DO NOT REDISTRIBUTE NOR DISCLOSE TO THE PUBLIC

Seminar for New Users of Supercomputer Fugaku

ver. 2025.04

Research Organization for Information Science and Technology (RIST)

- Introduction
- Access to Fugaku
- Compilers
- Mathematical Libraries
- Environment Module Packages
- Spack
- Scripting Languages
- Job Submission
- MPI Job Submissions and Precautions
- Checking the Resource Usage
- LLIO
- Advanced Usage

Purpose of the Seminar

This seminar is held by RIST for promotion of the use of Supercomputer Fugaku, as the Registered Institution and the Representative for HPCI operation.

In this “seminar for new users,” we provide information about how to use Fugaku:

- ■ How to login to Fugaku
- ■ Compilers and mathematical libraries
- ■ Job submission

Common knowledge of HPC and programming languages (e.g., Fortran and C/C++) is outside the scope of this seminar.

The stars ★/☆ at the top-left of each page indicates the level of the contents for Fugaku users.
★★★: Fundamental information that is valid for all users
★★☆: Somewhat advanced information
★☆☆: Highly advanced information that is valid in specific situations

Reference

These slides contain information extracted from the following references. For more detail, refer to the original materials.

- Supercomputer Fugaku Startup Guide Ver. 1.11
  - https://www.fugaku.r-ccs.riken.jp/doc_root/en/startup_guides/StartupGuide-en.pdf
  - You can also get this file from the URL in the email containing the client certificate.
- User's Guide - Use and Job Execution Ver. 1.48
  - https://www.fugaku.r-ccs.riken.jp/doc_root/en/user_guides/use_latest/ (*1)
- User's Guide - Language and Development Environment Ver. 1.32
  - https://www.Fugaku.r-ccs.riken.jp/doc_root/en/user_guides/lang_latest/ (*1)
- Fugaku Spack User Guide
  - https://www.fugaku.r-ccs.riken.jp/doc_root/en/user_guides/FugakuSpackGuide/
- Fugaku OpenOnDemand Guide
  - https://riken-rccs.github.io/ondemand_fugaku/index.html
- Pre/Post Environment User's Guide Ver. 1.14
  - https://www.fugaku.r-ccs.riken.jp/doc_root/en/user_guides/pps-slurm/ (*1)

*1 link to the latest version

Reference

- User’s Guide HPCI Login Manual For OAuth (2024.04.04)
  - https://www.hpci-office.jp/download_file/view/09073283-a26a-4e4c-bd6e-4dc6b4ac8d6c/2081
- HPCI Shared Storage User Manual For OAuth (2004.04.23)
  - https://www.hpci-office.jp/download_file/view/9fa79e23-98b3-4f21-a632-4faf10789442/2081
- Job Operation Software End-user's Guide
  - https://www.fugaku.r-ccs.riken.jp/doc_root/en/manuals/jos/j2ul-2534-01enz0.pdf
- Job Operation Software End-user's Guide for HPC Extensions
  - https://www.fugaku.r-ccs.riken.jp/doc_root/en/manuals/jos/j2ul-2535-01enz0.pdf
- User Support Tools User's Guide
  - https://www.fugaku.r-ccs.riken.jp/doc_root/en/user_guides/support_tool/
- Programming Guide (IO)
  - https://www.fugaku.r-ccs.riken.jp/doc_root/en/programming_guides/IO_part_Programming_Guide.pdf
- Programming Guide (Programming Common Part)
  - https://www.fugaku.r-ccs.riken.jp/doc_root/en/programming_guides/Programming_common_part_Programming_Guide.pdf

Reference

- Programming Guide (Processors)
  - https://www.fugaku.r-ccs.riken.jp/doc_root/en/programming_guides/Processors_Programming_Guide.pdf
- MPI User's Guide (tcsds-1.2.41)
  - https://www.fugaku.r-ccs.riken.jp/doc_root/ja/manuals/tcsds-1.2.41/lang/MPI/j2ul-2565-01z0.pdf
- Profiler User's Guide (tcsds-1.2.41)
  - https://www.fugaku.r-ccs.riken.jp/doc_root/ja/manuals/tcsds-1.2.41/lang/Tool/j2ul-2568-01z0.pdf
- HPCI Portal - Program Tuning Support
  - https://www.hpci-office.jp/en/user_support/tuning_support
- Documents from Fugaku User Briefing (Apr. 10th, 2025)
  - Latest information are provided on Fugaku User Briefing.
  - Operation status of Fugaku
    - https://www.fugaku.r-ccs.riken.jp/doc_root/ja/guidances/SystemReport_202504.pdf
  - Report by the help desk of the Registered Institution
    - https://www.fugaku.r-ccs.riken.jp/doc_root/ja/guidances/HelpdeskReport_202504.pdf
- Fugaku Support Site users’ guide (Fourth Edition)
  - https://www.fugaku.r-ccs.riken.jp/doc_root/en/other/FugakuSupportSite-guide.pdf

Basic Usage

The following is the basic procedure used to perform a calculation in Fugaku.

1. Upload the files to Fugaku

```
[_LNlogin]$ scp -i private_key local_file user_name@login.fugaku.r-ccs.riken.jp:remote_file
```

2. Connect to the Fugaku login node via ssh

```
[terminal]$ ssh -i private_key user_name@login.fugaku.r-ccs.riken.jp
```

3. Compile the program

```
[_LNlogin]$ frtpx -Kfast,openmp -o sample sample.f08
```

4. Submit the job

```
[_LNlogin]$ pjsub sample.sh
```

5. Display the job status

```
[_LNlogin]$ pjstat
```

6. Check the result

```
[_LNlogin]$ less ./sample.sh.jobid.out
```

■ In this material, terminals commands are executed are described as below.
`[terminal]$` means to execute the command at the user device
`[_LNlogin]$` means to execute the command at the login node (intel)
`[_CNlogin]$` means to execute the command at the compute node

- ■ Introduction
- ■ Access to Fugaku
- ■ Compilers
- ■ Mathematical Libraries
- ■ Environment Module Packages
- ■ Spack
- ■ Scripting Languages
- ■ Job Submission
- ■ MPI Job Submissions and Precautions
- ■ Checking the Resource Usage
- ■ LLIO
- ■ Advanced Usage

Account Setup

There are mainly two ways to access Fugaku:

- Using a local account of Fugaku
  - For all users of Fugaku
  - SSH connection using publickey registration
    - Reference
      - Start-up Guide for Supercomputer Fugaku
  - connection via Fugaku Open OnDemand (authentication by client certificate)
    - Open OnDemand: Web portal for utilizing supercomputer systems from web browser
    - Reference
      - Fugaku Open OnDemand Guide
- Using the HPCI account with a single sign-on
  - For users belonging to HPCI projects
  - hpcissh connection using OAuth certificate
  - Enables access to HPCI shared storage and from other HPCI resources
  - Reference
    - User’s Guide HPCI Login Manual For OAuth
    - HPCI Shared Storage User Manual For OAuth

Account Setup

The procedures to set up the local accounts of Fugaku are shown on this and the following pages.

For details, refer to the *Supercomputer Fugaku Startup Guide*.

1. Receive the client certificate and passphrase

- Client certificate: a file named "(your username).p12"
  - Sent to your e-mail address after your account has been issued
  - Used to access the Fugaku website via web browsers
  - Contains the private key, public key, and the root certificate of the certification authority

- Passphrase of the client certificate
  - Written on a postcard or in a pdf file to sent your mail or email address
  - Needed to install the client certificate on the web browsers
  - Keep it in a safe place

2. Install the client certificate on the web browser

- Recommended web browsers
  - Mozilla Firefox（Windows, Mac）
  - Google Chrome（Windows, Mac）

*1 When using other web browsers, the user is responsible for checking beforehand how to install a client certificate.

- Notes
  - Some web browsers call the passphrase of a client certificate a “password.”
  - When using Mozilla Firefox or similar web browsers, your master password may be required in addition to the passphrase.

Account Setup

3. Access the Fugaku website

URI: https://www.fugaku.r-ccs.riken.jp/en

- Note

  The Fugaku website only allows access via TLS 1.2 or TLS1.3 to avoid vulnerability issues.

- How to log out

  There is no “log out” button on the Fugaku website.

  Please close the web browser if you want to log out.

- How to switch accounts

  There is no “switch account“ option on the Fugaku website.

  You need to close the web browser and reopen it while choosing the corresponding client certificate. Alternatively, you can use different web browsers for different user accounts.

Fugaku Website

■ Home

Operation Status & Schedule

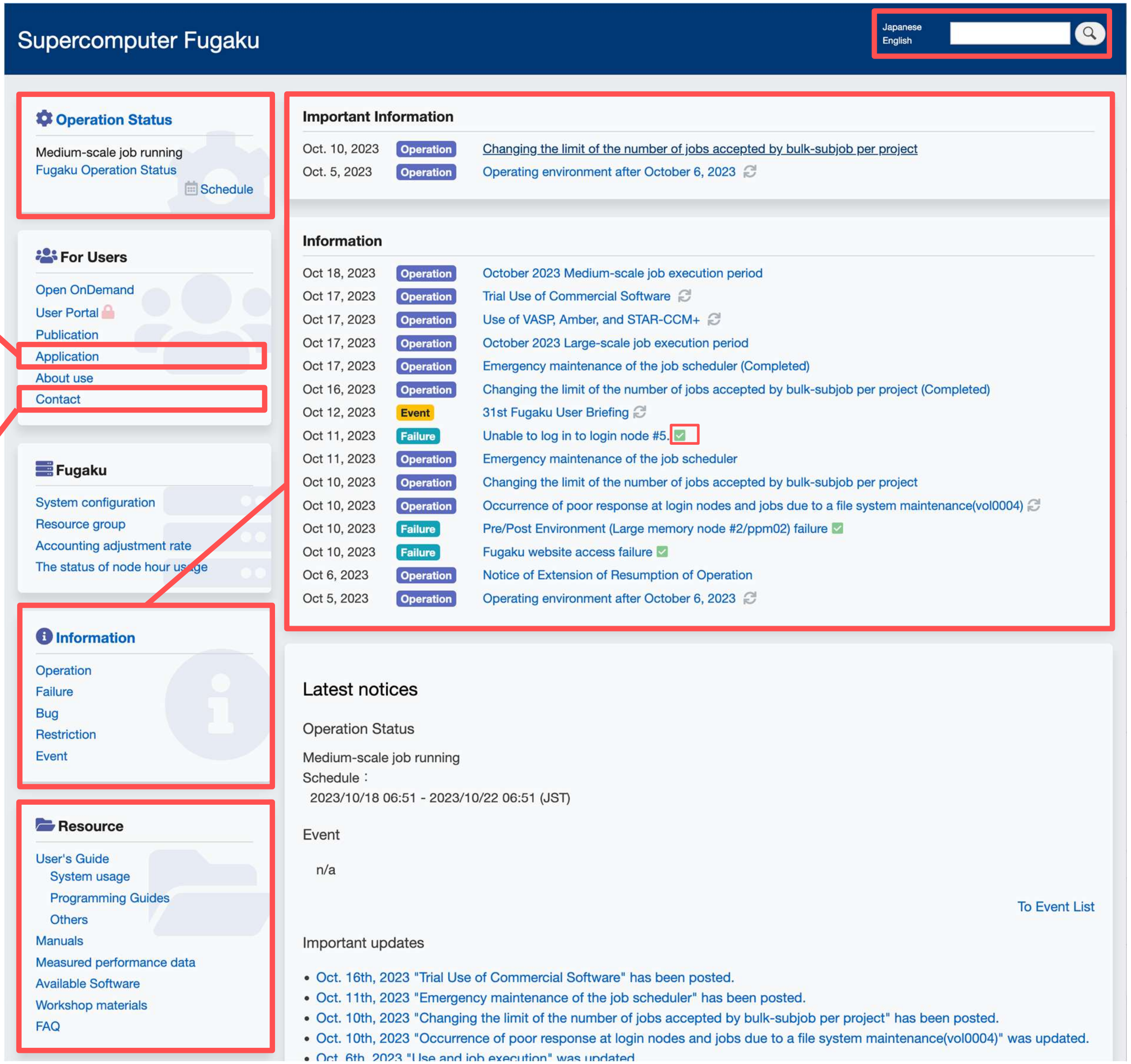
Application

e.g., expansion of the data area

Contact

a link to Fugaku Support Site

Documents



Supercomputer Fugaku

Japanese
English

Operation Status

Medium-scale job running
Fugaku Operation Status
Schedule

For Users

- Open OnDemand
- User Portal
- Publication
- Application
- About use
- Contact

Fugaku

- System configuration
- Resource group
- Accounting adjustment rate
- The status of node hour usage

Information

- Operation
- Failure
- Bug
- Restriction
- Event

Resource

- User's Guide
  - System usage
  - Programming Guides
  - Others
- Manuals
- Measured performance data
- Available Software
- Workshop materials
- FAQ

Important Information

Oct. 10, 2023	Operation	Changing the limit of the number of jobs accepted by bulk-subjob per project
Oct. 5, 2023	Operation	Operating environment after October 6, 2023 🔄

Information

Oct 18, 2023	Operation	October 2023 Medium-scale job execution period
Oct 17, 2023	Operation	Trial Use of Commercial Software 🔄
Oct 17, 2023	Operation	Use of VASP, Amber, and STAR-CCM+ 🔄
Oct 17, 2023	Operation	October 2023 Large-scale job execution period
Oct 17, 2023	Operation	Emergency maintenance of the job scheduler (Completed)
Oct 16, 2023	Operation	Changing the limit of the number of jobs accepted by bulk-subjob per project (Completed)
Oct 12, 2023	Event	31st Fugaku User Briefing 🔄
Oct 11, 2023	Failure	Unable to log in to login node #5. ✅
Oct 11, 2023	Operation	Emergency maintenance of the job scheduler
Oct 10, 2023	Operation	Changing the limit of the number of jobs accepted by bulk-subjob per project
Oct 10, 2023	Operation	Occurrence of poor response at login nodes and jobs due to a file system maintenance(vol0004) 🔄
Oct 10, 2023	Failure	Pre/Post Environment (Large memory node #2/ppm02) failure ✅
Oct 10, 2023	Failure	Fugaku website access failure ✅
Oct 6, 2023	Operation	Notice of Extension of Resumption of Operation
Oct 5, 2023	Operation	Operating environment after October 6, 2023 🔄

Latest notices

Operation Status

Medium-scale job running
Schedule：
2023/10/18 06:51 - 2023/10/22 06:51 (JST)

Event

n/a

To Event List

Important updates

- Oct. 16th, 2023 "Trial Use of Commercial Software" has been posted.
- Oct. 11th, 2023 "Emergency maintenance of the job scheduler" has been posted.
- Oct. 10th, 2023 "Changing the limit of the number of jobs accepted by bulk-subjob per project" has been posted.
- Oct. 10th, 2023 "Occurrence of poor response at login nodes and jobs due to a file system maintenance(vol0004)" was updated.
- Oct. 6th, 2023 "Use and job execution" was updated.

Site Search

Information

- Extract information of specific categories

Operation

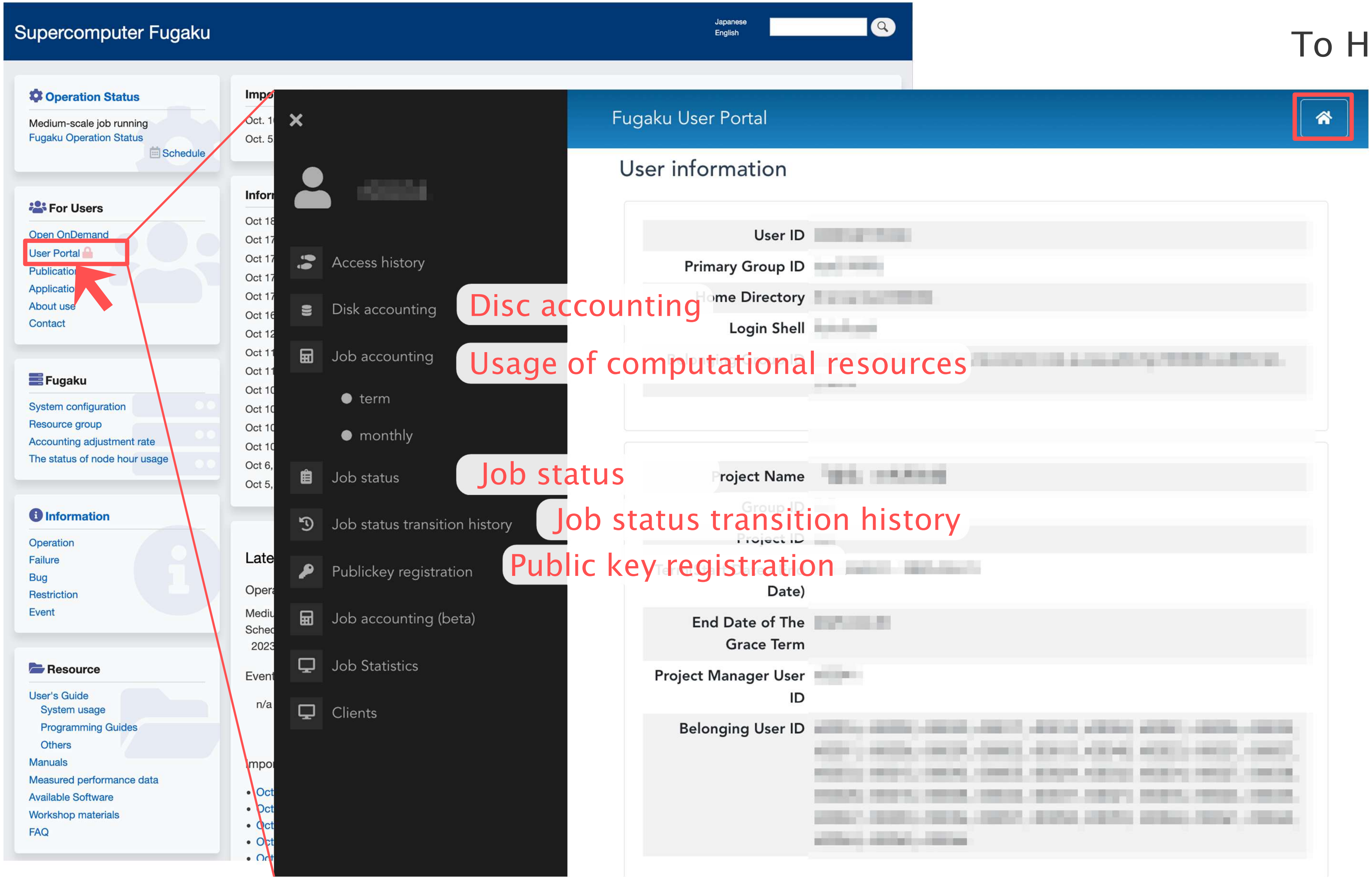
- Icon definitions

🔄 Updated

✅ Fixed

Fugaku Website

User portal



To Home

Disc accounting

Usage of computational resources

Job status

Job status transition history

Public key registration

How to Log in to Fugaku

SSH Version 2 with public key authentication is required to log in to Fugaku via your local account.

Key pair generation and public key registration are performed as follows.

1. Generate key pairs (secret and public keys)

   Accepted key types are shown below.

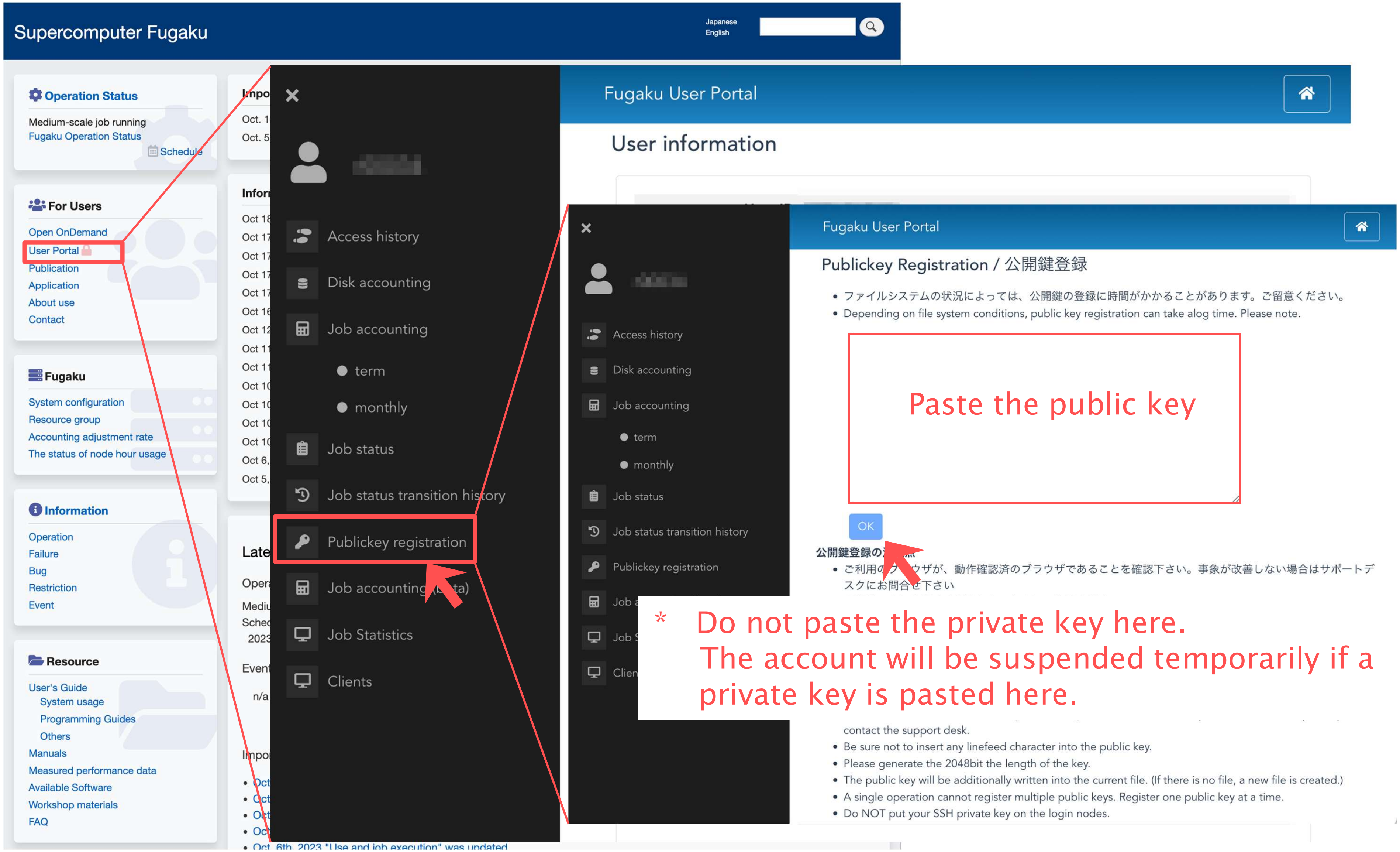
   Details of the generation procedures are given in *User's Guide - Use and Job Execution*, Section 4.4.1.

   - ■ Ed25519
   - ■ ECDSA (NIST P 521)

- It is strongly recommended to use passphrases (with more than 15 characters and difficult to predict) when generating key pairs.
- To use RSA keys, check the announcement below.
  (https://www.fugaku.r-ccs.riken.jp/en/operation/20230210_01)
- Open OnDemand is available as an alternative way to use Fugaku without public key authentication.

