FUJITSU

Fujitsu Software
Technical Computing Suite V4.0L20

Development Studio
Profiler User's Guide

J2UL-2483-02ENZ0(14)
September 2025

Preface

Purpose of This Manual

This guide describes the features and the usage of the Profiler function for the system that has the Fujitsu CPU A64FX installed.

Intended Readers

This document is intended for those who tune applications with the Profiler. It assumes the following knowledge:

- Knowledge of program development work on a Linux(R) operating system and associated basic command operations on a Linux operating system
- Knowledge of Microsoft(R) Excel(R)

Organization of This Manual

This manual consists of the following sections.

Chapter 1 Overview of the Profiler

This chapter provides an overview of the Profiler.

Chapter 2 Instant Performance Profiler

Explains the Instant Performance Profiler.

Chapter 3 Advanced Performance Profiler

Explains the Advanced Performance Profiler.

Chapter 4 CPU Performance Analysis Report

Explains the CPU Performance Analysis Report.

Chapter 5 Notes

This chapter provides notes on using the Profiler.

Appendix A Troubleshooting

This appendix describes troubleshooting for the Profiler.

Appendix B List of Messages

This appendix describes typical messages output by the Profiler.

Related Manuals

This book relates to the following manuals. If necessary, refer also to these manuals.

- "Fortran Language Reference"
- "Fortran User's Guide"
- "Fortran User's Guide Additional Volume COARRAY"
- "Fortran Compiler Messages"
- "C User's Guide"
- "C++ User's Guide"
- "C/C++ Compiler Optimization Messages"
- "Fortran/C/C++ Runtime Messages"
- "MPI User's Guide"

Also, refer to the manuals provided with the following related software:

- "Job Operation Software"

- "FEFS"

Syntax Description Symbols

A syntax description symbol is a symbol that has specific meaning in syntax. The following symbols are used in this guide.

Symbol name	Symbol	Description
Selection symbols	{ }	Indicates that only one of the enclosed items can be selected
	|	Indicates that it is used as a delimiter in a list of items
Optional symbol	[ ]	Indicates that the enclosed item can be omitted The { } (braces selection symbols) and [ ] (brackets optional symbol) have the same meaning.
Repeat symbol	...	Indicates that the item just before this can be specified repeatedly

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.
- OpenMP is a trademark of OpenMP Architecture Review Board.
- Microsoft, Windows, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
- Mac is registered trademarks of Apple Inc.
- Screenshots are used in accordance with Microsoft Corporation guidelines.
- Arm is a registered trademark of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.
- All other trademarks are the property of their respective owners.
- The trademark notice symbol (TM,(R)) is not necessarily added in the system name and the product name, etc. published in this material.

Date of Publication and Version

Version	Manual code
September 2025, Version 2.14	J2UL-2483-02ENZ0(14)
March 2024, Version 2.13	J2UL-2483-02ENZ0(13)
March 2023, Version 2.12	J2UL-2483-02ENZ0(12)
September 2022, Version 2.11	J2UL-2483-02ENZ0(11)
March 2022, Version 2.10	J2UL-2483-02ENZ0(10)
November 2021, Version 2.9	J2UL-2483-02ENZ0(09)
July 2021, Version 2.8	J2UL-2483-02ENZ0(08)
March 2021, Version 2.7	J2UL-2483-02ENZ0(07)
January 2021, Version 2.6	J2UL-2483-02ENZ0(06)
December 2020, Version 2.5	J2UL-2483-02ENZ0(05)
November 2020, Version 2.4	J2UL-2483-02ENZ0(04)
September 2020, Version 2.3	J2UL-2483-02ENZ0(03)
June 2020, Version 2.2	J2UL-2483-02ENZ0(02)

Version	Manual code
April 2020, Version 2.1	J2UL-2483-02ENZ0(01)
March 2020, 2nd Version	J2UL-2483-02ENZ0(00)
January 2020, 1st Version	J2UL-2483-01ENZ0(00)

Copyright

Copyright FUJITSU LIMITED 2020-2025

Update History

Changes	Location	Version
Updated the Arm trademark.	Preface	Version 2.14
Changed the supported combination of software.	1.1	
Changed the note.	3.1.1.1 3.1.1.2	Version 2.13
Removed Windows 8.1 from Basic Software.	1.1	Version 2.12
Added a note.	2.1.2 2.1.3	
Added "mpiexec command".	5.1.5	
Added a note.	2.1.3.1.2	Version 2.11
Change the explanation about "event".	3.2.3.2.4	
Added the explanation.	5.2.1	
Added the explanation.	5.2.9	
Added the explanation.	A.1.5	
Removed the explanation about Clang Mode.	2.1.3.1.2	Version 2.10
Changed the note.	4.2.2.3.2	
Improved the explanation.	5.2.7 5.2.9	
Added the explanation.	A.1.1	
Added the explanation.	A.1.5	
Change the output of -Ibalance option.	2.2.2.4.1 2.2.2.4.2	Version 2.9
Change the items and descriptions in the table of output items.	4.2.2.4.1 4.2.2.4.2 4.2.2.4.3 4.2.2.5.1 4.2.2.5.2	
Added the troubleshooting for CPU Performance Analysis Report.	A.3	
Added explanation about CPU Frequency.	3.2.2.1	Version 2.8
Added "Exit Status".	5.1.7	
Added "LD_PRELOAD".	5.1.8	
Added the following explanation: - Compiler option -g	5.2.1	

Changes	Location	Version
- Compiler option -Nline, -ffj-line		
Added "Sampling Number".	5.2.13	
Added "Signal Interrupt by Sampling".	5.2.14	
Added and Improved messages.	Appendix B	
Improved the explanation.	-	
Added execution environment in outputting profile result and obtaining the CPU Performance Analysis Report file.	1.2 4.1.6	Version 2.7
Added notes.	2.1.1.1	
Added the -M option of fipp command.	2.1.4	
Added a note.	2.2.2.6	
Added notes.	3.1.1.1	
Added notes about the -M option.	5.2.12	
Added a message.	B.1	
Change of target for counting up the cost of inlined function in Instant Performance Profiler.	5.2.7	Version 2.6
Change CPU Binding.	5.3.5	Version 2.5
Change Output format of the environment information for measuring profiling data.	3.2.2.1	Version 2.4
Change graph of CPU Performance Analysis Report.	4.2.1 4.2.2.2.1 4.2.2.2.2 4.2.2.3.1 4.2.2.3.2 4.2.2.5.1 4.2.2.5.2	
Added notes about CPU Performance Analysis Report.	4.2.2.2.1 4.2.2.2.2 4.2.2.4.1 4.2.2.4.2 4.2.2.4.3	
Added points about Compiler Options.	2.1.3.1	Version 2.3
Improved explanation of "Output items of the CPU performance characteristics".	2.2.2.3	
Added notes about CPU Performance Analysis Report.	4.2.2.2.1 4.2.2.2.2 4.2.2.4.1 4.2.2.4.2 4.2.2.4.3 4.2.2.7.1 4.2.2.7.2	
Added notes about COARRAY.	5.1.1	
Added Impact of Compiler Options.	5.1.2	
Added Impact of Using the MPI Profiling Interface.	5.1.5	
Added Mixed Language Programming for MPI Program.	5.1.6	
Added Impact of Compiler Options.	5.2.1	
Change Call Graph Information.	5.2.9	
Added the -u option of fipp command and fipppx command.	2.1.5 5.2.11	Version 2.2

Changes	Location	Version
Added notes about cntfrq.	3.2.3.2.2	
Change pictures of Data Transfer CMGs	4.2.2.10	
Change CPU Binding	5.3.5	
Improved the explanation.	-	
Added notes about fjprof_spawn_dir_name.	2.1.3.2	Version 2.1
Added the -b{max|total} options of fipp command and fipppx command.	2.1.5 2.2.2.1 2.2.2.4 2.2.2.4.1 2.2.2.4.2 2.2.2.4.3	
Added the -tcsv option of fipp command and fipppx command.	2.1.5	
Added "CPU frequency" to environment information for measuring profiling data.	2.2.2.1 2.2.3.2.2	
Added "Ratio of the measured value to the theoretical value for memory throughput" to CPU performance characteristics.	2.2.2.3 2.2.3.2.5	
Added notes about frequency.	2.2.3.2.2 3.2.3.2.2	
Added notes about measurement region specifying routine.	3.1.1.1 3.1.1.2	
Advanced Performance Profiler supported dynamically generated processes. - Added -W{spawn|nospawn} options of fipp command. - Added arguments to the -p option of the fapppx command and fapp command. - Modified profile result (TEXT format) output.	3.1.4 3.2.2.1 3.2.2.2 3.2.2.3	
Changed the notes of the -I option of the fapppx and fapp command.	3.1.5	
Added CPU performance analysis information to profile result (TEXT format) of the Advanced Performance Profiler. - Changed the description of the -I option of fapppx command and fapp command. - Added CPU performance analysis information to profile result (TEXT format).	3.1.5 3.2.2.4	
Advanced Performance Profiler supported profile result (XML format) output. - Added the -txml option of the fapppx command and fapp command. - Changed the description of the -p option of the fapppx command and fapp command. - Added profile result (XML format).	3.1.5 3.2.3 3.2.3.1 3.2.3.2 3.2.3.2.1 3.2.3.2.2 3.2.3.2.3 3.2.3.2.4 3.2.3.2.5	
Changed notes about CPU binding.	4.1.6.1 5.3.5	
Added messages.	B.2 B.3	
Improved the explanation.	-	
Changed macOS required for CPU performance analysis report to Catalina.	1.1	2nd Version

Changes	Location	Version
Clang Mode compile supported. - Added Clang Mode compiler options. - Removed Clang Mode notes. - Added Clang Mode " Names of Generated Procedures for Thread Parallel Programs."	2.1.3.1.2 2.2.2.4.1	
LLVM OpenMP library supported. Removed "LLVM OpenMP" library notes.	-	
Instant Performance Profiler supported dynamically generate the processes. - Added description of info key fjprof_spawn_dir_name. - Added fipp command -W{spawn|nospawn} options. - Modified profile result (TEXT format) Output.	2.1.3.2 2.1.4 2.2.2.2 2.2.2.3 2.2.2.4.1 2.2.2.4.2 2.2.2.4.3 2.2.2.5	
Instant Performance Profiler supported profile result (XML format) output. - Added fipppx command -txml option. - Added profile result (XML format).	2.1.5 2.2.1 2.2.3	
Modified fipppx command -p option.	2.1.5	
Added caveats about @barrier and @barrier-rate.	2.2.2.4.1 2.2.2.4.2 2.2.2.4.3	
Changed to allow the -A option to be specified with the fipp or fapp command.	chapter 2 2.1.4 2.1.5 chapter 3 3.1.4 3.1.5	
Added fapp command -I{cputime|nocputime} option.	3.1.4 3.2.2.2	
Removed notes about the -Hmethod=fast option of the fapp command.	3.1.4	
Added precaution for MPI routines belonging to Persistent Collective Communication Request Routine.	3.2.2.3	
Modified the formulas for "MPIX_REDUCE_SCATTER_BLOCK_INIT" and "MPIX_REDUCE_SCATTER_INIT."	3.2.2.3	
Added "Single Report" to the CPU Performance Analysis Report.	Chapter 4: All	
Modified content of CPU Performance Analysis Report.	Chapter 4: All	
Moved notes on "Precision of Input Data to the CPU Performance Analysis Report" to "Error and Warning Messages Output by the CPU Performance Analysis Report."	4.1.6.1 5.4	
Moved notes on "Targets of Measurement of Thread Parallelization Information" to "Notes Common to Profilers."	5.1.4 5.3	
Added notes on Cost Information.	5.2.7	
Modified notes on Source Code Information.	5.2.8	
Added notes on Call Graph Information.	5.2.9	
Added notes.	5.2.10 5.3.6	
Removed notes on NUMA nodes.	5.4	

Changes	Location	Version
Added Troubleshooting.	A.1.1 A.1.2	
Added Messages.	B.1 B.3	
Changed the look according to product upgrades.	-	

All rights reserved.
The information in this manual is subject to change without notice.

Contents

Chapter 1 Overview of the Profiler 1
1.1 Configuration of the Profiler 1
1.2 Flow of Tuning 1

Chapter 2 Instant Performance Profiler 3
2.1 Procedure for Using the Instant Performance Profiler 3
2.1.1 Adding a Measurement Region Specifying Routine 4
2.1.1.1 fipp_start / fipp_stop Subroutines (Fortran) 4
2.1.1.2 fipp_start function / fipp_stop Function (C language and C++) 6
2.1.2 Specifying Environment Variables 7
2.1.3 Compilation 7
2.1.3.1 Compiler Options 8
2.1.3.1.1 Fortran 8
2.1.3.1.2 C and C++ Languages 8
2.1.3.2 Info Key fjprof_spawn_dir_name 9
2.1.4 Measuring Profile Data 10
2.1.5 Outputting Profile Result 14
2.2 Profile Result 17
2.2.1 Overview of Profile Result 17
2.2.2 Details of Profile Result (TEXT Format) 17
2.2.2.1 Environment Information for Measuring Profiling Data 18
2.2.2.2 Statistics Time Information 19
2.2.2.3 CPU Performance Characteristics 20
2.2.2.4 Cost Information 22
2.2.2.4.1 Procedure Cost Distribution Information 22
2.2.2.4.2 Loop Cost Distribution Information 25
2.2.2.4.3 Line Cost Distribution Information 28
2.2.2.5 Call Graph Information 30
2.2.2.6 Source Code Information 31
2.2.3 Details of Profile Result (XML Format) 31
2.2.3.1 Structure of XML format 31
2.2.3.2 Details of XML format output 32
2.2.3.2.1 Profiling Information <profile> 32
2.2.3.2.2 Environment Information for Measuring Profiling Data <environment> 32
2.2.3.2.3 Performance Information <information> 34
2.2.3.2.4 Statistics Time Information <time> 35
2.2.3.2.5 CPU Performance Characteristics <cpupa> 35
2.2.3.2.6 Cost Information <cost> 37
2.2.3.2.7 Call Graph Information <call> 40

Chapter 3 Advanced Performance Profiler 42
3.1 Procedure for Using the Advanced Performance Profiler 42
3.1.1 Adding a Measurement Region Specifying Routine 43
3.1.1.1 fapp_start / fapp_stop Subroutines (Fortran) 43
3.1.1.2 fapp_start function / fapp_stop Function (C language and C++) 45
3.1.2 Specifying Environment Variables 47
3.1.3 Compilation 47
3.1.4 Measuring Profile Data 48
3.1.5 Outputting Profile Result 50
3.2 Profile Result 52
3.2.1 Overview of Profile Result 52
3.2.2 Detail of Profile Result (TEXT Format) 53
3.2.2.1 Environment Information for Measurement Profiling Data 53
3.2.2.2 Statistical Time Information 54
3.2.2.3 MPI Communication Cost Information 55
3.2.2.4 CPU Performance Analysis Information 62

3.2.3 Detail of Profile Result (XML Format)..... 65
3.2.3.1 Structure of XML format..... 65
3.2.3.2 Details of XML format output..... 66
3.2.3.2.1 Profiling Information <profile>..... 66
3.2.3.2.2 Environment Information for Measuring Profiling Data <environment>..... 66
3.2.3.2.3 Performance Information <information>..... 68
3.2.3.2.4 CPU Performance Analysis Information <cpupa>..... 70
3.2.3.2.5 MPI Communication Cost Information <mpi>..... 70

Chapter 4 CPU Performance Analysis Report..... 73
4.1 Procedure for Using the CPU Performance Analysis Report..... 75
4.1.1 Adding a Measurement Region Specifying Routine..... 75
4.1.2 Specifying Environment Variables..... 75
4.1.3 Compilation..... 75
4.1.4 Measuring Profile Data..... 76
4.1.5 Outputting Profile Result..... 77
4.1.6 Creating a CPU Performance Analysis Report..... 78
4.1.6.1 Error and Warning Messages Output by the CPU Performance Analysis Report..... 81
4.2 CPU Performance Analysis Report Output Result..... 82
4.2.1 Overview of CPU Performance Analysis Report Output Result..... 82
4.2.2 Detail of CPU Performance Analysis Report Output Result..... 84
4.2.2.1 Information..... 84
4.2.2.2 Statistics..... 85
4.2.2.2.1 Statistics (Single Report)..... 85
4.2.2.2.2 Statistics (Brief, Standard, and Detail Report)..... 88
4.2.2.3 Cycle Accounting..... 91
4.2.2.3.1 Cycle Accounting (Brief Report)..... 91
4.2.2.3.2 Cycle Accounting (Standard and Detail Reports)..... 93
4.2.2.4 Busy..... 94
4.2.2.4.1 Busy (Brief Report)..... 95
4.2.2.4.2 Busy (Standard Report)..... 96
4.2.2.4.3 Busy (Detail Report)..... 98
4.2.2.5 Cache..... 99
4.2.2.5.1 Cache (Brief Report)..... 100
4.2.2.5.2 Cache (Standard and Detail Reports)..... 102
4.2.2.6 Instruction..... 104
4.2.2.6.1 Instruction (Brief Report)..... 104
4.2.2.6.2 Instruction (Standard Report)..... 105
4.2.2.6.3 Instruction (Detail Report)..... 106
4.2.2.7 FLOPS..... 107
4.2.2.7.1 FLOPS (Brief Report)..... 108
4.2.2.7.2 FLOPS (Standard and Detail Reports)..... 109
4.2.2.8 Extra..... 110
4.2.2.9 Hardware Prefetch Rate (%) (/Hardware Prefetch)..... 110
4.2.2.10 Data Transfer CMGs..... 111
4.2.2.11 Power Consumption (W)..... 112

Chapter 5 Notes..... 115
5.1 Notes Common to Profilers..... 115
5.1.1 COARRAY..... 115
5.1.2 Impact of Compiler Options..... 115
5.1.3 Node-Sharing Job..... 115
5.1.4 Targets of Measurement of Thread Parallelization Information..... 115
5.1.5 mpiexec command..... 115
5.1.6 Impact of Using the MPI Profiling Interface..... 116
5.1.7 Mixed Language Programming for MPI Program..... 116
5.1.8 Exit Status..... 116
5.1.9 LD_PRELOAD..... 116

5.2 Notes on the Instant Performance Profiler....117
5.2.1 Impact of Compiler Options....117
5.2.2 Prohibition of Catching or Issuing SIGVTALRM Signal....117
5.2.3 Sampling Interval at the Time of Profile Data Measurement....117
5.2.4 Profiler Workspace....118
5.2.5 -pall Option....118
5.2.6 CPU Performance Characteristics....118
5.2.7 Cost Information....118
5.2.8 Source Code Information....119
5.2.9 Call Graph Information....119
5.2.10 Cost Information for Line Number 0....120
5.2.11 -u option....120
5.2.12 -Minlined option....121
5.2.13 Sampling Number....121
5.2.14 Signal Interrupt by Sampling....121
5.3 Notes on the Advanced Performance Profiler....121
5.3.1 MPI Thread Support....121
5.3.2 CPU Performance Analysis Information....121
5.3.3 -Hevent_raw Option....121
5.3.4 Elapsed Time Information for MPI Library....121
5.3.5 CPU Binding....121
5.3.6 Routines that Cannot Measure MPI Communication Cost Information....122
5.4 Notes on the CPU Performance Analysis Report....122
5.4.1 CPU Performance Analysis Report File....122
5.4.2 Dynamically Generated Processes....122

Appendix A Troubleshooting....123
A.1 Instant Performance Profiler....123
A.1.1 Performing profile data measurement results in longer execution time compared with normal execution....123
A.1.2 Memory usage increases when measuring the profile data compared with the normal operation....123
A.1.3 A procedure name that does not exist in the source code (such as a library name) appears....123
A.1.4 Fail to open profile data....123
A.1.5 The symbol __?unknown is output....124
A.2 Advanced Performance Profiler....124
A.2.1 Performing profile data measurement results in longer execution time compared with normal execution....124
A.2.2 Fail to open profile data....124
A.3 CPU Performance Analysis Report....124
A.3.1 Fail to load CSV Format File (File line limit exceeded)....124

Appendix B List of Messages....125
B.1 List of Message (fipp command)....125
B.2 List of Message (fipppx command)....131
B.3 List of Message (fapp command)....133
B.4 List of Message (fapppx command)....136

Chapter 1 Overview of the Profiler

The Profiler measures and outputs performance information useful for tuning to improve the execution performance of a program. Generally, tuning by identifying a region of a program that takes a long execution time (hereinafter referred to as an expensive operation region) can lead to a reduction in execution time.

1.1 Configuration of the Profiler

The Profiler consists of the three functions: "Instant Performance Profiler", "Advanced Performance Profiler", and "CPU Performance Analysis Report".

Instant Performance Profiler

The Instant Performance Profiler is a profiler used to grasp trends in the performance of the entire program without recompilation. Even for a large-scale parallel program, it can measure performance information with low overhead. The Instant Performance Profiler outputs statistical time information, CPU performance characteristics, cost information, call graph information, and source code information. For detail, see "Chapter 2 Instant Performance Profiler".

Note

If a program satisfies specific conditions, the source code may need to be modified or recompiled because the Instant Performance Profiler cannot perform a correct measurement. For detail, see "5.1 Notes Common to Profilers" and "5.2 Notes on the Instant Performance Profiler". In addition, the source code also needs to be modified and recompiled when a measurement region is specified. For details on specifying a measurement region, see "2.1.1 Adding a Measurement Region Specifying Routine".

Advanced Performance Profiler

The Advanced Performance Profiler is a profiler used to grasp detailed performance in a specific region. To use the Advanced Performance Profiler, the source code needs to be modified and recompiled. The Advanced Performance Profiler outputs statistical time information, MPI communication cost information, and CPU performance analysis information. For detail, see "Chapter 3 Advanced Performance Profiler".

CPU Performance Analysis Report

The CPU Performance Analysis Report aggregates CPU performance analysis information measured by the Advanced Performance Profiler and visualizes it in tables and graphs in an easy-to-understand way. For detail, see "Chapter 4 CPU Performance Analysis Report".

Note

To use the CPU Performance Analysis Report, Microsoft Excel and a Basic Software where the software runs are required. The operation of the CPU Performance Analysis Report has been verified in the following combinations.

Basic Software	Microsoft Excel
Microsoft Windows 11 (64bit)	Microsoft Excel for Microsoft 365 MSO 64bit
macOS Catalina	Microsoft Excel

1.2 Flow of Tuning

This section describes the flow of basic tuning with the Profiler.

1. Using the Instant Performance Profiler, identify an expensive operation region and grasp trends in the performance of the entire program.
2. Add a measurement region specifying routine for the Advanced Performance Profiler to the source code for the expensive operation region identified in step 1, and recompile it.

3. Using the Advanced Performance Profiler or CPU Performance Analysis Report, analyze detailed information on the expensive operation region. Which function to use depends on the performance information you want to view.
   - We recommend using the Advanced Performance Profiler to view statistical time information and MPI-related performance information.
   - We recommend using the CPU Performance Analysis Report to view detailed performance information on CPUs.

The following is a list of operations performed with each function.

Operation	Execution Environment	Chapter 2 Instant Performance Profiler	Chapter 3 Advanced Performance Profiler	Chapter 4 CPU Performance Analysis Report
Adding a measurement region specifying routine (Optional in the case of Instant Performance Profiler)	Any machine	2.1.1 Adding a Measurement Region Specifying Routine	3.1.1 Adding a Measurement Region Specifying Routine	
Specifying environment variables	Login node and compute node	2.1.2 Specifying Environment Variables	3.1.2 Specifying Environment Variables	
Compilation	Login node or compute node	2.1.3 Compilation	3.1.3 Compilation	
Measuring profile data	Compute node	2.1.4 Measuring Profile Data	3.1.4 Measuring Profile Data	
Outputting profile result	Login node and compute node	2.1.5 Outputting Profile Result	3.1.5 Outputting Profile Result	
Creating a CPU performance analysis report	Windows or macOS machine (*)	None		4.1.6 Creating a CPU Performance Analysis Report

*: CPU Performance Analysis Report file is stored in login node.

Chapter 2 Instant Performance Profiler

This chapter describes the Instant Performance Profiler.

The Instant Performance Profiler measures and outputs the statistical information of the entire program through sampling analysis. Since the Instant Performance Profiler performs sampling analysis, it cannot perform measurement on a program whose execution time is approximately 1 second or less. The Instant Performance Profiler consists of two commands: the fipp command, which measures profile data, and the fipppx command, which output profile result from measured data. The Instant Performance Profiler can output the following statistical information:

Statistical time information

As statistical time information, program elapsed time, user CPU time, and system CPU time are output.

CPU performance characteristics

As CPU performance characteristics, information related to CPU performance characteristics, such as memory throughput, the number of instructions, and the number of operations, is displayed.

Cost information

As cost information, the number of sampling times during program execution is output as the cost for each procedure, loop, or line.

Call graph information

As call graph information, procedure call traces and the cost for each procedure call trace are output.

Source code information

As source code information, a cost added to each line of the source code is output.

Note

The fipp command can be used both to measure the profile data and to output the profile results. In this User's Guide, we use this command to measure the profile data.

For notes on using the Instant Performance Profiler, see "5.1 Notes Common to Profilers" and "5.2 Notes on the Instant Performance Profiler".

Point

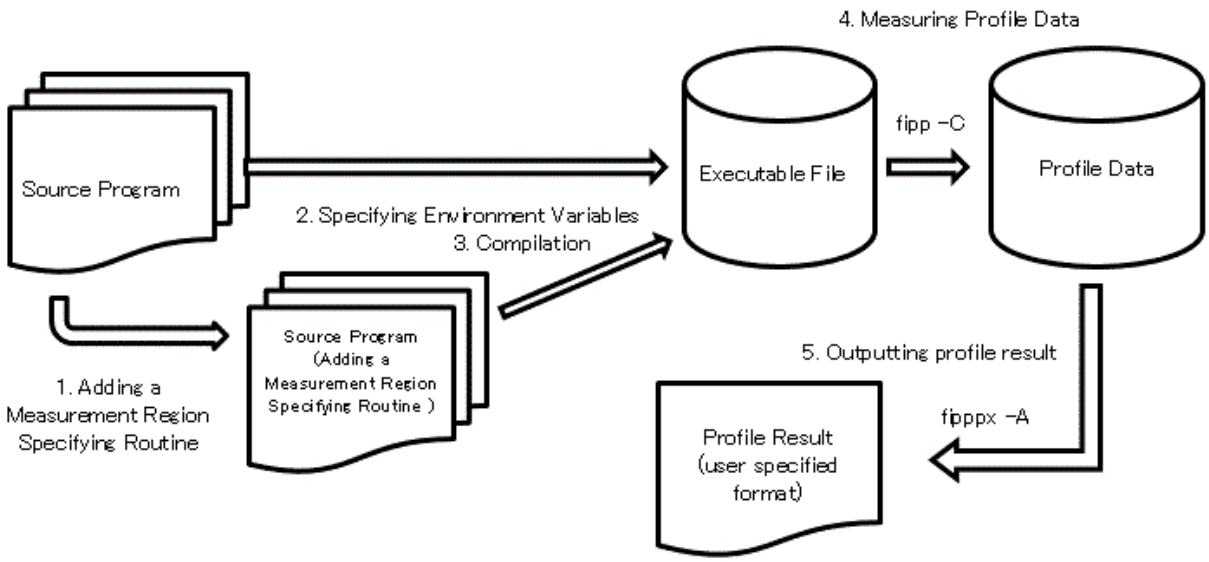
Although the Instant Performance Profiler normally measures and outputs the statistical information of the entire program, the -Sregion option enables it to perform measurement only on a specific region. Use the option when, for example, you want to perform measurement only on the operation processing part by excluding the file input/output part. To use the -Sregion option, however, you need to add a measurement region specifying routine to the source code. For details on the -Sregion option, see "2.1.4 Measuring Profile Data". For details on measurement region specifying routines, see "2.1.1 Adding a Measurement Region Specifying Routine".

In an MPI program, when you use the MPI_COMM_SPAWN routine or MPI_COMM_SPAWN_MULTIPLE routine to dynamically generate the parallel processes, the Job Operation Software allocates the numbers to the generated processes. We call this allocated number as "spawn number" in this User's Guide. For details, see the "MPI User's Guide".

2.1 Procedure for Using the Instant Performance Profiler

This section provides the procedure for using the Instant Performance Profiler.

Figure 2.1 Procedure for using the Instant Performance Profiler



The following describes each operation in detail.

2.1.1 Adding a Measurement Region Specifying Routine

Note

When measuring profile data without specifying the -Sregion option with the fipp command, you do not need to add any measurement region specifying routine. For details on the -Sregion option, see "2.1.4 Measuring Profile Data".

To the source code, add the measurement region specifying routine required for specifying the region (starting and stopping positions of measurement) from which profile data is measured.

The following describes measurement region specifying routines for the Instant Performance Profiler in detail.

2.1.1.1 fipp_start / fipp_stop Subroutines (Fortran)

Format

CALL fipp_start

CALL fipp_stop

Function Description

These subroutines start or stop profile data measurement by the Instant Performance Profiler. They are enabled only when the -Sregion option is specified with the fipp command. For -Sregion option, see "2.1.4 Measuring Profile Data".

fipp_start

This subroutine starts profile data measurement by the Instant Performance Profiler.

fipp_stop

This subroutine stops profile data measurement by the Instant Performance Profiler.

Example

Example of use of measurement region specifying routines

```
program main
...
do i=1,10000
    ...
    call fipp_start   ! Start measurement
    do j=1,10000
        ...
    end do
    call fipp_stop    ! Stop measurement
end do
end program main
```

Note

- When calling these subroutines multiple times, be sure to call them in the order from fipp_start to fipp_stop. If fipp_start is called again before fipp_stop is called or fipp_stop is called before fipp_start is called, a warning message is output and the call is ignored. If a process ends without calling fipp_stop, the profile data of the region is not measured.
- If these subroutines are called multiple times, results from all the specified measurement regions are totaled.
- In the case of a process parallel program, call these subroutines for all processes that you want to make targets of measurement. The profile data of the processes for which the subroutines are not called is not measured.

Example

Example for making all processes targets of measurement (starting measurement before calling the mpi_init subroutine)

```
call fipp_start         ! Start measurement
call mpi_init(err)
  ...
call mpi_finalize(err)
call fipp_stop          ! Stop measurement
```

Example for making all processes targets of measurement (starting measurement immediately after calling the mpi_init subroutine)

```
call mpi_init(err)
call fipp_start         ! Start measurement
  ...
call fipp_stop          ! Stop measurement
call mpi_finalize(err)
```

Example for making only process 0 the target of measurement

```
call mpi_init(err)
call mpi_comm_rank(mpi_comm_world,rank,err)
if(rank==0) then
    call fipp_start   ! Start measurement on process 0 only
end if
    ...
if(rank==0) then
    call fipp_stop    ! Stop measurement on process 0 only
end if
call mpi_finalize(err)
```

- When the compiler option -mldefault=cdecl is valid to compile a Fortran program, change the name of the measurement region specifying routine as follows.

Before	After
fipp_start	fipp_start_
fipp_stop	fipp_stop_

- When the compiler option -AU is valid, the name of the measurement region specifying routine must be entered in lowercase characters.

2.1.1.2 fipp_start function / fipp_stop Function (C language and C++)

Format

```
#include "fj_tool/fipp.h"

void fipp_start();
void fipp_stop();
```

Function Description

These subroutines start or stop profile data measurement by the Instant Performance Profiler. They are enabled only when the -Sregion option is specified with the fipp command. For -Sregion option, see "2.1.4 Measuring Profile Data".

fipp_start()

This subroutine starts profile data measurement by the Instant Performance Profiler.

fipp_stop()

This subroutine stops profile data measurement by the Instant Performance Profiler.

Example

Example of use of measurement region specifying routines

```
#include "fj_tool/fipp.h"       // Include the header file
...
int main(void)
{
    int i,j;
    for(i=0;i<10000;i++){
        ...
        fipp_start();            // Start measurement
        for(j=0;j<10000;j++){
            ...
        }
        fipp_stop();             // Stop measurement
    }
    return 0;
}
```

Note

- When calling these functions multiple times, be sure to call them in the order from fipp_start to fipp_stop. If fipp_start is called again before fipp_stop is called or fipp_stop is called before fipp_start is called, a warning message is output and the call is ignored. If a process ends without calling fipp_stop, the profile data of the region is not measured.
- If these functions are called multiple times, results from all the specified measurement regions are totaled.

- In the case of a process parallel program, call these functions for all processes that you want to make targets of measurement. The profile data of the processes for which the functions are not called is not measured.

Example

Example for making all processes targets of measurement (starting measurement before calling the mpi_init function)

```
fipp_start();             // Start measurement
MPI_Init(&argc, &argv);
  ...
MPI_Finalize();
fipp_stop();              // Stop measurement
```

Example for making all processes targets of measurement (starting measurement immediately after calling the mpi_init function)

```
MPI_Init(&argc, &argv);
fipp_start();             // Start measurement
  ...
fipp_stop();
MPI_Finalize();           // Stop measurement
```

Example for making only process 0 the target of measurement

```
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
if(rank==0){
    fipp_start();  // Start measurement on process 0 only
}
    ...
if(rank==0){
    fipp_stop();   // Stop measurement on process 0 only
}
MPI_Finalize();
```

2.1.2 Specifying Environment Variables

Specify environment variables required when using the Profiler.

Environment Variable	Value
PATH	/*installation_path*/bin
LD_LIBRARY_PATH	/*installation_path*/lib64

For details on "*installation_path*", contact the system administrator.

Note

- Environment variables whose names begin with "FIPP_", "FAPP_", "PROF_", "FJPROF_", or "FPROF_" are used by the Profiler. Do not specify these environment variables when using the Profiler.
- When you specify mpiexec command's option "-x LD_LIBRARY_PATH=value" and execute program, see "5.1.5 mpiexec command".

2.1.3 Compilation

Compile a program. The following describes how to compile a program to use the Profiler.

Note

- If you specify the -rpath option or the environment variable LD_RUN_PATH when compiling a program, be careful not to include "/*installation_path*/lib64" in the dynamic section attribute DT_RPATH. If "/*installation_path*/lib64" is included in DT_RPATH, profile data measurement fails when the Instant Performance Profiler is used. From the output of the readelf -d command, you can check the value of DT_RPATH that is specified in the program. See Type "(RPATH)" in the output information. For details on "*installation_path*", contact the system administrator.
- Do not use system calls or functions that generate processes in the program, such as "fork", "vfork", "popen" system call, or "system" functions. If one is used, profile data cannot be measured correctly.
- When using the Profiler, do not use the strip command for the program. When measuring the cost of a user's shared library, do not use the strip command for the shared library. If a symbol is deleted by using the strip command, profile data cannot be measured correctly.
- When using the Profiler, use the compile commands of the Fujitsu compiler to compile and link your programs. You cannot use the Profiler when using the compile commands of other compilers, such as the GNU compiler.

2.1.3.1 Compiler Options

To use the Profiler, you need to create a program to which a tool library is linked.
In the frtpx, fccpx, FCCpx, mpifrtpx, mpifccpx, or mpiFCCpx command, specify compiler options that specify the operation of the tools.
For details on how to specify compiler options, see the "Fortran User's Guide", "C User's Guide", "C++ User's Guide", or "MPI User's Guide".

Point

- When using the frtpx, fccpx, FCCpx, mpifrtpx, mpifccpx, or mpiFCCpx command, create a program to which a tool library is linked by default. Therefore, you do not usually need to be conscious of these options. However, since the Profiler hooks MPI functions, you cannot use the Profiler together with the MPI profiling interface. If you use the MPI profiling interface, use the following compiler options to prevent the tool library from being linked. For details, see "5.1.6 Impact of Using the MPI Profiling Interface".
- Compiler options for each command include ones that have an impact on the operation of the Profiler. For detail, see "5.1.2 Impact of Compiler Options".

The following describes compiler options for the Profiler for each language.

2.1.3.1.1 Fortran

This section describes compiler options for the Profiler that are used when compiling a Fortran program.

-N{fjprof|nofjprof}

This option specifies whether to link a tool library. The option is enabled at the linking time. If you do not specify the option, the -Nfjprof option is enabled.

-Nfjprof

This option links the tool library. The Profiler is available.

-Nnofjprof

This option does not link the tool library. The Profile is unavailable.

2.1.3.1.2 C and C++ Languages

For the C and C++ languages, there are two types of modes: Trad and Clang. For details on the difference between Trad and Clang Modes, see the "C User's Guide" or "C++ User's Guide". This section describes compiler options for the Profiler that are used when compiling a program in each mode.

Trad Mode

The following describes compiler options for the Profiler that are used when compiling a C or C++ language program in Trad Mode.

-N{fjprof|nofjprof}

This option specifies whether to link a tool library. The option is enabled at the linking time. If you do not specify the option, the -Nfjprof option is enabled.

-Nfjprof

This option links the tool library. The Profiler is available.

-Nnofjprof

This option does not link the tool library. The Profile is unavailable.

Clang Mode

The following describes compiler options for the Profiler that are used when compiling a C or C++ language program in Clang Mode.

-f{fj-fjprof|fj-no-fjprof}

This option specifies whether to link a tool library. The option is enabled at the linking time. If you do not specify the option, the -Nfjprof option is enabled.

-ffj-fjprof

This option links the tool library. The Profiler is available.

-ffj-no-fjprof

This option does not link the tool library. The Profile is unavailable.

Note

Since the compiler is different in Trad Mode and Clang Mode, you may get meaningfully different profiling results when you compile in Trad Mode and Clang Mode, even if the program for compilation is the same.

2.1.3.2 Info Key fjprof_spawn_dir_name

When you run the MPI_COMM_SPAWN routine or the MPI_COMM_SPAWN_MULTIPLE routine to dynamically generate the processes, you can extract the profile data of the specific process generated dynamically to the directory other than one that you specified with the -d option of the fipp command, by setting the following info key to MPI_Info. For details on the -d option of the fipp command, see "2.1.4 Measuring Profile Data". For details on dynamically generating processes and on the info key related to dynamically generating processes, see the "MPI User's Guide".

Table 2.1 Info key "fjprof_spawn_dir_name"

Info key	Setting values	Description
fjprof_spawn_dir_name	Path to the output directory	Specify the output directory of the profile data of the dynamic process.

Example

Info key "fjprof_spawn_dir_name" Usage Example

```
#include "mpi.h"
#include <stdio.h>
#include <stdlib.h>

#define NUM_SPAWNS 1
int main( int argc, char *argv[] )
{
    int np = NUM_SPAWNS;
    int errcodes[NUM_SPAWNS];
    MPI_Comm intercomm;
    MPI_Info info;

    MPI_Init( &argc, &argv );
```

```
    MPI_Info_create(&info);
    MPI_Info_set(info, "fjprof_spawn_dir_name", "./spawn_data");
    MPI_Comm_spawn( "./spawn.out", MPI_ARGV_NULL, np, info, 0, MPI_COMM_WORLD, &intercomm,
errcodes );

    MPI_Finalize();
    return 0;
}
```

Note

- If you specify the relative path to the directory, the profile data is generated in the current directory when the MPI_COMM_SPAWN routine or the MPI_COMM_SPAWN_MULTIPLE routine is called. Note that the current directory is affected by the info key "wdir". For details, see the "MPI User's Guide".
- The directory specified in the info key "fjprof_spawn_dir_name" must be different from the directory specified in the -d option of the fipp command. Results cannot be guaranteed if you specify the same directory.
- If the specified directory exists, it must be an empty directory. If the specified directory does not exist, a new directory is created.
- If the profile data cannot be created in the output directory specified by the info key "fjprof_spawn_dir_name", a segmentation fault or other error may occur in the process that executed the MPI_COMM_SPAWN routine or the MPI_COMM_SPAWN_MULTIPLE routine.
- If a program that dynamically generates processes terminates abnormally,Check for any of the following messages.
  - The files had already existed in the specified value of fjprof_spawn_dir_name key.
  - The specified value of fjprof_spawn_dir_name key is not directory.
  - The specified value of fjprof_spawn_dir_name key is permission denied.

2.1.4 Measuring Profile Data

Measure data by using the fipp command. Perform this operation from a compute node.

Note

- If any of the following is performed on profile data measured by the fipp command, the operation is not guaranteed:
  - Editing profile data
  - Adding, deleting, or renaming profile data
- If the program is interrupted while profile data is being measured, incomplete profile data may remain.
- When using the fipp command, specify a value that is equivalent to "TRUE" in the environment variable "FLIB_FASTOMP". If do not specified, the fipp command does not operate correctly. For details on the environment variable "FLIB_FASTOMP", see the "Fortran User's Guide", "C User's Guide", or "C++ User's Guide".
- You can also specify the -A option for the fipp command. However, if you specify the -A option for the fipp command, it is treated as a fipppx command of the "2.1.5 Outputting Profile Result". Therefore, you should work with it differently. For details, see "2.1.5 Outputting Profile Result". You cannot specify the -A option together with the -C option for the fipp command. If you specify both options, an error message is output and the program is aborted.

fipp command syntax

```
fipp -C -d profile_data [ -I{{call|nocall}|{cpupa|nocpupa}|{mpi|nompi}} ] [ -H[mode={all|user}] ]
[ -L{shared|noshared} ] [ -M{inlined|noinlined} ] [ -P{userfunc|nouserfunc} ] [ -S{all|region} ]
[ -W{spawn|nospawn} ] [ -i interval ] [ -l limit ] [ -m memsize ] exec-file [ exec_option ... ]
```

fipp command options

Point

If the description of an option contains a restriction such as "you cannot..." or "it must be..." and you violate it, an error message is output and the program is aborted.

If you specify multiple conflicting options, the last specified option is enabled. For example, if you specify the -L{shared|noshared} option, which specifies how to measure detailed shared library information, in the order of "-Lshared -Lnoshared", the -Lnoshared option is enabled.

-C

This option specifies to measure profile data. The option cannot be omitted. If you do not specify the option, an error message is output and the collecting command is terminated.

-d *profile_data*

This option specifies the directory where profile data is to be stored. The option cannot be omitted. If you do not specify the option, an error message is output and the collecting command is terminated.
profile_data cannot be omitted. *profile_data* specifies a relative or absolute path as the name of the directory where profile data is to be stored. If the specified directory exists, it must be an empty directory. If the specified directory does not exist, a new directory is created. To specify a directory name beginning with "-" in *profile_data*, specify its absolute path or a relative path that contains the current directory ("./"). To analyze a program that moves the current directory during execution, specify its absolute path in *profile_data*.

-I{{call|nocall}|{cpupa|nocpupa}|{mpi|nompi}}

This option specifies items to be measured by the Instant Performance Profiler. If the -I{call|nocall} option is omitted, the -Inocall option is enabled. If the -I{cpupa|nocpupa} option is omitted, the -Icpupa option is enabled. If the -I{mpi|nompi} option is omitted, specification depends on the type of the program. In the case of an MPI program, the -Impi option is enabled. In the case of a non-MPI program, the -Inompi option is enabled. If the -Icpupa option is enabled but the -H option is not specified, the -Hmode=all option is enabled. The -I option allows you to specify multiple suboptions by separating them with commas (,). For example, you can specify like this: -Icall,nocpupa.

call

This argument specifies to measure call graph information.

nocall

This argument specifies not to measure call graph information.

cpupa

This argument specifies to measure CPU performance characteristics.

nocpupa

This argument specifies not to measure CPU performance characteristics.

mpi

This argument specifies to measure MPI cost information. If you specify the argument for a non-MPI program, an error message is output and the collecting command is terminated.

nompi

This argument specifies not to measure MPI cost information.

-H[mode={all|user}]

This option specifies how CPU performance characteristics are measured. The suboption mode specifies either all or user. If this option or the suboption mode={all|user} is omitted, mode=all is enabled. If you specify the -Inocpupa option, a warning message is output and this option is disabled.

mode=all

This option specifies to measure information in kernel and user modes.

mode=user

This option specifies to measure information in user mode.

-L{shared|noshared}

This option specifies how to measure the shared library generated by specifying the compiler option -Nline or -ffj-line (shared library with line information). For details on compiler options, see the "Fortran User's Guide", "C User's Guide", or "C++ User's Guide". If this option is omitted, the -Lnoshared option is enabled.

shared

This argument specifies to measure the following information in the shared library with line information:

- Start line number of the procedure
- End line number of the procedure
- Loop cost distribution information
- Line cost distribution information

noshared

This argument specifies not to measure the following information in the shared library with line information:

- Start line number of the procedure
- End line number of the procedure
- Loop cost distribution information
- Line cost distribution information

-M{inlined|noinlined}

This option specifies where costs of the inlined function are recorded in an inlined program.

If this option is omitted, the -Mnoinlined option is enabled.

inlined

In an inlined program, the cost of the inlined portion is counted as the cost of the inlined callee's procedure.

noinlined

In an inlined program, the cost of the inlined portion is counted as the cost of the caller's procedure.

When this option is used, following line numbers may indicate line numbers of a source code containing the definition of inlined procedure.

- Loop start line number and loop end line number on loop cost distribution information
- Line number on line cost distribution information

The source code information is displayed as follows.

- The cost of the inlined portion is counted as the cost of the inlined callee's procedure.

-P{userfunc|nouserfunc}

This option specifies how to appropriate the procedure cost. It applies to a mix of an object for which the compiler option -Nline or -ffj-line is specified (object with line information) and an object for which the compiler option -Nnoline or -ffj-no-line is specified (object without line information). The standard library and a shared library when -Lnoshared is specified are handled as objects without line information. For details on compiler options, see the "Fortran User's Guide", "C User's Guide", or "C++ User's Guide". If this option is omitted, the -Pnouserfunc option is enabled.

userfunc

If a cost is appropriated to a procedure of the object without line information, the procedure that called the procedure of the object without line information is traced back from call graph information. If a procedure of the object with line information exists, the cost is appropriated to the procedure. If no procedures of the object with line information exist, the cost is not appropriated. When specifying the -Puserfunc option, you must specify the -Icall option at the same time. If you do not specify the -Icall option, an error message is output and the collecting command is terminated.

nouserfunc

If a cost is appropriated to a procedure of the object without line information, the cost is appropriated to the procedure. However, the procedure start and end lines are not output.

