HOKUSAI BigWaterfall2 User's Guide

Version 1.9

Apr. 2, 2024

Information Systems Division,

RIKEN

Revision History

Ver.	Issued on	Chapter	Reason for revision
1.0	Dec.4,2023	-	First edition
1.1	Dec.7,2023	1.2	Modified calculation formula for Theoretical peak performance of Massively Parallel Computer and Large Memory Computing Server
1.2	Dec.8,2023	5.7.5	Modified the description and command execution example
1.3	Dec.11,2023	5.6	Modified example job script
1.4	Dec.12,2023	2.2 5.6	Add Network Access Modified example job script
1.5	Dec.19,2023	3.2	Modified “lfs quota”command
1.6	Dec.21,2023	5.6	Modified example job script
1.7	Dec.22,2023	5.2	Modified concurrent resource usage limit
1.8	Dec.27,2023	5.2 5.3	Modified memory unit to GiB notation Modifide Table 5-6
1.9	Apr.2.2024	3.2 4.2 5.2 5.4 5.3.2 5.6.1	Added HPCI project Added compilation/link example for GPU Delete the description of mpc_ht Added description of GPU server job execution resources Added notes when running interactive jobs on GPU Added job submission option Added job script example for GPU server (BWGPU)

Copyright (C) RIKEN, Japan. All rights reserved.

Table of Contents

Copyright (C) RIKEN, Japan. All rights reserved.

INTRODUCTION

This User's Guide explains how to use the supercomputer system HOKUSAI BigWaterfall2 introduced by RIKEN. All users who use this system are strongly recommended to read this document, as this is helpful to gain better understanding of the system.

The content of this document is subject to change as required. The latest version of this document is available from the following User Portal:

https://hokusai.riken.jp/

The User Portal summarizes the available application versions and how to run them. You will also register your ssh public key here. For details, refer to "HOKUSAI BigWaterfall2(HBW2) Web Portal User's Guide ".

The system operation status is announced on the user portal and the user mailing list. Any questions about the HOKUSAI BigWaterfall2 system can be directed to the following email address:

Email: hpc@riken.jp

Unauthorized copy, reproduction, or republication of all or part of this document is prohibited.

Copyright (C) RIKEN, Japan. All rights reserved.

1. HOKUSAI BigWaterfall2 System

1.1 System Overview

HOKUSAI BigWaterfall2 system (HBW) consists of a Massively Parallel Computer (BWMPC), a Large Memory Computing Server (BWLMC), a front-end server that serves as the entry point for the system, and shared storage.

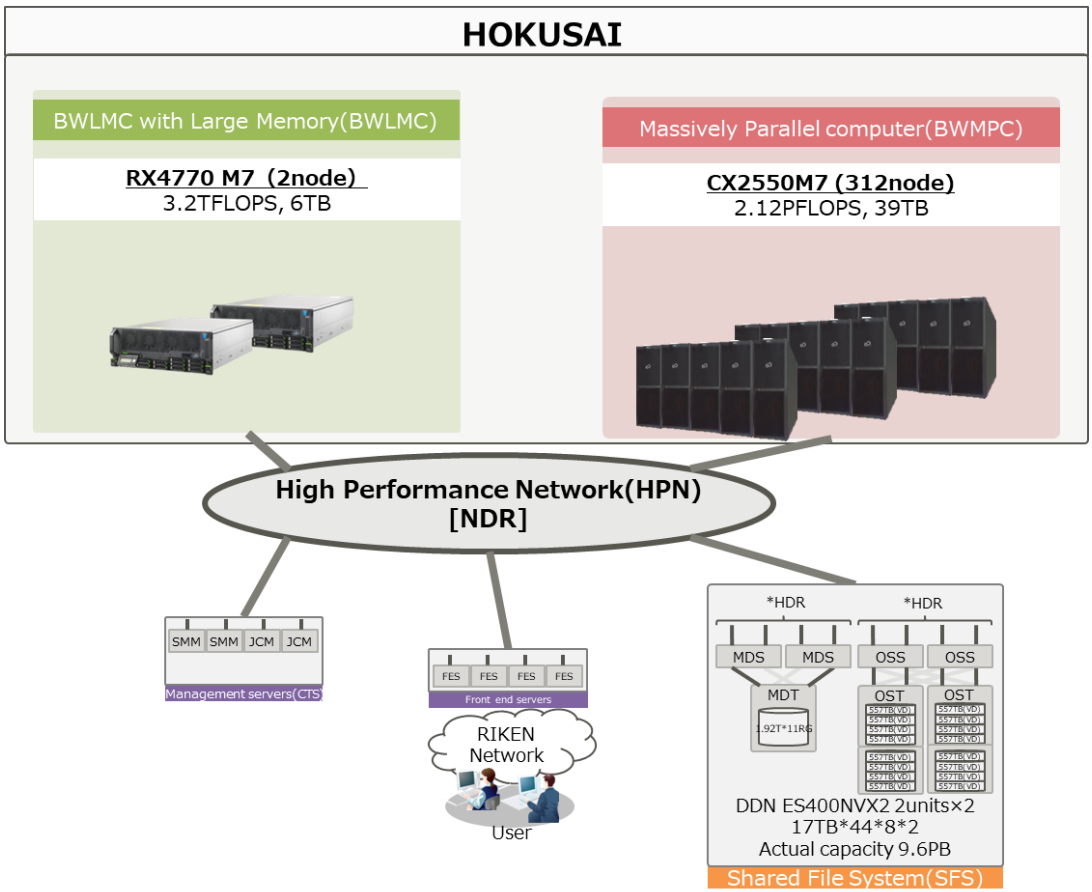

Figure 1 System diagram

The Massively Parallel Computer (BWMPC) comprises 312 nodes of PRIMERGY CX2550 M7. Each node provides a theoretical peak performance of 6.8 TFLOPS and a memory capacity of 128 GB. The InfiniBand NDR of 400 GB/s is used to connect each node to enable high performance communication and file sharing.