How to Log in to Fugaku

2. Public key registration



Publickey Registration / 公開鍵登録

- ファイルシステムの状況によっては、公開鍵の登録に時間がかかることがあります。ご留意ください。
- Depending on file system conditions, public key registration can take alog time. Please note.

Paste the public key

公開鍵登録の注意点

- ご利用のブラウザが、動作確認済のブラウザであることを確認下さい。事象が改善しない場合はサポートデスクにお問合せ下さい

* Do not paste the private key here.
The account will be suspended temporarily if a private key is pasted here.

contact the support desk.
- Be sure not to insert any linefeed character into the public key.
- Please generate the 2048bit the length of the key.
- The public key will be additionally written into the current file. (If there is no file, a new file is created.)
- A single operation cannot register multiple public keys. Register one public key at a time.
- Do NOT put your SSH private key on the login nodes.

How to Log in to Fugaku

3. Access via SSH2

Host name: `login.fugaku.r-ccs.riken.jp`

```
[terminal]$ ssh -i private_key user_name@login.fugaku.r-ccs.riken.jp
The authenticity of host 'XXXXXX (nnn.nnn.nnn.nnn)' can't be established.
XXXXX key fingerprint is XX: XX: XX: XX: XX: XX: XX: XX:XX:XX:XX:XX:XX:XX:XX:XX.
Are you sure you want to continue connecting (yes/no)? yes
Enter passphrase for key ‘private_key':
[_LNlogin]$
```

- Notes
  - Add option –X when using X11 forwarding for SSH.
  - Add option –A when using SSH agent forwarding.
  - Home area (/home), data area (/data), share area (/share), and 2ndfs area (/2ndfs) are the same for every login node.
  - The programing environments are the same for every login node.
  - The data size of the home area for each user is limited to 20 GiB, and the number of i-nodes is limited to 200,000.
    - Up to 750,000 i-nodes can be used for one week. After that, new file creation is disabled by the system. Therefore, you need to delete files ASAP.
    - The login shell is /bin/bash.

Refer to *User's Guide - Use and Job Execution, Section 4.4.3* for details about access via PuTTY and similar applications.

How to Log in to Fugaku

4. Transfer files with scp/sftp/rsync

Note that vulnerable protocols (ftp/rcp) are prohibited.

- Transfer from local environments to login nodes

```
[terminal]$ scp -i private_key local_file user_name@login.fugaku.r-ccs.riken.jp:remote_file
Enter passphrase for key '/home/user_name/.ssh/id_ed25519’:
[terminal]$
```

- Transfer from login nodes to local environments

```
[terminal]$ scp -i private_key user_name@login.fugaku.r-ccs.riken.jp:remote_file local_file
Enter passphrase for key '/home/user_name/.ssh/id_ed25519’:
[terminal]$
```

Refer *to User's Guide - Use and Job Execution,* Section 4.4.4 for details about how to transfer files via WinSCP.

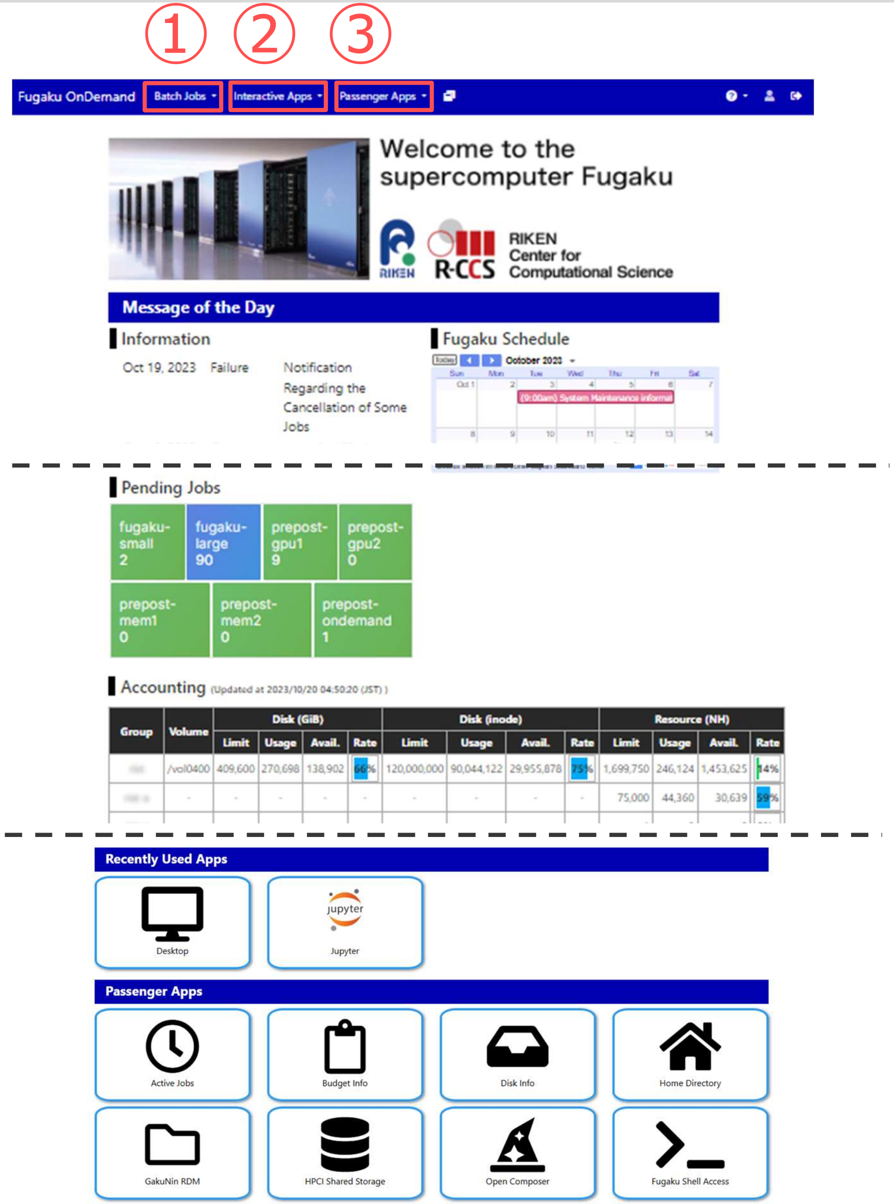
Fugaku Open OnDemand

Fugaku Open OnDemand is a web portal for using Fugaku applications via a web browser.

- What is Open OnDemand?
  - A web portal for using parallel computer systems via a web browser
  - Official website https://openondemand.org/
- URL
  - https://ondemand.fugaku.r-ccs.riken.jp/pun/sys/dashboard
    - The client certificate for Fugaku website is also necessary to access Fugaku OpenOndemand.
    - On Fugaku website, there is a link to Fugaku OpenOnDemand on the left menu.
- Merit to use
  - Public key authentication is not needed.
  - Various applications for scientific calculation and pre/post processing can be used without writing job scripts.
  - It is also possible to view and edit files on Fugaku directly using a web browser.

Fugaku Open OnDemand

- Available apps
  ## ① Batch Jobs
  - Non-interactive apps that run on compute nodes
  - Various scientific calculation apps are available
  - see p120 for the use of Open Composer
  ## ② Interactive Apps
  - Interactive apps that run on pre-post nodes and compute nodes
  - Applications for development and viewers are available
  ## ③ Passenger Apps
  - Utilities that run on the server where Open OnDemand is installed
  - Text editor, shell and file operations are available



Message of the Day

Information

Oct 19, 2023 Failure Notification Regarding the Cancellation of Some Jobs

Fugaku Schedule

Pending Jobs

fugaku-small	fugaku-large	prepost-gpu1	prepost-gpu2
2	90	9	0

prepost-mem1	prepost-mem2	prepost-ondemand
0	0	1

Accounting (Updated at 2023/10/20 04:50:20 (JST))

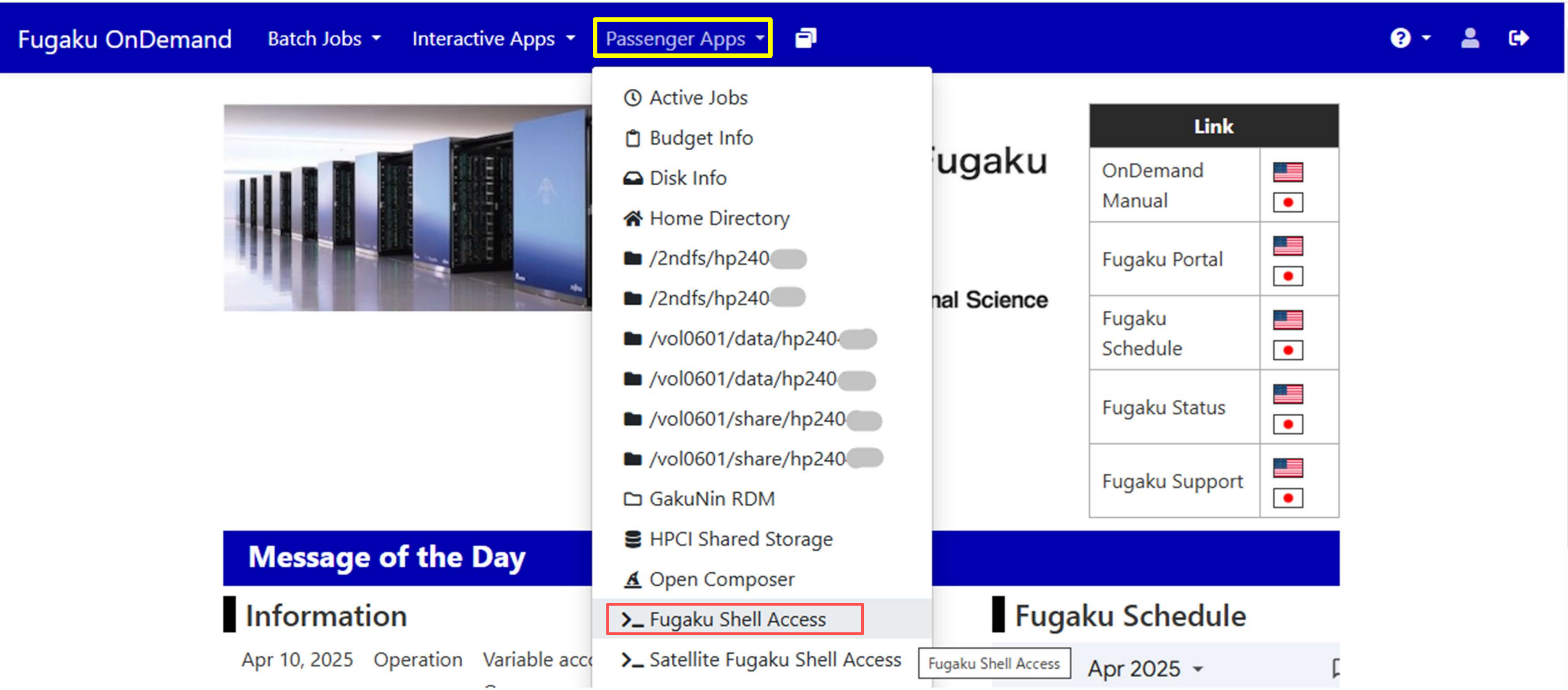
Group	Volume	Disk (GiB)				Disk (inode)				Resource (NH)			
		Limit	Usage	Avail.	Rate	Limit	Usage	Avail.	Rate	Limit	Usage	Avail.	Rate
[redacted]	/vol0400	409,600	270,698	138,902	66%	120,000,000	90,044,122	29,955,878	75%	1,699,750	246,124	1,453,625	14%
[redacted]	-	-	-	-	-	-	-	-	-	75,000	44,360	30,639	59%

Recently Used Apps

Passenger Apps

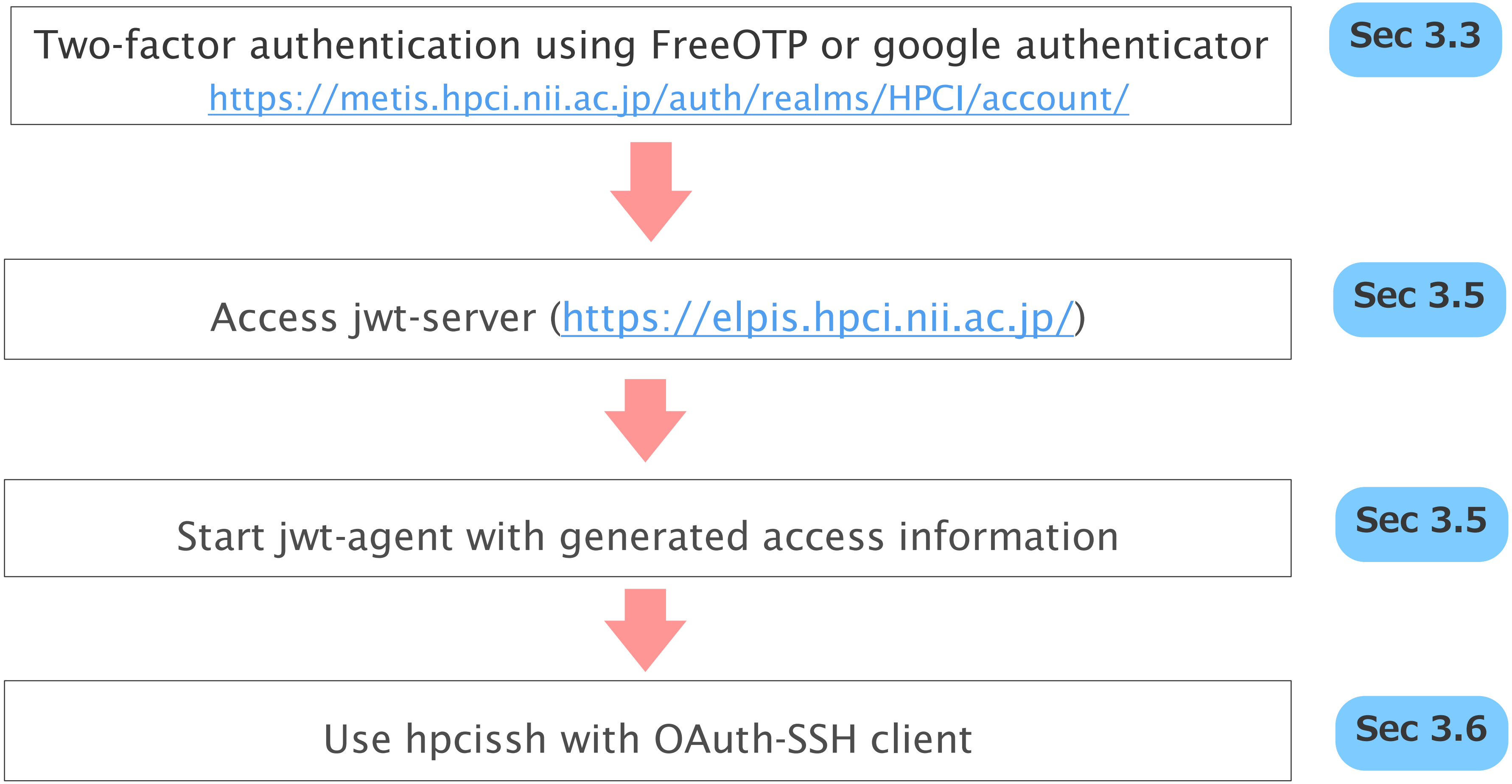
Fugaku Open OnDemand

Fugaku login nodes can be accessed from Fugaku Open OnDemand.



Refer to *Fugaku Open OnDemand Guide* for details

Access via an HPCI Account



Two-factor authentication using FreeOTP or google authenticator
https://metis.hpci.nii.ac.jp/auth/realms/HPCI/account/

Sec 3.3

Access jwt-server (https://elpis.hpci.nii.ac.jp/)

Sec 3.5

Start jwt-agent with generated access information

Sec 3.5

Use hpcissh with OAuth-SSH client

Sec 3.6

- Refer to the *User’s Guide HPCI Login Manual For OAuth* for more details.

Available Servers

The servers available in Fugaku are shown below.

- Intel login nodes
  - Accessible from external networks
  - Host name: `login.fugaku.r-ccs.riken.jp`
- Arm login nodes
  - Accessible from the Intel login nodes
  - Host name: `arm1`
- Pre-/post-processing nodes
  - Accessible from the Intel login nodes via Slurm
  - Reference: “Pre/Post Environment Users Guide”
- Cloud storage gateway nodes
  - Accessible from external networks
  - Nodes for data transfer
  - HPCI shared storage can be mounted here
  - Host name: `csgw.fugaku.r-ccs.riken.jp`

HPCI Shared Storage

It is possible to mount HPCI shared storage on Fugaku if you have the permissions to use it.

HPCI shared storage

- A large-scale data sharing platform with Gfarm, the distributed file system for large-scale cluster computing and wide-area data sharing
- Total logical size: 45.0 PB

Using the shared storage

1. Log in to a cloud storage gateway node using hpcissh
2. Use the `mount.hpci` command to mount the HPCI shared storage
3. Transfer data in parallel using copy commands like `gfpcopy`

Reference:

- *HPCI Shared Storage User Manual For OAuth*

Mail Notification of Operational Info.

Users can subscribe to be notified by mail about Fugaku operational information.

- [Fugaku] Reports the jobs affected by the failure
- [Fugaku] LLIO Usage Limit Exceeded Notification
- [Fugaku] Information

- How to subscribe
  - Create .forward file in your home directory as follows.
  - Multiple addresses can be specified by separating them with breaks or commas.

```
[_LNlogin]$ vi ~/.forward

*****@*****.com

*****@******.jp
```

- Refer to https://www.fugaku.r-ccs.riken.jp/faq/20220324_01 for details.

Precautions for use of login nodes

The login nodes (login 1 to 6) are shared by many Fugaku users. To avoid slowdown of a login node, observe the following items.

- Don’t run a program consuming a large amount of memory on a login node
- Don’t execute a program with many processes or threads in a login node
  - Use the pre/post environment in executing programs requiring a large amount of memory
  - Pay attention not to exceed the following criteria

	upper limit per one user
the number of threads	8 threads
the amount of memory	12 GB

- A process exceeding the limits will be terminated by the system, and the information is notified to the corresponding user.

- Introduction
- Access to Fugaku
- Compilers
- Mathematical Libraries
- Environment Module Packages
- Spack
- Scripting Languages
- Job Submission
- MPI Job Submissions and Precautions
- Checking the Resource Usage
- LLIO
- Advanced Usage

List of Available Compilers

The available compilers are listed below.

Compiler	Login node (LN) Intel	Login node (LN) Arm	Compute node (CN)
Fujitsu	Cross compiler	-	Native compiler
GCC	Cross compiler	CentOS default	Native compiler
Clang/LLVM	Cross compiler	(OSS)	Native compiler
Intel	Intel oneAPI Base	-	-
Arm	-	Arm Compiler for Linux	-
Java	(OSS)	-	(OSS)

- Cross compilers
  - Executed at the LN and generate executable files for the CN
- Native compilers
  - Executed at the CN and generate executable files for the CN
- (OSS) and Arm compilers are deployed but not supported

Fujitsu Compiler

The Fujitsu compiler provides commands for Fortran, C, and C++.

- Compile commands

Language	Commands		Compilation modes	Automatic parallelization	OpenMP option
	Cross	Native			
Fortran	frtpx	frt	-	-Kparallel	-Kopenmp
C	fccpx	fcc	trad	-Kparallel	-Kopenmp
			clang	-	-fopenmp
C++	FCCpx	FCC	trad	-Kparallel	-Kopenmp
			clang	-	-fopenmp

- The automatic (thread) parallelization and OpenMP options are deactivated by default.
- When including objects created with automatic parallel and/or OpenMP, specify identical options if linking is used.
- The automatic parallelization and OpenMP options can be specified simultaneously.

Fujitsu Compiler

- Compile commands for MPI programs

Lang-uage	Compile commands		Header or module files for MPI libraries
	Cross	Native	
Fortran	mpifrtpx	mpifrt	mpif.h / mpi.mod mpif-ext.h / mpi_ext.mod mpi_f08.mod mpi_f08_ext.mod
C	mpifccpx	mpifcc	mpi.h mpi-ext.h
C++	mpiFCCpx	mpiFCC	

- Users can compile and link without being aware of the location of the MPI header files and libraries.
- For separate compilation, use the same command to link objects created by the compile command “mpi***”. Do not use “***” without “mpi” to link them.

Fujitsu Compiler

How to compile

If not using MPI libraries

```
[_LNlogin]$ frtpx [compile options] source_file
```

- With OpenMP (thread) parallelization

```
[_LNlogin]$ frtpx -Kfast,openmp sample.f08
```

If using MPI libraries

```
[_LNlogin]$ mpifrtpx [compile options] source_file
```

- With OpenMP parallelization, which leads to hybrid parallelization

```
[_LNlogin]$ mpifrtpx -Kfast,openmp sample.f08
```

Fujitsu Compiler

■ Basic compile options for Fortran

Compile option	Description
`-c`	Creates an object file (not an executable file)
`-o` *exe_file*	Changes the executable file name/object file name to *exe_file*
`-O[0|1|2|3]`	Specifies the optimization level ■ If the number after `-O` is omitted, it will be set to `-O3` ■ The default is `-O2`
`-Kfast`	Induces a set of optimization options to increase the performance in the CPU of Fugaku (A64FX)
`-Ksimd[=1|2|auto]`	Generates objects using SIMD extension instructions ■ Deactivated if the optimization level is under -O2 ■ If an optimization level is `-O2` or `-O3`, the `-Ksimd=auto` option is used unless otherwise specified
`-Kparallel`	Proceeds with automatic (thread) parallelization ■ The default is `-Knoparallel` ■ The `-Kparallel` option is incompatible with the optimization level of `-O0` or `-O1`
`-Kopenmp`	Enables the OpenMP Fortran specification directive ■ The default is `-Knoopenmp`

* Refer to the *Fortran User's Guide*, Section 2.2 "Compiler Options," for details.

Fujitsu Compiler

- Recommended compile options for Fortran
  - Performance Focused

```
-Kfast,openmp[,parallel]
```

- -Kfast
  - Specify this option to use high-performance features in A64FX* system, e.g., to make full use of the processor core with SVE using auto-vectorization, to improve instruction-level parallelism by software pipelining, to change the order of operations, and to use reciprocal approximation for division and square-root functions.

  * A64FX: CPU of Fugaku

- -Kparallel
  - Enables automatic parallelization on shared memory (i.e., thread parallelization).

  - Precision Focused

```
-Kfast,openmp[,parallel],fp_precision
```

- -Kfp_precision
  - Supresses various optimizations that affect numerical precision in floating-point operations.

- Refer to the *Fortran User’s Guide* for details.

