                            <- (*2)
```

(*1) To save to the jobacl.txt file
(*2) To reflect the edited data

- Delete the contents of all definitions in the USER section.

```
[System management node or compute cluster management node]
# pmjacladm -c clstname --del 'USER: *'
```

- Delete data for all resource units in the USER section. Doing so also deletes the data of all resource groups under the resource units.

```
[System management node or compute cluster management node]
# pmjacladm -c clstname --del 'USER: CL, RU=*'
```

- Delete only the default user section within the cluster data in the USER section.

```
[System management node or compute cluster management node]
# pmjacladm -c clstname --del 'USER: CL { user=<def> }'
```

- Delete (initialize) the value of the defined item joblimit node-mem of the user user1 in the resource group rscgroup in the resource unit rscunit in the USER section.

```
[System management node or compute cluster management node]
# pmjacladm -c clstname \
--del 'USER: CL, RU=rscunit, RG=rscgroup { user=user1 { joblimit node-mem } }'
```

- Delete the contents of all definitions.

```
[System management node or compute cluster management node]
# pmjacladm -c clstname --del '*'
```

- Delete (initialize) the value of the defined item with pmjacladm command and -clear option to initialize all definitions and specified empty file(/dev/null)

```
[System management node or compute cluster management node]
# pmjacladm -c clstname --set --clear -f /dev/null.
```

Note

When deleting a user or group from a naming service like LDAP, /etc/passwd, or /etc/group, also delete information on the user or group from job ACL definitions. However, if the user or group is first deleted from the name service, /etc/passwd, or /etc/group, the job ACL function cannot recognize the user or group name. For this reason, delete it by specifying the user or group ID.

- Save the contents of job ACL settings.

```
[System management node or compute cluster management node]
# pmjacladm -c clstname --show -o jobaclbackup.data '*'
```

- Restore the contents of job ACL settings.

```
[System management node or compute cluster management node]
# pmjacladm -c clstname --set --clear -f jobaclbackup.data
```

Information

When executing the pmjacladm command on a compute cluster management node, you can omit specification of a cluster name in the -c option.

3.4.4.9 Precautions when applying the limit value of the job ACL function (definition item limit)

When the limit value (limit definition item) of the job ACL function is applied, the method of counting resource usage varies depending on the job model or type.
The following table lists the method of counting resource usage based on the job model or type. The limit value of the job ACL function is limited by the count values indicated in the table.

Table 3.28 A method to count the resource usage amount per job model or type when applying the limit definition item

Definition item	Batch job			Interactive job
	Normal job	Bulk job	Step job	
limit ru-accept or limit rg-accept	Counted up (+1) per job ID	Counted up (+1) per bulk job (job ID) Not counted up per sub job ID	Counted up (+1) per step job (job ID) Not counted up per sub job ID	-
limit ru-run-job or limit rg-run-job	Counted up (+1) per job in the following states: - RUNNING - RUNNING-A - RUNNING-P - RUNNING-E	Counted up (+1) per bulk job (job ID) in the following states: - RUNNING - RUNNING-A - RUNNING-P - RUNNING-E The sub jobs of a bulk job are not counted up.	Counted up (+1) per step job (job ID) in the following states: - RUNNING - RUNNING-A - RUNNING-P - RUNNING-E	-
limit ru-run-bulksubjob or limit rg-run-bulksubjob	-	Counted up (+1) per sub job in the following states in a bulk job: - RUNNING - RUNNING-A - RUNNING-P - RUNNING-E	-	-
limit ru-use-node or limit rg-use-node	Counted up by the number of requested nodes per node allocated job in the following states: - RUNNING - RUNNING-A - RUNNING-P - RUNNING-E	Counted up by the number of requested nodes for sub jobs in the following states in a node allocated job: - RUNNING - RUNNING-A - RUNNING-P - RUNNING-E	Counted up by the number of requested nodes for sub jobs in the following states in a node allocated job: - RUNNING - RUNNING-A - RUNNING-P - RUNNING-E	-
limit ru-use-core or limit rg-use-core	Counted up by the number of requested CPU cores per virtual node allocated job in the following states: - RUNNING - RUNNING-A - RUNNING-P - RUNNING-E (*)	Counted up by the number of requested CPU cores for sub jobs in the following states in a virtual node allocated job: - RUNNING - RUNNING-A - RUNNING-P - RUNNING-E (*)	Counted up by the number of requested CPU cores for sub jobs in the following states in a virtual node allocated job: - RUNNING - RUNNING-A - RUNNING-P - RUNNING-E (*)	-
limit ru-interact-accept or limit rg-interact-accept	-	-	-	Counted up (+1) per job ID
limit ru-interact-run-job or limit rg-interact-run-job	-	-	-	Counted up (+1) per job in the following states: - RUNNING - RUNNING-A

Definition item	Batch job			Interactive job
	Normal job	Bulk job	Step job	
				- RUNNING-P - RUNNING-E
limit ru-interact-use-node or limit rg-interact-use-node	-	-	-	Counted up by the number of requested nodes per job allocated node in the following states: - RUNNING - RUNNING-A - RUNNING-P - RUNNING-E
limit ru-interact-use-core or limit rg-interact-use-core	-	-	-	Counted up by the number of requested CPU cores per virtual node allocated job in the following states: - RUNNING - RUNNING-A - RUNNING-P - RUNNING-E (*)

-: Exception
(*) For virtual node allocated jobs or all jobs, the target of the limit on the number of CPU cores used simultaneously can be specified with the UseCoreLimit item in the papjm.conf file. If all jobs are subject to the limit, the number of CPU cores used by the node allocated jobs is a value obtained by multiplying the number of CPUs per node by the number of request nodes.

3.4.5 Settings for advanced job scheduling

The cluster administrator can configure advanced job scheduling as follows in the pjs.conf file (/var/opt/FJSVtcs/shared_disk/pjm/.private/pjs.conf on the active compute cluster management node):

- Settings related to fair share values

  - Setting of the unit of calculation for fair share values on FX servers (FshareUnit)

  - Disabling of fair share value recovery when a job ends before reaching the elapsed time limit value (FshareReturn)

  For details on these settings, see "2.5.2.2 Fair share function."

- Interpretation of submission times when scheduling sub jobs of a step job (StepJobSchedUnit)

  For details on this setting, see "2.5.4.3 Scheduling of sub jobs of a step job."

- Parameter for tuning the job rescheduling start wait time (NewJobBatchReschedAggregateTime, NewJobInterReschedAggregateTime, AggregateReschedNewJobFactor, GraceOverReschedAggregateTime, AggregateReschedGapFactor, PmpjmAlterReschedAggregateTime)

  For details on these settings, see "2.5.4.2 Job scheduling parameters."

- Setting for the guarantee period of planned execution start times (NotReschedPeriod)

  For details on this setting, see "2.5.4.4 Guarantee of planned job execution start time (setting that prevents a delay in the job execution start time)."

- Time setting that extends a scheduling trigger (ExtendSchedPeriod)

  For details on this setting, see "2.5.4.2 Job scheduling parameters."

These settings are common within the cluster.

Note

The pjs.conf file is installed on the active compute cluster management node. Edit it there.

The following example shows settings of the pjs.conf file.

```
[Active compute cluster management node]
# cat /var/opt/FJSVtcs/shared_disk/pjm/.private/pjs.conf
ResourceUnit {
    FshareUnit = 1                           (*1)
    FshareReturn = 0                         (*2)
    StepJobAcceptDate = 1                    (*3)
    StepJobSchedUnit = 1                     (*4)
    NotReschedPeriod = 2147483647            (*5)
    NewJobBatchReschedAggregateTime = 30     (*6)
    NewJobInterReschedAggregateTime = 0
    AggregateReschedNewJobFactor = 10
    GraceOverReschedAggregateTime = 60
    AggregateReschedGapFactor = 20
    PmpjmAlterReschedAggregateTime = 30
    ExtendSchedPeriod = 3600                 (*7)
}
```

(*1) Sets the number of CPU cores as the unit of calculation for fair share values.
(*2) Disables fair share value recovery at job end.
(*3) Changes the interpretation of submission times when scheduling sub jobs of a step job.
(*4) Changes scheduling unit of sub jobs in step job
(*5) The guarantee period of planned execution start times
(*6) For details on the parameters (NewJobBatchReschedAggregateTime to PmpjmAlterReschedAggregateTime) for tuning the job rescheduling start wait time, see "Table 3.29 Setting items for advanced job scheduling."
(*7) Sets time to extend scheduling trigger

Write the settings in the ResourceUnit section. You can set the following items.

Table 3.29 Setting items for advanced job scheduling

Item name	Description	Specifiable value	Default value
FshareUnit [FX]	Resource unit for calculating fair share values (*) This item is ignored for resource units on PRIMERGY servers.	0: Calculate based on the number of nodes. 1: Calculate based on the number of CPU cores.	0
FshareReturn	Whether to recover the fair share value when a job ends before reaching the elapsed time limit value	0: Do not recover the fair share value. 1: Recover the fair share value.	1
StepJobAcceptDate	Interpretation of submission times when scheduling sub jobs of a step job	0: Actual sub job submission time 1: Scheduled time of a sub job. However, if 0 is specified for StepJobSchedUnit, the sub jobs other than the first one of the submitted sub jobs in the same resource unit are the expected end times of the previous sub jobs.	1

Item name	Description	Specifiable value	Default value
StepJobSchedUnit	Scheduling unit of sub jobs in a step job	0: Simultaneously schedule all sub jobs already submitted to the same resource unit but not yet executed. 1: Schedule only the first sub job already submitted but not yet executed	0
NotReschedPeriod	Guarantee period of planned execution start times (seconds) (*1) If a value larger than or equal to the value of the scheduling period (setting of the SchedulePeriod and DynamicSchedulePeriod item in papjm.conf) is set, the guarantee period of planned execution start times is the same as the scheduling period. If 0 is set as the value, planned execution start times are not guaranteed.	0 - 2147483647 (seconds)	2147483647
NewJobBatchReschedAggregateTime	Maximum wait time until rescheduling starts when a new batch job is submitted (seconds)	0 - 2147483647 (seconds)	30
NewJobInterReschedAggregateTime	Maximum wait time until rescheduling starts when a new interactive job is submitted (seconds) (*2)	0 - 2147483647 (seconds)	0
AggregateReschedNewJobFactor	Coefficient for determining the rescheduling start wait time when a new batch job or interactive job is submitted (milliseconds) The wait time is the value obtained by multiplying the time by the number of queued jobs. (*2)(*3)	0 - 2000 (milliseconds)	10
GraceOverReschedAggregateTime	Maximum wait time until rescheduling starts when a job ends earlier than the upper limit of the elapsed execution time (seconds) (*2)	0 - 2147483647 (seconds)	60
AggregateReschedGapFactor	Coefficient for determining the rescheduling start wait time when a job ends earlier than the upper limit of the elapsed execution time (milliseconds) The wait time is the value obtained by multiplying the time by the number of queued jobs. (*2)(*4)	0 - 2000 (milliseconds)	20

Item name	Description	Specifiable value	Default value
PmpjmAlterReschedAggregateTime	Rescheduling start wait time (seconds) when a job parameter is changed by the pjalter or pmalter command (*2)	0 - 2147483647 (seconds)	30
ExtendSchedPeriod	Extended period when scheduling continues uninterrupted even if new scheduling occurs during scheduling processing (seconds) (*5)	0 - 2147483647 (seconds)	0

(*1)
For notes on the NotReschedPeriod item setting, see "2.5.4.4 Guarantee of planned job execution start time (setting that prevents a delay in the job execution start time)."
(*2)
Normally, it is not necessary to change this setting.
If the number of jobs is large, lower-priority jobs waiting for execution are not scheduled and the planned execution start time may not be displayed. In this case, the lower-priority jobs waiting for execution can be scheduled with an adjustment of the "rescheduling start wait time setting." Consult a Fujitsu systems engineer (SE) or the Fujitsu Support Desk to adjust the setting. For details on the setting, see "2.5.4.2 Job scheduling parameters."
(*3)
When a batch job is submitted, NewJobBatchReschedAggregateTime (seconds) is compared with the value obtained by multiplying AggregateReschedNewJobFactor (milliseconds) by the number of queued jobs, and the smaller value is selected as the wait time.
When an interactive job is submitted, NewJobInterReschedAggregateTime (seconds) is compared with the value obtained by multiplying AggregateReschedNewJobFactor (milliseconds) by the number of queued jobs, and the smaller value is selected.
(*4)
GraceOverReschedAggregateTime (seconds) is compared with the value obtained by multiplying AggregateReschedGapFactor (milliseconds) by the number of queued jobs, and the smaller value is selected.
(*5)
In a situation where many jobs are submitted, frequent rescheduling may cause scheduling delays. ExtendSchedPeriod is a setting that allows scheduling to continue for a certain period of time, even if new scheduling occurs, in order for the scheduling in progress to complete.

Note

To reflect the pjs.conf file contents, you need to restart the active compute cluster management node.
If the system with a single node serving as all of the system management node, compute cluster management node, and login node, restart the node with no jobs running on all compute nodes. Also, you need to restart the Job Operation Software services on all compute nodes. For details on this procedure, see "How to Restart the Node Serving as All of System Management Node, Compute Cluster Management Node, and Login Node" in "Chapter 3 Maintenance Work Problems" of the manual "Job Operation Software Troubleshooting".

3.4.6 Settings for other

Job operation management function has a function that notifies details by e-mail to a job submission user at the time of the execution start and end of a job, or an error. E-mail is transmitted to the user account of a compute cluster management node.

Build according to a system about the structure of the e-mail delivery to each user from a compute cluster management node.

3.5 Settings for the Job Operation Administrator

This section describes the contents of settings by the job operation administrator, among the settings of the job operation management function.

Information

- The setting and operation work of the job operation management function requires job operation administrator privileges or higher.
- The examples in this section contain commands specifying a cluster name in the -c option for operation on the system management node. Write the name in this way. However, if the environment variable PXMYCLST specifies the cluster name, you can omit specification of the cluster name in the -c option during actual operation.

3.5.1 Job operation management function settings in a resource unit (pmpjm.conf file)

The job operation administrator can make the following settings for the job operation management function in the pmpjm.conf file (system management node: /etc/opt/FJSVtcs/Rscunit.d/*rscunitname*/pmpjm.conf) by resource unit:

- Log output level setting for the job scheduler function
- Setting for enabling/disabling the backfill function
- Setting the target range of the backfill function (executing lower-priority jobs ahead of higher-priority jobs)
- Buffer time setting for the job execution interval
- Rescheduling grace time setting
- Setting of the grace time before the forced termination of a job that continued running after exceeding the minimum elapsed time limit value
- Interval setting for job resource map creation
- Scheduling period setting
- Setting of a dynamic scheduling period
- Settings of whether job models (normal, step, and bulk job) automatic re-execution
- E-mail transfer function setting
- Setting of whether to subtract the fair share value when job execution starts
- Setting of the fair share recovery value
- Resource unit type
- Setting for guarantee of planned execution start times
- Setting of the function for outputting a job runtime error to the standard error output
- Interpretation of the submission time when scheduling a held job that was released
- Setting of the directory path where the plugin libraries are located
- Setting of the maximum number of jobs to schedule
- Setting of the method for limiting the jobs to schedule
- Job selection policy settings for the resource unit
- Prologue and epilogue function settings
- Resource group settings
    - Setting of a resource group name
    - Setting of the node resources in resource groups
    - Setting of the priority of a resource group
    - Setting of allocation method for sharing Tofu units by multiple jobs
    - Setting of the execution mode policy of a resource group

    - Setting for enabling/disabling the backfill function
    - Setting for guarantee of planned execution start times
    - Setting of the function for outputting a job runtime error to the standard error output
    - Setting of whether to subtract the fair share value when job execution starts
    - Interpretation of the submission time when scheduling a held job that was released
    - Job selection policy settings for the resource group
    - Setting of custom resources in a resource group
    - Setting for the job manager exit function for a resource group
    - Setting for the job scheduler exit function for a resource group
- Setting of custom resources in a resource unit
- Job manager exit settings for a resource unit
- Job scheduler exit settings for a resource unit
- Settings for job scheduler function

The following example shows settings of the pmpjm.conf file.

```
[System management node]
# cat /etc/opt/FJSVtcs/Rscunit.d/unit1/pmpjm.conf
ResourceUnit {
    ResourceUnitName = unit1
    LogLevel = 1
    Backfill = yes
    BackfillTarget = rscgrp
    DecidedGap = 00:01:00
    Grace = 00:02:00
    CreateRscMap = "01:00:00, 00:10:00"
    CreateRscMap = "24:00:00, 01:00:00"
    CreateRscMap = "*, 24:00:00"
    SchedulePeriod = 25:00:00
    DynamicSchedulePeriod = 2,24:00:00
    RestartNormal = yes
    RestartStep = yes
    RestartBulk = no
    MailSend = yes
    Fairshare = off
    FshareRecoveryValue = 236
    StartTimeGuarantee = on
    JobStderrMsgLevel = 0
    HoldAcceptDate = release
    SchedulerPluginLoadPath = "/etc/opt/FJSVtcs/plugin/pjm/pjsd/normal_mode/"
    JobSchedulingTargetLimit = 10000
    JobSchedulingTargetMode = jobselectpolicy
    JobSelectPolicy {
        name = policy2
    }
#    ResourceGroup {
#        ResourceGroupName =
#        ResourceGroupNode =
#        Backfill =
#        StartTimeGuarantee =
#        ResourceGroupFairshare =
#        JobSelectPolicy {
#            name =
#        }
#        JobEvaluation {
#            name =
#        }
#        CustomResource {
#            Name =
#            ValueType =
#            Value =
#        }
#        ExitFunc {
#            ExitFuncLib =
#            ExitFuncPri =
#            ExitFuncType =
#        }
#    }
#    Scheduler {
#        Name =
#        Plugins =
#    }
}
```

For details on pmpjm.conf file settings, see the man page for the pmpjm.conf file.

Information

There is no pmpjm.conf file when the Job Operation Software is installed. If you want to change the default settings, create a new file or copy and edit the sample file (/etc/opt/FJSVtcs/sample/pmpjm.conf), and place the file in the above path.

Note

- You cannot change settings related to job operations simply by creating or editing the pmpjm.conf file. As described below, the pmpjmadm command reflects the contents of the pmpjm.conf file in job operations.

- The maximum length of a single line in the pmpjm.conf file is 511 characters.

3.5.1.1 Resource unit settings

The ResourceUnit section of the pmpjm.conf file enables you to configure the following for a specific resource unit:

Table 3.30 Resource Unit Settings (ResourceUnit Section)

Item name	Definition contents	Specifiable value	Default value
ResourceUnitName	Resource unit name	(*1)	Not omissible
Items that can be set in the ResourceUnit section of the papjm.conf file. (except UseCoreLimit)	For the detail of items, refer "3.4.1.2 Default value settings for resource units".		
AllocType	Resource unit type Set node or vnode, depending on the model of the compute nodes in the resource unit.	node: For the compute nodes of FX server vnode: For the compute nodes of PRIMERGY server	node
JobStderrMsgLevel	Setting of the function for outputting a job runtime error to the standard error output (*2)	0: Do not output. 1: Output only ERROR message. 2: Output ERROR and WARNING messages. 3: Output WARNING and INFO messages	0

Item name	Definition contents	Specifiable value	Default value
SchedulerPluginLoadPath	Absolute path of the directory where the plugin library is located in normal mode. For more information, refer to "Job Operation Software API user's Guide for Scheduler API."	List of directory names	/etc/opt/FJSVtcs/plugin/pjm/pjsd/normal_mode/

(*1) Specifies the resource unit name identified in the following command.

```
# pashowclst -c clstname --rscunit
```

(*2) The job runtime error is an error message corresponding to the job end code. For details on the error message, see "Job management function" in "Messages in job outputs" in "Chapter 3 Command Reference for End-users" of "Job Operation Software Command Reference." However, an interactive job are excluded.

Fairshare item

For the Fairshare setting item relating to subtraction of the fair share value, you can also define a setting for each fair share set in the pmpjm.conf file as follows. This definition cannot be specified by the item Fairshare in the papjm.conf file.

