FUJITSU Software
Technical Computing Suite V4.0L20

# Development Studio Overview

J2UL-2471-02ENZ0(00)
March 2020

# Preface

## Purpose of This Manual

This manual provides an overview of the functions of Development Studio (hereafter referred to as this software). It also contains points to be noted when migrating from existing products.

This software supports the system (hereinafter referred to as "FX system") based on Fujitsu CPU A64FX. The product for the PC cluster composed by PC server PRIMERGY is not bundled.

## Intended Readers

This manual is written for users who develop scientific and technical computation programs, and for system administrators who install this software and set the environment settings.

## Organization of This Manual

This manual consists of the following chapters:

Chapter 1 Development Studio Overview

Provides an overview of the software.

Chapter 2 Functions of Components

Provides an overview of the function of each component.

Chapter 3 Notes on Migrating from Technical Computing Language V2/V3

Explains points to be noted when migrating from Technical Computing Language V2/V3 to this software.

## Export Controls

Exportation/release of this manual 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.
- Mac(R) and macOS(R) are registered trademarks of Apple Inc.
- Microsoft, Windows, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
- 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 |
|---|---|
| March 2020, 2nd Version | J2UL-2471-02ENZ0(00) |
| January 2020, 1st Version | J2UL-2471-01ENZ0(00) |

## Copyright

# Update History

| Changes | Location | Version |
|---|---|---|
| Added versions of OpenMP standards supported by the compilers. | 1.1 | 2nd Version |
| Added functional overview of GUI-based IDE (Integrated Development Environment). | 1.3.1、2.3.3 | |
| Added IDE User's Guide to the manual list. | 1.5 | |
| Added versions of standards supported by the Fortran compiler. | 2.1.1 | |
| Added versions of standards supported by the C compiler. | 2.1.2 | |
| Added versions of standards supported by the C++ compiler. | 2.1.3 | |
| Changed the look according to product upgrades. | - | |

# Contents

# Chapter 1 Development Studio Overview

This chapter presents an overview of this software.

## 1.1 What is Development Studio?

This software supports development and execution of high-performance parallel programs written in Fortran, C, or C++.

This software has the following features:

Assists with development of "high-functionality" parallel programs

- Compiler optimization technology that elicits high CPU execution performance
- Various functions that assist with highly parallelized programming using hybrid parallels (*)
    - Compiler automatic parallelization function that simplifies thread parallelization
    - High-functionality MPI communication library that underpins process parallelization
    - Optimized mathematical library and its parallelized edition (thread parallels, MPI parallels)

  *) Hybrid parallel is a parallel programming model containing both thread parallels and process parallels.

Assists with "efficient" development of large-scale parallel programs

- Compilers equipped with high-level optimization functions and automatic parallelization functions
- Profiler that underpins program performance tuning
- Debugger for parallel applications
- Optimized mathematical library and its parallelized edition

Assists with development of "highly portable" programs

- Compilers that conform to international program language standards (Fortran, C, and C++)
- Support of various industry standard specifications
    - C and C++ compilers that support GNU compiler extended specifications
    - Compilers that conform to OpenMP 4.0, OpenMP 4.5 and a subset of OpenMP 5.0
    - MPI library which conforms to MPI-3.1 standard and a subset of MPI-4.0 standard (tentative name)
    - Provision of BLAS, LAPACK, and ScaLAPACK

## 1.2 Applicable Fields

Development Studio is effective in developing high performance technical applications and exert most capability in the following:

- Development and execution of large-scale scientific and technical computing application programs written in Fortran, C, and C++
- Development and execution of large-scale and highly parallelized application programs for process parallels and thread parallels