Fujitsu Compiler

Compiler modes for the C and C++ languages

Mode	Compile option	Description
trad	`-Nnoclang` or when omitted	Based on the Fujitsu compiler for systems prior to K and PRIMEHPC FX100
clang	`-Nclang`	Based on the Clang/LLVM compiler, which is open source software

Pros and cons

For both C and C++

	trad	clang
Applicability to K and FX100 code	○	×
Tuning for HPC	○	○ (with additional options)
Ease of code migration (e.g., GNU compatibility)	×	○
Supports C++17	○ (partially)	○
Supports ACLE and FP16	×	○

Fujitsu Compiler

- How to compile C programs in trad mode
  - If not using MPI libraries

```
[_LNlogin]$ fccpx [compile options] source_file
```

- Example with OpenMP (thread) parallelization

```
[_LNlogin]$ fccpx -Kfast,openmp sample.c
```

- If using MPI libraries

```
[_LNlogin]$ mpifccpx [compile options] source_file
```

- With OpenMP parallelization, which leads to hybrid parallelization

```
[_LNlogin]$ mpifccpx -Kfast,openmp sample.c
```

■ Basic compilation options of trad mode for C/C++

Compile options	Description
-c	Creates an object file (not an executable file)
-o *exe_file*	Changes the executable file name/object file name to *exe_file*
-O[0|1|2|3]	Specifies optimization level ■ If the number after -O is omitted, it will be set to -O2. Note that this feature is different from that for Fortran ■ The default is -O2
-Kfast	Induces a set of optimization options leading to high performance in the CPU of Fugaku (A64FX)
-Ksimd[=1|2|auto]	Generates objects using SIMD extension instructions. ■ Deactivated if the optimization level is under -O2 ■ If an optimization level is -O2 or -O3, the -Ksimd=auto option is used unless otherwise specified
-Kparallel	Proceeds automatic (thread) parallelization ■ The default is -Knoparallel ■ -Kparallel option option is incompatible with the optimization level of -O0 or -O1
-Kopenmp	Enables OpenMP C specification directive ■ The default is -Knoopenmp

* Refer to the *C User's Guide*, Section 2.2 “Compiler Options” for details.

Fujitsu Compiler

- Recommended compile options for C and C++
  - Performance Focused

```
-Kfast,openmp[,parallel]
```

- -Kfast
  - Specify this option to use high-performance features in A64FX* system, e.g., to make full use of the processor core with SVE using auto-vectorization, to improve instruction-level parallelism by software pipelining, to change the order of operations, and to use reciprocal approximation for division and square-root functions.

  * A64FX: CPU of Fugaku

- -Kparallel
  - Enables automatic parallelization on shared memory (i.e., thread parallelization).

- Precision Focused

```
-Kfast,openmp[,parallel],fp_precision
```

- -Kfp_precision
  - Supresses various optimizations that affect numerical precision in floating-point operations.

- Refer to the *C User’s Guide* for details.

Fujitsu Compiler

- How to compile C programs in clang mode
  - If not using MPI libraries

```
[_LNlogin]$ fccpx [compile options including -Nclang] source file
```

- Example of compile commands specifying OpenMP parallelization

```
[_LNlogin]$ fccpx –Nclang –Ofast -fopenmp sample.c
```

- If using MPI libraries

```
[_LNlogin]$ mpifccpx [compile options including -Nclang] source file
```

- Example of compile commands specifying OpenMP parallelization

```
[_LNlogin]$ mpifccpx –Nclang –Ofast -fopenmp sample.c
```

Fujitsu Compiler

■ Basic compile options of clang mode for C/C++

Compile options	Description
`-c`	Creates an object file (not an executable file)
`-o` *exe_file*	Changes the executable file name/object file name to *exe_file*
`-O[0|1|2|3|fast]`	Specifies optimization level. ■ If the number after `-O` is omitted, it will be set to `-O2` ■ The default is `-O2` ■ `-Ofast` corresponds to `-Kfast` in trad mode
`-fvectorize`	Generates objects using SIMD extension instructions ■ corresponds to `-Ksimd` in trad mode
`-fopenmp`	Enables OpenMP C directive specification ■ The default is `-fnoopenmp`

Differences from trad mode

* No automatic parallelization option in clang mode.

* Refer to the *C User's Guide*, Section 9.1.2 "Compiler Options" for details.

* The options absent in this guide are not guaranteed in Fugaku.

* Most Clang/LLVM options are supported.

Fujitsu Compiler

Recommended compile options for C/C++ clang mode

```
-Ofast
```

- -Ofast
  - Specify this option to use high-performance features in A64FX* system, e.g., to make full use of the processor core with SVE using auto-vectorization, to change the order of operations, and to use reciprocal approximation for division and square-root functions.

* A64FX: CPU of Fugaku

* Refer to the *C User’s Guide* for details.

- Optimization may affect the calculation result.
  - For details, see the *C User’s Guide*, Chapter 9 “Clang Mode – Floating-point arithmetic optimization and its side effects.”

Debugger

The Fujitsu compiler provides built-in debugging functions and debugger for parallel applications functions.

- Built-in debugging function
  - Use by specifying compile options for debugging.
  - The compiled programs take longer time to execute.

- Inspection items
  - Citation checking of undefined data
    - Fortran : -Hu
  - Check array size and the validity of the index range
    - Fortran : -Hs
    - C/C++ （trad mode） : -Nquickdbg=subchk
    - C/C++ （clang mode） : -fsanitize=undefined

For more detail, refer to the “Built-in debug function” section for each compiler (Fortran, C/C++ (trad mode), C/C++ (clang mode)) in *Supercomputer Fugaku User’s Guide - Language and Development Environment.*

Debugger

- MPI options for GDB (GNU debugger) is available for code inspection and debugger control of parallel applications.
  - To obtain the detailed information of argument variables, local variables etc., –g option is recommended at the compilation.

  - Inspection function
    - Abnormal termination inspection function
      - Acquires execution information such as the backtrace when a signal is received due to abnormal termination of the program.

```
$ mpiexec -fjdbg-sig signal
```

    - Deadlock inspection function
      - If the program does not end or does not respond, execution information such as backtrace is collected without ending the program for all processes of the job.

```
$ mpiexec -fjdbg-dlock
```

Refer to Section 3.1.10. “Debugger for Parallel Applications Function” in *Supercomputer Fugaku User’s Guide - Language and Development Environment* for details.

Debugger

- MPI options for GDB (GNU debugger) is available for code inspection and debugger control of parallel applications.

  - Duplication removal function
    - Data processing is performed on the investigation result file to improve readability

```
$ fjdbg_summary
```

  - Debugging control function with command files
    - Provides a debugger control function to perform different debugger control for each process
    - This function is not used with other inspection functions

```
$ mpiexec -gdbx "[ rank-no: ] command-file [ ;... ]"
```

Refer to Section 3.1.10. “Debugger for Parallel Applications Function” in *Supercomputer Fugaku User’s Guide - Language and Development Environment* for details.

- Introduction
- Access to Fugaku
- Compilers
- **Mathematical Libraries**
- Environment Module Packages
- Spack
- Scripting Languages
- Job Submission
- MPI Job Submissions and Precautions
- Checking the Resource Usage
- LLIO
- Advanced Usage

Overview

In Fugaku, the following mathematical libraries are available.

- SSL II
- C-SSL II
- BLAS
  - Linear algebra operations such as the dot product, matrix-vector multiplication, and matrix-matrix multiplication
  - http://www.netlib.org/blas/
- LAPACK
  - Functions for solving systems of linear equations and linear least squares problems, eigenvalue problems, and singular value decomposition
  - http://www.netlib.org/lapack/
- ScaLAPACK
  - LAPACK routines redesigned for MPI
  - http://www.netlib.org/scalapack/
- High-speed quadruple-precision basic arithmetic library

Overview

- Using BLAS, LAPACK, and ScaLAPACK in Fugaku
  - Available in Fortran, C, and C++
  - Specify the corresponding compile options
  - Static and dynamic link versions are available

- Static link version
  - Creates executable files including the library data
  - May increase memory consumption and decrease memory addressing time

- Dynamic link version
  - Create executable files without including the library data
  - May decrease memory consumption and increase memory addressing time

Static Link Version

In this and the following slides, examples of compiling programs using static-link versions of the BLAS, LAPACK, and ScaLAPACK libraries are shown.

- Example of Fortran compilation
  - Sequential code with BLAS and LAPACK (sequential)
    - -SSL2: option for the sequential versions of BLAS and LAPACK

```
[_LNlogin]$ frtpx –Kfast sample.f90 -SSL2
```

- OpenMP code with BLAS and LAPACK (thread parallel)
  - -SSL2BLAMP: option for the thread parallel version of BLAS and LAPACK
  - Thread parallelization options –Kopenmp and/or –Kparallel are required

```
[_LNlogin]$ frtpx -Kfast,openmp sample.f90 -SSL2BLAMP
```

- MPI code with ScaLAPACK, BLAS, and LAPACK (sequential)
  - -SCALAPACK: option for ScaLAPACK
  - MPI compile command mpifrtpx is required

```
[_LNlogin]$ mpifrtpx –Kfast sample.f90 -SCALAPACK -SSL2
```

Static Link Version

- Example of C compilation
  - Sequential code with BLAS and LAPACK (sequential)
    - -SSL2: option for the sequential versions of BLAS and LAPACK

```
[_LNlogin]$ fccpx –Kfast sample.c -SSL2
```

  - OpenMP code with BLAS and LAPACK (thread parallel)
    - -SSL2BLAMP: option for the thread parallel version of BLAS and LAPACK
    - Thread parallelization options –Kopenmp and/or –Kparallel are required

```
[_LNlogin]$ fccpx -Kfast,openmp sample.c -SSL2BLAMP
```

  - MPI code with ScaLAPACK, BLAS, and LAPACK (sequential)
    - -SCALAPACK: option for ScaLAPACK
    - MPI compile command mpifccpx is required

```
[_LNlogin]$ mpifccpx –Kfast sample.c -SCALAPACK -SSL2
```

Static Link Version

- Example of C++ compilation
  - Sequential code with BLAS and LAPACK (sequential)
    - -SSL2: option for the sequential versions of BLAS and LAPACK

```
[_LNlogin]$ FCCpx -Kfast sample.cc -SSL2
```

  - OpenMP code with BLAS and LAPACK (thread parallel)
    - -SSL2BLAMP: option for the thread parallel version of BLAS and LAPACK
    - Thread parallelization options `–Kopenmp` and/or `–Kparallel` are required

```
[_LNlogin]$ FCCpx -Kfast,openmp sample.cc -SSL2BLAMP
```

  - MPI code with ScaLAPACK, BLAS, and LAPACK (sequential)
    - -SCALAPACK: option for ScaLAPACK
    - MPI compile command `mpiFCCpx` is required

```
[_LNlogin]$ mpiFCCpx -Kfast sample.cc -SCALAPACK -SSL2
```

Static Link Version

- Notes
  - Put library related commands such as `-SSL2` after the source file name.
  - Because a single library file includes SSL II, BLAS, and LAPACK, they all can be linked using the options `–SSL2` or `–SSL2BLAMP`.
  - SVE (scalable vector extension)
    - SVE is a set of vector instructions that can be applied to vector registers with various lengths.
    - The linked library is switched depending on whether `-KSVE` or `–KNOSVE` is specified.
    - The default is `-KSVE`, which links the library built using SVE.
    - When `–KNOSVE` is specified, the library built without SVE is linked.

Dynamic Link Version

Compile options for the dynamic link versions of BLAS, LAPACK, and ScaLAPACK are shown on this and following slides.

Examples are given afterwards.

- Compile options for Fortran
  - Specify `–Kopenmp` and/or `-Kparallel` to use the thread parallel version.

Type	BLAS or LAPACK (sequential)	BLAS or LAPACK (thread parallel)	ScaLAPACK
LP64, non-SVE	`-lfjlapack`	`-lfjlapackex`	`-lfjscalapack`
LP64、SVE	`-lfjlapacksve`	`-lfjlapackexsve`	`-lfjscalapacksve`
ILP64, non-SVE	`-lfjlapack_ilp64`	`-lfjlapackex_ilp64`	-
ILP64, SVE	`-lfjlapacksve_ilp64`	`-lfjlapackexsve_ilp64`	-

Dynamic Link Version

- Compile options for C/C++
  - Specify –Kopenmp and/or -Kparallel to use the thread parallel version.
  - Differences from Fortran
    - -SSL2 or -SSL2BLAMP options are additionally required
    - –SCALAPACK option is additionally required for ScaLAPACK
    - -I${FJSVXTCLANGA}/include/lapack_ilp64 is required for ILP64

Type	BLAS or LAPACK (sequential)	BLAS or LAPACK (thread parallel)	ScaLAPACK
LP64, non-SVE	-lfjlapack	-lfjlapackex	-lfjscalapack
LP64、SVE	-lfjlapacksve	-lfjlapackexsve	-lfjscalapacksve
ILP64, non-SVE	-lfjlapack_ilp64	-lfjlapackex_ilp64	-
ILP64, SVE	-lfjlapacksve_ilp64	-lfjlapackexsve_ilp64	-

Dynamic Link Version

- Example of Fortran compilation
  - BLAS/LAPACK (sequential) and no SVE

```
[_LNlogin]$ frtpx -Kfast sample.f -lfjlapack
```

  - BLAS/LAPACK (thread parallel) and SVE

```
[_LNlogin]$ frtpx -Kfast,openmp sample.f -lfjlapackexsve
```

  - Sequential program, linking BLAS/LAPACK (thread parallel) and SVE

```
[_LNlogin]$ frtpx -Kfast -c sample.f
[_LNlogin]$ frtpx -Kfast,openmp sample.o -lfjlapackexsve
```

  - ScaLAPACK SVE + BLAS/LAPACK (thread parallel) and SVE

```
[_LNlogin]$ mpifrtpx -Kfast,openmp sample.f -lfjscalapacksve -lfjlapackexsve
```

Dynamic Link Version

- Example of C/C++ compilation
  - C program, BLAS/LAPACK (sequential) and no SVE

```
[_LNlogin]$ fccpx -Kfast sample.c -lfjlapack -SSL2
```

  - C++ program, BLAS/LAPACK (thread parallel) and SVE

```
[_LNlogin]$ FCCpx -Kfast,openmp sample.cpp -lfjlapackexsve -SSL2
```

  - sequential C program, linking BLAS/LAPACK (thread parallel) and SVE

```
[_LNlogin]$ fccpx -c -Kfast sample.c
[_LNlogin]$ fccpx -Kfast,openmp sample.o -lfjlapackexsve -SSL2
```

  - C++ program, ScaLAPACK SVE + BLAS/LAPACK (thread parallel) and SVE

```
[_LNlogin]$ mpiFCCpx -Kfast,openmp sample.cpp -lfjscalapacksve -lfjlapackexsve -SSL2 -SCALAPACK
```

- Introduction
- Access to Fugaku
- Compilers
- Mathematical Libraries
- Environment Module Packages
- Spack
- Scripting Languages
- Job Submission
- MPI Job Submissions and Precautions
- Checking the Resource Usage
- LLIO
- Advanced Usage

Overview

On Fugaku, the Environment Modules package is available.

Environment variables can easily be set using the `module` commands provided with Environment Module package.

- Official website
  - http://modules.sourceforge.net
- Examples of environment variables that can be set using the module commands
  - PATH
    - Variable to specify the location of executable files
  - MANPATH
    - Variable to specify the locations to search for reference manual pages when using the `man` command
  - LD_LIBRARY_PATH
    - Variable to specify the paths searched by the linker when the `-l` option in a compiler is set
    - When `-lhoge` is set, the file `libhoge.a` or `libhoge.so` is referenced at the location indicated by LD_LIBRARY_PATH

Overview

- Modulefile
  - A file containing the environment variables required to use the application. It is interpreted by the module commands.

- List of major module commands (details are on the following slides)

Command	Explanation
`module avail`	Views the list of available modulefiles
`module list`	Views the list of loaded modulefiles
`module load` *`modulefile`*	Loads a modulefile
`module unload` *`modulefile`*	Unloads a modulefile
`module purge`	Unloads all modulefiles
`module switch [`*`modulefile1`*`]` *`modulefile2`*	Changes modulefile
`module show` *`modulefile`*	Displays modulefile setting contents

Example of command use

Frequently used module commands are shown below along with examples of their use.

- module avail
  - Displays a list of provided modulefiles

```
[_LNlogin]$ module avail
-------------------------------------------- /opt/intel/oneapi/modulefiles ---------------------------------------------
advisor/2025.1                  compiler/2025.1.0       dpct/latest                  ishmem/1.3.0   umf/latest
advisor/latest                  compiler/latest         dpl/2022.6                   ishmem/latest  vtune/2025.1
ccl/2021.13.1                   compiler32/2024.2.1     dpl/2022.8                   mkl/2024.2     vtune/latest
ccl/2021.15.0                   debugger/2024.2.1       dpl/latest                   mkl/2025.1
ccl/latest                      debugger/2025.1.0       ifort/2024.2.1               mkl/latest
compiler-intel-llvm/2024.2.1    debugger/latest         ifort32/2024.2.1             mkl32/2024.2
compiler-intel-llvm/2025.1.0    dev-utilities/2024.2.0  intel_ipp_ia32/2021.12       mpi/2021.13
compiler-intel-llvm/latest      dev-utilities/2025.1.0  intel_ipp_intel64/2021.12    mpi/2021.15
compiler-intel-llvm32/2024.2.1  dev-utilities/latest    intel_ipp_intel64/2022.1     mpi/latest
compiler-rt/2024.2.1            dnnl/3.5.0              intel_ipp_intel64/latest     tbb/2021.13
compiler-rt/2025.1.0            dnnl/3.7.1              intel_ippcp_ia32/2021.12     tbb/2022.1
compiler-rt/latest              dnnl/latest             intel_ippcp_intel64/2021.12  tbb/latest
compiler-rt32/2024.2.1          dpct/2024.2.0           intel_ippcp_intel64/2025.1   tbb32/2021.13
compiler/2024.2.1               dpct/2025.1.0           intel_ippcp_intel64/latest   umf/0.10.0

----------------------------------------- /work/Fugaku-environment/modulefiles -----------------------------------------
lang/tcsds-1.2.39  lang/tcsds-1.2.40  lang/tcsds-1.2.41(default)
```

Example of command usage

- `module load`
  - Sets environment variables by loading the modulefile and make the application ready for use

- `module unload`
  - Unloads environment settings added by the `module load`

- `module list`
  - Views the list of loaded modulefiles

Example of command usage

- Example of using the list, unload, and load commands for the language environment lang

```
[_LNlogin]$ module list
Currently Loaded Modulefiles:
 1) lang/tcsds-1.2.41(default)
[_LNlogin]$ module unload lang
[_LNlogin]$ module list
No Modulefiles Currently Loaded.
[_LNlogin]$ module load lang
[_LNlogin]$ module list
Currently Loaded Modulefiles:
 1) lang/tcsds-1.2.41(default)
```

Example of command usage

- module switch
  - Change the modulefile to different versions (the texts after the slash are different)

```
[_LNlogin]$ module list
Currently Loaded Modulefiles:
 1) lang/tcsds-1.2.41(default)
[_LNlogin]$ module switch lang/tcsds-1.2.40
[_LNlogin]$ module list
Currently Loaded Modulefiles:
  1) lang/tcsds-1.2.40
```

- module purge
  - Delete the settings of all loaded modulefiles

```
[_LNlogin]$ module list
Currently Loaded Modulefiles:
 1) lang/tcsds-1.2.41(default)
[_LNlogin]$ module purge
[_LNlogin]$ module list
No Modulefiles Currently Loaded.
```

Example of command usage

- module show
  - Display current modulefile setting contents

```
[_LNlogin]$ module show lang
-------------------------------------------------------------------
/work/Fugaku-environment/modulefiles/lang/tcsds-1.2.41:

module-whatis   {Fujitsu Compiler 4.12.0 (Fortran/C/C++/Tool)}
conflict        lang
setenv          FJSVXTCLANGA /opt/FJSVxtclanga/tcsds-1.2.41
prepend-path    PATH /opt/FJSVxtclanga/tcsds-1.2.41/bin
prepend-path    LD_LIBRARY_PATH /opt/FJSVxtclanga/tcsds-1.2.41/lib64
-------------------------------------------------------------------
```

- Introduction
- Access to Fugaku
- Compilers
- Mathematical Libraries
- Environment Module Packages
- Spack
- Scripting Languages
- Job Submission
- MPI Job Submissions and Precautions
- Checking the Resource Usage
- LLIO
- Advanced Usage

Overview

Fugaku manages and provides open-source software (OSS) through Spack.

- Spack
  - Package management software for supercomputer systems
  - Official web site: `https://spack.io/`

- Features
  - Open-source software recipes written for Spack can be used to properly compile software while including dependencies with other software.
  - Compilers and libraries can easily be switched.
  - "Chaining" functionality allows packages installed in one Spack instance* to refer to packages installed in other instances.
  - In Fugaku, public instances prepared on the system side and private instances managed by each user can both be used.

* A Spack *instance* refers to all the files needed for compilation, including the Spack script files, related files, and managed packages.

Public Instance

Fugaku provides pre-built OSS in a Spack instance as the public instance.

To use it, you only need to source the environment script.

- Sourcing the environment script for the public instance
  - For bash:

```
$ . /vol0004/apps/oss/spack/share/spack/setup-env.sh
```

  - For csh/tcsh:

```
$ setenv SPACK_ROOT /vol0004/apps/oss/spack
$ source /vol0004/apps/oss/spack/share/spack/setup-env.csh
```

  - For batch jobs, insert the above settings at the top of the job script.
  - Currently, we do not recommend including this line in your login script, .bashrc, etc. This can cause the login to fail when the filesystem is not stable.

Major Commands of Spack

Spack's main commands are listed below.