-S{all|region}

This option specifies a profile data measurement region. If this option is omitted, the -Sall option is enabled.

all

This argument specifies to measure data across the entire program.

region

This argument specifies to measure data within the region specified by the measurement region specifying routine. You need to insert the measurement region specifying routine in the source code.

-W{spawn|nospawn}

This option specifies how to measure the dynamically generated process. Specification when you omit the -W{spawn|nospawn} option varies depending on the type of the program. When working with the MPI program, the -Wspawn option is enabled. When working with the non-MPI program, the -Wnospawn option is enabled.

spawn

This argument specifies to measure the statistic information of the dynamically generated process. If you specify the info key "fjprof_spawn_dir_name" for the specific dynamically generated process, the profile data of the dynamically generated process with the info key is stored in the directory that you specified for info key "fjprof_spawn_dir_name". If you do not specify the info key "fjprof_spawn_dir_name", the profile data of the dynamically generated process is stored in the directory which you specified with the -d option. If you specify this argument to the non-MPI program, an error message is output and the collecting command is terminated.

nospawn

This argument specifies not to measure the statistic information of the dynamically generated process. However, if you specify the info key "fjprof_spawn_dir_name" to the specific dynamically generated process, measurement is performed for that process with the info key, and the measured profile data is stored in the directory which you specified for the info key "fjprof_spawn_dir_name".

-i *interval*

This option specifies the sampling interval to measure profile data. *interval* specifies an integer that represents the sampling interval (in milliseconds). If this option is omitted, the -i 100 option is enabled. *interval* cannot be omitted. *interval* specifies an integer from 10 to 3,600,000. If you specify a value outside the range in *interval*, a warning message is output and the -i 100 option is enabled.

-l *limit*

This option specifies the number of measurements of procedure information. Procedure information that exceeds the number of outputs is measured by combining it into "__other__". If this option is omitted, the -l 0 option is enabled. *limit* cannot be omitted. *limit* specifies an integer from 0 to 2,147,483,647. If you specify 0 in *limit*, all procedure information is measured. If you specify a value outside the range in *limit*, a warning message is output and the -l 0 option is enabled.

-m *memsize*

This option specifies the size of the working memory to be used for measurement. An area of the working memory is secured for each thread. If this option is omitted, the -m 3000 option is enabled. *memsize* specifies an integer that represents the working memory size (in KB). *memsize* cannot be omitted. *memsize* specifies an integer from 1 to 2,147,483. If you specify a value outside the range in *memsize*, a warning message is output and the -m 3000 option is enabled.

exec-file [*exec_option* ...]

This option specifies the execution file that is the target of profile data measurement and options for the file. In the case of an MPI program, make a specification from mpiexec. To specify an execution file that begins with "-" in *exec-file*, specify its relative path that contains the current directory ("./") or absolute path. A shell script cannot be specified in *exec-file*. The character string following the execution file name (*exec_option* ...) is considered an option for the execution file.

Example

Example of measuring CPU performance characteristics and call graph information with the fipp command

```
fipp -C -d ./tmp -Icall ./a.out
```

2.1.5 Outputting Profile Result

Output the profile data which is measured using the fipp command. For this operation, use a different command depending on the node to be used.

When using a login node

Use the fipppx command.

When using a compute node

Use the fipp command.

fipppx command or fipp command syntax

```
{fipppx|fipp} -A [ -I{{balance|nobalance}|{call|nocall}|{cpupa|nocpupa}|{mpi|nompi}|
{src[:path ]|nosrc}} ] [ -Tt_no ] [ -b{max|total} ] [ -f func_name ] [ -l limit ]
[ -o outfile ] [ -pp_no ] [ -t{csv|text|xml} ] [ -u ] [ -d ] profile_data
```

fipppx command or fipp command options

Point

If the description of an option contains a restriction such as "you cannot..." or "it must be..." and you violate it, an error message is output and the program is aborted.

If you specify multiple conflicting options, the last specified option is enabled. For example, if you specify the -I{call|nocall} option, which specifies profile result items to be output, in the order of "-Icall,nocall", the -Inocall option is enabled.

-A

This option specifies to output the profile result. The option cannot be omitted. If you do not specify the option, an error message is output and the analyzing command is terminated.

-I{{balance|nobalance}|{call|nocall}|{cpupa|nocpupa}|{mpi|nompi}|{src[:*path*]|nosrc}}

This option specifies profile result items to be output. The -I option allows you to specify multiple suboptions by separating them with commas (,). For example, you can specify like this: -Icall,cpupa. If the -I{balance|nobalance} option is omitted, the -Inobalance option is enabled. If the -I{call|nocall} option is omitted, the -Inocall option is enabled. If the -I{cpupa|nocpupa} option is omitted, the -Icpupa option is enabled. If the -I{mpi|nompi} option is omitted, specification depends on the type of the target program for profile data measurement. In the case of an MPI program, the -Impi option is enabled. In the case of a non-MPI program, the -Inompi option is enabled. If the -I{src[:*path*]|nosrc} option is omitted, the -Inosrc option is enabled. To specify the -Icall, cpupa, or mpi option, relevant items need to have been measured with the fipp command. If you specify to output information that has not been measured, an error message is output and the analyzing command is terminated.

balance

This argument specifies to output cost balance information (information obtained by comparing the cost between parallel execution units) in the cost information. However, when you work on a serial program, or when you specify the -tcsv option or the -txml option, cost balance information will not be output even if you specify the -Ibalance option.

nobalance

This argument specifies not to output cost balance information as cost information.

call

This argument specifies to output call graph information.

nocall

This argument specifies not to output call graph information.

cpupa

This argument specifies to output CPU performance characteristics.

nocpupa

This argument specifies not to output CPU performance characteristics.

mpi

This argument specifies to output MPI cost information as cost information.

nompi

This argument specifies not to output MPI cost information as cost information.

src[:*path*] ...

This argument specifies to output source code information and per-line costs. The cost that is output when the -Impi option is specified is not included in the per-line costs. *path* specifies the directory path where the source code exists. To specify multiple *path* values, separate them with colons (:). If *path* is omitted, the directory path specified at the time of program compilation is referenced. If you specify the -tcsv option or the -txml option, the source code information will not be output even if you specify the -Isrc option.

nosrc

This argument specifies not to output source code information.

-T*t_no*

This option specifies the thread whose profile data is to be output. *t_no* specifies one or more of *N*, limit=*n*, and all. If this option is omitted, the -Tall option is enabled. *t_no* cannot be omitted. The -T option allows you to specify multiple *t_no* values by separating them with commas (,). For example, you can specify like this: -T3,5,limit=10.

N[,*N*] ...

This suboption specifies to output, at the beginning, the information of the thread number specified by *N*. If the information of the thread specified by *N* does not exist, the specification is ignored. If multiple *N* values are specified, information is output in the order of specification.

limit=*n*

This suboption specifies to output the information of *n* threads in the order of cost from highest. If you specify 0 or a value exceeding the total number of threads in *n*, the information of all threads is output.

all

This suboption specifies to output the information of all threads. This is the same as when the -Tlimit=0 option is specified. If you do not specify the suboption limit=*n*, this option is enabled.

-b{max|total}

Specifies how cost information is output when profiling results in TEXT format. If the -b{max|total} option is omitted, the -btotal option is enabled. This option is only valid with the -ttext option.

max

Cost information in TEXT format is output in "Max thread cost basis". "Max thread cost basis" outputs the value of the most expensive thread in the process as cost information for that process. It also outputs the value of the most expensive process in the program as cost information for that program. When you specify this suboption, CPU performance characteristics and call graph information will not be output even if you specify the -Icpupa option or -Icall option. And, when you specify this suboption, CPU frequency will not be output. You cannot specify this option together with the -u option. If you specify both options, an error message is output and the analyzing command is terminated.

total

Cost information in TEXT format is output in "Total thread cost basis". "Total thread cost basis" outputs the total value of all threads in a process as cost information for that process. It also outputs the total value of all processes in a program as cost information for that program.

-f *func_name*

This function specifies to output a specific procedure. *func_name* specifies a procedure name used by the program. Even if the cost of *func_name* is outside the range of the number of outputs that is specified with the -l option, information about *func_name* is output. However, if any of the following applies, specifying -f *func_name* does not output information:

- Information about the *func_name* procedure is not measured with the fipp command.
- The procedure cost of *func_name* is 0.

-l *limit*

This option specifies the number of outputs for procedure information to be output. If the -l option is omitted, specification depends on the -t{csv|text|xml} option. If the -ttext option is enabled, -l 10 option is enabled. If the -tcsv option or -txml option is enabled, the -l 0 option is enabled. *limit* specifies an integer from 0 to 2,147,483,647. If you specify 0 in *limit*, all are output. If you specify a value outside the range in *limit*, the -l 0 option is enabled.

-o *outfile*

This option specifies the output destination for the profile result. *outfile* specifies a relative or absolute path as the name of the output destination file or stdout. If this option is omitted, the -ostdout option is enabled. If you specify stdout in *outfile*, the profile result is output to the standard output. To specify a file name beginning with "-" in *outfile*, specify its absolute path or a relative path that contains the current directory ("./").

-p*p_no*

Specify the process to be output to the profile result. For *p_no*, specify one or more from those of *N*@*M* , input=*n* , limit=*m* , and all. If you omit this option, the value taken in varies depending on the specification of the -t{csv|text|xml} option. If the -ttext option is enabled, the "-pinput=0,limit=16" option is enabled. If the -tcsv option or -txml option is enabled, the -pall option is enabled. *p_no* cannot be omitted. For the -p option, you can specify more than one *p_no* by separating them with a comma (,). For example, you can specify such as "-p3,5,limit=10".

N[@*M*] ...

This suboption specifies to output, at the beginning, the information of the process number specified in *N*. If you specify the process number *N* that belongs to the spawn number *M*, specify in the format like "*N*@*M*". If the information of the process number specified with *N*[@*M*] does not exist, ignore the specification. You can specify more than one *N*[@*M*]. If you specify more than one *N*[@*M*], the result is output in the order of your specification. [@*M*] is only valid with the -ttext option.

input=*n*

This suboption specifies to read the information for the top *n* processes at cost. Although processing becomes faster because the number of files to read decreases, the information of processes that is not read, is not included in the denominator to perform ratio calculation. If you specify 0 or a value exceeding the number of processes in *n*, the information of all processes is read. If this suboption is omitted, input=0 is enabled. If you specify input=*n* and all together, the suboption input=*n* is enabled, regardless of the order in which options are specified. The suboptions input=*n* and limit=*m* can be specified together.

limit=*m*

This suboption specifies to output the information for the top *m* processes at cost. If the -ttext option is enabled, output processes in the order of cost from highest. The information of processes that is not output is included in the denominator to perform ratio calculation. If you specify 0 or a value exceeding the number of processes in *m*, the information of all processes is output. If you omit this suboption, the value taken in varies depending on the specification of the -t{csv|text|xml} option. If the -ttext option is enabled, limit=16 is enabled. If the -tcsv option or -txml option is enabled, limit=0 is enabled.

all

This suboption specifies to read and output the information for all processes. If the -ttext option is enabled, output processes in the order of cost from highest. This is the same as when the -pinput=0,limit=0 option is specified. If you do not specify either the suboption input=*n* or limit=*m*, this suboption is enabled.

-t{csv|text|xml}

This option specifies the output format of the profile result. If this option is omitted, the -ttext option is enabled.

csv

Outputs the profile result in the CSV format.

text

Outputs the profile result in the TEXT format.

xml

Outputs the profile result in the XML format.

-u

This option specifies how to output the generated procedure name of the thread parallel program (refer to "Table 2.6 Names of Generated Procedures for Thread Parallel Programs" for the generated procedure name). If this option is specified, the cost information of the procedure and the generated procedure are summed and output as procedure name. If this option is not specified, output procedure's row and generated procedure's one separately. This option is only valid with the -tcsv option or -ttext option. You cannot specify this option together with the -bmax option. If you specify both options, an error message is output and the analyzing command is terminated.

-d *profile_data*

profile_data specifies a relative or absolute path as the name of the directory where profile data is stored. This option cannot be omitted. However, as long as you specify *profile_data* at the end of an array of options, "-d" can be omitted. To specify a directory name beginning with "-" in *profile_data*, specify its absolute path or relative path that contains the current directory ("./").

Example

Example of outputting CPU performance characteristics and MPI cost information from the measurement result ./tmp of the fipp command

```
fipppx -A -Inobalance,cpupa,mpi,nocall,nosrc -d ./tmp
```

2.2 Profile Result

This section describes the contents of profile result output by the fipppx command or fipp command.

2.2.1 Overview of Profile Result

The profile result consists of the following statistic information. You can control the output of information respectively by specifying the -I option of the fipppx command or fipp command. For details on the -I option, see "2.1.5 Outputting Profile Result". The following information is output.

- Environment information for measuring profiling data
- Statistical time information
- CPU performance characteristics
- Cost information(procedure cost distribution information , loop cost distribution information , line cost distribution information)
- Call graph information
- Source code information (TEXT format only)

For details, see "2.2.2 Details of Profile Result (TEXT Format)" or "2.2.3 Details of Profile Result (XML Format)" and their subsections.

Point

Instant Performance Profiler outputs profile result in units of information total level "Application", "Process" and "Thread", but if the number of process or thread is 1, it skips outputting the corresponding information total level.

2.2.2 Details of Profile Result (TEXT Format)

If you specify the "2.1.5 Outputting Profile Result" with the -ttext option, the result is output in the TEXT format. The followings are applied to the TEXT format.

2.2.2.1 Environment Information for Measuring Profiling Data

As environment information for measuring profiling data, environment information as of when the profile data was measured is output.

Output format of the environment information for measuring profiling data

```
-----------------------------------------------------------------------------------------
Fujitsu Instant Performance Profiler Version @vl
   Measured time                 : @date
   CPU frequency                 : Process      @pno  @frequency (MHz) @sno
   Type of program               : @type
   Average at sampling interval  : @interval (ms)
   Measured region               : @region
   Virtual coordinate            : (@x, @y, @z)
-----------------------------------------------------------------------------------------
```

Table 2.2 Output items of the environment information for measuring profiling data

Output Item	Meaning of Output Item
@vl	Version number of the Profiler
@date	Measurement date and time of profile data
@pno	Process number
@frequency	CPU Frequency **Note** If any of the following conditions applies, "--" is output as the CPU frequency: - The -Inocpupa option is specified in "2.1.4 Measuring Profile Data" or "2.1.5 Outputting Profile Result" - The -Hmode=user option is specified in "2.1.4 Measuring Profile Data" - The -Sregion option is specified in "2.1.4 Measuring Profile Data" and measurement has never been performed on the measurement region specified in "2.1.1 Adding a Measurement Region Specifying Routine" - The -bmax option is specified in "2.1.5 Outputting Profile Result"
@sno	Spawn number It is taken in the (Spawn @num) format. (Where @num is a numerical value) This item is output only when all the following conditions are met. - The measurement target is the MPI program. - When the fipp command is set with the -Wspawn option enabled, or the info key "fjprof_spawn_dir_name" is set in the program. - The process is a dynamically generated process.
@type	Program execution format "SERIAL" : Serial "Thread (AUTO)" : Automatic parallelization only "Thread (OpenMP) " : OpenMP only "Thread (OpenMP & AUTO) " : OpenMP and automatic parallelization "MPI" : MPI only "MPI & Thread (AUTO) " : MPI, Automatic parallelization "MPI & Thread (OpenMP) " : MPI,OpenMP "MPI & Thread (OpenMP & AUTO) " : MPI, OpenMP, and automatic parallelization
@interval	Sampling interval Outputs the average value of actual sampling intervals.

Output Item	Meaning of Output Item
@region	Kind of measurement region All: Entire program Specified: Specified measurement region
(@x, @y, @z)	Logical shape at the time of MPI program execution **Point** Outputs the result only when the data measurement target is an MPI program. The logical shape of dynamically generated processes are not outputted.

Point

If CPU frequency(including dynamically generated process) is different for each process, multiple lines of CPU frequency are output. However, if there are consecutive processes with the same CPU frequency, or if the -Inocpupa option is specified, CPU frequency is output as a single line.

```
CPU frequency                  : Process      @spno - @lpno @frequency (MHz)   (Spawn @ssno - @lsno )
```

For consecutive processes, @spno is the lowest process number and @lpno is the highest process number. @ssno is the lowest spawn number and @lsno is the highest spawn number. If the -pinput=*n* option of the fipppx command or fipp command is used to restrict the process being read, the minimum and maximum process numbers or spawn numbers that are read and with the same CPU frequency in the range correspond to @spno and @lpno respectively. See "2.1.5 Outputting Profile Result" for the -pinput=*n* option.

2.2.2.2 Statistics Time Information

As statistical time information, elapsed time of the program is output for each Application, each Process, or each Thread. And user CPU time, and system CPU time of the program are output for each Application, each Process.

Output format of the statistics time information

```
Time statistics

       Elapsed(s)        User(s)      System(s)
  --------------------------------------------
          @elapse        @user          @system  @level  @pno  @sno
  --------------------------------------------
          @elapse        @user          @system  @level  @pno  @sno
```

Table 2.3 Output items of the statistics time information

Output Item	Meaning of Output Item
@elapse	Elapsed time (s)
@user	User CPU time (s) If the @type of "2.2.2.1 Environment Information for Measuring Profiling Data" is not "SERIAL" or "MPI", or if the @level is "Thread", this output item is fixed as "--".
@system	System CPU time (s) If the @type of "2.2.2.1 Environment Information for Measuring Profiling Data" is not "SERIAL" or "MPI", or if the @level is "Thread", this output item is fixed as "--".
@level	Information total level (Application, Process, Thread)
@pno	Process or thread number
@sno	Spawn number

Output Item	Meaning of Output Item
	It is taken in the (Spawn @num) format. (Where @num is a numerical value) This item is output only when all the following conditions are met. - The measurement target is the MPI program. - When the fipp command is set with the -Wspawn option enabled, or the info key "fjprof_spawn_dir_name" is set in the program. - When @level is Process. - The process is a dynamically generated process.

2.2.2.3 CPU Performance Characteristics

As CPU performance characteristics, information related to CPU performance characteristics, such as memory throughput, number of instructions, and number of operations, is displayed. This information is output when the -Icpupa option is enabled. CPU performance characteristics are totaled and output in the following units:

- Application
- Process
- Thread

Output format of the CPU performance characteristics

```
Performance monitor event : statistics

 ********************************************************************************
     @level  @pno  @sno    - performance monitors
 ********************************************************************************

   Execution                        Floating-point  Mem throughput  Mem throughput
          time(s)          GFLOPS   peak ratio(%)          (GB/s)   peak ratio(%)
 ------------------------------------------------------------------------------
           @time         @value1        @value2        @value3        @value4   @level   @pno   @sno
 ------------------------------------------------------------------------------
           @time         @value1        @value2        @value3        @value4   @level   @pno   @sno

   Effective       Floating-point  SIMD inst.      SVE operation
      instruction       operation         rate(%)         rate(%)
 ------------------------------------------------------------------------------
         @value5        @value6        @value7        @value8                  @level   @pno   @sno
 ------------------------------------------------------------------------------
         @value5        @value6        @value7        @value8                  @level   @pno   @sno

             IPC            GIPS
 ------------------------------------------------------------------------------
         @value9       @value10                                               @level   @pno   @sno
 ------------------------------------------------------------------------------
          @value9        @value10                                                  @level   @pno
@sno
```

Table 2.4 Output items of the CPU performance characteristics

Output Item	Meaning of Output Item
@level	Information total level (Application, Process, Thread)
@pno	Process or thread number
@sno	Spawn number It is taken in the (Spawn @num) format. (Where @num is a numerical value)

Output Item	Meaning of Output Item
	This item is output only when all the following conditions are met. - The measurement target is the MPI program. - When the fipp command is set with the -Wspawn option enabled, or the info key "fjprof_spawn_dir_name" is set in the program. - When @level is Process. - The process is a dynamically generated process.
@time	Time taken to execute instructions in the measurement target region (second) The value on the top shows the maximum execution time of each thread.
@value1	Count of floating-point operations performed per second Process: The value on the top shows the count of floating-point operations performed per second on the process. Application: The value on the top shows the count of floating-point operations performed per second on the application. Note The GFLOPS value is calculated by assuming that all elements are active. Therefore, in the case of a program with many inactive elements, a higher value than the original GFLOPS value is output.
@value2	Ratio of the measured value to the theoretical value for floating-point operation performance (%) Process: The value on the top shows the ratio of @value1 of the process to the theoretical value for floating-point operation performance. Application: The value on the top shows the ratio of @value1 of the application to the theoretical value for floating-point operation performance. Note The theoretical value for floating-point operation performance is calculated by assuming that double precision operations are performed. Therefore, in the case of single or half precision, a value 2 to 4 times higher than the actual percentage is output.
@value3	Memory throughput (GB/s) Process: The value on the top shows the ratio of the total amount of memory access to @time of the process. Application: The value on the top shows the ratio of the total amount of memory access to @time of the application.
@value4	Ratio of the measured value to the theoretical value for memory throughput (%) Process: The value on the top shows the ratio of @value3 of the process to the theoretical value for memory throughput. Application: The value on the top shows the ratio of @value3 of the application to the theoretical value for memory throughput.
@value5	Total number of instructions executed Process: The value on the top shows the total number of instructions executed on the process. Application: The value on the top shows the total number of instructions executed on the application. Note The total number of instructions executed does not include MOVPRFX instructions.
@value6	Total number of floating-point operations executed Process: The value on the top shows the total number of floating-point operations executed on the process.

Output Item	Meaning of Output Item
	Application: The value on the top shows the total number of floating-point operations executed on the application.
@value7	Ratio of the number of SIMD instructions to the total number of instructions executed (%) Process: The value on the top shows the ratio of the total number of SIMD instructions to @value5 of the process. Application: The value on the top shows the ratio of the total number of SIMD instructions to @value5 of the application.
@value8	Ratio of the number of SVE operations to the total number of floating-point operations executed (%) Process: The value on the top shows the ratio of the total number of SVE operations to @value6 of the process. Application: The value on the top shows the ratio of the total number of SVE operations to @value6 of the application.
@value9	Number of instructions executed per cycle Process: The value on the top shows the result obtained by dividing @value5 by the total number of cycles of the process. Application: The value on the top shows the result obtained by dividing @value5 by the total number of cycles of the application.
@value10	Number of instructions executed per second Process: The value on the top shows the result obtained by dividing @value5 by @time of the process. Application: The value on the top shows the result obtained by dividing @value5 by @time of the application.

Note

The output of @time and each @value is expected to be less than 12 digits. Therefore, if the output exceeds 13 digits, there is a discrepancy between the heading and the output.

2.2.2.4 Cost Information

The cost information consists of the following information:

- Procedure cost distribution information
- Loop cost distribution information
- Line cost distribution information

In the case of a parallel program, you can output cost balance information as part of procedure cost distribution information and loop cost distribution information. The -b option of the fipppx command specifies either "total thread cost basis" or "max thread cost basis" as the output format of Cost information. For details, see "2.1.5 Outputting Profile Result".

2.2.2.4.1 Procedure Cost Distribution Information

As procedure cost distribution information, procedure cost, waiting cost for synchronization between threads, procedure start line number, procedure end line number, and procedure name are output for each Application, each Process, or each Thread. Procedure cost distribution information is always output. MPI cost information is output only when the -Impi option is enabled. Cost balance information is output only when the -Ibalance option is enabled.

Output format of the procedure cost distribution information (Total thread cost basis)

Always output

```
Procedures profile (Total thread cost basis)

   ********************************************************************************
       @level @pno @sno - procedures
         Application and Process outputs the total value of the cost of each thread.
         Procedure outputs the total value of the procedure cost of each thread.
   ********************************************************************************
          Cost          %  Operation (s)        Barrier              %    Start      End
```

```
-----------------------------------------------------------------------------------------
       @cost @cost-rate         @ope       @barrier @barrier-rate      --      --   @level  @pno  @sno
-----------------------------------------------------------------------------------------
        @cost @cost-rate           @ope        @barrier @barrier-rate   @start     @end   @name
```

Additional output when the -Impi option is enabled

```
            MPI           %  Communication (s)    Start      End
  -------------------------------------------------------------
           @mpi   @mpi-rate              @comm       --       --   @level  @pno  @sno
  -------------------------------------------------------------
           @mpi   @mpi-rate              @comm   @start     @end   @name
```

Additional output when the -Ibalance option is enabled

```
  __                          __
  __  Parallel balance of cost  __

  @name  @start - @end
     +-------------------------+-------------------------+
     |                         |                         |   @balance  @cost2  @pno  @sno
     +-------------------------+-------------------------+
```

Output format of the procedure cost distribution information (Max thread cost basis)

Always output

```
Procedures profile (Max thread cost basis)

   ******************************************************************************************
       @level @pno @sno - procedures
         Application and Process outputs the max value of the cost of each thread.
         Procedure outputs the max value of the procedure cost of each thread.
   ******************************************************************************************

Spawn   Process   Thread     Cost          %  Operation (s)   Barrier             %    Start      End
-------------------------------------------------------------------------------------------------------
-
@spawn @process  @thread    @cost @cost-rate          @ope  @barrier @barrier-rate      --      --
@level @pno @sno
-------------------------------------------------------------------------------------------------------
-
@spawn @process  @thread    @cost @cost-rate          @ope  @barrier @barrier-rate   @start     @end
@name
```

Additional output when the -Impi option is enabled

```
  Spawn  Process   Thread          MPI          %  Communication (s)    Start      End
-------------------------------------------------------------------------------------------
 @spawn @process  @thread        @mpi  @mpi-rate            @comm      --      --   @level    @pno @sno
-------------------------------------------------------------------------------------------
 @spawn @process  @thread         @mpi  @mpi-rate              @comm   @start     @end   @name
```

Additional output when the -Ibalance option is enabled

```
   __                          __
   __  Parallel balance of cost  __

   @name  @start - @end
      +-------------------------+-------------------------+
      |                         |                         |   @balance  @cost2  @pno  @sno
      +-------------------------+-------------------------+
```

Table 2.5 Output items of the procedure cost distribution information

Output Item	Meaning of Output Item
@level	Information total level (Application, Process, Thread)
@pno	Process or thread number
@sno	Spawn number It is taken in the (Spawn @num) format. (Where @num is a numerical value) This item is output only when all the following conditions are met. - The measurement target is the MPI program. - When the fipp command is set with the -Wspawn option enabled, or the info key "fjprof_spawn_dir_name" is set in the program. - When @level is Process. - The process is a dynamically generated process.
@spawn	Spawn number chosen as max cost It is taken in the (Spawn @num) format. (Where @num is a numerical value) This item is output only when all the following conditions are met. If the condition is not met, only "--" is output for this item. - The measurement target is the MPI program. - When the fipp command is set with the -Wspawn option enabled, or the info key "fjprof_spawn_dir_name" is set in the program. - When @level is Process. The process is a dynamically generated process.
@process	Process number chosen as max cost
@thread	Thread number chosen as max cost
@cost	Procedure cost (includes @barrier cost)
@cost-rate	Proportion of the @level or @name cost in the cost at the information total level (%)
@ope	Operation time (s)
@barrier	Waiting cost for synchronization between threads Note If it is other than the thread parallel program, only "--" is output for this item.
@barrier-rate	Proportion of the waiting cost for synchronization between threads in the cost at the information total level (%) Note If it is other than the thread parallel program, only "--" is output for this item.
@mpi	MPI cost
@mpi-rate	Proportion of the MPI cost in the cost at the information total level (%)
@comm	Communication time (s)
@start	Procedure start line number
@end	Procedure end line number
@name	Procedure name

Output Item	Meaning of Output Item
@balance	Cost balance between procedure processes or threads (%) @balance = (@cost2 - (average value of @cost2)) / (average value of @cost2) * 100 @balance can exceed ± 100% If @balance is 4 digits or more, the display column will shift.
@cost2	Process or thread procedure cost (excluding thread barrier costs)

In the case of a thread parallel program, the parallelized part is output as information on the generated procedure. An identifier corresponding to the type of the generated procedure is appended to the name of the generated procedure. In the case of the C /C++ language, the name of the generated procedure varies depending on the mode of the compiler used at the time of compilation. For details, see the articles about the internal function name in "C User's Guide" or "C++ User's Guide". The table below lists names of generated procedures for thread parallel programs by generated procedure type.

Table 2.6 Names of Generated Procedures for Thread Parallel Programs

Language	Type of created procedure	Created procedure name
Fortran	Automatic parallelized procedure	*procedure_name*._PRL_*number*_
	OpenMP parallelized procedure	*procedure_name*._OMP_*number*_
	TASK construct	*procedure_name*._TSK_*number*_
C language/C++ (Trad Mode)	Automatic parallelized procedure	*procedure_name*._PRL_*number*_
	OpenMP parallelized procedure	*procedure_name*._OMP_*number*_
	TASK construct	*procedure_name*._TSK_*number*_
C language/C++ (Clang Mode)	OpenMP parallelized procedure	*procedure_name*.omp_outlined._debug__*number* (*1)
	TASK construct	*procedure_name*.omp_task_entry.*number*

*1: In Clang Mode, no automatic parallelized procedure is performed. Add "_debug__" when the compiler option -g is specified.

Information

You can check TASK construct cost information from procedure cost distribution information at the total level "Application". Also, you can check the distribution of TASK constructs across a thread from procedure cost distribution information or cost balance information at the information total level "Thread". However, the Instant Performance Profiler outputs top 10 procedure information items by default, and may not output TASK construct information where costs are smaller. Therefore, we recommend to change the number of procedure information items to be output by using the -l option of the "2.1.5 Outputting Profile Result" command to check the costs of TASK constructs.

2.2.2.4.2 Loop Cost Distribution Information

As loop cost distribution information, loop cost, waiting cost for synchronization between threads, nest level, loop type, loop compilation type, loop start line number, loop end line number, and name of the procedure to which the loop belongs are output. These are output for each Application, each Process, or each Thread. Loop cost distribution information is always output. MPI cost information is output only when the -Impi option is enabled. Cost balance information is output only when the -Ibalance option is enabled.

Output format of the loop cost distribution information (Total thread cost basis)

Always output

```
Loops profile (Total thread cost basis)

   ********************************************************************************
       @level @pno @sno - loops
         Application and Process outputs the total value of the cost of each thread.
         Procedure outputs the total value of the loop cost of each thread.
   ********************************************************************************
```

```
 Cost          %  Operation (s)    Barrier              %    Nest    Kind     Exec    Start      End
-----------------------------------------------------------------------------------------------------
@cost @cost-rate          @ope    @barrier @barrier-rate      --      --       --       --       --
@level  @pno  @sno
-----------------------------------------------------------------------------------------------------
@cost @cost-rate          @ope    @barrier @barrier-rate   @nest   @kind    @exec   @start     @end
@name
```

Additional output when the -Impi option is enabled

```
              MPI          %  Communication (s)    Nest    Kind     Exec    Start      End
  ---------------------------------------------------------------------------------------
           @mpi  @mpi-rate            @comm     --    --      --        --      --   @level  @pno  @sno
  ---------------------------------------------------------------------------------------
             @mpi  @mpi-rate              @comm   @nest   @kind    @exec   @start     @end   @name
```

Additional output when the -Ibalance option is enabled

```
  __                          __
  __  Parallel balance of cost  __

  @name  @start - @end
     +-------------------------+-------------------------+
     |                         |                         |   @balance  @cost2  @pno  @sno
     +-------------------------+-------------------------+
```

Output format of the loop cost distribution information (Max thread cost basis)

Always output

```
Loops profile (Max thread cost basis)

   ********************************************************************************
       @level @pno @sno - loops
         Application and Process outputs the max value of the cost of each thread.
         Procedure outputs the max value of the loop cost of each thread.
   ********************************************************************************

 Spawn  Process   Thread   Cost          %  Operation (s)    Barrier             %  Nest  Kind  Exec
Start  End
-----------------------------------------------------------------------------------------------------
------------
@spawn @process  @thread  @cost @cost-rate          @ope   @barrier @barrier-rate    --    --    --
--   --  @level  @pno @sno
-----------------------------------------------------------------------------------------------------
------------
@spawn @process  @thread  @cost @cost-rate           @ope   @barrier @barrier-rate @nest @kind @exec
@start @end   @name
```

Additional output when the -Impi option is enabled

```
Spawn  Process   Thread    MPI          %  Communication (s)  Nest  Kind   Exec    Start    End
-----------------------------------------------------------------------------------------------------
@spawn @process  @thread   @mpi  @mpi-rate            @comm    --    --     --      --     --   @level
@pno @sno
-----------------------------------------------------------------------------------------------------
@spawn @process  @thread   @mpi  @mpi-rate             @comm @nest @kind  @exec   @start   @end   @name
```

Additional output when the -Ibalance option is enabled

```
__                          __
 __  Parallel balance of cost  __

  @name  @start - @end
```

```
+-------------------------+-------------------------+
|                         |                         |   @balance  @cost2  @pno  @sno
+-------------------------+-------------------------+
```

Table 2.7 Output items of the loop cost distribution information

Output Item	Meaning of Output Item
@level	Information total level (Application/Process/Thread)
@pno	Process or thread number
@sno	spawn number It is taken in the (Spawn @num) format. (Where @num is a numerical value) This item is output only when all the following conditions are met. - The measurement target is the MPI program. - When the fipp command is set with the -Wspawn option enabled, or the fjprof_spawn_dir_name key is set in the program. - When @level is Process. - The process is a dynamically generated process.
@spawn	Spawn number chosen as max cost It is taken in the (Spawn @num) format. (Where @num is a numerical value) This item is output only when all the following conditions are met. If the condition is not met, only "--" is output for this item. - The measurement target is the MPI program. - When the fipp command is set with the -Wspawn option enabled, or the info key "fjprof_spawn_dir_name" is set in the program. - When @level is Process. The process is a dynamically generated process.
@process	Process number chosen as max cost
@thread	Thread number chosen as max cost
@cost	Loop cost (includes @barrier cost)
@cost-rate	Proportion of the @level or @name cost in the cost at the information total level (%)
@ope	Operation time (s)
@barrier	Waiting cost for synchronization between threads This item is output only in the case of a thread parallel program. Note If it is other than the thread parallel program, only "--" is output for this item.
@barrier-rate	Proportion of the waiting cost for synchronization between threads in the cost at the information total level (%) This item is output only in the case of a thread parallel program. Note If it is other than the thread parallel program, only "--" is output for this item.
@mpi	MPI cost
@mpi-rate	Proportion of the MPI cost in the cost at the information total level (%)

Output Item	Meaning of Output Item
@comm	Communication time (s)
@nest	Nest level
@kind	Loop type (DO/WHILE/UNTIL/ARRAY/FOR/GOTO/OTHER)
@exec	Loop compilation type (SERIAL: serial/OpenMP: OpenMP/AUTO: automatic parallelization)
@start	Loop start line number
@end	Loop end line number
@name	Procedure name
@balance	Cost balance between loop processes or threads (%) @balance = (@cost2 - (average value of @cost2)) / (average value of @cost2) * 100 @balance can exceed ± 100% If @balance is 4 digits or more, the display column will shift.
@cost2	Process or thread loop cost (excluding thread barrier costs)

2.2.2.4.3 Line Cost Distribution Information

As line cost distribution information, line cost, line number, and name of the procedure to which the line belongs are output for each Application, each Process, or each Thread. Line cost distribution information is always output. MPI cost information is output only when the -Impi option is enabled.

Output format of the line cost distribution information (Total thread cost basis)

Always output

```
Lines profile (Total thread cost basis)

   ********************************************************************************
       @level @pno @sno - lines
         Application and Process outputs the total value of the cost of each thread.
         Procedure outputs the total value of the line cost of each thread.
   ********************************************************************************

           Cost          %  Operation (s)        Barrier              %    Line
-------------------------------------------------------------------------------
          @cost @cost-rate           @ope        @barrier @barrier-rate      --   @level  @pno  @sno
-------------------------------------------------------------------------------
          @cost @cost-rate           @ope        @barrier @barrier-rate   @line   @name
```

Additional output when the -Impi option is enabled

```
             MPI           %  Communication (s)    Line
  ---------------------------------------------------
            @mpi   @mpi-rate              @comm      --   @level  @pno  @sno
  ---------------------------------------------------
            @mpi   @mpi-rate              @comm   @line   @name
```

Output format of the line cost distribution information (Max thread cost basis)

Always output

```
Lines profile (Max thread cost basis)

   ********************************************************************************
       @level @pno @sno - lines
         Application and Process outputs the max value of the cost of each thread.
         Procedure outputs the max value of the line cost of each thread.
```

```
   ********************************************************************************

 Spawn  Process   Thread     Cost          %  Operation (s)     Barrier             %     Line
-------------------------------------------------------------------------------------------
@spawn @process  @thread    @cost @cost-rate          @ope    @barrier @barrier-rate      --   @level
@pno @sno
-------------------------------------------------------------------------------------------
@spawn @process  @thread    @cost @cost-rate          @ope    @barrier @barrier-rate    @line   @name
```

Additional output when the -Impi option is enabled

```
               MPI          %  Communication (s)     Line
   -----------------------------------------------------
               @mpi  @mpi-rate              @comm       --   @level    @pno @sno
   -----------------------------------------------------
               @mpi  @mpi-rate              @comm    @line   @name
```

Table 2.8 Output items of the line cost distribution information

Output Item	Meaning of Output Item
@level	Information total level (Application, Process, Thread)
@pno	Process or thread number
@sno	spawn number It is taken in the (Spawn @num) format. (Where @num is a numerical value) This item is output only when all the following conditions are met. - The measurement target is the MPI program. - When the fipp command is set with the -Wspawn option enabled, or the fjprof_spawn_dir_name key is set in the program. - When @level is Process. - The process is a dynamically generated process.
@spawn	Spawn number chosen as max cost It is taken in the (Spawn @num) format. (Where @num is a numerical value) This item is output only when all the following conditions are met. If the condition is not met, only "--" is output for this item. - The measurement target is the MPI program. - When the fipp command is set with the -Wspawn option enabled, or the info key "fjprof_spawn_dir_name" is set in the program. - When @level is Process. The process is a dynamically generated process.
@process	Process number chosen as max cost
@thread	Thread number chosen as max cost
@cost	Line cost
@cost-rate	Proportion of the @level or @name cost in the cost at the information total level (%)
@ope	Operation time (s)
@barrier	Waiting cost for synchronization between threads **Note** If it is other than the thread parallel program, only "--" is output for this item.

Output Item	Meaning of Output Item
@barrier-rate	Proportion of the waiting cost for synchronization between threads in the cost at the information total level (%) Note If it is other than the thread parallel program, only "--" is output for this item.
@mpi	MPI cost
@mpi-rate	Proportion of the MPI cost in the cost at the information total level (%)
@comm	Communication time (s)
@line	Line number
@name	Procedure name

2.2.2.5 Call Graph Information

As call graph information, procedure call trace and the cost of each procedure call trace are output.

This information is output when the -Icall option is enabled.

Output format of the call graph information

```
Call graph

     Process @pno  @sno  - Thread @thno
     --------------------+
                         | @rate % <@nest> @name [@cost/@accumulation]
```

Table 2.9 Output items of the call graph information

Output Item	Meaning of Output Item
@pno	Process number
@sno	spawn number It is taken in the (Spawn *@num*) format. (Where *@num* is a numerical value) This item is output only when all the following conditions are met. - The measurement target is the MPI program. - When the fipp command is set with the -Wspawn option enabled, or the info key "fjprof_spawn_dir_name" is set in the program. - The process is a dynamically generated process.
@thno	Thread number
@rate	Proportion of the procedure cost in the cost of the entire thread (%)
@nest	Procedure call nest level
@name	Procedure name
@cost	Procedure cost
@accumulation	Procedure cost including the cost of the called procedure

Note

- If an interrupt occurs due to sampling by the Instant Performance Profiler during the execution of input/output statement processing, call graph information may not be output correctly.

- If "<???>" is output as the nest level in call graph information, it means either of the following:
  - The call trace of the procedure is unknown.
  - The nest level of procedure calls is 128 or higher.
  - Optimization causes call path not to be followed.

2.2.2.6 Source Code Information

Source code information is output with a cost added to each line of the source code. This information is output when the -Isrc option is enabled.

Output format of the source code information

```
Sources profile
  -----> @ file-name
    ---------------------------------------------------------------------------
      Line           Costs
    ---------------------------------------------------------------------------
     @line          @cost    @source-code
```

Table 2.10 Output items of the source code information

Output Item	Meaning of Output Item
@file-name	Source code file name
@line	Line number
@cost	Line cost
@source-code	Source code

Note

When using the -Mnoinlined option, the output of source code information may not display the inlined callee procedures and/or caller procedures.

2.2.3 Details of Profile Result (XML Format)

If the -txml option is specified in the "2.1.5 Outputting Profile Result", the output is in the XML format. The format of XML is described in this section.

2.2.3.1 Structure of XML format

The structure of the XML format output is described; the whole output of the XML format is enclosed by the <profile> element, and the <profile> element consists of the <environment> element and the <information> element.

XML Format

```
<?xml version="1.0" encoding="utf-8"?>                        XML declaration
<profile type="@type" version="@vid" output_version="@oid">   Profiling information
  <environment>                                               The environment information for
    ......                                                              measuring profiling data
  </environment>
  <information item="@item">                                  Performance information
    ......
  </information>
  <information item="@item">                                  Performance information
    ......
  </information>
```

```
  ......
</profile>
```

Output items

Element Name	Overview	Description
profile	Profiling information	This element includes the XML format output of profiler.
environment	Environment information for measuring profiling data	This element includes the following information of the TEXT format(*). This element is output once. - "2.2.2.1 Environment Information for Measuring Profiling Data"
information	Performance information	This element includes the following information of the TEXT format(*). This element is output multiple times. - "2.2.2.2 Statistics Time Information" - "2.2.2.3 CPU Performance Characteristics" - "2.2.2.4 Cost Information" - "2.2.2.5 Call Graph Information"

* Some entries do not match the TEXT and XML formats.

2.2.3.2 Details of XML format output

The following sections describe the elements used in the XML format output.

2.2.3.2.1 Profiling Information <profile>

This element includes the XML format output of profiler.

Element Name	Description
Profile	<profile type="*@type*" version="*@vid*" output_version="*@oid*"> </profile> This element includes the XML format output of profiler. *@type* indicates the kind of profiler, and is fixed to"fipp" *@vid* indicates the version number of the profiler. *@oid* indicates the version number of output format.

2.2.3.2.2 Environment Information for Measuring Profiling Data <environment>

As environment information for measuring profiling data, environment information as of when the profile data was measured is output.

XML format

```
<environment>
  <measured_time unit="date">@date</measured_time>
  <type_of_program program="@program"/>
  <measured_region region="@region"/>
  <coordinate x="@x" y="@y" z="@z"/>
  <spawn id="@id">
    <process id="@id">
      <sampling_interval unit="ms">@interval</sampling_interval>
      <frequency unit="MHz">@frequency</frequency>
    </process>
  </spawn>
</environment>
```

Output items

Element Name	Description
environment	<environment> </environment> This element includes the environment information for measuring profiling data.
measured_time	<measured_time unit="date"> *@date* </measured_time> This element shows the measurement date and time of profile data. *@date* shows the measurement data and time in YYYY-MM-DDThh:mm:ss format.
type_of_program	<type_of_program program="*@program* "/> This element shows the program execution format. *@program* is one of the followings: "SERIAL" : Serial "Thread(AUTO)" : Automatic parallelization only "Thread(OpenMP)" : OpenMP only "Thread(OpenMP+AUTO)" : OpenMP and automatic parallelization "MPI" : MPI only "MPI+Thread(AUTO)" : MPI, Automatic parallelization "MPI+Thread(OpenMP)" : MPI, OpenMP "MPI+Thread(OpenMP+AUTO)" : MPI, OpenMP, and automatic parallelization
measured_region	<measured_region region="*@region* " /> This element shows the kind of measurement region. *@region* is one of the followings: All: Entire program Specified: Specified measurement region
coordinate	<coordinate x="*@x* " y="*@y* " z="*@z* " /> This element shows the logical shape at the time of MPI program execution. The value of x-axis, y-axis, and z-axis is shown in *@x*, *@y*, and *@z*. **Point** Outputs the result only when the data measurement target is an MPI program. The logical shape of dynamically generated processes are not outputted.
spawn	<spawn id="*@id*"> </spawn> This element shows the spawn number. This element will be output multiple times when the multiple spawned processes exist. *@id* indicates the spawn number. If this process is not spawned, *@id* is 0.
process	<process id="*@id*"> </process> This element shows the process number. This element will be output multiple times when the multiple processes exist. *@id* indicates the process number.
sampling_interval	<sampling_interval unit="ms"> *@interval* </sampling_interval> This element shows the sampling interval. The unit is millisecond(unit="ms") *@interval* indicates the sampling interval.
frequency	<frequency unit="MHz"> *@frequency* </frequency>

Element Name	Description
	This element shows the CPU frequency. The unit is MHz(unit="MHz") *@frequency* indicates the CPU frequency. **Note** If any of the following conditions applies, "-" is output as the CPU frequency: - The -Inocpupa option is specified in "2.1.4 Measuring Profile Data" - The -Hmode=user option is specified in "2.1.4 Measuring Profile Data" - The -Sregion option is specified in "2.1.4 Measuring Profile Data" and measurement has never been performed on the measurement region specified in "2.1.1 Adding a Measurement Region Specifying Routine" - The -bmax option is specified in "2.1.5 Outputting Profile Result"

2.2.3.2.3 Performance Information <information>

As performance information, several types of performance information are output. The output of performance information differs with the *@item* attribute. For the detail, refer the description of each <information> element. In this section, the common format in each performance information is described.