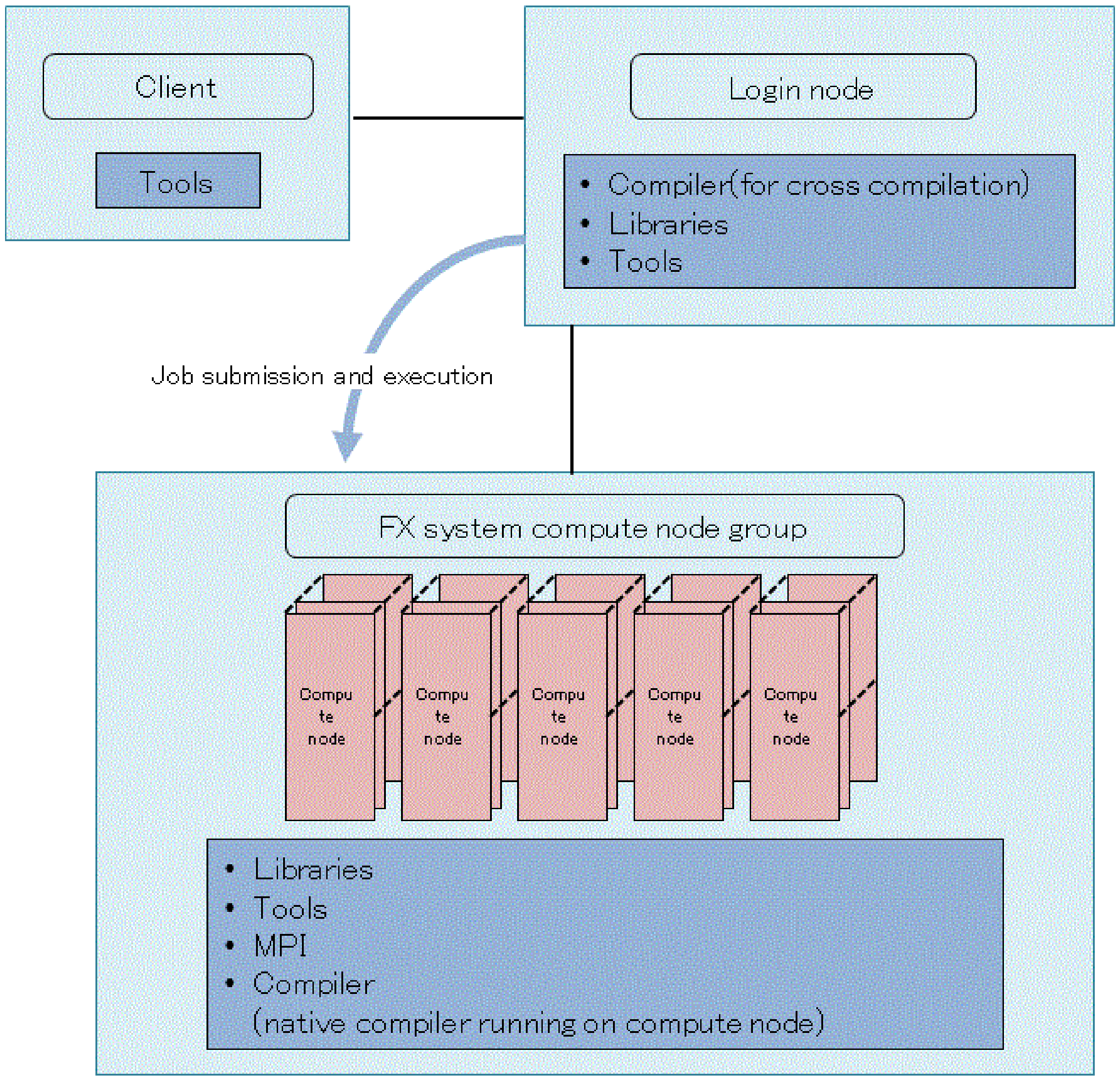
## 1.3 System on which this Software Runs

The types of system running this software can be broadly classified as follows:

1. Client
2. Login node
3. FX system (compute node group)

The example of system constitution is shown below. Refer to the Job Operation Software manual for details about system constitution.

Figure 1.1 Example of System constitution


Client
Tools
Login node
• Compiler(for cross compilation)
• Libraries
• Tools
Job submission and execution
FX system compute node group
Compu te node
Compu te node
Compu te node
Compu te node
Compu te node
• Libraries
• Tools
• MPI
• Compiler (native compiler running on compute node)

## 1.3.1 Client (Windows Operating System/ Mac Operating System/ Linux Operating System)

A client computer that uses an integrated development environment and GUI based operations to create and edit programs for FX system, launch compilers remotely for cross-compilation, submit jobs remotely for FX system displays the profiling results as CPU Performance Analysis Report.

- Tools
    - GUI-based components, for client, of the Integrated Development Environment
    - CPU Performance Analysis Report

## 1.3.2 Login Node (Linux Operating System)

The node on which the various compilers (for cross compilation) for FX system can run. Create an executable program for FX system by compilation and link-editing. This node uses job operation software to run programs as submit jobs on the FX system compute nodes.

To develop programs for FX system, install the following components for FX system:

- Compiler (for cross compilation)
    - Fortran compiler
    - C compiler
    - C++ compiler
    - MPI program compilation commands
- Libraries
    - Runtime libraries of each compiler
    - Mathematical library
    - MPI library
    - Library used for Software development support tools
- Tools
    - Processing part on the login node side of the Software development support tools
    - Commands to output the Profiling Results

### 1.3.3 FX System (Compute Nodes)

The node that runs an executable program for an FX system as a job.

In addition, there are compilers (for native compilation) that run on the compute nodes of an FX system. As part of the job execution, you can compilation and link-editing.

Install the following FX system components on the FX system compute nodes:

- Libraries
    - Runtime libraries of each compiler
    - Mathematical library
    - MPI library
    - Libraries, which are linked to executable files, of software development support tools
- Tools
    - Commands to get Profiling Data
- MPI
    - A command to execute MPI executable programs
- Compiler (for native compilation)
    - Fortran compiler
    - C compiler
    - C++ compiler
    - MPI program compilation commands

## 1.4 Program Development Flow

This section describes the flow of program development (Program creation, compilation, execution, debugging, tuning) using a file system (hereafter referred to as shared file system) that is shared by login nodes and compute nodes.