The LMC with Large memory (BWLMC) comprises two nodes of PRIMERGY RX4770 M7. Each node provides a theoretical peak performance of 6.4 TFLOPS and a memory capacity of 3TB. The InfiniBand NDR of 400 GB/s is used to connect each node to enable high performance communication and file sharing.

Copyright (C) RIKEN, Japan. All rights reserved.

The storage environment is configured with shared storage (SFS).

Shared storage (SFS) is a broadband, online file system that can be referenced by Massively Parallel Computer (BWMPC), Large Memory Computing Server (BWLMC), and front-end servers, including individual user home directories and shared directories for project groups. Available capacity is 9.8PB.

HBW can be accessed via ssh/scp and HTTPS (user portal). Users can edit, compile/link programs, operate batch jobs, run interactive jobs, tune, debug, and more on the front-end server.

Copyright (C) RIKEN, Japan. All rights reserved.

1.2 Hardware Overview

1.2.1 Massively Parallel Computer (BWMPC)

- Computing performance
  CPU: Intel Xeon Max 9480 (1.9 GHz) 312 units (624 CPUs, 34,944 cores)
  Theoretical peak performance: 2.12 PFLOPS (1.9 GHz x 56 cores x 32 FMA x 2 sockets x 312 nodes)
- Memory
  Memory capacity: 39TB (128 GB x 312 units)
  Memory bandwidth: 3.26TB/s
- Local disk
  Disk capacity: 300TB (480 GB x 2 x 312 units)
- Interconnect
  InfiniBand NDR Communication performance: 400GB/s between nodes

1.2.2 Large Memory Computing Server (BWLMC)

- Computing performance
  CPU: Intel Xeon Gold 6418 H (2.1 GHz) 2units (8 CPUs, 192 cores)
  Theoretical peak performance: 6.1 TFLOPS (2.1 GHz x 24 cores x 4 CPUs x 32 FMA)
- Memory
  Memory capacity: 6TB (3 TB x 2 units)
  Memory bandwidth: 1126.4GB/s
- Local disk
  Disk capacity: 2.4TB (600 GB x 2 units)
- Interconnect
  InfiniBand NDR Communication performance: 400GB/s between nodes

Copyright (C) RIKEN, Japan. All rights reserved.

1.3 Software Overview

The softwares available on HBW2 are listed as follows:

Table 1-1 Software overview

Item	Massively Parallel Computer (BWMPC)	Large Memory Computing Server (BWLMC)	Front End Server
OS	Rocky Linux 8 (x 312) (Linux kernel version 4.18)	Red Hat Enterprise Linux 8 (Linux kernel version 4.18)	Red Hat Enterprise Linux 8 (Linux kernel version 4.18)
Compiler	Intel oneAPI Base & HPC Toolkit Multinode Intel C/C++ and Fortran compilers Intel TBB Intel Distribution for Python		
Library	Intel oneAPI Base & HPC Toolkit Multinode Intel MKL Intel MPI Library Intel IPP Intel DAL		
Tool	Intel oneAPI Base & HPC Toolkit Multinode Intel VTune Intel Advisor Intel Inspector Intel Trace Analyzer & Collector		
Application	Gaussian, ADF, AMBER, GAMESS,GROMACS, NAMD, ROOT	Gaussian, ADF, AMBER, GAMESS, GROMACS, NAMD, ROOT	GaussView, VMD, ROOT

The latest information about the applications, libraries and so on, available for HBW2, will be published in the following User Portal:

https://hokusai.riken.jp/

1.4 Service Time Period

The services of HOKUSAI BigWaterfall2 system are regularly available for 24 hours except for the time periods of the periodical maintenance, the emergency maintenance, and the equipment maintenance such as the air conditioner maintenance and the power-supply facility maintenance. The availability of HOKUSAI BigWaterfall system is announced via the User Portal or the mailing list.

1.5 Usage Policy

For more detailed information, visit the following URL to see the "Supercomputer System Usage Policy".

https://i.riken.jp/supercom/

Copyright (C) RIKEN, Japan. All rights reserved.

1.5.1 Allocated Computational Resources (core time)

Allocated computational core time depends on usage category. You can check the project accounting summary such as project ID/project name, allocated computational core time, and used computational core time, expiration date of allocated computational core time by the listcpu command.

No new jobs can be submitted and executed when remained computational core time runs out.

```
[username@hokusai1 ~]$ listcpu -p projectname
[RB999999]
Node/User       Used(h)     Limit(h)     Used(%)     Expiry date
-----------     ---------   ----------   ---------   -------------
mpc                  0      5115801.6       0.0        2024-03-31
lmc                  0        28108.8       0.0        2024-03-31

userA                0              -         -        2024-03-31
userB                0              -         -        2024-03-31
:
```

Table 1-2 listcpu information

Item	Description
Used (h)	Used computational core time (hour)
Limit (h)	Allocated computational core time (hour)
Use (%)	Used computational core time / Allocated computational core time (%)
Expiry date	Expiration date of the allocated computational core time

Copyright (C) RIKEN, Japan. All rights reserved.