XML format

```
<information item="@item">
  <spawn id="@id">
    <process id="@id">
      <thread id="@id">
          ......
      </thread>
    </process>
  </spawn>
</information>
```

Output items

Element Name	Description
information	<information item="*@item*"> </information> This element includes the performance information. For *@item*, one of the following is output according to the performance information to be output. If there is more than one performance information to output, outputs the <information> element multiple times. Depending on the value of *@item*, different elements are output to the blue portion of "XML format". time:"2.2.3.2.4 Statistics Time Information <time>"information is output. cpupa:"2.2.3.2.5 CPU Performance Characteristics <cpupa>" information is output. cost:"2.2.3.2.6 Cost Information <cost> " information is output. call:"2.2.3.2.7 Call Graph Information <call>" information is output.
spawn	<spawn id="*@id*"> </spawn> This element shows the spawn number. This element will be output multiple times when the multiple spawned processes exist. *@id* indicates the spawn number. If this process is not spawned, *@id* is 0.
process	<process id="*@id*"> </process> This element shows the process number. This element will be output multiple times when the multiple processes exist.

Element Name	Description
	@id indicates the process number.
thread	<thread id="*@id*"> </thread> This element shows the thread number. This element will be output multiple times when the multiple threads exist. *@id* indicates the thread number.

2.2.3.2.4 Statistics Time Information <time>

As statistical time information, elapsed time of the program is output. This information is output where the *@item* attribute of the <information> element is "time". For the <information> elements, see "2.2.3.2.3 Performance Information <information>". Only the thread specific information is output and does not output the program specific or the process specific information.

XML format

```
<time>
  <elapsed unit="s">@elapse</elapsed>
  <user unit="s">@user</user>
  <system unit="s">@system</system>
</time>
```

Output items

Element Name	Description
time	<time> </time> This element includes the statistic time information.
elapsed	<elapsed unit="s" > *@elapse* </elapsed> This element shows the elapsed time of each thread in unit of second (unit="s"). *@elapse* indicates the elapsed time of each thread.
user	<user unit="s"> *@user* </user> This element shows the user CPU time of each thread in unit of second (unit="s"). This element appears only if the *@program* attribute of <type_of_program> element is "SERIAL" or "MPI". For the <type_of_program> element, refer "2.2.3.2.2 Environment Information for Measuring Profiling Data <environment>". *@user* indicates the user CPU time of each thread.
system	<system unit="s"> *@system* </system> This element shows the system CPU time of each thread in unit of second (unit="s"). This element appears only if the *@program* attribute of <type_of_program> element is "SERIAL" or "MPI". For the <type_of_program> element, refer "2.2.3.2.2 Environment Information for Measuring Profiling Data <environment>". *@system* indicates the system CPU time of each thread.

2.2.3.2.5 CPU Performance Characteristics <cpupa>

As CPU performance characteristics, information related to CPU performance characteristics, such as memory throughput, number of instructions, and number of operations, is displayed. This information is output where the *@item* attribute of the <information> element is "cpupa". For the <information> elements, see "2.2.3.2.3 Performance Information <information>". Only the thread specific information is output and does not output the program specific or the process specific information. This information is output when the -Icpupa option is enabled.

XML format

```
<cpupa>
  <execution_time unit="s">@execution_time</execution_time>
  <gflops unit="GFLOPS">@gflops</gflops>
  <floating_point_peak_ratio unit="%">@floating_point_peak_ratio</floating_point_peak_ratio>
  <mem_throughput unit="GB/s">@mem_throughput</mem_throughput>
  <mem_throughput_peak_ratio unit="%">@mem_throughput_peak_ratio</mem_throughput_peak_ratio>
  <effective_instruction unit="instructions">@effective_instruction</ffective_instruction>
  <floating_point_operation unit="operations">@floating_point_operation</floating_point_operation>
  <simd_inst_rate unit="%">@simd_inst_rate</simd_inst_rate>
  <sve_operation_rate unit="%">@sve_operation_rate</sve_operation_rate>
  <ipc unit="IPC">@ipc</ipc>
  <gips unit="GIPS">@gips</gips>
</cpupa>
```

Output items

Element Name	Description
cpupa	<cpupa> </cpupa> This element includes the CPU performance characteristics.
execution_time	<execution_time unit="s"> *@execution_time* </execution_time> This element shows the time taken to execute instructions in the measurement target region in unit of second (unit="s"). *@execution_time* indicates the time taken to execute instructions in the measurement target region.
gflops	<gflops unit="GFLOPS"> *@gflops* </gflops> This element shows the count of floating-point operations performed per second in unit of GFLOPS (unit="GFLOPS"). *@gflops* indicates the count of floating-point operations performed per second. Note The GFLOPS value is calculated by assuming that all elements are active. Therefore, in the case of a program with many inactive elements, a higher value than the original GFLOPS value is output.
floating_point_peak_ratio	<floating_point_peak_ratio unit="%"> *@floating_point_peak_ratio* </floating_point_peak_ratio> This element shows the ratio of the measured value to the theoretical value for floating-point operation performance in unit of percentage (unit="%"). *@floating_point_peak_ratio* indicates the ratio of the measured value to the theoretical value for floating-point operation performance. Note The theoretical value for floating-point operation performance is calculated by assuming that double precision operations are performed. Therefore, in the case of single or half precision, a value 2 to 4 times higher than the actual percentage is output.
mem_throughput	<mem_throughput unit="GB/s"> *@mem_throughput* </mem_throughput> This element shows the memory throughput in unit of GB per second (GB/s). *@mem_throughput* indicates the memory throughput.
mem_throughput_peak_ratio	<mem_throughput_peak_ratio unit="%"> *@mem_throughput_peak_ratio* mem_throughput_peak_ratio>

Element Name	Description
	This element shows the Ratio of the measured value to the theoretical value for memory throughput. The unit is %(unit="%") *@mem_throughput_peak_ratio* indicates the measured value to the theoretical value for memory throughput.
effective_instruction	<effective_instruction unit="instructions"> *@effective_instruction* </effective_instruction> This element shows the total number of instructions executed in unit of instructions (unit="instructions"). *@effective_instruction* indicates the total number of instructions executed. **Note** The total number of instructions executed does not include MOVPRFX instructions.
floating_point_operation	<floating_point_operation unit="operations"> *@floating_point_operation* </ floating_point_operation> This element shows the total number of floating-point operations executed in unit of operations (unit="operations"). *@floating_point_operation* indicates the total number of floating-point operations executed.
simd_inst_rate	<simd_inst_rate unit="%"> *@simd_inst_rate* </simd_inst_rate> This element shows the ratio of the number of SIMD instructions to the total number of instructions executed in unit of percentage (unit="%"). *@simd_inst_rate* indicates the ratio of the number of SIMD instructions to the total number of instructions executed.
sve_operation_rate	<sve_operation_rate unit="%"> *@sve_operation_rate* </sve_operation_rate> This element shows the ratio of the number of SVE operations to the total number of floating-point operations executed in unit of percentage (unit="%"). *@sve_operation_rate* indicates the ratio of the number of SVE operations to the total number of floating-point operations executed.
ipc	<ipc unit="IPC"> *@ipc* </ipc> This element shows the number of instructions executed per cycle in unit of IPC (unit="IPC"). *@ipc* indicates the number of instructions executed per cycle.
gips	<gips unit="GIPS"> @gips </gips> This element shows the number of instructions executed per second in unit of GIPS (unit="GIPS"). *@gips* indicates the number of instructions executed per second.

2.2.3.2.6 Cost Information <cost>

The cost information consists of the following information:

- Procedure cost distribution information
- Loop cost distribution information
- Line cost distribution information

This information is output where the *@item* attribute of the <information> element is "cost". For the <information> elements, see "2.2.3.2.3 Performance Information <information> ". Only the thread specific information is output and does not output the program specific or the process specific information.

XML format

```
<cost>
  <procedures>
    <procedure func="@func" start="@start" end="@end">
      <procedure_base_cost unit="hits">@procedure_base_cost</procedure_base_cost>
      <procedure_barrier_cost unit="hits">@procedure_barrier_cost</procedure_barrier_cost>
      <procedure_mpi_cost unit="hits">@procedure_mpi_cost</procedure_mpi_cost>
    </procedure>
      ......
  </procedures>
  <loops>
    <loop func="@func" start="@start" end="@end" nest="@nest" kind="@kind" exec="@exec">
      <loop_base_cost unit="hits">@loop_base_cost</loop_base_cost>
      <loop_barrier_cost unit="hits">@loop_barrier_cost</loop_barrier_cost>
      <loop_mpi_cost unit="hits">@loop_mpi_cost</loop_mpi_cost>
    </loop>
      ......
  </loops>
  <lines>
    <line func="@func" line_number="@line_number">
      <line_base_cost unit="hits">@line_base_cost</line_base_cost>
      <line_barrier_cost unit="hits">@line_barrier_cost</line_barrier_cost>
      <line_mpi_cost unit="hits">@line_mpi_cost</line_mpi_cost>
    </line>
      ......
  </lines>
</cost>
```

Output items

Element Name	Description
cost	<cost> </cost> This element includes the cost information.
procedures	<procedures> </procedures> This element includes the group of procedure cost distribution information. This element and the sibling element will be output only if the procedure cost information exists.
procedure	<procedure func="*@func*" start="*@start*" end="*@end*"> </procedure> This element shows the procedure cost distribution information. If there are multiple procedures to be output, this element is output multiple times. *@func* indicates the procedure name. *@start* indicates the procedure start line number. This attribute is output only when the start line number of the subject procedure is available. *@end* indicates the procedure end line number. This attribute is output only when the end line number of the subject procedure is available. See In the case of a thread parallel program, the parallelized part is output as information on the generated procedure. An identifier corresponding to the type of the generated procedure is appended to the name of the generated procedure. Refer "Table 2.6 Names of Generated Procedures for Thread Parallel Programs" for details.
procedure_base_cost	<procedure_base_cost unit="hits"> *@procedure_base_cost* </procedure_base_cost> This element shows the procedure cost.

Element Name	Description
	@procedure_base_cost indicates the procedure cost.
procedure_barrier_cost	<procedure_barrier_cost unit="hits"> *@procedure_barrier_cost* </procedure_barrier_cost> This element shows the waiting cost of procedure for synchronization between threads. This element is output only in the case of a thread parallel program. *@procedure_barrier_cost* indicates the waiting cost of procedure for synchronization between threads.
procedure_mpi_cost	<procedure_mpi_cost unit="hits"> @procedure_mpi_cost </procedure_mpi_cost> This element shows the MPI cost of procedure. This element is output only when the -Impi option is enabled to output the profile result. *@procedure_mpi_cost* indicates the MPI cost of procedure.
loops	<loops> </loops> This element includes the group of loop cost distribution information. This element and the sibling element will be output only if the loop cost information exists.

loop	<loop func="*@func*" start="*@start*" end="*@end*" nest="*@nest*" kind="*@kind*" exec="*@exec*"> </loop>
	This element shows the loop cost distribution information. If there are multiple loops to be output, this element is output multiple times.
	@func indicates the procedure name.
	@start indicates the loop start line number.
	@end indicates the loop end line number.
	@nest indicates the nest level.
	@kind indicates the loop type. One of the following types is output:
	`DO, WHILE, UNTIL, ARRAY, FOR, GOTO, OTHER`
	@exec indicates the loop compilation type. One of the following types is output:
	`SERIAL, OpenMP, AUTO`

loop_base_cost	<loop_base_cost unit="hits"> *@loop_base_cost* </loop_base_cost> This element shows the loop cost. *@loop_base_cost* indicates the loop cost.
loop_barrier_cost	<loop_barrier_cost unit="hits"> *@loop_barrier_cost* </loop_barrier_cost> This element shows the waiting cost of loop for synchronization between threads. This element is output only in the case of a thread parallel program. *@loop_barrier_cost* indicates the waiting cost of loop for synchronization between threads.
loop_mpi_cost	<loop_mpi_cost unit="hits"> *@loop_mpi_cost* </loop_mpi_cost> This element shows the MPI cost of loop. This element is output only when the -Impi option is enabled to output the profile result. *@loop_mpi_cost* indicates the MPI cost of loop.
lines	<lines> </lines> This element includes the group of line cost distribution information. This element and the sibling element will be output only if the line cost information exists.
line	<line func="*@func*" line_number="*@line_number*"> </line> This element shows the line cost distribution information. If there are multiple lines to be output, this element is output multiple times.

Element Name	Description
	@func indicates the procedure name. *@line_number* indicates the line number.
line_base_cost	<line_base_cost unit="hits"> *@line_base_cost* </line_base_cost> This element shows the line cost. *@line_base_cost* indicates the line cost.
line_barrier_cost	<line_barrier_cost unit="hits"> *@line_barrier_cost* </line_barrier_cost> This element shows the waiting cost of line for synchronization between threads. This element is output only in the case of a thread parallel program. *@line_barrier_cost* indicates the waiting cost of line for synchronization between threads.
line_mpi_cost	<line_mpi_cost unit="hits"> *@line_mpi_cost* </line_mpi_cost> This element shows the MPI cost of line. This element is output only when the -Impi option is enabled to output the profile result. *@line_mpi_cost* indicates the MPI cost of line.

2.2.3.2.7 Call Graph Information <call>

As call graph information, procedure call trace and the cost of each procedure call trace are output. This information is output where the *@item* attribute of the <information> element is "call". For the <information> elements, see "2.2.3.2.3 Performance Information <information>". This information is output when the -Icall option is enabled.

XML format

```
<call>
  <frames>
    <frame func="@func">
      <procedure_cost unit="hits">@procedure_cost</procedure_cost>
      <frame func="@func">
        <procedure_cost unit="hits">@procedure_cost</procedure_cost>
          ......
      </frame>
    </frame>
  </frames>
</call>
```

Output items

Element Name	Description
call	<call> </call> This element includes the call graph information.
frames	<frames> </frames> This element includes the group of call graph frame information.
frame	<frame func="*@func*" unknown="true"> </frame> This element shows the call graph frame. This element will be nested when multiple procedures are nested in the call graph. This element will be parallel when multiple procedures are parallel in the call graph. The name at *@func* is the procedure name. If the nest level is displayed <???> on the TEXT format output, the unknown="true" attribute is output.
procedure_cost	<procedure_cost unit="hits"> *@procedure_cost* </procedure_cost> This element shows the procedure cost.

Element Name	Description
	The number at *@procedure_cost* is the procedure cost.

Chapter 3 Advanced Performance Profiler

This chapter describes the Advanced Performance Profiler.

The Advanced Performance Profiler measures and outputs the execution performance information of the specified region of an application. It consists of two commands: the fapp command, which measures profile data, and the fapppx command, which outputs profile result from measured data. The Advanced Performance Profiler measures and outputs the following information:

Statistical time information

As statistical time information, the number of calls in the measurement target region, elapsed time, user CPU time, system CPU time breakdown, etc. are output.

MPI communication cost information

As MPI communication cost information, the number of executions of MPI functions in the measurement target region, message length, and average, maximum, and minimum execution time and waiting time are output.

CPU performance analysis information

As CPU performance analysis information, CPU performance characteristics at the time of application execution in the measurement target region are output. This information is used for the "Chapter 4 CPU Performance Analysis Report"

Note

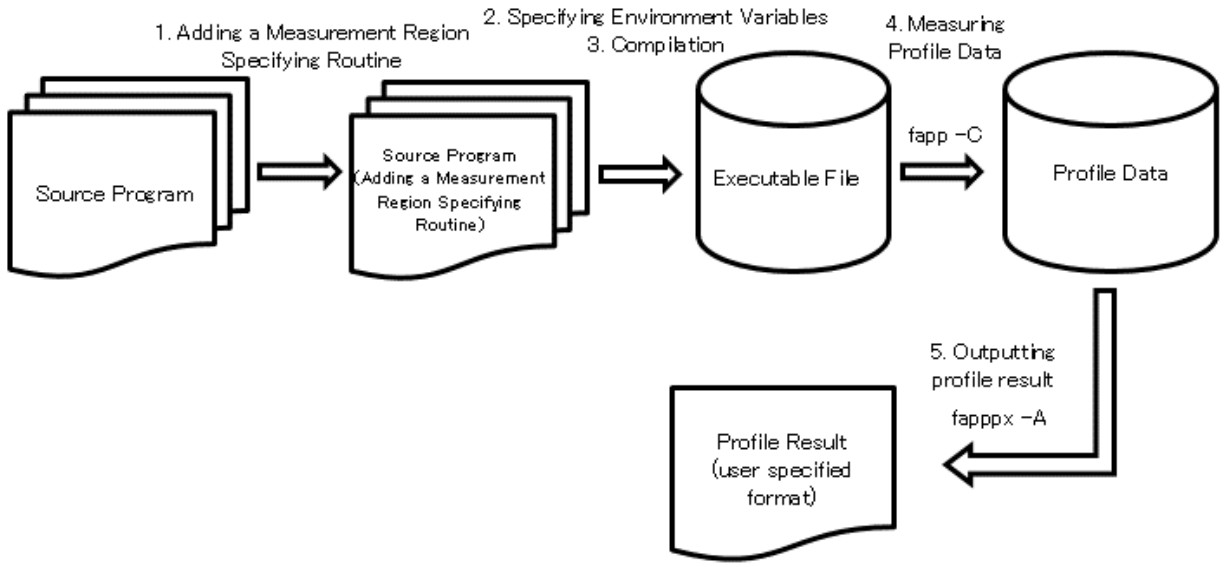
The fapp command can be used both to measure the profile data and to output the profile results. In this User's Guide, we use this command to measure the profile data.

For notes on using the Advanced Performance Profiler, see "5.1 Notes Common to Profilers" and "5.3 Notes on the Advanced Performance Profiler".

3.1 Procedure for Using the Advanced Performance Profiler

This section provides the procedure for using the Advanced Performance Profiler.

Figure 3.1 Procedure for using the Advanced Performance Profiler



Information

The Advanced Performance Profiler is also used to measure and output data for the CPU Performance Analyzer. See "4.1 Procedure for Using the CPU Performance Analysis Report" for an example of how to run a CPU Performance Analysis report.

The following describes each operation in detail.

3.1.1 Adding a Measurement Region Specifying Routine

To the source code, add the measurement region specifying routine required for specifying the region (starting and stopping positions of measurement) from which profile data is measured.

The following describes measurement region specifying routines for the Advanced Performance Profiler in detail.

3.1.1.1 fapp_start / fapp_stop Subroutines (Fortran)

Format

CALL fapp_start(*name*, *number*, *level*)

CALL fapp_stop(*name*, *number*, *level*)

Function Description

These subroutines start or stop profile data measurement by the Advanced Performance Profiler.

fapp_start(*name*, *number*, *level*)

This subroutine starts profile data measurement by the Advanced Performance Profiler.
A combination of the argument *name* (group name) and the argument *number* (detail number) is used as the measurement region name. Different measurement region names allow measurement to be done in parallel. The argument *level* has a meaning to the -L option of the fapp command. It enables only the region satisfying "-L option argument *level*" >= "argument *level*" as a measurement target. For details on the -L option, see "fapp command options"

fapp_stop(*name*, *number*, *level*)

This subroutine stops profile data measurement by the Advanced Performance Profiler.
A combination of the argument *name* (group name) and the argument *number* (detail number) is used as the measurement region name. Different measurement region names allow measurement to be done in parallel. The argument *level* has a meaning to the -L option of the fapp command. It enables only the region satisfying "-L option argument *level*" >= "argument *level*" as a measurement target. For details on the -L option, see "fapp command options"

Argument

name

This argument shows the group name, which is handled as a measurement region name in combination with the argument *number* (detail number).

Basic character type scalar. The following characters can be used for the argument *name*:

- Alphabetical characters

  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
  a b c d e f g h i j k l m n o p q r s t u v w x y z

- Numerical characters

  0 1 2 3 4 5 6 7 8 9

- Symbol

  _ (underscore)

number

This argument shows the detail number, which is handled as a measurement region name in combination with the argument *name* (group name).

Four-byte integer type.

level

This argument shows the start level, which is used in the -L option of the fapp command.

Four-byte integer type. However, it must be an integer from 0 to 2,147,483,647. If an incorrect value is specified, a warning message is output and this routine is ignored.

Example

Example of use of measurement region specifying routines

```
program main
...
call fapp_start("foo",1,0)     ! Start measurement for the measurement region name"foo1"
do i=1,10000
    ...
    call fapp_start("bar",1,0) ! Start measurement for the measurement region name"bar1"
    do j=1,10000
        ...
    end do
    call fapp_stop("bar",1,0)  ! End measurement for the measurement region name "bar1"
end do
call fapp_stop("foo",1,0)      ! End measurement for the measurement region name "foo1"
end program main
```

Note

- The Advanced Performance Profiler measures profile data for each measurement region name. To call a subroutine with the same measurement region name multiple times, be sure to call it in the order from fapp_start to fapp_stop. If fapp_start is called again before fapp_stop is called, or fapp_stop is called before fapp_start is called, a warning message is output and the call is ignored. If measurement region names are different, there is no problem if fapp_start or fapp_stop is successively called. If the process ends without calling fapp_stop, the profile data of the region is not measured.
- If measurement is performed multiple times for the same measurement region name, all the measurement results are totaled.
- Specify the same value of the argument *level* in fapp_start and fapp_stop. If a different value is specified, an unintended result may occur, depending on the specification by the -L option of the fapp command.
- When measuring a thread parallel program, add the measurement region specifying routines so that the measurement region includes the entire thread parallel region (including the thread parallel region generated by automatic parallelization) for each thread parallel interval. Check the compilation information for the state of parallelization. The operation is not guaranteed for out-of-specification use.
- Do not specify the following combination of the measurement region and detail number. This combination is reserved for making the entire program a target.

```
  name : "all"
number : 0
```

  However, if the -Hmethod=fast option is specified when measuring with the fapp command, this region is not measured. For about the -Hmethod=fast option, see "3.1.4 Measuring Profile Data".

- In the case of a process parallel program, call a subroutine with the same measurement region name in all the processes that are targets of measurement. The profile data of processes where this call is not performed is not measured.

Example

Example for making all processes targets of measurement (starting measurement before calling the mpi_init subroutine)

```
call fapp_start("foo",1,0)  ! Start measurement
call mpi_init(err)
```

```
  ...
call mpi_finalize(err)
call fapp_stop("foo",1,0)   ! Stop measurement
```

Example for making all processes targets of measurement (starting measurement immediately after calling the mpi_init subroutine)

```
call mpi_init(err)
call fapp_start("foo",1,0)  ! Start measurement
  ...
call fapp_stop("foo",1,0)   ! Stop measurement
call mpi_finalize(err)
```

Example for making only process 0 the target of measurement

```
call mpi_init(err)
call mpi_comm_rank(mpi_comm_world,rank,err)
if(rank==0) then
    call fapp_start("foo",1,0)   ! Start measurement on process 0 only
end if
    ...
if(rank==0) then
    call fapp_stop("foo",1,0)    ! Stop measurement on process 0 only
end if
call mpi_finalize(err)
```

- When the compiler option -mldefault=cdecl is valid to compile a Fortran program, change the name of the measurement region specifying routine as follows.

Before	After
fapp_start	fapp_start_
fapp_stop	fapp_stop_

- When the compiler option -AU is valid, the name of the measurement region specifying routine must be entered in lowercase characters.

3.1.1.2 fapp_start function / fapp_stop Function (C language and C++)

Format

#include "fj_tool/fapp.h"

void fapp_start(const char **name*, int *number*, int *level*);
void fapp_stop(const char **name*, int *number*, int *level*);

Function Description

These subroutines start or stop profile data measurement by the Advanced Performance Profiler.

fapp_start(const char **name*, int *number*, int *level*)

This subroutine starts profile data measurement by the Advanced Performance Profiler.
A combination of the argument name (group name) and the argument number (detail number) is used as the measurement region name. Different measurement region names allow measurement to be done in parallel. The argument level has a meaning to the -L option of the fapp command. It enables only the region satisfying "-L option argument level" >= "argument level" as a measurement target. For details on the -L option, see "fapp command options"

fapp_stop(const char **name*, int *number*, int *level*)

This subroutine stops profile data measurement by the Advanced Performance Profiler.
A combination of the argument name (group name) and the argument number (detail number) is used as the measurement region name. Different measurement region names allow measurement to be done in parallel. The argument level has a meaning to the -L option of the fapp command. It enables only the region satisfying "-L option argument level" >= "argument level" as a measurement target. For details on the -L option, see "fapp command options"

Argument

name

This argument shows the group name, which is handled as a measurement region name in combination with the argument number (detail number).

Basic character type scalar. The following characters can be used for the argument name:

- Alphabetical characters

  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
  a b c d e f g h i j k l m n o p q r s t u v w x y z

- Numerical characters

  0 1 2 3 4 5 6 7 8 9

- Symbol

  _ (underscore)

number

This argument shows the detail number, which is handled as a measurement region name in combination with the argument name (group name).

int type.

level

This argument shows the start level, which is used in the -L option of the fapp command.

int type. However, it must be an integer from 0 to 2,147,483,647. If an incorrect value is specified, a warning message is output and this routine is ignored.

Example

Example of use of measurement region specifying routines

```
#include "fj_tool/fapp.h" // Include the header file.
...

int main(void)
{
    int i,j;
    fapp_start("foo",1,0);   // Start measurement for the measurement region name "foo1"
    for(i=0;i<10000;i++){
        ...
        fapp_start("bar",1,0); // Start measurement for the measurement region name "bar1"
        for(j=0;j<10000;j++){
            ...
        }
        fapp_stop("bar",1,0);  // End measurement for the measurement region name "bar1"
    }
    fapp_stop("foo",1,0);    // End measurement for the measurement region name "foo1"
    return 0;
}
```

Note

- The Advanced Performance Profiler measures profile data for each measurement region name. To call a function with the same measurement region name multiple times, be sure to call it in the order from fapp_start to fapp_stop. If fapp_start is called again before fapp_stop is called, or fapp_stop is called before fapp_start is called, a warning message is output and the call is ignored. If measurement region names are different, there is no problem if fapp_start or fapp_stop is successively called. If the process ends without calling fapp_stop, the profile data of the region is not measured.

- If measurement is performed multiple times for the same measurement region name, all the measurement results are totaled.
- Specify the same value of the argument *level* in fapp_start and fapp_stop. If a different value is specified, an unintended result may occur, depending on the specification by the -L option of the fapp command.
- When measuring a thread parallel program, add the measurement region specifying routines so that the measurement region includes the entire thread parallel region (including the thread parallel region generated by automatic parallelization) for each thread parallel interval. Check the compilation information for the state of parallelization. The operation is not guaranteed for out-of-specification use.
- Do not specify the following combination of the measurement region and detail number. This combination is reserved for making the entire program a target.

```
  name : "all"
number : 0
```

  However, if the -Hmethod=fast option is specified when measuring with the fapp command, this region is not measured. For about the -Hmethod=fast option, see "3.1.4 Measuring Profile Data".

- In the case of a process parallel program, call a function with the same measurement region name in all the processes that are targets of measurement. The profile data of processes where this call is not performed is not measured.

Example

Example for making all processes targets of measurement (starting measurement before calling the mpi_init subroutine)

```
fapp_start("foo",1,0);   // Start measurement
MPI_Init(&argc, &argv);
  ...
MPI_Finalize();
fapp_stop("foo",1,0);    // Stop measurement
```

Example for making all processes targets of measurement (starting measurement immediately after calling the mpi_init subroutine)

```
MPI_Init(&argc, &argv);
fapp_start("foo",1,0);   // Start measurement
  ...
fapp_stop("foo",1,0);    // Stop measurement
MPI_Finalize();
```

Example for making only process 0 the target of measurement

```
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
if(rank==0){
    fapp_start("foo",1,0);  // Start measurement on process 0 only
}
    ...
if(rank==0){
    fapp_stop("foo",1,0);   // Stop measurement on process 0 only
}
MPI_Finalize();
```

3.1.2 Specifying Environment Variables

Specify environment variables required when using the Profiler. For details, see "2.1.2 Specifying Environment Variables"

3.1.3 Compilation

Compile a program. For detail, see "2.1.3 Compilation".

3.1.4 Measuring Profile Data

Measure data by using the fapp command. Perform this operation from a compute node.

Note

- If any of the following is performed on profile data measured by the fapp command, the operation is not guaranteed:
  - Editing profile data
  - Adding, deleting, or renaming profile data
- If the program is interrupted while profile data is being measured, incomplete profile data may remain.
- When using the fapp command, specify a value that is equivalent to "TRUE" in the environment variable "FLIB_FASTOMP". If do not specified, the fapp command does not operate correctly. For details on the environment variable "FLIB_FASTOMP", see the "Fortran User's Guide", "C User's Guide", or "C++ User's Guide".
- You can also specify the -A option for the fapp command. However, if you specify the -A option for the fapp command, it is treated as a fapppx command of the "3.1.5 Outputting Profile Result". Therefore, you should work with it differently. For details, see "3.1.5 Outputting Profile Result". You cannot specify the -A option together with the -C option for the fapp command. If you specify both options, an error message is output and the program is aborted.

fapp command syntax

```
fapp -C -d profile_data [ -I{{cpupa|nocpupa}|{cputime|nocputime}|{mpi|nompi}} ]
[ -H{event=event|event_raw=event_raw}[,method={fast|normal},mode={all|user}] ]
[ -L level ] [ -W{spawn|nospawn} ] exec-file [ exec_option ... ]
```

fapp command options

Point

- If the description of an option contains a restriction such as "you cannot..." or "it must be..." and you violate it, an error message is output and the execution is terminated.
- If you specify multiple conflicting options, the last specified option is enabled. For example, if you specify the -I{cpupa|nocpupa} option, which specifies whether to measure CPU performance analysis information, in the order of "-Icpupa -Inocpupa", the -Inocpupa option is enabled.
- To use the CPU performance analysis function, use the -Icpupa and -Hevent=*event* options. For details, see "4.1.4 Measuring Profile Data".

-C

This option specifies to measure profile data. The option cannot be omitted. If you do not specify the option, an error message is output and the collecting command is terminated.

-d *profile_data*

This option specifies the directory where profile data is to be stored. The option cannot be omitted. If you do not specify the option, an error message is output and the collecting command is terminated.
profile_data cannot be omitted. *profile_data* specifies a relative or absolute path as the name of the directory where profile data is to be stored. If the specified directory exists, it must be an empty directory. If the specified directory does not exist, a new directory is created. To specify a directory name beginning with "-" in *profile_data*, specify its absolute path or a relative path that contains the current directory ("./"). To analyze a program that moves the current directory during execution, specify its absolute path in *profile_data*.

-I{{cpupa|nocpupa}|{cputime|nocputiime}|{mpi|nompi}}

This option specifies items to be measured with the Advanced Performance Profiler. You can specify the -I option with multiple sub options separated with comma (,). For example, you can specify such as "-Inocpupa,mpi". Specification when you omit the -I{cpupa|nocpupa} option varies depending on the -H option. If the -H option is specified, the -Icpupa option is enabled. If the -H option is not

specified, the -Inocpupa option is enabled. Specification when you omit the -I{cputime|nocputime} option varies depending on the -I{cpupa|nocpupa} option. If the -Icpupa option is enabled, the -Inocputime option is enabled. If the -Inocpupa option is enabled, the -Icputime option is enabled. You can specify -Inocputime option only if the -Icpupa option is enabled. When the -Inocpupa option is enabled and if the -Inocputime option is specified, an error message is output and the collecting command is terminated. The specification differs depending on the type of the program if the -I{mpi|nompi} option is omitted. When working with an MPI program, the -Impi option is enabled. When working with a non-MPI program, the -Inompi option is enabled. When the -Icpupa option is enabled and the -H option is not specified, the -Hevent=statistics,method=normal,mode=all option is enabled.

cpupa

This argument specifies to measure CPU performance analysis information.

nocpupa

This argument specifies not to measure CPU performance analysis information.

cputime

This argument specifies to measure User CPU time and System CPU time.

nocputime

This argument specifies to does not measure CPU time and system CPU time, shortens the time to measure the CPU performance analysis information.

mpi

This argument specifies to measure MPI communication cost information. If you specify the argument for a non-MPI program, an error message is output and the collecting command is terminated.

nompi

This argument specifies not to measure MPI communication cost information.

-H{event=*event* |event_raw=*event_raw* }[,method={fast|normal},mode={all|user}]

This option specifies items for measurement of CPU performance analysis information. If you specify the Inocpupa option, a warning message is output and this option is disabled. Be sure to specify either of the suboptions event=*event* and event_raw=*event_raw*. If you specify the suboptions event=*event* and event_raw=*event_raw* together, the last specified suboption is enabled. If the suboption mode= is omitted, mode=all is enabled. If the suboption method= is omitted, method=normal is enabled.

event=*event*

Measure the information used in the CPU Performance Analysis Report. *event* cannot be omitted. Specify one of the following in *event*. pa1 is equivalent to statistics.

{ pa1 | pa2 | pa3 | pa4 | pa5 | pa6 | pa7 | pa8 | pa9 | pa10 | pa11 | pa12 | pa13 | pa14 | pa15 | pa16 | pa17 | statistics}

event_raw=*event_raw*

This suboption specifies the event number of PMU event information to measure CPU performance analysis information. *event_raw* cannot be omitted. In *event_raw*, specify an event number corresponding to a CPU in decimal or hexadecimal notation (0x or 0X). You can specify up to eight *event_raw* values by separating them with commas (,).

method=fast

This suboption specifies a measurement method for CPU performance analysis information. If you specify this suboption, high accuracy measurement is performed for CPU performance analysis information by the method of directly measuring hardware information.

method=normal

This suboption specifies a measurement method for CPU performance analysis information. If you specify this suboption, CPU performance analysis information is measured by the method of measurement via the OS. If you specify this suboption, you cannot specify the same event number for -Hevent_raw=*event_raw* option more than once.

mode=all

This suboption specifies a measurement mode for CPU performance analysis information. If you specify this suboption, performance is measured in kernel and user modes.

mode=user

This suboption specifies a measurement mode for CPU performance analysis information. If you specify this suboption, performance is measured in user mode.

-L *level*

This option specifies the start level for the measurement target. *level* specifies an integer from 0 to 2,147,483,647. If you specify a value outside the range in *level*, a warning message is output and the -L 0 option is enabled. The option has a meaning to the third argument *level* of the measurement region specifying routine. It enables only the region satisfying "*level*" >= "third argument *level* of measurement region specifying routine" as a measurement target. If this option is omitted, the -L 0 option is enabled.

-W{spawn|nospawn}

This option specifies how to measure the dynamically generated process. Specification when you omit the -W{spawn|nospawn} option varies depending on the type of the program. When working with the MPI program, the -Wspawn option is enabled. When working with the non-MPI program, the -Wnospawn option is enabled.

spawn

This argument specifies to measure the statistic information of the dynamically generated process. If you specify the info key "fjprof_spawn_dir_name" for the specific dynamically generated process, the profile data of the dynamically generated process with the info key is stored in the directory that you specified for info key "fjprof_spawn_dir_name". If you do not specify the info key "fjprof_spawn_dir_name", the profile data of the dynamically generated process is stored in the directory which you specified with the -d option. If you specify this argument to the non-MPI program, an error message is output and the collecting command is terminated.

nospawn

This argument specifies not to measure the statistic information of the dynamically generated process. However, if you specify the info key "fjprof_spawn_dir_name" to the specific dynamically generated process, measurement is performed for that process with the info key, and the measured profile data is stored in the directory which you specified for the info key "fjprof_spawn_dir_name".

exec-file [*exec_option* ...]

This option specifies the execution file that is the target of profile data measurement and options for the file. In the case of an MPI program, make a specification from mpiexec. To specify an execution file that begins with "-" in *exec-file*, specify its relative path that contains the current directory ("./") or absolute path. A shell script cannot be specified in *exec-file*. The character string following the execution file name (*exec_option* ...) is considered an option for the execution file.

Example

Example of measuring CPU performance analysis information (statistics) with the fapp command

```
fapp -C -d ./tmp -Icpupa -Hevent=statistics ./a.out
```

3.1.5 Outputting Profile Result

Output the result of profile data measured with the fapp command. You can also output an input file used for the CPU Performance Analysis Report. For this operation, use a different command depending on the node to be used.

When using a login node

Use the fapppx command.

compute node

Use the fapp command.

fapppx command or fapp command syntax

```
{fapppx|fapp} -A [ -I{{cpupa|nocpupa}|{mpi|nompi}} ] [ -o outfile ] [ -pp_no ] [ -t{csv|text|xml} ]
[ -d ] profile_data
```

fapppx command or fapp command options

Point

- If the description of an option contains a restriction such as "you cannot..." or "it must be..." and you violate it, an error message is output and the program is aborted.
- If you specify multiple conflicting options, the last specified option is enabled. For example, if you specify the -I{cpupa|nocpupa} option, which specifies profile result items to be output, in the order of "-Icpupa,nocpupa", the -Inocpupa option is enabled.
- To use the CPU performance analysis function, specify the -tcsv option.

-A

This option specifies to output the profile result. The option cannot be omitted. If you do not specify the option, an error message is output and the analyzing command is terminated.

-I{{cpupa|nocpupa}|{mpi|nompi}}

This option specifies the item(s) to the output as the profile result. You can specify the -I option with multiple sub options separated with comma (,). For example, you can specify such as "-Inocpupa,mpi". If the -I{cpupa|nocpupa} option is omitted, the -Inocpupa option is enabled. The specification when the -I{mpi|nompi} option is omitted differs depending on the type of the program of which the profile data is to be measured. If the measurement target of the profile data is an MPI program, the Impi option is enabled. If the measurement target of the profile data is a non-MPI program, the -Inompi option is enabled. If you specify the options of -Icpupa,mpi, the relevant items must be measured by executing the fapp command. If you specify the information to be output which is not measured, the options specified are ignored.

cpupa

This argument specifies to output CPU performance analysis information. The behavior of specifying this option and the -ttext option at the same time depends on the measured profile data. Outputs CPU Performance Analysis Information in text format when you measure profile data with the -Hevent=pa1 or -Hevent=statistics options. Otherwise, CPU Performance Analysis Report will not be output in text format. If this option and the -tcsv option or -txml option are specified at the same time, CPU Performance Analysis Information is output regardless of the -H option.

nocpupa

This argument specifies not to output CPU performance analysis information.

mpi

This argument specifies to output MPI communication cost information.

nompi

This argument specifies not to output MPI communication cost information.

-o *outfile*

This option specifies the output destination for the profile result. *outfile* specifies a relative or absolute path as the name of the output destination file or stdout. If this option is omitted, the -ostdout option is enabled. If you specify stdout in *outfile*, the profile result is output to the standard output. To specify a file name beginning with "-" in *outfile*, specify its absolute path or a relative path that contains the current directory ("./").

-p*p_no*

Specify the process to be output to the profile result. For *p_no*, specify one or more from those of *N*@*M* , input=*n* , limit=*m* , and all. If you omit this option, the value taken in varies depending on the specification of the -t{csv|text|xml} option. If the -ttext option is enabled, the "-pinput=0,limit=16" option is enabled. If the -tcsv option or -txml option is enabled, the -pall option is enabled. *p_no* cannot be omitted. For the -p option, you can specify more than one *p_no* by separating them with a comma (,). For example, you can specify such as "-p3,5,limit=10".

N[@*M*] ...

This suboption specifies to output, at the beginning, the information of the process number specified in *N*. If you specify the process number *N* that belongs to the spawn number *M*, specify in the format like "*N*@*M*". If the information of the process number specified

with *N*[@*M*] does not exist, ignore the specification. You can specify more than one *N*[@*M*]. If you specify more than one *N*[@*M*], the result is output in the order of your specification. [@*M*] is only valid with the -ttext option.

input=*n*

This suboption specifies to read the information for the top *n* processes at cost. Although processing becomes faster because the number of files to read decreases, the information of processes that is not read, is not included in the denominator to perform ratio calculation. If you specify 0 or a value exceeding the number of processes in *n*, the information of all processes is read. If this suboption is omitted, input=0 is enabled. If you specify input=*n* and all together, the suboption input=*n* is enabled, regardless of the order in which options are specified. The suboptions input=*n* and limit=*m* can be specified together.

limit=*m*

This suboption specifies to output the information for the top *m* processes at cost. If the -ttext option is enabled, output processes in the order of cost from highest. The information of processes that is not output is included in the denominator to perform ratio calculation. If you specify 0 or a value exceeding the number of processes in *m*, the information of all processes is output. If you omit this suboption, the value taken in varies depending on the specification of the -t{csv|text|xml} option. If the -ttext option is enabled, limit=16 is enabled. If the -tcsv option or -txml option is enabled, limit=0 is enabled.

all

This suboption specifies to read and output the information for all processes. If the -ttext option is enabled, output processes in the order of cost from highest. This is the same as when the -pinput=0,limit=0 option is specified. If you do not specify either the suboption input=*n* or limit=*m*, this suboption is enabled.

-t{csv|text|xml}

This option specifies an output format for profile result. If this option is omitted, -ttext is enabled.

csv

This option specifies to output the profile result in CSV format.

text

This option specifies to output the profile result in TEXT format.

xml

This option specifies to output the profile result in XML format.

-d *profile_data*

profile_data specifies a relative or absolute path as the name of the directory where profile data is stored. This option cannot be omitted. However, as long as you specify *profile_data* at the end of an array of options, "-d" can be omitted. To specify a directory name beginning with "-" in *profile_data*, specify its absolute path or relative path that contains the current directory ("./").

Example

Example of outputting profile results in TEXT format

```
fapppx -A -ttext -o tmp.txt -d ./tmp
```

3.2 Profile Result

This section describes the contents of profile result output by the fapppx command or fapp command.

3.2.1 Overview of Profile Result

Profile result consists of the following statistical information. You can use the -I option of the fapppx command or fapp command to control the output of each piece of the information. For details on the -I option, see "3.1.5 Outputting Profile Result" In TEXT format, the information is output in the following order:

- Environment information for measuring profiling data
- Statistical time information

- MPI communication cost information
- CPU performance analysis information

For details, see "3.2.2 Detail of Profile Result (TEXT Format)" or "3.2.3 Detail of Profile Result (XML Format)" and its subsections.

3.2.2 Detail of Profile Result (TEXT Format)

If you specify the "3.1.5 Outputting Profile Result" with the -ttext option, the result is output in the TEXT format. The followings are applied to the TEXT format.

3.2.2.1 Environment Information for Measurement Profiling Data

As environment information for measuring profiling data, environment information as of when the profile data was measured is output.

Output format of the environment information for measuring profiling data

```
-----------------------------------------------------------------------------------------
Fujitsu Advanced Performance Profiler Version @vl
   Measured time                 : @date
   CPU frequency                 : Process      @pno  @frequency (MHz)  @sno
   Type of program               : @type
   Virtual coordinate            : (@x, @y, @z)
-----------------------------------------------------------------------------------------
```

Table 3.1 Output items of the environment information for measuring profiling data

Output Item	Meaning of Output Item
@vl	Version number of the Profiler
@date	Measurement date and time of profile data
@pno	Process number
@frequency	CPU Frequency It is the value collected from the following "system CPU frequency file" for each process when the Profiler ends. `/sys/devices/system/cpu/cpuN/cpufreq/scaling_cur_freq  (N in cpuN is the core number)` If the "system CPU frequency file" does not exist, it is the value calculated inside the Profiler. **Note** If any of the following conditions applies, "--" is output as the CPU frequency: - The -Inocpupa option is enabled in "3.1.4 Measuring Profile Data" or "3.1.5 Outputting Profile Result" - The -Hmethod=fast option is specified in "3.1.4 Measuring Profile Data" and measurement has never been performed on the measurement region specified in "3.1.1 Adding a Measurement Region Specifying Routine" - The -Hmode=user option is specified in "3.1.4 Measuring Profile Data"
@sno	Spawn number It is taken in the (Spawn @num) format. (Where @num is a numerical value) This item is output only when all the following conditions are met. - The measurement target is the MPI program. - When the fapp command is set with the -Wspawn option enabled, or the info key "fjprof_spawn_dir_name" is set in the program. - The process is a dynamically generated process.

Output Item	Meaning of Output Item
@type	Program execution format (@thno shows the number of threads.) "SERIAL" : Serial "Thread (AUTO) @thno" : Automatic parallelization only "Thread (OpenMP) @thno" : OpenMP only "Thread (OpenMP & AUTO) @thno" : OpenMP and automatic parallelization "MPI" : MPI only "MPI & Thread (AUTO) @thno" : MPI and automatic parallelization "MPI & Thread (OpenMP) @thno" : MPI,OpenMP "MPI & Thread (OpenMP & AUTO) @thno" : MPI,OpenMP, and automatic parallelization
(@x, @y, @z)	Logical shape at the time of MPI program execution **Point** Outputs the result only when the data measurement target is an MPI program. The logical shape of dynamically generated processes are not outputted.

Point

If CPU frequency(including dynamically generated process) is different for each process, multiple lines of CPU frequency are output. However, if there are consecutive processes with the same CPU frequency, or if the -Inocpupa option is specified, CPU frequency is output as a single line.

```
CPU frequency                  : Process      @spno - @lpno @frequency (MHz)   (Spawn @ssno - @lsno )
```

For consecutive processes, @spno is the lowest process number and @lpno is the highest process number. @ssno is the lowest spawn number and @lsno is the highest spawn number. If the -pinput=*n* option of the fapppx command or fapp command is used to restrict the process being read, the minimum and maximum process numbers or spawn numbers that are read and with the same CPU frequency in the range correspond to @spno and @lpno respectively. See "3.1.5 Outputting Profile Result" for the -pinput=*n* option.