```
Fairshare=on@FairshareSetName (Example:  Fairshare=on@fs_rs)
```

You can specify a fair share set name consisting of the following:

- 1 to 15 characters
- Single-byte alphanumeric characters
- Hyphen
- Underscore

If "@fair share set name" is omitted when Fairshare=off is set, def_fs is applied as the fair share set name.

3.5.1.2 Resource group settings

You can configure specific resource groups by placing the ResourceGroup section within the ResourceUnit section of the pmpjm.conf file.

Table 3.31 Setting items of a resource group (ResourceGroup section)

Item name	Definition contents	Specifiable value	Default value
ResourceGroupName	Resource group name	Character string with 1 to 63 characters, consisting of single-byte alphanumeric characters, hyphen, and underscore	Not omissible
ResourceGroupNode	Node resources of the resource group	[FX] Specify the node number, node rate, or node shape. There can be a mixture within a single resource unit. - For the number of nodes, specify a numerical value (e.g., 10). (*1) - For the rate, specify a percentage (e.g., 100%). (*1) - For the shape, specify the minimum coordinates and maximum coordinates of a Tofu unit. Specify the unit is pointed unit in an entire	100%

Item name	Definition contents	Specifiable value	Default value
		cluster. Example: For the minimal coordinates (0, 0, 1) and the maximum coordinates (0, 0, 5), write "0, 0, 1-0, 0, 5".	
		[PG] Specify node IDs in the following format. - Delimit multiple node IDs with ",". - You can specify a range of node IDs with a hyphen "-". You can specify multiple ranges separated by commas ",". (Example: *nodeid1-nodeid2*, *nodeid3-nodeid4*) However, one range must be in the same node group. This item may appear more than once in the same ResourceGroup section.	All nodes within resource unit
ResourceGroupPrio	Priority of the resource group	0 (low priority) to 255 (high priority) It is valid when rscgrp_prio of the job selection policy is set.	127
ResourceGroupTsha [FX]	Assignment method for sharing Tofu units by multiple jobs	1: Only 1 node job can be shared. 2: 1 to 11 node jobs can be shared. However, this is effective only in Torus mode.	1
ResourceGroupExecPolicy [FX]	Execution mode policy of the resource group	share simplex However, the setting is valid only for a shape-specified resource group.	share
Backfill	Backfill function	yes: Enabled no: Disabled	Follow the ResourceUnit section setting.
StartTimeGuarantee	Setting for guarantee of planned execution start times	on: Planned execution start times are guaranteed. off: Planned execution start times may change at rescheduling.	Follow the ResourceUnit section setting.
JobStderrMsgLevel	Setting of the function for outputting a job runtime error to the standard error output (*2)	0: Do not output. 1: Output only ERROR message. 2: Output ERROR and WARNING messages. 3: Output WARNING and INFO messages.	0
ResourceGroupFairshare	Setting of whether to subtract the fair share value when job execution starts	on: Subtract off: Do not subtract (*3)	off
HoldAcceptDate	Interpretation of the submission time when scheduling a held job that was released	release: Schedule this job at its release time	release

Item name	Definition contents	Specifiable value	Default value
		accept: Schedule this job at its submission time	

(*1) For FX servers, even when the compute nodes to be allocated to a resource group are specified as a number of nodes or percentage for the setting of the ResourceGroupNode item, the value is rounded up to the nearest Tofu unit (12 nodes).
(*2) The job runtime error is an error message corresponding to the job end code. For details on the error message, see "Job management function" in "Messages in job outputs" in "Chapter 3 Command Reference for End-users" of "Job Operation Software Command reference" manual. However, an interactive job are excluded.
(*3) If the Fairshare setting item for the resource unit is on, the fair share value is subtracted when job execution starts, even when the ResourceGroupFairshare item is off.
To define and use a fair share set for the relevant resource group, specify the following:

on@*fair_share_set_name* (example: ResourceGroupFairshare=on@fs_rs)

You can specify a "*fair_share_set_name*" value consisting of the following:

- 1 to 15 characters

- Single-byte alphanumeric

- Hyphen

- Underscore

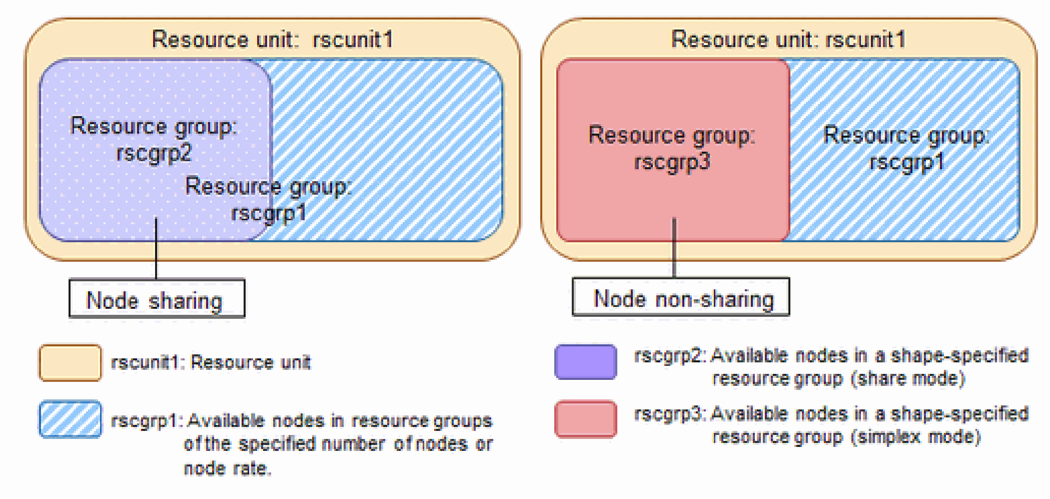
If "@*fair_share_set_name*" is omitted when off is set, def_fs is the applied fair share set name.

Information

FX servers allow a single resource unit to be a mix of resource groups in ranges specified by different methods (specified by shape, number of nodes, or node rate).

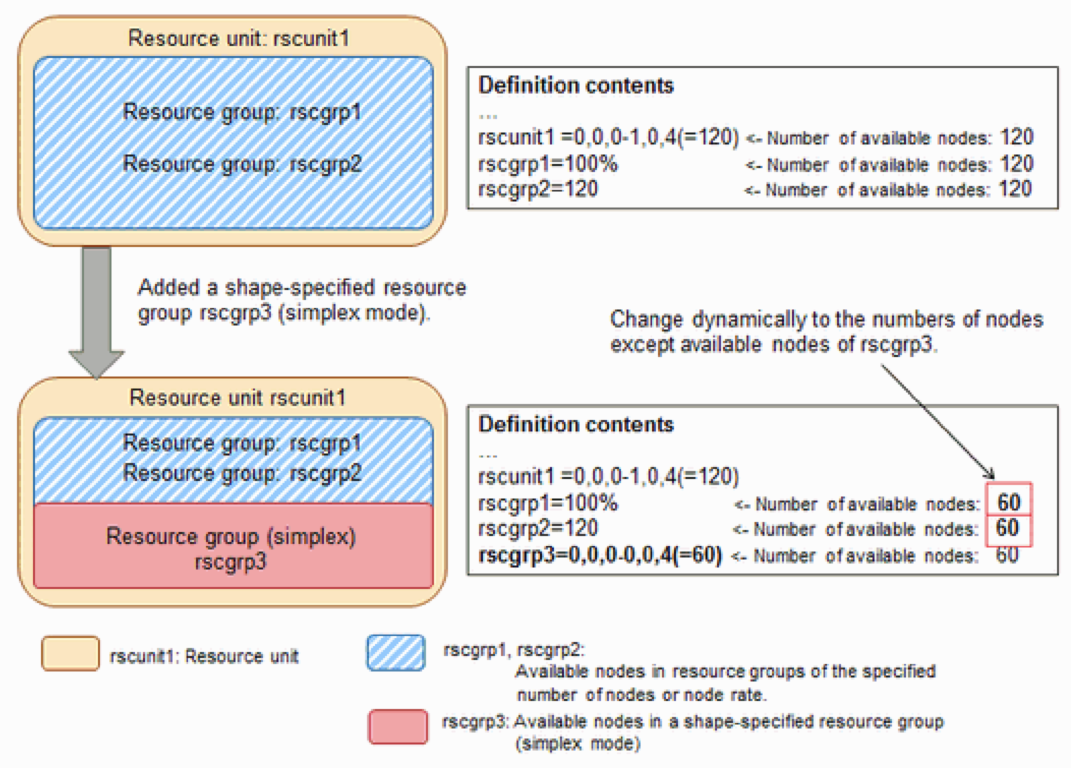
Suppose that the execution mode policy of a shape-specified resource group has the share setting in the ResourceGroupExecPolicy item. In this case, the nodes are shared with resource groups of the specified number of nodes or node rate.
If the execution mode policy of the resource group has the simplex setting, the nodes are not shared with resource groups of the specified number of nodes or node rate.

Figure 3.1 Relationship between the following resource groups with a set execution mode policy: resource group of the specified shape, and resource group of the specified number of nodes or node rate


Resource unit: rscunit1
Resource group: rscgrp2
Resource group: rscgrp1
Node sharing
Resource unit: rscunit1
Resource group: rscgrp3
Resource group: rscgrp1
Node non-sharing
rscunit1: Resource unit
rscgrp1: Available nodes in resource groups of the specified number of nodes or node rate.
rscgrp2: Available nodes in a shape-specified resource group (share mode)
rscgrp3: Available nodes in a shape-specified resource group (simplex mode)

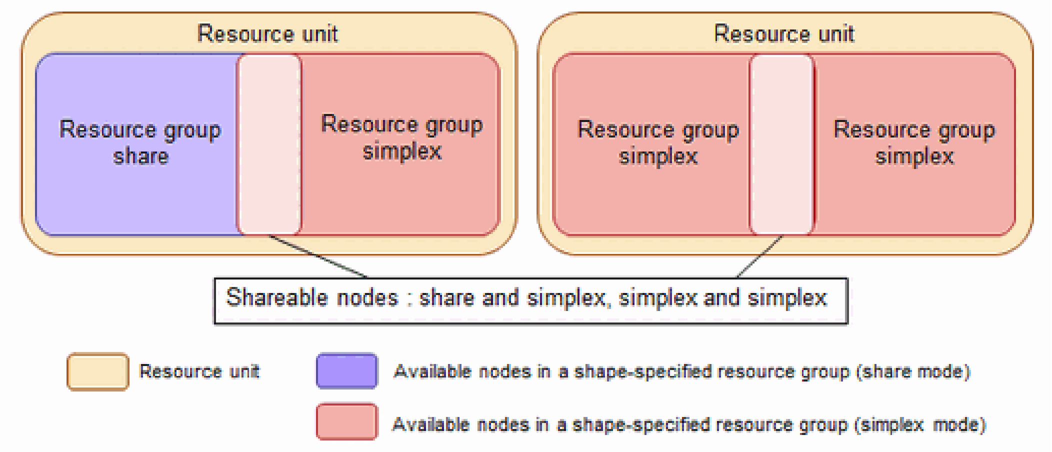
Suppose that a resource unit contains a resource group of the specified number of nodes or node rate. Also suppose that a shape-specified resource group that is set to simplex mode is added to the resource unit. Then, the number of nodes in the resource group of the specified number of nodes or node rate changes dynamically.
The following figure shows such a case.

Figure 3.2 Dynamic change of a resource group (when a shape-specified resource group (in simplex mode) is dynamically added)


Resource unit: rscunit1
Resource group: rscgrp1
Resource group: rscgrp2
Definition contents
...
rscunit1 =0,0,0-1,0,4(=120) <- Number of available nodes: 120
rscgrp1=100% <- Number of available nodes: 120
rscgrp2=120 <- Number of available nodes: 120
Added a shape-specified resource group rscgrp3 (simplex mode).
Change dynamically to the numbers of nodes except available nodes of rscgrp3.
Resource unit rscunit1
Resource group: rscgrp1
Resource group: rscgrp2
Resource group (simplex) rscgrp3
Definition contents
...
rscunit1 =0,0,0-1,0,4(=120)
rscgrp1=100% <- Number of available nodes: 60
rscgrp2=120 <- Number of available nodes: 60
rscgrp3=0,0,0-0,0,4(=60) <- Number of available nodes: 60
rscunit1: Resource unit
rscgrp1, rscgrp2: Available nodes in resource groups of the specified number of nodes or node rate.
rscgrp3: Available nodes in a shape-specified resource group (simplex mode)

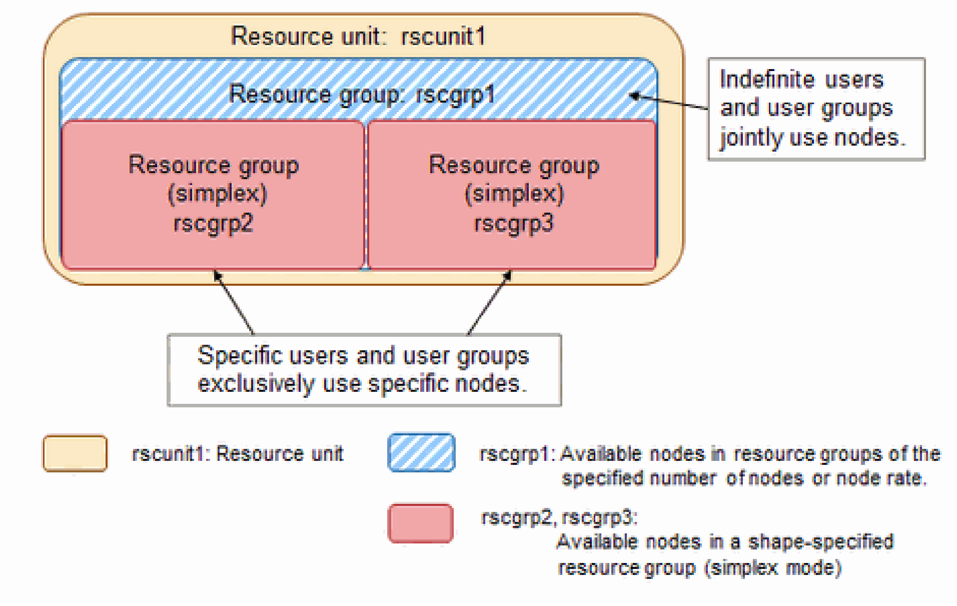
If a single resource unit contains multiple shape-specified resource groups, the nodes are shared regardless of whether the execution mode policies of the resource groups have the share or simplex setting. Also, the jobs operate within the specified range.

Figure 3.3 Relationship between shape-specified resource groups where one is set to simplex mode and the other is set to share mode


Resource unit
Resource group share
Resource group simplex
Resource unit
Resource group simplex
Resource group simplex
Shareable nodes : share and simplex, simplex and simplex
Resource unit
Available nodes in a shape-specified resource group (share mode)
Available nodes in a shape-specified resource group (simplex mode)

These settings enable operation with a mixture of the following compute node groups. One compute node group (shape-specified resource group that is set to simplex mode) permits specific users and user groups who run large-scale jobs to borrow and exclusively use specific nodes. The other compute node group (resource group of the specified number of nodes or node rate) is used jointly by indefinite users. The following example shows such a case.

Figure 3.4 Job operation example of resource groups that is set within a single resource unit


Resource unit: rscunit1
Resource group: rscgrp1
Resource group (simplex) rscgrp2
Resource group (simplex) rscgrp3
Indefinite users and user groups jointly use nodes.
Specific users and user groups exclusively use specific nodes.
rscunit1: Resource unit
rscgrp1: Available nodes in resource groups of the specified number of nodes or node rate.
rscgrp2, rscgrp3: Available nodes in a shape-specified resource group (simplex mode)

3.5.1.3 Prologue and epilogue function settings

To use the prologue and epilogue function, prologue and epilogue scripts must be created and incorporated into the job operation management function. For details, see "Prologue and Epilogue Function " in "Chapter 2 Creating and Incorporating Hooks " in "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

3.5.1.4 Job selection policy settings

You can make job selection policy settings in the pmpjm.conf file.

The job selection policy is set in the ResourceUnit section or ResourceGroup section in the same way as when configuring the job operation management settings in a cluster (papjm.conf file). The settings for a resource group have priority over the settings for a resource unit. If the settings for a resource group are omitted, the settings for the resource unit are applied.

For details on the setting method, see "3.4.1.3 Job selection policy settings".

Note

Note the following about the job selection policy settings in the pmpjm.conf file.

- When using the job evaluation point job_epoint, which is one of job selection policy elements, you can select a job evaluation definition within the cluster. From the job evaluation definitions set in the papjm.conf file by the cluster administrator, choose one appropriate to resource unit operations.
  For example, if the settings in the papjm.conf file described in "3.4.1.3 Job selection policy settings" apply to the cluster, you can apply the job evaluation definition jobeval1 to the resource unit.

```
# cat /etc/opt/FJSVtcs/Rscunit.d/unit1/pmpjm.conf
ResourceUnit {
    ResourceUnitName = unit1
    LogLevel = 1
    ...
    JobEvaluation {
        name = jobeval1    <- Job evaluation definition name applied
    }
    ...
```

When specifying an already defined job evaluation definition name in the pmpjm.conf file, you cannot change the definition contents.

- Without using an already defined job evaluation definition, you can even write a job evaluation definition specific to a resource unit.

```
# cat /etc/opt/FJSVtcs/Rscunit.d/unit2/pmpjm.conf
ResourceUnit {
    ResourceUnitName = unit2
    LogLevel = 1
    ...
    JobEvaluation {
        waittime  = 100    <- Job evaluation definition specific to resource unit
    }
    ...
```

- Only one job evaluation definition can be written in the pmpjm.conf file. In pmpjm.conf, a name for job evaluation definition cannot be set.

- To prioritize the execution of a job based on the fair share value of a fair share set using the group_fairshare, usr_in_grp_fairshare, and user_fairshare items, specify the fair share set as follows.

```
ItemName@FairshareSetName=Value   (Example: user_fairshare@fs_rg=1,desc)
```

  You can specify a fair share set name consisting of the following:

  - 1 to 15 characters
  - Single-byte alphanumeric characters
  - Hyphen
  - Underscore

  Though you can set these items in the papjm.conf file too, fair share sets can be specified only in the pmpjm.conf file.

3.5.1.5 Settings for the fair share function

To control job execution priorities with the fair share values in a single resource unit, set the fair share function and job selection policy in the pmpjm.conf file. The setting method is the same as the papjm.conf file. For details on the setting method, see "3.4.1.4 Settings for the fair share function ."

To control job execution priorities with fair share values in a single resource group, configure the pmpjm.conf file as follows.

1. Set the fair share function.
   To subtract the fair share value when job execution starts, set the ResourceGroupFairshare item in the pmpjm.conf file to on. Also, to define and use a fair share set for the relevant resource group, define "on@*fair_share_set_name*."

   Note

   If the Fairshare setting item for the resource unit is on, the fair share value is subtracted when job execution starts, even when the ResourceGroupFairshare item is off.

2. Set a job selection policy.

   To use fair share values as the job selection policy, add policies as follows to the job selection policy JobSelectPolicy in the pmpjm.conf file.

   - To include user fair share values in priority control, add user_fairshare.
   - To include group fair share values in priority control, add group_fairshare.
   - To include the fair share values of users in a group in priority control, add usr_in_grp_fairshare.

   To specify a fair share set and assign job execution priorities from fair share values, define "*policy_name*@*fair_share_set_name*."

In addition to the above settings, change the following settings as required: fair share value, initial fair share value, fair share recovery value, or recovery factor. For details on how to set these values, see "4.2.5 Monitoring and changing a fair share value and initial fair share value ."

3.5.1.6 Custom resource settings

You can make settings for custom resources in the pmpjm.conf file.

Custom resources are set in the CustomResource subsection in the ResourceUnit or ResourceGroup section. Settings for a resource group have priority over those for a resource unit. If settings for a resource group are omitted, those for a resource unit are applied.

Table 3.32 Setting items of custom resources (CustomResource subsection)

Item name	Definition contents	Specifiable value	Default value
Name	Custom resource name (*1)(*2)	Character string with 1 to 63 characters, consisting of single-byte alphanumeric characters (lowercase), hyphen, and underscore. The first character must be a single-byte alphanumeric character.	Not omissible
ValueType	Custom resource management pattern	numeric: Manage custom resources with a numerical value. string: Get a selection from several resources.	Not omissible
Value	If ValueType is set to numeric: Quantity of custom resources (*3) For custom resources per node (when a node ID is specified), this definition specifies the quantity of resources per node. For custom resources per resource unit or resource group (when no node ID is specified), this definition specifies the quantity of resources per resource unit or resource group.	Value from 1 to 999999999999, or "unlimited"	Not omissible
	If ValueType is set to string: Type of custom resources	Character string consisting of single-byte alphanumeric characters, hyphen, and underscore You can specify multiple strings by delimiting them with a comma (","). The length of one item ranges from 1 to 63 characters. In the entire item, you can specify up to 511 characters (including commas and space characters). Cannot contain only 1 hyphen character "-". You can insert a space character only before and after a comma.	
NodeID	Node ID (*2) Define this item when defining custom resources per node.	Specify node IDs in the following format. - Delimit multiple node IDs with ",". - You can specify a range of node IDs with a hyphen "-". You can specify multiple ranges separated by commas ",". (Example: *nodeid1-nodeid2*, *nodeid3-nodeid4*) However, one range must be in the same boot group for FX server or node group for PRIMERGY server. This item may appear more than once in the same CustomResource subsection.	Omissible (*4)

(*1)
Up to 64 custom resource names can be specified. Each must be unique in the pmpjm.conf configuration file.
The custom resource name "sys-power" is a reserved word. System power consumption is predefined as the custom resource sys-power in order to use the power cap scheduling function. This resource name cannot be used as the name of another custom resource.

(*2)
In cases with custom resources per node (that is, custom resources are defined for every node with the NodeID item), we recommend defining custom resource names in the "*CustomResourceName*-per-node" format. This helps determine whether a specific custom resource is a custom resource per node, based on the custom resource name.
Also, if this item is not defined, the resource is a custom resource per resource unit or resource group.

(*3)
When defining system power consumption as a custom resource to use the power cap scheduling function, set the value defined in the value item as the upper limit value on power consumption increasing due to job execution. You can calculate this value with the following formula.

```
Upper limit value on power consumption increasing due to job execution =
allowable power for all compute nodes in resource unit - idle power for a compute nodes(BasePowerIdle)
x number of compute nodes in resource unit
```

Idle power (BasePowerIdle) of a compute node is the power consumption of the compute node when no job is running. Use the value recorded in the /etc/opt/FJSVtcs/pwrm/base.resource unit name file on the system management node.
This file is created when the power management function is configured (when the papwrmgradm command is executed). It records the minimum idle power of compute nodes in a resource unit. For details, see "Power Cap Scheduling Function" in "Chapter 2 Details of the Power Management Function" in "Job Operation Software Administrator's Guide for PowerAPI."

(*4)
If the setting of item NodeID is omitted, it becomes a custom resource on a per-resource unit or per-resource group basis.

3.5.1.7 Settings for the job manager exit function and the job scheduler exit function

You can set the job manager exit function and job scheduler exit function in the pmpjm.conf file. For details on the settings, see "Job manager exit function and Job scheduler exit function" in "Chapter 2 Creating and Incorporating Hooks" in "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

3.5.1.8 Settings for Scheduler Plug-in

You can set the scheduler plug-in in the pmpjm.conf file. For details on the settings, see "Chapter 2 Use of Scheduler Plugin Functions" in "Job Operation Software API user's Guide for Scheduler API."

3.5.1.9 Reflecting and referencing the pmpjm.conf file

After setting the pmpjm.conf file, you need to execute the pmpjmadm command on the system management node so that the system reflects the setting contents.

```
[System management node]
# pmpjmadm -c clstname --set --rscunit unit1
```

The above operation incorporates settings immediately to reflect them in operation. Neither a node restart nor a complete system restart is required.

Note

If the pjstat command display items are changed by the custom resource setting, jobs previously in the EXIT/CANCEL/REJECT state are deleted from the list of jobs output by the -H option of the pjstat command. The information output by the pmdumpjobinfo command is not affected. For details on the settings, see "3.4.2.4 Custom resource item name."

To display the current settings, use the pmpjmadm command with the --show option.

```
[System management node]
# pmpjmadm -c clusterA --show --rscunit unit1
ResourceUnit {
    ResourceUnitName = unit1
    LogLevel = 1
    Backfill = yes
    DecidedGap = 00:01:00
    Grace = 00:00:10
    CreateRscMap = 01:00:00, 00:10:00
    CreateRscMap = 24:00:00, 00:30:00
    CreateRscMap = 240:00:00, 01:00:00
    CreateRscMap = *, 24:00:00
    SchedulePeriod = 240:00:00
    RestartNormal = yes
    RestartStep = yes
    RestartBulk = no
    MailSend = yes
    Fairshare = off
    FshareRecoveryValue = 236
    AllocType = node
    BackfillTarget = rscgrp
    StartTimeGuarantee = on
    JobStderrMsgLevel = 0
    HoldAcceptDate = release
    AdaptiveElapsedTimeJobTerminateGrace = 00:00:10
    SchedulerPluginLoadPath = /etc/opt/FJSVtcs/plugin/pjm/pjsd/normal_mode/
    DynamicSchedulePeriod = 2,24:00:00
    JobSchedulingTargetLimit = 10000
    JobSchedulingTargetMode = jobselectpolicy
    ResourceGroup {
        ResourceGroupName = group1
        ResourceGroupNode = 100
        ResourceGroupPrio = 127
        ResourceGroupTsha = 2
        ResourceGroupExecPolicy = simplex
        Backfill = yes
        StartTimeGuarantee = on
        JobStderrMsgLevel = 2
        JobSelectPolicy {
            name = policy2
        }
    }
    PrologueEpilogue {
        ShellName = /bin/sh
        ExecUser = ROOT
        PrologueName = /work/prologue
        EpilogueName = /work/epilogue
        PrologueTime = 300
        EpilogueTime = 180
        ContainElapse = no
    }
    JobSelectPolicy {
        name = policy1
    }
}
```

3.5.2 Job resource management function settings in a resource unit (pmrsc.conf file)

The job operation administrator can make the following settings for the job operation management function in the pmrsc.conf file (System management node: /etc/opt/FJSVtcs/Rscunit.d/*rscuname*/pmrsc.conf) by resource unit:

- Maximum time to wait for a response in communication between nodes of the job resource manager
- The percentage of memory for jobs in a compute node

- Setting for periodically collecting interval of the job statistical information
- Settings for the job resource manager exit function (can be set for each resource unit and resource group)

The following example shows settings of the pmrsc.conf file.

```
[System management node]
# cat /etc/opt/FJSVtcs/Rscunit.d/unit1/pmrsc.conf
Cluster {
    ClusterName = clusterA        <- Cluster name
    ResourceUnit {
        ResourceUnitName = unit1
        RespWaitTime = 180
        JobMem = 85
        RscWatchInterval = 15
        ExitFunc {
            ExitFuncTimer = 10
            ExitFuncScriptDir = /work/hook1
            ExitFuncPri = 100
        }
        ResourceGroup {
            ResourceGroupName = groupA
            ExitFunc {
                ExitFuncTimer = 30
                ExitFuncScriptDir = /work/hook2
                ExitFuncPri = 120
            }
        }
    }
}
```

For details, see the man page for the pmrsc.conf file.

Note

You cannot change settings related to job operations simply by creating or editing the pmrsc.conf file. As described below, the pmrscadm command reflects the contents of the pmrsc.conf file in job operations.

Information

There is no pmrsc.conf file when the Job Operation Software is installed. If you want to change the default settings, create a new file or copy and edit the sample file (/etc/opt/FJSVtcs/sample/pmrsc.conf), and place the file in the above path.

3.5.2.1 Settings for job resource management function in a resource unit

You can make the following settings for the job resource management function in the pmrsc.conf file.

Table 3.33 Job resource manager setting (Cluster section)

Item name	Definition contents	Specifiable value	Default value
ClusterName	Setting of a target cluster name	Character string	All cluster names are targets.

The job resource management function settings in a resource unit are written in the ResourceUnit subsection of the Cluster section.

Table 3.34 Job resource manager setting (ResourceUnit subsection)

Item name	Definition contents	Specifiable value	Default value
ResourceUnitName	Target resource unit name	(*)	Not omissible

Item name	Definition contents	Specifiable value	Default value
JobMem	Setting of the percentage of memory for jobs that is available to compute nodes (%)	1-100 (Can be specified to the first decimal place.) The OS may hang up if the JobMem value is greater than 90.	90
RscWatchInterval	The periodically collecting interval (unit: minutes) of the job statistical information The time of collection is on a per-job basis, and not all jobs are collected at the same time.	0-1440 (If it is 0, do not collect.)	10
RespWaitTime	Setting of the maximum time (unit: minutes) during which the job resource management function waits for an inter-node communication response	120 - 1440	120

(*) Specifies the resource unit name identified in the following command.

```
# pashowclst -c clstname --rscunit
```

The job resource manager settings in a resource group are written in the ResourceGroup subsection. Currently, the only resource group-related setting is that for the resource manager exit function, which is described below.

Table 3.35 Job resource manager setting (ResourceGroup subsection)

Item name	Definition contents	Specifiable value	Default value
ResourceGroupName	Target resource group name	Character string with 1 to 63 characters, consisting of alphanumeric characters, hyphen, and underscore	Not omissible

To use the job resource manager exit function, it must be set in the ExitFunc subsection in the ResourceUnit or ResourceGroup subsection. For details, see "Job resource management exit function" in "Chapter 2 Creating and Incorporating Hooks" in "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

Note

- If the value of the JobMem item is too large, the setting may fail. In this case, jobs cannot be executed until a value could be set normally, so job operations are affected. Exercise caution when changing this value. Normally, the JobMem item does not need to be changed. Consult with a Fujitsu systems engineer (SE) or Fujitsu Support Desk to tune up job employment according to the use situation of a system.
- If a node does not return a response even when the time set for the item RespWaitTime elapses, the node is isolated from operation.
- When using the job resource manager exit function (exit script), you need to pay attention to the value of the RespWaitTime item and the exit script processing time. For details, see "Job resource management exit function" in "Chapter 2 Creating and Incorporating Hooks" in "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

3.5.2.2 Reflecting and referencing the pmrsc.conf file

After setting the pmrsc.conf file, you need to execute the pmrscadm command on the system management node so that the system reflects the setting contents.

```
[System management node]
# pmrscadm --set --rscunit unit1
```

The above operation incorporates settings immediately to reflect them in operation. Neither a node restart nor a complete system restart is required (except the changes of the JobMem).

When the JobMem item is changed, restart the compute cluster management node, and all compute nodes of the cluster to which the resource unit belongs after executing the pmrscadm command. Then the value of the JobMem item is applied to each node.

Information

If it is failed to apply the setting to some compute nodes or if new compute nodes are added, the setting can be applied again by specifying only such compute nodes using the -f option of the pmrscadm command. In this case, all compute nodes in a cluster must have the same settings without specifying a different setting to each compute node using the -f option.
When changing the setting of the JobMem item with the -f option of the pmrscadm command, only the target compute nodes need to be restarted.

To display the current settings, use the pmrscadm command with the --show option.

```
[System management node]
# pmrscadm --show --rscunit unit1
Cluster {
    ClusterName = clusterA
    ResourceUnit {
        ResourceUnitName = unit1
        RespWaitTime = 180
        JobMem = 85
        RscWatchInterval = 15
        ExitFunc {
            ExitFuncTimer = 10
            ExitFuncScriptDir = /work/hook1
            ExitFuncPri = 100
        }
        ResourceGroup {
            ResourceGroupName = groupA
            ExitFunc {
                ExitFuncTimer = 30
                ExitFuncScriptDir = /work/hook2
                ExitFuncPri = 120
            }
        }
    }
}
```

3.5.3 Job ACL function settings in a resource unit

The job operation administrator makes job ACL function settings in the resource unit that is a management target.

For details on how to make the settings, see "3.4.4 Job ACL function settings in a cluster".

Note

The cluster administrator may have made settings for some resource units and resource groups. Confirm the settings beforehand so that the cluster administrator's settings do not conflict with the job operation administrator's settings.

Information

If you do not set the job ACL function, the initial values set when the job operation management function was installed are applied. For information about the initial values of the job ACL feature, see "Appendix F Defined items of the job ACL function."

The following example shows job ACL function settings.

```
USER: CL, RU=rscunit1 {
    user=<def> {                                                  <- (1)
```

```
        define rscgroup          rscgroup1

        limit  ru-accept                    10
        limit  ru-run-job                   10
        limit  rg-custom-customrscname1     6

        joblimit  elapse                    1:00:00    24:00:00   10:00:00
        joblimit  node                      1          5000       1
        joblimit  customrscname2            100000     5000000    100000

        select    customrscname3            a,b        a
    }
    user=<def>:group2 {                                           <- (2)
        define rscgroup          rscgroup2
    }
    user=user1 {                                                  <- (3)
        permit pjstat        allow all
        permit pjdel         allow all
        permit pjhold        allow all
        permit pjacl         allow all
    }
    user=user1:group3 {                                           <- (4)
        define rscgroup          rscgroup3
    }

}
GROUP: CL, RU=rscunit1 {
    group=<def> {                                                 <- (5)
        limit  ru-use-node       50
    }
    group=group1 {                                                <- (6)
        define pri-g             160
    }
}
ALL: CL, RU=rscunit1  {
        limit  ru-accept         500
}

USER: CL, RU=rscunit1, RG=rscgroup1 {
    user=<def> {                                                  <- (7)
        permit pjstat        allow all
    }
    user=<def>:group1 {                                           <- (8)
       permit pjstat        allow g(group1)
    }
    user=user2 {                                                  <- (9)
        permit pjstat        allow all
        permit pjdel         allow all
        permit pjhold        allow all
        permit pjacl         allow all
    }
}
GROUP: CL, RU=rscunit1, RG=rscgroup1 {
    group=<def> {                                                 <- (10)
        define pri-g           200
    }
    group=group3 {                                                <- (11)
        define pri-g           64
    }
}
ALL: CL, RU=rscunit1, RG=rscgroup1  {
        limit  ru-accept         200
}
```

(1) Default user definitions
(2) Default user definitions for the group group2
(3) Specific definitions for the user user1
(4) Specific definitions only if the target group of the user user1 is group3
(5) Default group definitions
(6) Group-specific definitions for the target group group1
(7) Default user definitions
(8) Default user definitions for the target group group1
(9) Specific definitions for the user user2
(10) Default group definitions
(11) Group-specific definitions for the target group group3

(*) The names of defined custom resources are displayed at *customrscname1* to *customrscname3*.

Make job ACL function settings in a resource group in the same way as for job ACL function settings in a cluster. For details, see "3.4.4 Job ACL function settings in a cluster".

3.5.4 Incorporating the job manager exit function and job scheduler exit function

To use the job manager exit function and the job scheduler exit function, an exit function library must be created and incorporated into the job operation management function. For details, see "Job manager exit function and Job scheduler exit function" in "Chapter 2 Creating and Incorporating Hooks" in "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

3.5.5 Incorporating the job resource manager exit function

To use the job resource manager exit function, an exit script must be created and incorporated into the job operation management function. For details, see "Job resource management exit function" in "Chapter 2 Creating and Incorporating Hooks" in "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

3.5.6 Customizing the display by the pjstat command

The administrator can set whether to display just some of the items displayed by the pjstat command according to the job operation.

Job information list displayed by pjstat and pjstat -v

The administrator can set whether to display the following virtual node allocation-related items.