1.6 Job exection order

In this system, the job execution order is decided by the priority of all jobs. The priority is evaluated by the following items.

Table 1-3 Evaluation items for priority control

Evaluation order	Evaluation item	Overview
1	Partition priority	Run from a high priority partition on a partition that shares nodes.
2	Job priority	Run from highest priority per job.
3	Job submittion time	Execute by the submission order.

Because the evaluation results of items with a low Evaluation Order value take precedence, jobs that are submitted to a high-priority partition have a higher priority than jobs that are submitted earlier.

Copyright (C) RIKEN, Japan. All rights reserved.

2. Login to HOKUSAI BigWaterfall2 System

2.1 Login Flow

To log in to HBW, you must apply for an account and register a public key. Please apply for an account or register a public key from the user portal. For details, refer to "HOKUSAI BigWaterfall2(HBW2) Web Portal User's Guide ".

2.2 Network Access

The servers within HBW in which you can access via SSH/HTTPS are the front end servers. The front end server consists of 4 servers. Table 2-1 shows the connection Destination hosts.

The front-end server consists of a total of four servers, and when you access hokusai.riken.jp, you can access any login node using DNS round robin. HBW2 Portal allows SSH public key registration and manual reference through HTTPS access.

Table 2-1 Destination hosts

Host Name	Protocol	Purpose to access
hokusai.riken.jp	ssh	• Virtual terminal • File transfer
hokusai1.riken.jp		
hokusai2.riken.jp		
hokusai3.riken.jp		
hokusai4.riken.jp		
hokusai.riken.jp	https	• Register the SSH public key • View manuals • Various applications

Copyright (C) RIKEN, Japan. All rights reserved.

3. File System Overview

3.1 Storage Area

The following storage areas are available in HBW.

Table 1 Storage areas

Area	Size	Purpose
/home	9.6PB	Home area
/data		Data area (Project units, target of usage fee)
/tmp_work		Temporary area
/bwfefs/home	5PB	Old Home Area (Unwritable)
/bwfefs/data		Old Data Area (Unwritable, New requests not possible)
/arc	Amount of application	Tiered Storage Area (by project, stopping new applications)

Each storage can be accessed from each server as follows.

Table 2 Accessibility of each server to each storage

Storage type	Front end servers	BWMPC	LMC
Shared storage	○	○	○
Hierarchical Storage	○	×	×

○ : available × : unavailable

3.1.1 Shared storage

Shared storage is the user's home and per-project data areas. Home space is allocated at 4TB per user. Data areas are subject to usage charges. If you want to use the data area, please apply for the usage contribution from the user portal.

Table 3 Quota of the Shared Storage

Directory	User Directory	Block quota	Inode quota
/home	/ home/username	4TB	10million
/data	/ data/projectID	1TB~	600,000 per 1 TB

Copyright (C) RIKEN, Japan. All rights reserved.

3.1.2 Hierarchical Storage

The Hierarchical Storage area (tape area) consist of cache storage and tape library devices for long-term storage. The Hierarchical Storage can be accessed from the front-end servers and is available on project units, but is not currently open for new acceptance.

Table 4 Tiered Storage Allocation Limits3

User Directory	Quota	Inode quota
/arc	2 tape volumes/8TB - 13 tape volumes/52TB	40,000 per tape volume

The main purposes of use are as follows:

- Store a large data for a long period
- Backup

Since the Hierarchical Storage stores data in tapes, it is not suitable to store smaller files such as less than 100 MB. If you store many small files, create single file archive and then store.

Copyright (C) RIKEN, Japan. All rights reserved.

3.2 **Disk usage**

You can use the lfs quota command to display your disk usage and quota. To display the usage status of the data area, specify the project ID with the “-p” option. In the case of RIKEN project, the project ID will be the GID of the project name starting with "RB" that was notified at the time of application. In the case of HPCI project, it will be the number obtained by adding “1” to the beginning of the GID of the project name starting with "HP" that was notified at the time of application.

```
[username@hokusai1 ~]$ lfs quota -p ${UID} ${HOME}
Disk quotas for usr username (uid XXXX):
     Filesystem  kbytes   quota   limit   grace   files   quota   limit   grace
 /home/username 8637844       0       0       -   35783       0       0       -
[username@hokusai1 ~] $lfs quota -p projectID /data/project_name
Disk quotas for usr username (uid XXXX):
     Filesystem  kbytes   quota   limit   grace   files   quota   limit   grace
 /data/RB999999 8637844       0  2000000000    -   35783       0       0       -
```

Table 5 lfs quota information

Item	Description
[kbytes]	Block usage (KB), quota (KB) and the ratio
[Files]	Inode usage (K), quota (K) and the ratio

3.3 **Temporary area**

/tmp_work is available to store temporary files.

The data stored under /tmp_work is automatically removed when its modified date becomes older than one week. Use this storage area only for storing temporary files exclusively.

Copyright (C) RIKEN, Japan. All rights reserved.