3.2.2.2 Statistical Time Information

As statistical time information, the number of calls, elapsed time, and average, maximum, and minimum user CPU time and system CPU time for each measurement target region.

Output format of the statistical time information

```
Basic profile

 ********************************************************************************
        Application
 ********************************************************************************

      Kind   Elapsed(s)      User(s)    System(s)         Call
    -------------------------------------------------------
       AVG      @elapse        @user         @sys        @call    @name @number @sno
       MAX      @elapse        @user         @sys        @call
       MIN      @elapse        @user         @sys        @call

 ********************************************************************************
        Process @pno @sno
 ********************************************************************************

             Elapsed(s)      User(s)    System(s)         Call
    -------------------------------------------------------
                @elapse        @user         @sys        @call    @name @number
```

Table 3.2 Output items of the statistics time information

Output item	Meaning of Output Item
@elapse	Elapsed time (s)
@user	User CPU time (s) If @type of "3.2.2.1 Environment Information for Measurement Profiling Data" is other than "SERIAL" or "MPI", or if you specified the -Inocputime option when measuring the profile data, only "--" is output for this item.
@sys	System CPU time (s) If @type of "3.2.2.1 Environment Information for Measurement Profiling Data" is other than "SERIAL" or "MPI", or if you specified the -Inocputime option when measuring the profile data, only "--" is output for this item.
@call	Call count
@name	Group name
@number	Advanced number
@sno	Spawn number It is taken in the (Spawn @num) format. (Where @num is a numerical value) This item is output only when all the following conditions are met. - The measurement target is the MPI program. - When the fapp command is set with the -Wspawn option enabled, or the info key "fjprof_spawn_dir_name" is set in the program. - The process is a dynamically generated process.
@pno	Process number

3.2.2.3 MPI Communication Cost Information

As MPI communication cost information, the number of MPI function executions, message length, and average, maximum, and minimum execution time and waiting time are output.

Output format of the MPI communication cost information

```
MPI profile

 ********************************************************************************
        Application
 ********************************************************************************

  Kind   Elapsed(s)   Wait(s)   Byte      Call (   0-4K     4K-64K    64K-1024K   1024KB-)
-------------------------------------------------------------------------------------------
          @elapse    @wait  ----     @call     @ma       @mb         @mc       @md   @name @number @sno
-------------------------------------------------------------------------------------------
   AVG      @elapse     @wait  @byte     @call      @ma        @mb          @mc        @md   @mfunc
   MAX      @elapse     @wait  @byte     @call      @ma        @mb          @mc        @md
   MIN      @elapse     @wait  @byte     @call      @ma        @mb          @mc        @md


 ********************************************************************************
        Process @pno @sno
 ********************************************************************************

         Elapsed(s)   Wait(s)   Byte      Call (   0-4K     4K-64K    64K-1024K   1024KB-)
-------------------------------------------------------------------------------------------
           @elapse   @wait    ----     @call     @ma       @mb         @mc       @md   @name @number
-------------------------------------------------------------------------------------------
           @elapse   @wait    @byte     @call      @ma        @mb          @mc        @md   @mfunc
```

Table 3.3 Output items of the MPI communication cost information

Output Item	Meaning of Output Item
@elapse	Elapsed time (s)
@wait	Waiting time (s)
@byte	Average message length (Byte)
@call	Number of MPI library calls
@ma	Number of MPI library calls when the message length is 0 (inclusive) to 4 KB (not inclusive)
@mb	Number of MPI library calls when the message length is 4 KB (inclusive) to 64 KB (not inclusive)
@mc	Number of MPI library calls when the message length is 64 KB (inclusive) to 1024 KB (not inclusive)
@md	Number of MPI library calls when the message length is 1024 KB or more
@name	Group name
@number	Advanced number
@sno	Spawn number It is taken in the (Spawn @num) format. (Where @num is a numerical value) This item is output only when all the following conditions are met. - The measurement target is the MPI program. - When the fapp command is set with the -Wspawn option enabled, or the info key "fjprof_spawn_dir_name" is set in the program. - The process is a dynamically generated process.
@mfunc	MPI library name
@pno	Process number

Note

If the MPI routine belongs to the Persistent Collective Communication Request, the output method for the MPI communication cost information changes. For details on the "Persistent Collective Communication Request" and the list of routines that are treated as a Persistent Collective Communication Request, see the "MPI User's Guide". The output method when using the MPI_START routine to initiate the Persistent Collective Communication Request differs from when using the MPI_STARTALL routine to initiate the communication.

1. When using the MPI_START routine to initiate the Persistent Collective Communication Request

   Both the Persistent Collective Communication Request routine and the MPI_START routine are output as the MPI communication cost information. They have the following features.

   Persistent Collective Communication Request routine

   - As for @mfunc, the name of the Persistent Collective Communication routine is output.
   - As for @byte is fixed at 0.
   - Outputs other than the above are the same as the normal outputs.

   MPI_START routine

   - As for @mfunc, the name of the MPI_START routine and the name of the Persistent Collective Communication Request routine initiated with MPI_START are output. The name of the Persistent Collective Communication Request routine is output with parentheses "()" following the MPI_START routine.
   - As for @elapsed, @wait, and @call, the information on the MPI_START routine is output.
   - As for @byte, @ma, @mb, @mc, and @md, the information on the Persistent Collective Communication Request routine which has been initiated is output.

- If multiple MPI_START routines exist, data is aggregated for each persistent communication request routine which has been initiated.
- Outputs other than the above are the same as the normal outputs.

2. When using the MPI_STARTALL routine to initiate the persistent communication request

   Both the Persistent Collective Communication Request routine and the MPI_STARTALL routine are output as the MPI communication cost information. They have the following features.

   Persistent Collective Communication Request routine

   - As for @mfunc, the name of the Persistent Collective Communication routine is output.
   - As for @byte, 0 is fixed.
   - Outputs other than the above are the same as the normal outputs.

   MPI_STARTALL routine

   Information on the MPI_STARTALL routine consists of the following two types of lines.

   - Lines in which @mfunc is the MPI_STARTALL routine (hereafter referred to as the "main lines")
   - Lines in which @mfunc is the "name of Persistent Collective Communication Request routine initiated with the "MPI_STARTALL routine (hereafter referred to as the "detail lines")

   They have the following features.

   Main lines

   - As for @mfunc, the MPI_STARTALL routine is output.
   - As for @elapsed, @wait, and @call, the information on the MPI_STARTALL routine is output.
   - As for @byte, @ma, @mb, @mc, and @md, the aggregated data of all the Persistent Collective Communication Request routines that have been initiated is output.

   Detail lines

   - Information on the detail lines is output with the parentheses "()", @mfunc is output with indentation.
   - As for @mfunc, the name of the Persistent Collective Communication Request routine is output that has been initiated with MPI_STARTALL. If more than one Persistent Collective Communication Request routine has been initiated, one line per routine is output.
   - Outputs for @elapsed and @wait are fixed as "--".
   - As for @byte, @call, @ma, @mb, @mc, and @md, the aggregated results of the routine corresponding to @mfunc are output with the parentheses "()".

   If more than one MPI_STARTALL routine exists, data is aggregated as the same routine only when the type and the order of invocation for the initiated Persistent Collective Communication Request routines are identical. However, if the same Persistent Collective Communication Request routine is called multiple times, only the first invocation counts.

Example

An example of output (excerpt) of the Persistent Collective Communication Request routine is shown below.

MPI_START

```
 0.0001       0.0000         0.0000        2        2        0        0        0   MPI_Send_init
 0.0000       0.0000      3999.5000        2        1        1        0        0
MPI_Start(MPI_Send_init)
```

MPI_STARTALL

```
    0.0000       0.0000         0.0000       2        2        0        0        0   MPI_Send_init
    0.0000       0.0000         0.0000       2        2        0        0        0   MPI_Bsend_init
    0.0000       0.0000         0.0000       2        2        0        0        0   MPI_Rsend_init
```

```
    0.0000       0.0000   1091998.5000       2         0         0         0         2   MPI_Startall
        --           --     (3999.5000)      (2)       (1)       (1)       (0)       (0)
(MPI_Send_init)
        --           --    (63999.5000)      (2)       (0)       (1)       (1)       (0)
(MPI_Bsend_init)
        --           --  (1023999.5000)      (2)       (0)       (0)       (1)       (1)
(MPI_Rsend_init)
    0.0000       0.0000        0.0000       1         1         0         0         0   MPI_Comm_rank
```

Formulas of the message length

The calculation formula of the MPI routine other than the Persistent Collective Communication Request is shown below. For the calculation formula of the MPI routine for the Persistent Collective Communication Request, see "Formulas of the message length (Persistent Collective Communication Request Routine)".

Table 3.4 Formulas of the message length

MPI subroutine/function	Formula
MPI_SEND MPI_BSEND MPI_SSEND MPI_RSEND MPI_ISEND MPI_IBSEND MPI_ISSEND MPI_IRSEND	Number of elements in send buffer*Size of data type of each element in send buffer
MPI_RECV	Number of elements received*Size of data type of each element in receive buffer **Note** If the MPI_RECV routine argument "status" is "MPI_STATUS_IGNORE", "Number of elements received" cannot be obtained and becomes 0. Do not specify "MPI_STATUS_IGNORE" for the argument "status".
MPI_IRECV	Number of elements in receive buffer*Size of data type of each element in receive buffer
MPI_SENDRECV	(Number of elements in send buffer*Size of data type of element in send buffer)+(Number of elements received*Size of data type of element in receive buffer) **Note** If the MPI_SENDRECV routine argument "status" is "MPI_STATUS_IGNORE", "Number of elements received" cannot be obtained and becomes 0. Do not specify "MPI_STATUS_IGNORE" for the argument "status".
MPI_SENDRECV_REPLACE	(Number of elements in send and receive buffer*Size of data type of element in send and receive buffer)*2
MPI_BCAST MPI_IBCAST	- Root process Number of elements in buffer*Size of data type of element in buffer*2 - Other than root process Number of elements in buffer*Size of data type of element in buffer
MPI_GATHER MPI_IGATHER	- Root process (Number of elements in send buffer*Size of data type of element in send buffer)+(Total number of processes*Number of elements of data received from each process*Size of data type of receive buffer element)

MPI subroutine/function	Formula
	- Other than root process Number of elements in send buffer*Size of data type of element in send buffer
MPI_GATHERV MPI_IGATHERV	- Root process (Number of elements in send buffer*Size of data type of element in send buffer)+(Total number of elements of data received from each process*Size of data type of receive buffer element) - Other than root process Number of elements in send buffer*Size of data type of element in send buffer
MPI_SCATTER MPI_ISCATTER	- Root process (Total number of processes*Number of elements sent to each process*Size of data type of element in send buffer)+(Number of elements in receive buffer*Size of data type of element in receive buffer) - Other than root process Number of elements in receive buffer*Size of data type of element in receive buffer
MPI_SCATTERV MPI_ISCATTERV	- Root process (Total number of elements of data sent to each process*Size of data type of element in send buffer)+(Number of elements in receive buffer*Size of data type of element in receive buffer) - Other than root process Number of elements in receive buffer*Size of data type of element in receive buffer
MPI_ALLGATHER MPI_IALLGATHER	(Number of elements in send buffer*Size of data type of element in send buffer)+(Total number of processes*Number of elements in receive buffer*Size of data type of element in receive buffer)
MPI_ALLGATHERV MPI_IALLGATHERV	(Number of elements in send buffer*Size of data type of element in send buffer)+(Total number of elements of data received from each process*Size of data type of element in receive buffer)
MPI_ALLTOALL MPI_IALLTOALL	(Total number of processes*Number of elements of data sent to each process*Size of data type of element in send buffer)+(Total number of processes*Number of elements of data received from a single process*Size of data type of element in receive buffer)
MPI_ALLTOALLV MPI_IALLTOALLV	(Total number of elements of data sent to each process*Size of data type of element in send buffer)+(Total number of elements of data received from each process*Size of data type of element in receive buffer)
MPI_REDUCE MPI_IREDUCE	- Root process Number of elements in send buffer*Size of data type of element in send buffer*2 - Other than root process Number of elements in send buffer*Size of data type of element in send buffer
MPI_ALLREDUCE MPI_IALLREDUCE	Number of elements in send buffer*Size of data type of element in send buffer*2
MPI_REDUCE_SCATTER MPI_IREDUCE_SCATTER	(Total number of elements of data sent to each process*Size of data type of element in buffer)+(Number of elements received*Size of data type of element in buffer)
MPI_SCAN MPI_ISCAN	Number of elements in input buffer*Size of data type of element in input buffer*2
MPI_PUT MPI_RPUT	Number of elements in origin buffer*Size of data type of element in origin buffer
MPI_GET MPI_RGET	Number of elements in target buffer*Size of data type of element in target buffer
MPI_ACCUMULATE MPI_RACCUMULATE	Number of elements in origin buffer*Size of data type of element in origin buffer

MPI subroutine/function	Formula
MPI_ALLTOALLW MPI_IALLTOALLW	(Number of elements in send buffer of each process*Size of data type of send buffer of each process)+(Number of elements in receive buffer of each process*Size of data type of receive buffer of each process)
MPI_EXSCAN MPI_IEXSCAN	Number of elements in input buffer*Size of data type of element in input buffer*2
MPI_REDUCE_SCATTER_BLOCK MPI_IREDUCE_SCATTER_BLOCK	(Size of data type of element in buffer*Total number of processes*Number of elements of each block)+(Size of data type of element in buffer*Number of elements of each block)
MPI_MRECV MPI_IMRECV	Size of data type of each receive buffer element*Number of elements in receive buffer
MPI_COMPARE_AND_SWAP	Size of data type of all buffer elements*2
MPI_GET_ACCUMULATE MPI_RGET_ACCUMULATE	(Size of data type of origin buffer*Number of entries of origin buffer)+(Size of data type of target buffer*Number of entries of target buffer)
MPI_NEIGHBOR_ALLGATHER MPI_INEIGHBOR_ALLGATHER	(Number of elements in send buffer*Size of data type of element in send buffer)+(Number of input edges*Number of elements in receive buffer*Size of data type of element in receive buffer)
MPI_NEIGHBOR_ALLGATHERV MPI_INEIGHBOR_ALLGATHERV	(Number of elements in send buffer*Size of data type of element in send buffer)+(Total number of elements in receive buffer of each input edge*Size of data type of element in receive buffer)
MPI_NEIGHBOR_ALLTOALL MPI_INEIGHBOR_ALLTOALL	(Number of output edges*Number of elements in send buffer*Size of data type of element in send buffer)+(Number of input edges*Number of elements in receive buffer*Size of data type of element in receive buffer)
MPI_NEIGHBOR_ALLTOALLV MPI_INEIGHBOR_ALLTOALLV	(Total number of elements in send buffer of each output edge*Size of data type of element in send buffer)+(Total number of elements in receive buffer of each input edge*Size of data type of element in receive buffer)
MPI_NEIGHBOR_ALLTOALLW MPI_INEIGHBOR_ALLTOALLW	(Total number of elements in send buffer of each output edge*Size of data type of element in send buffer of each output edge)+(Total number of elements in receive buffer of each input edge*Size of data type of element in receive buffer of each input edge)

Formulas of the message length (Persistent Collective Communication Request Routine)

The following table shows the calculation formula of the MPI routine for the Persistent Collective Communication Request. For the calculation formula of the MPI routine other than the Persistent Collective Communication Request, see "Formulas of the message length".

Table 3.5 Formulas of the message length (Persistent Collective Communication Request Routine)

MPI subroutine/function	Formula
MPI_SEND_INIT MPI_BSEND_INIT MPI_SSEND_INIT MPI_RSEND_INIT	Number of elements in send buffer*Size of data type of each element in send buffer
MPI_RECV_INIT	Number of elements in receive buffer*Size of data type of each element in receive buffer
MPIX_BCAST_INIT	- Root process Number of elements in buffer*Size of data type of element in buffer*2 - Other than root process Number of elements in buffer*Size of data type of element in buffer
MPIX_GATHER_INIT	- Root process (Number of elements in send buffer*Size of data type of element in send buffer)+(Total number of processes*Number of elements of data received from each process*Size of data type of receive buffer element)

MPI subroutine/function	Formula
	- Other than root process Number of elements in send buffer*Size of data type of element in send buffer
MPIX_GATHERV_INIT	- Root process (Number of elements in send buffer*Size of data type of element in send buffer)+(Total number of elements of data received from each process*Size of data type of receive buffer element) - Other than root process Number of elements in send buffer*Size of data type of element in send buffer
MPIX_SCATTER_INIT	- Root process (Total number of processes*Number of elements sent to each process*Size of data type of element in send buffer)+(Number of elements in receive buffer*Size of data type of element in receive buffer) - Other than root process Number of elements in receive buffer*Size of data type of element in receive buffer
MPIX_SCATTERV_INIT	- Root process (Total number of elements of data sent to each process*Size of data type of element in send buffer)+(Number of elements in receive buffer*Size of data type of element in receive buffer) - Other than root process Number of elements in receive buffer*Size of data type of element in receive buffer
MPIX_ALLGATHER_INIT	(Number of elements in send buffer*Size of data type of element in send buffer)+(Total number of processes*Number of elements in receive buffer*Size of data type of element in receive buffer)
MPIX_ALLGATHERV_INIT	(Number of elements in send buffer*Size of data type of element in send buffer)+(Total number of elements of data received from each process*Size of data type of element in receive buffer)
MPIX_ALLTOALL_INIT	(Total number of processes*Number of elements of data sent to each process*Size of data type of element in send buffer)+(Total number of processes*Number of elements of data received from a single process*Size of data type of element in receive buffer)
MPIX_ALLTOALLV_INIT	(Number of elements in send buffer*Size of data type of element in send buffer)+(Total number of elements of data received from each process*Size of data type of element in receive buffer)
MPIX_ALLTOALLW_INIT	(Number of elements in send buffer of each process*Size of data type of send buffer of each process)+(Number of elements in receive buffer of each process*Size of data type of receive buffer of each process)
MPIX_REDUCE_INIT	- Root process Number of elements in send buffer*Size of data type of element in send buffer*2 - Other than root process Number of elements in send buffer*Size of data type of element in send buffer
MPIX_ALLREDUCE_INIT	Number of elements in send buffer*Size of data type of element in send buffer*2
MPIX_REDUCE_SCATTER_BLOCK_INIT	(Size of data type of element in buffer*Total number of processes*Number of elements of each block)+(Size of data type of element in buffer*Number of elements of each block)
MPIX_REDUCE_SCATTER_INIT	(Total number of elements of data sent to each process*Size of data type of element in buffer)+(Number of elements received*Size of data type of element in buffer)
MPIX_SCAN_INIT	Number of elements in input buffer*Size of data type of element in input buffer*2

MPI subroutine/function	Formula
MPIX_EXSCAN_INIT	Number of elements in input buffer*Size of data type of element in input buffer*2
MPIX_NEIGHBOR_ALLGATHER_INIT	(Number of elements in send buffer*Size of data type of element in send buffer)+(Number of input edges*Number of elements in receive buffer*Size of data type of element in receive buffer)
MPIX_NEIGHBOR_ALLGATHERV_INIT	(Number of elements in send buffer*Size of data type of element in send buffer)+(Total number of elements in receive buffer of each input edge*Size of data type of element in receive buffer)
MPIX_NEIGHBOR_ALLTOALL_INIT	(Number of output edges*Number of elements in send buffer*Size of data type of element in send buffer)+(Number of input edge*Number of elements in receive buffer*Size of data type of element in receive buffer)
MPIX_NEIGHBOR_ALLTOALLV_INIT	(Total number of elements in send buffer of each output edge *Size of data type of element in send buffer)+(Total number of elements in receive buffer of each input edge*Size of data type of element in receive buffer)
MPIX_NEIGHBOR_ALLTOALLW_INIT	(Total number of elements in send buffer of each output edge* Size of data type of element in send buffer of each output edge)+(Total number of elements in receive buffer of each input edge*Size of data type of element in receive buffer of each input edge)

3.2.2.4 CPU Performance Analysis Information

As CPU performance analysis information, CPU performance characteristics at the time of application execution. Output this information only if the -Hevent option argument specified with "3.1.4 Measuring Profile Data" is pa1 or statistics.

Output format of the CPU performance analysis information

Application

```
Performance monitor event : statistics

*******************************************************************************************
    @level
*******************************************************************************************

               Execution                  Floating-point  Mem throughput  Mem throughput
  Kind          time(s)          GFLOPS   peak ratio(%)          (GB/s)   peak ratio(%)
-------------------------------------------------------------------------------------------
   AVG            @time         @value1         @value2         @value3         @value4  @name @number
   MAX            @time         @value1         @value2         @value3         @value4
   MIN            @time         @value1         @value2         @value3         @value4

               Execution                  Floating-point  Mem throughput  Mem throughput
  Kind          time(s)          GFLOPS   peak ratio(%)          (GB/s)   peak ratio(%)
-------------------------------------------------------------------------------------------

   AVG           @time        @value1        @value2        @value3        @value4  @name @number @sno
   MAX            @time         @value1         @value2         @value3         @value4
   MIN            @time         @value1         @value2         @value3         @value4

         Effective       Floating-point  SIMD inst.      SVE operation
  Kind      instruction       operation         rate(%)         rate(%)
-------------------------------------------------------------------------------------------
   AVG          @value5         @value6         @value7         @value8                  @name @number
   MAX          @value5         @value6         @value7         @value8
   MIN          @value5         @value6         @value7         @value8

         Effective       Floating-point  SIMD inst.      SVE operation
  Kind      instruction       operation         rate(%)         rate(%)
```

```
-------------------------------------------------------------------------------------
   AVG         @value5        @value6        @value7        @value8                @name @number @sno
   MAX          @value5         @value6         @value7         @value8
   MIN          @value5         @value6         @value7         @value8

  Kind              IPC            GIPS
-------------------------------------------------------------------------------------
   AVG          @value9        @value10                                                @name @number
   MAX          @value9        @value10
   MIN          @value9        @value10

  Kind              IPC            GIPS
-------------------------------------------------------------------------------------
   AVG         @value9        @value10                                                @name @number @sno
   MAX          @value9        @value10
   MIN          @value9        @value10
```

Process

```
*************************************************************************************
    @level @pno @sno
*************************************************************************************

               Execution                 Floating-point  Mem throughput  Mem throughput
  Kind          time(s)          GFLOPS   peak ratio(%)          (GB/s)   peak ratio(%)
-------------------------------------------------------------------------------------
   AVG            @time         @value1         @value2         @value3         @value4  @name @number
   MAX            @time         @value1         @value2         @value3         @value4
   MIN            @time         @value1         @value2         @value3         @value4

               Execution                 Floating-point  Mem throughput  Mem throughput
  Kind          time(s)          GFLOPS   peak ratio(%)          (GB/s)   peak ratio(%)
-------------------------------------------------------------------------------------
   AVG            @time         @value1         @value2         @value3         @value4  @name @number
   MAX            @time         @value1         @value2         @value3         @value4
   MIN            @time         @value1         @value2         @value3         @value4

         Effective       Floating-point  SIMD inst.      SVE operation
  Kind      instruction       operation         rate(%)         rate(%)
-------------------------------------------------------------------------------------
   AVG          @value5         @value6         @value7         @value8                 @name @number
   MAX          @value5         @value6         @value7         @value8
   MIN          @value5         @value6         @value7         @value8

         Effective       Floating-point  SIMD inst.      SVE operation
  Kind      instruction       operation         rate(%)         rate(%)
-------------------------------------------------------------------------------------
   AVG          @value5         @value6         @value7         @value8                 @name @number
   MAX          @value5         @value6         @value7         @value8
   MIN          @value5         @value6         @value7         @value8

  Kind              IPC            GIPS
-------------------------------------------------------------------------------------
   AVG          @value9        @value10                                                @name @number
   MAX          @value9        @value10
   MIN          @value9        @value10

  Kind              IPC            GIPS
-------------------------------------------------------------------------------------
   AVG          @value9        @value10                                                @name @number
   MAX          @value9        @value10
   MIN          @value9        @value10
```

Thread

```
*******************************************************************************
    @level @pno
*******************************************************************************

               Execution                  Floating-point  Mem throughput  Mem throughput
                time(s)          GFLOPS    peak ratio(%)          (GB/s)   peak ratio(%)
-------------------------------------------------------------------------------
                 @time         @value1         @value2         @value3         @value4  @name @number
                 @time         @value1         @value2         @value3         @value4  @name @number

         Effective       Floating-point  SIMD inst.      SVE operation
            instruction       operation         rate(%)         rate(%)
-------------------------------------------------------------------------------
               @value5         @value6         @value7         @value8                  @name @number
               @value5         @value6         @value7         @value8                  @name @number

                   IPC            GIPS
-------------------------------------------------------------------------------
               @value9        @value10                                                 @name @number
               @value9        @value10                                                 @name @number
```

Table 3.6 Output items of the CPU performance characteristics

Output Item	Meaning of Output Item
@level	Information total level (Application, Process, Thread)
@pno	Process or thread number
@sno	Spawn number It is taken in the (Spawn @num) format. (Where @num is a numerical value) This item is output only when all the following conditions are met. - The measurement target is the MPI program. - When the fapp command is set with the -Wspawn option enabled, or the info key "fjprof_spawn_dir_name" is set in the program. - When @level is Application or Process. - The process is a dynamically generated process.
@time	Time taken to execute instructions in the measurement target region (second)
@value1	Count of floating-point operations performed per second Note The GFLOPS value is calculated by assuming that all elements are active. Therefore, in the case of a program with many inactive elements, a higher value than the original GFLOPS value is output.
@value2	Ratio of the measured value to the theoretical value for floating-point operation performance (%) Note The theoretical value for floating-point operation performance is calculated by assuming that double precision operations are performed. Therefore, in the case of single or half precision, a value 2 to 4 times higher than the actual percentage is output. If the -Hmode=user option is specified in "3.1.4 Measuring Profile Data", this output item is fixed as "--".
@value3	Memory throughput (GB/s)
@value4	Ratio of the measured value to the theoretical value for memory throughput (%)

Output Item	Meaning of Output Item
@value5	Total number of instructions executed Note The total number of instructions executed does not include MOVPRFX instructions.
@value6	Total number of floating-point operations executed
@value7	Ratio of the number of SIMD instructions to the total number of instructions executed (%)
@value8	Ratio of the number of SVE operations to the total number of floating-point operations executed (%)
@value9	Number of instructions executed per cycle
@value10	Number of instructions executed per second

Note

The output of @time and each @value is expected to be less than 12 digits. Therefore, if the output exceeds 13 digits, there is a discrepancy between the heading and the output.

3.2.3 Detail of Profile Result (XML Format)

If the -txml option is specified in "3.1.5 Outputting Profile Result", the result is output in the XML format. The followings are applied to the XML format.

3.2.3.1 Structure of XML format

The structure of the XML format output is described; the whole output of the XML format is enclosed by the <profile> element, and the <profile> element consists of the <environment> element and the <information> element.

XML Format

```
<?xml version="1.0" encoding="utf-8"?>                          XML declaration
<profile type="@type" version="@vid" output_version="@oid">     Profiling information
  <environment>                                               The environment information for
    ......                                                            measuring profiling data
  </environment>
  <information item="advanced">                                 Performance information
    ......
  </information>
</profile>
```

Output items

Element Name	Overview	Description
profile	Profiling information	This element includes the XML format output of Profiler.
environment	Environment information for measuring profiling data	This element includes the following information of the TEXT format(*). - "3.2.2.1 Environment Information for Measurement Profiling Data"
information	Performance information	This element includes the following information of the TEXT format(*). - "3.2.2.2 Statistical Time Information" - "3.2.2.4 CPU Performance Analysis Information" - "3.2.2.3 MPI Communication Cost Information"

(*)Some entries do not match the TEXT and XML formats.

3.2.3.2 Details of XML format output

The following sections describe the elements used in the XML format output. Note that the output items for each element are the same as for TEXT output unless otherwise specified.

3.2.3.2.1 Profiling Information <profile>

This element includes the XML format output of the Advanced Performance Profiler.

Element Name	Description
profile	<profile type="*@type*" version="*@vid*" output_version="*@oid*"> </profile> This element includes the XML format output of Profiler. *@type* indicates the kind of Profiler, and is fixed to "fapp" *@vid* indicates the version number of the Profiler. *@oid* indicates the version number of output format.

3.2.3.2.2 Environment Information for Measuring Profiling Data <environment>

As environment information for measuring profiling data, environment information as of when the profile data was measured is output.

XML Format

```
<environment>
  <measured_time unit="date">@date</measured_time>
  <type_of_program program="@program" />
  <coordinate x="@x" y="@y" z="@z"/>
  <vector_length vlen="@vlen"/>
  <spawn id="@id">
    <process id="@id">
      <host name="@name"/>
      <frequency unit="MHz">@frequency</frequency>
      <cntfrq unit="Hz">@cntfrq</cntfrq>
      <thread id="@id">
        <cmg id="@cmg"/>
        <core id="@core"/>
      </thread>
    </process>
  </spawn>
</environment>
```

Output items

Element Name	Description
environment	<environment> </environment> This element includes the environment information for measuring profiling data.
measured_time	<measured_time unit="date"> *@date* </measured_time> This element shows the measurement date and time of profile data. *@date* shows the measurement data and time in YYYY-MM-DDThh:mm:ss format.
type_of_program	<type_of_program program="*@program* "/> This element shows the program execution format. *@program* is one of the followings:

Element Name	Description
	"SERIAL" : Serial "Thread(AUTO)" : Automatic parallelization only "Thread(OpenMP)" : OpenMP only "Thread(OpenMP+AUTO)" : OpenMP and automatic parallelization "MPI" : MPI only "MPI+Thread(AUTO)" : MPI and automatic parallelization "MPI+Thread(OpenMP)" : MPI and OpenMP "MPI+Thread(OpenMP+AUTO)" : MPI, OpenMP, and automatic parallelization
coordinate	<coordinate x="*@x*" y="*@y*" z="*@z*" /> This element shows the logical shape at the time of MPI program execution. The value of x-axis, y-axis, and z-axis is shown in *@x*, *@y*, and *@z*. **Note** Outputs the result only when the data measurement target is an MPI program. The logical shape of dynamically generated processes are not outputted.
vector_length	<vector_length vlen="*@vlen*"/> This element shows the SVE vector length. *@vlen* show the SVE vector length (bits).
spawn	<spawn id="*@id*"> </spawn> This element shows the spawn number. This element will be output multiple times when the multiple spawned processes exist. *@id* indicates the spawn number. If this process is not spawned, *@id* is 0.
process	<process id="*@id*"> </process> This element shows the process number. This element will be output multiple times when the multiple processes exist. *@id* indicates the process number.
host	<host name="*@name*"/> This element shows the host name of node. *@name* indicates the host name.
frequency	<frequency unit="MHz"> *@frequency* </frequency> This element shows the CPU frequency. The unit is MHz (unit="MHz"). *@frequency* indicates the CPU frequency. **Note** If any of the following conditions applies, "-" is output as the CPU frequency: - The -Inocpupa option is enabled in "3.1.4 Measuring Profile Data" - The -Hmethod=fast option is specified in "3.1.4 Measuring Profile Data" and measurement has never been performed on the measurement region specified in "3.1.1 Adding a Measurement Region Specifying Routine" - The -Hmode=user option is specified in "3.1.4 Measuring Profile Data"
cntfrq	<cntfrq unit="Hz"> *@cntfrq* </cntfrq> This element shows the timer clock frequency. The unit is Hz (unit="Hz").

Element Name	Description
	@cntfrq indicates the timer clock frequency. Note For "3.1.4 Measuring Profile Data", if the -Inocpupa option is enabled, the timer clock frequency outputs "0".
thread	<thread id="*@id*"> </thread> This element shows the thread number. This element will be output multiple times when the multiple threads exist. *@id* indicates the thread number.
cmg	<cmg id="*@cmg*"/> This element shows the CMG number. *@cmg* indicates the CMG number.
core	<core id="*@core*"/> This element shows the core number. *@core* indicates the core number.

3.2.3.2.3 Performance Information <information>

As performance information, several types of performance information are output. The outputs of performance information are the statistics time information, the CPU performance analysis information and the MPI communication cost information for each process or each thread.

XML Format

```
<information item="advanced">
  <region name="@name" id="@number">
    <spawn id="@id">
      <process id="@id">
        <time>
          <elapsed unit="s">@elapsed</elapsed>
          <user unit="s">@user</user>
          <system unit="s">@system</system>
        </time>
        <thread id="@id">
          <call_count>@call_count</call_count>
          <time>
            <user unit="s">@user</user>
            <system unit="s">@system</system>
          </time>
          <cpupa>
            .....
          </cpupa>
        </thread>
        <mpi>
          .....
        </mpi>
      </process>
    </spawn>
  </region>
</information>
```

Output items

Element Name	Description
information	<information item="*@item*"> </information>

Element Name	Description
	This element includes the performance information. For *@item*, the kind of performance information is output, and is fixed to "advanced".
region	<region name="*@name*" id="*@number*"> This element shows the measurement region name. *@name* indicates the group name and *@number* indicates the detail number.
spawn	<spawn id="*@id*"> </spawn> This element shows the spawn number. This element will be output multiple times when the multiple spawned processes exist. *@id* indicates the spawn number. If this process is not spawned, *@id* is 0.
process	<process id="*@id*"> </process> This element shows the process number. This element will be output multiple times when the multiple processes exist. *@id* indicates the process number.
time (directly under process)	<time> </time> This element includes the statistic time information of process.
elapsed (directly under process and time)	<elapsed unit="s" > *@elapsed* </elapsed> This element shows the elapsed time of each process in unit of second (unit="s"). *@elapsed* indicates the elapsed time of each process.
user (directly under process and time)	<user unit="s"> *@user* </user> This element shows the user CPU time of each process in unit of second (unit="s"). *@user* indicates the user CPU time of each process. If *@program* of <type_of_program> element is other than "SERIAL" or "MPI", or if you specified the -Inocputime option when measuring the profile data, this element does not output.
system (directly under process and time)	<system unit="s"> *@system* </system> This element shows the system CPU time of each process in unit of second (unit="s"). *@system* indicates the system CPU time of each process. If *@program* of <type_of_program> element is other than "SERIAL" or "MPI", or if you specified the -Inocputime option when measuring the profile data, this element does not output.
thread	<thread id="*@id*"> </thread> This element shows the thread number. This element will be output multiple times when the multiple threads exist. *@id* indicates the thread number.
call_count	<call_count> *@call_count* </call_count> This element shows the call count. *@call_count* indicates the call count.
time (directly under thread)	<time> </time> This element includes the statistic time information of thread.
user	<user unit="s"> *@user* </user> This element shows the user CPU time of each thread in unit of second (unit="s").

Element Name	Description
(directly under thread and time)	*@user* indicates the user CPU time of each thread. If *@program* of <type_of_program> element is other than "SERIAL" or "MPI", or if you specified the -Inocputime option when measuring the profile data, this element does not output.
system (directly under thread and time)	<system unit="s"> *@system* </system> This element shows the system CPU time of each thread in unit of second (unit="s"). *@system* indicates the system CPU time of each thread. If *@program* of <type_of_program> element is other than "SERIAL" or "MPI", or if you specified the -Inocputime option when measuring the profile data, this element does not output.
cpupa	<cpupa> </cpupa> This element includes the CPU performance analysis information. Refer "3.2.3.2.4 CPU Performance Analysis Information <cpupa>" for details.
mpi	<mpi> </mpi> This element includes the MPI communication cost information. Refer "3.2.3.2.5 MPI Communication Cost Information <mpi>" for details.

3.2.3.2.4 CPU Performance Analysis Information <cpupa>

As CPU performance analysis information, CPU performance characteristics at the time of application execution are output.

XML Format

```
<cpupa>
  <event name="@name"> @event </event>
</cpupa>
```

Output items

Element Name	Description
cpupa	<cpupa> </cpupa> This element includes the CPU performance analysis information.
event	<event name="@name"> @event </event> This element shows CNTVCT (Counter-timer Virtual Count), PMCCNTR (Performance Monitors Cycle Counter), and the measured information of the PMU event. This element outputs, in addition to CNTVCT and PMCCNTR, the information of the measured PMU events. @name indicates the PMU event number. @event indicates the measured result corresponding to the PMU event number. For more information about CNTVCT and PMCCNTR, see the documents and web site published by Arm. For PMU events, see "A64FX PMU Events" on https://github.com/fujitsu/A64FX/tree/master/doc/ .

3.2.3.2.5 MPI Communication Cost Information <mpi>

As MPI communication cost information, the number of MPI function executions, message length, execution time and waiting time are output.

XML format (MPI routines other than MPI_start or MPI_startall)

```
<mpi>
  <function name="@name">
```

```
    <time>
      <elapsed unit="s">@elapsed</elapsed>
      <wait unit="s">@wait</wait>
    </time>
    <call_count>@call_count</call_count>
    <total_message_length>@total_message_length</total_message_length>
    <message_length_histogram>
      <call_count min_length="@min_length" max_length="@max_length">@call_count</call_count>
    </message_length_histogram>
  </function>
</mpi>
```

XML format(MPI_start or MPI_startall routine)

```
<mpi>
  <function name="@name">
    <time>
      <elapsed unit="s">@elapsed</elapsed>
      <wait unit="s">@wait</wait>
    </time>
    <request>
      <function name="@name">
        <call_count>@call_count</call_count>
        <total_message_length>@total_message_length</total_message_length>
        <message_length_histogram>
          <call_count min_length="@min_length" max_length="@max_length">@call_count</call_count>
        </message_length_histogram>
      </function>
    </request>
  </function>
</mpi>
```

Output items

Element Name	Description
mpi	<mpi> </mpi> This element includes the MPI communication cost information.
function	<function name="*@name*"> </function> This element shows the MPI library routine name. *@name* indicates the MPI library routine name.
time	<time> </time> This element includes the statistic time information of MPI library.
elapsed	<elapsed unit="s" > *@elapsed* </elapsed> This element shows the elapsed time of each MPI library routine in unit of second (unit="s"). *@elapsed* indicates the elapsed time of each MPI library routine.
wait	<wait unit="s" > *@wait* </wait> This element shows the waiting time of each MPI library routine in unit of second (unit="s"). *@wait* indicates the waiting time of each MPI library routine.
request	<request> </request> This element includes the detailed information of MPI_start or MPI_startall routines.
call_count (directly under function)	<call_count> *@call_count* </call_count> This element shows the number of calls of each MPI library routine.

Element Name	Description
	@call_count indicates the number of calls of each MPI library routine.
total_message_length	<total_message_length> *@total_message_length* </total_message_length> This element shows the total message length of each MPI library routine. *@total_message_length* indicates the total message length of each MPI library routine.
message_length_histogram	<message_length_histogram> </message_length_histogram> This element includes the histogram of message length.
call_count (directly under message_length_histogram)	<call_count min_length="*@min_length*" max_length="*@max_length*"> *@call_count* </call_count> This element shows the number of calls in a range of message length. *@min_length* and *@max_length* indicate the lower bound and the upper bound of range of message length. *@call_count* indicates the number of calls.

Chapter 4 CPU Performance Analysis Report

This chapter describes the CPU Performance Analysis Report.

The CPU Performance Analysis Report aggregates a lot of CPU performance analysis information measured through multiple executions and visualizes it in an easy-to-understand way, using tables and associated graphs. The Advanced Performance Profiler is used to measure CPU performance analysis information. The CPU Performance Analysis Report is designed not to exceed one A3 sheets when printed. The CPU Performance Analysis Report provides the following four stages (Single Report, Brief Report, Standard Report, and Detail Report) according to the type and granularity of the information to be displayed. The CPU Performance Analysis Report file is in Microsoft Excel file format (.xlsm).

Single Report

This is a CPU performance analysis report that requires minimum number of measurements for creating a report. A high level of information is output such as execution time, operation performance, memory throughput and number of instructions. If you use a Single report, perform measurement once with the Advanced Performance Profiler.

Brief Report

This is a CPU Performance Analysis Report that can be created with the fewer number of measurement times. We recommend the Brief Report if you want to readily use the CPU Performance Analysis Report. Although the amount of information is less than the Standard Report, the number of measurement times for creating a report can be reduced compared with the Standard Report. To use the Brief Report, perform measurement five times with the Advanced Performance Profiler.

Standard Report

This is the standard CPU Performance Analysis Report. We recommend the Standard Report for normal use. To use the Standard Report, perform measurement eleven times with the Advanced Performance Profiler.

Detail Report

This is the most detailed CPU Performance Analysis Report. We recommend the Detail Report if the amount of information from the Standard Report is insufficient. Although the largest number of measurement times is required for report creation, all information for the CPU Performance Analysis Report is displayed. To use the Detail Report, perform measurement seventeen times with the Advanced Performance Profiler.

The following is a list of the information available in the CPU Performance Analysis Report. "all" in the table indicates that all information is output, "some" indicates that some information is output, and "-" indicates that no information is output.

Table 4.1 CPU Performance Analysis Report, List of Tables

Table Title	Report Type				Table Outline
	Single	Brief	Standard	Detail	
Information	all	all	all	all	Displays measurement environment information and user's specification.
Statistics	some	all	all	all	Displays information related to CPU performance characteristics, such as memory throughput, number of instructions, and number of operations. **[Simple]** Displays information concerning the CPU behaviors such as the memory throughput, number of instructions, and number of operations. **[Brief , Standard , Detail]** In addition to the contents of the Single report, displays the ratio of the active element in the floating-point operations.
Cycle Accounting	-	some	all	all	Displays program execution time breakdown. **[Brief]** Displays program execution time by classifying it into 9 types. **[Standard , Detail]** Displays program execution time by classifying it into 20 types.

Table Title	Report Type				Table Outline
	Single	Brief	Standard	Detail	
Busy	-	some	some	all	Displays the information on the program memory cache and busy rate for the operation pipeline. **[Brief]** Displays the busy rates such as for the primary cache, secondary cache, memory, and floating-point operation pipeline, as well as the occurrence rate of SFI (Store Fetch Interlock). **[Standard]** In addition to the contents of the brief report, displays the busy rates for the integer operation pipeline, address calculation operation pipeline, and predicate operation pipeline. **[Detail]** In addition to the contents of the standard report, displays the ratio of the active element in the L1 pipeline.
Cache	-	some	all	all	Displays information about cache misses. **[Brief]** In the case of the Brief Report, displays the number of cache misses of the primary data and secondary caches and the ratio to the number of load store instructions. **[Standard , Detail]** In the case of the Standard or Detail Report, the breakdown of the number of cache misses is added in addition to the contents of the Brief Report.
Instruction	-	some	some	all	Displays information about instruction mixes. **[Brief]** Displays information about instruction mixes by classifying it into 9 types. **[Standard]** Displays information about instruction mixes by classifying it into 25 types. **[Detail]** Displays information about instruction mixes by classifying it into 28 types.
FLOPS	-	some	all	all	Displays the information on the floating-point operations. **[Brief]** Displays the performance of the floating-point operation including the active element ratio. **[Standard , Detail]** In addition to the contents of the brief report, displays the number of floating-point operations of different precision.
Extra	-	-	-	all	Displays the details of gather instructions and information on instructions that are not included in instruction mixes.
Hardware Prefetch Rate (%) (/Hardware Prefetch)	-	-	-	all	Displays a breakdown of hardware prefetches.
Data Transfer CMGs	-	-	-	all	Displays information about throughput between the user-specified CMG and all other CMGs, memory, Tofu, and PCI.

Table Title	Report Type				Table Outline
	Single	Brief	Standard	Detail	
Power Consumption (W)	-	-	all	all	Displays the power consumption of cores, L2 cache, and memory.

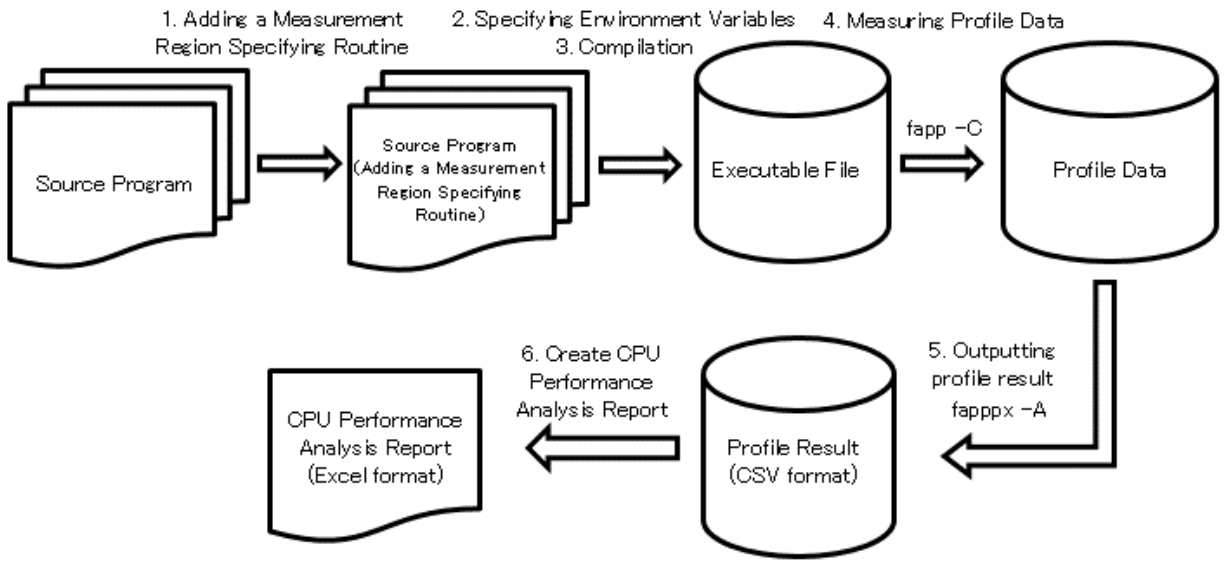
Point

The input file is common to these reports. Therefore, for example, if you want to change an already created Standard Report to a Detail Report, only six measurements (difference between the two reports) are additionally required.