- spack find
  - List and search installed packages
  - Use option -x to only show explicitly installed packages
    - You can use this to check the OSS provided on Fugaku
    - If option -x is not specified, many packages implicitly installed because of the dependencies of the provided software are also listed

```
$ spack find -x
-- linux-rhel8-a64fx / fj@4.8.1 ---------------------------------
ffvhc-ace@0.1

-- linux-rhel8-a64fx / fj@4.10.0 --------------------------------
adios2@2.9.2           frontistr@5.5            modylas-new@1.1.0       paraview@5.11.2        py-seaborn@0.12.2
akaikkr@2002v010       fugaku-frontistr@master  mptensor@0.3.0          parmetis@4.0.3         py-spglib@2.0.2
akaikkr@2021v001       fujitsu-fftw@1.1.0       mvmc@1.2.0              petsc@3.19.6           py-toml@0.10.2
akaikkr@2021v002       fujitsu-mpi@head         n2p2@2.1.4              pfapack@2014-09-17     py-xarray@2023.7.0
alamode@1.3.0          fujitsu-ssl2@head        nemo@4.2.0              phase0@2021.02         python@3.11.6
alamode@1.4.2          genesis@2.1.1            netcdf-c@4.9.2          phase0@2021.02         quantum-espresso@6.5
alamode@1.5.0          genesis@2.1.1            netcdf-cxx@4.2          phase0@2023.01         quantum-espresso@6.6
batchedblas@1.0        genesis@2.1.2            netcdf-cxx4@4.3.1       phase0@2023.01         quantum-espresso@6.7
bcftools@1.12          genesis@2.1.2            netcdf-fortran@4.6.1    picard@3.0.0           quantum-espresso@6.8
bedtools2@2.31.0       gmt@6.2.0                netlib-lapack@3.10.1    povray@3.7.0.8         quantum-espresso@7.0
biobambam2@2.0.177     grads@2.2.3              netlib-scalapack@2.2.0  py-ase@3.21.1          quantum-espresso@7.1
blitz@1.0.2            gromacs@2020.6           nwchem@master           py-dask@2022.10.2      quantum-espresso@7.2
boost@1.83.0           gromacs@2021.5           octa@8.4                py-devito@4.8.1        quantum-espresso@7.3
. . . .
```

Major Commands of Spack

- spack load
  - Modify environment variables such as PATH to make packages managed by an instance of Spack available.
  - For example, for the octa package:

```
$ spack load octa
```

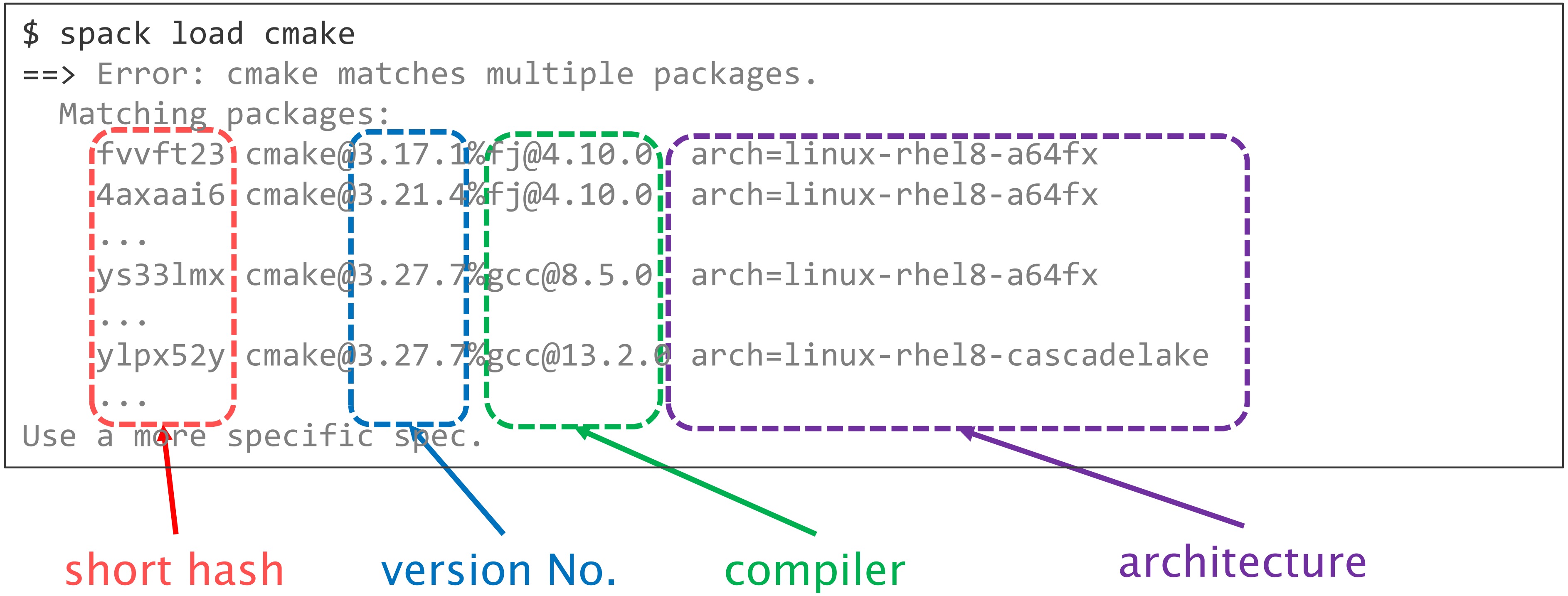
- spack unload
  - Unload environment variable settings added by the spack load command described above and restores them to their original state.
  - For example, for the octa package:

```
$ spack unload octa
```

Distinguishing between Multiple Packages

If there are packages with the same name on Spack, specifying only the package name in spack load will result in an error. In such a case, it is necessary to add the additional information that identifies the intended package.

- Example of a spack load error due to the same package name



```
$ spack load cmake
==> Error: cmake matches multiple packages.
  Matching packages:
    fvvft23 cmake@3.17.1%fj@4.10.0  arch=linux-rhel8-a64fx
    4axaai6 cmake@3.21.4%fj@4.10.0  arch=linux-rhel8-a64fx
    ...
    ys33lmx cmake@3.27.7%gcc@8.5.0  arch=linux-rhel8-a64fx
    ...
    ylpx52y cmake@3.27.7%gcc@13.2.0 arch=linux-rhel8-cascadelake
    ...
Use a more specific spec.
```

short hash

version No.

compiler

architecture

Distinguishing between Multiple Packages

- Designation using version number
  - Specify using “@” after the package name

```
$ spack load lammps@20201029
$ spack load lammps@20220623.2
$ spack load lammps@20230802.3
```

- Designation according to the compiler used
  - Specify using “%” after the package name

```
$ spack load tmux%fj@4.10.0
$ spack load tmux%gcc@13.2.0
```

Distinguishing between Multiple Packages

- Designation according to the the architecture
  - Specify using "arch=name-of-architecture" after the package name

```
$ spack load tmux arch=linux-rhel8-cascadelake
$ spack load tmux arch=linux-rhel8-a64fx
```

  - Architecture details

linux-rhel8-cascadelake:	build for login node
linux-rhel8-skylake_avx512:	build for login node
linux-rhel8-a64fx:	build for compute node

Distinguishing between Multiple Packages

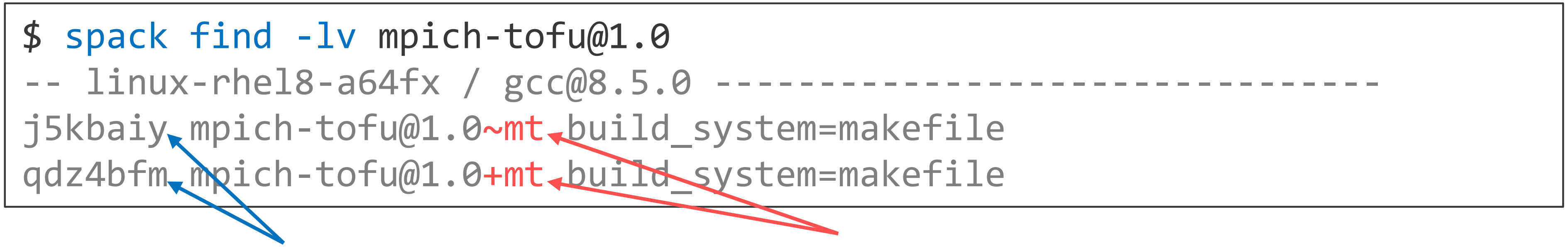
- Designation according to the the hash
  - Spack defines a unique hash for a build along with its detailed conditions called spec
  - Specify using a short hash (*) after "/"

```
[_LNlogin]$ spack find -l python

 (snip)

-- linux-rhel8-cascadelake / gcc@13.2.0 -------------------------
yjlixq5 python@3.11.6  so5pyv6 python@3.11.6

 (snip)

[_LNlogin]$ spack load /yjlixq5
[_LNlogin]$ spack load /so5pyv6
```

* A short hash is the first seven characters of the hash. The short hash can be found using "spack find -l" and full hash can be found using "spack find -L"

Distinguishing between Multiple Packages

- Enable/disable variants (installation options)
  - In some cases, multiple packages are installed that cannot be distinguished by version number, architecture, etc.
  - In such a case, the command “spack find -lv” can be used to check the installation details expressed in the variant.
  - In the following example, there is a difference between the “~mt” and “+mt” parts.
  - “~” indicates that the variant is disabled, “+” indicates that it is enabled.



```
$ spack find -lv mpich-tofu@1.0
-- linux-rhel8-a64fx / gcc@8.5.0 ---------------------------------
j5kbaiy mpich-tofu@1.0~mt build_system=makefile
qdz4bfm mpich-tofu@1.0+mt build_system=makefile
```

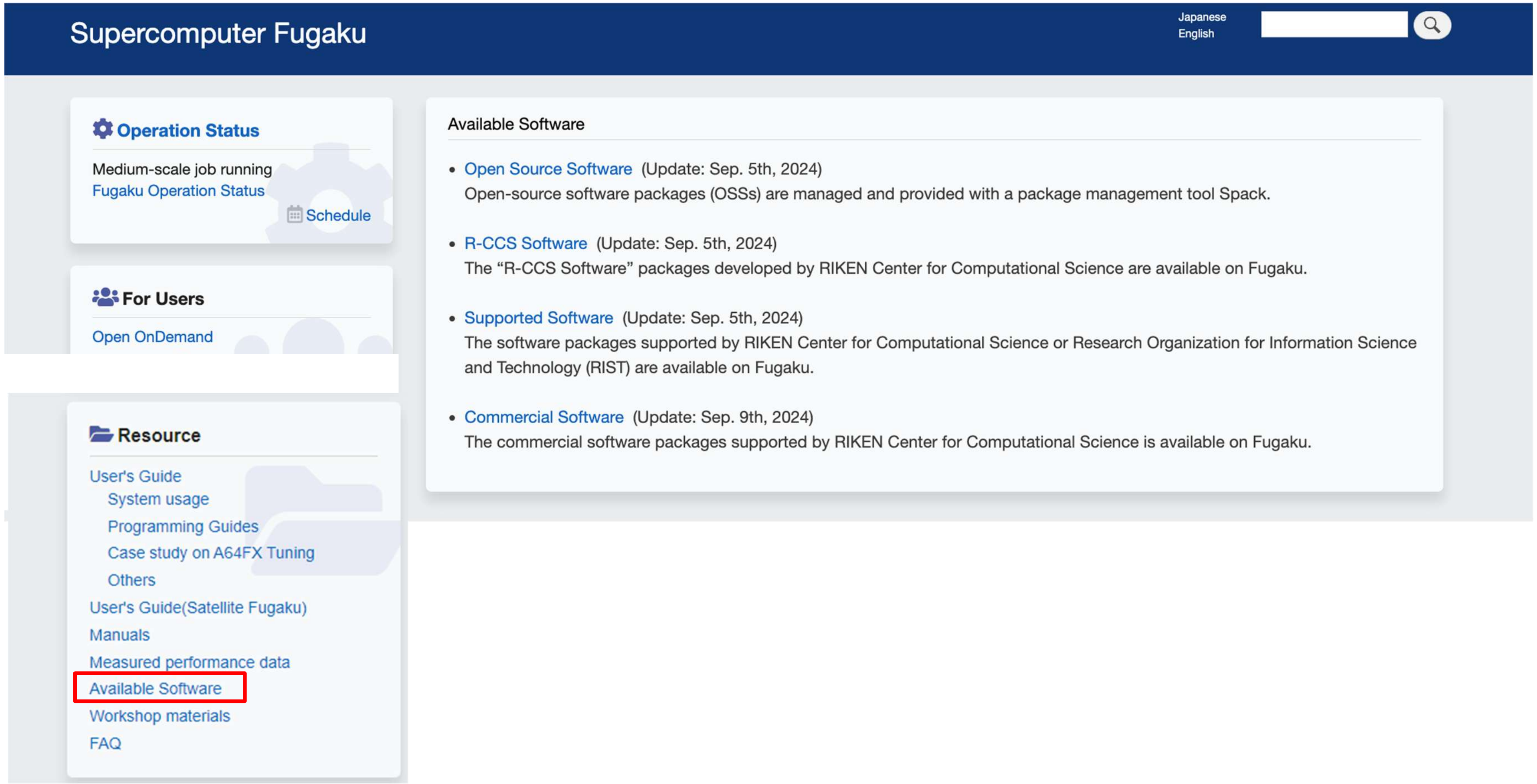
Each short hash

Variants (installation options)

Available Software Packages

Information of pre-installed software packages is provided in Fugaku Website.

- https://www.fugaku.r-ccs.riken.jp/en/docs/software_r01



Available Software

- Open Source Software (Update: Sep. 5th, 2024)
  Open-source software packages (OSSs) are managed and provided with a package management tool Spack.
- R-CCS Software (Update: Sep. 5th, 2024)
  The “R-CCS Software” packages developed by RIKEN Center for Computational Science are available on Fugaku.
- Supported Software (Update: Sep. 5th, 2024)
  The software packages supported by RIKEN Center for Computational Science or Research Organization for Information Science and Technology (RIST) are available on Fugaku.
- Commercial Software (Update: Sep. 9th, 2024)
  The commercial software packages supported by RIKEN Center for Computational Science is available on Fugaku.

Private Instances

In addition to the a forementioned public instance, each user can install a private instance of Spack in their home directory and use it to manage the apps.

- It is possible to refer to Spack packages provided by public instance using the chaining functionality.

- Reference materials for private instances
  - *Fugaku Spack User Guide*（listed on the Fugaku website)
  - Official Spack documentation (https://spack.readthedocs.io/en/stable/)

Known Issues and Solutions

This and the following slides describe problems that have been identified in the current environment using Spack and how they are addressed.

- Path for the dynamic link libraries of the operating system
  - When executing a program after loading some Spack packages, you may have the following warnings or error.

```
[WARN] xos LPG 2002 - Failed to map HugeTLBfs for data/bss: /usr/bin/file The
e_type of elf header must be ET_EXEC when using libmpg. You can check it on your
load module by readelf -h command.
```

```
[WARN] xos LPG 2003 - Failed to map HugeTLBfs for data/bss: Layout problem with
segments 0 and 1:
        Segments would overlap.
```

```
libmpg BUG!! mpiexec: __mpg_resolve_libc_symbol[776]: Assertion
`__libc_calloc_fp != ((void *)0)' failed.
```

  - After you call the "spack load" command, set the environment variable LD_LIBRARY_PATH again as follows.

```
export LD_LIBRARY_PATH=/lib64:$LD_LIBRARY_PATH
```

Known Issues and Solutions

- Performance degradation in multi-node jobs
  - Packages provided via Spack are stored in second-layer storage[*1]. Therefore, if you use these packages in a multi-node job, the performance may degrade due to concentrated access on a certain storage I/O node.
  - In such cases, after loading all necessary packages, you can avoid performance degradation by running the “dir_transfer” command on the paths set in LD_LIBRARY_PATH and PATH[*2] as follows.

*1 See the section on LLIO.

*2 This allows shared libraries and any referenced executables to be distributed to the cache area of the second-layer storage, which is on first-layer storage, avoiding access concentration.

```
$ spack load xxx
$ echo $LD_LIBRARY_PATH | sed –e ‘s/:/¥n/g’ | grep ‘^/vol0004/apps/oss/spack’ |
xargs /home/system/tool/dir_transfer
$ echo $PATH | sed -e 's/:/¥n/g' | grep '^/vol0004/apps/oss/spack' | xargs
/home/system/tool/dir_transfer
```

- Introduction
- Access to Fugaku
- Compilers
- Mathematical Libraries
- Environment Module Packages
- Spack
- **Scripting Languages**
- Job Submission
- MPI Job Submissions and Precautions
- Checking the Resource Usage
- LLIO
- Advanced Usage

Overview

This section describes how to use the OSS scripting languages for the compute nodes of Fugaku.

- Script languages provided explicitly and their versions

Languages	Spack (Login nodes)	Spack (Compute nodes)
Python3	3.11.6	3.10.8, 3.10.13, 3.11.6 (*)
Ruby	-----	3.1.0
R	-----	4.3.0
Julia	-----	1.9.3, 1.10.2

(as of Apr. 17th, 2025)

*The thread-parallel library `fjlapackexsve` that is optimized for a64fx is used for BLAS and LAPACK required by NumPy and SciPy.

Python

- How to use Python3 via Spack
  - When loading a python-related package, the corresponding Python3 packages and other depended packages are automatically loaded as well.

```
[_CNlogin]$ spack load /7hfnxfw   # short hash for py-pandas
[_CNlogin]$ which python
/vol0004/apps/oss/spack-v0.21/opt/spack/linux-rhel8-a64fx/fj-4.10.0/python-
3.10.13-bjhdatlk74sdqya3xxy2r6doz6y24iha/bin/python
[_CNlogin]$ spack load /pxi6xpt # short hash for py-pip
[_CNlogin]$ pip list
Package         Version
--------------- -------
Bottleneck      1.3.7
llvmlite        0.40.0
numba           0.57.0
numexpr         2.8.4
numpy           1.24.4   # numpy is loaded as well as pandas.
pandas          2.1.2
pip             23.1.2
python-dateutil 2.8.2
pytz            2023.3
setuptools      59.4.0
six             1.16.0
tzdata          2023.3
```

Python

- AI frameworks provided on Fugaku
  - scikit-learn

```
[_CNlogin]$ . /vol0004/apps/oss/spack/share/spack/setup-env.sh
[_CNlogin]$ spack load py-scikit-learn
```

  - Keras

```
[_CNlogin]$ . /vol0004/apps/oss/spack/share/spack/setup-env.sh
[_CNlogin]$ spack load py-keras
```

  - PyTorch-1.7.0 (*)

```
[_CNlogin]$ export PATH=/vol0004/apps/oss/PyTorch-1.7.0/bin:$PATH
[_CNlogin]$ export LD_LIBRARY_PATH=/vol0004/apps/oss/PyTorch-1.7.0/lib:$LD_LIBRARY_PATH
```

  - TensorFlow-2.2.0 (*)

```
[_CNlogin]$ export PATH=/vol0004/apps/oss/TensorFlow-2.2.0/bin:$PATH
[_CNlogin]$ export LD_LIBRARY_PATH=/vol0004/apps/oss/TensorFlow-2.2.0/lib:$LD_LIBRARY_PATH
```

* Spack-based environments of PyTorch and TensorFlow are still under construction. It is necessary to edit PATH and LD_LIBRARY_PATH in the current situation.

Ruby, R, Julia

On Fugaku, Ruby, R, and Julia are available. This slide shows how to use them.

■ How to use Ruby

```
[_CNlogin]$ . /vol0004/apps/oss/spack/share/spack/setup-env.sh
[_CNlogin]$ spack load ruby
[_CNlogin]$ ruby sample.rb
```

■ How to use R

```
[_CNlogin]$ . /vol0004/apps/oss/spack/share/spack/setup-env.sh
[_CNlogin]$ spack load r
[_CNlogin]$ Rscript sample.R
```

■ How to use Julia

```
[_CNlogin]$ . /vol0004/apps/oss/spack/share/spack/setup-env.sh
[_CNlogin]$ spack load julia@1.10.2
[_CNlogin]$ julia sample.jl
```

Java

The Java compiler can be used on both the login node and computer node of Fugaku. It is used as follows.

- Compiler environment setting
  - Compiling on the login node

```
[_LNlogin]$ . /vol0004/apps/oss/spack/share/spack/setup-env.sh # Environment setting of Spack
[_LNlogin]$ spack load openjdk arch=linux-rhel8-cascadelake  # Environment setting of OpenJDK
```

  - Compiling on the compute node

```
[_CNlogin]$ . /vol0004/apps/oss/spack/share/spack/setup-env.sh # Environment setting of Spack
[_CNlogin]$ spack load openjdk@17 arch=linux-rhel8-a64fx     # Environment setting of OpenJDK
```

- Usable environment (as of Apr. 17th, 2025)

Node	Software name	Language	Version
Login Node	OpenJDK	Java	11.0.20.1_1
Compute Node	OpenJDK	Java	11.0.20.1_1 17.0.8.1_1

Java

- Compile command
  - The translation command name is the same both on the login node and on the compute node.
  - If the MPI library is not used

```
[_LNlogin]$ javac [compile options] source_file
```

  - If the MPI library is used

```
[_LNlogin]$ mpijavac [compile options] source_file
```

■ List of compile options of mpijavac

Compile option	Description
`--showme`	Displays the calling line when the translation command of the MPI program calls the javac command. Does not perform translation.
`--verbose`	Displays the calling line when the translation command of the MPI program calls the javac command. Performs translation.
`--help, -help, -h`	Displays help message. Does not perform translation.
`avac_arguments`	Specifies options for the javac command.
`-classpath`	Specifies the path to the jar file.*

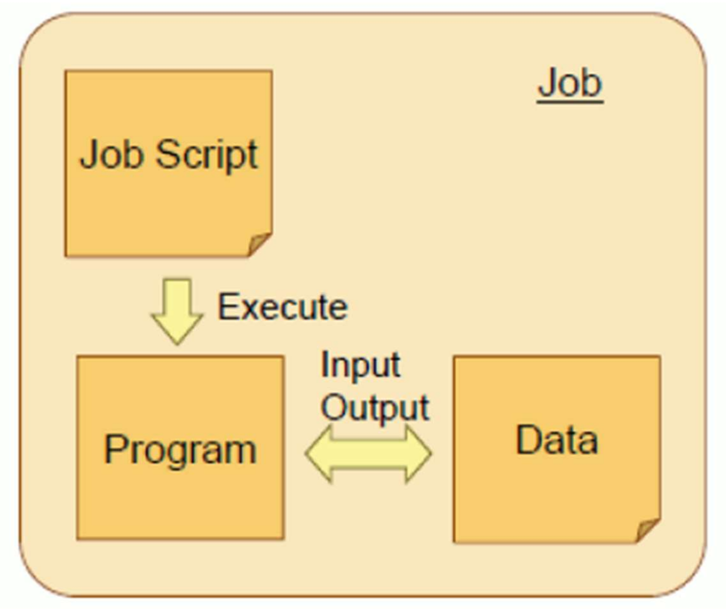
* If both the -classpath option and the CLASSPATH environment variable are set when specifying the jar file path, the -classpath option takes precedence.

- Introduction
- Access to Fugaku
- Compilers
- Mathematical Libraries
- Environment Module Packages
- Spack
- Scripting Languages
- **Job Submission**
- MPI Job Submissions and Precautions
- Checking the Resource Usage
- LLIO
- Advanced Usage

About the Jobs

In this section, we provide an overview of the job scheduling system in Fugaku.

- About the jobs
  - A job consists of programs, data, and job scripts.
  - The user requests execution to the job operation management function of the job operation software in Fugaku.
- The job operation management function secures the required computer resources and executes the program.



Job Scripts

A job script is a type of shell script.

- Job script example: without MPI

```
#!/bin/bash   # User’s login shell is used unless a shell is specified
#PJM --gname hp250xxx      # group name
#PJM -L "node=1"           # Number of nodes
#PJM -L "rscgrp=small"     # Specify resource group
#PJM -L "elapse=60:00"     # Job run time limit value
#PJM -s                    # Direction of statistic information file output

export OMP_NUM_THREADS=12   # Environment variable setting

# execute job
./a.out                     # Execute a program
```

- The specification of either --gname or --gid is mandatory.
- Lines starting with “#PJM” specify options of the pjsub command.
- As soon as a line that is not a comment appears, subsequent “#PJM” lines are simply ignored by the job scheduler as a comment line.
- In a job script, do not redirect to /dev/stdout or /dev/stderr.

Job Scripts

■ Job script example: with MPI

```
#!/bin/bash                # User's login shell is used unless a shell is specified
#PJM --gname hp250xxx                # group name
#PJM -L "node=4"             # Number of nodes
#PJM -L "rscgrp=small"       # Specify resource group
#PJM -L "elapse=10:00"       # Job run time limit value
#PJM --mpi "max-proc-per-node=4"
              # Upper limit of number of MPI process created at 1 node
#PJM -S


export OMP_NUM_THREADS=12
llio_transfer ./a.out  # Transfer the common files. (More on LLIO chapter)
mpiexec  -n  16  ./a.out  #  Execute  with  maximum  number  of  available  process
               (16 at this example)
```

- The specification of either --gname or --gid is mandatory.
- One node consists of four CMGs (Core Memory Groups). For this reason, we recommend running with four processes per node.
  * CMG will be explained later.