4. Compile and Link

4.1 Set environment settings

The module command enables you to set environment variables to use compilers, libraries, applications, tools and so on.

```
$ module <subcommand> <subcommand-args>
```

The sub-commands of the module command are the following:

Table 4 Sub-commands of the *module* command

Sub command	Description
avail	List all available settings
list	List loaded settings
load module…	Load setting(s) into the shell environment
unload module…	Remove setting(s) from the shell environment
purge	Unload all loaded settings
switch module1 module2	Switch loaded module1 with modules

Example) List all available settings.

```
[username@hokusai1 ~]$ module avail

------------------------------------------
/lustre/opt/modulefiles/hokusai/apps ------------------------------------
----------
adf/2022.103(default) gamess/2023R2
gaussview/6.0.16      gromacs/2023.3(default)
amber/22(default)     gaussian/16_C.02(default)
gold/2023.2(default)  vmd/1.9.4
-----------------------------------------
/lustre/opt/modulefiles/hokusai/compilers ----------------------------
-------------
boost/1.83.0          fftw-mpi-double/3.3.10
gcc/13.2.0(default)   intelmpi/impi_23.2.0
```

* The version listed with “(default)” is the recommended version on HOKUSAI BigWaterfall system.

Example) Load compiler's setting for the BWMPC.

```
[username@hokusai1 ~]$ module load intel
```

Example) List loaded settings.

```
[username@hokusai1 ~]$ module list
  Currently Loaded Modulefiles:
   1) intelmpi/impi_23.2.0   2) intel/23.02.1
```

Copyright (C) RIKEN, Japan. All rights reserved.

You cannot load the settings which conflict with the loaded settings at once. If you try to load setting, the following error messages are displayed and it failed to set.

```
[username@hokusai1 ~]$ module load openmpi/4.1.6
Loading openmpi/4.1.6
  ERROR: openmpi/4.1.6 cannot be loaded due to a conflict.
  HINT: Might try "module unload intelmpi" first.
```

Example) Cancels the current settings.

```
[username@hokusai1 ~]$ module purge
[username@hokusai1 ~]$ module list
No Modulefiles Currently Loaded.
```

Copyright (C) RIKEN, Japan. All rights reserved.

4.2 Compiler

On the HBW front-end server, a compiler is available to create load modules that can run on Massively Parallel Computer (BWMPC) and Large Memory Computing Servers (BWLMC).

Example) Compiling/linking for Massively Parallel omputer (BWMPC) / Large Memory Computing Server (LMC).

```
[username@hokusai1 ~]$ module load intel
Loading intel/23.02.1
  Loading requirement: intelmpi/impi_23.2.0
[username@hokusai1 ~]$ module list
Currently Loaded Modulefiles:
   1) intelmpi/impi_23.2.0   2) intel/23.02.1
```

A compiler can be used on the GPU server to create load modules that can be executed on the GPU server (GPU).

Example) When compiling/linking for GPU server (GPU).

```
[username@hokusai1 ~]$ srun --partition gpu_i --account RB999999 --pty $SHELL
[username@bwgpu4 ~]$ module load cuda/12.4

[username@bwgpu4 ~]$ module list
Currently Loaded Modulefiles:
 1) cuda/12.4
```

4.3 How to Compile and Link

The commands for compilation and linkage are as follows:

Table 4 Compile and link commands for the Massively Parallel Computer (BWMPC) and the Large Memory Computing Server (BWLMC).

Type	Programming language	Command	Automatic parallelization*1	OpenMP*1
Sequential (no MPI)	Fortran	ifort	-parallel	-qopenmp
	C	icc		
	C++	icpc		
MPI Parallel	Fortran	mpiifort		
	C	mpiicc		
	C++	mpiicpc		

*1: Automatic parallelization and OpenMP options are not set by default.

Copyright (C) RIKEN, Japan. All rights reserved.

4.3.1 Compile and Link for Massively Parallel Computer (BWMPC) / Large Memory Computing Server (LMC).

4.3.1.1 Compile and link sequential programs.

To compile and link sequential programs for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC). on the front-end servers, use the *ifort*/*icc*/*icpc* command.

```
ifort/icc/icpc[option] file [...]
```

Example 1) Compile and link a sequential Fortran program for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC).

```
[username@hokusai1 ~]$ ifort seq.f
```

Example 2) Compile and link a sequential C program for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC).

```
[username@hokusai1 ~]$ icc seq.c
```

4.3.1.2 Compile and link thread parallelization programs

To compile and link multi-threaded programs for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC). on the front-end servers, use the *ifort*/*icc*/*icpc* command.

```
ifort/icc/icpc thread-option [option] file [...]
```

Example 1) Compile and link a Fortran program with automatic parallelization for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC) .

```
[username@hokusai1 ~]$ ifort -parallel para.f
```

Example 2) Compile and link a C program with automatic parallelization for the Massively Parallel Computer (BWMPC) / LarL age Computing Server.

```
[username@hokusai1 ~]$ icc -parallel para.c
```

Example 3) Compile and link an OpenMP Fortran program for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC).

```
[username@hokusai1 ~]$ ifort -qopenmp omp.f
```

Copyright (C) RIKEN, Japan. All rights reserved.

Example 4) Compile and link an OpenMP C program for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC).

```
[username@hokusai1 ~]$ icc -qopenmp omp.c
```

Example 5) Compile and link an OpenMP Fortran program with automatic parallelization for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC).

```
[username@hokusai1 ~]$ ifort -parallel -qopenmp omp_para.f
```

Example 6) Compile and link an OpenMP C program with automatic parallelization for the Massively Parallel Computer (BWMPC) / LARGE Memory Computing Server (BWLMC).

```
[username@hokusai1 ~]$ icc -parallel -qopenmp omp_para.c
```

4.3.1.3 **Compile and link MPI programs**

To compile and link MPI programs for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC) on the front-end servers, use the *mpiifort*/*mpiicc*/*mpiicpc* command.

```
mpiifort/mpiicc/mpiicpc [option] file [...]
```

Example 1) Compile and link a MPI Fortran program for the Massively Parallel Computer (BWMPC) / Large MEmory Computing Server (BWLMC).

```
[username@hokusai1 ~]$ mpiifort mpi.f
```

Example 2) Compile and link a MPI C program for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC).

```
[username@hokusai1 ~]$ mpiicc mpi.c
```

Example 3) Compile and link a Hybrid (MPI + OpenMP) Fortran program for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC).

```
[username@hokusai1 ~]$ mpiifort -qopenmp mpi_omp.f
```

Example 4) Compile and link a Hybrid (MPI + OpenMP) C program for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server.