4.1 Procedure for Using the CPU Performance Analysis Report

This section provides the procedure for using the CPU Performance Analysis Report.

Figure 4.1 Procedure for using the CPU Performance Analysis Report



The following describes each operation in detail.

4.1.1 Adding a Measurement Region Specifying Routine

To the source code, add the measurement region specifying routine required for specifying the region (starting and stopping positions of measurement) from which profile data is measured.

For detail, see "3.1.1 Adding a Measurement Region Specifying Routine".

4.1.2 Specifying Environment Variables

Specify environment variables required when using the Profiler. For details, see "2.1.2 Specifying Environment Variables "

4.1.3 Compilation

Compile a program. For detail, see "2.1.3 Compilation".

4.1.4 Measuring Profile Data

Measure data by using the fapp command. To measure profile data for the CPU Performance Analysis Report, you need to specify the -Hevent option with the fapp command. For fapp command, see "3.1.4 Measuring Profile Data"

The following shows execution examples for each report.

Single Report

To create a Single Report, perform measurement once by specifying -Hevent=pa1.

```
# Measurement example for Single Report creation
fapp  -C  -d  ./rep1  -Hevent=pa1  ./a.out
```

Brief Report

To create a Brief Report, perform measurement five times by specifying -Hevent=pa1 to -Hevent=pa5.

```
# Measurement example for Brief Report creation
fapp  -C  -d  ./rep1  -Hevent=pa1  ./a.out
fapp  -C  -d  ./rep2  -Hevent=pa2  ./a.out
fapp  -C  -d  ./rep3  -Hevent=pa3  ./a.out
fapp  -C  -d  ./rep4  -Hevent=pa4  ./a.out
fapp  -C  -d  ./rep5  -Hevent=pa5  ./a.out
```

Standard Report

To create a Standard Report, perform measurement eleven times by specifying -Hevent=pa1 to -Hevent=pa11.

```
# Measurement example for Standard Report creation
fapp  -C  -d  ./rep1  -Hevent=pa1  ./a.out
fapp  -C  -d  ./rep2  -Hevent=pa2  ./a.out
fapp  -C  -d  ./rep3  -Hevent=pa3  ./a.out
fapp  -C  -d  ./rep4  -Hevent=pa4  ./a.out
fapp  -C  -d  ./rep5  -Hevent=pa5  ./a.out
fapp  -C  -d  ./rep6  -Hevent=pa6  ./a.out
fapp  -C  -d  ./rep7  -Hevent=pa7  ./a.out
fapp  -C  -d  ./rep8  -Hevent=pa8  ./a.out
fapp  -C  -d  ./rep9  -Hevent=pa9  ./a.out
fapp  -C  -d  ./rep10  -Hevent=pa10  ./a.out
fapp  -C  -d  ./rep11  -Hevent=pa11  ./a.out
```

Detail Report

To create a Detail Report, perform measurement seventeen times by specifying -Hevent=pa1 to -Hevent=pa17.

```
# Measurement example for Detail Report creation
fapp  -C  -d  ./rep1  -Hevent=pa1  ./a.out
fapp  -C  -d  ./rep2  -Hevent=pa2  ./a.out
fapp  -C  -d  ./rep3  -Hevent=pa3  ./a.out
fapp  -C  -d  ./rep4  -Hevent=pa4  ./a.out
fapp  -C  -d  ./rep5  -Hevent=pa5  ./a.out
fapp  -C  -d  ./rep6  -Hevent=pa6  ./a.out
fapp  -C  -d  ./rep7  -Hevent=pa7  ./a.out
fapp  -C  -d  ./rep8  -Hevent=pa8  ./a.out
fapp  -C  -d  ./rep9  -Hevent=pa9  ./a.out
fapp  -C  -d  ./rep10  -Hevent=pa10  ./a.out
fapp  -C  -d  ./rep11  -Hevent=pa11  ./a.out
fapp  -C  -d  ./rep12  -Hevent=pa12  ./a.out
fapp  -C  -d  ./rep13  -Hevent=pa13  ./a.out
fapp  -C  -d  ./rep14  -Hevent=pa14  ./a.out
fapp  -C  -d  ./rep15  -Hevent=pa15  ./a.out
fapp  -C  -d  ./rep16  -Hevent=pa16  ./a.out
fapp  -C  -d  ./rep17  -Hevent=pa17  ./a.out
```

Note

- When measuring profile data for the CPU Performance Analysis Report, do not specify the -Inocpupa option with the fapp command. If you specify the -Inocpupa option, the -Hevent option is disabled.
- Only one -Hevent option can be specified for one measurement. If you specify multiple -Hevent options, the last specified -Hevent option is enabled.
- The operation of the program must be the same across all measurements. For example, ensure that input data is the same across all measurement times.
- The measurement order is random. For example, it is no problem if measurement for which Hevent=pa1 is specified is performed after measurement for which -Hevent=pa2 is specified.
- According to the -Hevent option with which measurement is performed, the name of the CSV file output in "4.1.5 Outputting Profile Result " is determined. Therefore, we recommend giving a name to the directory that stores measured profile data in a way that you can tell the specified arguments.

4.1.5 Outputting Profile Result

Use the fapppx command or fapp command of the Advanced Performance Profiler to output CSV files to be used for the CPU Performance Analysis Report. To output CSV files that are to be input to the CPU Performance Analysis Report, specify the -tcsv option. For details on the options, see "3.1.5 Outputting Profile Result". The following shows an execution example for each report.

Single Report

To create a Single Report, output the five CSV files pa1.csv by using the profile data -Hevent=pa1.

```
# Example of outputting profile data stored in the directories "rep1" in CSV format for the CPU
Performance Analysis Report
fapppx  -A  -d ./rep1  -Icpupa,nompi  -tcsv  -o pa1.csv
```

Brief Report

To create a Brief Report, output the five CSV files pa1.csv to pa5.csv by using the profile data -Hevent=pa1 to -Hevent=pa5.

```
# Example of outputting profile data stored in the directories "rep1" to "rep5" in CSV format for
the CPU Performance Analysis Report
fapppx  -A  -d ./rep1  -Icpupa,nompi  -tcsv  -o pa1.csv
fapppx  -A  -d ./rep2  -Icpupa,nompi  -tcsv  -o pa2.csv
fapppx  -A  -d ./rep3  -Icpupa,nompi  -tcsv  -o pa3.csv
fapppx  -A  -d ./rep4  -Icpupa,nompi  -tcsv  -o pa4.csv
fapppx  -A  -d ./rep5  -Icpupa,nompi  -tcsv  -o pa5.csv
```

Standard Report

To create a Standard Report, output the five CSV files pa1.csv to pa11.csv by using the profile data -Hevent=pa1 to -Hevent=pa11.

```
# Example of outputting profile data stored in the directories "rep1" to "rep11" in CSV format for
the CPU Performance Analysis Report
fapppx  -A  -d ./rep1  -Icpupa,nompi  -tcsv  -o pa1.csv
fapppx  -A  -d ./rep2  -Icpupa,nompi  -tcsv  -o pa2.csv
fapppx  -A  -d ./rep3  -Icpupa,nompi  -tcsv  -o pa3.csv
fapppx  -A  -d ./rep4  -Icpupa,nompi  -tcsv  -o pa4.csv
fapppx  -A  -d ./rep5  -Icpupa,nompi  -tcsv  -o pa5.csv
fapppx  -A  -d ./rep6  -Icpupa,nompi  -tcsv  -o pa6.csv
fapppx  -A  -d ./rep7  -Icpupa,nompi  -tcsv  -o pa7.csv
fapppx  -A  -d ./rep8  -Icpupa,nompi  -tcsv  -o pa8.csv
fapppx  -A  -d ./rep9  -Icpupa,nompi  -tcsv  -o pa9.csv
fapppx  -A  -d ./rep10  -Icpupa,nompi  -tcsv  -o pa10.csv
fapppx  -A  -d ./rep11  -Icpupa,nompi  -tcsv  -o pa11.csv
```

Detail Report

To create a Detail Report, output the five CSV files pa1.csv to pa17.csv by using the profile data -Hevent=pa1 to -Hevent=pa17.

```
# Example of outputting profile data stored in the directories "rep1" to "rep17" in CSV format for
the CPU Performance Analysis Report
fapppx  -A  -d ./rep1  -Icpupa,nompi  -tcsv  -o pa1.csv
fapppx  -A  -d ./rep2  -Icpupa,nompi  -tcsv  -o pa2.csv
fapppx  -A  -d ./rep3  -Icpupa,nompi  -tcsv  -o pa3.csv
fapppx  -A  -d ./rep4  -Icpupa,nompi  -tcsv  -o pa4.csv
fapppx  -A  -d ./rep5  -Icpupa,nompi  -tcsv  -o pa5.csv
fapppx  -A  -d ./rep6  -Icpupa,nompi  -tcsv  -o pa6.csv
fapppx  -A  -d ./rep7  -Icpupa,nompi  -tcsv  -o pa7.csv
fapppx  -A  -d ./rep8  -Icpupa,nompi  -tcsv  -o pa8.csv
fapppx  -A  -d ./rep9  -Icpupa,nompi  -tcsv  -o pa9.csv
fapppx  -A  -d ./rep10  -Icpupa,nompi  -tcsv  -o pa10.csv
fapppx  -A  -d ./rep11  -Icpupa,nompi  -tcsv  -o pa11.csv
fapppx  -A  -d ./rep12  -Icpupa,nompi  -tcsv  -o pa12.csv
fapppx  -A  -d ./rep13  -Icpupa,nompi  -tcsv  -o pa13.csv
fapppx  -A  -d ./rep14  -Icpupa,nompi  -tcsv  -o pa14.csv
fapppx  -A  -d ./rep15  -Icpupa,nompi  -tcsv  -o pa15.csv
fapppx  -A  -d ./rep16  -Icpupa,nompi  -tcsv  -o pa16.csv
fapppx  -A  -d ./rep17  -Icpupa,nompi  -tcsv  -o pa17.csv
```

Note

- When outputting a CSV file for the CPU Performance Analysis Report, do not specify the -Inocpupa option with the fapppx command or fapp command. If you specify the -Inocpupa option, information necessary for the CPU Performance Analysis Report is not output.
- The names of CSV files used for the CPU Performance Analysis Report are fixed. Add .csv to the value specified in *event* in the -Hevent=*event* option at the time of measurement with the fapp command, and use it as the file name. It is no problem if you manually change the file name after outputting the file with a different name.
- The output order is random. For example, it is no problem if pa1.csv is output after pa2.csv.

4.1.6 Creating a CPU Performance Analysis Report

Read the CSV format file output by "4.1.5 Outputting Profile Result " into the CPU performance analysis report. See "4.1.6.1 Error and Warning Messages Output by the CPU Performance Analysis Report" for the messages output from the CPU Performance Analysis Report.

1. Store CSV files output in "4.1.5 Outputting Profile Result " and the CPU Performance Analysis Report file (cpu_pa_report.xlsm) in the same directory. The CPU Performance Analysis Report file is stored in the following location in login node.

   ```
   /installation_path/misc/cpupa/cpu_pa_report.xlsm
   ```

   For details on "*installation_path*", contact the system administrator.

2. Copy the directory prepared in 1. to an environment that can run Microsoft Excel.
3. Start the CPU Performance Analysis Report file (cpu_pa_report.xlsm).

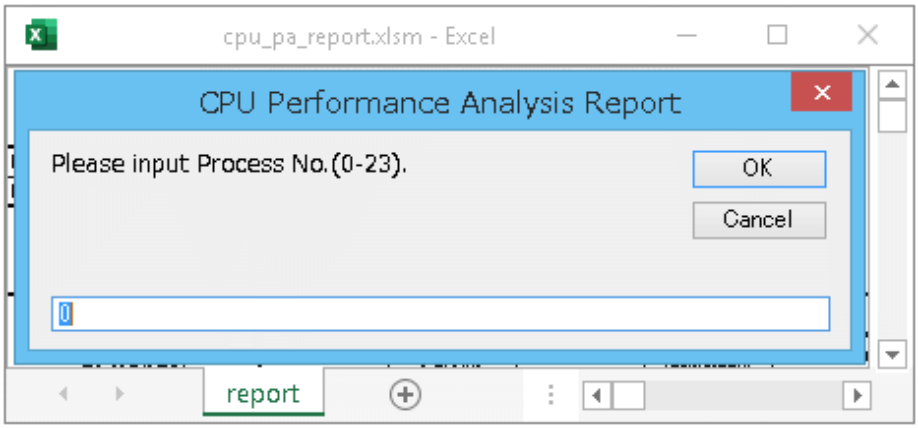
   Note

   The CPU Performance Analysis Report uses the macro function of Microsoft Excel. If a macro is disabled due to security settings, manually enable it. For details on how to enable a macro, see the help for your Microsoft Excel or other relevant documents.

4. A dialog appears where you select contents to be output to the CPU Performance Analysis Report. Enter necessary information.

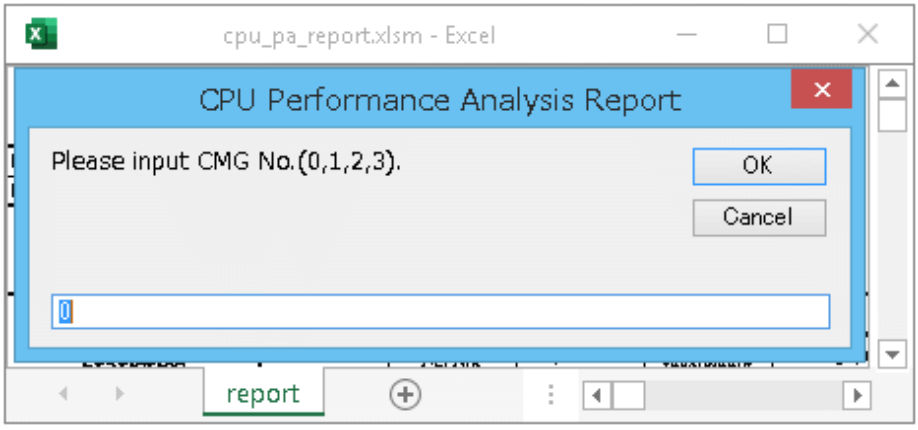
   a. If data from multiple processes exists in a CSV file, the process number input window appears. Enter the process number you want to reference. If the entered process number does not exist in data in pa1.csv, a warning message appears and you are returned to the process number input window. If there is only one process, the process number input window does not appear.

   Figure 4.2 Process number input window



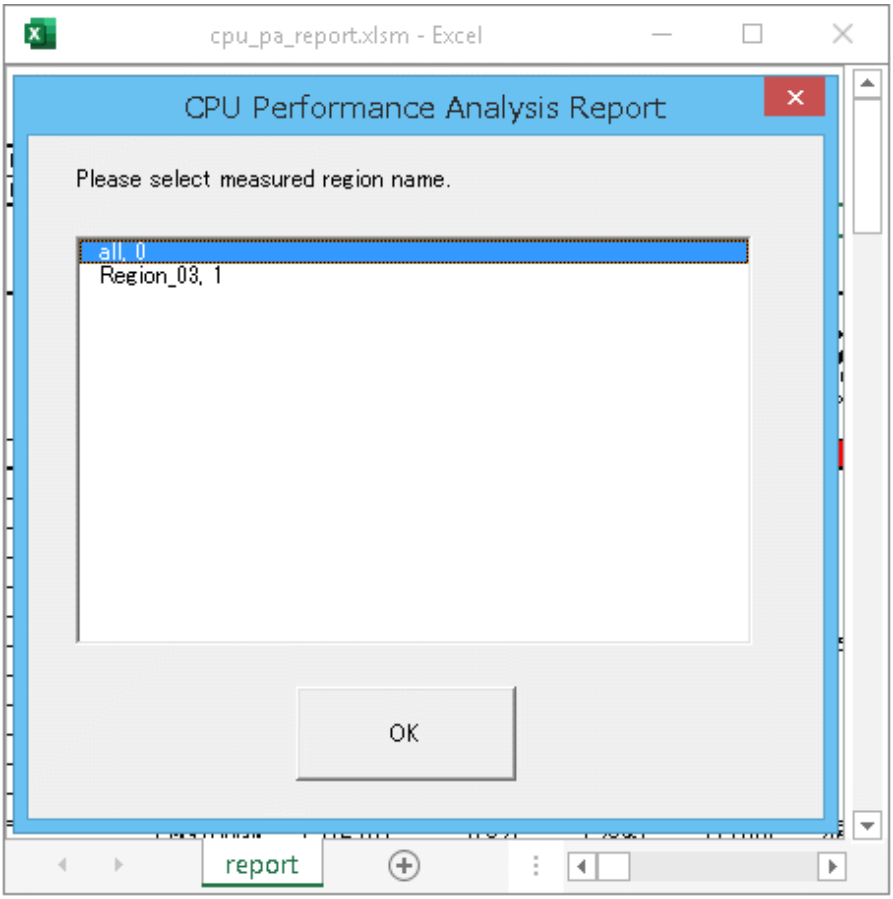
   b. If the specified process is executed in multiple CMGs, the CMG number input window appears. Enter the CMG number you want to reference. If the entered CMG number does not exist in data in pa1.csv, a warning message appears and you are returned to the CMG number input window. If there is only one CMG, the CMG number input window does not appear.

   Figure 4.3 CMG number input window



c. If multiple measurement regions specified by the measurement region specifying routine exist in the specified process, the measurement region selection window shown in the figure below appears. Select the measurement region name you want to reference. If there is only one measurement region, the measurement region selection window does not appear.

Figure 4.4 Measurement region selection window



5. CSV file reading starts. Only files with the names pa1.csv to pa17.csv are to be read, which are located in the same directory as the CPU Performance Analysis Report file. When reading ends normally, the following messages appear for each created report.

Message	Message Description
CPU Performance Analysis Report (Single Report) created.	A CPU Performance Analysis Report (Single Report) is created.
CPU Performance Analysis Report (Brief Report) created.	A CPU Performance Analysis Report (Brief Report) is created.
CPU Performance Analysis Report (Standard Report) created.	A CPU Performance Analysis Report (Standard Report) is created.
CPU Performance Analysis Report (Detail Report) created.	A CPU Performance Analysis Report (Detail Report) is created.

Point

The report type for CPU performance analysis is determined by the CSV files in the directory. The priority for selecting the report type is as follows:

1. If all of the files pa1.csv to pa17.csv exist, a Detail Report is created.
2. If all of the files pa1.csv to pa11.csv exist, a Standard Report is created.
3. If all of the pa1.csv to pa5.csv files exist, a Brief Report is created.

4. If the pa1.csv file exists, a Single Report is generated.

Note

Store only the CSV files required for the report type in the directory. If there is an excess or deficiency of the CSV file, a CPU performance analysis report is created, but the output results are not guaranteed.

Note

The created CPU performance analysis report file is not automatically saved. If necessary, save the file. However, CPU performance analysis report file that has already loaded the CSV files does not load new CSV files. When reusing the saved CPU performance analysis report file, save it with "Save As" instead of "Save".

4.1.6.1 Error and Warning Messages Output by the CPU Performance Analysis Report

The following lists error and warning messages output by the CPU Performance Analysis Report. If an error message appears, the Excel file is closed. If a warning message is output, processing continues. *paXX.csv* specifies the name of the relevant input file.

Table 4.2 Error Messages

Error Message	Error Description
pa1.csv : Cannot open the file.	pa1.csv: Cannot open the file.
paXX.csv : The version of the CPU Performance Analysis Report file does not match that of the fapppx command.	*paXX.csv*: The version of the CPU Performance Analysis Report file does not match that of the fapppx command.
paXX.csv : The file name does not match the argument in the -Hevent option.	*paXX.csv*: The file name does not match the specified value of the -Hevent option.
paXX.csv : Data measured on a static process is not found.	*paXX.csv*: Data from a static process is not found.
paXX.csv : The file format is not supported.	*paXX.csv*: The file format is not supported.
paXX.csv : No data found for specified process number.	*paXX.csv*: No data is found for the specified process number.
paXX.csv : No data found for specified CMG number.	*paXX.csv*: No data is found for the specified CMG number.
paXX.csv : No data found for specified region name.	*paXX.csv*: No data is found for the specified measurement region name.
paXX.csv : The environment seems to not bind process to core.	*paXX.csv*: It contains data that is not CPU binding or that is CPU binding incorrectly.
An unknown error has occurred.	An unknown error has occurred.
Missing input files.	Some input files are missing.

Table 4.3 Warning Messages

Warning Message	Warning Description
Data measured on a dynamically generated process is ignored.	Data from a dynamically generated process is included, but it will be ignored.
No data found for specified process number.	No data is found for the specified process number. (When you are in the process number input window)
No data found for specified CMG number.	No data is found for the specified CMG number. (When you are in the CMG number input window)
paXX.csv : The measured time difference with pa1 exceeds 5%. (XX=XX...)	The execution time for the measurement region difference with pa1 exceeds 5%.

Warning Message	Warning Description
	Note For this message only, the "paXX.csv" string at the beginning of the message is fixed. Only numbers separated by commas are output to (XX = XX...) of the file number for which the difference of measurement time is detected. For example, if you find a difference between the results of pa2.csv and pa5.csv, print (XX = 2,5). If this message is output, the precision of the output may be reduced. It is recommended to remeasure the corresponding CSV format file.
Because of the short measured time, there may be large error in the results. The average measured time of the region is less than 150 microseconds.	Because of the short execution time for the measurement region, there may be many measurement errors in the result. The average execution time per time of the measurement region is less than 150 microseconds.

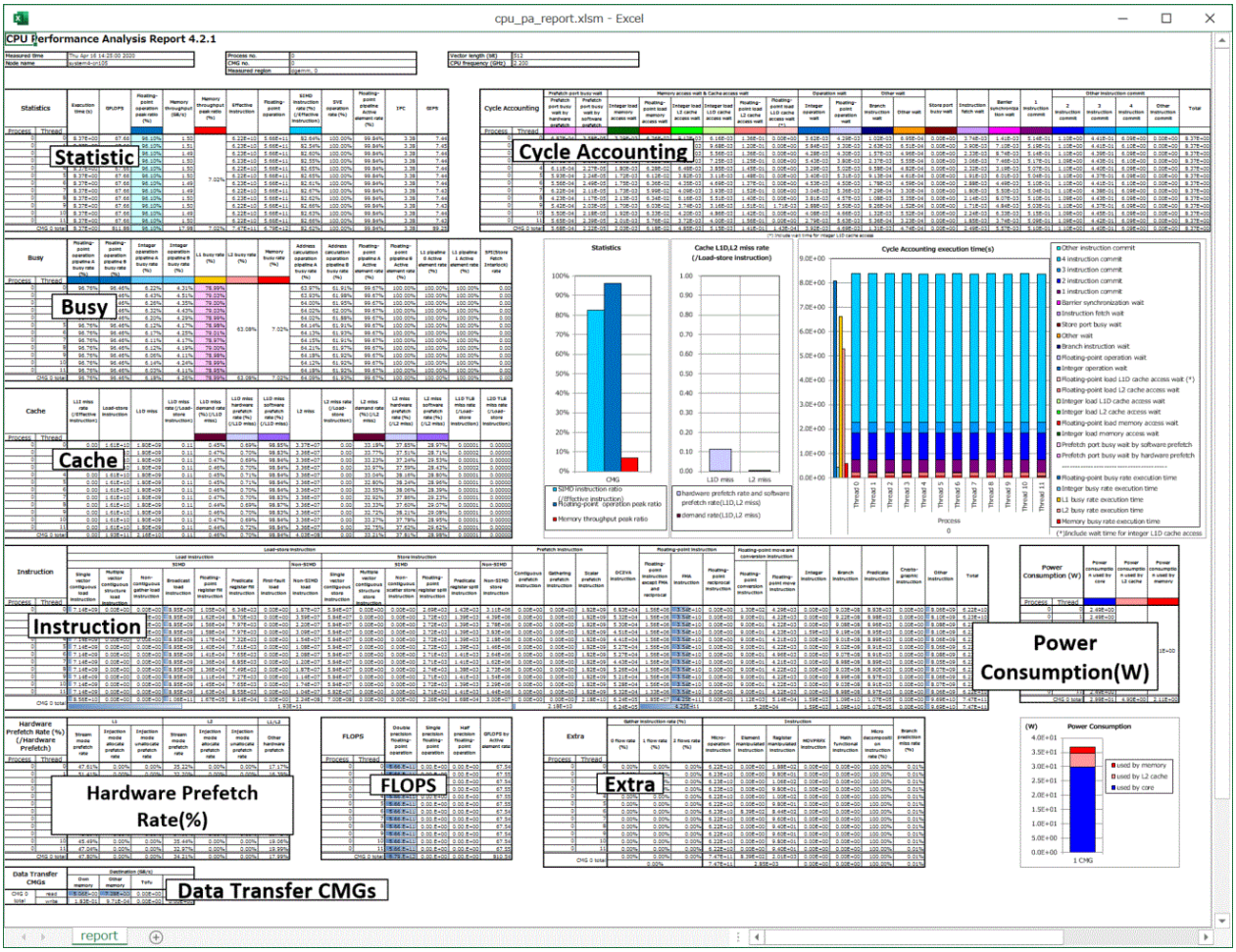
4.2 CPU Performance Analysis Report Output Result

Describes the CPU Performance Analysis Report output result.

4.2.1 Overview of CPU Performance Analysis Report Output Result

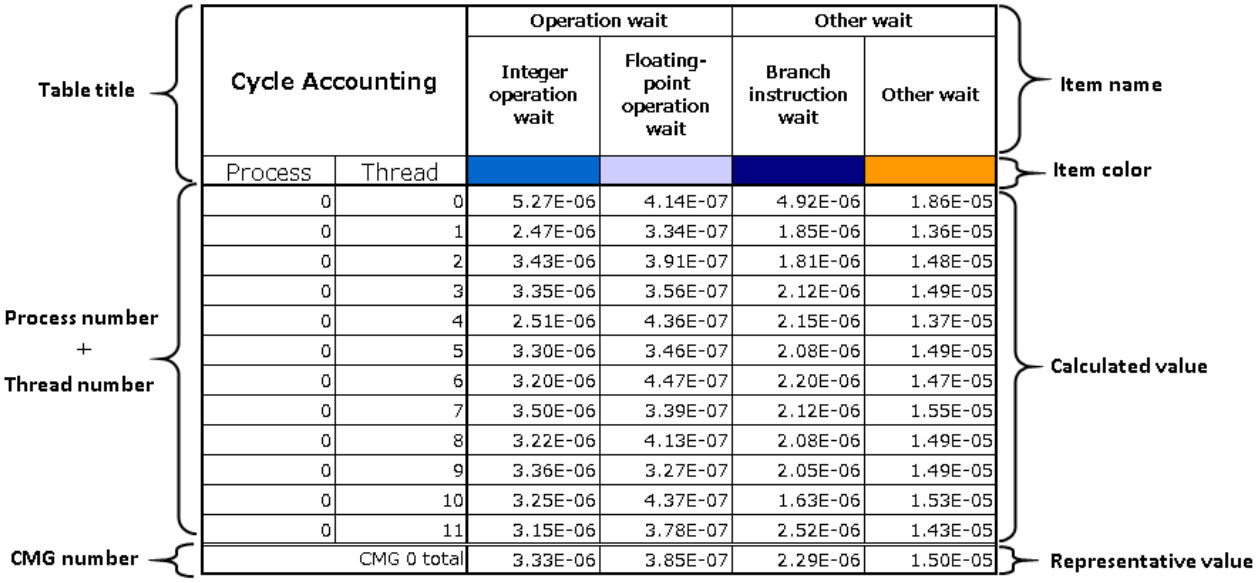
The structure of the CPU Performance Analysis Report is as follows. For a list of information that can be referenced in each table, see "Chapter 4 CPU Performance Analysis Report".

Figure 4.5 Structure of the CPU Performance Analysis Report



The following shows the structure of the respective tables.

Figure 4.6 Table structure



Process	Thread	Integer operation wait	Floating-point operation wait	Branch instruction wait	Other wait
0	0	5.27E-06	4.14E-07	4.92E-06	1.86E-05
0	1	2.47E-06	3.34E-07	1.85E-06	1.36E-05
0	2	3.43E-06	3.91E-07	1.81E-06	1.48E-05
0	3	3.35E-06	3.56E-07	2.12E-06	1.49E-05
0	4	2.51E-06	4.36E-07	2.15E-06	1.37E-05
0	5	3.30E-06	3.46E-07	2.08E-06	1.49E-05
0	6	3.20E-06	4.47E-07	2.20E-06	1.47E-05
0	7	3.50E-06	3.39E-07	2.12E-06	1.55E-05
0	8	3.22E-06	4.13E-07	2.08E-06	1.49E-05
0	9	3.36E-06	3.27E-07	2.05E-06	1.49E-05
0	10	3.25E-06	4.37E-07	1.63E-06	1.53E-05
0	11	3.15E-06	3.78E-07	2.52E-06	1.43E-05
CMG 0 total		3.33E-06	3.85E-07	2.29E-06	1.50E-05

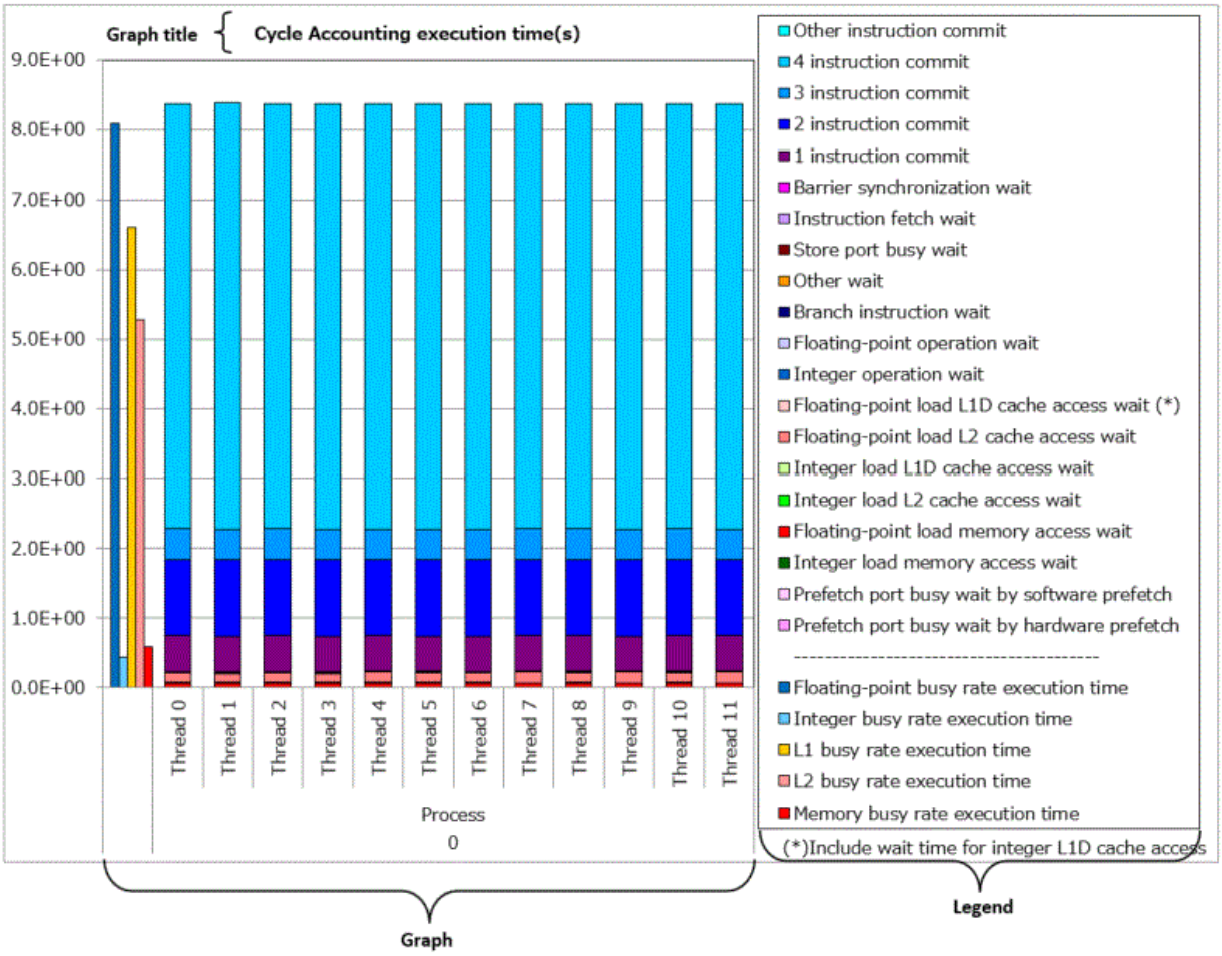
The following describes items in tables.

Table 4.4 Description of items

Name	Description
Table title	Title of the table. For details, see the respective subsections having the same header as the table title in "4.2.2 Detail of CPU Performance Analysis Report Output Result".
Process number + thread number	Shows process and thread numbers.
CMG number	Shows a CMG number.
Item name	Shows the classification and names of items.
Item color	Shows colors in the graph corresponding to the table.
Calculated value	Shows the values of items corresponding to each process and each thread to the second decimal place.
Representative value	Shows the average or total of the calculated values. Whether an average or total value is shown depends on the table. Some tables have no representative values.

Graphs are available for some tables.

Figure 4.7 Graph structure



The following describes items in graphs.

Table 4.5 Description of items

Name	Description
Graph title	Title of the graph. The naming convention is "corresponding table title" + "graph overview".
Graph	Displays a graph representing the contents of the table. The type of the graph depends on the items.
Legend	Legend of the graph. Legend colors in the graph correspond to "item colors" in the table.

4.2.2 Detail of CPU Performance Analysis Report Output Result

This section describes the details of the tables and graphs that are present in the CPU Performance Analysis Report file.

4.2.2.1 Information

Information displays measurement environment information based on the actual measurement results and the contents specified by user.

Figure 4.8 Layout of Information

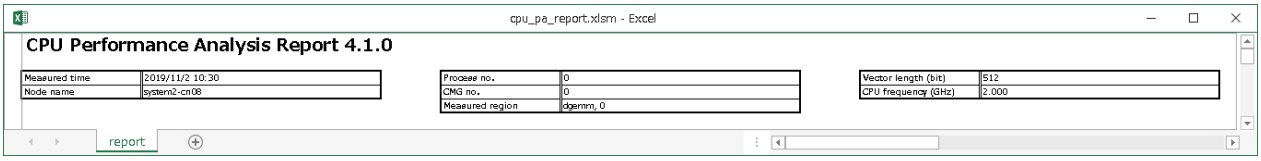


Table 4.6 Output Items in Information

Name	Description
Measured time	Measurement date/time
Node name	Name of the node on which the process is executed
Process no.	Specified process number
CMG no.	Specified CMG number
Measured region	Specified measurement region name
Vector length (bit)	Vector length (bit)
CPU frequency(GHz)	Clock frequency (GHz) of the CPU calculated from actual measurement results **Note** If you specify "4.1.4 Measuring Profile Data" with the -Hmode=user option or -Hmethod=normal option, the value of the CPU frequency (GHz) is not guaranteed. Since the calculation is based on the actual measurement results, it may differ from the CPU frequency of the environment information for measuring profiling data.

4.2.2.2 Statistics

"Statistics" displays the information concerning the CPU behaviors such as the memory throughput, number of instructions, and number of operations. In a brief report, it also shows the ratio of active element in the floating point operation. If the background color in the cell changes to pale blue, it indicates the good operation performance for that item. Statistics has a corresponding table and graph. It displays a bar graph representing the items shown in blue text in Table "Table 4.7 Output Items in Statistics (Single Report)" or "Table 4.8 Output Items in Statistics (Brief, Standard, and Detail Report)".

4.2.2.2.1 Statistics (Single Report)

Figure 4.9 Layout of Statistics (Single Report)

cpu_pa_report.xlsm - Excel

Statistics		Execution time (s)	GFLOPS	Floating-point operation peak ratio (%)	Memory throughput (GB/s)	Memory throughput peak ratio (%)	Effective instruction	Floating-point operation	SIMD instruction rate (%) (/Effective instruction)	SVE operation rate (%)	Floating-point pipeline Active element rate (%)	IPC	GIPS
Process	Thread												
0	0	9.44E+00	59.96	93.68%	1.34	6.21%	6.22E+10	5.66E+11	82.74%	100.00%		3.29	6.58
0	1	9.44E+00	59.95	93.68%	1.33		6.22E+10	5.66E+11	82.64%	100.00%		3.30	6.59
0	2	9.44E+00	59.96	93.68%	1.32		6.22E+10	5.66E+11	82.75%	100.00%		3.29	6.58
0	3	9.44E+00	59.96	93.68%	1.32		6.22E+10	5.66E+11	82.75%	100.00%		3.29	6.58
0	4	9.44E+00	59.95	93.68%	1.32		6.22E+10	5.66E+11	82.75%	100.00%		3.29	6.58
0	5	9.44E+00	59.96	93.68%	1.32		6.22E+10	5.66E+11	82.75%	100.00%		3.29	6.58
0	6	9.44E+00	59.96	93.68%	1.32		6.22E+10	5.66E+11	82.75%	100.00%		3.29	6.58
0	7	9.44E+00	59.96	93.68%	1.33		6.22E+10	5.66E+11	82.75%	100.00%		3.29	6.58
0	8	9.44E+00	59.96	93.68%	1.33		6.22E+10	5.66E+11	82.75%	100.00%		3.29	6.58
0	9	9.44E+00	59.96	93.68%	1.33		6.22E+10	5.66E+11	82.75%	100.00%		3.29	6.58
0	10	9.44E+00	59.96	93.68%	1.32		6.22E+10	5.66E+11	82.75%	100.00%		3.29	6.58
0	11	9.44E+00	59.96	93.68%	1.32		6.22E+10	5.66E+11	82.75%	100.00%		3.29	6.58
CMG 0 total		9.44E+00	719.47	93.68%	15.90	6.21%	7.46E+11	6.79E+12	82.74%	100.00%		3.29	78.99

report

Figure 4.10 Graph of Statistics (Single Report)

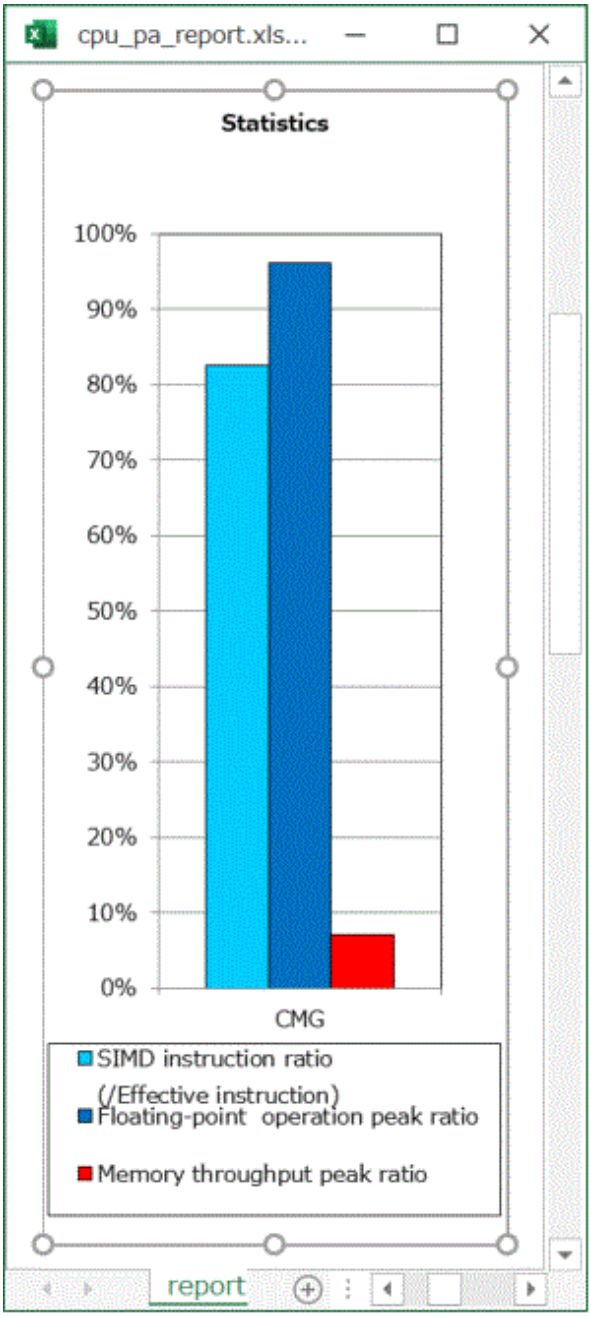


Table 4.7 Output Items in Statistics (Single Report)

Name	Description
Execution time(s)	Time taken to execute instructions in the measurement target region (second)
GFLOPS	Count of floating-point operations performed per second Note The GFLOPS value is calculated by assuming that all elements are active. Therefore, in the case of a program with many inactive elements, a higher value than the original GFLOPS value is output.
Floating-point operation peak ratio(%)	Ratio of the measured value to the theoretical value for floating-point operation performance (%)

Name	Description
	Note The theoretical value for floating-point operation performance is calculated by assuming that double precision operations are performed. Therefore, in the case of single or half precision, a value 2 to 4 times higher than the actual percentage is output. If you specify "4.1.4 Measuring Profile Data" with the -Hmode=user option or -Hmethod=normal option, the value of the Floating-point operation peak ratio(%) is not guaranteed.
Memory throughput(GB/s)	Memory throughput (GB/s)
Memory throughput peak ratio(%)	Ratio of the measured value to the theoretical value for memory throughput (%) **Note** - The theoretical value of memory throughput is for one CMG. If you access not only the memory of own CMG but also the memory of other CMGs at the same time, measured value of memory throughput may exceed the theoretical value. In this case, memory throughput peak ratio may exceed 100%. - The theoretical value for memory throughput assumes simultaneous memory access from all cores in the CMG. However, in fact, each core may perform memory accesses at different times. In such a case, if the memory access amount of all cores is aggregated within the measurement region, memory accesses at different timings are aggregated as simultaneous accesses. As a result, measured value of memory throughput obtained from the memory access amount and execution time may exceed the theoretical value. The memory throughput peak ratio may exceed 100%.
Effective instruction	Total number of instructions executed **Note** The total number of instructions executed does not include MOVPRFX instructions.
Floating-point operation	Total number of floating-point operations executed
SIMD Instruction rate(%) (/Effective Instruction)	Ratio of the number of SIMD instructions to the total number of instructions executed (%)
SVE operation rate(%)	Ratio of the number of SVE operations to the total number of floating-point operations executed (%)
IPC	Number of instructions executed per cycle
GIPS	Number of instructions executed per second

4.2.2.2.2 Statistics (Brief, Standard, and Detail Report)

Figure 4.11 Layout of Statistics (Brief, Standard, and Detail Report)

cpu_pa_report.xlsm - Excel

Statistics		Execution time (s)	GFLOPS	Floating-point operation peak ratio (%)	Memory throughput (GB/s)	Memory throughput peak ratio (%)	Effective instruction	Floating-point operation	SIMD instruction rate (%) (/Effective instruction)	SVE operation rate (%)	Floating-point pipeline Active element rate (%)	IPC	GIPS
Process	Thread												
0	0	9.50E+00	59.63	93.17%	1.33	6.18%	6.23E+10	5.66E+11	82.60%	100.00%	99.84%	3.28	6.56
0	1	9.50E+00	59.63	93.17%	1.33		6.22E+10	5.66E+11	82.74%	100.00%	99.84%	3.27	6.55
0	2	9.50E+00	59.63	93.17%	1.32		6.22E+10	5.66E+11	82.74%	100.00%	99.84%	3.27	6.55
0	3	9.50E+00	59.63	93.17%	1.32		6.22E+10	5.66E+11	82.74%	100.00%	99.84%	3.27	6.55
0	4	9.50E+00	59.63	93.17%	1.31		6.22E+10	5.66E+11	82.74%	100.00%	99.84%	3.27	6.55
0	5	9.50E+00	59.63	93.17%	1.31		6.22E+10	5.66E+11	82.74%	100.00%	99.84%	3.27	6.55
0	6	9.50E+00	59.63	93.17%	1.32		6.22E+10	5.66E+11	82.74%	100.00%	99.84%	3.27	6.55
0	7	9.50E+00	59.63	93.17%	1.31		6.22E+10	5.66E+11	82.74%	100.00%	99.84%	3.27	6.55
0	8	9.50E+00	59.63	93.17%	1.32		6.22E+10	5.66E+11	82.74%	100.00%	99.84%	3.27	6.55
0	9	9.50E+00	59.63	93.17%	1.32		6.22E+10	5.66E+11	82.74%	100.00%	99.84%	3.27	6.55
0	10	9.50E+00	59.63	93.17%	1.31		6.22E+10	5.66E+11	82.74%	100.00%	99.84%	3.27	6.55
0	11	9.50E+00	59.63	93.17%	1.32		6.22E+10	5.66E+11	82.75%	100.00%	99.84%	3.27	6.55
CMG 0 total		9.50E+00	715.53	93.17%	15.82	6.18%	7.46E+11	6.79E+12	82.73%	100.00%	99.84%	3.27	78.56

report

Figure 4.12 Graph of Statistics (Brief, Standard, and Detail Report)