### 1.4.1 Creating and Executing a Program

1. Creating a program

   The source program is created using various functions provided by the Mathematical library and MPI library.

2. Compiling the program

   Create an executable program by compiling the source program.

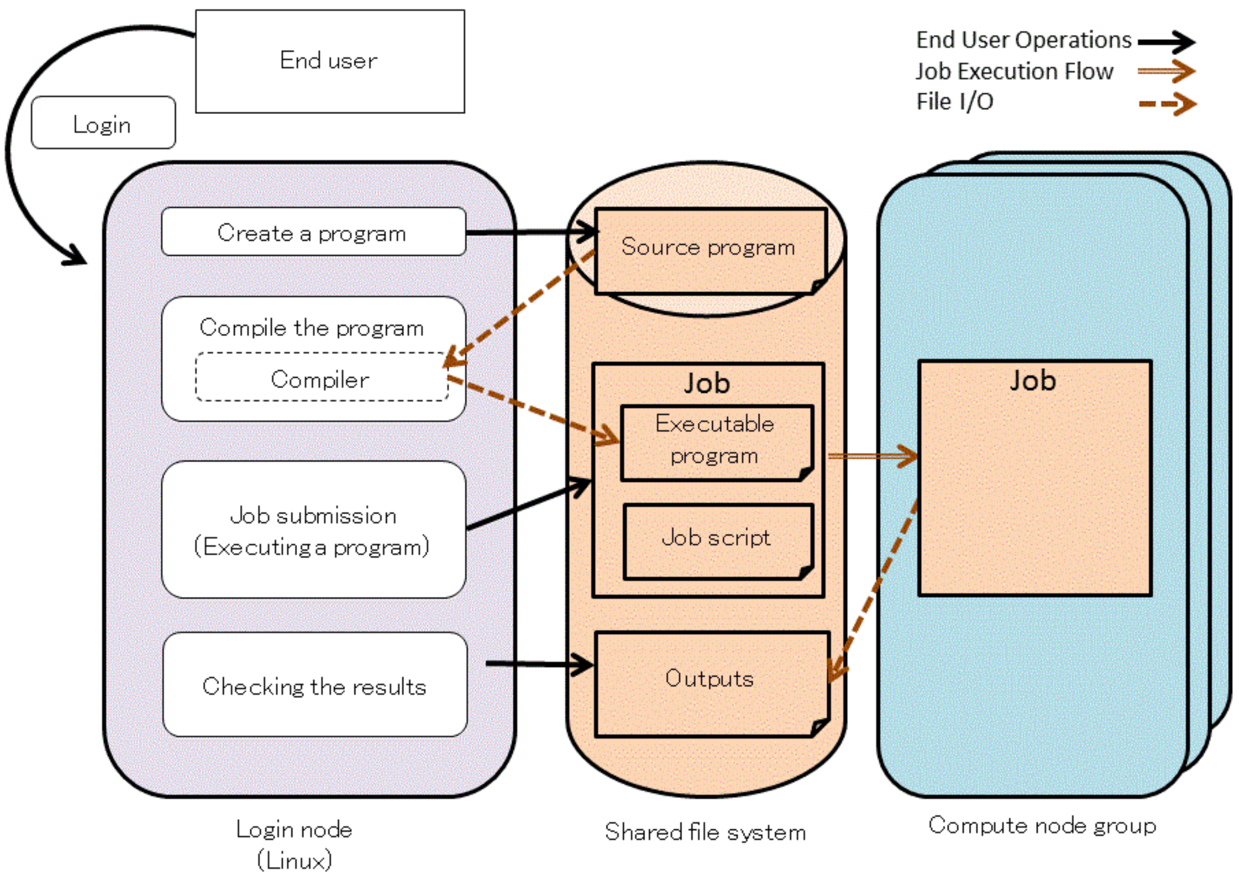
3. Job submission (Executing a program)

   Prepare a job script to execute the executable program as a job. Submit the job using the Job Operation Software.

4. Checking the results

   Verify that the execution results are as expected.

Figure 1.2 Creating and executing a program



### 1.4.2 Debugging a Program

Refer to the "Debugger for Parallel Applications User's Guide" for debugging details.

1. Compiling the program

   Compile the source program to create an executable program for debugging.

2. Job submission (Enabling the debugger for parallel applications)

   Prepare a job script (*) to execute the executable program as a job. Submit the job using the Job Operation Software.

   *) Specify the option to enable debugger for parallel applications.

3. Duplication removal of debug result

   De-duplicate the investigation result file to improve the readability by using the fjdbg_summary command.