```
VNODE, CORE, V_MEM, V_POL, E_POL, RANK
```

They are displayed by default. However, these items can be set to not be displayed for job operations that do not use virtual nodes.

```
[Displayed]
$ pjstat

JOB_ID    JOB_NAME   MD ST  USER    START_DATE      ELAPSE_LIM      NODE_REQUIRE   VNODE  CORE V_MEM
238       job.sh     NM RUN user1   11/17 09:01:41  0001:00:00      12:2x3x2       -       -    -
239       bulk.sh    BU RUN user1   11/17 09:01:42  0001:00:00      12:2x3x2       -       -    -
240       step.sh    ST RUN user1   11/17 09:01:42  -               -              -       -    -
241       job2.sh    NM RUN user1   11/17 09:01:42  0001:00:00      2              -       -    -

[Not displayed]
$ pjstat

JOB_ID    JOB_NAME   MD ST  USER    START_DATE      ELAPSE_LIM      NODE_REQUIRE
238       job.sh     NM RUN user1   11/17 09:01:41  0001:00:00      12:2x3x2
239       bulk.sh    BU RUN user1   11/17 09:01:42  0001:00:00      12:2x3x2
240       step.sh    ST RUN user1   11/17 09:01:42  -               -
241       job2.sh    NM RUN user1   11/17 09:01:42  0001:00:00      2
```

This setting has no effect on what is displayed by the -s or -S option of the pjstat command or the contents of the .stats file output by the -s or -S option of the pjsub command. This information is output according to the job statistical information settings (see "3.4.2 Settings for job statistical information in a cluster (papjmstats.conf file)").

See

This function is set with the JOBINFO_PRINT_VNODE_ITEMS item in the command API configuration file pmpjcmd.conf. For details, see "3.5.8 Command API settings."

Display by the -s or -S option of the pjstat command

The administrator can select which job statistical information items are displayed by the -s or -S option of the pjstat command.

See

This function is defined in the Command subsection in the papjmstats.conf configuration file of job statistical information. For details, see "3.4.2.2 Definitions of output items in job statistical information."

3.5.7 Configuring a Job Execution Environment

The following settings are necessary for using a job execution environment: After setting, notify the user of the operation information such as resource name of the job execution environment and whether SDI/UDI specification is used or not if necessary.

- Preparing an image file
- Registering an image file
- Creating a job execution environment information file
- Creating a container startup configuration file (Docker mode only)
- Setting custom resources
- Setting the job ACL function
- Configuring UDI specifications
- Setting the job manager exit function

Note

KVM mode is not available on the systems with a single node serving as all of the system management node, compute cluster management node, and login node.

This section describes these settings.

3.5.7.1 Preparing an image file

To build a job execution environment, an image file appropriate to the job execution environment must be prepared. For the procedure to build a job execution environment, see "Creating an Image File for a Job Execution Environment" in Appendix "Using Job Execution Environment" in "Job Operation Software End-user's Guide."

Note

When you use the Docker mode, make sure enough disk space to store all of the container image is available. The container image must be stored in disk space where each compute node can access, by default the system volume (/var/lib/docker) of each compute node.
When building the system, determine the size of the system volume so that it has enough space to store all of the container images, or set a different disk space for the container image.

In particular, when using UDI specification, the container image specified by the end user is generated in conjunction with job execution. To avoid tightness of the system volume capacity, it is recommened that the container image be stored in a dedicated disk space that is separate from the system volume.

3.5.7.2 Registering an image file

Docker mode

SDI specifications require that the container image be previously registered with each compute node, such as with docker import command.
UDI specifications do not require prior registration of the container image. Container images are manipulated by the job resource manager exit function. For configuring of UDI specifications, see "3.5.7.7 Configuring the job resource manager exit scripts (Docker mode only)."

KVM mode [PG]

For the SDI specification, put the virtual machine image file on the shared file system used for job execution.
The UDI specification does not require a prior placement of the virtual machine image.

3.5.7.3 Creating a job execution environment information file

Create a job execution environment information file containing information on the job execution environment to be used, and apply it to the system. The administrator is requested to make the following settings on the active system management node.

1. Creating a job execution environment information file

   Create a file named jobenv.conf as the job execution environment information file, and write the information necessary for starting the job execution environment. In the job execution environment information file, write job execution environment entries that the job execution user can select. Write the file in JSON format (JavaScript Object Notation). Write the following items as objects in this file.

   Information

   On FX servers, the jobenv.conf file is installed as /etc/opt/FJSVtcs/krm/jobenv.conf when the Job Operation Software is installed. For the following settings, copy the jobenv.conf file from any FX server to the active system management node, and edit it there.

Table 3.36 Setting items in the job execution environment information file

Item name/key	Key	Description	Default Value
Job execution environment name	Name	Name representing the job execution environment A character string with 1 to 63 characters consisting of single-byte alphanumeric characters (lowercase), hyphen, and underscore can be specified. However, the first character must be a single-byte alphanumeric character. The specified name cannot be a duplicate of any other entry.	Not omissible
Job execution environment type	Type	Type of job execution environment Specify one of the following character strings according to the job execution environment: Normal mode: "docker" Docker mode: "docker" KVM mode: "kvm" KVM mode is available on systems where the user's home directory is shared between the login and compute nodes. However, KVM mode is not available on systems with a single node serving as all of the system management	Not omissible

Item name/key	Key	Description	Default Value
		node, compute cluster management node, and login node.	
Image specification	Image	Container image used Normal mode: Specify "tcs-bare". Docker mode: Specify the container image registered with Docker on the compute node (for example, with docker import) in the following format: [*REGISTRY_HOST*[:*REGISTRY_PORT*]/]*NAME* [:*TAG*] If NeedCustomImage is specified as true, specify “tcs-bare” KVM mode: Specifies the absolute path name of the virtual machine image file placed on the shared file system used for job execution.	Not omissible
Whether to use UDI pecifications	NeedCustomImage	Whether to use UDI specifications true: Use the job execution environment image specified in the pjsub command at the job submission time (*1) false: Use the container image specified in Image	false
Startup configuration file	Conf	Absolute path to the configuration file containing startup options for the job execution environment (*2) This item is supported only in Docker mode.	(*3)
Startup timeout time	Timeout	Timeout time for startup processing of the job execution environment Specify a decimal number in seconds. In normal or Docker mode, this item is invalid.	146
Comment	Comment	Comment on the entry	None

(*1)
For details on how to specify this in the pjsub command, see "Specifying a job execution environment" in "Chapter 2 Job Operation Procedures" in "Job Operation Software End-user's Guide."

(*2)
For the detailed format of this configuration file, see "3.5.7.4 Creating a container startup configuration file (Docker mode only)." It is the administrator's own responsibility to fully understand the specifications of each job execution environment and to make this setting. After setting, distribute this configuration file to each compute node.

(*3)
If this item is omitted, the default configuration file (/etc/opt/FJSVtcs/krm/docker-image.conf) is applied.

The JSON format is a standard format that is commonly used. Tools for format checks are published as free software (e.g., https://jsonlint.com/). We recommend performing a format check in advance with these tools.

The following example shows settings.

```
[System management node]
# cat jobenv.conf
[
    {
        "Name"         : "docker1",                        <- Docker mode settings
        "Type"         : "docker",
        "Image"        : "image1"
    },
    {
        "Name"         : "kvm1",                          <- KVM mode settings
        "Type"         : "kvm",
        "Image"        : "/var/images/kvm.img"
    },
    {
        "Name"            : "UDI1",                       <- UDI specification settings
        "Type"            : "docker",                        in Docker mode
        "NeedCustomImage" : true,
        "Image"           : "tcs-bare"
    }
]
```

2. Setting permissions for the job execution environment information file

   Set permissions for the job execution environment information file as follows.

   ```
   [System management node]
   # chmod 0600 jobenv.conf
   ```

3. Distributing the job execution environment information file

   Distribute the job execution environment information file to the target compute nodes.

   ```
   [System management node]
   # pmscatter -c cluster -p -n NodeID jobenv.conf /etc/opt/FJSVtcs/krm/jobenv.conf
   ```

4. Reflecting the job execution environment information file

   Restart the compute nodes that were the target in the previous step.

   1. Isolate the compute nodes.

      ```
      [System management node]
      # paclstmgr -c cluster scope --disable
      (*) scope : scope option
      ```

   2. Check the status.

      Confirm state transitions by the target nodes to "Disable" in the STATUS field and "Manual" in the REASON field.

      ```
      [System management node]
      # pashowclst -c cluster -v scope
      (*) scope : scope option
      ```

   3. Transition the compute nodes to software maintenance mode.

      ```
      [System management node]
      # paclstmgr --soft-mainte -c cluster scope
      (*) scope: scope option
      ```

   4. Check the status.

      Confirm state transitions by the target nodes to "Disable" in the STATUS field and "SoftMaintenance" in the REASON field.

      ```
      [System management node]
      # pashowclst -c cluster -v scope
      (*) scope: scope option
      ```

   5. Recover the compute nodes.

      Because the job execution environment information file has been distributed, recover the compute nodes by updating the settings through a cold reboot.

      ```
      [System management node]
      # paclstmgr -c cluster scope --recover --cold-reboot
      (*) scope: scope option
      ```

6. Incorporate the compute node into operation.

```
[System management node]
# paclstmgr -c cluster scope --enable
(*) scope : scope option
```

3.5.7.4 Creating a container startup configuration file (Docker mode only)

Docker mode requires a startup configuration file (The file indicated by the item Conf in the job execution environment information file jobenv.conf). The following describes the necessary modifications to the default startup configuration file, docker-image.conf, and how to create a new startup configuration file.

[Modifying the default startup configuration file, docker-image.conf]

The default startup configuration file is installed on each compute node as /etc/opt/FJSVtcs/krm/docker-image.conf. Administrator must first add the following mount point to the "HostConfig" item Bindings in the startup configuration file docker-image.conf. Otherwise, the job will not run correctly in the Docker mode.

- For FX servers

```
"HostConfig" : {
         "Binds" : [
                    "/opt/FJSVxtclanga:/opt/FJSVxtclanga",
                    "/dev/shm:/dev/shm",
                    "/etc/opt/FJSVtcs/ple:/etc/opt/FJSVtcs/ple",
                    "/usr/bin/pjrsh:/usr/bin/pjrsh",
                    "/usr/bin/pjshowip:/usr/bin/pjshowip",
                    "/usr/bin/plestat:/usr/bin/plestat",
                    "/usr/bin/plexec:/usr/bin/plexec",
                    "/usr/lib/FJSVtcs/ple:/usr/lib/FJSVtcs/ple",
                    "/usr/lib64/libjrm.so.1:/usr/lib64/libjrm.so.1",
                    "/usr/lib64/libpel.so.1:/usr/lib64/libpel.so.1",
                    "/usr/lib64/libpmix.so:/usr/lib64/libpmix.so",
                    "/usr/lib64/libpmix.so.2:/usr/lib64/libpmix.so.2",
                    "/usr/lib64/libpmix.so.2.1.14:/usr/lib64/libpmix.so.2.1.14",
                    "/usr/sbin/pleio:/usr/sbin/pleio",
                    "/usr/sbin/plepmix:/usr/sbin/plepmix",
                    "/var/opt/FJSVtcs/ple:/var/opt/FJSVtcs/ple",
                    "/var/log/FJSVtcs/ple:/var/log/FJSVtcs/ple",
                    "userdir:userdir"
                    ],
                  ...
                }
```

userdir : User area on the host OS (job execution directory for users) (ex. /home)

- For PRIMERGY servers

```
"HostConfig" : {
         "Binds" : [
                    "/dev/shm:/dev/shm",
                    "/etc/opt/FJSVtcs/ple:/etc/opt/FJSVtcs/ple",
                    "/usr/bin/mpiexec.tcs_intel:/usr/bin/mpiexec.tcs_intel",
                    "/usr/bin/pjexe:/usr/bin/pjexe",
                    "/usr/bin/pjpbind:/usr/bin/pjpbind",
                    "/usr/bin/pjrsh:/usr/bin/pjrsh",
                    "/usr/bin/pjshowip:/usr/bin/pjshowip",
                    "/usr/bin/plestat:/usr/bin/plestat",
                    "/usr/bin/plexec:/usr/bin/plexec",
                    "/usr/lib64/libjrm.so.1:/usr/lib64/libjrm.so.1",
                    "/usr/lib64/libpel.so.1:/usr/lib64/libpel.so.1",
                    "/usr/sbin/pleio:/usr/sbin/pleio",
                    "/var/opt/FJSVtcs/ple:/var/opt/FJSVtcs/ple",
                    "/var/log/FJSVtcs/ple:/var/log/FJSVtcs/ple",
                    "userdir:userdir"
                    ],
                  ...
                }
```

userdir : User area on the host OS (job execution directory for users) (ex. /home)

Copy the startup configuration file docker-image.conf for any compute node to the working directory of the system management node, modify it, and distribute it on each compute node.

Note

Do not modify the startup configuration file docker-image.conf on compute nodes except for the above.

[Creating new startup configuration file]

If you create a new startup configuration file, copy and modify the default startup configuration file, docker-image.conf.

1. Copy a default startup configuration file docker-image.conf

   Copy a default startup configuration file (/ etc/opt/FJSVtcs/krm/docker-image.conf) from a compute node to any working directory on the system management node with a different name.

2. Add the mount points

   Add the mount points that need to be referenced from within the container to the copied startup configuration file. Add the necessary mount points to the list of mount points written in "Binds."
   By default, directories on the host OS cannot be referenced. Therefore, the following mount points must be set;

   - Device files (under /dev) and other files that need to be referenced according to the job

   Also, perform the configuration shown in [Modifying the default startup configuration file, docker-image.conf] above.

   The following example sets all device files so that they can be referenced.

```
"HostConfig" : {
         "Binds" : [
                    ...
                    "/dev:/dev"
                    ],
                  ...
                }
```

   See

   The format of the container startup configuration file is the same as in the API specifications for a Docker API create request. For the detailed format, see https://docs.docker.com/develop/sdk/, for example.

3. Set device file permissions

   Permissions must be set for device files. Add specifications to "Devices" as follows. In this example, /dev/sda1 can be referenced from within the container.

```
"Devices" : [{
                    "PathOnHost" : "/dev/sda1",
                    "PathInContainer" : "/dev/sda1",
                    "CgroupPermissions" : "mrw"
             }],
```

   Make settings as follows so as not to restrict permissions on the device files but to allow access to all devices.

```
"Devices" : [{
                   "PathOnHost" : "/dev",
                   "PathInContainer" : "/dev",
                   "CgroupPermissions" : "mrw"
            }],
```

Note

Ensure that the path specified for the PathOnHost item exists on the compute node. If a path that does not exist on the compute node is specified, the job fails and cannot run successfully.

4. Deploy the startup configuration file to the compute nodes

   Deploy the modified startup configuration file to each compute node. The destination path is the path set to the Conf item in the jobenv.conf job execution environment information file.

[About configuring mount points for PRIMERGY servers]

For PRIMERGY servers, by using the "{{ keyword }}" format for a mount point instead of a fixed character string, you can apply values (dynamic parameters) appropriate to the environment at the job runtime.

The current directory "{{ PJM_JOBDIR }}" at the start of job script execution and the group name "/fefs/{{ PJM_GROUP }}" of the job execution user are set as a mount point in the following example.

```
...
"HostConfig" : {
    "Binds" : [
        "{{ PJM_JOBDIR }}:{{ PJM_JOBDIR }}",
        "/fefs/{{ PJM_GROUP }}:/fefs/{{ PJM_GROUP }}"
    ],
    ...
}
...
```

For the above setting, the keywords are replaced as follows at container startup.

```
...
"HostConfig" : {
    "Binds" : [
        "/fefs/user1:/fefs/user1",
        "/fefs/group1:/fefs/group1"
    ],
}
...
```

The following table lists available keywords.

Table 3.37 Keywords that can be specified for the Binds setting item

Keyword	Description	Example of replacement
PJM_JOBNAME	Job name	job.sh
PJM_JOBDIR	Path to the current directory when job script execution begins	/fefs/user1/
PJM_O_HOME	Environment variable HOME of the user executing the pjsub command	/home/user1/
PJM_O_NODEINF	Path to the allocated node list file	/fefs/user1/d0000000100_nodeinfo
PJM_RSCGRP	Resource group name specified in the -L rscgrp (or -L rg) option of the pjsub command	group_a

Keyword	Description	Example of replacement
PJM_RSCUNIT	Resource unit name specified in the -L rscunit (or -L ru) option of the pjsub command	rscunit_pg01
PJM_UID	User ID of the job execution user	1000
PJM_USER	User name of the job execution user	user1
PJM_GID	Group ID of the job execution user	1000
PJM_GROUP	Group name of the job execution user	user1
PJM_APPNAME	Character string specified in the --appname option of the pjsub command However, if ".." is specified, which indicates the directory that is one level higher, an error occurs.	app1
PJM_FSNAME	Character string specified in the --fs option of the pjsub command However, if ".." is specified, which indicates the directory that is one level higher, an error occurs.	fs1
PJM_CUSTOM_RESOURCES.jobenv	Value of the jobenv custom resource specified in the -L option of the pjsub command	docker
PJM_JOBENV_DOCKER_IMAGE	Container image name specified by a user	centos

Note

This specification method that uses dynamic parameters is not available by default. To use it, configure settings as described in "Appendix E How to Use Dynamic Parameters in Startup Configuration Files (Docker Mode) [PG]."

3.5.7.5 Setting custom resources

Define the job execution environment name defined in the job execution environment information file as a custom resource (custom resource name: jobenv). For details on custom resources, see "2.5.5 Job scheduling function using custom resources" and "3.5.1.6 Custom resource settings."

1. Editing the pmpjm.conf file

   Specify the following as definition items in the CustomResource subsection.

Item name	Specification
Name	jobenv
ValueType	string
Value	Job execution environment name defined in the job execution environment information file (jobenv.conf)

   The following example shows settings.

```
[System management node]
# vi /etc/opt/FJSVtcs/Rscunit.d/resourceunit/pmpjm.conf
ResourceUnit {
    ResourceUnitName = resourceunit
    ...
    CustomResource {
        Name = jobenv
        ValueType = string
        Value = linux,docker1,kvm1,UDI1
    }
    ...
}
```

2. Reflecting the pmpjm.conf file

   Execute the following command.

```
[System management node]
# pmpjmadm -c cluster --set --rscunit resourceunit
```

3.5.7.6 Setting the job ACL function

The job ACL function can restrict specification of a job execution environment name defined as a custom resource and set default values. For details on the job ACL function, see "2.4.1.2 Job ACL function,""3.4.4 Job ACL function settings in a cluster," and "3.5.3 Job ACL function settings in a resource unit."

1. Editing the configuration file of the job ACL function

   Set specifiable types and default values for the job execution environment in the "select custom-jobenv" definition item.

   The following example shows settings.

```
[System management node or Compute cluster management node]
# vi jobacl.txt
USER: CL, RU=resourceunit, RG=resourcegroup {
    user=<def> {
    select    custom-jobenv       linux,docker1,kvm1,UDI1     linux
    }
}
```

2. Reflecting job ACL definitions

   Execute the following command.

```
[System management node or Compute cluster management node]
# pmjacladm -c cluster --set -f jobacl.txt
```

3.5.7.7 Configuring the job resource manager exit scripts (Docker mode only)

In addition to setting the item NeedCustomImage to true in the jobenv.conf file, you must set the job resource manager exit scripts to take advantage of UDI specification in Docker mode. The administrator must perform the following tasks on the active system management node.
For KVM mode, configuring the job resource manager exit scripts is not necessary.

Note

Note the following for the configuring:

- This setting uses the job resource manager exit function (prealloc and postfree scripts). In UDI specifications, the job resource manager exit function operates container images. For details on the job resource manager exit function, see "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

- Before using UDI specifications, be sure to thoroughly read the notes written in "4.3.2 Job Execution Environment Customization Function."

- If the environment variable PJM_JOBENV_DOCKER_IMAGE is specified outside of the UDI specifications after these settings are made, jobs end with job end code (PJM code) 27 and then transition to the ERROR state.

- Only one job resource manager exit script for UDI specification can be registered per resource unit. If you registered more than one exit script, the job ends with job end code (PJM Code) 27 and transitions to the ERROR state.

Creating job resource manager exit scripts to be used when specifying UDI

When UDI is specified, in Docker mode, the job resource manager exit script performs the necessary operations.

Be sure to use the prealloc and postfree scripts to register with the job resource manager exit function by copying the sample files prealloc.docker-udi and postfree.docker-udi under /etc/opt/FJSVtcs/krm/sample/, respectively. Also, do not modify these sample scripts except for the path settings of the commands described in the note below and the settings listed below.

Note

The sample script shows the paths of the commands to use in the script. Ensure that these commands are available on the compute node. If the path of the command is different from the path used on the compute node, change the following path specification in the script as appropriate.

```
PYTHON='/usr/bin/python'
DOCKER='/usr/bin/docker'
SYSTEMCTL='/usr/bin/systemctl'
CAT='/usr/bin/cat'
TIMEOUT='/usr/bin/timeout'
FILE='/usr/bin/file'
PRINTF='/usr/bin/printf'
CURL='/usr/bin/curl'
```

In the prealloc script, the following settings are available.

Selecting a container specification method

Set "IMAGE_TYPE_NUM" in the prealloc script as follows according to the container specification method selected at the job submission time.

- Exported tar file specified

  ```
  IMAGE_TYPE_NUM=0
  ```

  Specify the absolute path to an image in the file format generated by the docker export command, etc. at the job submission time.

- Repository specified

  ```
  IMAGE_TYPE_NUM=1
  ```

  Specify the repository registered in advance with the repository containing jobs at the job submission time.
  For the repository specified, add authentication settings (docker login command, etc.) according to the authentication settings of the prepared repository after the following code in the prealloc script.

  ```
  ###
  # Add Authentication Settings here if needed.
  ###
  ```

Setting of the Image Operation Timeout

The prealloc script you edit in above already contains a timeout value (Units: Seconds) for the docker command's image operations.

```
DOCKER_IMPORT_TIMEOUT_SEC=600
DOCKER_PULL_TIMEOUT_SEC=600
DOCKER_TAG_TIMEOUT_SEC=600
DOCKER_RMI_TIMEOUT_SEC=600
```

The variable "DOCKER_*XXXX*_TIMEOUT_SEC" refers to the timeout value of the image operation "docker *XXXX*" in the docker command.
If the image you are using is large and the prealloc script processing causes a timeout for the docker image operation, change the settings

for DOCKER_IMPORT_TIMEOUT_SEC and DOCKER_PULL_TIMEOUT_SEC as necessary.
The timeout value (the item ExitFuncTimer in the pmrsc.conf file) for the execution time of the prealloc script must be greater than the setting of DOCKER_IMPORT_TIMEOUT_SEC and DOCKER_PULL_TIMEOUT_SEC.
The followings are recommended settings.

Table 3.38 The recommended timeout values for the image operation with the docker command

Image Size	Settings for DOCKER_IMPORT_TIMEOUT_SEC, DOCKER_PULL_TIMEOUT_SEC	Setting for ExitFuncTimer
1GiB or less	600 seconds (Default value)	610 seconds
2GiB or less	1200 seconds	1210 seconds
More	1790 seconds	1800 seconds (*)

(*) The maximum value for ExitFuncTimer is 1800 seconds.

Registering the job resource manager exit script

Register the job resource manager exit scripts for Docker mode (prealloc and postfree script) edited in the above to the job resource manager exit function.
For details on the job resource manager exit function, see "Job Operation Software Administrator's Guide for Job Operation Manager Hook."
The job ACL function can restrict specification of a job execution environment name defined as a custom resource and set default values.
For details on the job ACL function, see "3.4.4 Job ACL function settings in a cluster" and "3.5.3 Job ACL function settings in a resource unit."

1. Placing the exit script

   Place the exit script in any directory on the active system management node.

2. Setting the job resource manager exit function in the pmrsc.conf file

   Set the function by writing settings in the ExitFunc subsection in the pmrsc.conf configuration file of the job resource manager. If multiple ExitFunc subsections are set, give the highest priority to the exit script so that it will definitely be executed regardless of the results of the subsections.

   The following example shows settings in the pmrsc.conf file.

```
[System management node]
# cat /etc/opt/FJSVtcs/Rscunit.d/resourceunit/pmrsc.conf
Cluster {
    ClusterName = cluster
    ResourceUnit {
        ResourceUnitName = resourceunit
        ExitFunc {
            ExitFuncTimer = 60                 <- Timeout value
            ExitFuncScriptDir = /work/udi      <- Directory where exit script is placed
            ExitFuncPri = 100                  <- Priority of exit script
        }
    }
}
```

   See

   For details on how to set the job resource manager exit function in the pmrsc.conf file, see "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

3. Incorporating into job operations

   Reflect the set contents of the pmrsc.conf file in the system with the pmrscadm command. Perform this work from the system management node.

```
[System management node]
# pmrscadm --set --rscunit resourceunit
```

3.5.7.8 Setting the job manager exit function

The pjmx_quejob() exit function of the job manager exit function can check end-user specified environment variables (PJM_JOBENV_DOCKER_IMAGE and PJM_JOBENV_KVM_IMAGE) to determine whether to permit acceptance of a job.

To check the environment variables, the administrator is requested to create and register an exit function as necessary by following the procedure below.

See

For details on the job manager exit function, see "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

1. Creating the source file of the pjmx_quejob() exit function

   Modify the source file (/usr/src/FJSVtcs/pjm/hook/pjmx/pjmx_quejob.c) of the pjmx_quejob() exit function to create a process for checking environment variables.

   Environment variables can be referenced with the job information acquisition function pjmx_getinfo_envdata(). In addition, upon detection of an error, the job information setting function pjmsx_setinfo_reason() can set a reason for the error.

2. Installing an exit function library

   Create and install an exit function library on the active compute cluster management node.

   The following example shows the steps.

   1. Create a Makefile.

      Using the sample Makefile as a reference, specify the pjmx_quejob.o object file corresponding to the source file created in PJMXLIBOBJS, and specify an exit function library name in PJMXLIBNAME. In the following example, the exit function library name is libpjmx-jobenv.so.

      ```
      PJMXLIBOBJS = \
          pjmx_quejob.o
      PJMXLIBNAME = pjmx-jobenv   (*) File name for created exit library: libpjmx-jobenv.so
      ```

      Note

      Give a unique name to the exit function library to avoid duplication with other exit library names.

   2. Install the exit function library.

      Create the exit function library as follows. Perform this work with root privileges.

      ```
      [Active compute cluster management node]
      # make install
      ```

3. Incorporating the exit function library into job operation

   To incorporate the installed exit function library into job operation, configure the pmpjm.conf file and execute the pmpjmadm command on the system management node.

   1. Set the job manager exit function in the pmpjm.conf file.

      Set the function by writing settings in the ExitFunc subsection in pmpjm.conf. The following example shows settings.

      ```
      [System management node]
      # cat /etc/opt/FJSVtcs/Rscunit.d/resourceunit/pmpjm.conf
      ResourceUnit {
          ResourceUnitName = resourceunit
          ExitFunc {
      ```

```
        ExitFuncLib = libpjmx-jobenv.so  <- Specifies exit function library
        ExitFuncPri = 127                <- Execution priority of exit function
        ExitFuncType = pjm               <- Type of exit function
    }
}
```

Note

Specify pjm in the ExitFuncType type of the exit function.
For details on how to set the exit function in the pmpjm.conf file, see "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

2. Incorporate it into job operations.

   Incorporate the installed exit function library into job operations by using the pmpjmadm command to reflect the contents of the pmpjm.conf file to the system.

   ```
   [System management node]
   # pmpjmadm -c clstname --set --rscunit unit1
   ```

3.5.7.9 Note on the setting time

Compute nodes must be restarted to reflect updates of the job execution environment information file jobenv.conf.

3.5.7.10 Use of Singularity [PG]

Singularity is container virtualization software for HPC. The Job Operation Software can execute Singularity in Docker mode in a job execution environment (only for PRIMERGY server). To do so, the job execution environment and custom resources must be configured. For details, see "Appendix D Settings for Using Singularity [PG]."

3.5.8 Command API settings

The following settings relate to using the command API:

- Authority for using the command API

- Parameters relating to command API operations

The execute command-api definition item of the job ACL function controls whether to permit use of the command API. For details on the setting method, see "3.4.4 Job ACL function settings in a cluster."

If you want to change parameters relating to command API operations, edit the pmpjcmd.conf files on the login, compute cluster management, and system management nodes that can use the command API.