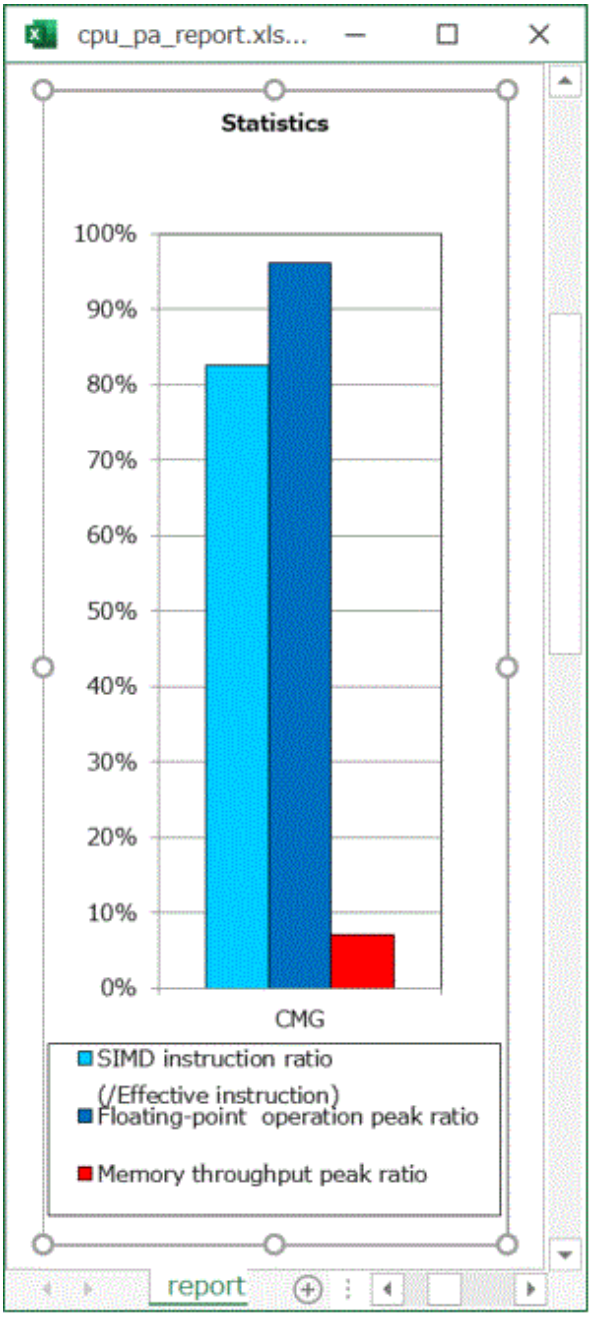


Table 4.8 Output Items in Statistics (Brief, Standard, and Detail Report)

Name	Description
Execution time(s)	Time taken to execute instructions in the measurement target region (second)
GFLOPS	Count of floating-point operations performed per second Note The GFLOPS value is calculated by assuming that all elements are active. Therefore, in the case of a program with many inactive elements, a higher value than the original GFLOPS value is output.
Floating-point operation peak ratio(%)	Ratio of the measured value to the theoretical value for floating-point operation performance (%)

Name	Description
	Note The theoretical value for floating-point operation performance is calculated by assuming that double precision operations are performed. Therefore, in the case of single or half precision, a value 2 to 4 times higher than the actual percentage is output. If you specify "4.1.4 Measuring Profile Data" with the -Hmode=user option or -Hmethod=normal option, the value of the Floating-point operation peak ratio(%) is not guaranteed.
Memory throughput(GB/s)	Memory throughput (GB/s)
Memory throughput peak ratio(%)	Ratio of the measured value to the theoretical value for memory throughput (%) **Note** - The theoretical value of memory throughput is for one CMG. If you access not only the memory of own CMG but also the memory of other CMGs at the same time, measured value of memory throughput may exceed the theoretical value. In this case, memory throughput peak ratio may exceed 100%. - The theoretical value for memory throughput assumes simultaneous memory access from all cores in the CMG. However, in fact, each core may perform memory accesses at different times. In such a case, if the memory access amount of all cores is aggregated within the measurement region, memory accesses at different timings are aggregated as simultaneous accesses. As a result, measured value of memory throughput obtained from the memory access amount and execution time may exceed the theoretical value. The memory throughput peak ratio may exceed 100%.
Effective instruction	Total number of instructions executed **Note** The total number of instructions executed does not include MOVPRFX instructions.
Floating-point operation	Total number of floating-point operations executed
SIMD Instruction rate(%) (/Effective Instruction)	Ratio of the number of SIMD instructions to the total number of instructions executed (%)
SVE operation rate(%)	Ratio of the number of SVE operations to the total number of floating-point operations executed (%)
Floating-point pipeline Active element rate(%)	Ratio of active elements in floating-point operations (%) **Note** - Except for the SVE instruction, the percentage of the Active element is converted to 100%. In cases of many non-SVE instructions, a higher ratio than that of the original Active element is output. You should limitedly use it only when the ratio of SVE instructions is close to 100%. - If the ratio of store instructions is high, the ratio of active elements may degrade. We recommend to limitedly use it only when the ratio of store instructions is low. - In the one-byte data type operation, the percentage of the Active element is converted to 100%. In cases of one-byte data type operation, a higher ratio than that of the original Active element is output. You should limitedly use it only when the ratio of one-byte data type operation is low.
IPC	Number of instructions executed per cycle

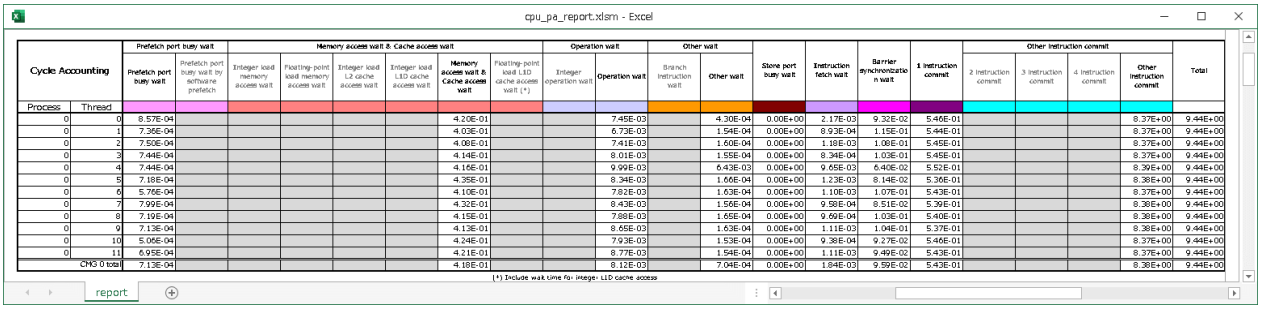
Name	Description
GIPS	Number of instructions executed per second

4.2.2.3 Cycle Accounting

Cycle Accounting displays a breakdown of program execution time (unit: second). The Brief Report displays the information by classifying it into nine types. The Standard and Detail Reports display the information by classifying it into 20 types. Cycle Accounting has a corresponding table and graph. It displays a stacked bar graph representing Table "Table 4.9 Output Items in Cycle Accounting (Brief Report)" or output items (excluding Total) in "Table 4.10 Output Items in Cycle Accounting (Standard and Detail Reports)". In addition, for the purpose of comparison, it displays a bar graph representing the execution times converted from the values of the following output items in "4.2.2.4 Busy": "L1 busy rate(%)", "L2 busy rate(%)", "Memory busy rate(%)", "Floating-point operation pipeline busy rate(%)", "Integer operation pipeline busy rate(%)".

4.2.2.3.1 Cycle Accounting (Brief Report)

Figure 4.13 Layout of Cycle Accounting (Brief Report)



cpu_pa_report.xlsm - Excel

Cycle Accounting		Prefetch port busy wait		Memory access wait & Cache access wait						Operation wait		Other wait							Other instruction commit				Total
		Prefetch port busy wait	Prefetch port busy wait by software prefetch	Integer load memory access wait	Floating-point load memory access wait	Integer load L2 cache access wait	Integer load L1D cache access wait	Memory access wait & Cache access wait	Floating-point load L1D cache access wait (*)	Integer operation wait	Operation wait	Branch instruction wait	Other wait	Store port busy wait	Instruction fetch wait	Barrier synchronization wait	1 instruction commit	2 instruction commit	3 instruction commit	4 instruction commit	Other instruction commit	
Process	Thread																						
0	0	8.57E-04						4.20E-01			7.45E-03		4.30E-04	0.00E+00	2.17E-03	9.32E-02	5.46E-01				8.37E+00	9.44E+00
0	1	7.36E-04						4.03E-01			6.73E-03		1.54E-04	0.00E+00	8.93E-04	1.15E-01	5.44E-01				8.37E+00	9.44E+00
0	2	7.50E-04						4.08E-01			7.41E-03		1.60E-04	0.00E+00	1.18E-03	1.08E-01	5.45E-01				8.37E+00	9.44E+00
0	3	7.44E-04						4.14E-01			8.01E-03		1.55E-04	0.00E+00	8.34E-04	1.03E-01	5.45E-01				8.37E+00	9.44E+00
0	4	7.44E-04						4.16E-01			9.99E-03		6.43E-03	0.00E+00	9.65E-03	6.40E-02	5.52E-01				8.39E+00	9.44E+00
0	5	7.18E-04						4.35E-01			8.34E-03		1.66E-04	0.00E+00	1.23E-03	8.14E-02	5.36E-01				8.38E+00	9.44E+00
0	6	5.76E-04						4.10E-01			7.82E-03		1.63E-04	0.00E+00	1.10E-03	1.07E-01	5.43E-01				8.37E+00	9.44E+00
0	7	7.99E-04						4.32E-01			8.43E-03		1.56E-04	0.00E+00	9.58E-04	8.51E-02	5.39E-01				8.38E+00	9.44E+00
0	8	7.19E-04						4.15E-01			7.88E-03		1.65E-04	0.00E+00	9.69E-04	1.03E-01	5.40E-01				8.38E+00	9.44E+00
0	9	7.13E-04						4.13E-01			8.65E-03		1.63E-04	0.00E+00	1.11E-03	1.04E-01	5.37E-01				8.38E+00	9.44E+00
0	10	5.06E-04						4.24E-01			7.93E-03		1.53E-04	0.00E+00	9.38E-04	9.27E-02	5.46E-01				8.37E+00	9.44E+00
0	11	6.95E-04						4.21E-01			8.77E-03		1.54E-04	0.00E+00	1.11E-03	9.49E-02	5.43E-01				8.37E+00	9.44E+00
CMG 0 total		7.13E-04						4.18E-01			8.12E-03		7.04E-04	0.00E+00	1.84E-03	9.59E-02	5.43E-01				8.38E+00	9.44E+00

(*) Include wait time for integer L1D cache access

report

Figure 4.14 Graph of Cycle Accounting (Brief Report)

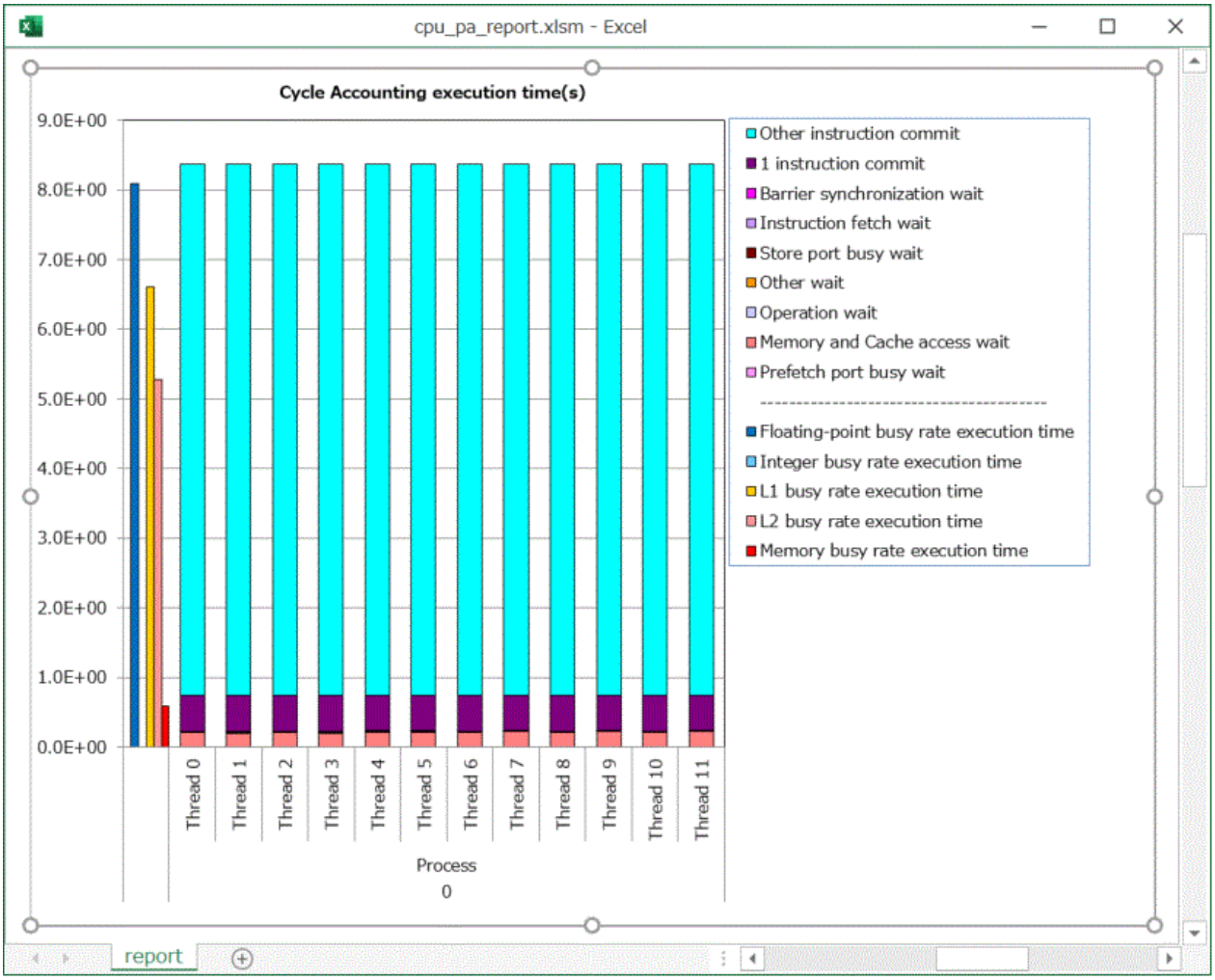


Table 4.9 Output Items in Cycle Accounting (Brief Report)

Name	Description
Prefetch port busy wait	Time during which the number of committed instructions was 0 because the prefetch port was busy
Memory access wait & Cache access wait	Time during which the number of committed instructions was 0 because of memory access wait and cache access wait
Operation wait	Time during which the number of committed instructions was 0 because the oldest instruction among instructions in execution was in execution of an operation
Other wait	Time during which the number of committed instruction was 0 due to other factors
Store port busy wait	Time during which the number of committed instructions was 0 because the store port was full
Instruction fetch wait	Time during which the number of committed instructions was 0 because of wait for instruction load
Barrier synchronization wait	Time during which the number of committed instructions was 0 because the instruction controller was stopped due to a WFE or WFI instruction
1 instruction commit	Time during which 1 instruction was executed in 1 cycle
Other instruction commit	Time during which 2 to 8 instructions were executed in 1 cycle
Total	Total time

4.2.2.3.2 Cycle Accounting (Standard and Detail Reports)

Figure 4.15 Layout of Cycle Accounting (Standard and Detail Reports)

cpu_pa_report.xlsm - Excel

Cycle Accounting		Prefetch port busy wait		Memory access wait & Cache access wait						Operation wait		Other wait						Other instruction commit				Total
		Prefetch port busy wait by hardware prefetch	Prefetch port busy wait by software prefetch	Integer load memory access wait	Floating-point load memory access wait	Integer load L2 cache access wait	Integer load L1D cache access wait	Floating-point load L2 cache access wait	Floating-point load L1D cache access wait (*)	Integer operation wait	Floating-point operation wait	Branch instruction wait	Other wait	Store port busy wait	Instruction fetch wait	Barrier synchronization wait	1 instruction commit	2 instruction commit	3 instruction commit	4 instruction commit	Other instruction commit	
Process	Thread																					
0	0	8.10E-04	4.69E-05	2.88E-03	2.66E-01	3.85E-03	3.05E-03	1.24E-01	2.01E-02	2.09E-03	5.37E-03	1.87E-04	2.43E-04	0.00E+00	2.17E-03	9.32E-02	5.46E-01	1.22E+00	4.71E-01	6.69E+00	0.00E+00	9.44E+00
0	1	7.10E-04	2.68E-05	1.07E-03	2.89E-01	2.83E-03	9.78E-04	1.11E-01	0.00E+00	2.09E-03	4.64E-03	7.36E-05	8.02E-05	0.00E+00	8.93E-04	1.15E-01	5.44E-01	1.23E+00	4.68E-01	6.69E+00	0.00E+00	9.44E+00
0	2	7.12E-04	3.86E-05	1.13E-03	2.79E-01	3.12E-03	9.69E-04	1.25E-01	0.00E+00	2.02E-03	5.39E-03	6.58E-05	9.41E-05	0.00E+00	1.18E-03	1.08E-01	5.45E-01	1.22E+00	4.70E-01	6.69E+00	0.00E+00	9.44E+00
0	3	7.07E-04	3.79E-05	1.10E-03	2.71E-01	3.19E-03	9.91E-04	1.33E-01	4.39E-03	1.99E-03	6.02E-03	7.13E-05	8.36E-05	0.00E+00	8.34E-04	1.03E-01	5.45E-01	1.22E+00	4.73E-01	6.69E+00	0.00E+00	9.44E+00
0	4	7.06E-04	3.73E-05	1.35E-03	2.71E-01	3.18E-03	1.04E-03	1.20E-01	2.05E-02	4.65E-03	5.34E-03	5.13E-03	1.30E-03	0.00E+00	9.65E-03	6.40E-02	5.52E-01	1.21E+00	4.83E-01	6.69E+00	0.00E+00	9.44E+00
0	5	6.90E-04	2.80E-05	9.81E-04	2.64E-01	3.58E-03	1.02E-03	1.40E-01	2.65E-02	1.93E-03	6.40E-03	6.55E-05	1.00E-04	0.00E+00	1.23E-03	8.14E-02	5.36E-01	1.22E+00	4.71E-01	6.69E+00	8.31E-04	9.44E+00
0	6	5.51E-04	2.42E-05	1.20E-03	2.88E-01	3.27E-03	1.07E-03	1.31E-01	0.00E+00	2.06E-03	5.76E-03	6.70E-05	9.57E-05	0.00E+00	1.10E-03	1.07E-01	5.43E-01	1.22E+00	4.74E-01	6.69E+00	0.00E+00	9.44E+00
0	7	7.62E-04	3.74E-05	1.63E-03	2.72E-01	3.55E-03	8.91E-03	1.43E-01	2.42E-03	1.99E-03	6.44E-03	6.94E-05	8.64E-05	0.00E+00	9.58E-04	8.51E-02	5.39E-01	1.23E+00	4.69E-01	6.69E+00	0.00E+00	9.44E+00
0	8	6.94E-04	2.51E-05	1.09E-03	2.77E-01	3.42E-03	9.88E-04	1.33E-01	0.00E+00	2.09E-03	5.79E-03	6.74E-05	9.73E-05	0.00E+00	9.69E-04	1.03E-01	5.40E-01	1.21E+00	4.75E-01	6.69E+00	6.97E-04	9.44E+00
0	9	6.83E-04	2.94E-05	1.41E-03	2.83E-01	3.59E-03	1.03E-03	1.44E-01	0.00E+00	2.16E-03	6.48E-03	6.28E-05	1.01E-04	0.00E+00	1.11E-03	1.04E-01	5.37E-01	1.22E+00	4.69E-01	6.69E+00	0.00E+00	9.44E+00
0	10	4.82E-04	2.33E-05	1.21E-03	2.66E-01	3.38E-03	1.01E-03	1.39E-01	1.36E-02	2.00E-03	5.93E-03	6.79E-05	8.53E-05	0.00E+00	9.38E-04	9.27E-02	5.46E-01	1.21E+00	4.79E-01	6.69E+00	2.30E-05	9.44E+00
0	11	6.64E-04	3.13E-05	1.37E-03	2.68E-01	3.40E-03	1.00E-03	1.48E-01	0.00E+00	2.05E-03	6.72E-03	6.57E-05	8.87E-05	0.00E+00	1.11E-03	9.49E-02	5.43E-01	1.22E+00	4.71E-01	6.69E+00	0.00E+00	9.44E+00
CHS 0 total		6.81E-04	3.22E-05	1.37E-03	2.75E-01	3.36E-03	1.84E-03	1.33E-01	7.30E-03	2.26E-03	5.86E-03	4.99E-04	2.05E-04	0.00E+00	1.84E-03	9.59E-02	5.43E-01	1.22E+00	4.73E-01	6.69E+00	1.29E-04	9.44E+00

(*) Include wait time for integer L1D cache access

report

Figure 4.16 Graph of Cycle Accounting (Standard and Detail Reports)

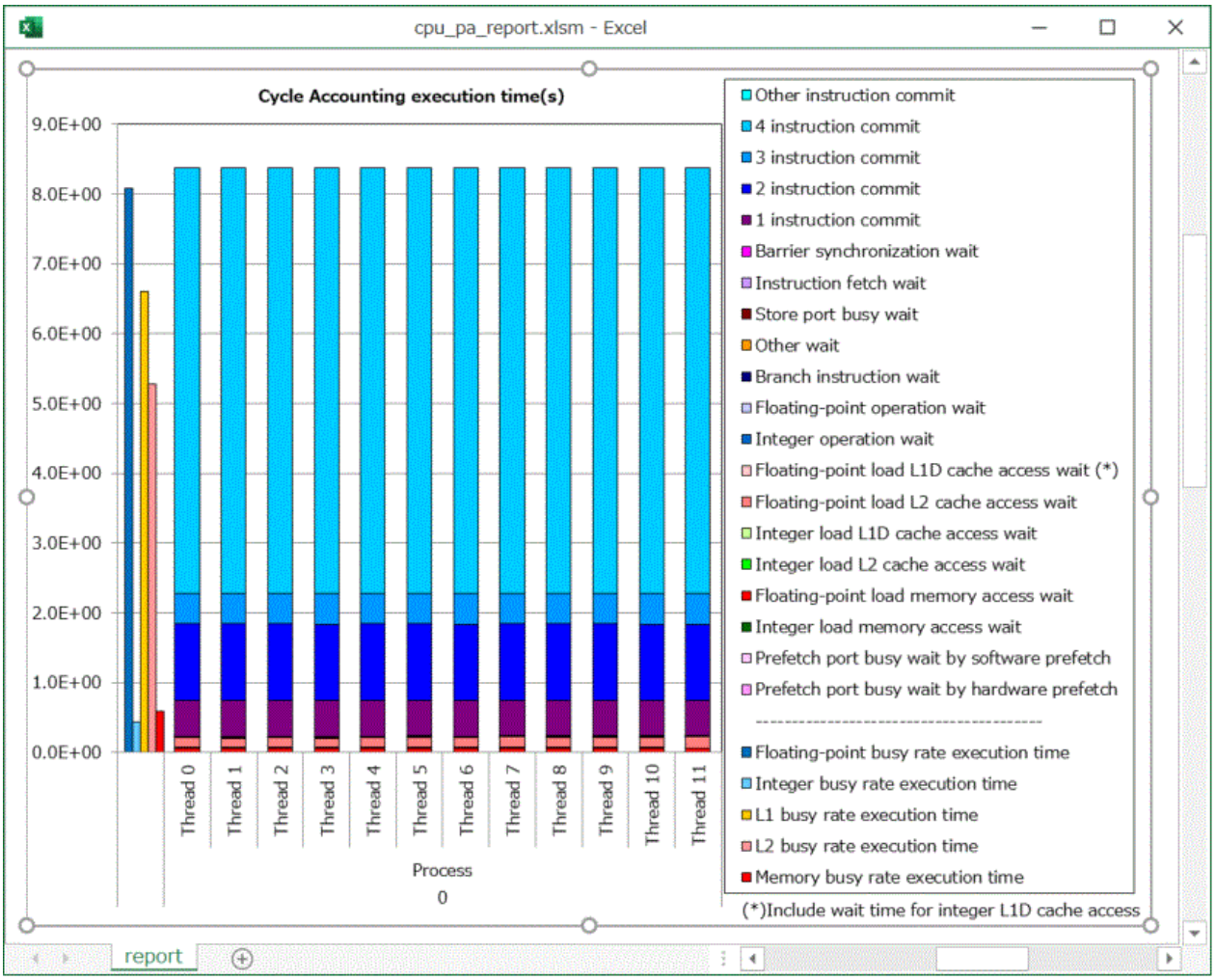


Table 4.10 Output Items in Cycle Accounting (Standard and Detail Reports)

Name	Description
Prefetch port busy wait by hardware prefetch	Time during which the number of committed instructions was 0 because of prefetch port busy wait due to hardware prefetch
Prefetch port busy wait by software prefetch	Time during which the number of committed instructions was 0 because of prefetch port busy wait due to software prefetch
Integer load memory access wait	Time during which the number of committed instructions was 0 because the oldest instruction among instructions in execution was waiting for data due to memory access for integer load

Name	Description
Floating-point load memory access wait	Time during which the number of committed instructions was 0 because the oldest instruction among instructions in execution was waiting for data due to memory access for floating-point load
Integer load L2 cache access wait	Time during which the number of committed instructions was 0 because the oldest instruction among instructions in execution was waiting for data due to secondary cache access for integer load
Integer load L1D cache access wait	Time during which the number of committed instructions was 0 because the oldest instruction among instructions in execution was waiting for data due to primary data cache access for integer load
Floating-point load L2 cache access wait	Time during which the number of committed instructions was 0 because the oldest instruction among instructions in execution was waiting for data due to secondary cache access for floating-point load
Floating-point load L1D cache access wait	Time during which the number of committed instructions was 0 because the oldest instruction among instructions in execution was waiting for data due to primary data cache access for floating-point load **Note** Floating-point load L1D cache access wait may include wait time for L1D cache access for general-purpose registers.
Integer operation wait	Time during which the number of committed instructions was 0 because the oldest instruction among instructions in execution was in an integer operation execution
Floating-point operation wait	Time during which the number of committed instructions was 0 because the oldest instruction among instructions in execution was in a floating-point operation execution
Branch instruction wait	Time during which the number of committed instructions was 0 because the oldest instruction among instructions in execution was in a branching execution
Other wait	Time during which the number of committed instructions was 0 due to other factors
Store port busy wait	Time during which the number of committed instructions was 0 because the store port was full
Instruction fetch wait	Time during which the number of committed instructions was 0 because of wait for instruction load
Barrier synchronization wait	Time during which the number of committed instructions was 0 because the instruction controller was stopped due to a WFE or WFI instruction
1 instruction commit	Time during which 1 instruction was executed in 1 cycle
2 instruction commit	Time during which 2 instructions were executed in 1 cycle
3 instruction commit	Time during which 3 instructions were executed in 1 cycle
4 instruction commit	Time during which 4 instructions were executed in 1 cycle
Other instruction commit	Time during which 5 to 8 instructions were executed in 1 cycle
Total	Total time

4.2.2.4 Busy

Busy displays the information on the busy rate for the program memory cache and operation pipeline. In a brief report, it shows the busy rates such as for the primary cache, secondary cache, memory, and floating-point operation pipeline, as well as the occurrence rate of SFI (Store Fetch Interlock). In a standard report, in addition to the contents of the brief report, it shows the busy rates for the integer operation pipeline, address calculation operation pipeline, and predicate operation pipeline. In a detail report, in addition to the contents of the standard report, it shows the ratio of the active element in the L1 pipeline. If the background color in the cell changes to pink, it indicates that item may be the performance bottleneck.

4.2.2.4.1 Busy (Brief Report)

Figure 4.17 Layout of Busy (Brief Report)

cpu_pa_report.xlsm - Excel

Busy		Floating-point operation pipeline A busy rate (%)	Floating-point operation pipeline B busy rate (%)	Integer operation pipeline A busy rate (%)	Integer operation pipeline B busy rate (%)	L1 busy rate (%)	L2 busy rate (%)	Memory busy rate (%)	Address calculation operation pipeline A busy rate (%)	Address calculation operation pipeline B busy rate (%)	Floating-point pipeline A Active element rate (%)	Floating-point pipeline B Active element rate (%)	L1 pipeline 0 Active element rate (%)	L1 pipeline 1 Active element rate (%)	SFI(Store Fetch Interlock) rate
Process	Thread														
0	0	94.02%	93.74%			76.78%	61.13%	6.21%			99.67%	100.00%			0.00
0	1	94.02%	93.74%			76.73%					99.67%	100.00%			0.00
0	2	94.02%	93.74%			76.75%					99.67%	100.00%			0.00
0	3	94.02%	93.74%			76.76%					99.67%	100.00%			0.00
0	4	94.02%	93.74%			76.85%					99.67%	100.00%			0.00
0	5	94.02%	93.74%			76.79%					99.67%	100.00%			0.00
0	6	94.02%	93.74%			76.76%					99.67%	100.00%			0.00
0	7	94.02%	93.74%			76.79%					99.67%	100.00%			0.00
0	8	94.02%	93.74%			76.75%					99.67%	100.00%			0.00
0	9	94.02%	93.74%			76.78%					99.67%	100.00%			0.00
0	10	94.02%	93.74%			76.77%					99.67%	100.00%			0.00
0	11	94.02%	93.74%			76.79%					99.67%	100.00%			0.00
CMG 0 total		94.02%	93.74%			76.78%	61.13%	6.21%			99.67%	100.00%			0.00

report

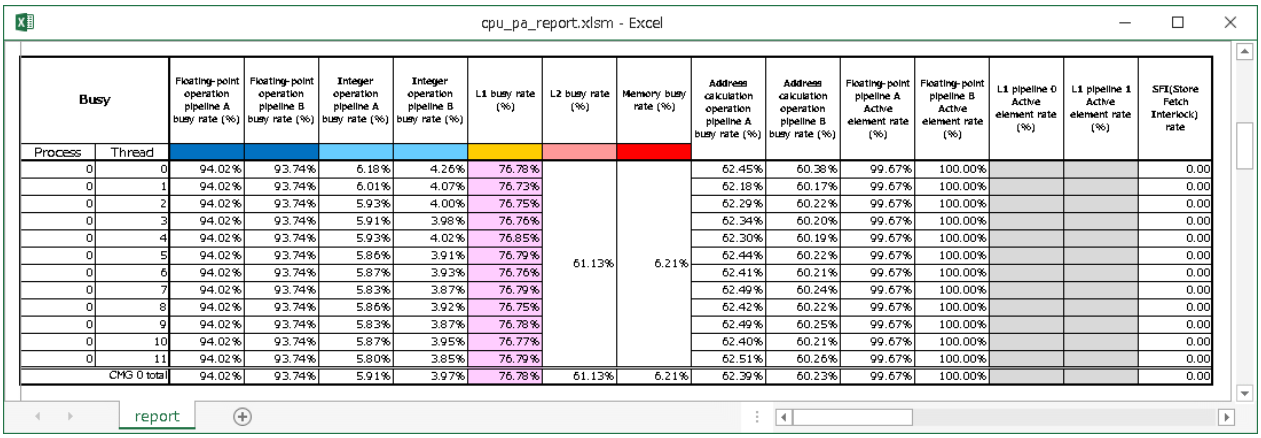
Table 4.11 Output Items in Busy (Brief Report)

Name	Description
Floating-point operation pipeline A busy rate(%)	Busy rate for the floating-point operation pipeline A (%)
Floating-point operation pipeline B busy rate(%)	Busy rate for the floating-point operation pipeline B (%)
L1 busy rate(%)	Busy rate for the primary cache (%)
L2 busy rate(%)	Busy rate for the secondary cache (%)
Memory busy rate(%)	Memory busy rate (%) **Note** - The theoretical value of memory throughput is for one CMG. If you access not only the memory of own CMG but also the memory of other CMGs at the same time, measured value of memory throughput may exceed the theoretical value. In this case, memory busy ratio (same as memory throughput peak ratio) may exceed 100%. - The theoretical value for memory throughput assumes simultaneous memory access from all cores in the CMG. However, in fact, each core may perform memory accesses at different times. In such a case, if the memory access amount of all cores is aggregated within the measurement region, memory accesses at different timings are aggregated as simultaneous accesses. As a result, measured value of memory throughput obtained from the memory access amount and execution time may exceed the theoretical value. The memory busy ratio (same as memory throughput peak ratio) may exceed 100%.
Floating-point pipeline A Active element rate(%)	Active element ratio in the floating-point operation pipeline A (%) **Note** - Except for the SVE instruction, the percentage of the Active element is converted to 100%. In cases of many non-SVE instructions, a higher ratio than that of the original Active element is output. You should limitedly use it only when the ratio of SVE instructions is high. - If the ratio of store instructions is high, the ratio of active elements may degrade. We recommend to limitedly use it only when the ratio of store instructions is low.

Name	Description
	- In the one-byte data type operation, the percentage of the Active element is converted to 100%. In cases of one-byte data type operation, a higher ratio than that of the original Active element is output. You should limitedly use it only when the ratio of one-byte data type operation is low.
Floating-point pipeline B Active element rate(%)	Active element ratio in the floating-point operation pipeline B (%) **Note** - Except for the SVE instruction, the percentage of the Active element is converted to 100%. In cases of many non-SVE instructions, a higher ratio than that of the original Active element is output. You should limitedly use it only when the ratio of SVE instructions is high. - If the ratio of store instructions is high, the ratio of active elements may degrade. We recommend to limitedly use it only when the ratio of store instructions is low. - In the one-byte data type operation, the percentage of the Active element is converted to 100%. In cases of one-byte data type operation, a higher ratio than that of the original Active element is output. You should limitedly use it only when the ratio of one-byte data type operation is low.
SFI(Store Fetch Interlock) rate	Queuing time ratio by SFI (Store Fetch Interlock)

4.2.2.4.2 Busy (Standard Report)

Figure 4.18 Layout of Busy (Standard Report)



cpu_pa_report.xlsm - Excel

Busy		Floating-point operation pipeline A busy rate (%)	Floating-point operation pipeline B busy rate (%)	Integer operation pipeline A busy rate (%)	Integer operation pipeline B busy rate (%)	L1 busy rate (%)	L2 busy rate (%)	Memory busy rate (%)	Address calculation operation pipeline A busy rate (%)	Address calculation operation pipeline B busy rate (%)	Floating-point pipeline A Active element rate (%)	Floating-point pipeline B Active element rate (%)	L1 pipeline 0 Active element rate (%)	L1 pipeline 1 Active element rate (%)	SFI(Store Fetch Interlock) rate
Process	Thread														
0	0	94.02%	93.74%	6.18%	4.26%	76.78%	61.13%	6.21%	62.45%	60.38%	99.67%	100.00%			0.00
0	1	94.02%	93.74%	6.01%	4.07%	76.73%			62.18%	60.17%	99.67%	100.00%			0.00
0	2	94.02%	93.74%	5.93%	4.00%	76.75%			62.29%	60.22%	99.67%	100.00%			0.00
0	3	94.02%	93.74%	5.91%	3.98%	76.76%			62.34%	60.20%	99.67%	100.00%			0.00
0	4	94.02%	93.74%	5.93%	4.02%	76.85%			62.30%	60.19%	99.67%	100.00%			0.00
0	5	94.02%	93.74%	5.86%	3.91%	76.79%			62.44%	60.22%	99.67%	100.00%			0.00
0	6	94.02%	93.74%	5.87%	3.93%	76.76%			62.41%	60.21%	99.67%	100.00%			0.00
0	7	94.02%	93.74%	5.83%	3.87%	76.79%			62.49%	60.24%	99.67%	100.00%			0.00
0	8	94.02%	93.74%	5.86%	3.92%	76.75%			62.42%	60.22%	99.67%	100.00%			0.00
0	9	94.02%	93.74%	5.83%	3.87%	76.78%			62.49%	60.25%	99.67%	100.00%			0.00
0	10	94.02%	93.74%	5.87%	3.95%	76.77%			62.40%	60.21%	99.67%	100.00%			0.00
0	11	94.02%	93.74%	5.80%	3.85%	76.79%			62.51%	60.26%	99.67%	100.00%			0.00
CMG 0 total		94.02%	93.74%	5.91%	3.97%	76.78%	61.13%	6.21%	62.39%	60.23%	99.67%	100.00%			0.00

report

Table 4.12 Output Items in Busy (Standard Report)

Name	Description
Floating-point operation pipeline A busy rate(%)	Busy rate for the floating-point operation pipeline A (%)
Floating-point operation pipeline B busy rate(%)	Busy rate for the floating-point operation pipeline B (%)
Integer operation pipeline A busy rate(%)	Busy rate for the integer operation pipeline A (%)
Integer operation pipeline B busy rate(%)	Busy rate for the integer operation pipeline B (%)

Name	Description
L1 busy rate(%)	Busy rate for the primary cache (%)
L2 busy rate(%)	Busy rate for the secondary cache (%)
Memory busy rate(%)	Memory busy rate (%) Note - The theoretical value of memory throughput is for one CMG. If you access not only the memory of own CMG but also the memory of other CMGs at the same time, measured value of memory throughput may exceed the theoretical value. In this case, memory busy ratio (same as memory throughput peak ratio) may exceed 100%. - The theoretical value for memory throughput assumes simultaneous memory access from all cores in the CMG. However, in fact, each core may perform memory accesses at different times. In such a case, if the memory access amount of all cores is aggregated within the measurement region, memory accesses at different timings are aggregated as simultaneous accesses. As a result, measured value of memory throughput obtained from the memory access amount and execution time may exceed the theoretical value. The memory busy ratio (same as memory throughput peak ratio) may exceed 100%.
Address calculation operation pipeline A busy rate(%)	Busy rate for the address calculation operation pipeline A (%)
Address calculation operation pipeline B busy rate(%)	Busy rate for the address calculation operation pipeline B (%)
Floating-point pipeline A Active element rate(%)	Active element ratio in the floating-point operation pipeline A (%) Note - Except for the SVE instruction, the percentage of the Active element is converted to 100%. In cases of many non-SVE instructions, a higher ratio than that of the original Active element is output. You should limitedly use it only when the ratio of SVE instructions is high. - If the ratio of store instructions is high, the ratio of active elements may degrade. We recommend to limitedly use it only when the ratio of store instructions is low. - In the one-byte data type operation, the percentage of the Active element is converted to 100%. In cases of one-byte data type operation, a higher ratio than that of the original Active element is output. You should limitedly use it only when the ratio of one-byte data type operation is low.
Floating-point pipeline B Active element rate(%)	Active element ratio in the floating-point operation pipeline B (%) Note - Except for the SVE instruction, the percentage of the Active element is converted to 100%. In cases of many non-SVE instructions, a higher ratio than that of the original Active element is output. You should limitedly use it only when the ratio of SVE instructions is high. - If the ratio of store instructions is high, the ratio of active elements may degrade. We recommend to limitedly use it only when the ratio of store instructions is low. - In the one-byte data type operation, the percentage of the Active element is converted to 100%. In cases of one-byte data type operation, a higher ratio than that of the original Active element is output. You should limitedly use it only when the ratio of one-byte data type operation is low.

Name	Description
SFI(Store Fetch Interlock) rate	Queuing time ratio by SFI (Store Fetch Interlock)

4.2.2.4.3 Busy (Detail Report)

Figure 4.19 Layout of Busy (Detail Report)

cpu_pa_report.xlsm - Excel

Busy		Floating-point operation pipeline A busy rate (%)	Floating-point operation pipeline B busy rate (%)	Integer operation pipeline A busy rate (%)	Integer operation pipeline B busy rate (%)	L1 busy rate (%)	L2 busy rate (%)	Memory busy rate (%)	Address calculation operation pipeline A busy rate (%)	Address calculation operation pipeline B busy rate (%)	Floating-point pipeline A Active element rate (%)	Floating-point pipeline B Active element rate (%)	L1 pipeline 0 Active element rate (%)	L1 pipeline 1 Active element rate (%)	SFI(Store Fetch Interlock) rate
Process	Thread														
0	0	94.02%	93.74%	6.18%	4.26%	76.78%	61.13%	6.21%	62.45%	60.38%	99.67%	100.00%	100.00%	100.00%	0.00
0	1	94.02%	93.74%	6.01%	4.07%	76.73%			62.18%	60.17%	99.67%	100.00%	100.00%	100.00%	0.00
0	2	94.02%	93.74%	5.93%	4.00%	76.75%			62.29%	60.22%	99.67%	100.00%	100.00%	100.00%	0.00
0	3	94.02%	93.74%	5.91%	3.98%	76.76%			62.34%	60.20%	99.67%	100.00%	100.00%	100.00%	0.00
0	4	94.02%	93.74%	5.93%	4.02%	76.85%			62.30%	60.19%	99.67%	100.00%	100.00%	100.00%	0.00
0	5	94.02%	93.74%	5.86%	3.91%	76.79%			62.44%	60.22%	99.67%	100.00%	100.00%	100.00%	0.00
0	6	94.02%	93.74%	5.87%	3.93%	76.76%			62.41%	60.21%	99.67%	100.00%	100.00%	100.00%	0.00
0	7	94.02%	93.74%	5.83%	3.87%	76.79%			62.49%	60.24%	99.67%	100.00%	100.00%	100.00%	0.00
0	8	94.02%	93.74%	5.86%	3.92%	76.75%			62.42%	60.22%	99.67%	100.00%	100.00%	100.00%	0.00
0	9	94.02%	93.74%	5.83%	3.87%	76.78%			62.49%	60.25%	99.67%	100.00%	100.00%	100.00%	0.00
0	10	94.02%	93.74%	5.87%	3.95%	76.77%			62.40%	60.21%	99.67%	100.00%	100.00%	100.00%	0.00
0	11	94.02%	93.74%	5.80%	3.85%	76.79%			62.51%	60.26%	99.67%	100.00%	100.00%	100.00%	0.00
CMG 0 total		94.02%	93.74%	5.91%	3.97%	76.78%	61.13%	6.21%	62.39%	60.23%	99.67%	100.00%	100.00%	100.00%	0.00

report

Table 4.13 Output Items in Busy (Detail Report)

Name	Description
Floating-point operation pipeline A busy rate(%)	Busy rate for the floating-point operation pipeline A (%)
Floating-point operation pipeline B busy rate(%)	Busy rate for the floating-point operation pipeline B (%)
Integer operation pipeline A busy rate(%)	Busy rate for the integer operation pipeline A (%)
Integer operation pipeline B busy rate(%)	Busy rate for the integer operation pipeline B (%)
L1 busy rate(%)	Busy rate for the primary cache (%)
L2 busy rate(%)	Busy rate for the secondary cache (%)
Memory busy rate(%)	Memory busy rate (%) **Note** - The theoretical value of memory throughput is for one CMG. If you access not only the memory of own CMG but also the memory of other CMGs at the same time, measured value of memory throughput may exceed the theoretical value. In this case, memory busy ratio (same as memory throughput peak ratio) may exceed 100%. - The theoretical value for memory throughput assumes simultaneous memory access from all cores in the CMG. However, in fact, each core may perform memory accesses at different times. In such a case, if the memory access amount of all cores is aggregated within the measurement region, memory accesses at different timings are aggregated as simultaneous accesses. As a result, measured value of memory throughput obtained from the memory access amount and execution time may exceed the theoretical value. The memory busy ratio (same as memory throughput peak ratio) may exceed 100%.

Name	Description
Address calculation operation pipeline A busy rate(%)	Busy rate for the address calculation operation pipeline A (%)
Address calculation operation pipeline B busy rate(%)	Busy rate for the address calculation operation pipeline B (%)
Floating-point pipeline A Active element rate(%)	Active element ratio in the floating-point operation pipeline A (%) Note - Except for the SVE instruction, the percentage of the Active element is converted to 100%. In cases of many non-SVE instructions, a higher ratio than that of the original Active element is output. You should limitedly use it only when the ratio of SVE instructions is high. - If the ratio of store instructions is high, the ratio of active elements may degrade. We recommend to limitedly use it only when the ratio of store instructions is low. - In the one-byte data type operation, the percentage of the Active element is converted to 100%. In cases of one-byte data type operation, a higher ratio than that of the original Active element is output. You should limitedly use it only when the ratio of one-byte data type operation is low.
Floating-point pipeline B Active element rate(%)	Active element ratio in the floating-point operation pipeline B (%) Note - Except for the SVE instruction, the percentage of the Active element is converted to 100%. In cases of many non-SVE instructions, a higher ratio than that of the original Active element is output. You should limitedly use it only when the ratio of SVE instructions is high. - If the ratio of store instructions is high, the ratio of active elements may degrade. We recommend to limitedly use it only when the ratio of store instructions is low. - In the one-byte data type operation, the percentage of the Active element is converted to 100%. In cases of one-byte data type operation, a higher ratio than that of the original Active element is output. You should limitedly use it only when the ratio of one-byte data type operation is low.
L1 pipeline 0 Active element rate(%)	Active element ratio in the L1 pipeline 0 (%)
L1 pipeline 1 Active element rate(%)	Active element ratio in the L1 pipeline 1 (%)
SFI(Store Fetch Interlock) rate	Queuing time ratio by SFI (Store Fetch Interlock)

4.2.2.5 Cache

Cache displays information about cache misses. In the case of the Brief Report, displays the number of cache misses of the primary data and secondary caches and the ratio to the number of load store instructions. In the case of the Standard or Detail Report, comparison with the multiple numbers of instructions is added in addition to the contents of the Brief Report. The background color of cells may change to pink, which means that the item may have resulted in a performance bottleneck. Cache has a corresponding table and graph. It displays a stacked bar graph representing the items shown in blue text in Table "Table 4.14 Output Items in Cache (Brief Report)" or "Table 4.15 Output Items in Cache (Standard and Detail Reports)".