Copyright (C) RIKEN, Japan. All rights reserved.

```
[username@hokusai1 ~]$ mpiicc -qopenmp mpi_omp.c
```

Optimize for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC)

Optimization may affect computation results. If you apply optimization, verify execution of the execution result. For details, refer to the manual (man command) of each compilation command.

Figure 4-1 General optimization option

Compile Options	Description
-O0	Disables all optimizations.
-O1	Enables optimizations for speed and disables some optimizations that increase code size and affect speed. To limit code size, this option.
-O2	Enables optimizations for speed. This is the generally recommended optimization level. Vectorization is enabled at -O2 and higher levels.
-O3	Performs -O2 optimizations and enables more aggressive loop transformations such as Fusion, Block-Unroll-and-Jam, and collapsing IF statements.
-fast	Maximizes speed across the entire program. It sets the following options: -ipo, -O3, -no-prec-div, -static, -fp-model fast=2, -xHost * The -static option is not available when linking MPI programs.
-qopt-report[=*n*]	Tells the compiler to generate an optimization report. You can specify values 0 through 5 as *n*. The default is level 2.
-qopt-report-phase[=*list*]	Specifies one or more optimizer phases for which optimization reports are generated. For more detail, refer to man manual.
-qopt-report-help	Displays the optimizer phases available for report generation and a short description of what is reported at each level. No compilation is performed.
-qopt-report-routine=string	Tells the compiler to generate an optimization report for each of the routines whose names contain the specified substring. When optimization reporting is enabled, the default is -qopt-report-phase=all.

Copyright (C) RIKEN, Japan. All rights reserved.

Table 4-2 Parallel peformance options

Compile Options	Description
-qopenmp	Enables the parallelizer to generate multi-threaded code based on OpenMP directives.
-parallel	Tells the auto-parallelizer to generate multi-threaded code for loops that can be safely executed in parallel.
-par-threshold[*n*]	Sets a threshold for the auto-parallelization of loops. (from n=0 to n=100. Default: n=100)。 0 – Loops get auto-parallelized always, regardless of computation work volume. 100 – Loops get auto-parallelized when performance gains are predicted based on the compiler analysis data. Loops get auto-parallelized only if profitable parallel execution is almost certain. To use this option, you must also specify option -parallel.
-guide[=*n*]	Lets you set a level of guidance for auto-vectorization, auto parallelism, and data transformation. When this option is specified, the compiler does not produce any objects or executables. You must also specify the -parallel option to receive auto parallelism guidance. The values available are 1 through 4. Value 1 indicates a standard level of guidance. Value 4 indicates the most advanced level of guidance. If n is omitted, the default is 4.
-qopt-matmul	Enables or disables a compiler-generated Matrix Multiply (matmul). The -qopt-matmul options tell the compiler to identify matrix multiplication loop nests (if any) and replace them with a matmul library call for improved performance. The resulting executable may get additional performance gain. This option is enabled by default if options -O3 and -parallel are specified. To disable this optimization, specify -qno-opt-matmul. This option has no effect unless option O2 or higher is set.
-coarray=shared	Enables the coarray feature of the Fortran 2008 standard.

Table 4-3 Processor-specific optimization options

Compile Options	Description
-x*target*	Tells the compiler which processor features it may target, including which instruction sets and optimizations it may generate. When you build only for ACS with GPU, specify option -xCORE-AVX2 for the Haswell microarchitecture.
-xhost	Tells the compiler to generate instructions for the highest instruction set available on the compilation host processor.

Copyright (C) RIKEN, Japan. All rights reserved.

Table 4-4 Interprocedural Optimization (IPO) options and Profile-guided Optimization (PGO) options

Compile Options	Description
-ip	Determines whether additional interprocedural optimizations for single-file compilation are enabled.
-ipo[=*n*]	Enables interprocedural optimization between files. If n is 0, the compiler decides whether to create one or more object files based on an estimate of the size of the application. It generates one object file for small applications, and two or more object files for large applications. If you do not specify n, the default is 0.
-ipo-jobs[*n*]	Specifies the number of commands (jobs) to be executed simultaneously during the link phase of Interprocedural Optimization (IPO). The default is -ipo-jobs1.
-finline-functions -inline-level=2	Enables function inlining for single file compilation. Interprocedural optimizations occur. if you specify -O0, the default is OFF.
-inline-factor=*n*	Specifies the percentage multiplier that should be applied to all inlining options that define upper limits. The default value is 100 (a factor of 1). (Fortran only)
-prof-gen	Produces an instrumented object file that can be used in profile-guided optimization.
-prof-use	Enables the use of profiling information during optimization.