```
/etc/opt/FJSVtcs/pjm/pmpjcmd.conf
```

Information

There is no pmpjcmd.conf file when the Job Operation Software is installed. If you want to change the default settings, create a new file or copy and edit the sample file (/etc/opt/FJSVtcs/sample/pmpjcmd.conf), and place the file in the above path on the node where the settings should be applied.

The contents of the pmpjcmd.conf file are updated every time a command that uses a command API function is executed. The command API loads configuration files from the called nodes, so if you want each node to have the same set values, make sure that configuration files on the respective nodes have the same contents.

Set the pmpjcmd.conf file permissions as follows:

- Owner/group: root/root

- File mode: 0644

The format of the pmpjcmd.conf file is as follows.

```
OwnNode {
    Parameter=Value
    ...
}
```

You can set the following parameters.

Table 3.39 Parameters specifying command API operations

Parameter	Description
EXECUTE_INTERVAL	Minimum time interval between operation requests from the command API to the job operation management function (milliseconds) If the parameter is not set, the value of the compute cluster management node is applied. If no value is set for the compute cluster management node too, the default value of 100 milliseconds is applied. The administrator sets this parameter to restrict the time interval between calls to prevent numerous operation requests from being issued to the job operation management function in a short time. If the time specified in this parameter has not elapsed since the the command API function pjcmd_operation_execute() was last called to issue an operation request to the job operation management function, requests sleep within the function until the time has elapsed. This parameter does not restrict the interval between calls from other than the pjcmd_operation_execute() function. The upper limit is 1,000 (milliseconds). If 0 is specified, operation requests are issued without waiting.
MAX_JOB_RESULT_NUM	Upper limit on the number of jobs included in job operation results If the parameter is not set, the value of the compute cluster management node is applied. If no value is set for the compute cluster management node too, the default value of 100,000 jobs is applied. The job operation API (submit, delete, hold, release, send signal, wait for completion, and change job parameter) returns the results of individual jobs. The command API can instruct that the results of jobs, including non-existing jobs, be returned when a range of job IDs is specified as an operation target. In these cases, the amount of information increases, and traffic with the job operation management function may increase. The administrator sets this parameter to restrict the amount of operation result data. If jobs beyond this value are included as operation targets, the operations are performed but the results of extra jobs are not returned. The operation results of jobs are returned in order from the smallest job ID until the number of jobs reaches the maximum MAX_JOB_RESULT_NUM value.
JOBINFO_PRINT_VNODE_ITEMS	Specifies whether to display virtual node allocation-related items (*) when the job information level is 0 or 1 in the job information acquisition function pjcmd_jobinfo_print_resp(). Use this parameter when you do not want to display unnecessary items in systems that do not use virtual node allocation. yes: Display (Default) no: Do not display (*) VNODE, CORE, V_MEM, V_POL, E_POL, RANK

3.6 Setting Log Rotation

The log rotation function (logrotate) of the OS backs up the log files and job statistic information files that are output while the job operation management function operates because the size of the files increases as time elapses.

If you need to change the amount of logs saved for a long period of time to investigate errors, edit the following configuration files. The setting can be changed during operation. For details on how to write a configuration file, see the logrotate command manual of the OS.

pjmd log file (/var/log/FJSVtcs/pjm/pjmd.log)

The pjmd log file is a log file output by the daemon (pjmd) of the job manager function.
This file is checked once every hour. If the size exceeds 50 MB, the file is backed up. The files for a maximum of 10 generations are stored in the /var/opt/FJSVtcs/pjm directory.
To change backup settings, edit the /etc/opt/FJSVtcs/logrotate.d/pjmlogd file on the compute cluster management node.

pjsd log file (/var/log/FJSVtcs/pjm/pjsd*.log)

The pjsd log file is a log file output by the daemon (pjsd) of the job scheduler function.
This file is checked once every hour. If the size exceeds 20 MB, the file is backed up. The files for a maximum of 10 generations are stored in the /var/opt/FJSVtcs/pjm directory.
To change backup settings, edit the /etc/opt/FJSVtcs/logrotate.d/pjslogd file on the compute cluster management node.

Job statistic information file (/var/opt/FJSVtcs/shared_disk/pjm/jsti/jobinfo)

This file contains information, including resource usage by jobs and various limit values.
The file is backed up once a day, and the /var/opt/FJSVtcs/shared_disk/pjm/jsti directory on the compute cluster management node stores the files from the past 31 days.

To change backup settings, edit the /etc/opt/FJSVtcs/logrotate.d/pjmd file on the compute cluster management node. Edit the file so that it is inclded in the rotation when the path of the job statistical information file was changed or added (See "3.4.2.3 Path to a job statistical information file").

jobrscusage file (/var/opt/FJSVtcs/prm/jobrscusage)

This file contains job statistical information collected periodically by the job resource manager.
The file is checked once a day, and it is backed up if the size exceeds 100 MB. The /var/opt/FJSVtcs/prm directory stores up to 10 generations of files.
To change backup settings, edit the /etc/opt/FJSVtcs/logrotate.d/jobrscusage file on the compute cluster management node.

Note

Do not change the location of any log files other than the job statistics information file.

The following table shows the rate of increase in the file size. Refer to the table when estimating the check interval of the log size or the number of stored generations.

Table 3.40 Increase rate of a log file

File	Node	Increase rate
pjmd log file pjsd log file	Compute cluster management node	The rate of increase in the log file size varies depending on the job operation status and the set log level (Item 'LogLevel') in the papjm.conf and pmpjm.conf files. Assuming that the number of jobs submitted per hour is n, the rate of increase in the log file size per hour is as follows: - Log level 1: 0.03*n MB - Log level 2: 0.14*n MB - Log level 3: 0.49*n MB
Job statistic information file	Compute cluster management node	Assuming that the number of jobs submitted per day is n, and an average of y compute nodes is used per job, then the rate of increase in the job statistic information file size per day is (6.4+1.6*y)*n (KB).

File	Node	Increase rate
jobrscusage file	Compute cluster management node	Assuming that the value of the RscWatchInterval item in the mrsc.conf configuration file of the job resource manager is *t*, the number of jobs executed per hour is *n*, and the average number of compute nodes used per job is *y*, then the rate of increase in the jobrscusage file size per hour is: {(280**n*)+(420**n***y*)}*60/*t* Bytes

Note

If the setting is changed to increase the amount of logs that can be stored, it may exceed the disk capacity stated in the Software Description. The Job Operation Software may not operate normally when the disk space is insufficient.

3.7 Procedure for avoiding disturbance to job execution performance

This section describes how to avoid disturbances that affect job performance.

Disturbance due to periodic collection of job statistical information by the job resource management function

The periodic collection of job statistical information by the job resource management function can affect the execution performance of a job.

To prevent this, set the RscWatchInterval in the parsc.conf and pmrsc.conf to 0 to stop the periodic collection of job statistical information for running jobs. See "3.4.3.1 Settings for job resource management function in a cluster","3.5.2.1 Settings for job resource management function in a resource unit" for more information.

Note that with this setting, some of the job statistical information (CPU time, memory usage, number of execution cycles, etc.) will not be available to the pjstat -s/-S command until the end of the desired job.

Disturbance due to OS memory recovery processing

If the OS memory recovery process occurs during the execution of a job, it may cause disturbance to the execution performance of the job. To prevent this, you can use the job resource management exit function to free the memory cache. For details, refer to "Creating exit scripts" in the chapter " Creating and Incorporating Hooks" of the manual "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

Chapter 4 Operation with the Job Operation Management Function

This chapter describes job operations with the job operation management function.

Basically, the job operation administrator performs job operations, though only the cluster administrator can perform some parts of job operations.

Each of these administrators performs the following work related to job operations.

Cluster administrator

- Cluster state monitoring
- Cluster deadline scheduling management
- Changing job ACL function settings for a cluster
- Saving ended job script files

Job operation administrator

- Resource unit state monitoring
- Job state monitoring
- Job operations
- Job resource monitoring
- Submitting jobs and suppressing and canceling job execution
- Monitoring and changing fair share values
- Changing job ACL function settings for a resource group
- Changing the job submission range and job execution range
- Displaying job statistical information

The following sections describe the procedures for this work.

4.1 Operational Work of the Cluster Administrator

This section describes work of the cluster administrator.

The cluster administrator mainly performs the following work:

- Cluster state monitoring
- Cluster deadline scheduling management
- Changing job ACL function settings for a cluster
- Saving ended job script files

Information

The examples in this section contain commands specifying a cluster name in the -c option for operation on the system management node. Write the name in this way. However, if the environment variable PXMYCLST specifies the cluster name, you can omit specification of the cluster name in the -c option during actual operation.

4.1.1 Cluster state monitoring

You can check the states of nodes and services for cluster state monitoring by using the pashowclst command.

For details, see "Displaying System Configuration Information" in "Chapter 3 Details of the System Management Function" in "Job Operation Software Administrator's Guide for System Management."

4.1.2 Cluster deadline scheduling management

Use the padeadline command on the system management node or the compute cluster management node to set and check deadline schedules.

For details, see the man page for the padeadline command.

Setting a deadline schedule

When an FX server and a PRIMERGY server are included in the same cluster, it is necessary to specify by option which of the FX server, or PRIMERGY server is to be the target.

Specify one of the following options:

- Option for FX server: --ph
- Option for PRIMERGY server: --pg

The necessity to specify one of these options depends on the cluster configuration as follows.

Cluster configuration	FX server	PRIMERGY server	Specification of --ph or --pg option
One cluster	Included	Included	Required
	Included	Not included	Not required
	Not included	Included	Not required
Multiple clusters	Included	Included	Not required (*)
	Included	Not included	Not required
	Not included	Included	Not required

(*)In cases where there are multiple clusters (in an environment consisting of two clusters, one being an cluster of FX servers, and the other one being a cluster of PRIMERGY servers), there is no need to specify the --ph or --pg option.

Note

- A deadline schedule cannot be set for a resource group (*) with node resources defined based on the number of nodes or a ratio of the number of nodes. If a deadline schedule needs to be set for the resource group, set the deadline schedule for the resource unit to which the resource group belongs.
  (*) ResourceGroupNode item in the pmpjm.conf file (See "3.5.1.2 Resource group settings.")
- The compute nodes contained in a resource group defined when the padeadline command is executed are subject to the deadline schedule even if the resource group range is changed after the deadline schedule is set. After changing the range of a resource group, delete the set deadline schedule from the resource group, and set the deadline schedule again.
- Jobs in the following state are forcibly terminated after the start of the deadline schedule:
  RUNNING-A, RUNNING, RUNNING-E, RUNNING-P, RUNOUT.

The following is an example of setting a deadline schedule for cases where the FX server and PRIMERGY server are both included in one cluster.

[For FX server]

This setting excludes the resources in the entire range of the x- and y-axes (this always includes the entire z-axis) from the job resources that can be selected from 00:00:00 on August 20, 2019 to 23:59:59 on August 21, 2019.
Specify --ph as an option of the padeadline command.

```
[System management node or Compute cluster management node]
# padeadline -c clstname --ph --coord all,all --period 'start=20190820,end=20190821'
```

Note

- If the --cmu option is specified when the padeadline command is executed, it sets the deadline schedule for all the compute nodes included in the CMU of the specified node.
  For details on the options that specify the setting range for other deadline schedules, see the man manual of the padeadline command.

- For resource groups that can use jobs that are allocated in node units in Mesh mode, when the deadline schedule is set by specifying nodes, the jobs allocated in node units in Mesh mode may be affected if the specified nodes stop. In this case, set the deadline schedule for all nodes in Tofu units including the stopped nodes.

- Note the following during hardware maintenance of FX servers belonging to a resource unit that allows the submission of non-contiguously allocated jobs: To set a deadline schedule for the compute nodes to be maintained, you need to execute the padeadline command with the --ic option specified. This option prevents the allocation of new jobs that are using the interconnect installed on any of the compute nodes to be maintained as a communication path.

- Enabling the setting that dynamically changes a Tofu interconnect communication path extends the range of nodes that can be used as a communication path. Therefore, even when deadline scheduling is set so that a node should not be the communication path of a job (padeadline --ic), the node may fall in the range of nodes for a running job.
  In that case, the padeadline command causes an error, and the deadline schedule cannot be set. If the error occurs, use the -v 3 option of the pjshowrsc command to check the node used as the communication path of the running job. Also, review the nodes covered by the deadline schedule. Note that the -v 3 option of the pjshowrsc command also displays nodes that can be used as alternative routes.

[For PRIMERGY server]

This setting excludes the node IDs 0x41FF0000 to 0x41FF0009 and 0x41FF0020 to 0x41FF0029 from the job resources that can be selected from 00:00:00 on August 20, 2019 to 23:59:59 on August 21, 2019.
Specify --pg as an option of the padeadline command.

```
[System management node or Compute cluster management node]
# padeadline -c clstname --pg --n 0x41FF0000-0x41FF0009,0x41FF0020-0x41FF0029 \
  --period 'start=20190820,end=20190821'
```

Displaying the set deadline schedules

```
[System management node or Compute cluster management node]
# padeadline -c clstname --show
NO.    TYPE    START                 END                   TARGET
3      ru      2019-08-06 00:00:00   2019-08-06 23:59:59   unit0
5      f       2019-08-20 00:00:00   unlimited             nodelist1
2      n       2019-08-27 00:00:00   2019-08-27 23:59:59   InterConnect 0x01010010
1      bg      2019-09-06 00:00:00   2019-09-06 23:59:59   0x0103
```

The following table lists each item and its meaning.

<table>
<tr><th>Item</th><th colspan="2">Description</th></tr>
<tr><td>NO.</td><td colspan="2">Deadline schedule-specific number given when the deadline schedule is set</td></tr>
<tr><td>TYPE</td><td colspan="2">Type of the target resource specified when the deadline schedule is set.
The meaning of the output character string is listed below.</td></tr>
<tr><td></td><td>Output character string</td><td>Target resource specification method</td></tr>
<tr><td></td><td>a</td><td>-a</td></tr>
<tr><td></td><td>ng</td><td>--nodegrp</td></tr>
<tr><td></td><td>bg [FX]</td><td>--bootgrp</td></tr>
</table>

Item	Description	
	ru	--rscunit or --ru
	rg	--rscgrp or --rg
	cmu [FX]	--cmu
	n	-n
	f	-f
TARGET	Whether the use of the resource specified by TYPE and the use of the interconnect are suppressed (If the use of the interconnect installed on the resource specified by TYPE is also suppressed, "InterConnect" is displayed in addition to the resource information.)	
START	Start date and time of the target period of the deadline schedule	
END	End date and time of the target period of the deadline schedule	

Deleting a set deadline schedule

The deadline schedule number 2 is canceled.

```
[System management node or Compute cluster management node]
# padeadline -c clstname --cancel 2
```

Changing a set deadline schedule

To change a resource or time in a set deadline schedule, delete the set deadline schedule once and then set a new deadline schedule.

4.1.3 Changing job ACL function settings for a cluster

Use the pmjacladm command on the system management node or the compute cluster management node to check and change job ACL function settings.

For details, see the man page for the pmjacladm command.

To display all the definition sections and job ACL configuration definitions registered in the job ACL, specify the --show option in the pmjacladm command, and execute the command with cluster administrator privileges. An example is shown below.

```
[System management node or Compute cluster management node]
# pmjacladm -c clstname --show '*'
#
# JOBACL definition - clst1
# [ * ]
#
USER: CL {
    user=<def> {  # last update 2019-03-01 19:26:33
        define rscunit         rscunit1
        define rscgroup        rscgroup1
    }
}

USER: CL, RU=rscunit1 {
    user=<def> {  # last update 2019-03-01 19:26:33
        limit ru-accept        unlimited
        limit ru-run-job       unlimited
        joblimit node-cpu      -          unlimited  unlimited
        joblimit elapse        1          24:00:00   24:00:00
        joblimit node          1          80000      1
        joblimit node-mem      1M         10G        2G
    }

    user=<def>:group2 { # last update 2011-03-01 19:26:33
        define rscgroup        rscgroup2
    }

    user=user1 {  # last update 2019-03-01 19:26:33
        permit pjstat          allow own
        permit pjstat          + allow g(group1)
        permit pjdel           allow own
        permit pjdel           + allow g(group1)
    }

}

USER: CL, RU=rscunit1, RG=rscgroup2 {
    user=user2 {  # last update 2019-03-01 19:26:33
        execute pjsub          disable
    }

    user=user3 {  # last update 2019-03-01 19:26:33
        permit pjhold          allow own
        permit pjrls           allow own
    }

    user=user3:group2 {  # last update 2019-03-01 19:26:33
        define ingroup-pri     128
    }
}

GROUP: CL, RU=rscunit1 {
    group=group2 {  # last update 2019-03-01 19:26:33
        define pri-g           160
    }
}
```

To display the job ACL values that have been actually applied, use the pmjacladm command with the --show option and --apply option. For the --apply option, you need to specify a resource unit, resource group, user, and group.

Note

The actual display of definitions is different in the following points. To use the displayed content as definitions, you need to do some editing such as by adding appropriate sections.

- For the application range, resource units and resource groups are displayed. Exclusive defined items of the cluster and exclusive resource unit items are also output.
- For the user definition targets, user=uname:gname is displayed, and the items for which group-related specification (user=<def>:gname, user=uname:gname) is not allowed are also displayed.

Information

When executing the pmjacladm command on a compute cluster management node, you can omit specification of a cluster name in the -c option.

The following example displays the values applied to the user user1 and the group group1.

```
[System management node or Compute cluster management node]
# pmjacladm -c clstname --show \
--apply 'USER: CL, RU=rscunit1, RG=rscgroup1 { user=user1:group1 }'
#
# JOBACL applied result - clst1
# [ USER: CL, RU=rscunit1, RG=rscgroup1 { user=user1:group1 } ]
#
```

```
USER: CL, RU=rscunit1, RG=rscgroup1 {
    user=user1:group1 {
        limit ru-accept                     <RU->  unlimited
        limit ru-run-job                    <RU->  unlimited
        limit ru-use-node                   <--->  unlimited
        limit ru-interact-accept            <--->  unlimited
        limit ru-interact-run-job           <--->  unlimited
        limit ru-interact-use-node          <--->  unlimited
        limit ru-use-core                   <--->  unlimited
        limit ru-interact-use-core          <--->  unlimited
        limit rg-accept                     <--->  unlimited
        limit rg-run-job                    <--->  unlimited
        limit rg-use-node                   <--->  unlimited
        limit rg-interact-accept            <--->  unlimited
        limit rg-interact-run-job           <--->  unlimited
        limit rg-interact-use-node          <--->  unlimited
        limit rg-use-core                   <--->  unlimited
        limit rg-interact-use-core          <--->  unlimited
        limit ru-accept-allsubjob           <--->  unlimited
        limit ru-accept-bulksubjob          <--->  unlimited
        limit ru-accept-stepsubjob          <--->  unlimited
        limit ru-run-bulksubjob             <--->  unlimited
        limit rg-accept-allsubjob           <--->  unlimited
        limit rg-accept-bulksubjob          <--->  unlimited
        limit rg-accept-stepsubjob          <--->  unlimited
        limit rg-run-bulksubjob             <--->  unlimited
        define rscunit                      <CL->  rscunit1
        define rscgroup                     <CL->  rscgroup1
        define pri                          <RUu>  128
        define ingroup-pri                  <RGug> 100
 (--omitted--)
    }
}
```

An item application indicator is added between the defined item name and value. The following table lists the displayed content.

Table 4.1 Item application indicators in the user-specific definition section (USER:)

Item application indicator	Description
<--->	Application of a system default value (*)
<CL->	Application from the definition of user=<def> in the cluster definition section
<CL-g>	Application from the definition of user=<def>:gname in the cluster definition section
<CLu>	Application from the definition of user=uname in the cluster definition section
<CLug>	Application from the definition of user=uname:gname in the cluster definition section
<RU->	Application from the definition of user=<def> in the resource unit definition section
<RU-g>	Application from the definition of user=<def>:gname in the resource unit definition section
<RUu>	Application from the definition of user=uname in the resource unit definition section
<RUug>	Application from the definition of user=uname:gname in the resource unit definition section
<RG->	Application from the definition of user=<def> in the resource group definition section
<RG-g>	Application from the definition of user=<def>:gname in the resource group definition section
<RGu>	Application from the definition of user=uname in the resource group definition section
<RGug>	Application from the definition of user=uname:gname in the resource group definition section

(*) The application indicator for the cluster is hidden from users other than the cluster administrator (though the value itself is displayed, application indicator is displayed "<--->")

Table 4.2 Item application indicators in the group-specific definition section (GROUP:)

Item application indicator	Description
<--->	Application of a system default value (*)
<CL->	Application from the definition of group=<def> in the cluster definition section
<CLg>	Application from the definition of group=gname in the cluster definition section
<RU->	Application from the definition of group=<def> in the resource unit definition section
<RUg>	Application from the definition of group=gname in the resource unit definition section
<RG->	Application from the definition of group=<def> in the resource group definition section
<RGg>	Application from the definition of group=gname in the resource group definition section

(*) The application indicator for the cluster is hidden from users other than the cluster administrator (though the value itself is displayed, application indicator is displayed "<--->")

Table 4.3 Item application indicators in the All-encompassing definition section (ALL:)

Item application indicator	Description
<--->	Application of a system default value (*)
<CL->	Application from the cluster definition section
<RU->	Application from the resource unit definition section
<RG->	Application from the resource group definition section

(*) The application indicator for the cluster is hidden from users other than the cluster administrator (though the value itself is displayed, application indicator is displayed "<--->")

To display job ACL settings in a format where the fields are delimited with a delimiter, use the pmjacladm command with the --show option and --data option. Exercise caution when specifying the delimiter because the list items may contain commas or colons.

The following example displays job ACL function settings with ";" as the delimiter character.

```
[System management node or Compute cluster management node]
# pmjacladm -c clstname --show --data --delimiter ";" '*'

USER:;CL
define;rscunit;rscunit1
USER:;CL;RU=rscunit1
user=<def>
limit;ru-accept;unlimited
limit;ru-run-job;unlimited
joblimit;elapse;1;24:00:00;24:00:00
joblimit;node;1;80000;1
joblimit;node-mem;1M;10G;2G
 (--omitted--)
user=user1
permit;pjstat;allow own
permit;pjstat;+ allow g(group1)
permit;pjdel;allow own
permit;pjdel;+ allow g(group1)
 (--omitted--)
```

To list the defined items that can be set in the job ACL and display the list in sections, use the pmjacladm command with the --show option and --items option.

The following example displays, in the form of sections, definition items that can be set by the job ACL function.

```
[System management node or Compute cluster management node]
# pmjacladm -c clstname --show --items
```

```
#
# JOBACL items list - clst1
#
#USER: CL {                              <- Exclusive defined items of the cluster
#    user=<def> {
#    user=<def>:group {
#    user=user {
#    user=user:group {
#        execute pjstat                 enable       # CL
#        execute pjdel                  enable       # CL
 (--omitted--)
#   }
#}
#USER: CL, RU=runame {                   <- Exclusive defined items of the resource unit
#    user=<def> {
#    user=<def>:group {
#    user=user {
#    user=user:group {
#        limit ru-accept                unlimited    # CLRU nogrp
#        limit ru-run-job               unlimited    # CLRU nogrp
#        limit ru-use-node              unlimited    # CLRU nogrp
 (--omitted--)
#   }
#}
#USER: CL, RU=runame, RG=rgname {        <-Defined items of the resource group
#    user=<def> {
#    user=<def>:group {
#    user=user {
#    user=user:group {
#        define pri                     128          # CLRURG nogrp
#        define ingroup-pri             128          # CLRURG
#        joblimit subjobnum   -          unlimited  -           # CLRURG
#        joblimit priv-pri    -          255        128         # CLRURG
 (--omitted--)
#   }
#}
#GROUP: CL, RU=runame {
#    group=<def> {
#    group=group {
#        limit ru-accept                unlimited    # CLRU
#        limit ru-run-job               unlimited    # CLRU
 (--omitted--)
#   }
#}
#ALL: CL, RU=runame {
#        limit ru-accept                unlimited    # CLRU
#        limit ru-run-job               unlimited    # CLRU
 (--omitted--)
#}
```

The layers that can be specified are written as a comment at the end of each defined item.

Table 4.4 Layers that can be specified

Definition name	Description
CL	Exclusive defined items of the cluster
CLRU	Exclusive defined items of the resource unit (can also be specified as common definition values in a cluster range)
CLRURG	Defined items of the resource group (can also be specified as common definition values in a higher layer range)
nogrp	Defined items for each user for which group-related specification (user=<def>:group, user=user:group) is not allowed

4.1.4 Saving ended job script files

When jobs enters the EXIT or CANCEL state, the ended job script file is saved in the following directory on the compute cluster management node.

```
[Compute cluster management node]
/var/opt/FJSVtcs/shared_disk/pjm/private/root/save/subdir
```

Information

- To be precise, the job script save timing is when the job state transitions to EXIT or CANCEL and then disappears from the display by the pjstat command.
- If the input in an interactive job is not a script, no job script is saved because there is none.

subdir is a number from 0 to 99, which is the remainder of the division of the job ID by 100. The job script file is saved in the *subdir* directory.

The file name of a job script file is "d" + (10-digit job ID).

For example, the job script file of job ID 13856 is saved with the following path name.

```
/var/opt/FJSVtcs/shared_disk/pjm/private/root/save/56/d0000013856
```

For a step job, the file name of a job script file is "d" + (10-digit job ID) + "_" + (5-digit step number).

For example, the job script file of step job 15927_35 is saved with the following path name.

```
/var/opt/FJSVtcs/shared_disk/pjm/private/root/save/27/d0000015927_00035
```

Note

The job manager function does not delete saved job script files. The cluster administrator is requested to set periodic deletion of job scripts, such as by using the cron command.

4.2 Operational Work of the Job Operation Administrator

This section describes work of the job operation administrator.

The job operation administrator mainly performs the following work:

- Resource unit state monitoring
- Job state monitoring
- Job resource monitoring
- Submitting jobs, suppressing and canceling job execution, and canceling errors
- Monitoring and changing fair share values
- Changing job ACL function settings for a resource unit
- Changing whether jobs can be submitted or can be executed
- Displaying job statistical information

Information

The examples in this section contain commands specifying a cluster name in the -c option for operation on the system management node. Write the name in this way. However, if the environment variable PXMYCLST specifies the cluster name, you can omit specification of the cluster name in the -c option during actual operation.

4.2.1 Resource unit state monitoring

By using the pashowclst command by cluster, the job operation administrators can check the state of nodes in the resource units that they themselves manage.

Management of the resource units unit3, unit5 and unit6 by the job operation administrator

```
[System management node]
# pashowclst -c cluster1 --rscunit unit3
[ CLST: cluster1 ]
[ RSCUNIT: unit3 ]
RSCUNIT           RUNNING  STOPPED  ERROR    DISABLE
unit3             1632     0        0        0
# pashowclst -c cluster2 --rscunit unit5,unit6
[ CLST: cluster2 ]
[ RSCUNIT: unit5 ]
RSCUNIT           RUNNING  STOPPED  ERROR    DISABLE
unit5             393      10       2        3
[ RSCUNIT: unit6 ]
RSCUNIT           RUNNING  STOPPED  ERROR    DISABLE
unit6             816      0        0        0
```

Displaying details of abnormal nodes in resource units by specifying the -v and -d options

```
[System management node]
# pashowclst -c cluster1 --rscunit -v -d
[ CLST: cluster1 ]
[ RSCUNIT: unit2 ]
NODE        NODETYPE   STATUS     REASON          PWR_STATUS     ARCH_STATUS    SRV_STATUS
0x02010010  CN         Disable    Manual          off            ICC_Disable    -
```

4.2.2 Job state monitoring

Using the pjstat command, the job operation administrator can monitor the states of jobs running on the resource units managed by the administrator.