Job Scripts

- When using volumes other than those assigned to your group, specify the disk volume using the PJM_LLIO_GFSCACHE environment variable

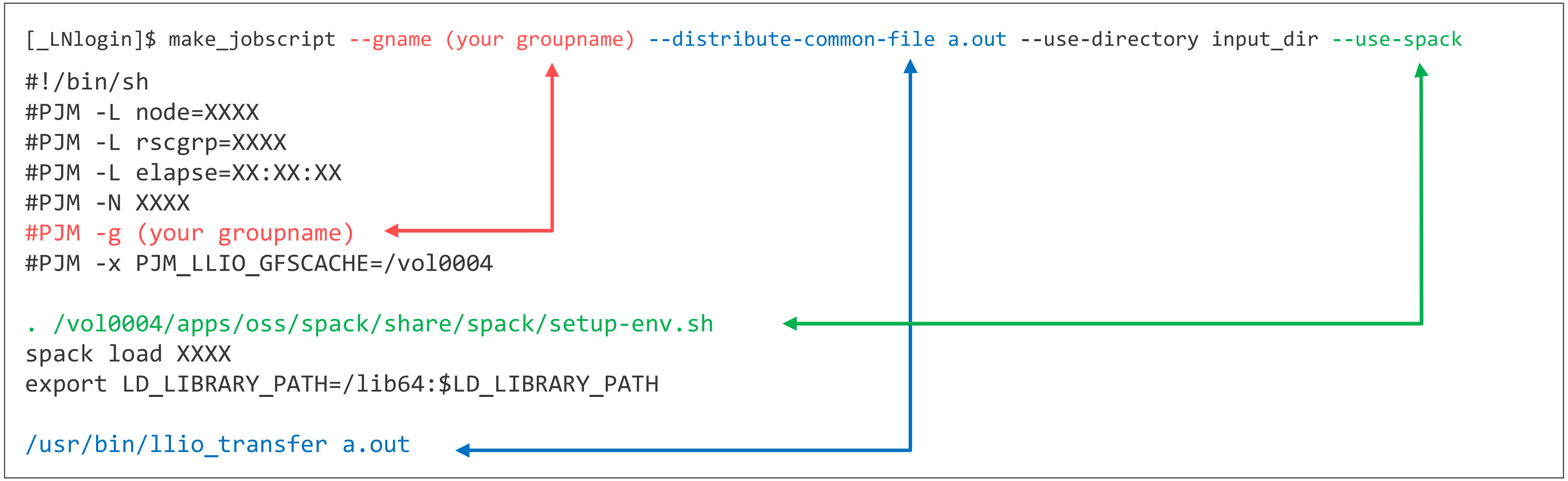
```
#!/bin/bash               # User's login shell is used unless a shell is specified
#PJM --gname  hp250xxx              # group name
#PJM -L  "node=1"            # Number of nodes
#PJM -L  "rscgrp=small"      # Specify resource group
#PJM -L  "elapse=10:00"      # Job run time limit value
#PJM -x  PJM_LLIO_GFSCACHE=/vol0004    # To use spack, specify /vol0004.
#PJM -s

. /vol0004/apps/oss/spack/share/spack/setup-env.sh
spack load dssp
export LD_LIBRARY_PATH=/lib64:$LD_LIBRARY_PATH
```

- The specification of either --gname or --gid is mandatory.
- You can check the disk volume of the current directory using the pwd –P command on a login node.
- You do not need to specify /vol0001 (2ndfs).

Generating Job Script Templates

Users can generate job script templates as below.

- Command and options
  - make_jobscript: a command to generate a job script template
  - --gname: Specifies group name (mandatory)
  - --distribute-common-file: Specifies file names to be distributed as the common file
  - --use-directory: Specifies the directory to use for the job
  - --use-spack: Specify this option when you use spack



```
[_LNlogin]$ make_jobscript --gname (your groupname) --distribute-common-file a.out --use-directory input_dir --use-spack

#!/bin/sh
#PJM -L node=XXXX
#PJM -L rscgrp=XXXX
#PJM -L elapse=XX:XX:XX
#PJM -N XXXX
#PJM -g (your groupname)
#PJM -x PJM_LLIO_GFSCACHE=/vol0004

. /vol0004/apps/oss/spack/share/spack/setup-env.sh
spack load XXXX
export LD_LIBRARY_PATH=/lib64:$LD_LIBRARY_PATH

/usr/bin/llio_transfer a.out
```

Job Scripts

- Basic options for the pjsub command (from the manual)

Option	Description
`-L "resource=value[,...]"`	Specifies the options related to job resources (more details follow) * -L and --rsc-list are the same option.
`--mpi "parameter[,...]"`	Specifies various parameters for the MPI job (more details follow)
`-g gname | -g gid`	Specifies the group name or group ID (mandatory) * You can use either `--gname` or `--gid` option to specify the *gname* or the *gid*.
`-j`	Writes the standard error output to the standard output
`-m "mailoption[,...]"`	Specifies whether to send email notifications regarding the job status and other information ■ When using this option, be sure to specify an email address using --mail-list.
`--mail-list "mailaddress[,...]"`	Specifies the mail destination ■ When specifying multiple addresses, separate them with commas (“,”). ■ The size of the specified string is limited to 255 characters. ■ If you enter a wrong e-mail address, it will not reach the recipient and there will be no error notification.
`--name "name"`	Specifies the name of the job ■ You can specify up to 63 bytes for the job name. The first character of the job name can only be a one-byte alphabet. ■ If the script file name is not specified, “STDIN” is used as the job name.

Job Scripts

■ Options for resource specification

Option	Description
-L "rscgrp=*rscgname*"	Specifies the resource group name to submit the job ■ Resource group information: https://www.fugaku.r-ccs.riken.jp/resource_group_config ■ Default values ■ small (batch job) ■ int (interactive type job)
-L "node=*nodeshape*"	Specifies the number of nodes and the shape to be assigned to the job ■ For 1 dimension: `node=N1` ■ For 2 dimensions: `node=N1xN2` ■ For 3 dimensions: `node=N1xN2xN3` ■ The torus/mesh can de specified. (resource group: small) ■ To check the default value, use the following command: pjacl --rg <*resource group name*>
-L "elapse=*elapsetimelimit*"	Sets the elapsed time limit for each job ■ The value of `elapsetimelimit` is specified in the format "[[time:] minute:] second". ■ `-L "elapse=30"` (30 seconds) ■ `-L "elapse=2:30"` (2 minutes and 30 seconds) ■ `-L "elapse=1:00:00"` (1 hour) ■ Default values ■ 1 minute （batch job） ■ 10 seconds （interactive job）

Job Scripts

- Options to set parameters related to MPI operations

Option	Description
--mpi "shape=*shape*"	Specifies the shape of the process to be started statically ■ You must specify the same number of dimensions specified by the node value in the -L (--rsc-list) option. ■ If omitted, the same value as specified in node is used.
--mpi "proc=*num*"	Specifies the maximum number of processes to start statically ■ If omitted, it is the product of the values specified by shape. ■ If the specified value is greater than the product of the value specified by shape times the number of CPU cores in the node, the job submission is rejected.
--mpi "max-proc-per-node=*mppnnum*"	Specifies the maximum number of processes per node ■ If omitted, the maximum number of process per node will be the value obtained by converting the value specified in proc to the number of processes created in one node (A). ■ If this value is smaller than (A), the job submission is rejected.

Job Scripts

- Options to output the job statistics

Option	Description
-s (Small letter)	Outputs the statistical information of the submitted job to a file ■ Cannot use with the -S option
-S (Large letter)	In addition to the information output by the -s option, the information per node of the submitted job is output ■ Cannot use with the -s option
--spath *pathname*	When changing the output file name, specify the *pathname* ■ The –s or –S option must also be used

- See the following references for details on job statistics
  - *Job Operation Software End-user's Guide*
  - *Job Operation Software Command Reference*, Section 3.3.2 "pjstatsinfo"
  - execute `man manual pjstatsinfo`

Job Scripts

■ Check the limit and default values

```
[_LNlogin]$ pjacl --rscgrp <resource group name>
```

■ Output example specifying “small” to a resource group:

```
 (Omitted)
defines
    default rscunit              rscunit_ft01
    default rscgroup             small
 (Omitted)
pjsub option parameters
    (-L/--rsc-list)                         lower            upper            default
        (elapse=)                           00:01:00         72:00:00         00:01:00
        (adaptive elapsed time min)         00:01:00         72:00:00         00:01:00
        (adaptive elapsed time max)         00:01:01         144:00:00        144:00:00
 (Omitted)
        (node=)                             1                384              1
 (Omitted)
```

* If the resource group is not specified, this will output the value of the default resource group (small)

Types of Jobs

Jobs are classified into several categories depending on the job type and job model.

- Classification by job type
  - Batch jobs
  - Interactive jobs

- Categories based on job model
  - Normal jobs
  - Bulk jobs
  - Step jobs
  - Workflow jobs

* Batch jobs

A normal job is the basis of job execution.

This seminar first explains job submission, checking the job status, job cancelation, and checking the exit status of normal jobs.

Normal Jobs

Normal jobs are batch jobs that do not have a special processing form.

The order of job execution is not always the same as the order of job submission.

- Normal job submission
  - Submit the job in the data area.

```
[_LNlogin]$ pjsub ./sample.sh
[INFO] PJM 0000 pjsub Job 9714 submitted.
```

Job ID

- See the following reference when error messages are generated on job submission
  - *Users Guide - Use and job execution*, Section 5.2.1 “Job submission”

Normal Jobs

- Displaying the job status

```
[_LNlogin]$ pjstat

JOB_ID   JOB_NAME   MD  ST   USER    START_DATE      ELAPSE_LIM  NODE_REQUIRE  VNODE  CORE  V_MEM
238      job.sh     NM  RUN  user1   11/17 09:01:41  0001:00:00  12:2x3x2      -      -     -
239      bulk.sh    BU  RUN  user1   11/17 09:01:42  0001:00:00  12:2x3x2      -      -     -
240      step.sh    ST  RUN  user1   11/17 09:01:42  -           -             -      -     -
241      job2.sh    NM  RUN  user1   11/17 09:01:42  0001:00:00  2             -      -     -
```

- Multiple job IDs or a range of IDs can be specified.

```
[_LNlogin]$ pjstat 238 239
```

```
[_LNlogin]$ pjstat 238-240
```

Normal Jobs

■ Canceling a job

```
[_LNlogin]$ pjdel 12345678
[INFO] PJM 0100 pjdel Accepted job 12345678.
```

■ Canceling multiple jobs

```
[_LNlogin]$ pjdel 12345678 12345680
[INFO] PJM 0100 pjdel Accepted job 12345678.
[INFO] PJM 0100 pjdel Accepted job 12345680.
```

- ■ If a job ID that does not exist or has already been deleted is specified, the message below will be displayed.

```
[ERR.] PJM 0112 pjdel Job “Unexisting job ID" does not exist.
```

Normal Jobs

- Job execution result
- When the job is completed, the job execution result is output to a file in the directory where the job was submitted
  - Single process job

Style	Description
Job_name.*Job_ID*.out	Data written to standard output by the job.
Job_name.*Job_ID*.err	Data written to standard error output by the job.
Job_name.*Job_ID*.stats	This file contains the job's statistical information. This will be output if the –s or -S option is specified when the job is submitted.

Normal Jobs

- MPI jobs
  - The output directories are automatically generated in the working directory as follows for the output from each rank and mpiexec.

```
output.XXXXXXXX/          #XXXXXXXX: job id
└ 0/                      #directory carrying the outputs of ranks 0-999
   ├ 1/ #1st MPI execution in the job script (mpiexec = 1)
   └ 2/ #2nd MPI execution in the job script (mpiexec = 2)
```

- The output files of each *mpiexec** and *rank*** are generated in each directory as follows.

Style	Description
stdout.*mpiexec*.*rank*	Standard output of rank *rank* of the *mpiexec*-th MPI execution
stderr.*mpiexec*.*rank*	Standard error output of rank *rank* of the *mpiexec*-th MPI execution

* *mpiexec*: order of MPI execution in the job script
** *rank*: rank number in the *mpiexec*-th execution

- Page 125 explains how to change the name of MPI output files.

Check the Job Exit Status

The following describes how to check whether the job ended normally or abnormally.

- Job manager exit code
  - To output the job manager exit code, specify the -s or -S option when executing pjsub.
- Message output during job execution
  - An error message may be output during job execution.

Check the Job Exit Status

- Job manager exit code
  - Check the PJM code in the job statistical information file.
  - Some PJM codes are defined as follows:

PJM code	Meaning
0	Successful job completion
1	Cancelled by the pjdel command controlled by the user
2	Rejected based on job acceptance criteria; the `pjsub` command will return an error
11	Job execution timeout because the time limit has been exceeded
12	Forced termination due to excessive memory usage
16	Termination due to inaccessibility of the current directory or standard input, standard output, or standard error output files
20	Node down

Check the Job Exit Status

- The meaning of error messages
  - Check the error messages in the standard error file.
  - Error message types and their reference manuals are listed below.

Error message	Reference manual
PLE *nnnn plexec*	*Job Operation Software* ■ *Command Reference* ■ *End-user's Guide*
PJM *nnnn xxxxxx*	*Job Operation Software* ■ *Command Reference* ■ *End-user's Guide*
mpi::	*MPI User's Guide*
jwe*nnnn*	*Development Studio Fortran/C/C++ Runtime Messages*

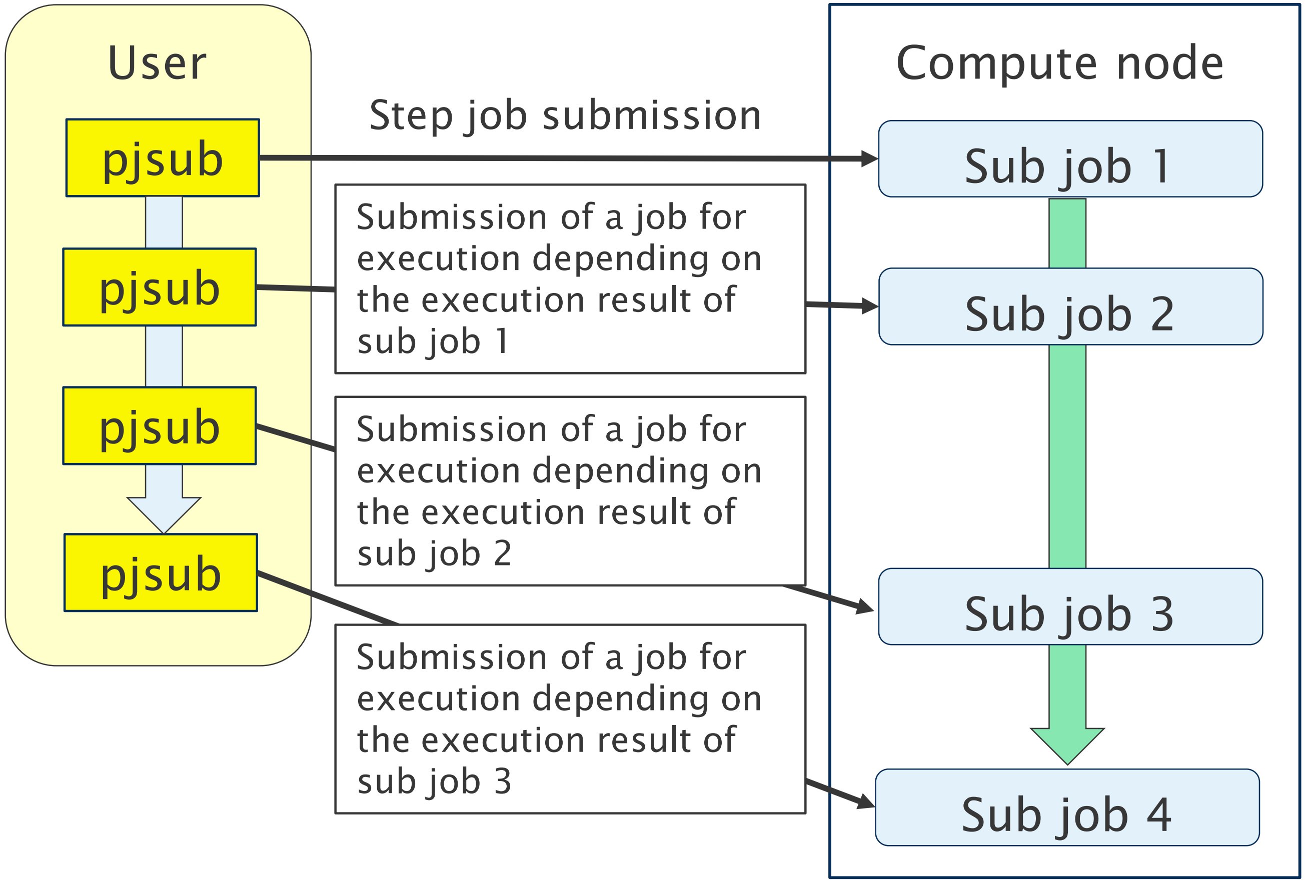
- PJM code does not respond to some hardware failures.
  The following command confirms that the job was stopped by a hardware failure or other system failures.

```
[_LNlogin]$ job_events
```

Step Jobs

A step job is a batch job that gives the jobs an execution order or dependencies.

■ Schematic of a step job


User
pjsub
pjsub
pjsub
pjsub
Step job submission
Submission of a job for execution depending on the execution result of sub job 1
Submission of a job for execution depending on the execution result of sub job 2
Submission of a job for execution depending on the execution result of sub job 3
Compute node
Sub job 1
Sub job 2
Sub job 3
Sub job 4

Sub job:
A step job comprises several sub jobs

- ■ Sub jobs are submitted one by one
- ■ The sub jobs can be controlled depending on the results of the completed job.

Step jobs are useful for long and divisible jobs.

Step Jobs

Job script files for normal jobs can be used for step jobs. You can specify different resources (the number of compute nodes, amount of memory, etc.) for each sub job.

- Step job submission
  - Start the step job (use --step)

```
[_LNlogin]$ pjsub --step job_1.sh
[INFO.] PJM 0000 Job 1234_0 submitted
```

1234: Job ID
0 : Step number
1234_0: Sub job ID

  - Execute the sub job after the second (specify the Job ID to jid)

```
[_LNlogin]$ pjsub --step --sparam "jid=1234" job_2.sh
[INFO.] PJM 0000 Job 1234_1 submitted
```

  - Example of submitting multiple sub jobs in a step job

```
[_LNlogin]$ pjsub --step job_1.sh job_2.sh
[INFO.] PJM 0000 Job 1235_0 submitted
[INFO.] PJM 0000 Job 1235_1 submitted
```

Step Jobs

- Specify a job name when submitting a step job
  - Submit the first sub job （use the --step option）

```
[_LNlogin]$ pjsub --step job_1.sh      # job name is ”moon"  specified in job script.
[INFO] PJM 0000 pjsub Job 2345_0 submitted.
```

- Execute the second and subsequent sub jobs
  - Specify the sub job name to jnam

```
[_LNlogin]$ pjsub --step --sparam "jnam=moon" job_2.sh
[INFO] PJM 0000 pjsub Job 2345_1 submitted.
```

* Two or more step jobs with the same job name might exist. If you submit a sub job that specifies the job name using the --sparam "jnam=” option, the sub job is assumed to be associated with the latest step job.

* If you specify a jnam that is different than the one specified using the --name option of pjsub, the job name becomes “jnam.”

Step Jobs

- Displaying the status of a step job
  - Normal display

```
[_LNlogin]$ pjstat
JOB_ID      JOB_NAME    MD ST  USER  START_DATE       ELAPSE_LIM  NODE_REQUIRE
1234        galaxy      ST RUN user  12/31 23:45:01   -           -
```

  - Display including the sub jobs (using the -E option)

```
[_LNlogin]$ pjstat -E
JOB_ID      JOB_NAME    MD ST  USER  START_DATE       ELAPSE_LIM  NODE_REQUIRE
1234        galaxy      ST RUN user  12/31 23:45:01   -           -
1234_0      galaxy      ST EXT user  12/31 23:45:01   0000:30:00  8
1234_1      galaxy      ST RUN user  01/01 00:12:34   0000:30:00  8
1234_2      galaxy      ST CCL user  -                0002:00:00  4
1234_3      galaxy      ST QUE user  -                0002:00:00  8
```

    - The next sub job is not scheduled until the execution of the current sub job is completed.

Step Jobs

- Canceling a step job
  - Canceling all sub jobs by specifying the job ID

```
[_LNlogin]$ pjdel 1234
[INFO] PJM 0100 pjdel Accepted job 1234_0.
[INFO] PJM 0100 pjdel Accepted job 1234_1.
[INFO] PJM 0100 pjdel Accepted job 1234_2.
```

  - Canceling a sub job by specifying one sub job ID

```
[_LNlogin]$ pjdel 1234_2
[INFO] PJM 0100 pjdel Accepted job 1234_2.
```

  - Canceling some sub jobs by specifying a sub job range

```
[_LNlogin]$ pjdel 1234_1-2
[INFO] PJM 0100 pjdel Accepted job 1234_1.
[INFO] PJM 0100 pjdel Accepted job 1234_2.
```

* Even if a sub job is canceled, the step job can be submitted later.

Step Jobs

■ Output of the job execution results

When the job ends, the job execution result is output to a file in the directory where the job was submitted.

■ Output file for each sub job

Style	Description
Job name.Sub job ID.out	Standard output of each sub job
Job name.Sub job ID.err	Standard error output of each sub job
Job name.Sub job ID.stats	Statistical information of each sub job

■ Output file for each step job

Style	Description
(The first job name).Job ID.stats	Statistical information of the entire step job

* The job statistical information file will be outputted if the –s or -S option is specified when submitting the job.

Step Jobs

When using step jobs, it is possible to control whether subsequent jobs are executed by referring to the return value of the job executed previously.

■ Job submission example with dependency

```
[_LNlogin]$ pjsub --step --sparam "jid=1234,sd=pc!=0:all:2" job_4.sh job_5.sh
```

Specify the step number of the sub jobs, referring to the end status.

■ Exit status（assign to “sd”）

ec: the job script end status of the dependent sub job
pc: the job exit code

＊When the job has ended abnormally, the job script end status (ec) may not be 0 (for example, when the elapsed time limit has been exceeded).

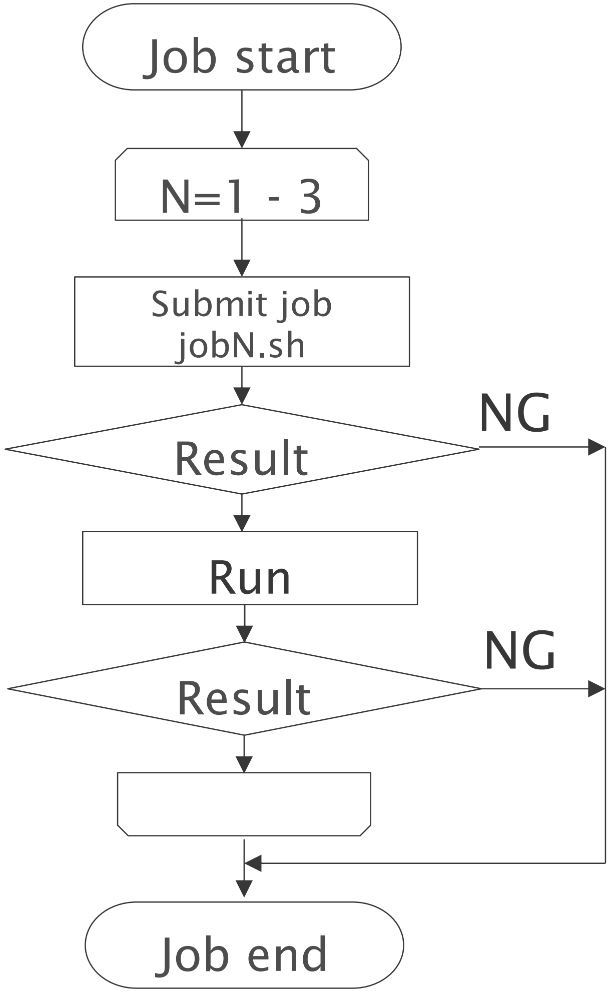
■ Delete type (specify after “:”)

one: only this sub job is deleted and subsequent sub jobs that depend on the results of this sub job are not deleted.
after: delete this sub job and subsequent sub jobs that depend on the results of this sub job.
all: delete this sub job and all subsequent sub jobs.