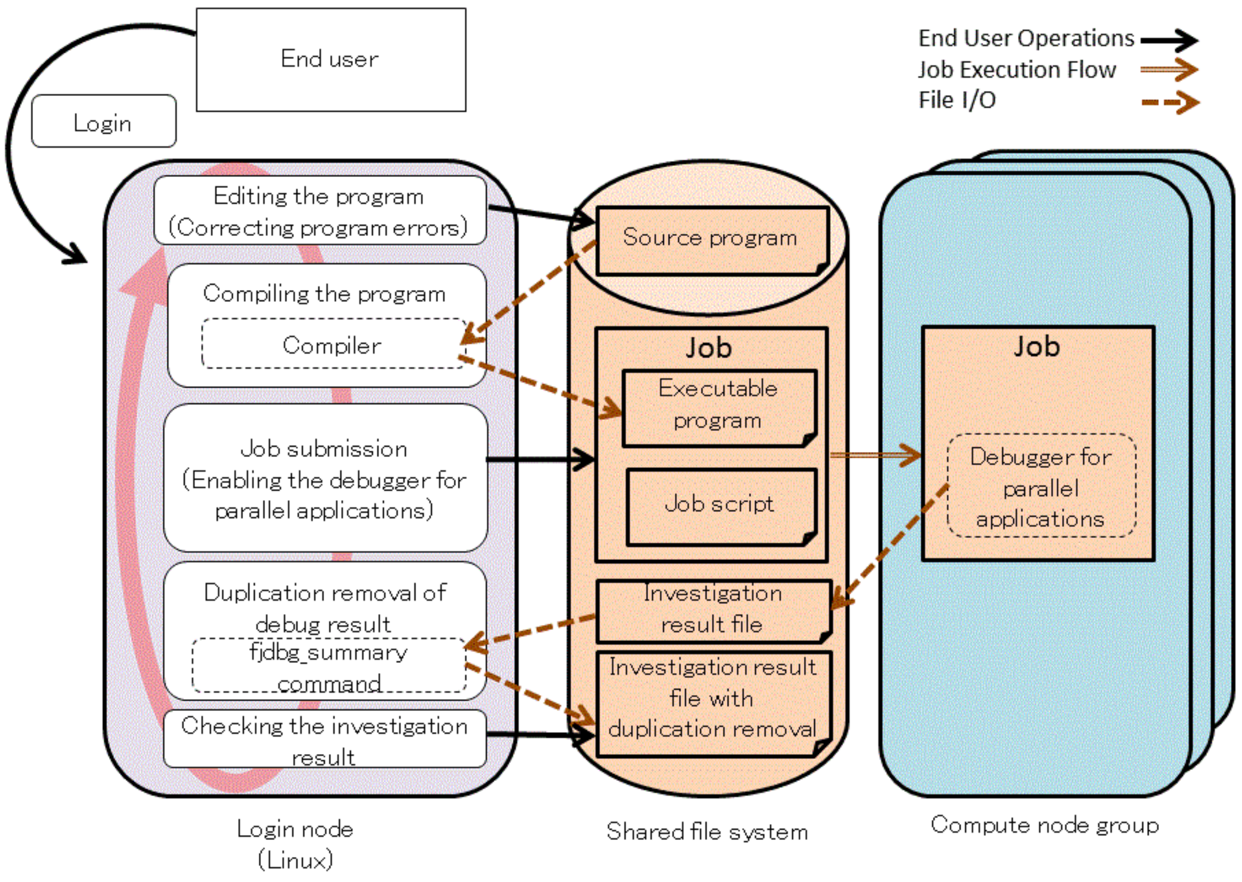
4. Checking the investigation result

   Refer the de-duplicated investigation result file and find the program error.

5. Editing the program (Correcting program errors)

   Correct errors in the source program.

Figure 1.3 Debugging a program


End user
Login
End User Operations
Job Execution Flow
File I/O
Editing the program (Correcting program errors)
Compiling the program
Compiler
Job submission (Enabling the debugger for parallel applications)
Duplication removal of debug result
fjdbg_summary command
Checking the investigation result
Source program
Job
Executable program
Job script
Investigation result file
Investigation result file with duplication removal
Job
Debugger for parallel applications
Login node (Linux)
Shared file system
Compute node group

## 1.4.3 Tuning a Program

Refer to the "Profiler User's Guide" for tuning details.

1. Compiling the program

   Create an executable program by compiling the source program.

2. Job submission (Performance measurement)

   Prepare a job script (*) to measure profiling data. Submit the job using the Job Operation Software.

   *) Use the profiler fipp or fapp command.

3. Output profiling results

   Use the fipppx or fapppx command to output profiling results from profiling data.