```
[Login node or system management node or compute cluster management node]
# pjstat
JOB_ID    JOB_NAME   MD ST  USER    START_DATE     ELAPSE_LIM       NODE_REQUIRE   VNODE  CORE V_MEM
12345678   jobname1   NM RUN usrname  01/01 00:00:00  0100:00:00        12:2x3x2        -      -    -
12345679   jobname2   ST RUN usrname  01/01 00:10:00  -                 -               -      -    -
12345680   jobname3   BU RUN usrname  01/01 01:10:11  0100:00:00        24:2x3x4        -      -    -
12345681   jobanem4   ST QUE usrname  (01/01 12:00)   -                 -               -      -    -
12345682   jobname5   NM QUE usrname  (01/05 12:30)   0000:20:00        12:2x3x2        -      -    -
12345683   jobname8   BU QUE usrname  (01/01 15:00)   0100:00:00        24:2x3x4/3      -      -    -
12345684   jobname9   NM RUN username 01/01 02:00:00  0050:00:00         -               32     8
unlimited
```

For details on the displayed contents, see "Checking the Job Status" in "Chapter 2 Job Operation Procedures" in "Job Operation Software End-user's Guide."

4.2.3 Job Operations

The job operation administrator can perform, as required, the following operations for all the jobs of end users:

- Deleting a job (pjdel command)
- Holding a job on hold (pjhold command)
- Canceling the hold on a job (pjrls command)
- Changing job parameters (pmalter command)
- Recovering from a job error state (pmerls command)

This section describes each operation.

Deleting a job

Use the pjdel command to delete a job. This command can be executed on the login node or compute cluster management node.

```
[Login node or compute cluster management node]
# pjdel jobid
```

The deleted job is aborted. The user who submitted the job receives e-mail notification of the submitting in pjsub command with -m option. Since the job is deleted, it will not be executed unless it is submitted again.

The following example deletes the job with job ID 453023.

```
[Login node or compute cluster management node]
# pjdel 453023
```

The pjdel command outputs the following message when the job stop process has been executed normally.

```
[INFO] PJM 0100 pjdel Accepted job 453023.
```

If the specified job does not exist, the pjdel command outputs the following message.

```
[ERR.] PJM 0112 pjdel Job non-existing_job_ID does not exist.
```

Holding a job on hold

Use the pjhold command to hold a job in the hold state. This command can be executed on the login node or compute cluster management node.

```
[Login node or compute cluster management node]
# pjhold jobid
```

The following examples place jobs on hold.

- Example of holding two jobs (job IDs 1 and 2) on hold

```
[Login node or compute cluster management node]
# pjhold 1 2
[INFO] PJM 0300 pjhold Accepted job 1.
[INFO] PJM 0300 pjhold Accepted job 2.

# pjstat 1-2
JOB_ID  ....  JOB_NAME  MD ST   USER   ....  REASON
     1  ....  jobname1  NM HLD  user1  ....  held by user1
     2  ....  jobname2  NM HLD  user2  ....  held by user1
```

- Example where a job (job ID 2) cannot be hold on hold

```
[Login node or compute cluster management node]
# pjhold 1 2 3 4
[INFO] PJM 0300 pjhold Accepted job 1.
[ERR.] PJM 0313 pjhold Job 2 is not in the state that pjhold can be accepted.
[INFO] PJM 0300 pjhold Accepted job 3.
[INFO] PJM 0300 pjhold Accepted job 4.
```

Canceling the hold on a job

Use the pjrls command to cancel the hold on a job. This command can be executed on the login node or compute cluster management node.

```
[Login node or compute cluster management node]
# pjrls jobid
```

This reschedules the job to execute it again.
The following example cancels the hold on two jobs (job IDs 1 and 2).

```
[Login node or compute cluster management node]
# pjrls 1 2
[INFO] PJM 0400 pjrls Job 1 released.
[INFO] PJM 0400 pjrls Job 2 released.
```

See

For details on deleting jobs, placing jobs on hold, and canceling the hold on jobs, also see "Job Operations" in "Chapter 2 Job Operation Procedures" in "Job Operation Software End-user's Guide."

Changing job parameters

Use the pmalter command to change job parameters such as the elapsed time limit value and the priority of a running job. This command can be executed on the system management node or compute cluster management node.

```
[System management node or compute cluster management node]
# pmalter -c clstname options jobid
```

Note

- Job parameters can be changed when the job status is QUEUED, HOLD, or ERROR.
  If the RunJobAlterElapse item in the papjm.conf file is set to on, the elapsed time limit value of a job can be changed even when the job status is RUNNING. However, this applies only to jobs on an FX server, and the limit value cannot be changed to a value less than the time that has already elapsed.

- If you are changing a resource unit or resource group to run a job, the job ACL function must be set to the following (the parentheses denote the output of the pjacl command) for the user who submitted the job, not the administrator running the pmalter command:

  - Job submission is allowed for the destination resource unit or resource group ('execute pjsub' is set to 'enable').

  - The options specified at the time of job submission are also allowed in the resource unit or resource group to which it is being changed ('execute pjsub(*opt*)' is set to 'enable').

  - The resource amount specified at the time of job submission is within the upper and lower limits ('upper' and 'lower' in 'pjsub option parameters') in the resource unit or resource group to which it is being changed.

  - If the job uses custom resources, the resource unit or resource group to which it is being changed must also have the custom resources defined (The custom resources to be use are displayed in 'pjsub option parameters').

  To check the above settings, specify the user submitted a job, and the resource unit or resource group you want to change.in the pjacl command.

  ```
  # pjacl -u jobuser --rscunit resourceunit
  # pjacl -u jobuser --rscgrp resourcegroup
  ```

- If the sub job ID of the bulk job is specified, it is possible only to change the elapsed time limit value.

Information

End users use the pjalter command to change the parameters of their jobs.

Recovering from a job error state

Suppose that a submitted job has a temporary error. Use the pmerls command to cancel the job ERROR state of the job, and queue the job again. This command can be executed on the system management node or compute cluster management node.
The following shows the execution format of the pmerls command.

```
[System management node or compute cluster management node]
# pmerls -c clstname jobid
```

The following example cancels the ERROR state of the jobs of job IDs 1 to 3, and queues the jobs again. (Job ID 3 does not exist.)

```
[System management node or compute cluster management node]
# pmerls -c clstname 1 2 3
[INFO] PJM 0900 pmerls Job 1 released.
[INFO] PJM 0900 pmerls Job 2 released.
[ERR.] PJM 0912 pmerls Job 3 does not exist.
```

Information

When executing the pmalter or pmerls command on a compute cluster management node, you can omit specification of a cluster name in the -c option.

4.2.4 Job resource monitoring

Use the pjshowrsc command to refer to the total amounts and the usage status of computer resources for jobs.

For examples of pjshowrsc command execution, see "Checking Job-related Information" in "Chapter 2 Job Operation Procedures" in "Job Operation Software End-user's Guide."

To output system limit values (concurrent job acceptance limit, concurrent job execution limit, and concurrent sub job execution limit), use the pjstat command with the --limit option.

If the job operation administrator executes the command, the output system limit values are those of the resource units managed by the job operation administrator.

Note

If the cluster administrator executes the command, the system limit values of the entire cluster are output. If an end user executes the command, the system limit values for the end user are output.

An execution example on the cluster of FX servers is shown below.

```
[Login node, system management node, or compute cluster management node]
# pjstat --limit
 System Resource Information:
 RSCUNIT: runit1
 USER: user1
  LIMIT-NAME                           LIMIT         ALLOC
  ru-accept                        unlimited             5
  ru-accept-allsubjob              unlimited             0
  ru-accept-bulksubjob             unlimited             0
  ru-accept-stepsubjob             unlimited             0
  ru-run-job                       unlimited             3
  ru-run-bulksubjob                unlimited             0
  ru-use-node                      unlimited          1011
  ru-interact-accept               unlimited             0
  ru-interact-run-job              unlimited             0
  ru-interact-use-node             unlimited             0
  ru-use-core                      unlimited            30
  ru-interact-use-core             unlimited             0
  ru-custom-customresourcename            10             1
  ru-interact-custom-customresourcename    5             1
 GROUP: group1
  LIMIT-NAME                           LIMIT         ALLOC
  ru-accept                        unlimited             5
  ru-accept-allsubjob              unlimited             0
  ru-accept-bulksubjob             unlimited             0
  ru-accept-stepsubjob             unlimited             0
  ru-run-job                       unlimited             3
  ru-run-bulksubjob                unlimited             0
  ru-use-node                      unlimited          1011
  ru-interact-accept               unlimited             0
  ru-interact-run-job              unlimited             0
  ru-interact-use-node             unlimited             0
  ru-use-core                      unlimited            30
  ru-interact-use-core             unlimited             0
 ALL:
  LIMIT-NAME                           LIMIT         ALLOC
  ru-accept                        unlimited             5
  ru-accept-allsubjob              unlimited             0
  ru-accept-bulksubjob             unlimited             0
  ru-accept-stepsubjob             unlimited             0
  ru-run-job                       unlimited             3
  ru-run-bulksubjob                unlimited             0
  ru-use-node                      unlimited          1011
  ru-interact-accept               unlimited             0
  ru-interact-run-job              unlimited             0
  ru-interact-use-node             unlimited             0
  ru-use-core                      unlimited            30
  ru-interact-use-core             unlimited             0
```

The following tables describe output item of --limit option in pjstat command.

Table 4.5 Output item of --limit option in pjstat command

Item	Description
LIMIT-NAME	Limit value name
RSCUNIT	Resource unit name
GROUP	Group name in the operating system
LIMIT	Limit value
ALLOC	Current allocation value

4.2.5 Monitoring and changing a fair share value and initial fair share value

Use the pmpjmopt command on the system management node to change the fair share value or initial fair share value of a job.

For details, see the man page for the pmpjmopt command.

The following shows the execution format.

```
pmpjmopt -c clstname --set-fairshare {--rscunit | --ru rscuname} {--rscgroup | --rg rscgname}
                   [ --all-fairshares | --fairshare fairshareset]
                   { --value newvalue| --reset-value}
                   {{-g group[,...] | -u user[,...] | --gu group:user[,...]}
                    | {--all-groups | --all-users | --all-gu}}
pmpjmopt -c clstname --show-fairshare {--rscunit | --ru rscuname}{--rscgroup | --rg rscgname}
                   [ --all-fairshares | --fairshare fairshareset]
                   {{-g group[,...] | -u user[,...] | --gu group:user[,...]}
                    | {--all-groups | --all-users | --all-gu}}
```

If the --set-fairshare option is specified, a fair share value is set.

If the --set-fairshare option is specified with the --reset-value option, the fair share value returns to the initial value.

If the --show-fairshare option is specified, the fair share value is displayed.

The following shows setting examples.

Set and display fair share values

- Returning the user fair share value of the fair share set def_fs to the initial value

```
[System management node]
# pmpjmopt --set-fairshare -c cluster1 --rscunit unit0 --fairshare def_fs \
--reset-value --all-users
[INFO]PJM 6100 pmpjmopt Operation completed.
```

- Changing the user fair share value of the users user1 and user2 of the fair share set def_fs to 1500

```
[System management node]
# pmpjmopt --set-fairshare -c cluster1 --rscunit unit0 --fairshare def_fs \
--value 1500 -u user1,user2
[INFO]PJM 6100 pmpjmopt Operation completed.
```

- Displaying the user fair share values of all the users of the fair share set def_fs

```
[System management node]
# pmpjmopt --show-fairshare -c cluster1 --rscunit unit0 --fairshare def_fs \
--all-users

ClusterName      = cluster1
ResourceUnitName = unit0

Now 2018/12/20 15:25:48

FairShareSet=def_fs
NO      TYPE    GROUP    USER     CURRENT  INIT     RECOVERY
1       U       --       user1    1500     0        100
2       U       --       user2    1500     0        100
3       U       --       root     0        0        100
```

The following tables describe output item.

Item	Description
TYPE	U: User G: Group G/U: In-group user
GROUP	Group name
USER	User name
CURRENT	Current fair share value
INIT	Default user fair share value
RECOVERY	fair share recovery ratio

Set and display in-group fair share values

- Returning the group fair share values of all the fair share sets to the initial values

```
[System management node]
# pmpjmopt --set-fairshare -c cluster1 --rscunit unit0 --all-fairshare \
--reset-value --all-groups
[INFO]PJM 6100 pmpjmopt Operation completed.
```

- Changing the group fair share value of the group group1 of the fair share set grp_fs1 to 1200

```
[System management node]
# pmpjmopt --set-fairshare -c cluster1 --rscunit unit0 --fairshare grp_fs1 \
--value 1200 -g group1
[INFO]PJM 6100 pmpjmopt Operation completed.
```

- Displaying the group fair share values of all the groups of all the fair share sets

```
[System management node]
# pmpjmopt -c cluster1 --show-fairshare --rscunit unit0 --all-fairshare \
--all-groups

ClusterName      = cluster1
ResourceUnitName = unit0

Now 2018/12/20 15:26:12

FairShareSet=def_fs
NO      TYPE    GROUP    USER     CURRENT  INIT     RECOVERY
1       G       root     --       0        0        100
2       G       group1   --       1200     0        100

FairShareSet=grp_fs1
NO      TYPE    GROUP    USER     CURRENT  INIT     RECOVERY
1       G       root     --       0        0        100
2       G       group1   --       1200     0        100
```

Set and display in-group user fair share values

- Returning the user fair share values to the initial values in all the groups of the fair share set def_fs

```
[System management node]
# pmpjmopt --set-fairshare -c cluster1 --rscunit unit0 --fairshare def_fs \
--reset-value --all-gu
[INFO]PJM 6100 pmpjmopt Operation completed.
```

- Changing the user fair share value of the group group1 of all the fair share sets to 1000

```
[System management node]
# pmpjmopt --set-fairshare -c cluster1 --rscunit unit0 --all-fairshare --value 1000 \
--gu group1:user1
[INFO]PJM 6100 pmpjmopt Operation completed.
```

- Displaying the user fair share values in all the groups of all the fair share sets

```
[System management node]
# pmpjmopt -c cluster1 --show-fairshare --rscunit unit0 --all-fairshare --all-gu

ClusterName      = cluster1
ResourceUnitName = unit0

Now 2018/12/20 15:26:21

FairShareSet=def_fs
NO      TYPE    GROUP    USER     CURRENT  INIT     RECOVERY
1       G/U     root     root     0        0        100
2       G/U     group1   user1    1000     0        100
3       G/U     group1   user2    0        0        100

FairShareSet=grp_fs1
NO      TYPE    GROUP    USER     CURRENT  INIT     RECOVERY
1       G/U     root     root     1100     0        100
2       G/U     group1   user1    1000     0        100
3       G/U     group1   user2    1200     0        100
```

Set the initial fair share value

The initial fair share value is set in the job ACL function.

- This example changes the initial user fair share value to 10 in the resource unit unit1

```
[System management node]
# pmjacladm -c cluster1 --set 'USER: CL, RU=unit1 {user=<def> { define fshare-init 10 }}'
```

- Changing the initial user fair share value of the fair share set grp_fs1 to 20 in the resource unit unit1

```
[System management node]
# pmjacladm -c cluster1 --set 'USER: CL, RU=unit1 \
{user=<def> { define fshare-init 20@grp_fs1 }}'
```

The above example changes the initial fair share value of a user. Use the following items to set the initial fair share value of a group or a user in a group:

- Group: define fshare-init-g
- User in a group: define ingroup-fshare-init

For more information about job ACL function settings, see "Appendix F Defined items of the job ACL function" or man page pmjacladm(8).

Note

Even if you change the initial fair share value (an upper limit of fair share value) partway through job operation, the fair share value does not change at that point in time. Changing the initial fair share value to a smaller value may cause the fair share value to be larger than the initial fair share value. The fair share value recovers only when it is smaller than the initial fair share value. So if your change may result in the above situation, also change the fair share value to an initial value as required.

Display the initial user fair share value

The method of displaying the initial fair share value is the same method shown as "Displaying the user fair share values of all the users of the fair share set def_fs" in the above "Set and display fair share values." The INIT column shows the initial fair share value.

4.2.6 Changing job ACL function settings for a resource unit

Check job ACL function settings in a resource unit in the same way as for job ACL function settings in a cluster. For details, see "4.1.3 Changing job ACL function settings for a cluster".

4.2.7 Changing whether jobs can be submitted or can be executed

Use the pmpjmopt command on the system management node to change whether jobs can be submitted or can be executed. For details, see the man page for the pmpjmopt command.

The following table lists combinations to allow or deny job submission and execution.

Table 4.6 Combinations to allow or deny job submission and execution

Allow/deny submission	Allow/deny execution	Job operation in resource unit
enable	start	Jobs can be both submitted and executed.
enable	stop	Jobs can be submitted but cannot be executed.(Execution of running jobs continues. Submitted jobs remain QUEUED.)
disable	start	Jobs cannot be submitted but already submitted jobs can be executed.
disable	stop	Jobs can be neither submitted nor executed.(Execution of running jobs continues. Submitted jobs remain QUEUED.)

Execution examples are shown below.

This example allows submission of jobs and denies execution of jobs for the resource unit unit1.

```
[System management node]
# pmpjmopt -c clstname --set-rsc-ug --rscunit unit1 enable,stop --rscgrp group1
[INFO] PJM 6100 pmpjmopt Operation completed.
```

This example allows submission of jobs and denies execution of jobs for the resource group group1.

```
[System management node]
# pmpjmopt -c clstname --set-rsc-ug --rscunit unit1 enable,stop --rscgrp group1
[INFO] PJM 6100 pmpjmopt Operation completed.
```

This example checks the state of the resource unit unit1.

```
[System management node]
# pmpjmopt -c clstname --show-rsc-ug --rscunit unit1
ResourceUnitName = unit1
Apply Value                : ENABLE,STOP
```

This example checks the state of the resource group group1.

```
[System management node]
# pmpjmopt -c clstname --show-rsc-ug --rscunit unit1 --rscgrp group1
ResourceGroupName = group1
Apply Value                : ENABLE,STOP
```

4.2.8 Displaying job statistical information

The following methods are methods for checking job statistical information:

- By displaying job output results (-s/-S of the pjsub command)
- By obtaining job state check results (-s/-S of the pjstat command)
- By referring to the job statistical information file (pmdumpjobinfo command)

This section describes how to refer to the job statistical information file with the pmdumpjobinfo command. For details on how to obtain its data with the pjsub command or pjstat command, see "Submitting a Job" and "Confirming Job Results" in "Chapter 2 Job Operation Procedures" in "Job Operation Software End-user's Guide."

To refer to the contents of the job statistical information file, use the pmdumpjobinfo command on the system management node or compute cluster management node.

```
[System management node or compute cluster management node]
# pmdumpjobinfo -c clstname
```

The pmdumpjobinfo command displays the contents of the job statistical information file in a format suited for program.
For details, see the man page for the pmdumpjobinfo command.

Information

When executing the pmdumpjobinfo command on a compute cluster management node, you can omit specification of a cluster name in the -c option.

4.3 Notes for Job Operation

This section describes the notes for job operation.

4.3.1 Job Scheduler Function

Compute cluster management nodes may continue to experience high CPU utilization with the pjsd daemon for the job scheduler function. This is normal because the job is being scheduled.

4.3.2 Job Execution Environment Customization Function

This subsection provides notes on operating the job execution environment customization function.

[Considerations for the startup processing of the job execution environment]

In the following parameters, set a longer time than the time taken by the startup processing of the job execution environment:

- DecidedGap item in the pmpjm.conf file
- RespWaitTime item in the pmrsc.conf file

The time taken by the startup processing depends on the image of the job execution environment. Adjust the above parameters by taking into account the job execution environment images (SDI specifications and UDI specifications) that can be used in the system.

Information

There is no way to calculate in advance the time taken by the startup processing for the job execution environment image. Execute in advance a job using the job execution environment. Obtain the accumulated time of the RUNNING-A state from job statistical information and use it as a reference. (For details on the accumulated time, see the item named "tratm" in the man page for pjstatsinfo(7).)

Resource allocation to interactive jobs may time out when the start time of the job execution environment is longer. If this event occurs, instruct end users to adjust the wait time (the --wait-time option of the pjsub command).

[Notes on Docker mode]

Note the following when using Docker mode.

On SDI specifications

- As a security measure, it is prohibited to execute a binary file with setuid or setgid in a job. Commands such as su cannot be used either.
- The uids are the same as on the host, but Technical Computing Suite does not map uids and user names. To use a user name in a job program, bind-mount /etc/passwd of the host OS, or prepare a user name for use in the container.
- Technical Computing Suite uses the following directories as mount points. Therefore, these directories cannot be used as mount points in the container.
  - /etc/opt/FJSVtcs
  - /var/opt/FJSVtcs
  - /usr/libexec/FJSVtcs/krm
  - /var/opt/FJSVtcs/krm/sys/fs/cgroup
  - /run/systemd/journal
  - /dev/log
- Basically, settings can be specified as create requests of the Docker API in the container startup configuration file. However, Technical Computing Suite guarantees operation only when the startup configuration file docker-image.conf is changed as shown in Table 3.36 Setting items in the job execution environment information file within the range described in "3.5.7.4 Creating a container startup configuration file (Docker mode only)," that is, the coding of "Binds" and "Devices."
  Other coding is changed at the discretion and responsibility of the administrator.
  Note the following points when changing the contents of the startup configuration file docker-image.conf:
  - Cmd
    Do not change because that would make it impossible to start as a job.

- HostConfig, Memory, MemorySwap, MemoryReservation, CpusetCpus, and CpusetMems
  Do not change these settings because they are related to job resource management.

On UDI specifications

- Since this function uses the job resource manager exit function (prealloc and postfree) to generate and delete a container image, prealloc and postfree script processing may take a relatively long time. For this reason, note the following points.

  - Control over all jobs on the relevant node has to wait until the end of processes in the prealloc and postfree scripts. Consequently, processing to allocate job resources, release job resources, interrupt a job, etc. may take a while when an interrupt command (pjdel, pjhold, or pjsig) is executed. Especially container operations of the prealloc and postfree scripts may take a while when, for example, a huge container image is specified. That may reduce the throughput of this node. Therefore, take sufficient care when using this function.

  - Resources are allocated during the period of the RUNNING-A state, and this period includes the processing time of the prealloc script. For this reason, the resource allocation wait in an interactive job may time out and the job may be canceled when the processing time of the prealloc script is extended. To prevent this, instruct users to specify an appropriate value (equal to or greater than the time taken by prealloc script processing) in the --sparam "wait-time=*time*" option of the pjsub command before submitting an interactive job.

- If a job fails to start because the user specified an invalid image, such as one without glibc, job execution is retried in the same way as with a normal job execution failure. However, the job may be repeatedly re-executed in some cases (such as after 5 failures within 60 seconds). Therefore, specify a POSIX-compatible container image.

- If a container image is specified in the environment variable PJM_JOBENV_DOCKER_IMAGE at the job submission time, the load time (time required for docker import or docker pull) increases according to the container image size. This may cause the RUNNING-A state to continue.
  Container image load processing times out in 60 seconds, which is enough time for normal applications. However, if a timeout occurs, the job transitions to the ERROR state with job end code (PJM code) 27. If such an error occurs, either provide the end user with the appropriate instructions, such as reducing the size of the container image, or refer to "Creating job resource manager exit scripts to be used when specifying UDI" in "3.5.7.7 Configuring the job resource manager exit scripts (Docker mode only)", "Setting the Image Operation Timeout" to set the docker image operation timeout value appropriately.

[Notes on KVM mode]

The Job Operation Software sets an upper limit on memory usage, but usage exceeding that limit by any job executed in KVM mode in the Job execution environment will not be detected. The behavior when excessive memory is used in a virtual machine depends on the settings of the virtual machine image file used.

Appendix A Invalid Values for Job Statistical Information at State Transition

As stated in "Table 2.6 Job information record output conditions" in "2.4.2.4 Job statistical information function," some of the output values of job statistical information in job information records may be invalid values, depending on the job information record output condition. Specifically, the job statistical information listed in the following table has invalid values in the job information records output during a job state transition from QUEUED, HOLD, or ERROR to CANCEL.
For the meaning of each job statistical information item, see the man page for the pjstatsinfo(7).

Table A.1 Invalid values for job statistical information at state transition

Job statistical information	Item name
Job script exit code	ec
Signal number when the job script ended with the receipt of a signal	sn
Job end code (PJM code)	pc
Job execution start time	sdt
Job execution end time	edt
Elapsed time from job execution start to end	elp
Number of unavailable nodes	nnumv
Number of nodes used	nnumu
Number of virtual nodes used	vnnumu
Node ID list of nodes used	nidlu
Tofu coordinate list of nodes used	tofulu
Node IDs of allocated nodes, and their corresponding rank numbers	rank
Maximum physical memory usage	mmszu
Total user CPU time spent on job execution	uctmut
Total system CPU time spent on job execution	sctmut
Total user CPU time and system CPU time	usctmut
Time when the job resource manager exit function collected data from each node	prmdt
Prologue execution start time	psdt
Prologue execution end time	pedt
Prologue script exit code	pec
Prologue execution start time	esdt
Epilogue execution end time	eedt
Epilogue script exit code	eec
Maximum physical memory usage for each virtual node	vmszu
Number of assistant cores used by the job	acnumut
Total user CPU time for assistant cores	auctmut
Total system CPU time (milliseconds) for assistant cores	asctmut
Node ID affected the job results	affectednid
Whether the job resource manager exit script prealloc was executed	prealloctrigger
Execution trigger of the job resource manager exit script prealloc	predeltrigger
Execution trigger of the job resource manager exit script postfree	postfreetrigger

Job statistical information	Item name
Exit code of the job resource manager exit script prealloc	preallocec
Exit code of the job resource manager exit script predel	predelec
Exit code of the job resource manager exit script postfree	postfreeec
Start time of the job resource manager exit script prealloc	preallocsdt
End time of the job resource manager exit script prealloc	preallocedt
Start time of the job resource manager exit script predel	predelsdt
End time of the job resource manager exit script predel	predeledt
Start time of the job resource manager exit script postfree	postfreesdt
End time of the job resource manager exit script post free	postfreeedt
Starting time of the job execution environment	jebt
Ending time of the job execution environment	jest
Number of CPU cores actually used by the job	cnumut
Receive data size for Tofu user communication	tucrb
Send data size for Tofu user communication	tucsb
Receive data size for Tofu system communication	tscrb
Send data size for Tofu system communication	tscsb
Number of Fujitsu profiler use times	fjprofiler
Number of starts of a program using the sector cache	sectorcache
Number of starts of a program using the inter-core hardware barrier	intranodebarrier
Average power consumption of the compute core group for each CMG (Estimate)	avgpcocc
Maximum power consumption of the compute core group for each CMG (Estimate)	maxpcocc
Minimum power consumption of the compute core group for each CMG (Estimate)	minpcocc
Power consumption of the compute core group for each CMG (Estimate)	ecocc
Average power consumption of the L2 cache for each CMG (Estimate)	avgpcolc
Maximum power consumption of the L2 cache for each CMG (Estimate)	maxpcolc
Minimum power consumption of the L2 cache for each CMG (Estimate)	minpcolc
Power consumption of the L2 cache for each CMG (Estimate)	ecolc
Average power consumption of the memory for each CMG (Estimate)	avgpcomc
Maximum power consumption of the memory for each CMG (Estimate)	maxpcomc
Minimum power consumption of the memory for each CMG (Estimate)	minpcomc
Memory power consumption for each CMG (Estimate)	ecomc
Average Tofu power consumption (Estimate)	avgpcot
Maximum Tofu power consumption (Estimate)	maxpcot
Minimum Tofu power consumption (Estimate)	minpcot
Tofu power consumption (Estimate)	ecot
Average power consumption at the periphery inside the CPU (Estimate)	avgpcocp
Maximum power consumption at the periphery inside the CPU (Estimate)	maxpcocp
Minimum power consumption at the periphery inside the CPU (Estimate)	minpcocp
Power consumption at the periphery inside the CPU (Estimate)	ecocp

Job statistical information	Item name
Average power consumption of the optical module (Estimate)	avgpcoom
Maximum power consumption of the optical module (Estimate)	maxpcoom
Minimum power consumption of the optical module (Estimate)	minpcoom
Power consumption of the optical module (Estimate)	ecoom
Average PCI-E power consumption (Estimate)	avgpcop
Maximum PCI-E power consumption (Estimate)	maxpcop
Minimum PCI-E power consumption (Estimate)	minpcop
PCI-E power consumption (Estimate)	ecop
Average node power consumption (Estimate)	avgpcon
Maximum node power consumption (Estimate)	maxpcon
Minimum node power consumption (Estimate)	minpcon
Node power consumption (Estimate)	econ
Average node power consumption (actual measurement)	avgpconm
Maximum node power consumption (actual measurement)	maxpconm
Minimum node power consumption (actual measurement)	minpconm
Node power consumption (actual measurement)	econm
Power knob usage information	uiopa
All of node statistical information	-

Appendix B Settings Related to Execution in MPI Processing Systems Other Than Development Studio

B.1 Settings Related to Environment Variables

The job operation administrator can set the default values of the following environment variables in the ple_env_file file (compute node: /etc/opt/FJSVtcs/ple/ple_env_file) for execution in MPI processing systems other than Development Studio:

- Environment variable: PLE_MPI_PIN_DOMAIN
- Environment variable: PLE_MPI_PIN_CELL
- Environment variable: PLE_MPI_PIN_ORDER
- Environment variable: PLE_I_MPI_PJPBIND
- Environment variable: PLE_I_MPI_PJPBIND_OPT
- Environment variable: PLE_I_MPI_INTEL_OPTIONS
- Environment variable: PLE_MEMORY_ALLOCATION_POLICY

For details on each environment variable, see this chapter and "Appendix C Executing programs of MPI processing system other than Development Studio" in "Job Operation Software End-user's Guide."

1. Edit the ple_env_file file with the job operation administrator privileges to set the following.