Figure 4-5 Floating-point operation optimization options

Compile Options	Description
-fp-model *name*	Controls the semantics of floating-point calculations.
-ftz[-]	Flushes denormal results to zero when the application is in the gradual underflow mode. It may improve performance if the denormal values are not critical to your application's behavior.
-fimf-precision:*name*	Lets you specify a level of accuracy (precision) that the compiler should use when determining which math library functions to use. The *name* is high, medium or low. This option can be used to improve run-time performance if reduced accuracy is sufficient for the application, or it can be used to increase the accuracy of math library functions selected by the compiler. In general, using a lower precision can improve run-time performance and using a higher precision may reduce run-time performance.
-fimf-arch-consistency=*true*	Ensures that the math library functions produce consistent results across different microarchitectural implementations of the same architecture. The -fimf-arch-consistency option may decrease run-time performance. Deafult is "false”.
-prec-div	Improves precision of floating-point divides. The result is more accurate, with some loss of performance.
-prec-sqrt	Improves precision of square root implementations. The result is fully precise square root implementations, with some loss of performance.

Copyright (C) RIKEN, Japan. All rights reserved.

Figure 4-6 Detailed tuning options

Compile Options	Description
-unroll[*n*]	Tells the compiler the maximum number of times to unroll loops. To disable loop enrolling, specify 0. The default is -unroll, and the compiler uses default heuristics when unrolling loops.
-qopt-prefetch[=*n*]	Enables or disables prefetch insertion optimization. The n (0:Disable-4) is the level of software prefetching optimization desired. The option -qopt-prefetch=3 is enabled by default if option -O2 or higher is set.
-qopt-block-factor=*n*	Lets you specify a loop blocking factor.
-qopt-streaming-stores=*mode*	This option enables generation of streaming stores for optimization. The *mode* is as follows: "always": Enables generation of streaming stores for optimization. The compiler optimizes unde the assumption that the application is memory bound. "never": Disables generation of streaming stores for optimization. "auto": Lets the compiler decide which instructions to use.
-fno-alias	Determines whether aliasing should be assumed in the program. Default is -fno-alias.
-fno-fnalias	Specifies that aliasing should be assumed within functions. Default is -ffnalias.
-fexceptions	Enables exception handling table generation. This option enables C++ exception handling table generation, preventing Fortran routines in mixed-language applications from interfering with exception handling between C++ routines. The -fno-exceptions option disables C++ exception handling table generation, resulting in smaller code. When this option is used, any use of C++ exception handling constructs (such as try blocks and throw statements) when a Fortran routine is in the call chain will produce an error.
-vec-threshold *n*	Sets a threshold for the vectorization of loops based on the probability of profitable execution of the vectorized loop in parallel. (from n=0 to n=100. Default: n =100) 0 – loops get vectorized always, regardless of computation work volume. 100 – loops get vectorized when performance gain are predicted based on the compiler analysis data.

Copyright (C) RIKEN, Japan. All rights reserved.

4.4 BLAS/LAPACK/ScaLAPACK

BLAS/LAPACK/ScaLAPACK is available as a numerical library.

4.4.1 Massively Parallel Computer (BWMPC) / Large Memory Computing Server Numerical Computing Library (BWLMC)

When you use BLAS/LAPACK/ScaLAPACK libraries for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC), the following options are available:

Table 4-7 BLAS/LAPACK/ScaLAPACK options

Library	Parallelism	Option	Remark
BLAS	Sequential	-qmkl=sequential	
	Thread parallel	-qmkl=parallel	
LAPACK	Sequential	-qmkl=sequential	
	Thread parallel	-qmkl=parallel	
ScaLAPACK	MPI parallel	-qmkl=cluster	

Example 1) Use the sequential version BLAS/LAPACK.

```
$ ifort -qmkl=sequential blas.f
```

Example 2) Use the thread parallel version BLAS/LAPACK.

```
$ ifort -qmkl=parallel blas.f
```

Example 3) Use ScaLAPACK (linking the sequential version of BLAS/LAPACK).

```
$ mpiifort -qmkl=cluster scalapack.f
```

About the combinations than the above, refer to the following URL:

https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor

Copyright (C) RIKEN, Japan. All rights reserved.

5. Batch Job and Interactive Job

5.1 Job Management System Overview

The job management system manages all batch jobs and interactive jobs over HBW. Users request job requirements such as partition name, number of nodes, number of cores, and elapsed time to the job management system for the job to be executed.

There are two types of jobs users can submit to HBW.

Table 5-1 Job types

Job type	Usage
Batch job	Execute jobs in batch mode. When a node failure occurs, your job is re-executed if the --restart option is given.
Interactive job	Execute jobs in interactive mode by entering data on user terminals. Mainly used for debugging. Jobs are not re-executed when an error such as a node failure occurs.

Batch jobs can be classified into three types depending on how they are submitted.

Table 5 Batch Job Types

Job classification	Purpose	How to submit
Normal job	Execute jobs based on a job script.	Refer to "5.5.1 Noma Jobl"
Step job	Handle multiple jobs as a group having the execution order or dependency relationship.	Refer to "5.5.2 Step Job"
Array job	Consist of multiple instances of the same normal job submitted at the same time for execution.	Refer to "5.5.3 Array Job"

Users can use the following commands to run jobs.

Table 5-2 Job commands

Function	Command
Submit a job	sbatch / srun
See a job status	squeue / sacct
Delete a job	scancel

Copyright (C) RIKEN, Japan. All rights reserved.

5.2 Job Execution Resource

When submitting a job, specify the "partition" that refers to the hardware on which the job will be executed.

5.2.1 Partition

Partition is an option that specifies the hardware on which the job runs.