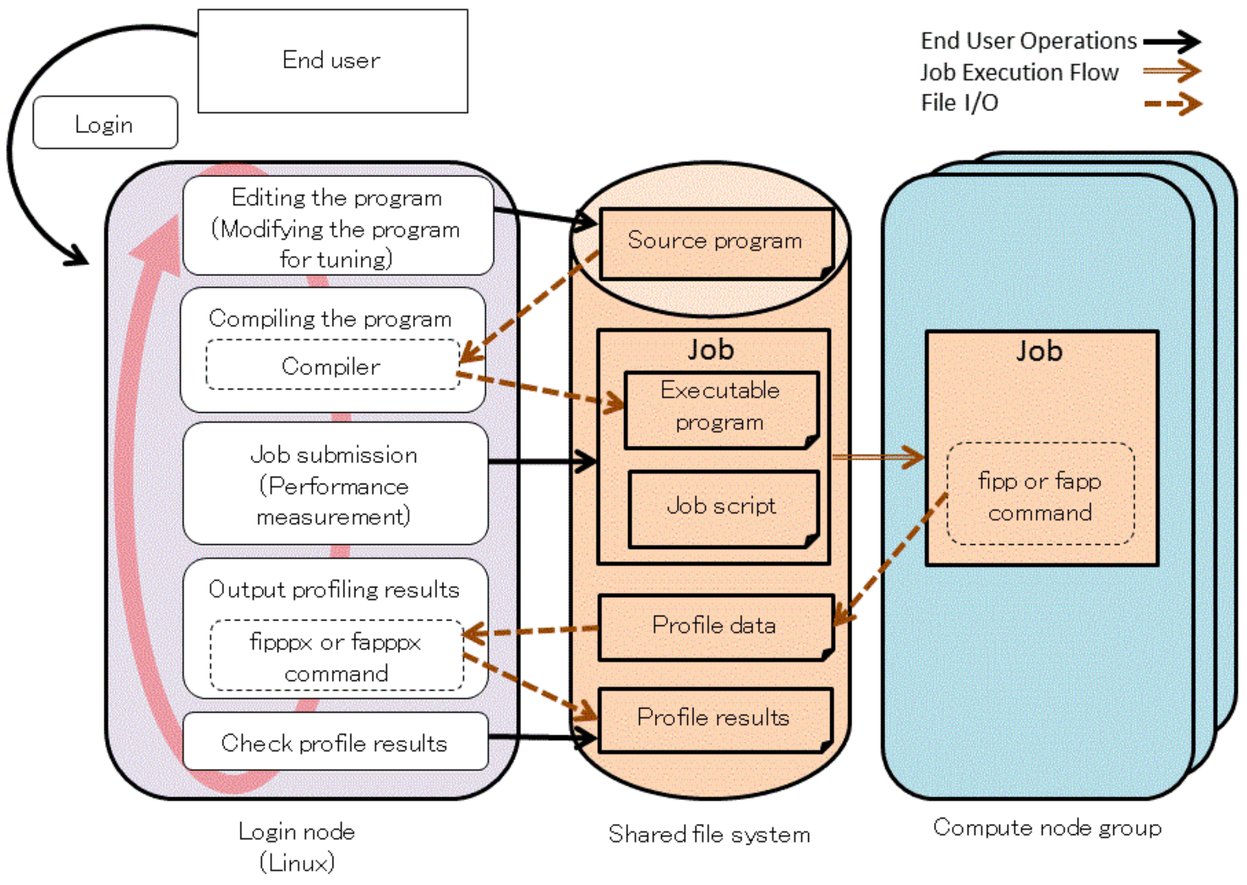
4. Check profile results

   Consider where to tune from the profile results.

5. Editing the program (Modifying the program for tuning)

   Modify and improve the source program.

Figure 1.4 Tuning a program



## 1.5 Manual List

The following manuals are provided with this software:

Table 1.1 Manual List

| Manual Name | Contents | Format |
|---|---|---|
| Overview | An overview of Development Studio (this document) | PDF |
| Fortran Language Reference | Describes the Fortran language specification supported by the Fortran compiler. | PDF |
| Fortran User's Guide | How to use the Fortran compiler<br>How to program in Fortran | PDF |
| Fortran User's Guide Additional Volume COARRAY | How to use the COARRAY feature<br>How to programming with COARRAY feature | PDF |
| C User's Guide | How to use the C compiler<br>Description of extended language specifications based on the C and OpenMP specifications for the C compiler | PDF |
| C++ User's Guide | How to use the C++ compiler<br>Description of extended language specifications based on the C++ and OpenMP specifications for the C++ compiler | PDF |
| Fortran Compiler Messages | Descriptions of messages during compilation using the Fortran compiler | PDF |