```
[System management node]
# vi ./ple_env_file
PLE_MPI_PIN_DOMAIN=omp
PLE_MPI_PIN_CELL=unit
PLE_I_MPI_PJPBIND=disable
...
```

2. Set the permission of the ple_env_file file as follows.

```
[System management node]
# chmod 0600 ./ple_env_file
```

Note

Set the permission securely to prevent modification by unauthorized users.

3. Distribute the edited ple_env_file file to all compute nodes executing programs of MPI processing systems other than Development Studio.

```
[System management node]
# pmscatter -c clstname --nodetype CN ./ple_env_file /etc/opt/FJSVtcs/ple/ple_env_file
```

4. Restart the compute nodes to which the ple_env_file file was distributed. This operation must be performed with cluster administrator privileges or higher.

   a. Stop the compute nodes.
   In this step shown below, specify the -w option in the papwrctl command in order to wait until the stop process of the compute nodes ends. This step covers all the compute nodes in a cluster.

```
[System management node]
# papwrctl -c clstname --nodetype CN -w -a off
```

b. Start the compute nodes.

```
# papwrctl -c clstname --nodetype CN -a on
```

B.2 Settings Related to the Wrapper Command mpiexec.tcs_intel

The parallel execution environment provides the wrapper command mpiexec.tcs_intel of the mpiexec.hydra commands of Intel MPI so that Intel MPI has the same execution view as Development Studio MPI, as described in Appendix "Executing programs of MPI processing system other than Development Studio" in "Job Operation Software End-user's Guide."

An option specified in the mpiexec.tcs_intel command has been changed to an mpiexec.hydra command option of Intel MPI based on the file for option analysis specified in the PLE_I_MPI_INTEL_OPTIONS environment variable. Specify the PLE_I_MPI_INTEL_OPTIONS environment variable with an absolute path that includes the file name.

The administrator can create a file for option analysis based on the specifications of options of the mpiexec.hydra command executed. Set the absolute path for the PLE_I_MPI_INTEL_OPTIONS environment variable when creating a file for option analysis.

The following procedure shows how to create a file for option analysis. This example uses mpiexec.hydra_options.conf.alt as the file name.

1. Create the mpiexec.hydra_options.conf.alt file as follows with job administrator privileges.

```
[System management node]
# cd <work directory>
# vi ./mpiexec.hydra_options.conf.alt
-n 1
-np 1
-perhost 1
-hostfile 1
-f 1
-machinefile 1
-machine 1
-genv 2
-genvall 0
-rr 0
-tune 0,1
-binding 1
...
* The content of the above-mentioned description is an example.
```

Write one option per line in the following format.

```
<option name> <value>
```

<*value*> represents the number of arguments that can be specified in the option. If there are multiple available arguments, specify the range with values delimited by a comma (",") or hyphen ("-"). To set no upper limit on the number of arguments, specify the character "n".

```
[Example]
-A 1,3       <- The number of arguments for the option -A is one or three.
-B 1-3       <- The number of arguments for the option -B is one to three.
-C n         <- No upper limit is placed on the number of arguments for the option -C.
```

2. Set the permission of the mpiexec.hydra_options.conf.alt file as follows.

```
[System management node]
# chmod 0644 ./mpiexec.hydra_options.conf.alt
```

Note

Set the permission securely to prevent modification by unauthorized users.

3. Distribute the mpiexec.hydra_options.conf.alt file to all compute nodes executing programs of MPI processing systems other than Development Studio. The mpiexec.hydra_options.conf.alt file can be placed on any directory on a compute node. The following example distributes the file to the /etc/opt/FJSVtcs/ple/mpiexec.tcs_intel directory.

```
[System management node]
# pmscatter -c clstname --nodetype CN ./mpiexec.hydra_options.conf.alt \
/etc/opt/FJSVtcs/ple/mpiexec.tcs_intel/mpiexec.hydra_options.conf.alt
```

4. Create the ple_env_file file, and set the absolute path of the mpiexec.hydra_options.conf.alt file in the PLE_I_MPI_INTEL_OPTIONS environment variable.
   This absolute path is the path used to place the mpiexec.hydra_options.conf.alt file on a compute node.

```
[System management node]
# vi ./ple_env_file
PLE_I_MPI_INTEL_OPTIONS=/etc/opt/FJSVtcs/ple/mpiexec.tcs_intel/mpiexec.hydra_options.conf.alt
```

5. Set the permission of the ple_env_file file as follows.

```
[System management node]
# chmod 0600 ./ple_env_file
```

Note

Set the permission securely to prevent modification by unauthorized users.

6. Distribute the ple_env_file file to the /etc/opt/FJSVtcs/ple directory on all compute nodes executing programs of MPI processing systems other than Development Studio.

```
[System management node]
# pmscatter -c clstname --nodetype CN ./ple_env_file /etc/opt/FJSVtcs/ple/ple_env_file
```

7. Restart the compute nodes to which the ple_env_file file was distributed. This operation must be performed with cluster administrator privileges or higher.

   a. Stop the compute nodes.
      In this step shown below, specify the -w option in the papwrctl command in order to wait until the stop process of the compute nodes ends. This step covers all the compute nodes in a cluster.

```
[System management node]
# papwrctl -c clstname --nodetype CN -w -a off
```

   b. Start the compute nodes.

```
[System management node]
# papwrctl -c clstname --nodetype CN -a on
```

B.3 Settings Related to the mpiexec.tcs_intel Command

If you want to define an arbitrary environment variable for all jobs, create the /etc/opt/FJSVtcs/ple/mpiexec.tcs_intel/intel.sh file and place it on compute nodes. If the file exists on a compute node, the mpiexec.tcs_intel command, when executed, reads the intel.sh file as the source command in a shell.

A job operation administrator performs this operation.

1. Create the intel.sh file.
   The following example executes the compilervar.sh and mpivars.sh scripts.

```
[System management node]
# vi ./intel.sh
/opt/intel/bin/compilervars.sh intel64
/opt/intel/impi_latest/bin64/mpivars.sh
```

2. Set the permission of the intel.sh file as follows.

```
[System management node]
# chmod 0644 ./intel.sh
```

Note

Set the permission securely to prevent modification by unauthorized users.

3. Distribute the intel.sh file to all compute nodes executing programs of MPI processing systems other than Development Studio.

```
[System management node]
# pmscatter -c clstname --nodetype CN ./intel.sh
/etc/opt/FJSVtcs/ple/mpiexec.tcs_intel/intel.sh
```

B.4 Settings when Using the Intel MPI 2019

When using Intel MPI 2019, set up the OS environment so that the host names of other compute nodes can be resolved on each compute node.

Appendix C Settings for Using GPUs [PG]

This appendix describes settings for job execution using GPUs in PRIMERGY compute nodes. The GPUs covered here are the NVIDIA V100 and A100. Below, the term "GPU" refers to the NVIDIA V100 or A100.

The Job Operation Software treats one GPU as one custom resource to implement the following in combination with the hooks of the job operation management function (job manager exit function and job resource manager exit function):

- Job allocation to GPU-equipped nodes

  Jobs can be scheduled and allocated with consideration of GPU-equipped nodes.

- Job allocation to GPUs (exclusive/shared)

  - A GPU can be exclusive to each job. This prevents performance from deteriorating due to contention.
  - A GPU can be shared by multiple jobs. If so, contention from other jobs affects job execution performance, but many users can use the GPU.

- Use of the NVIDIA Container Toolkit

  The NVIDIA Container Toolkit has tools for using GPUs from a Docker container. The toolkit can be used from a job.

- Use of the GPU MPS (Multi-Process Service) function

  The MPS function is also treated as a custom resource.

- Acquisition of GPU statistical information

  GPU statistical information (GPU utilization rate, GPU memory utilization rate, GPU power, etc.) can be acquired as job statistical information.

See

- For details on the NVIDIA Container Toolkit, see https://github.com/nvidia/nvidia-docker or other sites.
- For details on MPS, see https://docs.nvidia.com/deploy/mps/index.html or other sites.

Note

Jobs executed in normal mode or Docker mode in the job execution environment can use GPUs.

The following section describes the procedure for settings for job execution using GPUs.

C.1 Configuring Custom Resources and the Job Manager Exit Function

Configure GPUs as custom resources in order to allocate jobs to GPU-equipped nodes. This manual refers to these custom resources as "GPU custom resources."
Also, the Job Operation Software provides an exit function library. Configure the exit function library as the job manager exit function in order to acquire GPU statistical information as job statistical information.

The administrator configures custom resources and the job manager exit function in the pmpjm.conf file (system management node: /etc/opt/FJSVtcs/Rscunit.d/*ResourceUnit*/pmpjm.conf) for each resource unit.

GPU custom resources are classified as follows according to the type of GPU use.

Table C.1 Classification of GPU custom resources

Classification	Description
GPU exclusive allocation	1 GPU corresponds to 1 GPU custom resource. The GPU custom resource is allocated to a job, and the job has exclusive use of the corresponding GPU. This prevents contention from other jobs. Also, defining the MPS function for GPUs as a custom resource (MPS custom resource) and allocating the MPS custom resource will enable the MPS function to be used from a job. The MPS function allows programs that use GPUs in multiple processes, such as MPI programs, to use the GPUs effectively.
GPU shared allocation	Multiple jobs can share a GPU allocated to them. 1 GPU belongs to 1 or more GPU custom resources. So when any of these GPU custom resources is allocated to multiple jobs, the jobs can share the same GPU. Many users can execute jobs using the GPU, although they will be affected by contention from other jobs. The MPS function cannot be used with GPU shared allocation.

Note

GPU exclusive allocation and GPU shared allocation cannot both be used on the same node.

The following sections describe settings for each of these classifications.

C.1.1 GPU exclusive allocation

This section describes settings for exclusive use of a GPU by a job.

The following example has GPU custom resource settings (CustomResource section) and job manager exit function settings (ExitFunc section) for a compute node (node ID: 0xFFFF0004) equipped with four GPUs in the group_a resource group under the rscunit_pg01 resource unit.

```
[pmpjm.conf file]

ResourceUnit {
    ResourceUnitName = rscunit_pg01
    AllocType = vnode
    ResourceGroup {
        ResourceGroupName = group_a
        ResourceGroupNode = 0xFFFF0004
        CustomResource {
            Name = gpu
            ValueType = numeric
            Value = 4
            NodeID = 0xFFFF0004
        }
        ExitFunc {
            ExitFuncLib = libdevice.so
            ExitFuncType = pjm
            ExitFuncPri = 150
        }
    }
}
```

Table C.2 GPU custom resource definitions (CustomResource section)

Item name	Definition	Specified value
Name	Custom resource name	Specify "gpu" as the prefix. After "gpu", you can set any character string.

Item name	Definition	Specified value
ValueType	Custom resource management type	Specify "numeric".
Value	Number of custom resources	Specify the number of mounted GPUs on the compute node.
NodeID	Node ID	Specify the node ID of the compute node equipped with the GPUs.

Table C.3 Job manager exit function settings for acquiring GPU statistical information (ExitFunc section)

Item name	Definition	Specified value
ExitFuncLib	File name of the exit function library	Specify "libdevice.so".
ExitFuncType	Exit function library type	Specify "pjm".
ExitFucPri	Execution priority of the exit function library	Specify a numerical value greater than 127.

To use the MPS function for GPUs, also define an MPS custom resource.

```
[pmpjm.conf file]

ResourceUnit {
    ResourceUnitName = rscunit_pg01
    AllocType = vnode
    ResourceGroup {
        ResourceGroupName = group_a
        ResourceGroupNode = 0xFFFF0004
        CustomResource {
            Name = gpu
            ValueType = numeric
            Value = 4
            NodeID = 0xFFFF0004
        }
        CustomResource {
            Name = mps
            ValueType = numeric
            Value = 1
            NodeID = 0xFFFF0004
        }
        ExitFunc {
            ExitFuncLib = libdevice.so
            ExitFuncType = pjm
            ExitFuncPri = 150
        }
    }
}
```

Table C.4 MPS custom resource definitions (CustomResource section)

Item name	Define	Specified value
Name	Custom resource name	Specify "mps".
ValueType	Custom resource management type	Specify "numeric".
Value	Number of custom resources	Specify 1. Only 1 job per node can use the MPS function.
NodeID	Node ID	Specify the node ID of a compute node that is equipped with a GPU and can use the MPS function.

Information

- There is no pmpjm.conf file when the Job Operation Software is installed. If you want to change the default settings, create a new file or copy and edit the sample file (/etc/opt/FJSVtcs/sample/pmpjm.device.conf), and place the file in the above path.
- If the system contains compute nodes equipped with varying quantities or different types of GPU, you can manage them separately by writing multiple custom resource definitions.
- For details on how to configure custom resources (CustomResource section), see "3.5.1.6 Custom resource settings." For details on how to configure the job manager exit function (ExitFunc section), see "Job manager exit function and Job scheduler exit function" in "Chapter 2 Creating and Incorporating Hooks" in the "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

C.1.2 GPU shared allocation

This section describes settings for sharing of a GPU by multiple jobs.
To share one GPU among multiple GPU custom resources, configure more GPU custom resources than the number of mounted GPUs.

The following example has GPU custom resource settings (CustomResource section) and job manager exit function settings (ExitFunc section) for a compute node (node ID: 0xFFFF0004) equipped with four GPUs in the group_a resource group under the rscunit_pg01 resource unit. This example defines 12 custom resources (gpu-share) for 4 GPUs mounted on the compute node.

```
[pmpjm.conf file]

ResourceUnit {
    ResourceUnitName = rscunit_pg01
    AllocType = vnode
    ResourceGroup {
        ResourceGroupName = group_a
        ResourceGroupNode = 0xFFFF0004
        CustomResource {
            Name = gpu-share
            ValueType = numeric
            Value = 12
            NodeID = 0xFFFF0004
        }
        ExitFunc {
            ExitFuncLib = libdevice.so
            ExitFuncType = pjm
            ExitFuncPri = 150
        }
    }
}
```

Table C.5 gpu-share custom resource definitions (CustomResource section)

Item name	Definition	Specified value
Name	Custom resource name	Specify "gpu-share" as the prefix. After "gpu-share", you can specify any character string consisting of single-byte alphanumeric characters (lowercase letters), hyphens, and underscores. A custom resource name can contain up to 63 characters.
ValueType	Custom resource management type	Specify "numeric".
Value	Number of custom resources	Specify the number of GPUs to share. You can specify a number greater than the number of GPUs mounted on the compute node. In that case, 1 GPU is shared among multiple GPU custom resources. If a job shares a GPU with another job, that job will be disturbed by other jobs. For this reason, it is recommended that you specify a relatively small

Item name	Definition	Specified value
		number of custom resources (Example: 2x the installed capacity). The maximum number you can specify is 88.
NodeID	Node ID	Specify the node ID of the compute node equipped with the GPUs.

Information

The prepared sample file is /etc/opt/FJSVtcs/sample/pmpjm.device-share.conf.

Note

Even in GPU shared allocation, the maximum number of GPUs available to a single job is the number of mounted GPUs. When submitting a job, a user may request GPU custom resources exceeding the number of mounted GPUs. If so, the job enters the ERROR state after the submission is accepted.
For details on the error, see "Jobs in ERROR State" in the "Job Operation Software Troubleshooting."

C.2 Reflecting Job Operation Settings

Use the pmpjmadm command to apply the contents of the pmpjm.conf file to the system. The following example applies settings to the rscunit_pg01 resource unit under the compute cluster.

```
[System management node]
# pmpjmadm -c compute --set --rscunit rscunit_pg01
```

C.3 Configuring the Job Resource Manager Exit Function

To allocate GPUs to jobs, configure the prealloc and postfree exit scripts of the job resource manager exit function. Use these exit scripts provided by the Job Operation Software to use GPUs.

The administrator configures the exit scripts in the pmrsc.conf file (system management node: /etc/opt/FJSVtcs/Rscunit.d/*ResourceUnitName*/pmrsc.conf) for each resource unit.

The following example of the pmrsc.conf file specifies a job resource manager exit (device-hook) for exclusive use of a GPU by the group_a resource group under the rscunit_pg01 resource unit.

```
Cluster {
    ClusterName = compute
    ResourceUnit {
        ResourceUnitName = rscunit_pg01
        ResourceGroup {
        ResourceGroupName = group_a
            ExitFunc {
                ExitFuncScriptDir = /etc/opt/FJSVtcs/plugin/krm/device-hook/
                ExitFuncTimer = 300
            }
        }
    }
}
```

Table C.6 Job resource manager exit function settings for GPU exclusive allocation

Item name	Definition	Specified value
ExitFuncScriptDir	Directory for placing exit scripts	Specify /etc/opt/FJSVtcs/plugin/krm/device-hook/.
ExitFuncTimer	Timeout value for the exit script execution time	Specify 300.

Modify the following files in system management node:

/etc/opt/FJSVtcs/plugin/krm/device-hook/prealloc

The change is 3 lines with **

```
# nvidia-docker <*****>
rpm -qa | grep nvidia-container-runtime > /dev/null 2>&1        ** delete
if [ "$?" = 0 ]; then                                           ** delete
if [ -e "/usr/bin/nvidia-container-runtime" ]; then             ** add
    OPT="nvidia-docker"
else
```

<*****> : This part is garbled because it has a Japanese comment.

After the change, set the job resource management exit function.

See

For details on configuring the job resource manager exit function, see "Job resource management exit function" in "Chapter 2 Creating and Incorporating Hooks" in the "Job Operation Software Administrator's Guide for Job Operation Manager Hook."

Use the pmrscadm command to apply the contents of the pmrsc.conf file to the system. The following example applies settings to the rscunit_pg01 resource unit.

```
[System management node]
# pmrscadm --set --rscunit rscunit_pg01
```

C.4 Changing Startup Options for the Job Execution Environment

C.4.1 Using the GPUs in normal mode

When jobs are executed in normal mode in the job execution environment, all of the jobs can access GPUs because all devices are available by default. Therefore, in the startup configuration file /etc/opt/FJSVtcs/krm/tcs-bare.conf for normal mode installed on the compute node, set only the necessary devices other than GPU to item Devices in the following procedure. As a result, jobs cannot access GPUs by default. When you want to use a GPU, allocate GPU custom resources to a job and execute the job.

Note

Do not modify the startup configuration file tcs-bare.conf for normal mode except for the item Devices. Operation cannot be guaranteed if any item other than Devices is changed.

In the example shown below, a compute node (node ID: 0xFFFF0004) in the compute cluster named "compute" can access only /dev/sda1 from the container.

1. Preparing for software maintenance

   Make preparations to change startup options for the job execution environment.

   For the preparation procedure, see "Preparation for Software Maintenance" in the "Job Operation Software Administrator's Guide for Maintenance."

2. Transferring tcs-bare.conf

   Transfer /etc/opt/FJSVtcs/krm/tcs-bare.conf from the compute node to the system management node as follows.

   ```
   [System management node]
   # pmgather -c compute -n 0xFFFF0004 /etc/opt/FJSVtcs/krm/tcs-bare.conf ./
   ```

3. Editing tcs-bare.conf

   The Devices item in the transferred tcs-bare.conf file looks like the following.

```
...
        "Devices"       : [{
                              "PathOnHost" : "/dev ",
                              "PathInContainer" : "/dev ",
                              "CgroupPermissions" : "mrw"
                           }],
...
```

4. Edit the tcs-bare.conf file as follows.

```
...
        "Devices"       : [{
                              "PathOnHost" : "/dev/sda1",
                              "PathInContainer" : "/dev/sda1",
                              "CgroupPermissions" : "mrw"
                           }],
...
```

5. Setting the permission for the tcs-bare.conf file

   Set the permission for the tcs-bare.conf file as follows.

```
[System management node]
# chmod 0600 tcs-bare.conf
```

6. Distributing the tcs-bare.conf file

   Distribute the tcs-bare.conf file to the target compute node as follows.

```
[System management node]
# pmscatter -c compute -p -n 0xFFFF0004 tcs-bare.conf /etc/opt/FJSVtcs/krm/tcs-bare.conf
```

7. Incorporating into operation after software maintenance

   Incorporate the node with the changed settings into operation.
   For the procedure for incorporating into operation, see "Incorporating Into Operation After Software Maintenance" in the "Job Operation Software Administrator's Guide for Maintenance."

Note

If you want to use GPUs in Docker mode, similar settings are required in the startup configuration file used.

C.4.2 Using the MPS function in docker mode

When using the GPU MPS feature in Docker mode, Docker containers access /tmp on the host.

Therefore, the definition must be added to "Binds" in the startup configuration file of the job execution environment to allow access to /tmp.
Here is an example of adding the definition for /tmp to "Binds".

```
...
"HostConfig"        : {
        "Binds"         : [
                              ... ,
                              "/tmp:/tmp"
                          ],
...
```

See

See "3.5.7.4 Creating a container startup configuration file (Docker mode only)" for more information about the container startup configuration file.

C.5 Configuring Job Statistical Information

Configure job statistical information so that GPU statistical information can be output as job statistical information.

The administrator defines GPU statistical information as job statistical information items in the papjmstats.conf file (system management node: /etc/opt/FJSVtcs/papjmstats.conf). The following GPU statistical information can be defined.

Table C.7 GPU statistical information that can be defined

Item name	Description	RecordNameList	DataType	Example
gpuids	ID list of GPUs used by jobs	JN	PJMX_DATATYPE_STRING	0,1,2,3
gpu_driver_version	Version of the GPU driver	JN	PJMX_DATATYPE_STRING	48.67
gpu_cuda_version	Version of CUDA	JN	PJMX_DATATYPE_STRING	10.1
gpu_name_*n*	Name of a GPU (ID: *n*)	JN	PJMX_DATATYPE_STRING	Tesla V100-PCIE-32GB
gpu_busid_*n*	Bus ID of a GPU (ID: *n*)	JN	PJMX_DATATYPE_STRING	00000000:3E:00.0
gpu_perf_*n*	Performance status of a GPU (ID: *n*)	JN	PJMX_DATATYPE_STRING	P0
gpu_persistence_*n*	Persistence mode status of a GPU (ID: *n*)	JN	PJMX_DATATYPE_STRING	Disabled
gpu_display_*n*	Display mode status of a GPU (ID: *n*)	JN	PJMX_DATATYPE_STRING	Disabled
gpu_compute_*n*	Compute mode status of a GPU (ID: *n*)	JN	PJMX_DATATYPE_STRING	Default
gpu_ave_util_*n*	Average utilization rate (%) of a GPU (ID: *n*) in the job execution time	JN	PJMX_DATATYPE_STRING	60%
gpu_max_util_*n*	Maximum utilization rate (%) of a GPU (ID: *n*) in the job execution time	JN	PJMX_DATATYPE_STRING	90%
gpu_total_mem_*n*	Total memory amount (MiB) of a GPU (ID: *n*)	JN	PJMX_DATATYPE_STRING	16130 MiB
gpu_ave_mem_*n*	Average memory usage (MiB) of a GPU (ID: *n*) in the job execution time	JN	PJMX_DATATYPE_STRING	7000 MiB
gpu_max_mem_*n*	Maximum memory usage (MiB) of a GPU (ID: *n*) in the job execution time	JN	PJMX_DATATYPE_STRING	14679 MiB
gpu_power_cap_*n*	Upper limit on the power consumption (W) of a GPU (ID: *n*)	JN	PJMX_DATATYPE_STRING	300.00 W
gpu_ave_pwr_*n*	Average power consumption (W) of a	JN	PJMX_DATATYPE_STRING	56.00 W

Item name	Description	RecordNameList	DataType	Example
	GPU (ID: *n*) in the job execution time			
gpu_max_pwr_*n*	Maximum power consumption (W) of a GPU (ID: *n*) in the job execution time	JN	PJMX_DATATYPE_STRING	218.00 W
gpu_ave_temp_*n*	Average temperature (Celsius) of a GPU (ID: *n*) in the job execution time	JN	PJMX_DATATYPE_STRING	38 C
gpu_max_temp_*n*	Maximum temperature (Celsius) of a GPU (ID: *n*) in the job execution time	JN	PJMX_DATATYPE_STRING	43 C
gpu_volatile_ecc_*n*	Memory error correction count of a GPU (ID: *n*) in the job execution time	JN	PJMX_DATATYPE_STRING	0
gpu_ave_util	Average value (%) of gpu_ave_util_*n*	JI	PJMX_DATATYPE_STRING	60%
gpu_max_util	Maximum value (%) of gpu_max_util_*n*	JI	PJMX_DATATYPE_STRING	90%
gpu_ave_mem	Average value (MiB) of gpu_ave_mem_*n*	JI	PJMX_DATATYPE_STRING	700 MiB
gpu_max_mem	Maximum value (MiB) of gpu_max_mem_*n*	JI	PJMX_DATATYPE_STRING	14679 MiB
gpu_ave_pwr	Average value (W) of gpu_ave_pwr_*n*	JI	PJMX_DATATYPE_STRING	56.64 W
gpu_max_pwr	Maximum value (W) of gpu_max_pwr_*n*	JI	PJMX_DATATYPE_STRING	218.00 W
gpu_ave_temp	Average value (Celsius) of gpu_ave_temp_*n*	JI	PJMX_DATATYPE_STRING	38 C
gpu_max_temp	Maximum value (Celsius) of gpu_max_temp_*n*	JI	PJMX_DATATYPE_STRING	43 C

The following example of the papjmstats.conf file adds the GPU IDs of the GPUs used by a job and the GPU names of two GPUs to job statistical information.