Workflow Jobs

Workflow jobs are job models in which a user controls job submission using a shell script.

■ Example of a workflow job



* Job scripts (job1.sh-job3.sh) can be used as the job script of a normal job.

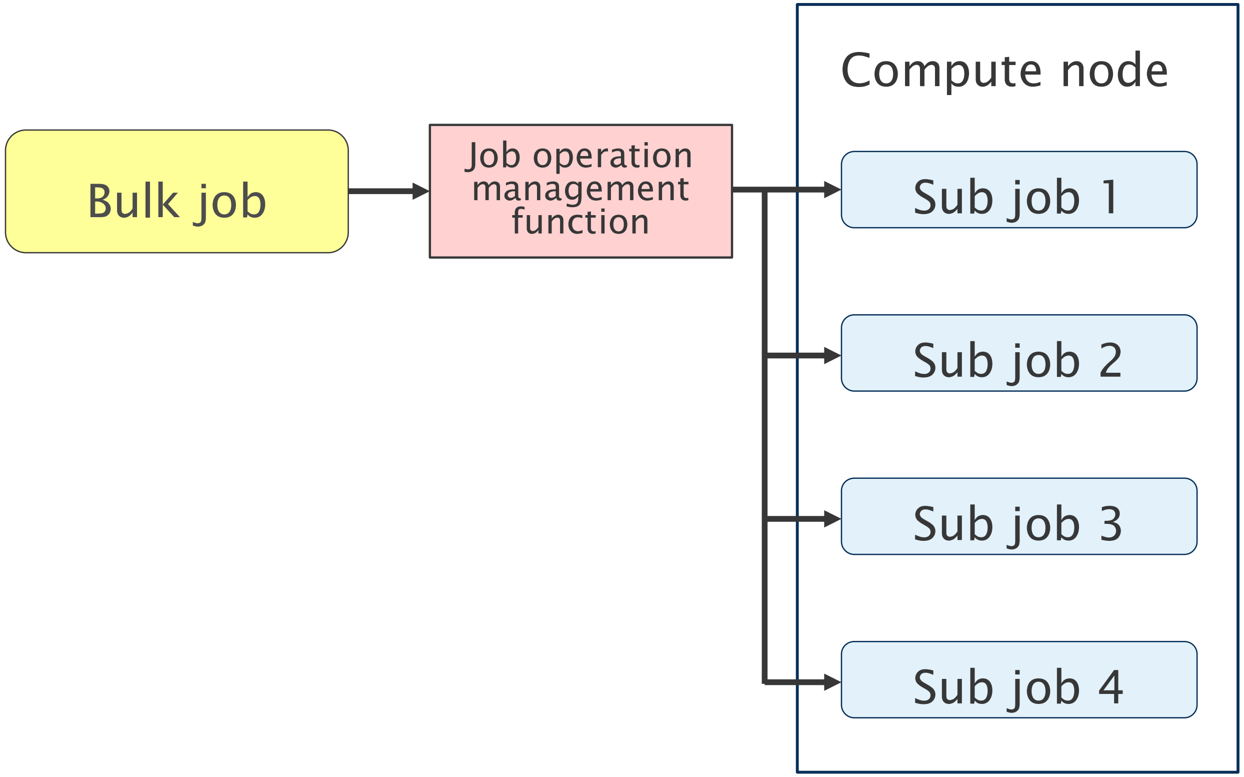
```
#!/bin/sh -x
for no in {1..3}
do
  JID=`pjsub -z jid job${no}.sh`
  if [ $? -ne 0 ];   # $?: job submission result
    exit 1
  fi
  set -- `pjwait $JID`
         # pjwait: job waiting command
  if [ $2 != "0" -o $3 != "0" ]; then
    exit 1
  fi
done

# pjwait $JID =>   $2: job exit code
                   $3: exit status of job script
```

* See the *Job Operation Software Command Reference* for instructions on “pjwait” and “pjsub –z jid”.

Bulk Jobs

Bulk jobs are job models for submitting many jobs that have the same form with different parameters as one job.

■ Schematic of a bulk job



Sub job:
Bulk jobs comprise several sub jobs.

Sub jobs are submitted in parallel.

Bulk jobs are useful for performing many jobs with different parameters.

Bulk Jobs

In a bulk job, the names of input/output files can contain the bulk number to change the input/output files for sub jobs.

■ An example job script for a bulk job

```
#!/bin/bash
#PJM -g hp250xxx
#PJM -L "node=2"
#PJM -L "rscgrp=small"
#PJM -L "elapse=01:00:00"
#PJM --mpi "max-proc-per-node=4
#PJM -S

llio_transfer ${PJM_JOBDIR}/a.out # Transfer common file (See LLIO session)
# ${PJM_JOBDIR} is a directory where the job was submitted

cd ${PJM_JOBDIR}/param_id${PJM_BULKNUM}
# ${PJM_BULKNUM} is interpreted as a bulk number

mpiexec --stdin test.inp --std-proc test.out ${PJM_JOBDIR}/a.out
```

Bulk Jobs

- Bulk job submission
  - Use the `--bulk` option to submit a bulk job with the `pjsub` command

```
[_LNlogin]$ pjsub --bulk --sparam "1-4" bulkjob.sh
[INFO] PJM 0000 pjsub Job 8123 submitted.
```

  - `--sparam` options are needed for bulk job submission
  - Specify the bulk number using `--sparam "startbulkno-endbulkno"`
    - Bulk numbers must be consecutive

      * Cannot specify, e.g., --sparam “3, 5, 8”
    - startbulkno must be smaller than endbulkno

      * Cannot specify, e.g., --sparam “4-1”.
    - Bulk numbers can start with anything other than 0
      - Bulk job can be submitted multiple times
Example: First time: --sparam “1-100"
Second time: --sparam “101-200"

Bulk Jobs

Displaying the status of a bulk job

- Normal display

```
[_LNlogin]$ pjstat
JOB_ID      JOB_NAME    MD ST  USER  START_DATE         ELAPSE_LIM  NODE_REQUIRE
1234        planet      BU RUN user  -                  0001:00:00  2
```

- Display including sub jobs (using the -E option)

```
[_LNlogin]$ pjstat -E
JOB_ID      JOB_NAME    MD ST  USER  START_DATE         ELAPSE_LIM  NODE_REQUIRE
1234        planet      BU RUN user  -                  0001:00:00  2
1234[1]     planet      BU RUN user  01/01 06:42:34     0001:00:00  2
1234[2]     planet      BU RUN user  01/01 06:42:34     0001:00:00  2
1234[3]     planet      BU RUN user  01/01 06:50:56     0001:00:00  2
1234[4]     planet      BU RUN user  01/01 07:06:23     0001:00:00  2
1234[5]     planet      BU QUE user  -                  0001:00:00  2
```

* Bulk job status is not always the same as the sub job status, because sub jobs will run one by one when the resources are ready.

* "[1]" is the bulk number, in JOB_ID("1234[1]").

Bulk Jobs

- Canceling a bulk job
  - Canceling all sub jobs by specifying a job ID

```
[_LNlogin]$ pjdel 1234
[INFO] PJM 0100 pjdel Accepted job 1234.
```

  - Canceling one sub job by specifying one sub job ID

```
[_LNlogin]$ pjdel 1234[2] 1234[3]
[INFO] PJM 0100 pjdel Accepted job 1234[2].
[INFO] PJM 0100 pjdel Accepted job 1234[3].
```

  - Canceling some sub jobs by specifying a sub job range

```
[_LNlogin]$ pjdel 1234[2-3]
[INFO] PJM 0100 pjdel Accepted job 1234[2-3].
```

Bulk Jobs

- Output of job execution results
- When the job ends, the job execution result is output to a file in the directory where the sub job was submitted

  - Output file for each sub job

Style	Description
Job name.Sub job ID.out	Standard output of each sub job
Job name.Sub job ID.err	Standard error output of each sub job
Job name.Sub job ID.stats	Statistical information of each sub job

  - Output file of the bulk job

Style	Description
Job name.Job ID.stats	Statistical information of the entire bulk job

* Job statistical information file will be outputted if the –s or -S option is specified when submitting the job.

Interactive Jobs

Interactive jobs are job types that can execute commands interactively within the allocated resources.

■ How to submit an interactive job

```
[_LNlogin]$ pjsub --interact -g hp250xxx -L "rscgrp=int" --mpi "proc=12" ¥
> -L "node=1, elapse=1:00:00" --sparam "wait-time=60" -x PJM_LLIO_GFSCACHE=/vol0004

[INFO] PJM 0000 pjsub Job 5678 submitted.
[INFO] PJM 0081 .connected.
[INFO] PJM 0082 pjsub Interactive job 5678 started.

[_CNlogin]$ python test.py
...test program running...
[_CNlogin]$ exit
[INFO] PJM 0083 pjsub Interactive job 5678 completed.
```

- elapse: Specify the maximum job execution time. The default is 10 seconds. This value must be specified.
- sparam "wait-time": Specify the computer resource allocation wait time in seconds or as unlimited. The default value is 0. Specify 60 or more.

* In interactive jobs, computer resources are used until the end of the job.

Interactive Jobs

In an interactive job, a shell script can be run in the compute nodes.

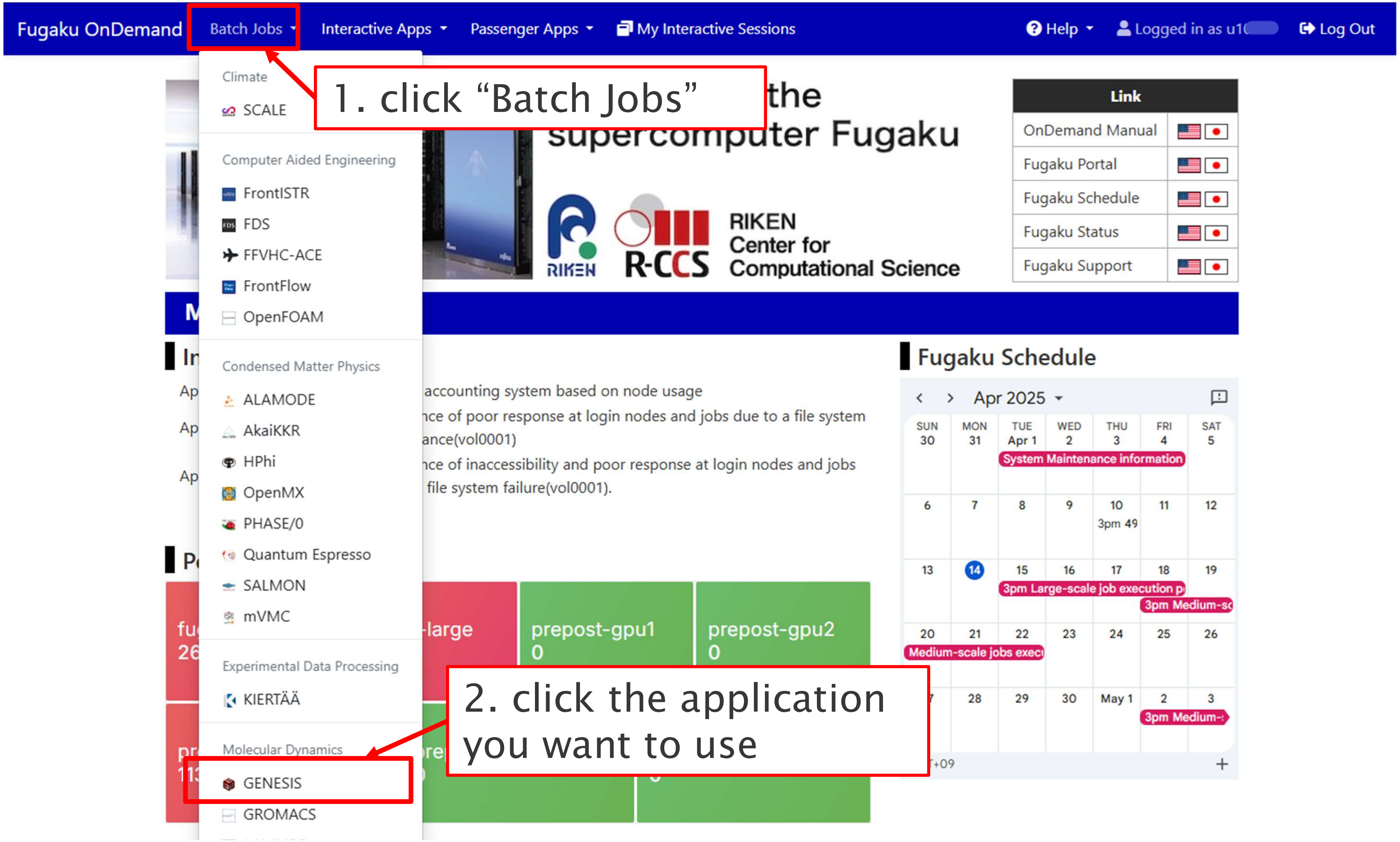
- Example of submitting an interactive job that specifies a job script

```
[_LNlogin]$ cat ./sample.sh
#! /bin/bash
# In interactive job, PJM parameters(#PJM) are not needed.
echo "HELLO, Fugaku"

[_LNlogin]$ pjsub --interact -g hp250xxx -L "rscgrp=int, elapse=0:10:00" ¥
> --sparam "wait-time=unlimited" ./sample.sh
[INFO] PJM 0000 pjsub Job 4567 submitted.
[INFO] PJM 0081 .....connected.
[INFO] PJM 0082 pjsub Interactive job 4567 started.
HELLO, Fugaku
[INFO] PJM 0083 pjsub Interactive job 4567 completed.
# Interactive job is ended with submitted shell script.
```

* In interactive job execution, a pseudo terminal is used to perform a series of job operations interactively. However, if a script file is specified, the pseudo terminal is not used.

Job Submission via Open Composer

Jobs with a preinstalled application can be submitted from Fugaku Open OnDemand.



1. click “Batch Jobs”

2. click the application you want to use

Link	
OnDemand Manual	🇺🇸 🇯🇵
Fugaku Portal	🇺🇸 🇯🇵
Fugaku Schedule	🇺🇸 🇯🇵
Fugaku Status	🇺🇸 🇯🇵
Fugaku Support	🇺🇸 🇯🇵

accounting system based on node usage

nce of poor response at login nodes and jobs due to a file system

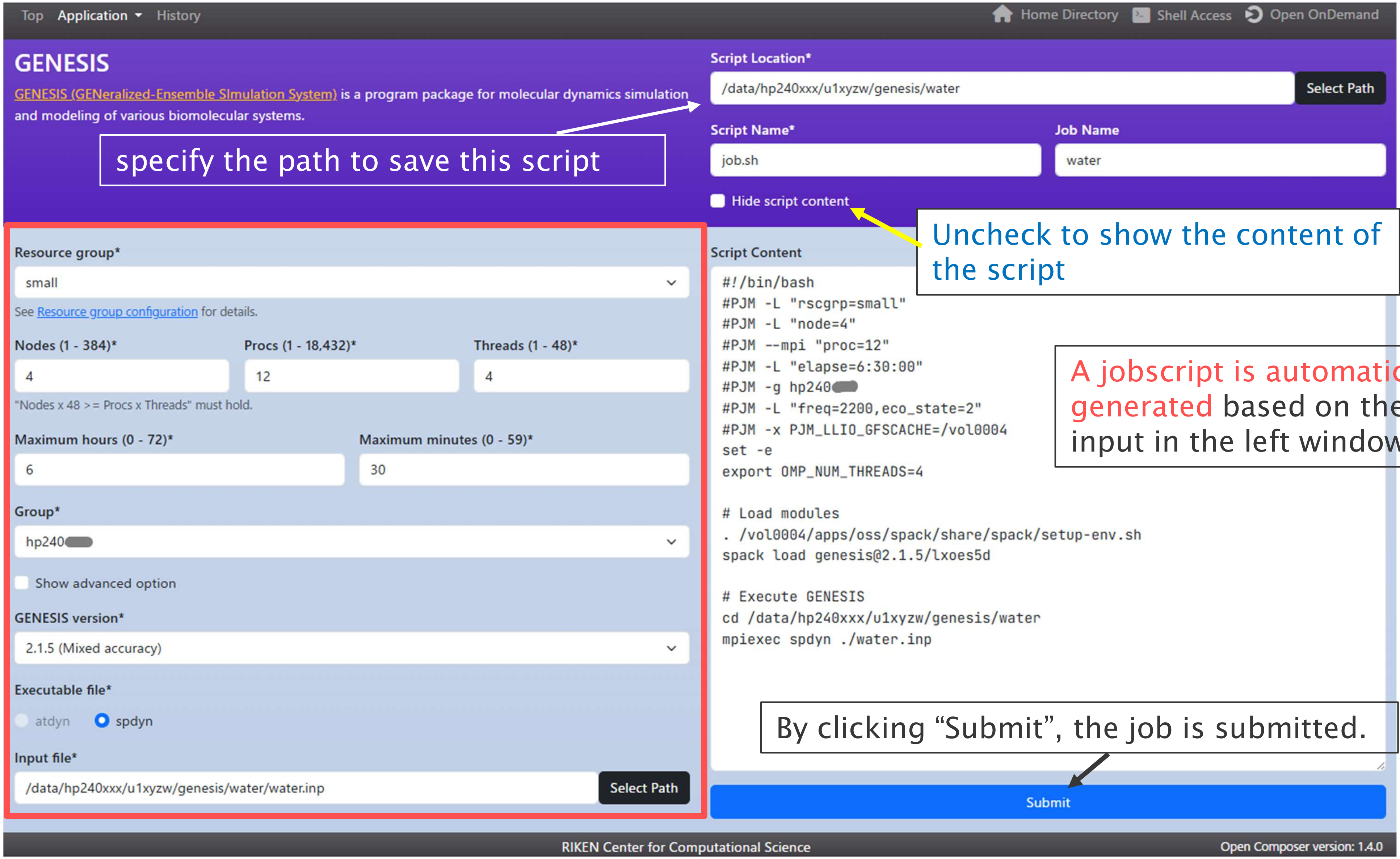
ance(vol0001)

nce of inaccessibility and poor response at login nodes and jobs

file system failure(vol0001).

Job Submission via Open Composer

Open Composer enables semi-automatic job submission as follows.



GENESIS

GENESIS (GENeralized-Ensemble SImulation System) is a program package for molecular dynamics simulation and modeling of various biomolecular systems.

specify the path to save this script

Script Location*: /data/hp240xxx/u1xyzw/genesis/water [Select Path]

Script Name*: job.sh

Job Name: water

☐ Hide script content

Uncheck to show the content of the script

Resource group*: small

See Resource group configuration for details.

Nodes (1 - 384)*: 4

Procs (1 - 18,432)*: 12

Threads (1 - 48)*: 4

"Nodes x 48 >= Procs x Threads" must hold.

Maximum hours (0 - 72)*: 6

Maximum minutes (0 - 59)*: 30

Group*: hp240[redacted]

☐ Show advanced option

GENESIS version*: 2.1.5 (Mixed accuracy)

Executable file*: ○ atdyn ● spdyn

Input file*: /data/hp240xxx/u1xyzw/genesis/water/water.inp [Select Path]

Script Content

```
#!/bin/bash
#PJM -L "rscgrp=small"
#PJM -L "node=4"
#PJM --mpi "proc=12"
#PJM -L "elapse=6:30:00"
#PJM -g hp240[redacted]
#PJM -L "freq=2200,eco_state=2"
#PJM -x PJM_LLIO_GFSCACHE=/vol0004
set -e
export OMP_NUM_THREADS=4

# Load modules
. /vol0004/apps/oss/spack/share/spack/setup-env.sh
spack load genesis@2.1.5/lxoes5d

# Execute GENESIS
cd /data/hp240xxx/u1xyzw/genesis/water
mpiexec spdyn ./water.inp
```

A jobscript is automatically generated based on the input in the left window

By clicking “Submit”, the job is submitted.

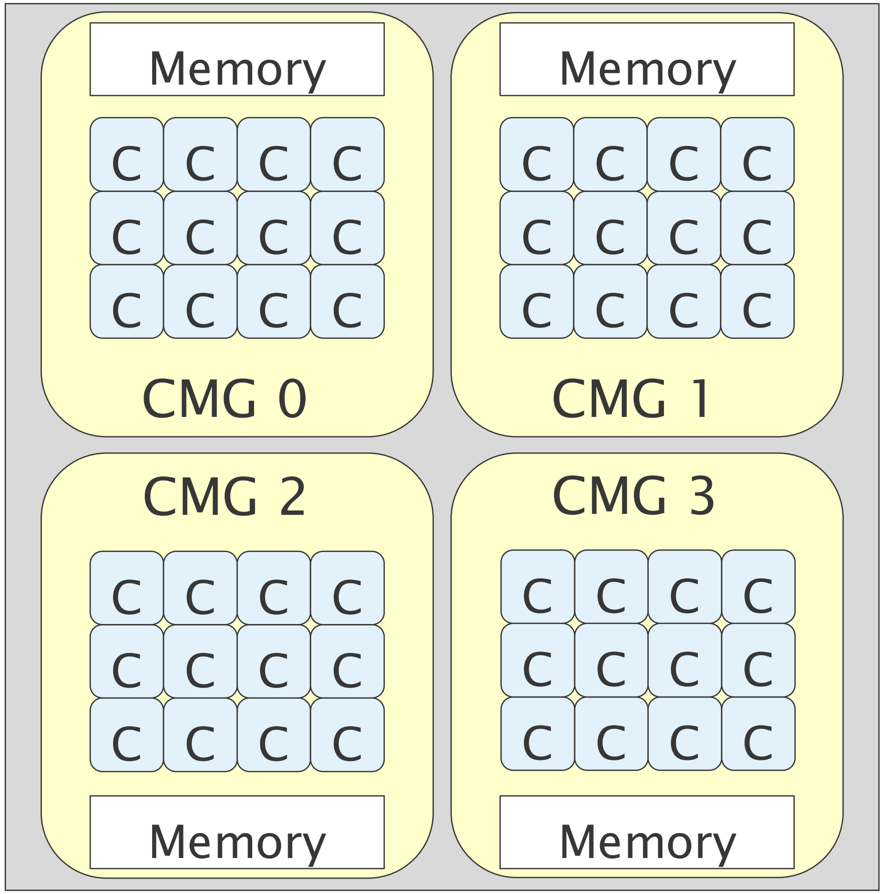
Submit
RIKEN Center for Computational Science
Open Composer version: 1.4.0

- Introduction
- Access to Fugaku
- Compilers
- Mathematical Libraries
- Environment Module Packages
- Spack
- Scripting Languages
- Job Submission
- MPI Job Submissions and Precautions
- Checking the Resource Usage
- LLIO
- Advanced Usage

A64FX Architecture

The specs of A64FX architecture of Fugaku are shown below.