| Manual Name | Contents | Format |
|---|---|---|
| C/C++ Compiler Optimization Messages | Descriptions of optimization messages from the C and C++ compilers | PDF |
| Fortran/C/C++ Runtime Messages | Description of runtime messages from programs written in Fortran, C, and C++ | PDF |
| MPI User's Guide | How to use the MPI library | PDF |
| MPI User's Guide Additional Volume Java Interface | MPI Library Java Interface Description | HTML |
| uTofu User's Guide | How to use uTofu, the API for the Tofu interconnect | PDF |
| Profiler User's Guide | Description of features and usage of the profiler, a performance analysis tool | PDF |
| Debugger for Parallel Applications User's Guide | Descriptions of the functions and usage of the Debugger for Parallel Applications | PDF |
| IDE User's Guide | How to use the Integrated Development Environment | PDF |
| Programmer's Guide for Usage of Mathematical Libraries | How to use the Mathematical Libraries | PDF |
| SSL II User's Guide | Descriptions of the functions and use of the Scientific Subroutine Library II (SSL II) | PDF |
| FUJITSU SSL II Extended Capabilities User's Guide | | PDF |
| FUJITSU SSL II Extended Capabilities User's Guide II | | PDF |
| Fujitsu SSL II Thread-Parallel Capabilities User's Guide | Descriptions of the functions and usage of the Scientific Subroutine Library II Thread-Parallel Capabilities | PDF |
| Fujitsu C-SSL II User's Guide | How to use C-SSLII (C-Scientific Subroutine Library II), a scientific library available from C | PDF |
| Fujitsu C-SSL II Thread-Parallel Capabilities User's Guide | How to use the functions and usage of the C Scientific Subroutine Library II Thread-Parallel Capabilities | PDF |
| Fujitsu SSL II/MPI User's Guide | Descriptions of the functions and usage of the Scientific Subroutine Library II/MPI (SSL II/MPI) | PDF |
| BLAS LAPACK ScaLAPACK User's Guide | Obtain an overview of the functions provided by BLAS, LAPACK and ScaLAPACK, and how they relate to the BLAS, LAPACK, and ScaLAPACK published by Netlib | PDF |
| Fast Basic Operations Library for Quadruple Precision User's Guide | How to use Fast Basic Operations Library for Quadruple Precision available in Fortran and C++ | PDF |

View PDF manuals in the latest Adobe(R) Reader(R) version.

# Chapter 2 Functions of Components

## 2.1 Compiler

### 2.1.1 Fortran Compiler

The Fortran compiler compiles programs written in Fortran and can create executable programs that are highly optimized, in particular to achieve high CPU execution performance for targeted compute nodes. If required, the Fortran compiler can also create executable programs capable of automatic parallelization or thread-level parallel execution using OpenMP specifications. An execution environment for the fast and secure execution of these executable programs is also prepared.

An overview of Fortran compiler functions is presented below. Refer to the "Fortran User's Guide" for Fortran compiler details.

1. Language specifications

   The Fortran compiler supports the following standards:

   - ISO/IEC 1539-1:2018 (Fortran 2018 standard) a subset of specification
   - ISO/IEC 1539-1:2010 (Fortran 2008 standard)
   - ISO/IEC 1539-1:2004, JIS X 3001-1:2009 (Fortran 2003 standard)
   - ISO/IEC 1539-1:1997, JIS X 3001-1:1998 (Fortran 95 standard)
   - Fortran 90 standard and FORTRAN 77 standard
   - OpenMP API Version 5.0 a subset of specification
   - OpenMP API Version 4.5
   - OpenMP API Version 4.0

   The Fortran compiler also supports the main industry standard Fortran language specifications. Refer to the "Fortran Language Reference" for information on the language specifications supported by the Fortran compiler.

2. Main functions

   The Fortran compiler is equipped with an "Automatic parallelization function" that enables programs that automatically perform thread-level parallel processing to be created just by specifying the compile options. The Fortran compiler also supports the "OpenMP API specifications" that perform thread-level parallel processing by means of directive lines specified within the program. A shared memory system is a prerequisite for the automatic parallelization and parallel processing that use OpenMP specifications, and these functions are enabled within the compute nodes. Used in conjunction with the MPI library, these functions support the highly efficient hybrid parallel programming model (thread parallels + MPI process parallels).

   The Fortran compiler also supports a wide variety of other functions, such as program debugging (procedure reference verification, undefined data reference, array size validity), precision sensitivity diagnosis through application of precision reduction, trace-back map, error monitoring, and output of compilation information containing parallelization and optimization information.

3. Optimization functions

   The Fortran compiler is equipped with optimization functions listed below, and can create object programs capable of fast execution at compute nodes. In addition, various optimization control lines are provided in order to facilitate these optimizations from within programs.

   - Optimizations reconfiguring nested loops
   - Instruction scheduling suitable for the CPU characteristics
   - Increasing degree of parallelism by SIMD utilizing SVE
   - Reducing save and restore instruction counts for register contents
   - Efficient use of cache by means of prefetch instructions

- Optimization functions that effectively use the HPC tag address override function of A64FX processor:
    - Control of hardware prefetch and software prefetch
    - Software control of sector cache by optimization control lines

## 2.1.2 C Compiler

The C compiler compiles programs written in C and, like the Fortran compiler, can create executable programs highly optimized to achieve high CPU execution performance for targeted compute nodes. If required, the C compiler can also create executable programs capable of automatic parallelization or thread-level parallel execution using OpenMP specifications.

An overview of C compiler functions is presented below. Refer to the "C User's Guide" for C compiler details.

1. Language specifications

   The C compiler supports the following standards:

   - ISO/IEC 9899:2011 (C11 standard)
   - ISO/IEC 9899:1999 (C99 standard)
   - ISO/IEC 9899:1990 (C89 standard)
   - OpenMP API Version 5.0 a subset of specification
   - OpenMP API Version 4.5
   - OpenMP API Version 4.0

   The C compiler also supports GNU compiler extended specifications.