4.2.2.5.1 Cache (Brief Report)

Figure 4.20 Layout of Cache (Brief Report)

cpu_pa_report.xlsm - Excel

Cache		L1I miss rate (/Effective Instruction)	Load-store Instruction	L1D miss	L1D miss rate (/Load-store Instruction)	L1D miss demand rate (%) (/L1D miss)	L1D miss hardware prefetch rate (%) (/L1D miss)	L1D miss software prefetch rate (%) (/L1D miss)	L2 miss	L2 miss rate (/Load-store Instruction)	L2 miss demand rate (%) (/L2 miss)	L2 miss hardware prefetch rate (%) (/L2 miss)	L2 miss software prefetch rate (%) (/L2 miss)	L1D TLB miss rate (/Load-store Instruction)	L2D TLB miss rate (/Load-store Instruction)
Process	Thread														
0	0		1.60E+10	1.80E+09	0.11	0.46%			3.37E+07	0.00	41.70%				
0	1		1.60E+10	1.80E+09	0.11	0.44%			3.36E+07	0.00	41.63%				
0	2		1.60E+10	1.80E+09	0.11	0.44%			3.35E+07	0.00	40.79%				
0	3		1.60E+10	1.80E+09	0.11	0.46%			3.35E+07	0.00	41.53%				
0	4		1.60E+10	1.80E+09	0.11	0.44%			3.36E+07	0.00	41.82%				
0	5		1.61E+10	1.80E+09	0.11	0.44%			3.36E+07	0.00	40.16%				
0	6		1.60E+10	1.80E+09	0.11	0.46%			3.35E+07	0.00	42.08%				
0	7		1.60E+10	1.80E+09	0.11	0.45%			3.35E+07	0.00	40.65%				
0	8		1.60E+10	1.80E+09	0.11	0.46%			3.35E+07	0.00	41.92%				
0	9		1.60E+10	1.80E+09	0.11	0.46%			3.36E+07	0.00	41.83%				
0	10		1.60E+10	1.80E+09	0.11	0.47%			3.35E+07	0.00	43.01%				
0	11		1.60E+10	1.80E+09	0.11	0.42%			3.35E+07	0.00	39.80%				
CMG 0 total			1.93E+11	2.16E+10	0.11	0.45%			4.03E+08	0.00	41.41%				

report

Figure 4.21 Graph of Cache (Brief Report)

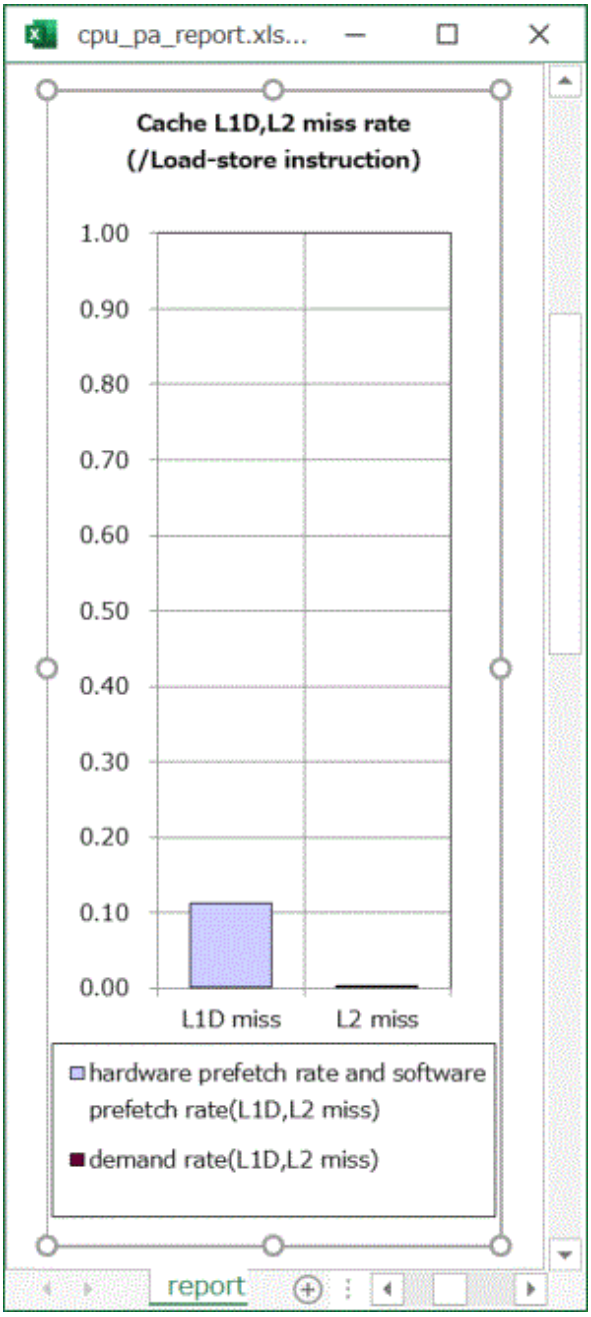


Table 4.14 Output Items in Cache (Brief Report)

Name	Description
Load-store instruction	Number of load store instructions
L1D miss	Number of primary data cache misses
L1D miss rate (/Load-store instruction)	Ratio of primary data cache misses to the number of load store instructions
L1D miss demand rate(%) (/L1D miss)	Ratio of primary data cache misses due to demand access among primary data cache misses (%)
L2 miss	Number of secondary cache misses
L2 miss rate (/Load-store instruction)	Ratio of secondary cache misses to the number of load store instructions
L2 miss demand rate(%) (/L2 miss)	Ratio of secondary cache misses due to demand access among secondary cache misses (%)

Point

The total value in the stacked bar graph matches to "L1D miss rate(%)(/Load-store instruction)" or "L2 miss rate(%)(/Load-store instruction)".

4.2.2.5.2 Cache (Standard and Detail Reports)

Figure 4.22 Layout of Cache (Standard and Detail Reports)

cpu_pa_report.xlsm - Excel

Cache		L1I miss rate (/Effective instruction)	Load-store instruction	L1D miss	L1D miss rate (/Load-store instruction)	L1D miss demand rate (%) (/L1D miss)	L1D miss hardware prefetch rate (%) (/L1D miss)	L1D miss software prefetch rate (%) (/L1D miss)	L2 miss	L2 miss rate (/Load-store instruction)	L2 miss demand rate (%) (/L2 miss)	L2 miss hardware prefetch rate (%) (/L2 miss)	L2 miss software prefetch rate (%) (/L2 miss)	L1D TLB miss rate (/Load-store instruction)	L2D TLB miss rate (/Load-store instruction)
Process	Thread														
0	0	0.00	1.60E+10	1.80E+09	0.11	0.46%	0.68%	98.86%	3.37E+07	0.00	42.09%	64.42%	0.00%	0.00001	0.00000
0	1	0.00	1.60E+10	1.80E+09	0.11	0.44%	0.71%	98.85%	3.36E+07	0.00	41.64%	65.08%	0.00%	0.00001	0.00000
0	2	0.00	1.60E+10	1.80E+09	0.11	0.44%	0.72%	98.83%	3.35E+07	0.00	40.52%	65.27%	0.00%	0.00001	0.00000
0	3	0.00	1.60E+10	1.80E+09	0.11	0.46%	0.69%	98.86%	3.35E+07	0.00	41.67%	64.62%	0.00%	0.00001	0.00000
0	4	0.00	1.60E+10	1.80E+09	0.11	0.44%	0.71%	98.84%	3.36E+07	0.00	41.70%	64.73%	0.00%	0.00001	0.00000
0	5	0.00	1.61E+10	1.80E+09	0.11	0.44%	0.72%	98.85%	3.36E+07	0.00	40.29%	65.63%	0.00%	0.00001	0.00000
0	6	0.00	1.60E+10	1.80E+09	0.11	0.46%	0.68%	98.86%	3.35E+07	0.00	42.47%	64.34%	0.00%	0.00001	0.00000
0	7	0.00	1.60E+10	1.80E+09	0.11	0.45%	0.70%	98.86%	3.35E+07	0.00	40.91%	65.20%	0.00%	0.00001	0.00000
0	8	0.00	1.60E+10	1.80E+09	0.11	0.46%	0.70%	98.84%	3.35E+07	0.00	41.91%	64.56%	0.00%	0.00001	0.00000
0	9	0.00	1.60E+10	1.80E+09	0.11	0.46%	0.70%	98.83%	3.36E+07	0.00	41.35%	64.29%	0.00%	0.00001	0.00000
0	10	0.00	1.60E+10	1.80E+09	0.11	0.47%	0.72%	98.80%	3.35E+07	0.00	41.84%	63.95%	0.00%	0.00001	0.00000
0	11	0.00	1.60E+10	1.80E+09	0.11	0.42%	0.69%	98.89%	3.35E+07	0.00	40.53%	65.69%	0.00%	0.00001	0.00000
CMG 0 total		0.00	1.93E+11	2.16E+10	0.11	0.45%	0.70%	98.85%	4.03E+08	0.00	41.41%	64.81%	0.00%	0.00001	0.00000

report

Figure 4.23 Graph of Cache (Standard and Detail Reports)

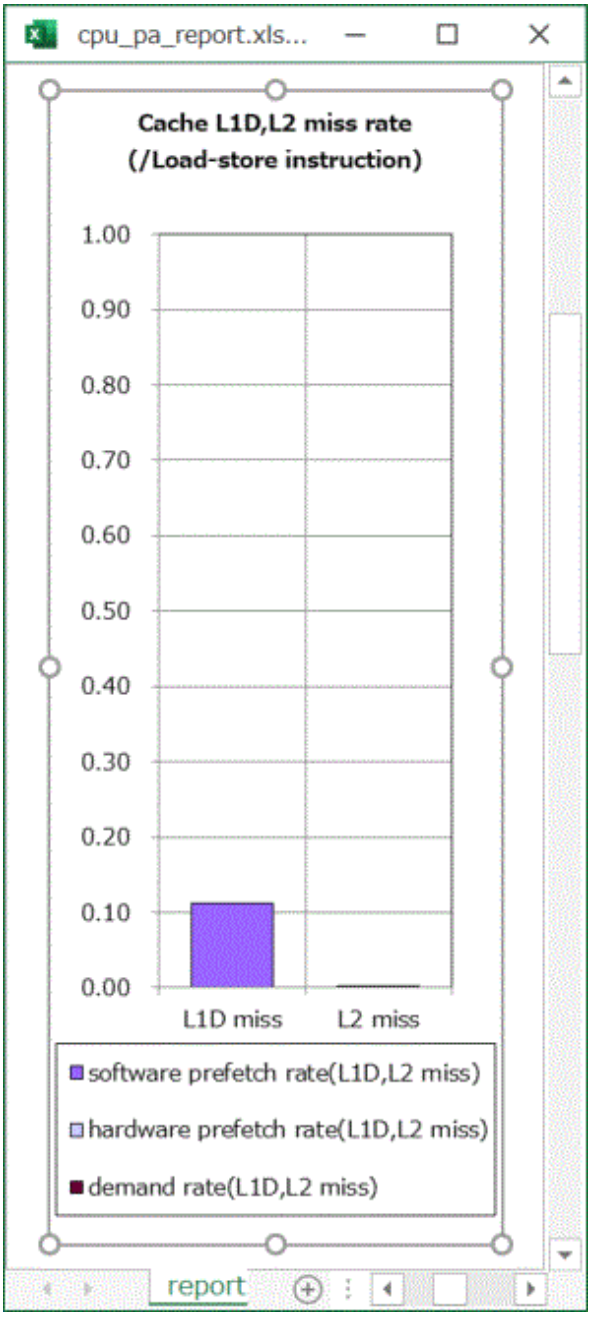


Table 4.15 Output Items in Cache (Standard and Detail Reports)

Name	Description
L1I miss rate (/Effective instruction)	Ratio of primary instruction cache misses to the total number of executed instructions
Load-store instruction	Number of load store instructions
L1D miss	Number of primary data cache misses
L1D miss rate (/Load-store instruction)	Ratio of primary data cache misses to the number of load store instructions
L1D miss demand rate(%) (/L1D miss)	Ratio of primary data cache misses due to demand access among primary data cache misses (%)
L1D miss hardware prefetch rate(%) (/L1D miss)	Ratio of primary data cache misses due to hardware prefetch among primary data cache misses (%)

Name	Description
L1D miss software prefetch rate(%) (/L1D miss)	Ratio of primary data cache misses due to software prefetch among primary data cache misses (%)
L2 miss	Number of secondary cache misses
L2 miss rate (/Load-store instruction)	Ratio of secondary cache messes to the number of load store instructions
L2 miss demand rate(%) (/L2 miss)	Ratio of secondary cache misses due to demand access among secondary cache misses (%)
L2 miss hardware prefetch rate(%) (/L2 miss)	Ratio of secondary cache misses due to hardware prefetch among secondary cache misses (%)
L2 miss software prefetch rate(%) (/L2 miss)	Ratio of secondary cache misses due to software prefetch among secondary cache misses (%)
L1D TLB miss rate (/Load-store instruction)	Ratio of L1D TLB misses to the number of load store instructions
L2D TLB miss rate (/Load-store instruction)	Ratio of L2D TLB misses to the number of load store instructions

Point

The total value in the stacked bar graph matches to "L1D miss rate(%)(/Load-store instruction)" or "L2 miss rate(%)(/Load-store instruction)".

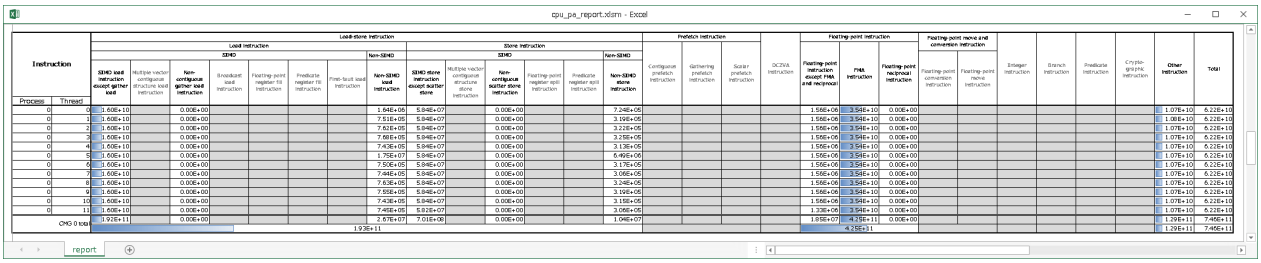
The ratio of cache misses may be out of range due to exceed the effect of measurement errors or measurement variation. The ratio of cache misses greater than 100% is considered 100%, and the ratio of cache misses less than 0% is considered 0%.

4.2.2.6 Instruction

Instruction displays information about instruction mixes. The Brief Report displays the information by classifying it into 10 types; the Standard Report, 25 types; and the Detail Report, 28 types.

4.2.2.6.1 Instruction (Brief Report)

Figure 4.24 Layout of Instruction (Brief Report)



Instruction		Load-store instruction														Prefetch instruction				Floating-point instruction			Floating-point move and conversion instruction		Integer instruction	Branch instruction	Predicate instruction	Crypto-graphic instruction	Other instruction	Total
		Load instruction							Store instruction																					
		SIMD						Non-SIMD	SIMD					Non-SIMD																
Process	Thread	SIMD load instruction except gather load	Multiple vector contiguous structure load instruction	Non-contiguous gather load instruction	Broadcast load instruction	Floating-point register fill instruction	Predicate register fill instruction	Free-fault load instruction	Non-SIMD load instruction	SIMD store instruction except scatter store	Multiple vector contiguous structure store instruction	Non-contiguous scatter store instruction	Floating-point register spill instruction	Predicate register spill instruction	Non-SIMD store instruction	Contiguous prefetch instruction	Gathering prefetch instruction	Scatter prefetch instruction	DCZVA instruction	Floating-point instruction except FMA and reciprocal	FMA instruction	Floating-point reciprocal instruction	Floating-point conversion instruction	Floating-point move instruction						
0	0	1.60E+10		0.00E+00					1.64E+06	5.84E+07		0.00E+00			7.24E+05					1.56E+06	3.54E+10	0.00E+00							1.07E+10	6.22E+10
0	1	1.60E+10		0.00E+00					7.51E+05	5.84E+07		0.00E+00			3.19E+05					1.56E+06	3.54E+10	0.00E+00							1.08E+10	6.22E+10
0	2	1.60E+10		0.00E+00					7.62E+05	5.84E+07		0.00E+00			3.22E+05					1.56E+06	3.54E+10	0.00E+00							1.07E+10	6.22E+10
0	3	1.60E+10		0.00E+00					7.56E+05	5.84E+07		0.00E+00			3.25E+05					1.56E+06	3.54E+10	0.00E+00							1.07E+10	6.22E+10
0	4	1.60E+10		0.00E+00					7.45E+05	5.84E+07		0.00E+00			3.13E+05					1.56E+06	3.54E+10	0.00E+00							1.07E+10	6.22E+10
0	5	1.60E+10		0.00E+00					1.75E+07	5.84E+07		0.00E+00			6.49E+06					1.56E+06	3.54E+10	0.00E+00							1.07E+10	6.22E+10
0	6	1.60E+10		0.00E+00					7.50E+05	5.84E+07		0.00E+00			3.17E+05					1.56E+06	3.54E+10	0.00E+00							1.07E+10	6.22E+10
0	7	1.60E+10		0.00E+00					7.44E+05	5.84E+07		0.00E+00			3.06E+05					1.56E+06	3.54E+10	0.00E+00							1.07E+10	6.22E+10
0	8	1.60E+10		0.00E+00					7.63E+05	5.84E+07		0.00E+00			3.24E+05					1.56E+06	3.54E+10	0.00E+00							1.07E+10	6.22E+10
0	9	1.60E+10		0.00E+00					7.55E+05	5.84E+07		0.00E+00			3.15E+05					1.56E+06	3.54E+10	0.00E+00							1.07E+10	6.22E+10
0	10	1.60E+10		0.00E+00					7.43E+05	5.84E+07		0.00E+00			3.15E+05					1.56E+06	3.54E+10	0.00E+00							1.07E+10	6.22E+10
0	11	1.60E+10		0.00E+00					7.45E+05	5.82E+07		0.00E+00			3.06E+05					1.33E+06	3.54E+10	0.00E+00							1.07E+10	6.22E+10
CMG total		1.92E+11		0.00E+00					2.67E+07	7.01E+08		0.00E+00			1.04E+07					1.85E+07	4.25E+11	0.00E+00							1.29E+11	7.46E+11
		1.92E+11																			4.25E+11									

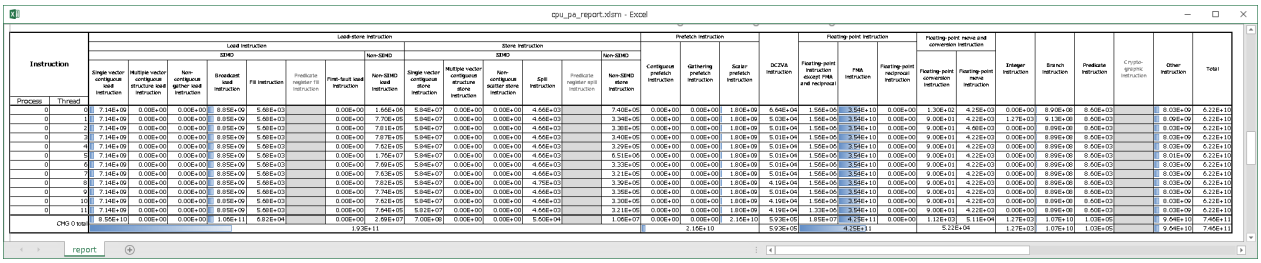
Table 4.16 Output Items in Instruction (Brief Report)

Name	Description
SIMD load instruction except gather load	Number of SIMD load instructions excluding gather load instructions
Non-contiguous gather load instruction	Number of non-contiguous gather load instructions
Non-SIMD load instruction	Number of non-SIMD load instructions
SIMD store instruction except scatter store	Number of SIMD store instructions excluding scatter store instructions
Non-contiguous scatter store instruction	Number of non-contiguous scatter store instructions
Non-SIMD store instruction	Number of non-SIMD store instructions
Floating-point instruction except FMA and reciprocal	Number of floating-point operation instructions excluding floating-point multiply and add operation instructions and floating-point precision conversion instructions

Name	Description
FMA instruction	Number of floating-point multiply and add operation instructions
Floating-point reciprocal instruction	Number of floating-point reciprocal operation instructions
Other instruction	Number of other instructions
Total	Total number of instructions **Note** Total number of instructions does not include MOVPRFX instructions.

4.2.2.6.2 Instruction (Standard Report)

Figure 4.25 Layout of Instruction (Standard Report)



Instruction		Load-store instruction																Prefetch instruction			DCZVA instruction	Floating-point instruction			Floating-point move and conversion instruction		Integer instruction	Branch instruction	Predicate instruction	Crypto-graphic instruction	Other instruction	Total
		Load instruction								Store instruction																							
		SIMD							Non-SIMD	SIMD							Non-SIMD																
Process	Thread	Single vector contiguous load instruction	Multiple vector contiguous structure load instruction	Non-contiguous gather load instruction	Broadcast load instruction	Fill instruction	Predicate register fill instruction	First-fault load instruction	Non-SIMD load instruction	Single vector contiguous store instruction	Multiple vector contiguous structure store instruction	Non-contiguous scatter store instruction	Spill instruction	Predicate register spill instruction	Non-SIMD store instruction	Contiguous prefetch instruction	Gathering prefetch instruction	Scalar prefetch instruction	DCZVA instruction	Floating-point instruction except FMA and reciprocal	FMA instruction	Floating-point reciprocal instruction	Floating-point conversion instruction	Floating-point move instruction	Integer instruction	Branch instruction	Predicate instruction	Crypto-graphic instruction	Other instruction	Total
0	0	7.14E+09	0.00E+00	0.00E+00	8.85E+09	5.68E+03		0.00E+00	1.66E+06	5.84E+07	0.00E+00	0.00E+00	4.66E+03		7.40E+05	0.00E+00	0.00E+00	1.80E+09	6.64E+04	1.56E+06	3.54E+10	0.00E+00	1.30E+02	4.25E+03	0.00E+00	8.90E+08	8.60E+03		8.03E+09	6.22E+10
0	1	7.14E+09	0.00E+00	0.00E+00	8.85E+09	5.68E+03		0.00E+00	7.70E+05	5.84E+07	0.00E+00	0.00E+00	4.66E+03		3.34E+05	0.00E+00	0.00E+00	1.80E+09	5.01E+04	1.56E+06	3.54E+10	0.00E+00	9.00E+01	4.22E+03	1.27E+03	9.13E+08	8.60E+03		8.03E+09	6.22E+10
0	2	7.14E+09	0.00E+00	0.00E+00	8.85E+09	5.68E+03		0.00E+00	7.81E+05	5.84E+07	0.00E+00	0.00E+00	4.66E+03		3.36E+05	0.00E+00	0.00E+00	1.80E+09	5.01E+04	1.56E+06	3.54E+10	0.00E+00	9.00E+01	4.00E+03	0.00E+00	8.89E+08	8.60E+03		8.03E+09	6.22E+10
0	3	7.14E+09	0.00E+00	0.00E+00	8.85E+09	5.68E+03		0.00E+00	7.87E+05	5.84E+07	0.00E+00	0.00E+00	4.66E+03		3.40E+05	0.00E+00	0.00E+00	1.80E+09	5.01E+04	1.56E+06	3.54E+10	0.00E+00	9.00E+01	4.22E+03	0.00E+00	8.89E+08	8.60E+03		8.03E+09	6.22E+10
0	4	7.14E+09	0.00E+00	0.00E+00	8.85E+09	5.68E+03		0.00E+00	7.61E+05	5.84E+07	0.00E+00	0.00E+00	4.66E+03		3.29E+05	0.00E+00	0.00E+00	1.80E+09	5.01E+04	1.56E+06	3.54E+10	0.00E+00	9.00E+01	4.22E+03	0.00E+00	8.89E+08	8.60E+03		8.03E+09	6.22E+10
0	5	7.14E+09	0.00E+00	0.00E+00	8.85E+09	5.68E+03		0.00E+00	1.76E+07	5.84E+07	0.00E+00	0.00E+00	4.66E+03		6.51E+06	0.00E+00	0.00E+00	1.80E+09	5.01E+04	1.56E+06	3.54E+10	0.00E+00	9.00E+01	4.22E+03	0.00E+00	8.89E+08	8.60E+03		8.01E+09	6.22E+10
0	6	7.14E+09	0.00E+00	0.00E+00	8.85E+09	5.68E+03		0.00E+00	7.69E+05	5.84E+07	0.00E+00	0.00E+00	4.66E+03		3.33E+05	0.00E+00	0.00E+00	1.80E+09	5.01E+04	1.56E+06	3.54E+10	0.00E+00	9.00E+01	4.22E+03	0.00E+00	8.89E+08	8.60E+03		8.03E+09	6.22E+10
0	7	7.14E+09	0.00E+00	0.00E+00	8.85E+09	5.68E+03		0.00E+00	7.63E+05	5.84E+07	0.00E+00	0.00E+00	4.66E+03		3.21E+05	0.00E+00	0.00E+00	1.80E+09	5.01E+04	1.56E+06	3.54E+10	0.00E+00	9.00E+01	4.22E+03	0.00E+00	8.89E+08	8.60E+03		8.03E+09	6.22E+10
0	8	7.14E+09	0.00E+00	0.00E+00	8.85E+09	5.68E+03		0.00E+00	7.82E+05	5.84E+07	0.00E+00	0.00E+00	4.75E+03		3.30E+05	0.00E+00	0.00E+00	1.80E+09	4.19E+04	1.56E+06	3.54E+10	0.00E+00	9.00E+01	4.22E+03	0.00E+00	8.89E+08	8.60E+03		8.03E+09	6.22E+10
0	9	7.14E+09	0.00E+00	0.00E+00	8.85E+09	5.68E+03		0.00E+00	7.74E+05	5.84E+07	0.00E+00	0.00E+00	4.66E+03		3.35E+05	0.00E+00	0.00E+00	1.80E+09	5.01E+04	1.56E+06	3.54E+10	0.00E+00	9.00E+01	4.22E+03	0.00E+00	8.89E+08	8.60E+03		8.03E+09	6.22E+10
0	10	7.14E+09	0.00E+00	0.00E+00	8.85E+09	5.68E+03		0.00E+00	7.61E+05	5.84E+07	0.00E+00	0.00E+00	4.66E+03		3.30E+05	0.00E+00	0.00E+00	1.80E+09	4.19E+04	1.56E+06	3.54E+10	0.00E+00	9.00E+01	4.22E+03	0.00E+00	8.89E+08	8.60E+03		8.03E+09	6.22E+10
0	11	7.14E+09	0.00E+00	0.00E+00	8.85E+09	5.68E+03		0.00E+00	7.64E+05	5.82E+07	0.00E+00	0.00E+00	4.66E+03		3.21E+05	0.00E+00	0.00E+00	1.80E+09	4.19E+04	1.33E+06	3.54E+10	0.00E+00	9.00E+01	4.22E+03	0.00E+00	8.89E+08	8.60E+03		8.03E+09	6.22E+10
CMG 0 total		8.55E+10	0.00E+00	0.00E+00	1.06E+11	6.82E+04		0.00E+00	2.69E+07	7.00E+08	0.00E+00	0.00E+00	5.60E+04		1.06E+07	0.00E+00	0.00E+00	2.16E+10	5.93E+05	1.85E+07	4.25E+11	0.00E+00	1.12E+03	5.11E+04	1.27E+03	1.07E+10	1.03E+05		9.54E+10	7.46E+11
					1.93E+11													2.16E+10	5.93E+05		4.25E+11		5.22E+04		1.27E+03	1.07E+10	1.03E+05			

Table 4.17 Output Items in Instruction (Standard Report)

Name	Description
Single vector contiguous load instruction	Number of SIMD contiguous load instructions
Multiple vector contiguous structure load instruction	Number of SIMD contiguous structure load instructions
Non-contiguous gather load instruction	Number of non-contiguous gather load instructions
Broadcast load instruction	Number of broadcast load instructions
Fill instruction	Number of fill instructions
First-fault load instruction	Number of First-fault load instructions
Non-SIMD load instruction	Number of non-SIMD load instructions
Single vector contiguous store instruction	Number of SIMD contiguous store instructions
Multiple vector contiguous structure store instruction	Number of SIMD contiguous structure store instructions
Non-contiguous scatter store instruction	Number of non-contiguous scatter store instructions
Spill instruction	Number of spill instructions
Non-SIMD store instruction	Number of non-SIMD store instructions
Contiguous prefetch instruction	Number of contiguous prefetch instructions
Gathering prefetch instruction	Number of gather prefetch instructions
Scalar prefetch instruction	Number of normal prefetch instructions
DCZVA instruction	Number of DCZVA instructions
Floating-point instruction except FMA and reciprocal	Number of floating-point operation instructions excluding floating-point multiply and add operation instructions and floating-point precision conversion instructions
FMA instruction	Number of floating-point multiply and add operation instructions
Floating-point reciprocal instruction	Number of floating-point reciprocal operation instructions

Name	Description
Floating-point conversion instruction	Number of floating-point precision conversion instructions
Floating-point move instruction	Number of floating-point move instructions
Integer instruction	Number of integer operation instructions
Branch instruction	Number of branch instructions
Predicate instruction	Number of predicate operation instructions
Other instruction	Number of other operations
Total	Total number of instructions **Note** Total number of instructions does not include MOVPRFX instructions.

4.2.2.6.3 Instruction (Detail Report)

Figure 4.26 Layout of Instruction (Detail Report)

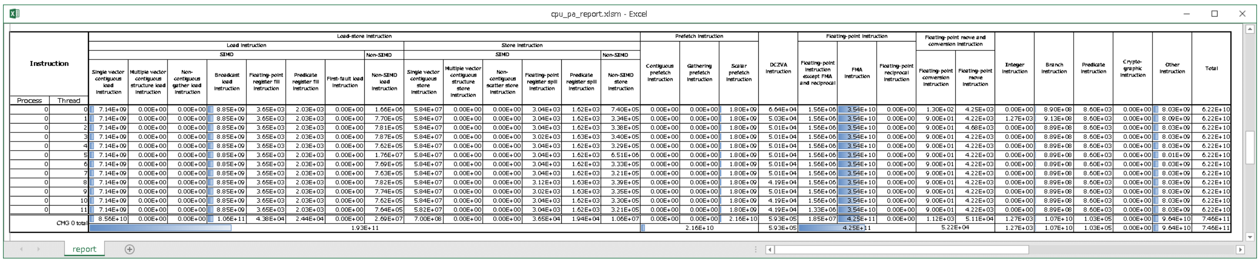


Table 4.18 Output Items in Instruction (Detail Report)

Name	Description
Single vector contiguous load instruction	Number of SIMD contiguous load instructions
Multiple vector contiguous structure load instruction	Number of SIMD contiguous structure load instructions
Non-contiguous gather load instruction	Number of non-contiguous gather load instructions
Broadcast load instruction	Number of broadcast load instructions
Floating-point register fill instruction	Number of fill instructions into floating point register
Predicate register fill instruction	Number of fill instructions into predicate register
First-fault load instruction	Number of First-fault load instructions
Non-SIMD load instruction	Number of non-SIMD load instructions
Single vector contiguous store instruction	Number of SIMD contiguous store instructions
Multiple vector contiguous structure store instruction	Number of SIMD contiguous structure store instructions
Non-contiguous scatter store instruction	Number of non-contiguous scatter store instructions
Floating-point register spill instruction	Number of spill instructions from floating-point register
Predicate register spill instruction	Number of spill instructions from predicate register
Non-SIMD store instruction	Number of non-SIMD store instructions
Contiguous prefetch instruction	Number of contiguous prefetch instructions
Gathering prefetch instruction	Number of gather prefetch instructions

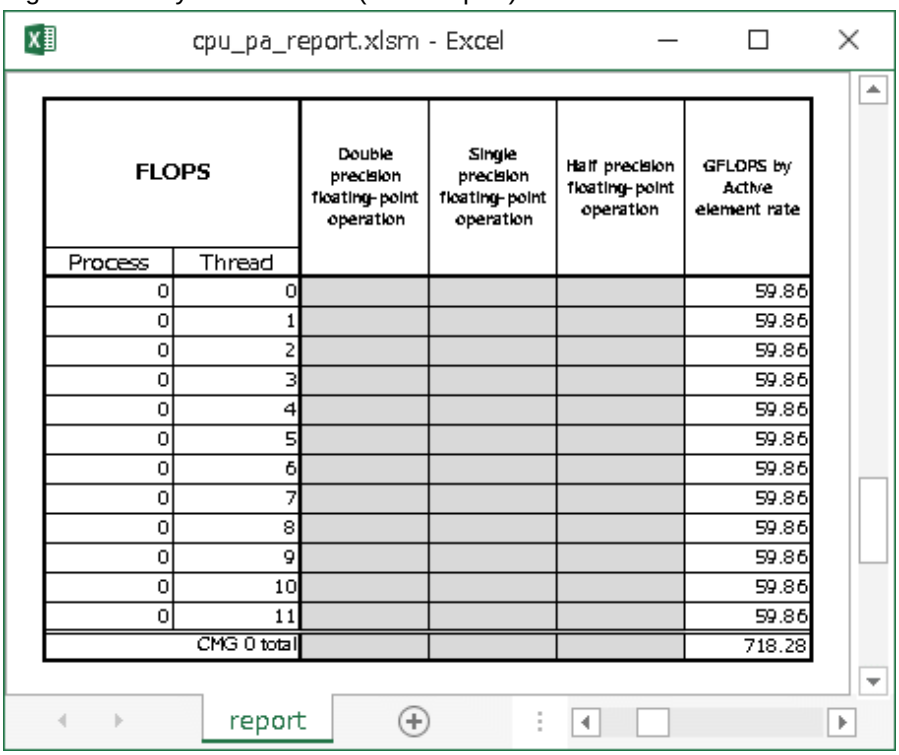
Name	Description
Scalar prefetch instruction	Number of normal prefetch instructions
DCZVA instruction	Number of DCZVA instructions
Floating-point instruction except FMA and reciprocal	Number of floating-point operation instructions excluding floating-point multiply and add operation instructions and floating-point precision conversion instructions
FMA instruction	Number of floating-point multiply and add operation instructions
Floating-point reciprocal instruction	Number of floating-point reciprocal operation instructions
Floating-point conversion instruction	Number of floating-point precision conversion instructions
Floating-point move instruction	Number of floating-point move instructions
Integer instruction	Number of integer operation instructions
Branch instruction	Number of branch instructions
Predicate instruction	Number of predicate operation instructions
Crypto-graphic instruction	Number of cryptographic instructions
Other instruction	Number of other operations
Total	Total number of instructions Note Total number of instructions does not include MOVPRFX instructions.

4.2.2.7 FLOPS

FLOPS displays information about floating-point operations. In the case of the Brief Report, it displays the ratio of active elements and floating-point operation performance considering the ratio of active elements. In the case of the Standard or Detail Report, it displays the ratio of the number of floating-point operations at each precision level, in addition to the contents of the Brief Report.

4.2.2.7.1 FLOPS (Brief Report)

Figure 4.27 Layout of FLOPS (Brief Report)



cpu_pa_report.xlsm - Excel

FLOPS		Double precision floating-point operation	Single precision floating-point operation	Half precision floating-point operation	GFLOPS by Active element rate
Process	Thread				
0	0				59.86
0	1				59.86
0	2				59.86
0	3				59.86
0	4				59.86
0	5				59.86
0	6				59.86
0	7				59.86
0	8				59.86
0	9				59.86
0	10				59.86
0	11				59.86
CMG 0 total					718.28

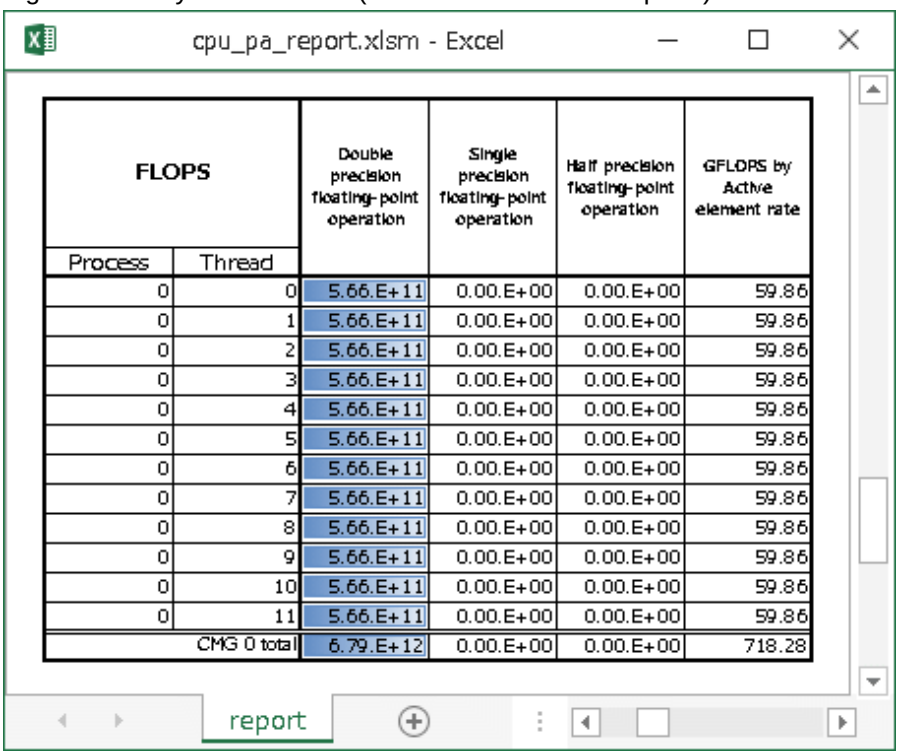
report

Table 4.19 Output Items in FLOPS (Brief Report)

Name	Description
GFLOPS by Active element rate	floating-point operation performance considering the ratio of active elements **Note** - Normally, all GFLOPS values are calculated by assuming that all elements are active. Therefore, in the case of a program with many inactive elements, a higher value than the original GFLOPS value is output. For a program that uses many Predicate operations, use of the number of floating-point operations executed per second that takes into account the ratio of active elements makes it possible to calculate more precise values. However, if the ratio of store instructions is high, the precision of the number of floating-point operations executed per second that takes into account the ratio of active elements may degrade. We recommend to limitedly use it only when the ratio of store instructions is low. - Except for the SVE instruction, the percentage of the Active element is converted to 100%. In cases of many non-SVE instructions, a higher ratio than that of the original Active element is output. You should limitedly use it only when the ratio of SVE instructions is high. - In the one-byte data type operation, the percentage of the Active element is converted to 100%. In cases of one-byte data type operation, a higher ratio than that of the original Active element is output. You should limitedly use it only when the ratio of one-byte data type operation is low.

4.2.2.7.2 FLOPS (Standard and Detail Reports)

Figure 4.28 Layout of FLOPS (Standard and Detail Reports)



cpu_pa_report.xlsm - Excel

FLOPS		Double precision floating-point operation	Single precision floating-point operation	Half precision floating-point operation	GFLOPS by Active element rate
Process	Thread				
0	0	5.66.E+11	0.00.E+00	0.00.E+00	59.86
0	1	5.66.E+11	0.00.E+00	0.00.E+00	59.86
0	2	5.66.E+11	0.00.E+00	0.00.E+00	59.86
0	3	5.66.E+11	0.00.E+00	0.00.E+00	59.86
0	4	5.66.E+11	0.00.E+00	0.00.E+00	59.86
0	5	5.66.E+11	0.00.E+00	0.00.E+00	59.86
0	6	5.66.E+11	0.00.E+00	0.00.E+00	59.86
0	7	5.66.E+11	0.00.E+00	0.00.E+00	59.86
0	8	5.66.E+11	0.00.E+00	0.00.E+00	59.86
0	9	5.66.E+11	0.00.E+00	0.00.E+00	59.86
0	10	5.66.E+11	0.00.E+00	0.00.E+00	59.86
0	11	5.66.E+11	0.00.E+00	0.00.E+00	59.86
	CMG 0 total	6.79.E+12	0.00.E+00	0.00.E+00	718.28

report

Table 4.20 Output Items in FLOPS (Standard and Detail Reports)

Name	Description
Double precision floating-point operation	Double precision floating-point operation count
Single precision floating-point operation	Single precision floating-point operation count
Half precision floating-point operation	Half precision floating-point operation count
GFLOPS by Active element rate	floating-point operation performance considering the ratio of active elements **Note** - Normally, all GFLOPS values are calculated by assuming that all elements are active. Therefore, in the case of a program with many inactive elements, a higher value than the original GFLOPS value is output. For a program that uses many Predicate operations, use of the number of floating-point operations executed per second that takes into account the ratio of active elements makes it possible to calculate more precise values. However, if the ratio of store instructions is high, the precision of the number of floating-point operations executed per second that takes into account the ratio of active elements may degrade. We recommend to limitedly use it only when the ratio of store instructions is low. - Except for the SVE instruction, the percentage of the Active element is converted to 100%. In cases of many non-SVE instructions, a higher ratio than that of the original Active element is output. You should limitedly use it only when the ratio of SVE instructions is high.

Name	Description
	- In the one-byte data type operation, the percentage of the Active element is converted to 100%. In cases of one-byte data type operation, a higher ratio than that of the original Active element is output. You should limitedly use it only when the ratio of one-byte data type operation is low.

4.2.2.8 Extra

Extra displays the details of gather instructions and information on instructions that are not included in instruction mixes.

Figure 4.29 Layout of Extra

cpu_pa_report.xlsm - Excel

Extra		Gather instruction rate (%)			Instruction						
Process	Thread	0 flow rate (%)	1 flow rate (%)	2 flows rate (%)	Micro-operation instruction	Element manipulated instruction	Register manipulated instruction	MOVPRFX instruction	Math functional instruction	Micro decomposition instruction rate (%)	Branch prediction miss rate (%)
0	0	0.00%	0.00%	0.00%	6.22E+10	0.00E+00	1.52E+02	0.00E+00	0.00E+00	100.00%	0.01%
0	1	0.00%	0.00%	0.00%	6.22E+10	0.00E+00	1.08E+02	0.00E+00	0.00E+00	100.00%	0.00%
0	2	0.00%	0.00%	0.00%	6.23E+10	0.00E+00	1.00E+02	0.00E+00	0.00E+00	100.00%	0.19%
0	3	0.00%	0.00%	0.00%	6.22E+10	0.00E+00	9.60E+01	0.00E+00	0.00E+00	100.00%	0.00%
0	4	0.00%	0.00%	0.00%	6.22E+10	0.00E+00	1.04E+02	0.00E+00	0.00E+00	100.00%	0.00%
0	5	0.00%	0.00%	0.00%	6.22E+10	0.00E+00	9.80E+01	0.00E+00	0.00E+00	100.00%	0.00%
0	6	0.00%	0.00%	0.00%	6.22E+10	0.00E+00	9.60E+01	0.00E+00	0.00E+00	100.00%	0.00%
0	7	0.00%	0.00%	0.00%	6.22E+10	0.00E+00	9.60E+01	0.00E+00	0.00E+00	100.00%	0.00%
0	8	0.00%	0.00%	0.00%	6.22E+10	0.00E+00	9.60E+01	0.00E+00	0.00E+00	100.00%	0.00%
0	9	0.00%	0.00%	0.00%	6.22E+10	0.00E+00	1.00E+02	0.00E+00	0.00E+00	100.00%	0.00%
0	10	0.00%	0.00%	0.00%	6.22E+10	5.58E+02	5.51E+02	0.00E+00	0.00E+00	100.00%	0.00%
0	11	0.00%	0.00%	0.00%	6.22E+10	0.00E+00	9.80E+01	0.00E+00	0.00E+00	100.00%	0.00%
CMG 0 total		0.00%	0.00%	0.00%	7.46E+11	5.58E+02	1.70E+03	0.00E+00	0.00E+00	100.00%	0.02%
		0.00%			7.46E+11	2.25E+03		0.00E+00	0.00E+00	100.00%	0.02%

report

Table 4.21 Output Items in Extra

Name	Description
0 flow rate(%)	Ratio of instructions that resulted in 0-flow execution because they were Inactive elements, among gather instructions (%)
1 flow rate(%)	Ratio of instructions that resulted in 1-flow execution among gather instructions (%)
2 flows rate(%)	Ratio of instructions that resulted in 2-flow execution among gather instructions (%)
Micro-operation instruction	Number of micro instructions
Element manipulated instruction	Number of inter-element operation instructions
Register manipulated instruction	Number of inter-register operation instructions
MOVPRFX instruction	Number of MOVPRFX instructions
Math functional instruction	Number of mathematical function auxiliary operation instructions
Micro decomposition instruction rate(%)	Micro instruction decomposition rate (%)
Branch prediction miss rate(%)	Branch misprediction rate (%)

4.2.2.9 Hardware Prefetch Rate (%) (/Hardware Prefetch)

Hardware Prefetch Rate (%) (/Hardware Prefetch) displays a breakdown of hardware prefetch. For hardware prefetch, there are modes such as "Stream detect mode" and "Prefetch injection mode". This table shows how much each mode was operating.