To perform parallel computations, it is necessary to determine the computational condition considering these specs.



∗ CMG: Core Memory Group

- Node Configuration
  - 4 CMG/node
  - 12 cores/CMG
  - The limit of memory size that can be used stably: 23.2 GiB/node
- Maximum number of threads = 48
- Recommended numbers of processes and threads

```
#PJM --mpi max-proc-per-node=4
           # Recommended number of
                  processes per node= 4
export OMP_NUM_THREADS=12
              # 12 threads per process
```

MPI Program Execution Command

In Supercomputer Fugaku, the mpiexec command is used to run MPI programs on compute nodes.

- Running MPI programs using 12 processes

```
$ mpiexec -n 12 -std-proc outfile ./a.out
```

- Running Java programs using 12 processes

```
$ mpiexec -n 12 -std-proc outfile java -classpath ./dir JavaTest
```

- Running multiple MPI programs together
  - Use an MPMD (Multiple Program Multiple Data stream)

```
$ mpiexec -std-proc outfile -n 2 ./a.out : -n 4 ./b.out : -n 6 ./c.out
```

- Use the –app option to enable specification by execution definition file format.
- The total number of processes given by –n should be equal to or less than `--mpi proc=`*N*.
- In Java, execution using an MPMD model is not guaranteed.

Stdin/Stdout/Stderr on MPI Programs

The mpiexec command has options to specify file names for the standard input, standard output, and standard error output files.

- Input the standard input of the parallel process from a specified file

```
$ mpiexec -n 12 -stdin in_file ./a.out
```

- Output the standard output of each parallel process to a separate file

```
$ mpiexec -n 12 {-stdout-proc | -ofout-proc} stdout_file ./a.out
```

- Output the standard error output of each parallel process to a separate file

```
$ mpiexec -n 12 {-stderr-proc | -oferr-proc} stderr_file ./a.out
```

  - The actual name of the output file is, for example, stdout_file.1.0 for the 0th rank output of the 1st mpi execution in the job.
  - The options to output the standard [error] output from all parallel process to single files, `-std`, `-stdout`, and `-stderr`, are now disabled.
  - Redirection is not available in `mpiexec` command in computer nodes.

Specification of the Output Directory

When running a large-scale job, it is required to separate the output directories by using metacharacters to reduce the load of the file system.

- How to change the output directory of standard output and standard error output for each 1000 ranks
  - Directories are generated by default or depending on the below specification

```
$ mpiexec -std-proc ./output.%j/%/1000R/stdout_err ./a.out
```

- Available metacharacters used in the output file specification

Metacharacters	Meaning
%n	Job name
%j, %J, %b, %s	Job ID, sub job ID, bulk number, step number
%R	Rank number If the output is separated for each mpiexec command, this is replaced with empty.
%/***NR***	Rounds a rank number down to the nearest N. In the example above, *N*=500 is specified.

Precautions for File System Usage

Follow these precautions to avoid slowing down the file system and bringing down of the compute nodes.

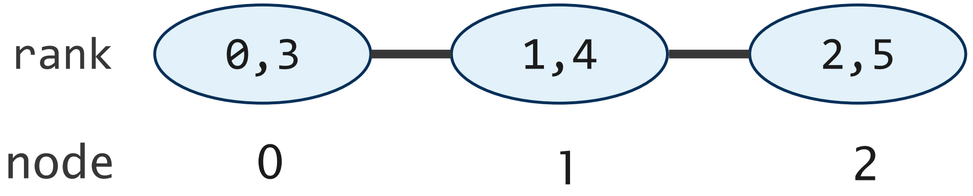
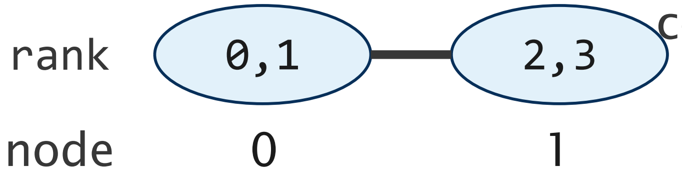
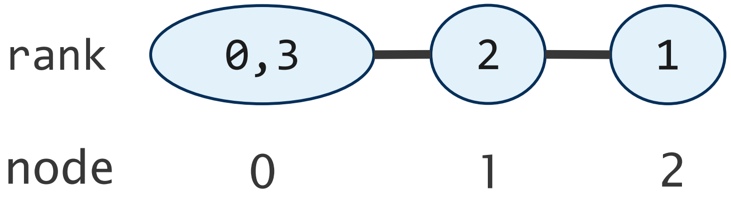
- Do not run more than 1,000 small jobs in the same directory, and do not create/delete files and directories within those jobs.
  - To prevent accessing the same directory simultaneously, it is better to prepare an output directory for each job before submitting jobs.
- It is necessary to avoid accessing a single file from multiple processes simultaneously.
- The upper limit of the number of files and directories that can be generated under a single directory is 100,000.

- If you do not follow the precautions above,
  - the load on the file system is very high
  - the system may forcibly suspend the job IO
- The file system usage rules are subject to the change depending on the computational environment. Check the user briefing for the latest change.

Rank Assignment (Single Dimension)

The rule of rank assignment can be specified using the `--mpi` option of the `pjsub` command.

- The rules for assigning nodes for the ranks are as follows:

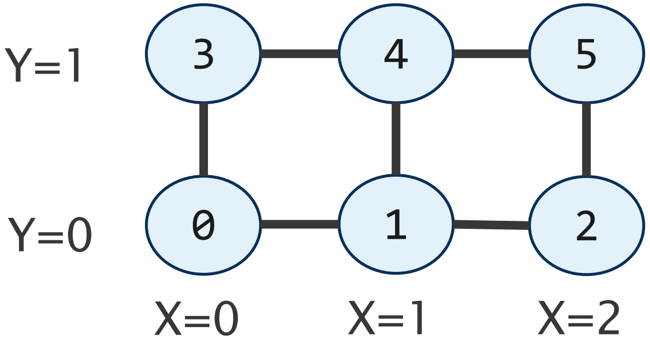
Option	Rank assignment method
`rank-map-bynode`	Assign different nodes in sequence for consecutive ranks 
`rank-map-bychip` (default setting)	Assign nodes in sequence from the same node for consecutive ranks per *n* ranks  $n = \dfrac{\text{\# of processes}}{\text{\# of nodes}}$
`rank-map-hostfile`	Assign nodes based on *hostfile*  Example of a *hostfile* `(0)` # rank 0 `(2)` # rank 1 `(1)` # rank 2 `(0)` # rank 3

Rank Assignment（Two Dimensions）

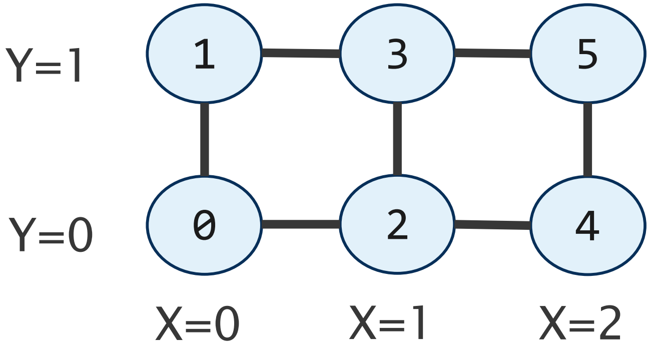
When the shape of the process is two or three dimensions, the rule for rank assignment (*rankmap*) can be also specified.

- rank-map-bynode

```
#PJM --mpi "shape=3x2"
#PJM --mpi rank-map-bynode=XY
```

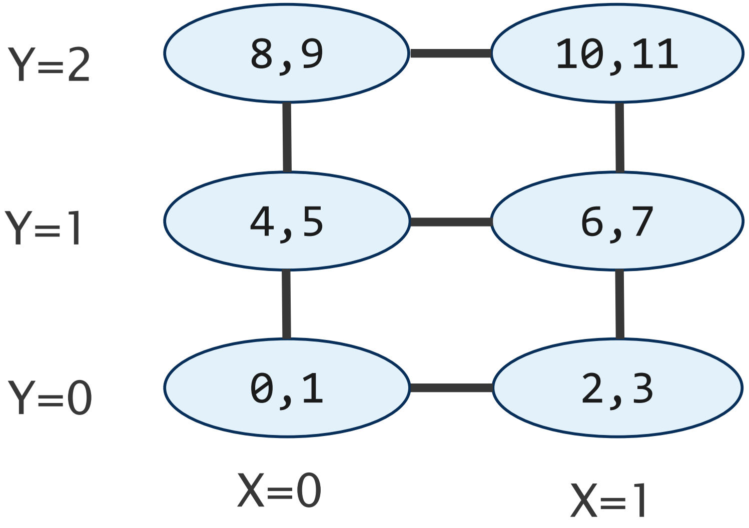


```
#PJM -L "node=3x2"
#PJM --mpi rank-map-bynode=YX
```



- rank-map-bychip

```
#PJM --mpi "proc=12"
#PJM --mpi "shape=2x3"
#PJM --mpi rank-map-bychip:XY
```



- The default setting of *rankmap* is XY (2D) / XYZ (3D)

- Introduction
- Access to Fugaku
- Compilers
- Mathematical Libraries
- Environment Module Packages
- Spack
- Scripting Languages
- Job Submission
- MPI Job Submissions and Precautions
- Checking the Resource Usage
- LLIO
- Advanced Usage

The accountj command displays the used node-time product as well as the electricity used by the project group and by each user.

- Example: display the usage data of your group

```
[_LNlogin]$ accountj
COLLECTDATE : 2022-03-29 16:59:10   unit[Ms,MWh]
*-----[ SUBTHEME ]---------------------------------------------------------*
SUBTHEME                PARENT      LIMIT(N)      USAGE(N)      LIMIT(E)      USAGE(E)
rist                  Y21-RIST        11,668        11,644     unlimited           228
*-----[ SUBTHEME_PERIOD ]--------------------------------------------------*
                        PERIOD      LIMIT(N)      USAGE(N)      LIMIT(E)      USAGE(E)
rist                         1         5,188           720           ---             0
rist                         2         6,480         6,456           ---             0
*-----[ GROUP ]------------------------------------------------------------*
GROUP                   PARENT      LIMIT(N)      USAGE(N)      LIMIT(E)      USAGE(E)
rist                      rist     unlimited         7,176     unlimited           228
```

Allocated recourses

Used resources

Consumed power

- Use the “-g <group name>” option to display only the status of the specified group.
- Use the “-r n” option to display only the node-time product.
  - The unit of the node-time product can be changed using the option –s (sec) or –h (hour).

Job Execution History

The pjstata command displays the job execution history.

- Example: Display the information of the jobs that were started between 26Apr2021 and 28Apr2021

```
[_LNlogin]$ pjstata -d 20210426:20210428
JOBID    SNO BLKNO GENNO JOB_NAME  MD JTYPE USER     GROUP    ST   RSC_UNT  RSC_GRP EC   PC
367430               0     g11F2DB.s NM BT    user1    group1   EXT  unit1    small   0    0
369249               0     parmer8K1 NM BT    user1    group1   EXT  unit1    small   0    0
```

Follow below

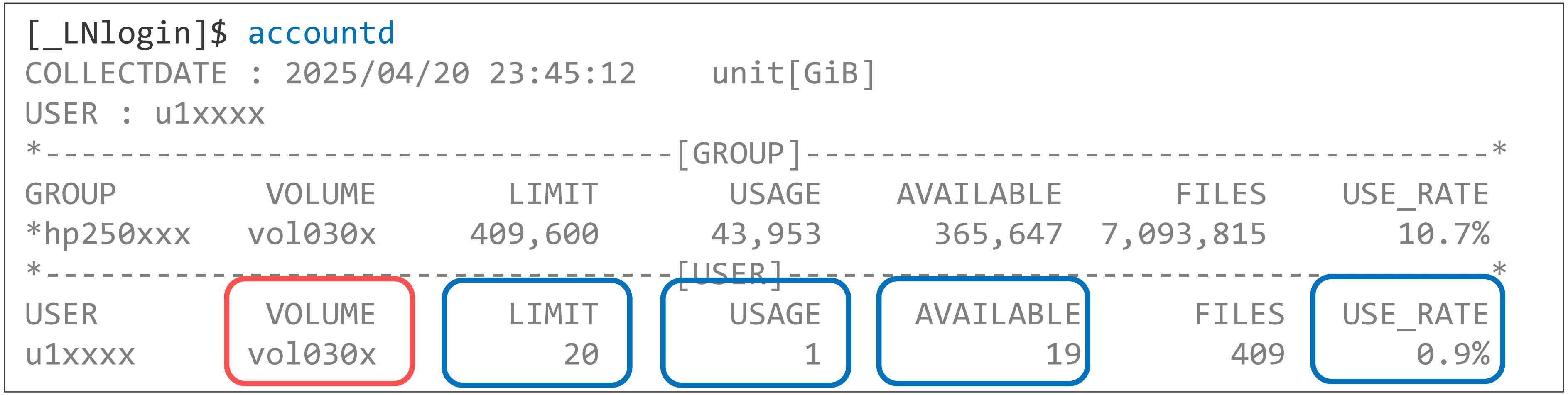
```
ERR_CD  JOB_START       JOB_END         ELAPSE_TIM  NODE_NUM  RATE  ACCT_RSC  ACCT_ECON  PERIOD_NUM
0       04/26 16:12:17  04/26 16:23:32  0000:11:15  1         100%  675       7.518036   2
0       04/27 13:18:41  04/27 13:20:05  0000:01:24  2         100%  168       1.815250   2
```

- The option "-d" is used to filter jobs by job start dates.
- The option "-t" is used to filter jobs by job end dates.
- The RSC_GRP column shows the resource group job executed.
- The ACCT_RSC column shows the node-time product in node seconds.

Disk Usage Status

The accountd command displays the disk usage status of the group and the user separately.

- Example: Check the disk usage status of the data and share areas of the user’s group and home area of the user.



```
[_LNlogin]$ accountd
COLLECTDATE : 2025/04/20 23:45:12    unit[GiB]
USER : u1xxxx
*----------------------------------[GROUP]-------------------------------------*
GROUP          VOLUME       LIMIT       USAGE     AVAILABLE       FILES     USE_RATE
*hp250xxx    vol030x     409,600      43,953      365,647   7,093,815        10.7%
*----------------------------------[USER]--------------------------------------*
USER           VOLUME       LIMIT       USAGE     AVAILABLE       FILES     USE_RATE
u1xxxx       vol030x          20           1           19         409         0.9%
```

disk size allocated

disk size used

disk size available

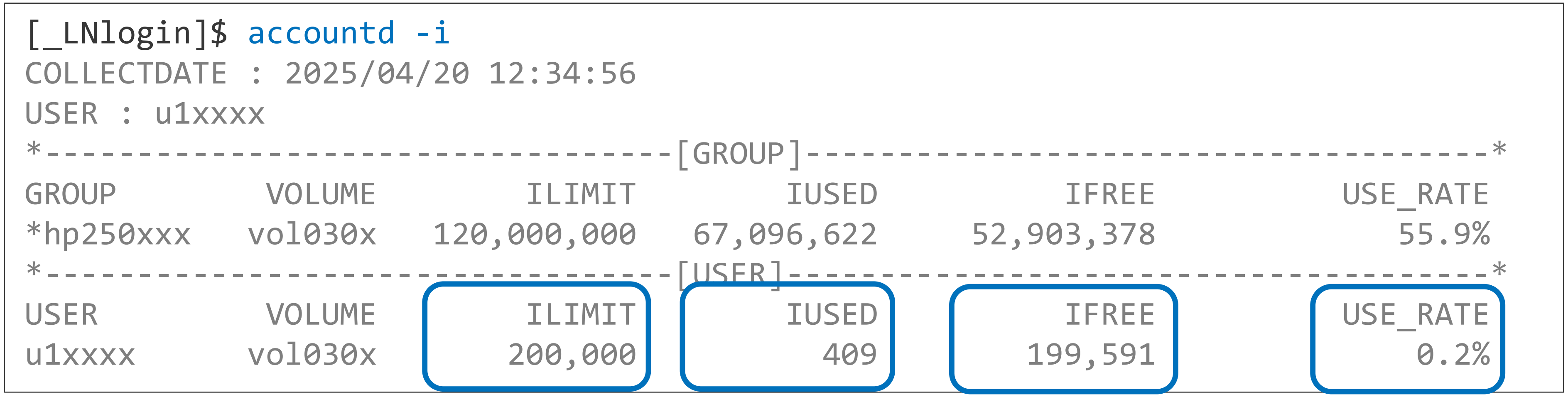
disk usage rate

- First two numbers after vol in the VOLUME column represents the volume number.
  Example: vol0300 ⇒ /vol0003
- Use the -E option to display the access paths of the data as well as the share and home areas.

Disk Usage Status

When the –i option of the accountd command is specified, the i-node usage status of the group and user are displayed separately.

- Example: Check the i-node usage status of the data and share areas of the user's group and home area of the user.



```
[_LNlogin]$ accountd -i
COLLECTDATE : 2025/04/20 12:34:56
USER : u1xxxx
*---------------------------------[GROUP]------------------------------------*
GROUP          VOLUME         ILIMIT          IUSED           IFREE         USE_RATE
*hp250xxx     vol030x    120,000,000     67,096,622      52,903,378            55.9%
*---------------------------------[USER]-------------------------------------*
USER           VOLUME         ILIMIT          IUSED           IFREE         USE_RATE
u1xxxx        vol030x        200,000            409         199,591             0.2%
```

i-node allocated

i-node used

i-node available

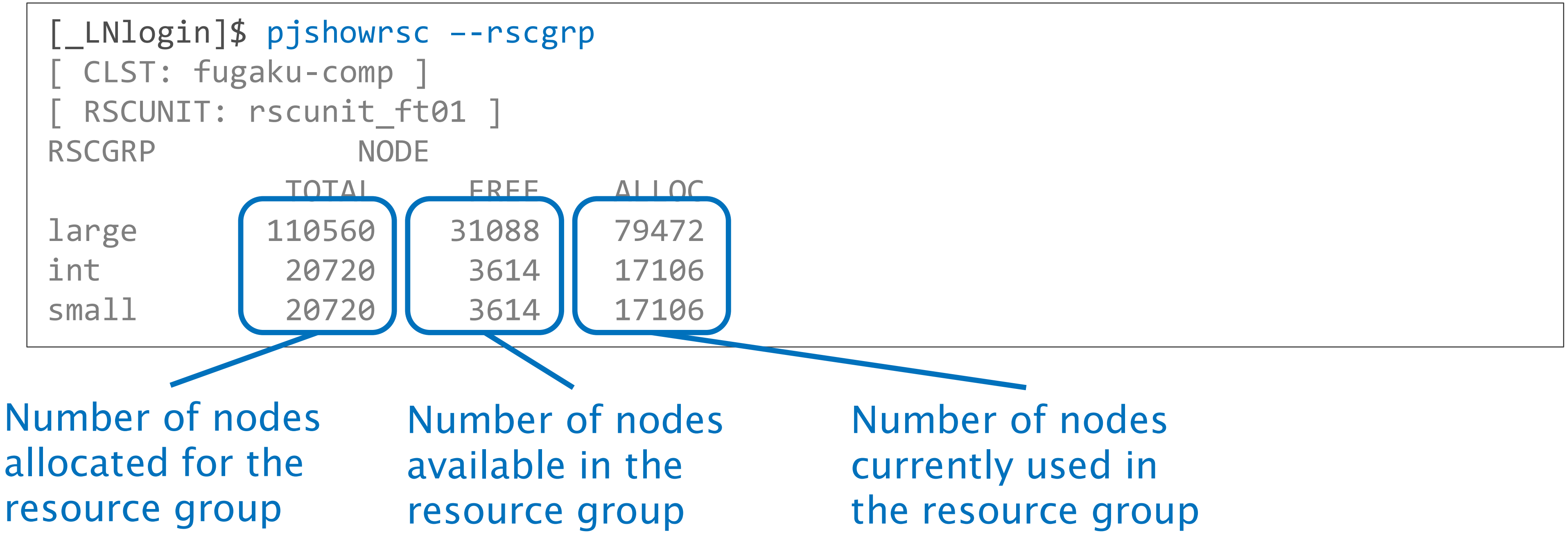
i-node usage rate

- Files in hidden directories like ".local" in your home area are also counted in the i-node usage.
- The upper limit of the number of files and directories allowed in a single directory is 100,000. The tar command is useful for packing many files into one file.

Resource Group Use Status

The pjshowrsc command displays the resource group use status (congestion situation) of all computer nodes.

- Example: Display the use status of all resource groups



```
[_LNlogin]$ pjshowrsc --rscgrp
[ CLST: fugaku-comp ]
[ RSCUNIT: rscunit_ft01 ]
RSCGRP           NODE
               TOTAL     FREE    ALLOC
large         110560    31088    79472
int            20720     3614    17106
small          20720     3614    17106
```

Number of nodes allocated for the resource group

Number of nodes available in the resource group

Number of nodes currently used in the resource group

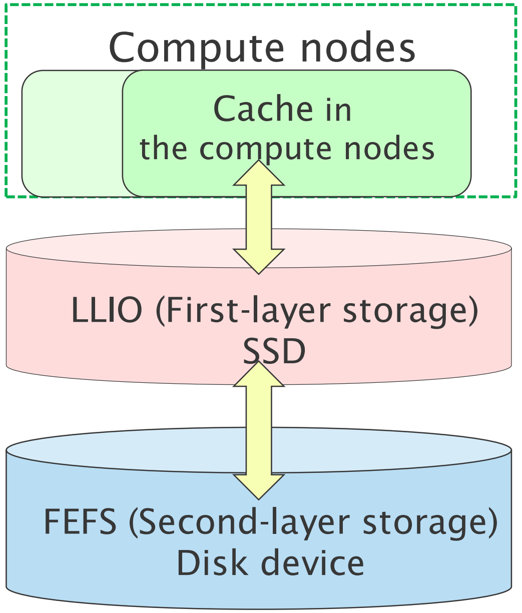
- If some resource groups share same compute nodes, the use status of the nodes shown in ALLOC represents the status of the nodes for both resource groups.
- Single account users also need to specify the group name or ID using the -g option.

- Introduction
- Access to Fugaku
- Compilers
- Mathematical Libraries
- Environment Module Packages
- Spack
- Scripting Languages
- Job Submission
- MPI Job Submissions and Precautions
- Checking the Resource Usage
- LLIO
- Advanced Usage

Overview

The LLIO (Lightweight Layered IO-Accelerator) is a technology that realizes a high-performance file system using SSD. It is located between the FEFS parallel distributed file system and the compute nodes.