2. Main functions

   The C compiler is equipped with an "Automatic parallelization function" that enables programs that automatically perform thread-level parallel processing to be created just by specifying the compile options. The C compiler also supports the "OpenMP API specifications" that perform thread-level parallel processing by means of directive lines specified within the program. A shared memory system is a prerequisite for the automatic parallelization and parallel processing that use OpenMP specifications, and these functions are enabled within the compute nodes. Used in conjunction with the MPI library, these functions support the highly efficient hybrid parallel programming model (thread parallels + MPI process parallels).

   The C compiler also supports GNU compiler extended specifications and has superior portability from a variety of systems.

3. Optimization functions

   The C compiler is equipped with optimization functions listed below, and can create object programs capable of fast execution at compute nodes. In addition, various optimization control lines are provided in order to facilitate these optimizations from within programs.

   - Optimizations reconfiguring nested loops
   - Instruction scheduling suitable for the CPU characteristics
   - Increasing degree of parallelism by SIMD utilizing SVE
   - Reducing save and restore instruction counts for register contents
   - Efficient use of cache by means of prefetch instructions
   - Optimization functions that effectively use the HPC tag address override function of A64FX processor:
       - Control of hardware prefetch and software prefetch
       - Software control of sector cache by optimization control lines

## 2.1.3 C++ Compiler

The C++ compiler compiles programs written in C++ language and, like the Fortran compiler, can create executable programs highly optimized to achieve high CPU execution performance for targeted compute nodes. If required, the C++ compiler can also create executable programs capable of automatic parallelization or thread-level parallel execution using OpenMP specifications.

An overview of C++ compiler functions is presented below. Refer to the "C++ User's Guide" for C++ compiler details.

1. Language specifications

   The C++ compiler supports the following standards:

   - ISO/IEC 14882:2017 (C++17 standard)
   - ISO/IEC 14882:2014 (C++14 standard)
   - ISO/IEC 14882:2011 (C++11 standard)
   - ISO/IEC 14882:2003 (C++03 standard)
   - OpenMP API Version 5.0 a subset of specification
   - OpenMP API Version 4.5
   - OpenMP API Version 4.0

   The C++ compiler also supports GNU compiler extended specifications.

2. Main functions

   The C++ compiler is equipped with an "Automatic parallelization function" that enables programs that automatically perform thread-level parallel processing to be created just by specifying the compile options. The C++ compiler also supports the "OpenMP API specifications" that perform thread-level parallel processing by means of directive lines specified within the program. A shared memory system is a prerequisite for the automatic parallelization and parallel processing that use OpenMP specifications, and these functions are enabled within the compute nodes. Used in conjunction with the MPI library, these functions support the highly efficient hybrid parallel programming model (thread parallels + MPI process parallels).

   The C++ compiler also supports GNU compiler extended specifications and has superior portability from a variety of systems.

3. Optimization functions

   The C++ compiler is equipped with optimization functions listed below, and can create object programs capable of fast execution at compute nodes. In addition, various optimization control lines are provided in order to facilitate these optimizations from within programs.

   - Optimizations reconfiguring nested loops
   - Instruction scheduling suitable for the CPU characteristics
   - Increasing degree of parallelism by SIMD utilizing SVE
   - Reducing save and restore instruction counts for register contents
   - Efficient use of cache by means of prefetch instructions
   - Optimization functions that effectively use the HPC tag address override function of A64FX processor:
     - Control of hardware prefetch and software prefetch
     - Software control of sector cache by optimization control lines

## 2.2 Communication Library

### 2.2.1 MPI Library

The MPI library (message passing library) conforms to the MPI-3.1 standard and a subset of the MPI-4.0 standard (tentative name) prescribed by the MPI Forum. The MPI library supports the interconnect, known as Tofu, comprised of a 6-dimensional mesh/torus, and achieves high performance and conserves memory. The MPI library provides optimization suited to network topologies having from one to three dimensional torus structures.

The MPI library switches to the transfer method that best suits the transmitted data length in order to make point-to-point communication faster. In addition to transmitted data length, process allocation (rank allocation) and the number of hops are also taken into account when switching to the optimum transfer method.

The MPI library does not use point-to-point communication for collective communication functions that are used frequently. Instead, it recognizes the interconnect topology structure and adopts special-purpose algorithms that reduce congestion in order to make collective communications faster. High-speed barrier and reduction using the interconnect hardware functions are also possible.

The MPI library can be used from Fortran compiler, C compiler, C++ compiler, and Java.

Refer to the "MPI User's Guide" for information on the MPI library.

### 2.2.2 uTofu

uTofu is a low-level application programming interface (API) used by software in the user space to communicate using the Tofu interconnect. uTofu supports the one-sided communication and barrier communication of the Tofu interconnect. The interface of uTofu is provided as functions of C. uTofu implementations are provided in the form of a library.

The design of uTofu assumes that it is used from libraries or the like for communication between processes by application programs, the MPI library, and other libraries. Nonetheless, application programs can use uTofu directly. The interface can also be used for communication (memory access) within a single thread or between threads in a single process.

Refer to the "uTofu User's Guide" for uTofu details.