Figure 4.30 Layout of Hardware Prefetch Rate(%) (/Hardware Prefetch)

cpu_pa_report.xlsm - Excel

Hardware Prefetch Rate (%) (/Hardware Prefetch)		L1			L2			L1/L2
Process	Thread	Stream mode prefetch rate	Injection mode allocate prefetch rate	Injection mode unallocate prefetch rate	Stream mode prefetch rate	Injection mode allocate prefetch rate	Injection mode unallocate prefetch rate	Other hardware prefetch
0	0	0.09%	0.00%	0.00%	0.10%	0.00%	0.00%	99.81%
0	1	0.10%	0.00%	0.00%	0.11%	0.00%	0.00%	99.79%
0	2	0.13%	0.00%	0.00%	0.10%	0.00%	0.00%	99.77%
0	3	0.14%	0.00%	0.00%	0.11%	0.00%	0.00%	99.76%
0	4	0.10%	0.00%	0.00%	0.06%	0.00%	0.00%	99.84%
0	5	0.19%	0.00%	0.00%	0.12%	0.00%	0.00%	99.69%
0	6	0.19%	0.00%	0.00%	0.13%	0.00%	0.00%	99.68%
0	7	0.18%	0.00%	0.00%	0.12%	0.00%	0.00%	99.70%
0	8	0.31%	0.00%	0.00%	0.22%	0.00%	0.00%	99.47%
0	9	0.11%	0.00%	0.00%	0.13%	0.00%	0.00%	99.76%
0	10	0.13%	0.00%	0.00%	0.13%	0.00%	0.00%	99.74%
0	11	0.18%	0.00%	0.00%	0.14%	0.00%	0.00%	99.68%
CMG 0 total		0.15%	0.00%	0.00%	0.12%	0.00%	0.00%	99.72%

report

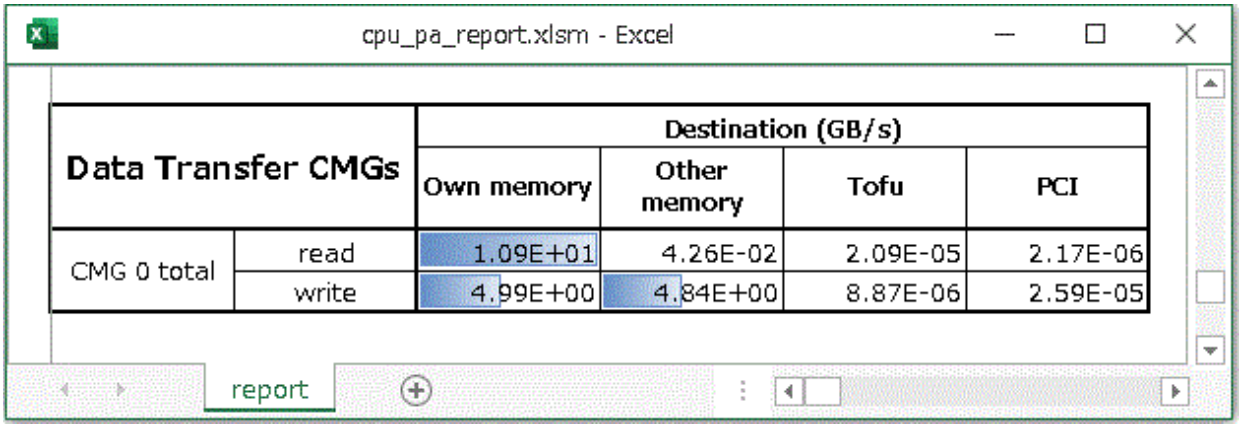
Table 4.22 Output Items in Hardware Prefetch Rate(%) (/Hardware Prefetch)

	Name	Description
L1 cache	Stream mode prefetch rate	Ratio of prefetches in stream mode among hardware prefetches
	Injection mode allocate prefetch rate	Ratio of prefetches whose prefetch injection mode is ALLOCATE mode among hardware prefetches
	Injection mode unallocate prefetch rate	Ratio of prefetches whose prefetch injection mode is UNALLOCATE mode among hardware prefetches
L2 cache	Stream mode prefetch rate	Ratio of prefetches in stream mode among hardware prefetches
	Injection mode allocate prefetch rate	Ratio of prefetches whose prefetch injection mode is ALLOCATE mode among hardware prefetches
	Injection mode unallocate prefetch rate	Ratio of prefetches whose prefetch injection mode is UNALLOCATE mode among hardware prefetches
L1/L2 cache	Other hardware prefetch	Ratio of other prefetches among hardware prefetches

4.2.2.10 Data Transfer CMGs

Data Transfer CMGs displays information about throughput between the user-specified CMG and Own memory, Other memory, Tofu, and PCI.

Figure 4.31 Layout of Data Transfer CMGs



Data Transfer CMGs		Destination (GB/s)			
		Own memory	Other memory	Tofu	PCI
CMG 0 total	read	1.09E+01	4.26E-02	2.09E-05	2.17E-06
	write	4.99E+00	4.84E+00	8.87E-06	2.59E-05

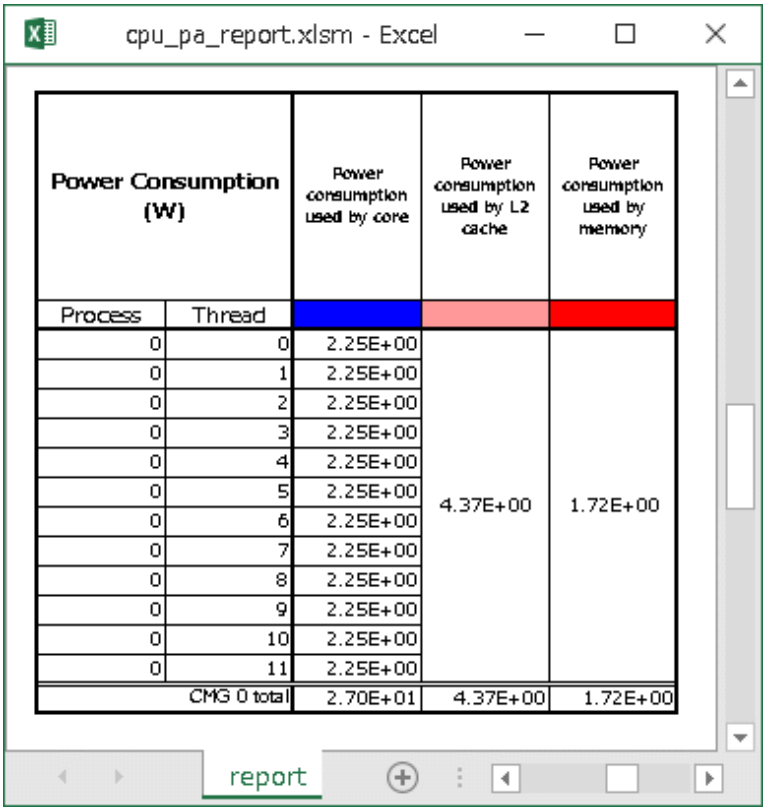
Table 4.23 Output Items in Data Transfer CMGs

	Name	Description
read	Own memory	Throughput performance of data transfer from Own memory to the target CMG (GB/s)
	Other memory	Throughput performance of data transfer from Other memory to the target CMG (GB/s)
	Tofu	Throughput performance of data transfer from Tofu to the target CMG (GB/s)
	PCI	Throughput performance of data transfer from PCI to the target CMG (GB/s)
write	Own memory	Throughput performance of data transfer from the target CMG to Own memory (GB/s)
	Other memory	Throughput performance of data transfer from the target CMG to Other memory (GB/s)
	Tofu	Throughput performance of data transfer from the target CMG to Tofu (GB/s)
	PCI	Throughput performance of data transfer from the target CMG to PCI (GB/s)

4.2.2.11 Power Consumption (W)

Power consumption (W) displays power consumption in W (watts) calculated by converting values obtained from the cores, L2 cache, and PMU counter. Power consumption (W) has a corresponding table and graph. It displays a stacked bar graph representing the output items in Table "Table 4.24 Output Items in Power Consumption (W)".

Figure 4.32 Layout of Power Consumption (W)



Power Consumption (W)		Power consumption used by core	Power consumption used by L2 cache	Power consumption used by memory
Process	Thread			
0	0	2.25E+00	4.37E+00	1.72E+00
0	1	2.25E+00		
0	2	2.25E+00		
0	3	2.25E+00		
0	4	2.25E+00		
0	5	2.25E+00		
0	6	2.25E+00		
0	7	2.25E+00		
0	8	2.25E+00		
0	9	2.25E+00		
0	10	2.25E+00		
0	11	2.25E+00		
	CMG 0 total	2.70E+01	4.37E+00	1.72E+00

Figure 4.33 Graph of Power Consumption (W)

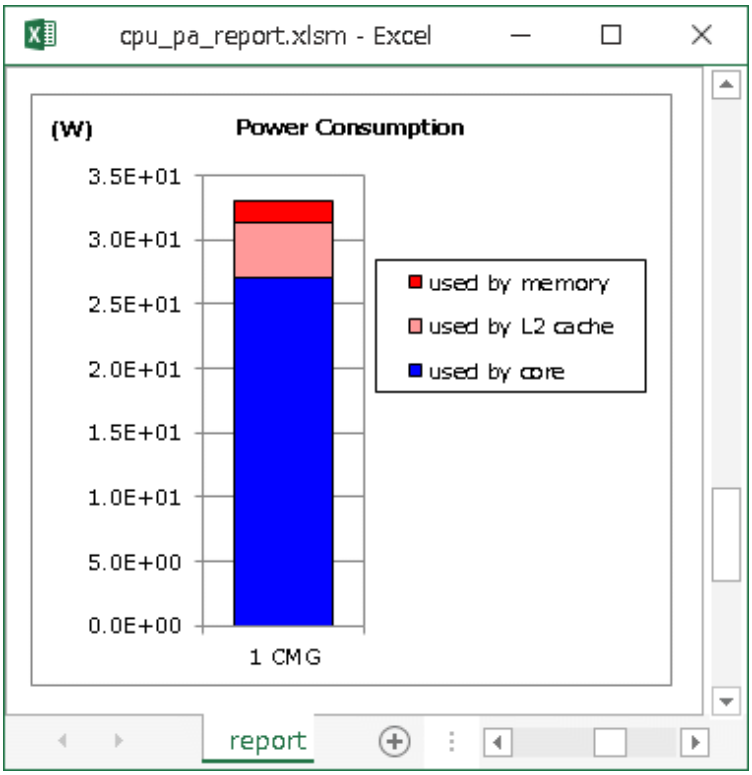


Table 4.24 Output Items in Power Consumption (W)

Name	Description
Power consumption used by core	Power consumed by cores

Name	Description
Power consumption used by L2 cache	Power consumed by L2 cache
Power consumption used by memory	Power consumed by memory

Chapter 5 Notes

This chapter provides notes on using the Profiler.

5.1 Notes Common to Profilers

5.1.1 COARRAY

When the compiler option -Ncoarray is enabled, note the following:

- Instant Performance Profiler and Advanced Performance Profiler
  - A value obtained by adding 1 to the rank or process number is equivalent to the image index.
  - The costs of the MPI library used by the COARRAY function may be appropriated.
  - "Type of program" of "Environment Information for Measurement Profiling Data" is not "MPI".
  - "Virtual coordinate" of "Environment Information for Measurement Profiling Data" is not output.
- Instant Performance Profiler
  - All Process numbers are output as "Process0".
- Advanced Performance Profiler
  - MPI communication cost information may be empty.
  - The image number is the rank number or process number plus 1.

5.1.2 Impact of Compiler Options

If you specify the following compiler options when compiling a program, the Profiler may behave in an unexpected way. For details on the compiler options mentioned in the text, see the "Fortran User's Guide", "C User's Guide", or "C++ User's Guide".

Compiler option -f{pie|PIE}

When you specify position-independent executable (PIE) program as execution file, profile data cannot be measured correctly.

5.1.3 Node-Sharing Job

If the type of a job is node-sharing job, profile data measurement may fail or correct information may not be output. In the case of MPI execution, profile data measurement fails and the program ends. For details on the node-sharing job, see "Job Operation Software" manuals.

5.1.4 Targets of Measurement of Thread Parallelization Information

The Performance Profiler operates in cooperation with a Fujitsu compiler. Therefore, only Fujitsu compiler automatic parallelization and OpenMP parallel processing can be targets of thread parallelization measurement. Pthread parallelization is not measured.

5.1.5 mpiexec command

- When you specify "mpiexec" command's option "-x LD_LIBRARY_PATH=value" and execute program, note the following:

  Instant Performance Profiler

  You have to specify "value" in the environment variable "LD_LIBRARY_PATH" to take preference over "/*installation_path*/lib64/fipp".

Example

```
fipp -C -d ./tmp mpiexec -x LD_LIBRARY_PATH=/installation_path/lib64/fipp:/installation_path/
lib64:./tmplib ./a.out
```

Advanced Performance Profiler

You have to specify "value" in the environment variable "LD_LIBRARY_PATH" to take preference over "/*installation_path*/lib64/fapp".

Example

```
fapp -C -d ./tmp mpiexec -x LD_LIBRARY_PATH=/installation_path/lib64/fapp:/installation_path/
lib64:./tmplib ./a.out
```

- Measurement performed with the mpiexec command with { -app | --app } option of the execution definition file specification method does not guarantee operation.

  For details on { -app | --app } option, see the "MPI User's Guide".

5.1.6 Impact of Using the MPI Profiling Interface

The Profiler measures information by hooking MPI functions, so the Profiler and MPI Profiling Interface cannot be used together. When the Profiler measures a program that hooks MPI functions using the MPI profiling interface, the following information may be output.

- Instant Performance Profiler and Advanced Performance Profiler
  - "Type of program" of "Environment Information for Measurement Profiling Data" is not "MPI".
  - "Virtual coordinate" of "Environment Information for Measurement Profiling Data" is not output.
- Instant Performance Profiler
  - All Process numbers are output as "Process0".
- Advanced Performance Profiler
  - MPI communication cost information may be empty.

5.1.7 Mixed Language Programming for MPI Program

You may use the option command when implementing including calling MPI library with different languages. Specify as follows at that time for linkage. For details on how to implement with different languages, see the "Fortran User's Guide", "C User's Guide", or "C++ User's Guide".

Command for linkage	Option
mpifccpx command (native compiler : mpifcc command) mpiFCCpx command (native compiler :mpiFCC command)	-lfjprofmpif
mpifrtpx command (native compiler : mpifrt command)	-lfjprofmpi

5.1.8 Exit Status

When measured by the Profiler, the outputted exit status may be the exit status of a target program or the exit status of the Profiler.

5.1.9 LD_PRELOAD

Do not use the environment variable LD_PRELOAD. The fipp or fapp command does not work correctly when LD_PRELOAD is used.

5.2 Notes on the Instant Performance Profiler

5.2.1 Impact of Compiler Options

If you specify the following compiler options when compiling a program, Instant Performance Profiler cannot be measured correctly. For details on the compiler options mentioned in the text, see the "Fortran User's Guide", "C User's Guide", or "C++ User's Guide".

Optimization option

In the case of a C or C++ language program, specify the optimization option (-O1 or greater) at the time of compilation in order to measure the loop information of the program with the Instant Performance Profiler.

Compiler option -g

When the compiler option -g is enabled, do not use the Instant Performance Profiler. Using -g increases the amount of debug information, which can increase execution time and memory usage.

Compiler option -Nline, -ffj-line

Use the compiler option -Nline(default option) or -ffj-line when using the Instant Performance Profiler.

Compiler option -Nnoline , -ffj-no-line

When the compiler option -Nnoline or -ffj-no-line is enabled, costs for which the -Puserfunc option of the fipp command is used cannot be measured correctly. For such a program, enable the -Pnouserfunc option so that you can measure costs correctly. In addition, when the compiler option -Nnoline or -ffj-no-line is enabled, MPI library costs cannot be measured correctly.

Compiler option -Klto, -flto

When the compiler option -Klto or -flto is enabled, the following information output by the Instant Performance Profiler may not be displayed correctly.

Loop Cost Distribution Information

When you compile your C or C++ language program, the loop cost distribution information of its functions in the object file is not output.

Call graph information

A procedure name generated internally by link time optimization may be displayed as the procedure name to be displayed.

Source code information

The cost of each line may not be displayed correctly.

Compiler option -fno-debug-info-for-profiling

If the compiler option -fno-debug-info-for-profiling is enabled when you compile your C++ language program in Clang Mode, in the information that the compiler generates for the profiler, the same name may be assigned to different entities. In that case, you cannot distinguish between different entities with the same name in the result of the Instant Performance Profiler.

5.2.2 Prohibition of Catching or Issuing SIGVTALRM Signal

The Instant Performance Profiler measures profile data by catching a SIGVTALRM signal. If a SIGVTALRM signal is caught or issued in the program, profile data cannot be measured correctly.

5.2.3 Sampling Interval at the Time of Profile Data Measurement

The sampling interval at the time of profile data measurement may not achieve the specified time because it is affected by a proximate OS timer interrupt interval that is smaller than the sampling interval value. Specifically, the following impacts occur:

When a sampling interval smaller than the OS timer interrupt interval is specified

It is rounded up to the OS timer interrupt interval value.

When a sampling interval larger than the OS timer interrupt interval is specified

It is rounded to a multiple of the OS timer interrupt interval that is proximate and smaller than the specified value.

Although it depends on the anti-noise measures by the OS, the timer interrupt interval is approximately 11 to 14 milliseconds. The following shows an example regarding the contradistinction between sampling interval and timer interrupt interval.

Example

When the timer interrupt interval is 14 milliseconds

- Value specified in the -i is 10: Sampling interval of 14 milliseconds
- Value specified in the -i is 25: Sampling interval of 14 milliseconds
- Value specified in the -i is 100: Sampling interval of 98(=14*7) milliseconds

5.2.4 Profiler Workspace

If the workspace of the Instant Performance Profiler becomes insufficient at the time of profile data measurement, the following message is output when the program ends.

```
fipp: work memory overflowed. specify memsize or more to -m option and retry.
```

memsize in the message is a recommended value for the Instant Performance Profiler workspace used at the time of profile data measurement. If this message is output, enlarge the Instant Performance Profiler workspace by specifying the -m *memsize* option with the fipp command. Then, measure profiled data again.

5.2.5 -pall Option

When the -pall option is specified with the fipppx command or fipp command, the information of all processes is output. If a parallel process is large, processing may take time. The fipp command measures the profile data of all processes. If the execution of the fipppx command or fipp command takes time, we recommend to output data in batches by using saved profile data and limiting processes, instead of outputting the information of all processes at one time.

5.2.6 CPU Performance Characteristics

The Instant Performance Profiler itself is included in the CPU Performance Characteristics.

5.2.7 Cost Information

- The costs of the functions with the same name are appropriated as the costs of a single function, even if they are substantially different. If the optimization option (-O1 or higher) is specified at the time of compilation, the line information of each instruction may become different from the source lines due to an impact of optimization. This applies, for example, to the case where the line information of each instruction becomes the starting position of a loop and costs are appropriated at the starting point of the loop.
- When the information total level for output items is Application, the cost of the entire program will be output.
- Since loop cost distribution information and line cost distribution information contain costs that are not included in aggregate calculation, the cost of the entire program may not match the total of individual costs.
- It is expected that the percentage of the cost of each task to the cost of the entire program is small. If you do not specify the number of outputs (-l option) for procedure information when outputting profile result, 10 procedure information items are output by default in the order of cost from highest. Therefore, task information may not be output.
- For a program created using a source code that includes a header file, profile data is output as the line numbers in the header file by handling cost information corresponding to the header file.
- In an inlined program, the cost of the inlined portion is counted as the cost of the inlined caller's procedure.
- When you measure costs with the Instant Performance Profiler, the following cost information may appear on the next line number of the function call line.
  - Waiting cost for synchronization between threads
  - MPI cost

- User-defined function cost measured with -Puserfunc option

Example

Ignore lines with no processing, such as comment lines.

```
67 void call_barrier()
68 {
69   int idx=0, cnt=10000;
70   for (idx=0; idx<cnt; idx++)
71     {
72       MPI_Barrier(MPI_COMM_WORLD);
73       /* comment */
74     }
75   return;
76 }
```

```
            MPI          %  Communication (s)     Line
-----------------------------------------------------
             33    94.2857             0.3430       --   Process      1
-----------------------------------------------------
             20    57.1429             0.2079       74   call_barrier
             12    34.2857             0.1247       18   main
              1     2.8571             0.0104       64   main
```

If the next operation (in the example, " } ") is on the same line, the line numbers do not shift.

```
67 void call_barrier()
68 {
69   int idx=0, cnt=10000;
70   for (idx=0; idx<cnt; idx++) { MPI_Barrier(MPI_COMM_WORLD); } return; }
```

```
               MPI          %  Communication (s)     Line
   -----------------------------------------------------
                33    91.6667             0.3362       --   Process      1
   -----------------------------------------------------
                12    33.3333             0.1223       18   main
                 1     2.7778             0.0102       64   main
                20    55.5556             0.2038       70   call_barrier
```

- MPI cost and Waiting cost for synchronization between threads information may not be output due to optimization effects.

5.2.8 Source Code Information

If no user procedures are included in the top five procedures in the procedure cost distribution information of the Instant Performance Profiler, source code information is not output. If this applies, the following message is output.

```
Symbol information up to the 5th do not include information which relates to the source code.
```

5.2.9 Call Graph Information

- For the call graph information of the Instant Performance Profiler, the call traces of the following programs may not be able to be analyzed.

  When the frame pointer register is not guaranteed due to optimization effects

  If the nest level of the call trace is 1, the nest level is output as "<???>". If the nest level of the call trace is 2 or higher, call graph information is output in error. In order to output call graph information correctly, specify the following option at the time of compilation.

  - For Fortran and C/C++ (Trad Mode) : -Knoomitfp option

- For C/C++ (Clang Mode) : -fno-omit-frame-pointer option

For more information, see the "Fortran User's Guide", "C User's Guide", or the "C++ User's Guide".

When the nest level of a call trace becomes 128 or higher

The nest level is output as "<???>".

When the sampling interrupt occurs at the entry or exit of the procedure

call graph information may be output incorrectly, failing to correctly analyze the call source. This phenomenon may have occurred if line cost distribution information contains the costs of the following lines:

For a Fortran program

- SUBROUTINE statement
- FUNCTION statement
- ENTRY statement
- RETURN statement
- END statement

For C/C++ program

- Brace that indicates the start of a function
- Brace that indicates the end of a function
- return statement

- In an inlined program, the cost of the inlined portion is counted as the cost of the inlined caller's procedure.
- The following two of a certain procedure may not be equal:
  - The value of the procedure cost at the Application level in the procedure cost distribution information
  - The total value of the procedure cost in the call graph information

5.2.10 Cost Information for Line Number 0

There may be costs accrued at line number 0. If cost information for line number 0 exists, the output is as follows:

Cost Information

Outputs cost information on line number 0 in the process unit in which the cost was accrued. Multiple lines may be output. The procedure name includes the procedure name in which the cost was accrued.

Source Code Information

Outputs the total cost accrued at line number 0 to the first line of source code (line0). Fixed values "/* other costs */" are output to the source code.

5.2.11 -u option

- If the procedure name is changed when compiled, the procedure name and the generated procedure name may not match and sum correctly. For the procedure name after compiling, see the "Fortran User's Guide", "C User's Guide", or "C++ User's Guide".
- When specifying -u option and -f *func_name* option at the same time, there are the following precautions.
  - If a generated procedure name is specified for the -f *func_name* option, outputs the procedure information to which the generated procedure belongs.
  - If the cost of procedure or generated procedure is 0 and the total cost of procedure and generated procedure is 1 or greater, it outputs a warning message and ignores the -f *func_name* option.
- In some cost information, *procedure_name* may not be output (refer to Created procedure name on "Table 2.6 Names of Generated Procedures for Thread Parallel Programs" for *procedure_name*). In this case, if you specify -u option, the output item, "@name", of the cost information will be blank.

5.2.12 -Minlined option

In the case when the executable contains many functions inlined by compiler optimization, the Instant Performance Profiler may take a long time to process or use more memory.

5.2.13 Sampling Number

If you use the Instant Performance Profiler to measure a program that includes sleep functions(sleep, usleep, nanosleep) or input/output statement processing, the sampling number of cost information may not be output correctly.

5.2.14 Signal Interrupt by Sampling

The Instant Performance Profiler sampling is implemented with signal interrupts (SIGVTALRM). Therefore, if you use a function that is affected by a signal interrupt, it may not work as expected. In this case, you should modify the program assuming that a signal interrupt occurs.

5.3 Notes on the Advanced Performance Profiler

5.3.1 MPI Thread Support

If the thread support level is MPI_THREAD_SERIALIZED or MPI_THREAD_MULTIPLE, the Advanced Performance Profiler cannot measure profile data correctly.

5.3.2 CPU Performance Analysis Information

If you specify the -Hmethod=fast option, information is measured even while the process for which measurement is performed is in the Sleep state and is not assigned to a CPU. For example, the value of the execution time of the CPU performance analysis function may be greater compared with the case where the -Hmethod=normal option is specified.

The Advanced Performance Profiler itself is included in the CPU Performance Analysis information.

5.3.3 -Hevent_raw Option

If the -Hevent_raw option is specified together with the -Hmethod=normal option, you cannot specify the same event number multiple times. The following error message is output, and the program is aborted.

```
RTINF2xxx : Internal error. PAPI return code = xxx.
```

5.3.4 Elapsed Time Information for MPI Library

In the case of an application created with the mpiFCC command, the name of a C++ language member function is output as the MPI library name for the elapsed time information of an MPI library. The Advanced Performance Profiler cannot measure the elapsed time information of an MPI library for MPI functions called from a child thread.

5.3.5 CPU Binding

When measuring profile data, you must control the bindings so that threads and CPU have a one-to-one relationship.

For more information about CPU binding, see the "Fortran User's Guide", "C User's Guide", or "C++ User's Guide" when using thread-parallel program.
Use the VCOORD file for CPU binding when using non-thread-parallel and MPI program (using mpiexec command). See { -vcoordfile | --vcoordfile } option in "MPI User's Guide" to specify the VCOORD file. Set the number of CPUs (cores) to "1" (core=1) for all processes in the VCOORD file.
Use "taskset" or "numactl" command for CPU binding when using neither thread-parallel nor MPI program. For more information, see the man page.

5.3.6 Routines that Cannot Measure MPI Communication Cost Information

Some routines defined in "mpi_f08.mod" or "mpi_f08_ext.mod" cannot measure "3.2.2.3 MPI Communication Cost Information". For "mpi_f08.mod" and "mpi_f08_ext.mod", see "MPI User's Guide".

5.4 Notes on the CPU Performance Analysis Report

5.4.1 CPU Performance Analysis Report File

If CPU Performance Analysis Report File with "4.1.6 Creating a CPU Performance Analysis Report"(cpu_pa_report.xlsm) is saved to a file, new read processing is not performed. To read different measurement result, do not save the file after data reading, or copy a new CPU Performance Analysis Report file.

5.4.2 Dynamically Generated Processes

The CPU Performance Analysis Report does not support dynamically generated processes.

Appendix A Troubleshooting

The following describes troubleshooting for the Profiler.

A.1 Instant Performance Profiler

A.1.1 Performing profile data measurement results in longer execution time compared with normal execution

When measuring profile data, if the execution time is longer than the normal execution time, one of the following causes may occur.

- This may be due to the sampling overhead of the Instant Performance Profiler. By increasing the sampling interval with the -i option of the fipp command, you can reduce the number of overhead occurrences and reduce the execution time of the program. For information about the fipp command, see the "2.1.4 Measuring Profile Data ".
- The Instant Performance Profiler reads the information required for measurement from the program, but if the number of lines, files, or symbols in program is large, the loading process may take much time. If the scope of measurement can be limited to a specific object (for example, by using the -Sregion option), you can reduce the time of loading process by specifying the -Nnoline or -ffj-no-line option when generating objects that are not to be measured. For -Nnoline or -ffj-no-line option, see the "Fortran User's Guide", "C User's Guide", or "C++ User's Guide". However, objects with the -Nnoline or -ffj-no-line option do not output the following information.
  - "Procedure start line number" and "Procedure end line number" in "2.2.2.4.1 Procedure Cost Distribution Information "
  - "2.2.2.4.2 Loop Cost Distribution Information"
  - "2.2.2.4.3 Line Cost Distribution Information"

A.1.2 Memory usage increases when measuring the profile data compared with the normal operation

If the memory usage increases when measuring the profile data compared with the normal operation, the following causes may apply.

When measuring the profile data by the Instant Performance Profiler, area of memory for the procedure information, loop information and line information of the executable file (hereinafter referred to as a debug information) is stored as a process in an area of memory separate from the area allocated by using the fipp command with the -m memsize option. Therefore, the memory usage may increase to cause the profile data measurement to fail due to insufficient memory. In that case, reduce the debug information for the executable file and measure the profile data. As a guide for the memory usage for debug information, appropriately 300 byte per procedure, 150 byte per loop, and 150 byte per line are required, respectively. However, the debug information changes depending on the length of the procedure name, number of execution threads, and options specified for the Instant Performance Profiler. In addition, you can reduce the debug information by enabling the compiler option -Nnoline or -ffj-no-line when compiling the objects so that the memory for the loop information and line information will not be allocated any more. For -Nnoline option, see the "Fortran User's Guide", "C User's Guide", or "C++ User's Guide". However, objects with the -Nnoline option do not output the following information.

- "Procedure start line number" and "Procedure end line number" in "2.2.2.4.1 Procedure Cost Distribution Information "
- "2.2.2.4.2 Loop Cost Distribution Information"
- "2.2.2.4.3 Line Cost Distribution Information"

A.1.3 A procedure name that does not exist in the source code (such as a library name) appears

To measure profile data only with user procedure names, specify the -Puserfunc option with the fipp command. For details on the fipp command, see "2.1.4 Measuring Profile Data".

A.1.4 Fail to open profile data

It is possible that the application that is a target of profile data measurement has not ended normally. Measure the profile data again.

A.1.5 The symbol __?unknown is output

During profile data measurement, it may be found that a cost does not correspond to any procedure. This cost is output with the symbol name "__?unknown". When measuring thread parallel programs, waiting cost for synchronization between threads may apply to this case. The output of this symbol may be suppressed by enlarging the sampling interval with the -i option of the fipp command at the time of profile data measurement. Also, the -Icall and -Inocall options of the fipp command differ in the process of tracing the frame pointers, and if you measure call graph information with the -Icall option of the fipp command at the time of profile data measurement, the symbol may not be output. For details of the fipp command, see the "2.1.4 Measuring Profile Data".

A.2 Advanced Performance Profiler

A.2.1 Performing profile data measurement results in longer execution time compared with normal execution

You can shorten the execution time of the Advanced Performance Profiler by reducing the number of measurement regions and the number of calls in the Advanced Profiler routine.

A.2.2 Fail to open profile data

It is possible that the application that is a target of profile data measurement has not ended normally. Measure the profile data again.

A.3 CPU Performance Analysis Report

A.3.1 Fail to load CSV Format File (File line limit exceeded)

If the CSV format file has more than 1048576 lines, Excel cannot load the CSV format file because it exceeds the maximum number of lines that Excel can handle. In this case, the following message will be output. The workaround is to keep the number of lines in the CSV format file within the maximum number of lines (1048576 lines).

```
The file format is not supported.
```

Here is how to reduce the number of lines in the CSV file. (Can be used in combination)

1. Reduce the number of measurement regions of fapp_start/fapp_stop.
2. Use the argument level of fapp_start/fapp_stop.
   After dividing the levels by the level argument, use the -L option to suppress the output of unnecessary regions.
3. The -Inompi option of the fapppx command suppresses the output of MPI information.
4. By specifying the -p<n>,limit=<m> option of the fapppx command, reduce the output information.

Appendix B List of Messages

This appendix describes typical messages output by the Profiler. Messages are output to the standard error output.

B.1 List of Message (fipp command)

fipp: -C or -A option is not specified.

[Message Explanation]

The -C option or -A option is not specified.

[System Behavior]

The system terminates the processing.

[User Response]

Specify the -C option or -A option.

fipp: -d option is not specified.

[Message Explanation]

The -d option is not specified.

[System Behavior]

The system terminates the processing.

[User Response]

Specify the -d option.

fipp: The specified argument of -d option is not directory.

[Message Explanation]

The argument of the -d option is incorrectly specified.

[System Behavior]

The system terminates the processing.

[User Response]

Confirm whether the argument of the -d option is correctly specified.

fipp: The specified directory in -d option is permission denied.

[Message Explanation]

The directory specified in the -d option does not have read, write, or execute permission.

[System Behavior]

The system terminates the processing.

[User Response]

Set read, write, and execute permission on the directory specified in the -d option.

fipp: The executable program was not specified to an operand.

[Message Explanation]

No executable file is specified, or the executable file does not exist.

[System Behavior]

The system terminates the processing.

[User Response]

Specify the existing executable file.

---

fipp: The specified argument of -I *parm* option is invalid.

[Message Explanation]

The wrong argument *parm* is specified in the -I option.

[Parameters Explanation]

parm **:** Argument specified by the user

[System Behavior]

The system terminates the processing.

[User Response]

Correct the argument *parm*.

---

fipp: The specified argument of -l *parm* option is not integer.

[Message Explanation]

The non-numerical argument *parm* is specified in the -l option.

[Parameters Explanation]

parm **:** Argument specified by the user

[System Behavior]

The system terminates the processing.

[User Response]

Correct the argument *parm*.

---

fipp: The specified value of -l *parm* option is outside the range. The default value is applied. { limit = 0 }

[Message Explanation]

The argument *parm* specified in the -l option is outside the range.

[Parameters Explanation]

parm **:** Argument specified by the user

[System Behavior]

The system enables the -l 0 option and continues the processing.

[User Response]

Correct the argument *parm*.

---

fipp: The specified argument of -H option is invalid.

[Message Explanation]

The argument of the -H option is incorrectly specified.

[System Behavior]

The system terminates the processing.

[User Response]

Correct the argument of the -H option.

---

fipp: The specified argument of -P *parm* option is invalid.

[Message Explanation]

The argument *parm* specified in the -P option is incorrect.

[Parameters Explanation]

parm : Argument specified by the user

[System Behavior]

The system terminates the processing.

[User Response]

Correct the argument *parm*.

---

fipp: -Inocall option cannot be specified together with -Puserfunc option.

[Message Explanation]

The -Puserfunc and -Inocall options are specified together.

[System Behavior]

The system terminates the processing.

[User Response]

When the -Puserfunc option is specified, specify the -Icall option.

---

fipp: The -Icall option is necessary for specified -Puserfunc option.

[Message Explanation]

The -Puserfunc option is specified, but the -Icall option is not specified.

[System Behavior]

The system terminates the processing.

[User Response]

When the -Puserfunc option is specified, specify the -Icall options.

---

fipp: The specified argument of -S *parm* option is invalid.

[Message Explanation]

The argument *parm* specified in the -S option is incorrect.

[Parameters Explanation]

parm : Argument specified by the user

[System Behavior]

The system terminates the processing.

[User Response]

Correct the argument *parm*.

fipp: The specified argument of -i *parm* option is not integer.

[Message Explanation]

The non-numerical argument *parm* is specified in the -i option.

[Parameters Explanation]

parm **:** Argument specified by the user

[System Behavior]

The system terminates the processing.

[User Response]

Correct the argument *parm*.

fipp: The specified value of -i *parm* option is outside the range. The default value is applied. { interval = 100 }

[Message Explanation]

The argument *parm* specified in the -i option is outside the range.

[Parameters Explanation]

parm **:** Argument specified by the user

[System Behavior]

The system enables the -i 100 option and continues the processing.

[User Response]

Correct the argument *parm*.

fipp: The specified argument of -m *parm* option is not integer.

[Message Explanation]

The non-numerical argument *parm* is specified in the -m option.

[Parameters Explanation]

parm **:** Argument specified by the user

[System Behavior]

The system terminates the processing.

[User Response]

Correct the argument *parm*.

fipp: The specified value of -m *parm* option is outside the range. The default value is applied. { memsize = 3000 }

[Message Explanation]

The argument *parm* specified in the -m option is outside the range.

[Parameters Explanation]

parm **:** Argument specified by the user

[System Behavior]

The system enables the -m 3000 option and continues the processing.

[User Response]

Correct the argument *parm*.

fipp: The specified value of -m *parm* option is within the range but large. Therefore, the working memory may not be allocated.

[Message Explanation]

The argument *parm* specified in the -m option is large. If the product of the number of processes per node and the number of threads per process is large, the working memory may not be allocated.

[Parameters Explanation]

parm : Argument specified by the user

[System Behavior]

The system continues the processing.

[User Response]

If necessary, reduce the product of the number of processes per node and the number of threads per process, or the argument *parm*.

fipp: The specified argument of -L *parm* option is invalid.

[Message Explanation]

The argument *parm* specified in the -L option is incorrect.

[Parameters Explanation]

parm : Argument specified by the user

[System Behavior]

The system terminates the processing.

[User Response]

Correct the argument *parm*.

fipp: The specified argument of -W *parm* option is invalid.

[Message Explanation]

The argument *parm* specified in the -W option is incorrect.

[Parameters Explanation]

parm : Argument specified by the user

[System Behavior]

The system terminates the processing.

[User Response]

Correct the argument *parm*.

fipp: Cannot specify the "-Wspawn" option when executing non-MPI program.

[Message Explanation]

Specifies the non-MPI program with the -Wspawn option.

[System Behavior]

The system terminates the processing.

[User Response]

Do not specify the -Wspawn option.

fipp: The profiling data is not correctly generated.

[Message Explanation]

The file format of the executable file is incorrect.

[System Behavior]

The system terminates the processing.

[User Response]

Check the file format of the executable file.

fipp: The files had already existed in the specified value of fjprof_spawn_dir_name key.

[Message Explanation]

A directory specified for the info key "fjprof_spawn_dir_name" already exists.

[System Behavior]

The system terminates the processing.

[User Response]

Check whether the value specified for fjprof_spawn_dir_name is correct.

fipp: The specified value of fjprof_spawn_dir_name key is not directory.

[Message Explanation]

A file having the same name as the directory specified for the info key "fjprof_spawn_dir_name" already exists.

[System Behavior]

The system terminates the processing.

[User Response]

Check whether the value specified for fjprof_spawn_dir_name is correct.

fipp: The specified value of fjprof_spawn_dir_name key is permission denied.

[Message Explanation]

You do not have permission to read, write, or execute on the directory specified for the info key "fjprof_spawn_dir_name".

[System Behavior]

The system terminates the processing.

[User Response]

Set read, write, and execute permission on the directory specified for fjprof_spawn_dir_name.

fipp: obsolete option *parm1* changed to *parm2*.

[Message Explanation]

parm1 is an old option. Change it to the *parm2* option.

[Parameters Explanation]

parm1 **:** old option name

parm2 : option name

[System Behavior]

The system enables the *parm2* option and continues the processing.

[User Response]

Change the *parm1* option to the *parm2* option.

fipp: *parm1* option was specified. *parm2* option is ignored.

[Message Explanation]

Since the *parm1* option is specified, the *parm2* option is disabled.

[Parameters Explanation]

parm1 **:** Option name to be enabled

parm2 : Option name to be disabled

[System Behavior]

The system enables only the *parm1* option and continues the processing.

[User Response]

When the *parm1* option is specified, do not specify the *parm2* option.

fipp: *parm1* option cannot be specified together with *parm2* option.

[Message Explanation]

You specified the *parm1* option and the *parm2* option at the same time.

[Parameters Explanation]

parm1 : Option name

parm2 : Option name

[System Behavior]

The system terminates the processing.

[User Response]

Delete either the *parm1* option or the *parm2* option.

fipp: The specified argument of -M *parm* option is invalid.

[Message Explanation]

The wrong argument *parm* is specified in the -M option.

[Parameters Explanation]

parm : Argument specified by the user

[System Behavior]

The system terminates the processing.

[User Response]

Correct the argument *parm*.

B.2 List of Message (fipppx command)

fipppx: -A option is not specified.

[Message Explanation]

The -A option is not specified.

[System Behavior]

The system terminates the processing.

[User Response]

Specify the -A option.

---

fipppx: invalid argument of option -- *parm*

[Message Explanation]

The argument of *parm* option is incorrectly specified.

[Parameters Explanation]

parm: Invalid option name or argument

[System Behavior]

The system terminates the processing.

[User Response]

Correct the option or argument based on the *parm* information.

---

fipppx: No information on the specified region. : *func_name*

[Message Explanation]

There is no cost information for the procedure name *func_name* specified with the -f option.

[Parameters Explanation]

func_name : procedure name

[System Behavior]

The system ignores this option and continues the processing.

[User Response]

No special action is required.

---

fipppx: *parm1* option cannot be specified together with *parm2* option.

[Message Explanation]

You specified the *parm1* option and the *parm2* option at the same time.

[Parameters Explanation]

parm1 : Option name

parm2 : Option name

[System Behavior]

The system terminates the processing.

[User Response]

Delete either the *parm1* option or the *parm2* option.

B.3 List of Message (fapp command)

fapp: -C or -A option is not specified.

[Message Explanation]

The -C option or -A option is not specified.

[System Behavior]

The system terminates the processing.

[User Response]

Specify the -C option or -A option.

fapp: -d option is not specified.

[Message Explanation]

The -d option is not specified.

[System Behavior]

The system terminates the processing.

[User Response]

Specify the -d option.

fapp: The specified argument of -d option is not directory.

[Message Explanation]

The argument of the -d option is incorrectly specified.

[System Behavior]

The system terminates the processing.

[User Response]

Correct the argument of the -d option.

fapp: The specified directory in -d option is permission denied.

[Message Explanation]

The directory specified in the -d option does not have read, write, or execute permission.

[System Behavior]

The system terminates the processing.

[User Response]

Set read, write, and execute permission on the directory specified in the -d option.

fapp: The specified argument of -I *parm* option is invalid.

[Message Explanation]

The wrong argument *parm* is specified in the -I option.

[Parameters Explanation]

parm **:** Argument specified by the user

[System Behavior]

The system terminates the processing.

[User Response]

Correct the argument *parm*.

---

fapp: The specified argument of -H option is invalid.

[Message Explanation]

The argument of the -H option is incorrectly specified.

[System Behavior]

The system terminates the processing.

[User Response]

Correct the argument of the -H option.

---

fapp: The specified *parm* argument of -H option is invalid.

[Message Explanation]

The argument of the -H option is incorrectly specified.

[Parameters Explanation]

parm **:** "event=", "event_raw=", "mode=", or "method="

[System Behavior]

The system terminates the processing.

[User Response]

Correct the argument of the -H option.

---

fapp: The number of PMU event specified to -Hevent_raw option exceed the limit. (max = 8)

[Message Explanation]

Too many numerical values were specified as arguments of the -Hevent_raw= option.

[System Behavior]

The system terminates the processing.

[User Response]

Correct the argument of the -H option.

---

fapp: The specified argument of -W *parm* option is invalid.

[Message Explanation]

The argument *parm* specified in the -W option is incorrect.

[Parameters Explanation]

parm **:** Argument specified by the user

[System Behavior]

The system terminates the processing.

[User Response]

Correct the argument *parm*.

fapp: The executable program was not specified to an operand.

[Message Explanation]

No executable file is specified, or the executable file does not exist.

[System Behavior]

The system terminates the processing.

[User Response]

Specify the existing executable file.

fapp: The files had already existed in the specified value of fjprof_spawn_dir_name key.

[Message Explanation]

A directory specified for the info key "fjprof_spawn_dir_name" already exists.

[System Behavior]

The system terminates the processing.

[User Response]

Check whether the value specified for fjprof_spawn_dir_name is correct.

fapp: The specified value of fjprof_spawn_dir_name key is not directory.

[Message Explanation]

A file having the same name as the directory specified for the info key "fjprof_spawn_dir_name" already exists.

[System Behavior]

The system terminates the processing.

[User Response]

Check whether the value specified for fjprof_spawn_dir_name is correct.

fapp: The specified value of fjprof_spawn_dir_name key is permission denied.

[Message Explanation]

You do not have permission to read, write, or execute on the directory specified for the info key "fjprof_spawn_dir_name".

[System Behavior]

The system terminates the processing.

[User Response]

Set read, write, and execute permission on the directory specified for fjprof_spawn_dir_name.

fapp: *parm1* option was specified. *parm2* option is ignored.

[Message Explanation]

Since the *parm1* option is specified, the *parm2* option is disabled.

[Parameters Explanation]

parm1 **:** Option name to be enabled

parm2 : Option name to be disabled

[System Behavior]

The system enables only the *parm1* option and continues the processing.

[User Response]

When the *parm1* option is specified, do not specify the *parm2* option.

fapp: obsolete option *parm1* changed to *parm2*.

[Message Explanation]

parm1 is an old option. Change it to the *parm2* option.

[Parameters Explanation]

parm1 **:** Old option name

parm2 : Option name

[System Behavior]

The system enables the *parm2* option and continues the processing.

[User Response]

Change the *parm1* option to the *parm2* option.

fapp: *parm1* option cannot be specified together with *parm2* option.

[Message Explanation]

You specified the *parm1* option and the *parm2* option at the same time.

[Parameters Explanation]

parm1 : Option name

parm2 : Option name

[System Behavior]

The system terminates the processing.

[User Response]

Delete either the *parm1* option or the *parm2* option.

B.4 List of Message (fapppx command)

fapppx: -A option is not specified.

[Message Explanation]

The -A option is not specified.

[System Behavior]

The system terminates the processing.

[User Response]

Specify the -A option.

fapppx: invalid argument of option -- *parm*

[Message Explanation]

The argument of the *parm* option is incorrectly specified.

[Parameters Explanation]

parm **:** invalid option name or argument

[System Behavior]

The system terminates the processing.

[User Response]

Correct the argument or option *parm*.

---

fapppx: The -Hpa option is obsolete and will be removed in a future release. The option is ignored.

[Message Explanation]

The -Hpa option is no longer necessary. It is scheduled to be discarded in the future.

[System Behavior]

The system ignores this option and continues the processing.

[User Response]

No special action is required. However, we recommend not to specify the -Hpa option.

---

fapppx: obsolete option *parm1* changed to *parm2*.

[Message Explanation]

parm1 is an old option. Change it to the *parm2* option.

[Parameters Explanation]

parm1 **:** Old option name

parm2 : Option name

[System Behavior]

The system enables the *parm2* option and continues the processing.

[User Response]

Change the *parm1* option to the *parm2* option.