```
Cluster {
    Item {
        ItemName = gpuids
        RecordNameList = JN
        DataType = PJMX_DATATYPE_STRING
    }
    Item {
        ItemName = gpu_name_0
        RecordNameList = JN
        DataType = PJMX_DATATYPE_STRING
    }
    Item {
        ItemName = gpu_name_1
        RecordNameList = JN
        DataType = PJMX_DATATYPE_STRING
    }
    ...
}
```

Use the papjmstatsadm command to apply the contents of the papjmstats.conf file. The following example applies settings to the compute cluster.

```
[System management node]
# papjmstatsadm -c compute --set
```

Information

The prepared sample file is /etc/opt/FJSVtcs/sample/papjmstats.device.conf.

See

- For details on how to define job statistical information, see "3.4.2 Settings for job statistical information in a cluster (papjmstats.conf file)."
- For details on GPU statistical information, see https://developer.nvidia.com/nvidia-system-management-interface.

Note

- For a system containing a compute node equipped with multiple GPUs, as many statistical information definitions as the number of GPUs must be defined.
- When a GPU is shared among multiple jobs, the GPU load, memory usage, etc. by the jobs using the same GPU are information that may be included.

C.6 Job Submission Options

This section describes how to submit a job that uses a GPU defined as a custom resource. The definition of custom resource varies from system to system. Provide end-users with guidance appropriate to the system.

C.6.1 Specifying GPU custom resources

To allocate a GPU defined as the gpu custom resource to one job (job.sh), submit the job as follows.

```
$ pjsub -L gpu=1 job.sh
```

C.6.2 Environment variable for the NVIDIA Container Toolkit

For the NVIDIA Container Toolkit, you can specify an environment variable for changing the operation of containers when submitting a job. To do so, use the -x option of the pjsub command.
For details on the environment variable for the NVIDIA Container Toolkit, see https://github.com/NVIDIA/nvidia-container-runtime. Add "PJM_" to the beginning of the environment variable name when specifying the environment variable in the -x option of the pjsub command.

The following example submits the job.sh job, which uses one gpu custom resource, and specifies "cuda>=8.0" in the environment variable PJM_NVIDIA_REQUIRE_CUDA.

```
$ pjsub -x PJM_NVIDIA_REQUIRE_CUDA="cuda>=8.0" -L gpu=1 job.sh
```

Table C.8 Example of an environment variable for the NVIDIA Container Toolkit

Environment variable name	Definition	Specification example
PJM_NVIDIA_REQUIRE_CUDA	Specify the version of the CUDA toolkit used in the container. After the submission of a job is accepted, the job enters the ERROR state if it cannot use the specified version.	cuda>=8.0

C.6.3 Environment variable for GPU statistical information

GPU time-series statistical information can be output as a statistical information file when a job ends. If you want to output the statistical information file, specify the environment variable PJM_STATS_DIR, which specifies the file output directory, by using the -x option of the pjsub command when submitting a job.

The following example specifies the nvidia_stats directory so that GPU time-series statistical information is output there.

```
$ pjsub -x PJM_STATS_DIR="nvidia_stats" -L gpu=1 job.sh
```

Table C.9 Environment variable for GPU statistical information

Environment variable name	Definition	Specification example
PJM_STATS_DIR	Specify the statistical information file output directory as a relative path from the current directory at the job submission time. After the submission of a job is accepted, the job enters the ERROR state if it does not have access permission for the specified path.	nvidia_stats

nvidiad_*job ID_node ID*.out is the file name of the statistical information file. In the following example, the job ID is 1024 and the node ID is 0xFFFF0003.

```
$ ls nvidia_stats
nvidiad_1024_0xFFFF0003.out
```

Appendix D Settings for Using Singularity [PG]

Singularity is container virtualization software for HPC.
The Job Operation Software can execute Singularity as a job in Docker mode in the job execution environment when the following settings have been completed:

1. Job execution environment settings
2. Startup configuration file settings
3. Custom resource settings

See

For details on Singularity, see https://sylabs.io/singularity/ or other sites.

Note

Jobs that execute Singularity cannot be executed with root privileges.

D.1 Job Execution Environment Settings

For Singularity, create a jobenv.conf information file for the job execution environment.

In the definitions for Singularity, specify /etc/opt/FJSVtcs/krm/singularity.conf in the Conf item in the startup configuration file. The following example shows definitions for Singularity.

```
[
    ...
    {
        "Name"      : "singularity",
        "Type"      : "docker",
        "Image"     : "tcs-bare",
        "Conf"      : "/etc/opt/FJSVtcs/krm/singularity.conf"
    }
    ...
]
```

Table D.1 Job execution environment setting items for Singularity

Item name	Definition	Specified value
Name	Job execution environment name	Specify a name identifying the job execution environment. You can specify any character string consisting of 1 to 63 characters, including single-byte alphanumeric characters (lowercase letters), hyphens, and underscores. However, the first character must be a single-byte alphanumeric character. The specified name cannot be a duplicate of another entry.
Type	Job execution environment type	Specify "docker".
Image	Container image used	Specify "tcs-bare".
Conf		Specify the startup configuration file /etc/opt/FJSVtcs/krm/singularity.conf.

Information

The startup configuration file /etc/opt/FJSVtcs/krm/singularity.conf specified in the Conf item is placed when the Job Operation Software is installed.

D.2 Startup Configuration File Settings

Follow these steps to configure the /etc/opt/FJSVtcs/krm/singularity.conf file on the system management node and then distribute it to each Compute node.

1. loop device setting

   Singularity must be able to access loop devices (such as /dev/loop0). When editing the Devices item in the startup configuration file /etc/opt/FJSVtcs/krm/singularity.conf for Singularity, make loop devices accessible as shown below. The following example makes /dev/loop0 and /dev/loop1 accessible.

```
...
        "Devices"       : [{
                              "PathOnHost" : "/dev/loop0 ",
                              "PathInContainer" : "/dev/loop0",
                              "CgroupPermissions" : "mrw"
                           },
                           {
                              "PathOnHost" : "/dev/loop1 ",
                              "PathInContainer" : "/dev/loop1",
                              "CgroupPermissions" : "mrw"
                           }
        ],
...
```

   Since at least one loop device is required per Singularity process, define multiple loop devices existing on compute nodes.

2. When using docker 19.03 and Singularity 3.7.2 or later

   To grant permission to run jobs on Singularity, set the following:

   a. Settings IpcMode

      Add the "IpcMode" setting to the "HostConfig" section on the system management node. Below are additional examples, put ** at the add part.

```
        {
          "Hostname"          : "",
                    (snip)
          "HostConfig"        : {
              "Binds"         : [],
                    (snip)
              "NetworkMode"   : "host",
              "IpcMode"       : "shareable",  ** Add **
              "Devices"       : [{
```

   b. Settings CapAdd and CapDrop

      In the "CapDrop" section of the "HostConfig" section, move "DAC_READ_SEARCH" and "SYS_PTRACE" to the "CapAdd" section. Below are description of "CapAdd", "CapDrop" section after moved.

      The part marked with ** in the following description is the changed part.

```
        "CapAdd"        : [
                              "CHOWN",
                              "DAC_OVERRIDE",
                              "FSETID",
                              "FOWNER",
                              "MKNOD",
                              "NET_RAW",
                              "SETGID",
                              "SETUID",
                              "SETFCAP",
                              "SETPCAP",
                              "NET_BIND_SERVICE",
                              "SYS_CHROOT",
                              "KILL",
                              "AUDIT_WRITE",
                              "SYS_ADMIN",
                              "SYS_NICE",
                              "DAC_READ_SEARCH",      ** Move from "CapDrop" section **
                              "SYS_PTRACE",           ** Move from "CapDrop" section **
                              "SYS_RESOURCE"
                          ],
        "CapDrop"       : [
                              "AUDIT_CONTROL",
                              "BLOCK_SUSPEND",
                              "IPC_LOCK",
                              "IPC_OWNER",
                              "LEASE",
                              "LINUX_IMMUTABLE",
                              "MAC_ADMIN",
                              "MAC_OVERRIDE",
                              "NET_ADMIN",
                              "NET_BROADCAST",
                              "SYS_BOOT",
                              "SYS_MODULE",
                              "SYS_PACCT",
                              "SYS_RAWIO",
                              "SYS_TIME",
                              "SYS_TTY_CONFIG",
                              "SYSLOG",
                              "WAKE_ALARM"
                          ],
```

See

For details on how to configure a job execution environment, see "3.5.7 Configuring a Job Execution Environment."

D.3 Custom Resource Settings

The jobenv.conf information file defines a job execution environment. Define the job execution environment as the jobenv custom resource.

1. Correcting the pmpjm.conf file

   Specify the following as definition items in the CustomResource subsection.

Item name	Specification
Name	Specify "jobenv".
ValueType	Specify "string".
Value	Specify the job execution environment name.

   The following example configures the job execution environment "singularity" as the custom resource "jobenv" for the resource unit "rscunit_pg01".

```
[System management node]
# vi /etc/opt/FJSVtcs/Rscunit.d/rscunit_pg01/pmpjm.conf
ResourceUnit {
    ResourceUnitName = group_pg01
    ...
    CustomResource {
        Name = jobenv
        ValueType = string
        Value = singularity
    }
...
}
```

Information

The prepared sample file is /etc/opt/FJSVtcs/sample/pmpjm.singularity.conf.

2. Reflecting the pmpjm.conf file

   Use the pmpjmadm command to apply the contents of the pmpjm.conf file to the system. The following example applies settings to the rscunit_pg01 resource unit under the compute cluster.

```
[System management node]
# pmpjmadm -c compute --set --rscunit rscunit_pg01
```

Appendix E How to Use Dynamic Parameters in Startup Configuration Files (Docker Mode) [PG]

The Binds item in the startup configuration file of a job execution environment specifies a mount point. Instead of a fixed character string as the specified mount point, an alternative method applies values (dynamic parameters) appropriate to the environment at the job runtime (Docker mode). (See "3.5.7.4 Creating a container startup configuration file (Docker mode only).")
However, this specification method that uses dynamic parameters is not available by default. This appendix describes settings for using dynamic parameters.

The dynamic parameter settings are implemented using the job resource manager exit function. Therefore, to use dynamic parameters, the job resource manager exit function must be configured.
As the administrator, configure the job resource manager exit function in the pmrsc.conf file (system management node: /etc/opt/FJSVtcs/Rscunit.d/*resource unit name*/pmrsc.conf) for each resource unit.

In the following example, the pmrsc.conf file specifies an exit script (under docker-hook) for the job resource manager exit function to exclusively allocate a GPU to the group_a resource group under the rscunit_pg01 resource unit.

```
Cluster {
    ClusterName = compute
    ResourceUnit {
        ResourceUnitName = rscunit_pg01
        ResourceGroup {
        ResourceGroupName = group_a
            ExitFunc {
                ExitFuncScriptDir = /etc/opt/FJSVtcs/plugin/krm/docker-hook/
                ExitFuncPri = 150
            }
        }
    }
}
```

Table E.1 Job resource manager exit function settings for using dynamic parameters

Item name	Definition	Specified value
ExitFuncScriptDir	Directory for placing exit scripts	Specify /etc/opt/FJSVtcs/plugin/krm/docker-hook/.
ExitFuncPri	Priority of an executed exit script	Specify a numerical value greater than 127.

Information

- The prepared sample file is /etc/opt/FJSVtcs/sample/pmrsc.docker.conf.
- Exit scripts are placed in the /etc/opt/FJSVtcs/plugin/krm/docker-hook directory when the Job Operation Software is installed.

Note

When the number of OS version of PRIMERGY Compute node is RHEL8 series

When running a job using Docker mode on a PRIMERGY compute node with an OS version of RHEL8, change the function __render_string in the /etc/opt/FJSVtcs/plugin/krm/docker-hook/prealloc file located on the system management node to the following and then configure the job resource manager exit function.

Two additional lines and three changed lines with ** in the following description.

```
def __render_string(datalist, envdict):
    """<*****>"""
    try:
        # datalist : <*****>
        # envdict  : <*****>
        env = jinja2.Environment(loader=jinja2.BaseLoader(),
                                 undefined=jinja2.StrictUndefined,
                                 keep_trailing_newline=True)
        envdata = os.environ.copy()

        # <*****>
        # PJM_USER, PJM_GROUP
        envdata["PJM_USER"] = __get_username()
        envdata["PJM_GROUP"] = __get_groupname()
        envdata.update(envdict)

        # <*****>
        newenvdata={} #                            ** Add **
        for key, val in envdata.items():
            # <*****>
            if not key.startswith("PJM"):
                continue #                         ** Change del envdata[key] **
            # <*****>
            elif not val:
                continue #                         ** Change del envdata[key] **
            # <*****>
            elif key == "PJM_APPNAME":
                # <*****>
                if "/.." in val or "../" in val or val == "..":
                    err_msg = ("[ERR.] KRM 1002 {command} INVALID OPTION.(PJM_APPNAME=" + val + ")")
                    raise devutil.common.EnvInvalidError(err_msg)
            elif key == "PJM_FSNAME":
                # <*****>
                if "/.." in val or "../" in val or val == "..":
                    err_msg = ("[ERR.] KRM 1002 {command} INVALID OPTION.(PJM_FSNAME=" + val + ")")
                    raise devutil.common.EnvInvalidError(err_msg)

            newenvdata[key]=val #                  ** Add **
        try:
            # list to str
            string = json.dumps(datalist)
        except TypeError:
            err_msg = ("[ERR.] KRM 1303 {command} INVALID FORMAT {conffile} ")
            err_msg += ("(FAILED : type conversion from list to str)")
            raise devutil.common.InvalidContainerFormat(err_msg)

        # <*****>
        template = env.from_string(string)
        newdata = template.render(**newenvdata) #  ** Change newdata = template.render(**envdata) **
        try:
            # str to list
            datalist = json.loads(newdata)
        except ValueError:
            err_msg = ("[ERR.] KRM 1303 {command} INVALID FORMAT {conffile} ")
            err_msg += ("(FAILED : type conversion from str to list)")
            raise devutil.common.InvalidContainerFormat(err_msg)

        return datalist

    except (jinja2.exceptions.TemplateSyntaxError, jinja2.exceptions.UndefinedError) as ex:
        msg = str(ex)
        err_msg = ("[ERR.] KRM 1304 {command} INVALID VARIABLE {conffile} (" + msg + ")")
        raise devutil.common.InvalidVariable(err_msg)
```

<*****> : This part is garbled because it has a Japanese comment.

The pmrscadm command applies the contents of the pmrsc.conf file to the system. The following example applies settings to the rscunit_pg01 resource unit.