Table 3 Partitions

Partition	Job execution destination
mpc	Massively Parallel Computer (BWMPC)
lmc	Large Memory Computing Server (BWLMC)
gpu	GPU Server (BWGPU)

Partition settings for Project

Each project can use the following resources at a time.

Table 5 Concurrent resource usage limit for Project

Partition	Number of running cores	Number of submitted jobs
mpc	2688	5000
lmc	96	1000
gpu	448	100

Table 5-6 Settings for each partition

Partition	Number of nodes	Maximum elapsed time	Maximum Number of cores	Maximum Number of nodes	Memory per node
mpc	260	24h	1344	12	112GiB
mpc_l	156	72h	672	6	112GiB
mpc_a	312	24h	34944	312	112GiB
lmc	2	24h	96	1	2880GiB
lmc_a	2	24h	192	2	2880GiB
gpu	3	72h	112	1	448GiB
Gpu_i	1	24h	112	1	448GiB

*The default elapsed time is 1 hour.

mpc_a and lmc_a are generally restricted in execution. If you want to run a job on this partition, see "5.2.1.2 Partitions that require application."

Copyright (C) RIKEN, Japan. All rights reserved.

Partition that requires application

Some partitions must be claimed before they can be used.

Regardless of whether the job is executed or not, the fairshare value for the requested time will be consumed.

Table 5 Partition that requires application

Partition	Description
mpc_a	A Massively Parallel Computer (BWMPC) can execute massively parallel jobs that cannot normally be executed within a specific time period.
lmc_a	A Large Memory Computing Server (BWLMC) can run massively parallel jobs that cannot normally be run for a specific period of time.

The user who wants to use above resource groups should prepare the following information and contact hpc@riken.jp

- UserName, Project ID, Period
- Reason

Copyright (C) RIKEN, Japan. All rights reserved.

5.3 Job Submission Options

When submitting jobs, specify the following three options as necessary.

- Basic Options
- Resource Options

5.3.1 Basic Options

The basic options you can specify to your job are the following.

Table 4 Basic options for submitting a job

Options	Description
--account <projectID>	Specify the project name to be used for job execution (users with multiple projects must be specified).
--mail-user	Specify where to send mail
--mail-type	Specify email notification
BEGIN	Send email notification on starting a job
END	Send email notification on finishing a job
REQUEUE	Send email notification on re-executing a job
--job-name <name>	Specify a job name
--output <filename>	Output standard output to the specified file. (If -e is not specified, the standard error output is also output to the same file.)
--error <filename>	Output standard error output to the specified file
--requeue	Specifies that the job will be re-executed when a failure occurs. (Default: --no-request)
--dependency	Submit Job as Step Job
afterany:<jobid>	Submitted after the specified job (jobid) ends
afterok:<jobid>	The specified job (jobid) is submitted after normal completion.
afternotok:<jobid>	The specified job (jobid) is submitted after an abnormal end.
after:<jobid[+minute]>	The specified job (jobid) is submitted after the specified time has elapsed.
singleton	Executes jobs with the same user and job name sequentially
--array <start-end>	Submit job as array job
--export=<env>	Inherits environment variables at job submission to the job execution environment
--bigin=<time>	Specify job start time

Copyright (C) RIKEN, Japan. All rights reserved.

5.3.2 Resource Options

Key options for resources used by jobs include.

Table 5 Resource options (common)

Options		Description
--partition <partition_name>		Specify queue (partition)
--nodes <num>		Specify number of nodes
--ntasks <num>		Specify number of processes
--ntasks-per-node <num>		Specify number of processes per node
--cpus-per-task <num>		Specifies the number of logical CPU cores to allocate per process
--ntasks-per-core=<num>		Specifies the number of processes per CPU core
--mem=<num[K|M|G|T]>		Specifies the amount of memory per node
--mem-per-cpu=<num[K|M|G|T]>		Specifies the amount of memory per CPU core Maximum amount of BWMPC: 112G Maximum amount of BWLMC: 2880G
--cpus-per-gpu=<num>		Specifies the number of logical CPU cores to allocate per GPU
--mem-per-gpu=<num[K|M|G|T]>		Specify the amount of memory per GPU Maximum amount of BWGPU: 448G
--time		Specify elapsed time
	<mm>	Specify in minutes
	<mm:ss>	minutes, seconds
	<hh:mm:ss>	Specify in hours, minutes, and seconds
	<days-hh>	Specify in days and hours
	<days-hh:mm>	Specify in days, hours, minutes
	<days-hh:mm:ss>	Days, hours, minutes, and seconds

When you set the amount of memory, the units can be set as following string:

Table 6 Available Units of Memory

Unit	Description
KiB	Kibibyte
MiB	Mebibyte
GiB	Gibibyte
TiB	Tebibyte

The default amount of memory per core is as follows:

Table 5-7 Default amount of memory per core

System	Default amount of memory per core
Massively Parallel Computer (BWMPC)	1G
Large Memory Computing Server(BWLMC)	30G
GPU Server(BWGPU)	4G

Copyright (C) RIKEN, Japan. All rights reserved.

Note that if you request more memory than the memory allocation per core, the calculation time is calculated according to the requested memory amount. It has less memory per CPU core than the HBW system.

When you don't specify the number of processes/threads with the MPI options/OMP_NUM_THREADS environment variable, the program may run with the unintended number of processed/threads and the performance may degrade.

Copyright (C) RIKEN, Japan. All rights reserved.

5.4 **Execute Interactive Jobs**

To run an interactive job, run the *srun* command. The job management system allocates compute resources to the interactive job, and the job starts executing.