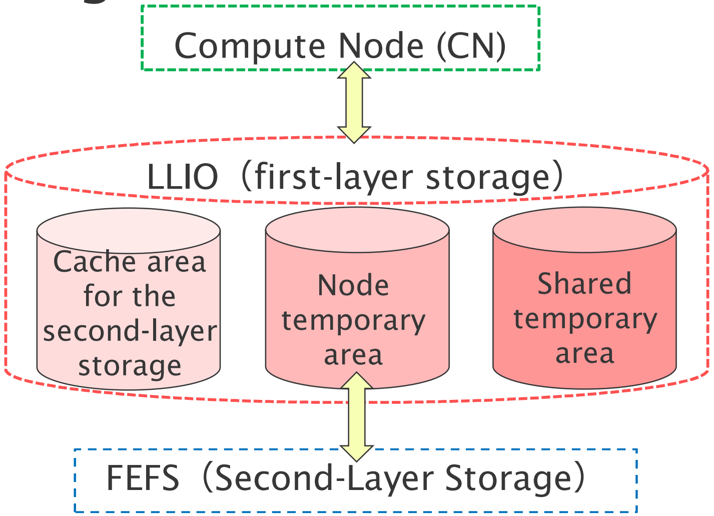
- Storage configuration of Fugaku
  - First-layer storage
    - SSD storage for high-speed access for jobs, managed by LLIO
  - Second-layer storage
    - Disk storage for data storage, managed by FEFS
  - Third-layer storage (not described here)
- LLIO features
  - The SSD storage of which area is secured temporarily during a job execution, which can be used to store temporary files.
  - The asynchronous close option is also available, which allows to write data to 2nd-layer storage asynchronously during other calculation process.


Compute nodes
Cache in the compute nodes
LLIO (First-layer storage) SSD
FEFS (Second-layer storage) Disk device

Three Areas of the First-Layer Storage

LLIO offers three areas in the first-layer storage.

- Cache area for second-layer storage
- Node temporary area
- Shared temporary area


Compute Node (CN)
LLIO（first-layer storage）
Cache area for the second-layer storage
Node temporary area
Shared temporary area
FEFS（Second-Layer Storage）

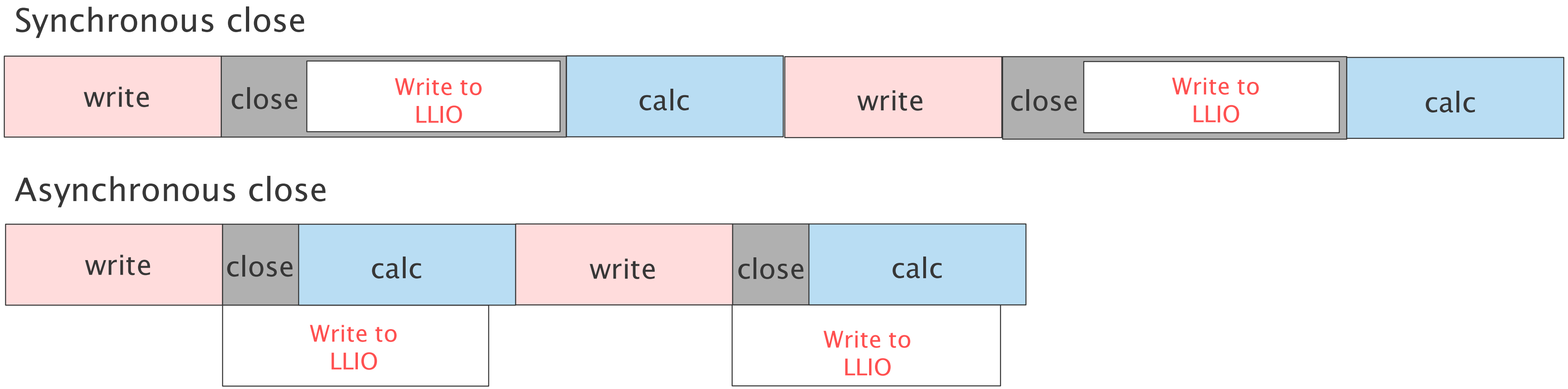
- Features

Area name	Ref. range	Application example	Max size	Area size per node
Node temporary area	Only inside individual compute nodes	Stores intermediate and temporary files independently for each rank and node	87 GiB /node for all areas	Specified by `pjsub --llio localtmp-size` 0 MiB if omitted
Shared temporary area	Overall compute nodes for the job	Stores files referred from other nodes and ranks, manipulates large files		Specified by pjsub `--llio sharedtmp-size` 0 MiB if omitted
Cache area for second-layer storage	Overall compute nodes for the job	Output files such as stdout or stderr saved to second-layer storage		87GiB – (localtmp-size + sharedtmp-size) at least 128 MiB

Better I/O perform-ance

Asynchronous Close Function

This section explains the asynchronous close function provided by LLIO on Fugaku.



- Synchronous close
  - At the end of the close, writing to LLIO and the second-layer storage is guaranteed.
- Asynchronous close
  - At the end of the close, writing to LLIO and the second-layer storage is NOT guaranteed.
  - At the end of the job, writing to LLIO and the second-layer storage is guaranteed unless it exceeds the elapsed time limit.

Asynchronous Close Function

- Activation and deactivation of asynchronous close

```
$ pjsub --llio async-close=on sample.sh
```

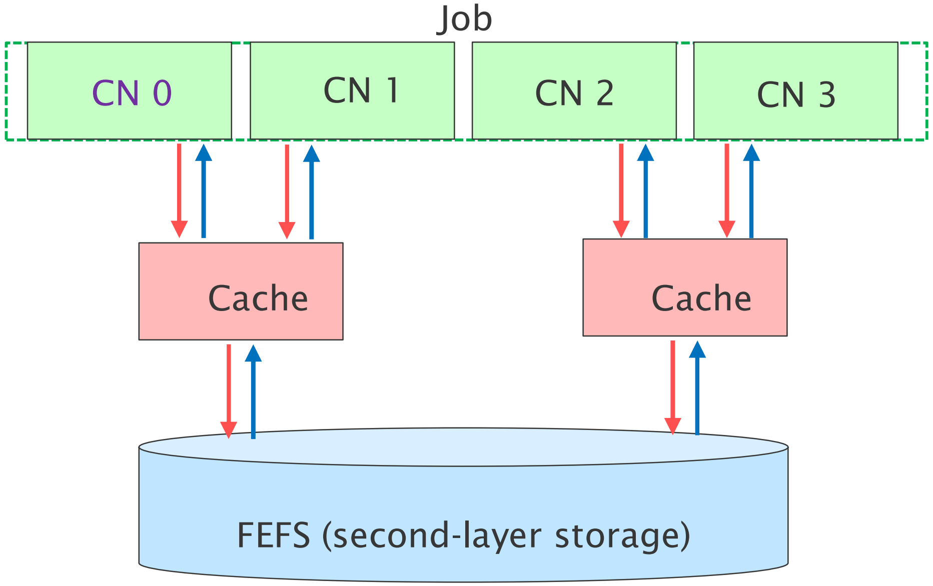
  - `--llio async-close=on` asynchronous close
  - `--llio async-close=off` synchronous close (default)

- Note
  - If the compute node goes down or the job does not end within its elapsed time limit, the transfer from the cache to the second-layer storage will be interrupted. Writing from the cache to second-layer storage is not guaranteed.
  - A list of files that failed to be written to second-layer storage can be obtained by specifying the pjsub option `--llio uncompleted-fileinfo-path`.

Cache Area for Second-Layer Storage

An overview and usage of the cache area for second-layer storage is described below.

- Features
  - In a job, the data of second-layer storage are cached on the first-layer storage.
  - Enabling asynchronous close allows for faster writing from compute nodes to second-layer storage during calculation processing.
  - This area is referred to by all compute nodes assigned to the job.
- Usage
  - Path: /vol0x0y/data/<groupname>
    - Can be used the same way the second-layer storage is used
  - The I/O from a compute node to second-layer storage is done across the cache area of the second-layer storage.
    - To directly access second-layer storage, the 2ndfs area (Path: /2ndfs/<groupname>) is available.



Cache Area for Second-Layer Storage

- Lifetime
  - Until job termination (including error termination) or the job is deleted.
  - In the following cases, the cache in the cache area is deleted:
    - when the files on the second-layer storage are deleted;
    - when direct I/O is performed;
    - when data are removed from the cache area because the area's disk is full.
  - If you cancel a job using the `--llio-flush` option of the `pjsub` command, the job does not stop until the writing to output files is complete within the elapsed time of the job.
- Common file distribution function
  - Using the `llio_transfer` command, files accessed by multiple nodes can be distributed as common files to the cache area of second-storage area.
    - It is recommended to use this command for executable and input files.
  - Example of distributing `a.out` as a common file:

```
$ llio_transfer ./a.out
```

  - Example of deleting `a.out` from the cache area:

```
$ llio_transfer --purge ./a.out
```

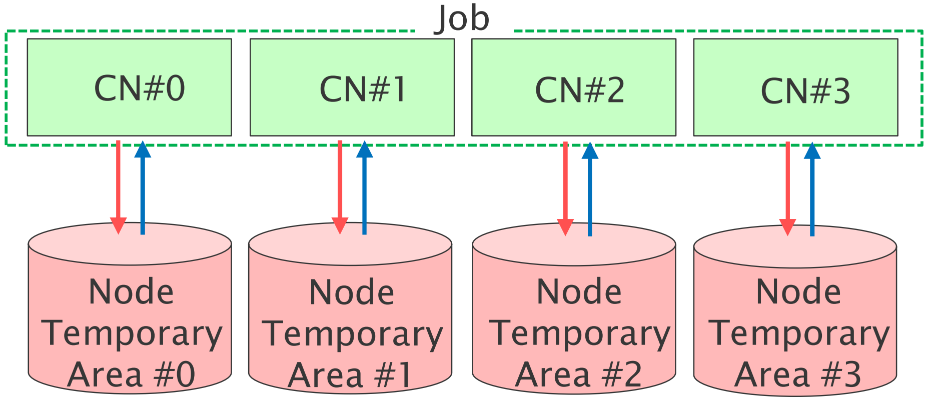
Cache Area for Second-Layer Storage

- Note about the common file distribution function
  - The `llio_transfer` command is available for read-only files.
  - The number of files that can be transferred is ≦ 16,384, and the number of common files each computer node in a job can open at the same time is ≦ 1,024.
  - Deleting a common file that becomes unnecessary during a job script using `llio_transfer --purge` makes room in the cache area.
  - Do not change or delete the original common file in the second-layer storage while the job is running.
    - If the original file on second-layer storage is changed, the contents of the common files copied to the cache of second-layer storage may become undefined.
    - If the original file is deleted from second-layer storage, the corresponding common file becomes unable to be deleted using `llio_transfer --purge` until the job terminates.
  - File locking operations for common files are not supported.
  - Deleting or updating the original file may fail with an error immediately after deleting common files with `llio_transfer --purge`.
  - Do not open the target file for the `llio_transfer` command before its execution. Otherwise, it could cause an error in `llio_transfer` because the cache of the file could be generated while it is opened.

Node Temporary Area

An overview and the usage of the node temporary area of LLIO are described below.

- Feature
  - The node temporary area provides a temporary area locally accessible within individual compute nodes allocated to the job.
- Usage
  - It is necessary to specify the area size using the `--llio localtmp-size` option of the `pjsub` command.
  - The path of the node temporary area is referred to using the environment variable `$PJM_LOCALTMP` in a job.



- Lifetime
  - Initialized before the job starts and deleted after the job closes.

Node Temporary Area

- Usage example
  - Example of the pjsub command option

```
[_LNlogin]$ pjsub --llio localtmp-size=10Gi jobscript.sh
```

- Simple example of a job using the node temporary area:
  - The result of prog1 is saved temporarily in the node temporary area.
  - The output of prog1 is used as the input of prog2. The result of prog2 is output to the node temporary area.
  - The result in the node temporary area is copied to the second-layer storage.

```
$ prog1 -o ${PJM_LOCALTMP}/out.data
$ prog2 -i ${PJM_LOCALTMP}/out.data -o ${PJM_LOCALTMP}/result.data
$ cp ${PJM_LOCALTMP}/result.data ${PJM_JOBDIR}/result_${PJM_JOBID}.data
```

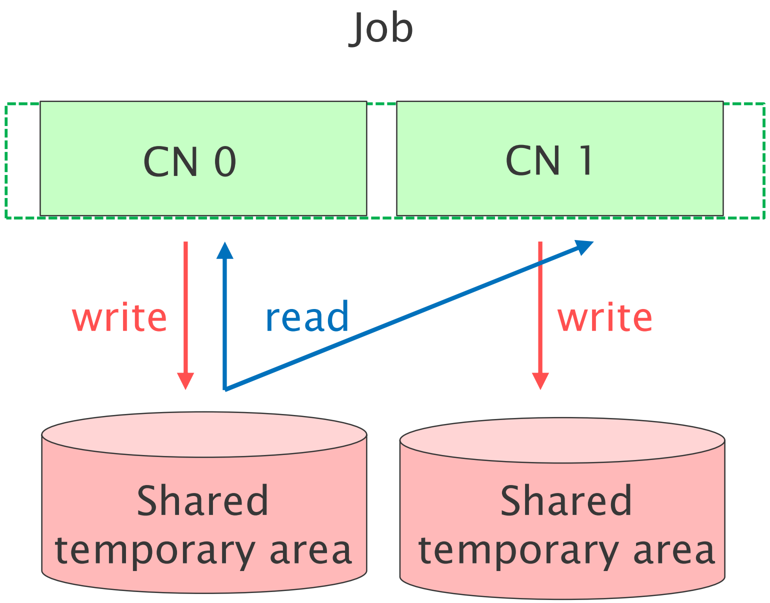
- Example of copying files in the second-layer storage to the node temporary area
  - Copying is done using one process in the compute node.

```
$ mpiexec sh -c 'if [ ${PLE_RANK_ON_NODE} == 0 ]; then cp -rf ./data/ ${PJM_LOCALTMP} ; fi'
```

Shared Temporary Area

An overview and the usage of the shared temporary area of LLIO are described below.

- Feature
  - The shared temporary area provides a temporary area accessible from all compute nodes allocated to the job
- Usage
  - It is necessary to specify the area size per node using the `--llio sharedtmp-size` option of the `pjsub` command.
  - The path of the node temporary area is referred using the environment variable `$PJM_SHAREDTMP` in a job



- Lifetime
  - Initialized before the job starts and deleted after the job closes.

Shared Temporary Area

- Usage example
  - Example of the `pjsub` command option
    - The total area size of the shared temporary area is the number of nodes times that specified by the option `--llio sharedtmp-size`.

```
[_LNlogin]$ pjsub --llio sharedtmp-size=10Gi jobscript.sh
```

  - Simple example of job using shared temporary area:
    - The result of prog1 is saved temporarily in the shared temporary area.
    - The output of prog1 is used as the input of prog2. The result of prog2 is output to the shared temporary area.
    - The result in the shared temporary area is copied to the second-layer storage.

```
$ prog1 -o ${PJM_SHAREDTMP}/out.data
$ prog2 -i ${PJM_SHAREDTMP}/out.data -o ${PJM_SHAREDTMP}/result.data
$ cp ${PJM_SHAREDTMP}/result.data ${PJM_JOBDIR}/result_${PJM_JOBID}.data
```

Note for the Selection of Temporary Areas

The following is information helpful for area selection

- Comparison of node and shared temporary areas
  - Characteristics of the node temporary area
    - The area size must be less than 87 GiB.
    - Meta-data operations are free from interference from other jobs.
  - Characteristics of the shared temporary area
    - The area size depends on the number of compute nodes allocated to the job. Thus, large files can be handled in this area.
    - Other jobs interfere with meta-data operations.
  - It is recommended to use the node temporary area if your data can be stored in its disk space.
  - When you use the shared temporary area, do not access single files from multiple processes at the same time.
- Area selection when using MPI-IO
  - MPI-IO works with both the shared temporary area and cache area of the second-layer storage.
  - Using the shared temporary area yields better performance.

Restrictions on Using LLIO

When using LLIO, there are the restriction on the number of files that a job can handle.

- The number of files that a job can open at the same time must be less than 1,024 * (the # of compute nodes used by the job), which is the total of the three areas provided by LLIO.
- The number of files that can be used by a job must be less than 16,384 * (the # of compute nodes used by the job), which is the total of the shared temporary area and the cache area of second-layer storage.
    - There is no restriction on the node temporary area, and up to 10 million files can be used per compute node.
- When one file is accessed from multiple processes, I/O will slow unless both of the following conditions are fulfilled.
    - # of nodes where processes using the same file exist is < 7,000.
    - The total number of processes that use the same file is < 28,000.
- The stripe count is set to 24 in the default settings.

I/O Profiling

It is possible to output the LLIO performance information when submitting a job with `--llio perf` option of `pjsub` command.

- How to use

```
[_LNlogin]$ pjsub --llio perf jobscript.sh
```

- Output file of LLIO performance information

```
(jobname).(jobid).llio_perf
```

- Refer to *User's Guide - Use and Job Execution, Section 8.7* for details.

I/O Profiling

Darshan, a scalable HPC I/O characterization tool, is provided by Spack and can be used to investigate I/O behaviour of a job.

Getting I/O profiling data

```
[_CNlogin]$ spack load darshan-runtime scheduler=fj
```

- Refer to *User's Guide - Use and Job Execution, Section 8.7* for a job script example.
- Use /2ndfs area as the data output destination.
  - Because the profiling data is accessed by all compute nodes, the nodes will be slowed down due to the LLIO limitation if the job uses thousands nodes and more and the profiling data is output on the data area.

Analyzing the I/O profiling data

```
[_LNlogin]$ spack load darshan-util@3.4.0 arch=linux-rhel8-cascadelake
[_LNlogin]$ darshan-parser --file-list usr_a.out_(JOBID).darshan
```

- Refer to *User's Guide - Use and Job Execution, Section 8.8* for how to utilize the I/O profiling data.

- Introduction
- Access to Fugaku
- Compilers
- Mathematical Libraries
- Environment Module Packages
- Spack
- Scripting Languages
- Job Submission
- MPI Job Submissions and Precautions
- Checking the Resource Usage
- LLIO
- Advanced Usage

Large Page Function

The large page function extends the page size (unit of data for memory management), reducing the cost of OS address translation processing and improving memory access performance.

- Comparison of page sizes

Normal page in Fugaku	Large page in Fugaku
64 KiB	2 MiB

  - In Fugaku, large pages are used by default when the Fujitsu compiler is used.
  - A side-effect of using large pages is that 2-3 times the amount of memory is consumed in the static data areas.

- Advantages and disadvantages

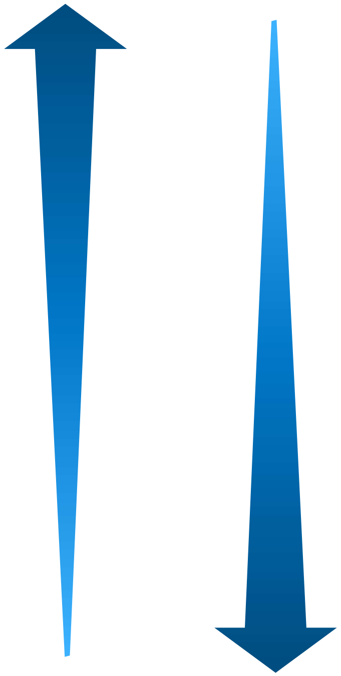
	Normal page	Large page
Memory initialization cost /memory use efficiency	Low cost/high efficiency	High cost/low efficiency
# of pages accessed/TLB miss rate	Many/high	Small/low
Suitable application type	Low memory consumption	Memory hogging

Profiler

Fujitsu compiler provides the three profilers listed below.

- Profiler types
  - Instance performance profiler
    - measures profile data using the `fipp` command
    - cost information for each procedure, loop, or line, cost balance information between processes and between threads
  - Advance performance profiler
    - measures profile data using the `fapp` command
    - Performance and execution time information of the specified region, MPI communication cost information
  - CPU performance analysis report
    - Requires multiple measurements using `fapp`
    - Information about the busy rate, cache misses, and floating operations
      - Useful for checking program performance on the A64FX

Lower cost



More information

For more detail, refer to the *Supercomputer Fugaku User’s Guide - Language and Development Environment* or *Profiler User’s Guide*

Materials Helpful for Advanced Usage

- CPU performance tuning based on the type of application:
  https://www.fugaku.r-ccs.riken.jp/doc_root/en/programming_guides/app-tuning-pattern-English.pdf

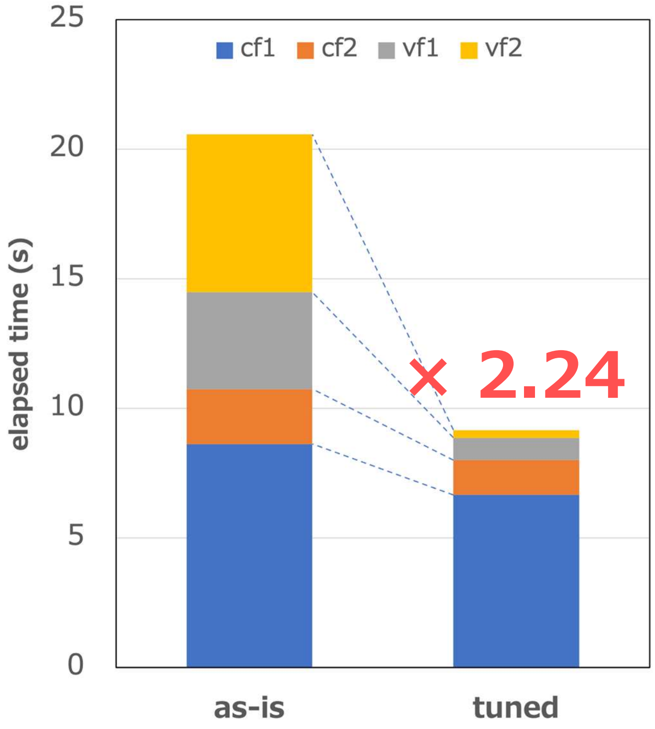
- Seminar on Fugaku Users (Intermediate courses, Hands-on)
  - Intermediate Course (Optimization Techniques of CPU Performance: SIMD, Software piplining, etc.)
  - Intermediate Course (MPI/LLIO)
  - Intermediate Course (Optimization Techniques of CPU Performance: Efficient use of operations, Cache tuning, etc.)
  - Fugaku in Practice (Hands-on)
  - ➢ These seminar materials are available from "Workshop materials" in Fugaku Website.

- R-CCS/RIST Joint Seminar on Advanced use of Supercomputer Fugaku and Arm computer systems (held in English)
  https://www.hpci-office.jp/en/events/seminars#FugakuAndArm

Introduction to Program Tuning Support

RIST provides tuning support such as porting, serial, and scalability optimization for HPCI users available free of charge.

- Target
  - The project categories of General/Junior Researches, Industrial Access and Special call
    https://www.hpci-office.jp/en/using_hpci/project_categories_overview
  - Research promotion projects of Fugaku
- Support content
  - Porting
    - Validation of the performance on each actual HPCI resource
  - Serial and scalability optimization
    - Performance analysis (performance check of a program, hotspot/load balance check, and parallel performance check)
    - Proposals for measurements to improve performance
- How to apply
  - Please check the HPCI website:
    https://www.hpci-office.jp/en/user_support/tuning_support



This graph shows the speedup of FFVHC-ACE's kernel program. Source: https://www.hpci-office.jp/pages/e_meeting_A64FX_210427/#topic-2 by H. Kobayashi, H. Sawai, and E. Tomiyama, RIST.

Your application is welcome!