```
[System management node]
# pmrscadm --set --rscunit rscunit_pg01
```

Appendix F Defined items of the job ACL function

This appendix describes the defined items of the job ACL function.

The items that can be defined vary depending on the definition target. The following tables list the defined items available for each of the USER definitions, GROUP definitions, and ALL definitions.

CL, RU, and RG indicate whether a defined item can be specified in a cluster range, resource unit range, and resource group range, respectively.
"E" indicates an exclusive defined item, "C" indicates a definition value common in the target range, and "x" indicates an item that cannot be specified for this range.

Note

- In PRIMERGY server, specified setting items for FX server are ignored.
- In FX server, specified setting items for PRIMERGY are ignored.

Table F.1 Defined items available in the USER definitions (1)

Definition name	Description	Default value	Specifiable layer		
			CL	RU	RG
limit ru-accept	Concurrent job acceptance limit within a resource unit (*1)	unlimited	C	E	x
limit ru-accept-allsubjob	Concurrent sub job acceptance limit within a resource unit (*1) A normal job of batch job is counted as one sub job.	unlimited	C	E	x
limit ru-accept-bulksubjob	Concurrent bulk sub job acceptance limit within a resource unit (*1)	unlimited	C	E	x
limit ru-accept-stepsubjob	Concurrent step sub job acceptance limit within a resource unit (*1)	unlimited	C	E	x
limit ru-run-job	Concurrent job execution limit within a resource unit (*1)	unlimited	C	E	x
limit ru-run-bulksubjob	Concurrent bulk sub job execution limit within a resource unit (*1)(*2)	unlimited	C	E	x
limit ru-use-node	Concurrent node usage limit within a resource unit (*1)(*3)(*12)	unlimited	C	E	x
limit ru-use-core	Concurrent CPU core usage limit within a resource unit (*1)(*4)(*13)	unlimited	C	E	x
limit ru-custom-*CustomResourceName*	Limit on the number of custom resources (defined in *CustomResourceName*) that can be used concurrently within a resource unit	unlimited	C	E	x
limit ru-interact-accept	Concurrent job acceptance limit (interactive jobs) within a resource unit (*1)	unlimited	C	E	x
limit ru-interact-run-job	Concurrent job execution limit (interactive jobs) within a resource unit (*1)	unlimited	C	E	x
limit ru-interact-use-node	Concurrent node usage limit (interactive jobs) within a resource unit (*1)(*3)(*12)	unlimited	C	E	x
limit ru-interact-use-core	Concurrent CPU core usage limit (interactive jobs) within a resource unit (*1)(*4)(*13)	unlimited	C	E	x

Definition name	Description	Default value	Specifiable layer		
			CL	RU	RG
limit ru-interact-custom-*CustomResourceNam*e	Limit on the number of custom resources (defined in *CustomResourceName*) that can be used concurrently (by interactive jobs) within a resource unit	unlimited	C	E	x
limit rg-accept	Concurrent job acceptance limit within a resource group (*1)	unlimited	C	C	E
limit rg-accept-allsubjob	Concurrent sub job acceptance limit within a resource group(*1) A normal job of batch job is counted as one sub job.	unlimited	C	C	E
limit rg-accept-bulksubjob	Concurrent bulk sub job acceptance limit within a resource group (*1)	unlimited	C	C	E
limit rg-accept-stepsubjob	Concurrent step sub job acceptance limit within a resource group (*1)	unlimited	C	C	E
limit rg-run-job	Concurrent job execution limit within a resource group (*1)	unlimited	C	C	E
limit rg-run-bulksubjob	Concurrent bulk sub job execution limit within a resource group (*1)(*12)	unlimited	C	C	E
limit rg-use-node	Concurrent node usage limit within a resource group (*1)(*3)(*12)	unlimited	C	C	E
limit rg-use-core	Concurrent CPU core usage limit within a resource group (*1)(*4)(*13)	unlimited	C	C	E
limit rg-custom-*CustomResourceName*	Limit on the number of custom resources (defined in *CustomResourceName*) that can be used concurrently within a resource group	unlimited	C	C	E
limit rg-interact-accept	Concurrent job acceptance limit (interactive jobs) within a resource group (*1)	unlimited	C	C	E
limit rg-interact-run-job	Concurrent job execution limit (interactive jobs) within a resource group (*1)	unlimited	C	C	E
limit rg-interact-use-node	Concurrent node usage limit (interactive jobs) within a resource group (*1)(*3)(*12)	unlimited	C	C	E
limit rg-interact-use-core	Concurrent CPU core usage limit (interactive jobs) within a resource group (*1)(*4)(*13)	unlimited	C	C	E
limit rg-interact-custom-*CustomResourceName*	Limit on the number of custom resources (defined in *CustomResourceName*) that can be used concurrently (by interactive jobs) within a resource group	unlimited	C	C	E
define rscunit	Default name for a submitting resource unit	rscunit000	E	x	x
define rscgroup	Default name for a submitting resource group	def_grp	C	E	x
define interact-rscunit	Default name for a submitting resource unit (interactive job)	rscunit000	E	x	x
define interact-rscgroup	Default name for a submitting resource group (interactive job)	def_grp	C	E	x

Definition name	Description	Default value	Specifiable layer		
			CL	RU	RG
define pri	User priority within a resource unit. (*1) It is valid when user_prio of the job selection policy is set.	127	C	C	E
define ingroup-pri	User priority in the same group within a resource unit. It is valid when usr_in_grp_prio of the job selection policy is set.	127	C	C	E
define fshare-init	Default user fair share value within a resource unit. (*1) It is valid when user_fairshare of the job selection policy is set. (*5) For details on how to specify a fair share set to define initial fair share values, see "3.4.4.4 How to define a fair share set."	0	C	E	x
define fshare-recovery	Fair share recovery factor for users within a resource unit (*1) It is valid when user_fairshare of the job selection policy is set. (*5) For details on how to specify a fair share set to define initial fair share values, see "3.4.4.4 How to define a fair share set."	100	C	E	x
define ingroup-fshare-init	Default user fair share value in the same group within a resource unit. It is valid when usr_in_grp_fairshare of the job selection policy is set. (*5) For details on how to specify a fair share set to define initial fair share values, see "3.4.4.4 How to define a fair share set."	0	C	E	x
define ingroup-fshare-recovery	Fair share recovery factor for users in the same group within a resource unit. It is valid when usr_in_grp_fairshare of the job selection policy is set. (*5) For details on how to specify a fair share set to define initial fair share values, see "3.4.4.4 How to define a fair share set."	100	C	E	x
define allocation-mode [FX]	Node allocation mode for node allocated jobs. The specifiable values include torus (torus mode), mesh (mesh mode), and noncont (non-contiguous mode). (*10)	torus	C	C	E
define numa-policy	NUMA allocation policy : pack, unpack	pack	C	C	E
define load-policy [PG]	Method of selecting nodes. balancing (Distribution), concentration (concentration) (*6)	balancing	C	C	E
define vn-policy [PG]	Virtual node placement policy abs_pack, pack, abs_unpack[=n], unpack[=n] (*6)	pack	C	C	E
define exec-policy [PG]	Execution mode simplex (node occupation), share (share) (*6)	share	C	C	E

Definition name	Description	Default value	Specifiable layer		
			CL	RU	RG
define node-priority [PG]	Priority control of allocated nodes (0 - 63) (*6)	31	C	C	E
define alloc-granularity	Granularity in node resource allocation node (node), vnode (virtual node) The node allocated job or virtual node allocated job is determined based on this setting unless both the -L node and -L vnode options are specified when a job is submitted with the pjsub command. node: node allocated jobs The default values of the joblimit node and joblimit interact-node items are used as the number of nodes. vnode: virtual node allocated jobs The default values of the joblimit vnode and joblimit interact-vnode items are used as the number of virtual nodes.	node	C	C	E
define assign-logical-cpu [PG]	Range types for the logical CPUs that can be used for job processes: job, all (*15) job Job processes can use only the logical CPUs for the job in the allocated CPU core. all Job processes can use all the logical CPUs in the allocated CPU core. For details on the logical CPUs for a job, see "Setting the Range of CPU Resources Available to a Job" in "Executing programs of MPI processing system other than Development Studio" in "Job Operation Software End-user's Guide."	job	C	C	E
define pjstat-display-mode	Mode for the pjstat command displaying other users' jobs without access privileges anonymous Display a list of jobs and also include the jobs in summary information. However, mask some information, such as user names, so that is not known. nothing Neither display a list of jobs nor include the jobs in summary information summary Do not display a list of jobs, but include the jobs in summary information	anonymous	C	C	E
define elapsed-time-mode	Method of specifying the elapsed time limit of a job - fixed: Specify only a maximum value	fixed	C	C	E

Definition name	Description	Default value	Specifiable layer		
			CL	RU	RG
	- adaptive: Specify minimum and maximum values [FX]				
define pjstat-sdt-format	Display format for planned job execution start times of the pjstat command fine custom=*<format>* For details on how to specify the format, see "3.4.4.5 Changing the display format of planned job execution start times."	fine	C	C	E
define pjstat-sdt-mark	Display of scheduling symbols for planned job execution start times displayed by the pjstat command @ Display the @ symbol for jobs with a specified start time < Display the < symbol for jobs that are backfill scheduled # Display the # symbol when the scheduling period is exceeded all Display all the symbols (@, <, and #) nothing Display none of the symbols (@, <, and #) For details on how to specify it, see "3.4.4.5 Changing the display format of planned job execution start times."	all	C	C	E
define strict-mode [FX]	Default settings for the strict, strict-io parameters of the pjsub command. nostrict : Neighther strict nor strict-io is specified. strict : strict is specified strict-io : strcit-io is specified	nostrict	C	C	E
define net-route [FX]	Whether to dynamically change the communication path when a Tofu interconnect link goes down during job execution dynamic: Dynamically change the communication path. static: Do not change the communication path. This item is invalid for jobs executed on a PRIMERGY server.	static	C	C	E

Definition name	Description	Default value	Specifiable layer		
			CL	RU	RG
define mpiexec-stdouterr-unit [FX]	Output units for the standard output/ standard error output of the mpiexec command. - mpiexec For each mpiexec command. - proc For each process (rank)	mpiexec	C	C	E
define mpiexec-stdout [FX]	The default output destination for the standard output of the mpiexec command in a batch job. (*20)	%o	C	C	E
define mpiexec-stderr [FX]	The default output destination for the standard error output of the mpiexec command in a batch job. (*20)	%e	C	C	E
define interact-mpiexec-stdout [FX]	The default output destination for the standard output of the mpiexec command in an interactive job. (*20)	noset	C	C	E
define interact-mpiexec-stderr [FX]	The default output destination for the standard error output of the mpiexec command in an interactive job. (*20)	noset	C	C	E
define mpiexec-std-emptyfile [FX]	Whether to create an empty file if there is no standard output/standard error output for the mpiexec command. - on : Creates - off : Do not create. - force-on : Creates regardless of user specification. - force-off : Do not creates regardless of user specification.	on	C	C	E
execute pjsub	Authority for execution of the pjsub command	enable	C	C	E
execute pjsub-interact	Authority for submitting interactive jobs	enable	C	C	E
execute pjsub-step	Authority for submitting step jobs	enable	C	C	E
execute pjsub-bulk	Authority for submitting bulk jobs	enable	C	C	E
execute pjsub-fixed-elapsed-time [FX]	Authority for executing jobs specifying only an upper limit for the elapsed time limit value for a job (pjsub -L elapse=*limit*)	enable	C	C	E
execute pjsub-adaptive-elapsed-time [FX]	Authority for executing jobs specifying a range of elapsed time limit values for a job (extra time specified) (pjsub -L elapse=*min_limit-max_limit*)	disable	C	C	E
execute pjstat	Authority for executing the pjstat command	enable	E	x	x
execute pjdel	Authority for executing the pjdel command	enable	E	x	x
execute pjhold	Authority for executing the pjhold command	enable	E	x	x

Definition name	Description	Default value	Specifiable layer		
			CL	RU	RG
execute pjrls	Authority for executing the pjrls command	enable	E	x	x
execute pjwait	Authority for executing the pjwait command	enable	E	x	x
execute pjalter	Authority for executing the pjalter command	enable	E	x	x
execute pjsig	Authority for executing the pjsig command	enable	E	x	x
execute pjacl	Authority for executing the pjacl command	enable	E	x	x
execute mpiexec-std	Authority for using the -of/-std option of the mpiexec command to change the standard output and standard error output destinations of parallel processes for batch jobs If disable is set, the -of/-std option of the mpiexec command is ignored.	enable	C	C	E
execute mpiexec-stdout	Authority for using the -ofout/-stdout option of the mpiexec command to change the standard output destination of parallel processes for batch jobs If disable is set, the -ofout/-stdout option of the mpiexec command is ignored.	enable	C	C	E
execute mpiexec-stderr	Authority for using the -oferr/-stderr option of the mpiexec command to change the standard error output destination of parallel processes for batch jobs If disable is set, the -oferr/-stderr option of the mpiexec command is ignored.	enable	C	C	E
execute interact-mpiexec-std	Authority for using the -of/-std option of the mpiexec command to change the standard output and standard error output destinations of parallel processes for interactive jobs If disable is set, the -of/-std option of the mpiexec command is ignored.	enable	C	C	E
execute interact-mpiexec-stdout	Authority for using the -ofout/-stdout option of the mpiexec command to change the standard output destination of parallel processes for interactive jobs If disable is set, the -ofout/-stdout option of the mpiexec command is ignored.	enable	C	C	E
execute interact-mpiexec-stderr	Authority for using the -oferr/-stderr option of the mpiexec command to change the standard error output destination of parallel processes for interactive jobs If disable is set, the -oferr/-stderr of the mpiexec command is ignored.	enable	C	C	E
execute command-api	Authority for using the command API	disable	E	x	x

Definition name	Description	Default value	Specifiable layer		
			CL	RU	RG
execute pjsub-P-exec-policy [PG]	Authority for executing the -P exec-policy option of the pjsub command.	enable	C	C	E
execute pjsub-P-vn-policy [PG]	Authority for executing the -P vn-policy option of the pjsub command.	enable	C	C	E
execute pjsub-torus [FX]	Authority for submitting jobs with torus mode (":torus") specified (*10)	enable	C	C	E
execute pjsub-mesh [FX]	Authority for submitting jobs with mesh mode (":mesh") specified (*10)	enable	C	C	E
execute pjsub-noncont [FX]	Authority for submitting jobs with non-contiguous mode (":noncont") specified (*10)	enable	C	C	E
execute pjdel-enforce	Authority for using the option for canceling prologue and epilogue scripts in the pjdel command	disable (*21)	C	C	E
execute pjdel-no-history	Authority to suppress output of job information to job history information when deleting jobs.	disable	C	C	E
execute pjhold-enforce	Authority for using the option for canceling prologue and epilogue scripts in the pjhold command	disable (*21)	C	C	E
execute pjsub-batch	Authority for submitting batch jobs	enable	C	C	E
execute pjsub-normal	Authority for submitting normal jobs	enable	C	C	E
execute pjsub-node	Authority for submitting node allocated jobs	enable	C	C	E
execute pjsub-vnode	Authority for submitting virtual node allocated jobs	enable	C	C	E
execute pjsub-nostrict [FX]	Authority for submitting jobs with neither strict nor strict-io specified	enable	C	C	E
execute pjsub-strict [FX]	Authority for submitting jobs with strict specified	enable	C	C	E
execute pjsub-strict-io [FX]	Authority for submitting jobs with strict-io specified	disable	C	C	E
execute pjsub-at	Authority for specifying the execution start time	enable	C	C	E
execute pjsub-net-route [FX]	Authority for submitting jobs with the --net-route option specified in the pjsub command	disable (*21)	C	C	E
permit pjsub	Permission to be able to specify groups with pjsub -g (executable group). This includes permission for the current group without specifying the -g option.	allow own	C	C	E
permit pjstat	Permission to be able to display target jobs with pjstat (*11)	allow own	C	C	E
permit pjdel	Permission to be able to execute target jobs with pjdel (*11)	allow own	C	C	E

Definition name	Description	Default value	Specifiable layer		
			CL	RU	RG
permit pjhold	Permission to be able to execute target jobs with pjhold (*11)	allow own	C	C	E
permit pjrls	Permission to be able to execute target jobs with pjrls (*11)	allow own	C	C	E
permit pjwait	Permission to be able to execute target jobs with pjwait (*11)	allow own	C	C	E
permit pjalter	Permission to be able to execute target jobs with pjalter (*11)	allow own	C	C	E
permit pjsig	Permission to be able to execute target jobs with pjsig (*11)	allow own	C	C	E
permit pjacl	Permission to be able to display target users/ groups with pjacl	allow own	C	C	E
permit pmerls	Permission to be able to execute target jobs with pmerls (*11)	deny all	C	C	E
permit pmalter	Permission to be able to execute target jobs with pmalter (*11)	deny all	C	C	E
permit pjshowrsc	Permission to display job information by using pjshowrs, or the users and groups denied this permission	allow all	C	C	E
select custom-*CustomResourceName*	Specifiable type of custom resource (defined in *CustomResourceName*)	none	C	C	E

Table F.2 Defined items available in the USER definitions (2)

Definition name	Description	Default value			Specifiable layer		
		Lower limit	Upper limit	Default value	CL	RU	RG
joblimit subjobnum	Limit on the number of sub jobs for each bulk job	none	unlimited	none	C	C	E
joblimit priv-pri	Priority among users for each job within a resource unit. It is valid when job_prio of the job selection policy is set.	none	255	127	C	C	E
joblimit elapse	Elapsed time limit for each job (batch jobs)	1	24:00:00	01:00:00	C	C	E
joblimit adaptive-elapsed-time-min [FX]	Minimum elapsed time limit value for a job (batch job)	1	24:00:00	01:00:00	C	C	E
joblimit adaptive-elapsed-time-max [FX]	Maximum elapsed time limit value for a job (batch job)	2	48:00:00	2:00:00	C	C	E
joblimit node	Node number limit for each job (batch jobs)	1	2147483647	2 (*14)	C	C	E
joblimit node-mem	Memory usage limit for each node (batch jobs)	1	unlimited	unlimited	C	C	E
joblimit proc-core	Core file size limit for each process (batch jobs)	none	unlimited	0	C	C	E

Definition name	Description	Default value			Specifiable layer		
		Lower limit	Upper limit	Default value	CL	RU	RG
joblimit proc-cpu	CPU time limit for each process (batch jobs)	none	unlimited	unlimited	C	C	E
joblimit proc-crproc	Limit on the number of user processes created for each process (batch jobs)	none	512	512	C	C	E
joblimit proc-data	Data segment size limit for each process (batch jobs) (*7)	none	unlimited	unlimited	C	C	E
joblimit proc-lockm	Lock memory size limit for each process (batch jobs) (*7)	none	unlimited	unlimited	C	C	E
joblimit proc-msgq	Message queue size limit for each process (batch jobs)	none	unlimited	unlimited	C	C	E
joblimit proc-openfd	Limit on the number of file descriptors for each process (batch jobs)	none	unlimited	1024	C	C	E
joblimit proc-psig	Limit on the number of pending signals for each process (batch jobs)	none	unlimited	unlimited	C	C	E
joblimit proc-filesz	File size limit for each process (batch jobs)	none	unlimited	unlimited	C	C	E
joblimit proc-stack	Stack size limit for each process (batch jobs) (*7)(*8)	none	unlimited	unlimited	C	C	E
joblimit proc-vmem	Virtual memory limit for each process (batch jobs) (*7)(*8)	none	unlimited	unlimited	C	C	E
joblimit vnode-core	Number of CPU cores per virtual node, for virtual node allocated jobs (batch jobs)	1	2147483647	1	C	C	E
joblimit vnode-mem	Limit on memory usage per virtual node for virtual node allocated jobs (limit on memory usage based on the value obtained by multiplying the number of CPU cores per virtual node by memory usage per CPU core) (batch jobs) (*8)(*9)	1	unlimited	unlimited	C	C	E
joblimit vnode	Virtual node number limit for each job (batch jobs)	1	2147483647	1 (*14)	C	C	E
joblimit total-cores	Limit on the total number of CPU cores used per job (batch jobs)	1	unlimited	none	C	C	E
joblimit node-elapse	Limit on the node time spent (Limit based on the value obtained by multiplying the number of request nodes for a batch job by the elapsed time limit value)	1	unlimited	none	C	C	E
joblimit adaptive-node-elapse-min [FX]	Limit on the node time spent (Limit based on the value obtained by multiplying the number of requested nodes for a batch job by the minimum elapsed time limit value)	1	unlimited	none	C	C	E

Definition name	Description	Default value			Specifiable layer		
		Lower limit	Upper limit	Default value	CL	RU	RG
joblimit adaptive-node-elapse-max [FX]	Limit on the node time spent (Limit based on the value obtained by multiplying the number of requested nodes for a batch job by the maximum elapsed time limit value)	2	unlimited	none	C	C	E
joblimit total-cores-elapse	Limit on the CPU core time spent (Limit based on the value obtained by multiplying the number of CPU cores used for a batch job by the elapsed time limit value)	1	unlimited	none	C	C	E
joblimit custom-*CustomResourceName*	Limit on the number of custom resources (defined in *CustomResourceName*) used (*16)	0	unlimited	0	C	C	E
joblimit custom-total-*CustomResourceName*	Limit on the number of custom resources (defined in *CustomResourceName*) used (*17)	0	unlimited	0	C	C	E
joblimit custom-node-*CustomResourceName*	Limit on the number of custom resources (defined in *CustomResourceName*) used (*18)	0	unlimited	0	C	C	E
joblimit custom-vnode-*CustomResourceName*	Limit on the number of custom resources (defined in *CustomResourceName*) used (*19)	0	unlimited	0	C	C	E
joblimit interact-elapse	Elapsed time limit for each job (interactive jobs)	1	24:00:00	01:00:00	C	C	E
joblimit interact-adaptive-elapsed-time-min [FX]	Minimum elapsed time limit value for a job (interactive job)	1	24:00:00	01:00:00	C	C	E
joblimit interact-adaptive-elapsed-time-max [FX]	Maximum elapsed time limit value for a job (interactive job)	2	48:00:00	2:00:00	C	C	E
joblimit interact-node	Node number limit for each job (interactive jobs)	1	2147483647	1 (*14)	C	C	E
joblimit interact-node-mem	Memory usage limit for each node (interactive jobs)	1	unlimited	unlimited	C	C	E
joblimit interact-proc-core	Core file size limit for each process (interactive jobs)	none	unlimited	0	C	C	E
joblimit interact-proc-cpu	CPU time limit for each process (interactive jobs)	none	unlimited	unlimited	C	C	E
joblimit interact-proc-crproc	Limit on the number of user processes created for each process (interactive jobs)	none	512	512	C	C	E
joblimit interact-proc-data	Data segment size limit for each process (interactive jobs)	none	unlimited	unlimited	C	C	E
joblimit interact-proc-lockm	Lock memory size limit for each process (interactive jobs)	none	unlimited	unlimited	C	C	E
joblimit interact-proc-msgq	Message queue size limit for each process (interactive jobs)	none	unlimited	unlimited	C	C	E

Definition name	Description	Default value			Specifiable layer		
		Lower limit	Upper limit	Default value	CL	RU	RG
joblimit interact-proc-openfd	Limit on the number of file descriptors for each process (interactive jobs)	none	unlimited	1024	C	C	E
joblimit interact-proc-psig	Limit on the number of pending signals for each process (interactive jobs)	none	unlimited	unlimited	C	C	E
joblimit interact-proc-filesz	File size limit for each process (interactive jobs)	none	unlimited	unlimited	C	C	E
joblimit interact-proc-stack	Stack size limit for each process (interactive jobs)	none	unlimited	unlimited	C	C	E
joblimit interact-proc-vmem	Virtual memory limit for each process (interactive jobs)	none	unlimited	unlimited	C	C	E
joblimit interact-vnode-core	Number of CPU cores per virtual node, for virtual node allocated jobs (interactive jobs)	1	2147483647	1	C	C	E
joblimit interact-vnode-mem	Limit on memory usage per virtual node for virtual node allocated jobs (limit on memory usage based on the value obtained by multiplying the number of CPU cores per virtual node by memory usage per CPU core) (interactive jobs) (*8)(*9)	1	unlimited	unlimited	C	C	E
joblimit interact-vnode	Virtual node number limit for each job (interactive jobs)	1	2147483647	1 (*14)	C	C	E
joblimit interact-total-cores	Limit on the total number of CPU cores used per job (interactive jobs)	1	unlimited	none	C	C	E
joblimit interact-node-elapse	Limit on the node time spent (Limit based on the value obtained by multiplying the number of request nodes for an interactive job by the elapsed time limit value) (interactive jobs)	1	unlimited	none	C	C	E
joblimit interact-adaptive-node-elapse-min [FX]	Limit on the node time spent (Limit based on the value obtained by multiplying the number of requested nodes for an interactive job by the minimum elapsed time limit value)	1	unlimited	none	C	C	E
joblimit interact-adaptive-node-elapse-max [FX]	Limit on the node time spent (Limit based on the value obtained by multiplying the number of requested nodes for an interactive job by the maximum elapsed time limit value)	2	unlimited	none	C	C	E
joblimit interact-total-cores-elapse	Limit on the CPU core time spent (Limit based on the value obtained by multiplying the number of CPU cores used for an interactive job by the elapsed time limit value)	1	unlimited	none	C	C	E

Definition name	Description	Default value			Specifiable layer		
		Lower limit	Upper limit	Default value	CL	RU	RG
joblimit interact-custom-*CustomResourceName*	Limit on the number of custom resources (defined in *CustomResourceName*) used (by interactive jobs) (*16)	0	unlimited	0	C	C	E
joblimit interact-custom-total-*CustomResourceName*	Limit on the number of custom resources (defined in *CustomResourceName*) used (by interactive jobs) (*17)	0	unlimited	0	C	C	E
joblimit interact-custom-node-*CustomResourceName*	Limit on the number of custom resources (defined in *CustomResourceName*) used (by interactive jobs) (*18)	0	unlimited	0	C	C	E
joblimit interact-custom-vnode-*CustomResourceName*	Limit on the number of custom resources (defined in *CustomResourceName*) used (by interactive jobs) (*19)	0	unlimited	0	C	C	E

(*1)

These items cannot have a group specification (user=<def>:gname, user=uname:gname) for a definition target.

(*2)

The concurrent execution limit of bulk sub jobs is the total value for the sub jobs in all the running target bulk jobs.

(*3)

This is limited to the total excluding virtual node allocated jobs.

(*4)

This is limited to the total of only virtual node allocated jobs in FX server.

(*5)

The fair share-related definition item is always used when subtraction of the fair share value is valid in the job operation configuration file (papjm.conf or pmpjm.conf), regardless of the job selection policy settings.

(*6)

For details, see "3.4.4.3 Priority control of allocated nodes [PG] " .

(*7)

If a process in a job uses an amount of memory exceeding this upper limit, the system may forcibly terminate the process. Do not set these values too small.

(*8)

Suppose that a limit value is specified for memory usage per CPU core when a job is submitted. Then, the limit value for memory usage per virtual node is assumed to be the specified value multiplied by the number of CPU cores in the virtual node.

(*9)

If the nodes are allowed to be shared with other jobs, set values other than "unlimited" for the limits on memory usage per virtual node (joblimit vnode-mem and joblimit interact-vnode-mem). If you specify "unlimited" for the limit value on the memory usage for a job and when a job whose limit value on the memory usage is other than "unlimited" exists on the same node, there is a possibility that the job cannot use the amount of memory of the upper limit.

(*10)

Suppose that you disable all of the following definition items that define the authority for submitting jobs with a node allocation method specified: execute pjsub-torus (torus mode), execute pjsub-mesh (mesh mode), and execute pjsub-noncont (non-contiguous mode). You are now unable to specify a node allocation method when submitting a node allocated job. In this case, nodes are allocated in the mode set with define allocation-mode.

(*11)

The permission for target jobs is specified together with the users or groups executing the jobs.

(*12)

If the item UseCoreLimit in the papjm.conf file is set to all (see "3.4.1.2 Default value settings for resource units"), do not specify a limit on the number of nodes used concurrently.

(*13)

In the item UseCoreLimit in the papjm.conf file, you can specify whether all jobs or only the jobs to which virtual nodes are allocated are subject to the limit on the number of CPU cores used concurrently. If all jobs are targeted, the number of CPU cores used by a node allocated job is the number of mounted CPUs in one node multiplied by the requested number of nodes.

(*14)

The default values of joblimit node and joblimit interact-node are valid only when define alloc-granularity is node. On the other hand, the default values of joblimit vnode and joblimit interact-vnode are valid only when define alloc-granularity is vnode. The pjsub command allows only the value 1 as the number of virtual nodes for the FX server. Therefore, unless 1 is specified for the upper limit value, lower limit value, or standard value of joblimit vnode, the specified value does not have meaning.

(*15)

This setting is valid for the PRIMERGY servers with valid Intel(R) Hyper-Threading Technology and job processes other than those of "C/C++/Fortran programs created in Development Studio." Set a logical CPU usage range with the FLIB_BINDSMT environment variable that is set at runtime, for "C/C++/Fortran programs created in Development Studio." For details, see the manuals provided with Development Studio.

(*16)

Use the items joblimit custom-*CustomResourceName* and joblimit interact-custom-*CustomResourceName* with custom resources per resource unit or resource group.

(*17)

Use the items joblimit custom-total-*CustomResourceName* and joblimit interact-custom-total-*CustomResourceName* with custom resources per node. Set a limited number of uses per job as a setting value.

(*18)

Use the items joblimit custom-node-*CustomResourceName* and joblimit interact-custom-node-*CustomResourceName* with custom resources per node. Set a limited number of uses per node within one job as a setting value.

(*19)

Use the items joblimit custom-vnode-*CustomResourceName* and joblimit interact-custom-vnode-*CustomResourceName* with custom resources per node. Set a limited number of uses per virtual node within one job as a setting value.

(*20)

Outputs in units of mpiexec command or rank according to the setting value of "define mpiexec-stdouterr-unit". The following metacharacters can be specified.

Metacharacter	Description
%j	Job ID
%J	Subjob ID
%b	Bulk number For other than bulk job, this is replaced with empty.
%s	Step number For other than step job, this is replaced with empty.
%n	Job name

Metacharacter	Description
%o	The standard output file name for the job For an interactive job, the file name is "./%n.%J.out".
%e	The standard error output file name for the job For an interactive job, the file name is "./%n.%J.err".
%m	Number of times the mpiexec command runs in the job
%r	Rank number and spawn number For static process: rank number For dynamic process: rank number@spawn number If the destination is for each mpiexec command, this is replaced with empty.
%R	Rank number If the destination is for each mpiexec command, this is replaced with empty.
%S	spawn number For static process: 0 For dynamic process: spawn number If the destination is for each mpiexec command, this is replaced with empty.

For %r, %R, and %S, the following formats can also be used.

Format	Description
%0*N*r %0*N*R %0*N*S	If the display string of the rank or spawn number is less than the minimum field width N, it is padded with 0.
%/*N*r %/*N*R %/*N*S	Rounds a rank or spawn number down to the nearest *N*.
%0*M*/Nr %0*M*/*N*R %0*M*/*N*S	Rounds a rank or spawn number down to the nearest *N*. If the display string of the rounded value is less than the minimum field width *M*, it is padded with 0.

(*21)

"disable" is the default value when an end-user executes the command. "enable" is the default value when the administrator executes the command.

Table F.3 Defined items available in the GROUP definitions

Definition name	Description	Default value	Specifiable layer		
			CL	RU	RG
limit ru-accept	Concurrent job acceptance limit within a resource unit	unlimited	C	E	x
limit ru-accept-allsubjob	Concurrent sub job acceptance limit within a resource unit A normal job of batch job is counted as one sub job.	unlimited	C	E	x
limit ru-accept-bulksubjob	Concurrent bulk sub job acceptance limit within a resource unit	unlimited	C	E	x
limit ru-accept-stepsubjob	Concurrent step sub job acceptance limit within a resource unit	unlimited	C	E	x
limit ru-run-job	Concurrent job execution limit within a resource unit	unlimited	C	E	x

Definition name	Description	Default value	Specifiable layer		
			CL	RU	RG
limit ru-run-bulksubjob	Concurrent bulk sub job execution limit within a resource unit	unlimited	C	E	x
limit ru-use-node	Concurrent node usage limit within a resource unit (*1)(*4)	unlimited	C	E	x
limit ru-use-core	Concurrent CPU core usage limit within a resource unit (*2)(*5)	unlimited	C	E	x
limit ru-custom-*CustomResourceName*	Limit on the number of custom resources (defined in *CustomResourceName*) that can be used concurrently within a resource unit	unlimited	C	E	x
limit ru-interact-accept	Concurrent job acceptance limit (interactive jobs) within a resource unit	unlimited	C	E	x
limit ru-interact-run-job	Concurrent job execution limit (interactive jobs) within a resource unit	unlimited	C	E	x
limit ru-interact-use-node	Concurrent node usage limit (interactive jobs) within a resource unit (*1)(*4)	unlimited	C	E	x
limit ru-interact-use-core	Concurrent CPU core usage limit (interactive jobs) within a resource unit (*2)(*5)	unlimited	C	E	x
limit ru-interact-custom-*CustomResourceName*	Limit on the number of custom resources (defined in *CustomResourceName*) that can be used concurrently (by interactive jobs) within a resource unit	unlimited	C	E	x
limit rg-accept	Concurrent job acceptance limit within a resource group	unlimited	C	C	E
limit rg-accept-allsubjob	Concurrent sub job acceptance limit within a resource group A normal job of batch job is counted as one sub job.	unlimited	C	C	E
limit rg-accept-bulksubjob	Concurrent bulk sub job acceptance limit within a resource group	unlimited	C	C	E
limit rg-accept-stepsubjob	Concurrent step sub job acceptance limit within a resource group	unlimited	C	C	E
limit rg-run-job	Concurrent job execution limit within a resource group	unlimited	C	C	E
limit rg-run-bulksubjob	Concurrent bulk sub job execution limit within a resource group	unlimited	C	C	E
limit rg-use-node	Concurrent node usage limit within a resource group (*1)(*4)	unlimited	C	C	E
limit rg-use-core	Concurrent CPU core usage limit within a resource group (*2)(*5)	unlimited	C	C	E
limit rg-custom-*CustomResourceName*	Limit on the number of custom resources (defined in *CustomResourceName*) that can be used concurrently within a resource group	unlimited	C	C	E
limit rg-interact-accept	Concurrent job acceptance limit (interactive jobs) within a resource group	unlimited	C	C	E

Definition name	Description	Default value	Specifiable layer		
			CL	RU	RG
limit rg-interact-run-job	Concurrent job execution limit (interactive jobs) within a resource group	unlimited	C	C	E
limit rg-interact-use-node	Concurrent node usage limit (interactive jobs) within a resource group (*1)(*4)	unlimited	C	C	E
limit rg-interact-use-core	Concurrent CPU core usage limit (interactive jobs) within a resource group (*2)(*5)	unlimited	C	C	E
limit rg-interact-custom-*CustomResourceName*	Limit on the number of custom resources (defined in *CustomResourceName*) that can be used concurrently (by interactive jobs) within a resource group	unlimited	C	C	E
define pri-g	Group priority within a resource unit. It is valid when group_prio of the job selection policy is set. (*3)	127	C	C	E
define fshare-init-g	Default group fair share value within a resource unit. It is valid when group_fairshare of the job selection policy is set. (*3) For details on how to specify a fair share set to define initial fair share values, see "3.4.4.4 How to define a fair share set."	0	C	E	x
define fshare-recovery-g	Fair share recovery factor for group within a resource unit. It is valid when group_fairshare of the job selection policy is set. (*3) For details on how to specify a fair share set to define initial fair share values, see "3.4.4.4 How to define a fair share set."	100	C	E	x

(*1)

This is limited to the total excluding node allocated jobs.

(*2)

This is limited to the total of only virtual node allocated jobs in FX server.

(*3)

The fair share-related definition item is always used when subtraction of the fair share value at job execution start is valid in the job operation configuration file (papjm.conf or pmpjm.conf), regardless of the job selection policy settings.

(*4)

If the item UseCoreLimit in the papjm.conf file is set to all (see "3.4.1.2 Default value settings for resource units"), do not specify a limit on the number of nodes used concurrently.

(*5)

In the item UseCoreLimit in the papjm.conf file, you can specify whether all jobs or only the jobs to which virtual nodes are allocated are subject to the limit on the number of CPU cores used concurrently. If all jobs are targeted, the number of CPU cores used by a node allocated job is the number of mounted CPUs in one node multiplied by the requested number of nodes.

Table F.4 Defined items available in the ALL definitions

Definition name	Description	Default value	Specifiable layer		
			CL	RU	RG
limit ru-accept	Concurrent job acceptance limit within a resource unit	unlimited	C	E	x
limit ru-accept-allsubjob	Concurrent sub job acceptance limit within a resource unit A normal job of batch job is counted as one sub job.	unlimited	C	E	x
limit ru-accept-bulksubjob	Concurrent bulk sub job acceptance limit within a resource unit	unlimited	C	E	x
limit ru-accept-stepsubjob	Concurrent step sub job acceptance limit within a resource unit	unlimited	C	E	x
limit ru-run-job	Concurrent job execution limit within a resource unit	unlimited	C	E	x
limit ru-run-bulksubjob	Concurrent bulk sub job execution limit within a resource unit	unlimited	C	E	x
limit ru-use-node	Concurrent node usage limit within a resource unit (*1)(*3)	unlimited	C	E	x
limit ru-use-core	Concurrent CPU core usage limit within a resource unit (*2)(*4)	unlimited	C	E	x
limit ru-custom-*CustomResourceName*	Limit on the number of custom resources (defined in *CustomResourceName*) that can be used concurrently within a resource unit	unlimited	C	E	x
limit ru-interact-accept	Concurrent job acceptance limit (interactive jobs) within a resource unit	unlimited	C	E	x
limit ru-interact-run-job	Concurrent job execution limit (interactive jobs) within a resource unit	unlimited	C	E	x
limit ru-interact-use-node	Concurrent node usage limit (interactive jobs) within a resource unit (*1)(*3)	unlimited	C	E	x
limit ru-interact-use-core	Concurrent CPU core usage limit (interactive jobs) within a resource unit (*2)(*4)	unlimited	C	E	x
limit ru-interact-custom-*CustomResourceName*	Limit on the number of custom resources (defined in *CustomResourceName*) that can be used concurrently (by interactive jobs) within a resource unit	unlimited	C	E	x
limit rg-accept	Concurrent job acceptance limit within a resource group	unlimited	C	C	E
limit rg-accept-allsubjob	Concurrent sub job acceptance limit within a resource group A normal job of batch job is counted as one sub job.	unlimited	C	C	E
limit rg-accept-bulksubjob	Concurrent bulk sub job acceptance limit within a resource group	unlimited	C	C	E
limit rg-accept-stepsubjob	Concurrent step sub job acceptance limit within a resource group	unlimited	C	C	E

Definition name	Description	Default value	Specifiable layer		
			CL	RU	RG
limit rg-run-job	Concurrent job execution limit within a resource group	unlimited	C	C	E
limit rg-run-bulksubjob	Concurrent bulk sub job execution limit within a resource group	unlimited	C	C	E
limit rg-use-node	Concurrent node usage limit within a resource group (*1)(*3)	unlimited	C	C	E
limit rg-use-core	Concurrent CPU core usage limit within a resource group (*2)(*4)	unlimited	C	C	E
limit rg-custom-*CustomResourceName*	Limit on the number of custom resources (defined in *CustomResourceName*) that can be used concurrently within a resource group	unlimited	C	C	E
limit rg-interact-accept	Concurrent job acceptance limit (interactive jobs) within a resource group	unlimited	C	C	E
limit rg-interact-run-job	Concurrent job execution limit (interactive jobs) within a resource group	unlimited	C	C	E
limit rg-interact-use-node	Concurrent node usage limit (interactive jobs) within a resource group (*1)(*3)	unlimited	C	C	E
limit rg-interact-use-core	Concurrent CPU core usage limit (interactive jobs) within a resource group (*2)(*4)	unlimited	C	C	E
limit rg-interact-custom-*CustomResourceName*	Limit on the number of custom resources (defined in *CustomResourceName*) that can be used concurrently (by interactive jobs) within a resource group	unlimited	C	C	E

(*1)

This is limited to the total excluding virtual node allocated jobs.

(*2)

This is limited to the total of only virtual node allocated jobs in FX server.

(*3)

If the item UseCoreLimit in the papjm.conf file is set to all (see "3.4.1.2 Default value settings for resource units"), do not specify a limit on the number of nodes used concurrently.

(*4)

In the item UseCoreLimit in the papjm.conf file, you can specify whether all jobs or only the jobs to which virtual nodes are allocated are subject to the limit on the number of CPU cores used concurrently. If all jobs are targeted, the number of CPU cores used by a node allocated job is the number of mounted CPUs in one node multiplied by the requested number of nodes.