## 2.3 Software Development Support Tools

### 2.3.1 Profiler

The profiler is a performance analysis tool that can measure various types of information required for analyzing the performance of application programs written in Fortran, C, and C++. The profiler can also measure profiler information for programs that support thread parallels and MPI process parallels. The profiler is comprised of Instant Performance Profiler, Advanced Performance Profiler, and CPU Performance Analysis Report.

- Instant Performance Profiler

  Instant Performance Profiler helps users understand performance tendency of an application without recompilation. It is possible to measure performance information with low overhead for large-scale parallel programs.

  The output items of Instant Performance Profiler are time statistics information, CPU activity information, cost information, call graph information, and source code information.

- Advanced Performance Profiler

  Advanced Performance Profiler helps users understand the detailed performance of specific regions.

  The output items of Advanced Performance Profiler are time statistics information, MPI communication cost information, and CPU performance analysis information.

- CPU Performance Analysis Report

  CPU Performance Analysis Report aggregates CPU performance analysis information measured by Advanced Performance Profiler, and visualizes them using tables and graphs so that users can easily understand them.

Refer to the "Profiler User's Guide" for profiler details.

### 2.3.2 Debugger for Parallel Applications

The debugger for parallel applications consists of the following functions. It is available for MPI parallelized application programs (written in Fortran, C, C++).

- Abnormal Termination Investigative Function

  This function supports investigations conducted to identify the cause of an abnormal termination of an application. It obtains a backtrace or other execution information when a signal is issued due to an abnormal termination at execution time of an application.

- Deadlock Investigative Function

  This function supports investigations conducted to determine whether an application deadlock has occurred and identify the cause of the deadlock. If a program does not end or respond, the function obtains a backtrace or other execution information on all job processes without ending the program.

- Duplication Removal Function

  This function performs data manipulation for readability improvement on an investigation result file from the abnormal termination investigative function and deadlock investigative function.

- Debugging Control Function with Command Files

  This debugging control function uses files in which GDB commands are written (hereinafter called as command files). Since it is possible to specify a command file appropriate for the target process, debugging that is flexible according to the program properties can be conducted.

Refer to the "Debugger for Parallel Applications User's Guide" for debugger for parallel applications details.

### 2.3.3 IDE (Integrated Development Environment)

Eclipse, the most major and proven open source integrated development environment, is adopted as the IDE. Parallel Tools Platform, a plugin for parallel program development, works with the job scheduler to submit jobs, check job status, and so on.

IDE mainly consists of the following features.

- Editing files
- Building applications
- Monitoring jobs
- Controlling jobs
- Displaying profiler results (CPU performance analysis report)

## 2.4 Mathematical Libraries

In addition to Fujitsu's own mathematical libraries (SSL II and C-SSL II) which are widely used within Japan by R&D users, the linear algebra field libraries (BLAS, LAPACK, and ScaLAPACK) developed in the US are also provided. Fast Basic Operations Library for Quadruple Precision, which is a library in which a quadruple precision number is expressed in double-double format and arithmetic operations are performed on such formatted numbers, is also provided.

These mathematical libraries (except for ScaLAPACK) make it possible for the same routine to be called from multiple threads simultaneously (thread safe).

Important SSL II functions also provide the thread parallel routines that describe the parallel mathematical computing algorithms intended for shared memory type scalar parallel computing in OpenMP Fortran. In addition, these parallel mathematical computing algorithms are also provided as C-SSL II thread parallel routines.

Three-dimensional Fourier transforms parallelized by MPI are provided by SSL II/MPI.

Thread parallel routines are also provided for the major BLAS routines and for the major LAPACK routines.

All of these mathematical libraries are tuned so that optimized execution performance is obtained at each compute node.

Refer to the "Programmer's Guide for Usage of Mathematical Libraries" for information on Usage of Mathematical Libraries.

Refer to the "SSL II User's Guide", "FUJITSU SSL II Extended Capabilities User's Guide", "FUJITSU SSL II Extended Capabilities User's Guide II", and "Fujitsu SSL II Thread-Parallel Capabilities User's Guide" for information on SSL II.

Refer to the "Fujitsu C-SSL II User's Guide" and the "Fujitsu C-SSL II Thread-Parallel Capabilities User's Guide" for information on C-SSL II.

Refer to the "Fujitsu SSL II/MPI User's Guide" for information on SSL II/MPI.

Refer to the "BLAS LAPACK ScaLAPACK User's Guide" for information on BLAS, LAPACK and ScaLAPACK.

Refer to the "Fast Basic Operations Library for Quadruple Precision User's Guide" for information on Fast Basic Operations Library for Quadruple Precision.

# Chapter 3 Notes on Migrating from Technical Computing Language V2/V3

This chapter provides notes on migrating from Technical Computing Language V2/V3 to this software.

## Fortran, C, and C++ programs

At the source code level, Fortran, C, and C++ programs are basically forward-compatible, but there are incompatibilities at the object program and executable program level. Use this software to perform recompilation.

## Other

In addition, there have been some specification changes, such as command names, option names, and environment variable names, for component functions provided by this software.