When an interactive job is submitted, the job submission options are specified as arguments. The following is the execution of an interactive job that logs into a compute node.

```
srun --partition partitionname --account projectname --pty $SHELL
```

Because the login node has a limited number of processes, we recommend that you use interactive jobs to perform compilations that require many processes or other resources.

Example1) Execute an interactive job for the BWMPC.

```
[username@hokusai1 ~]$ srun --account RB999999 --pty $SHELL
[username@bwmpc001 ~]$ ./a.out
Hello world
[username@bwmpc001 ~] $ exit
exit
```

When executing a program directly from a Front End Server to the calculation node, specify the job submission option as an argument as follows. Please refer to "5.3 Job Submission Options" and specify the job submission options according to the program to be executed.

```
srun --partition=mpc --account projectname [option] programfile
```

Copyright (C) RIKEN, Japan. All rights reserved.

Example2) Example of Running a Program on BWMPC.

```
[username@hokusai1 ~]$ module load intel
[username@hokusai1  ~]$  srun  --partition=mpc  --account  RB999999  --nodes=1  --
ntasks=2 ./a.out

Hello world
[username@hokusai1 ~]$
```

Set environment settings according to the program to be executed before executing the srun command.
Please specify partition=gpu_i when executing an interactive job a GPU server.

Copyright (C) RIKEN, Japan. All rights reserved.

5.5 Submit Batch Jobs

To execute a batch job, the user creates a "job script" in addition to a program and submits the job script to the job management system as a batch job. The description of a command line includes options such as a partition, elapsed time and the number of nodes as well as commands to be executed. The user uses the sbatch command to submit a job script. The submitted jobs are automatically executed by the job management system based on the status of free computing resources and the priority among projects.

5.5.1 Normal Job

To submit a normal job, use the sbatch command with the job script which is executed as a batch job.

```
sbatch [option] [job-script]
```

- If a job script is not specified, a script is read from standard input.
- Job submission options can be set by defining directives in a job script or in standard input.
- If a job is successfully submitted, an identification number (job ID) is assigned to the job.

Example) Submit a normal job.

```
[username@hokusai1 ~]$ sbatch run.sh
Submitted batch job  1234.
```

Copyright (C) RIKEN, Japan. All rights reserved.

5.5.2 Step Job

A step job is a job model that aggregates multiple batch jobs and defines a job chain having an execution order and dependency of the batch jobs. A step job consists of multiple sub-jobs, which are not executed concurrently. The figure below outlines the process sequence of a step job.

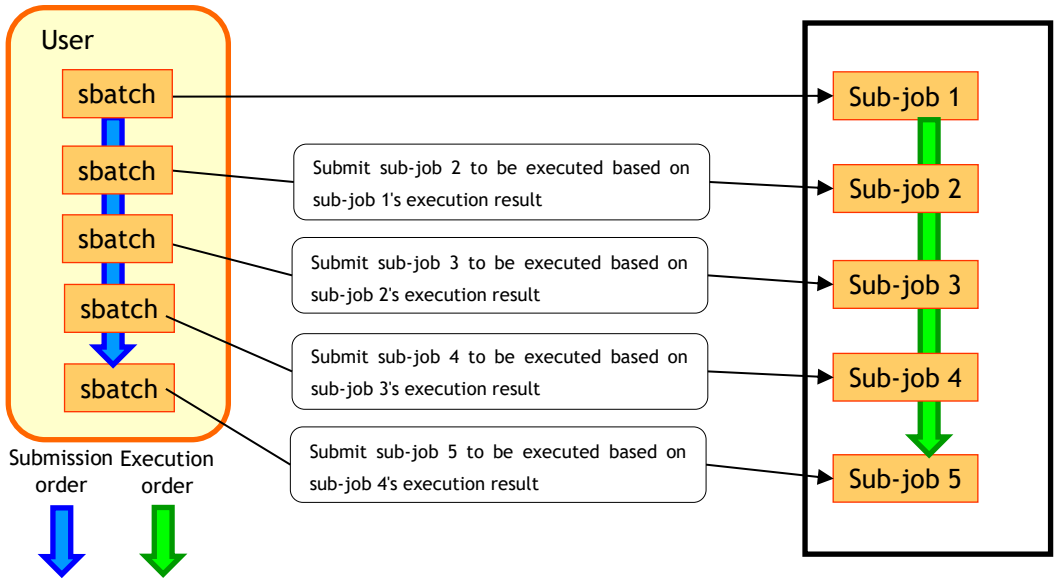

Figure 1 General flow of a step job

The format of submitting a step job is as follows:

```
sbatch --dependency=[option] [job-script]
```

Example 1-1) Submitting a job after the specified job (jobid) ends

```
[username@hokusai1 ~]$ sbatch step1.sh
Submitted batch job 1234
[username@hokusai1 ~]$ sbatch --dependency=afterany:1234 step2.sh
Submitted batch job 1235
[username@hokusai1 ~]$ sbatch --dependency=afterany:1235 step3.sh
Submitted batch job 1236
```

Copyright (C) RIKEN, Japan. All rights reserved.

Example 1-2) Executing jobs with the same user and job name sequentially (only one job can be executed simultaneously)

```
[username@hokusai1 ~]$ sbatch step1.sh
Submitted batch job 1234
[username@hokusai1 ~]$ sbatch --dependency=singleton step2.sh
Submitted batch job 1235
```

Example 1-3) Submitting a job after the specified job (jobid) ends normally

```
[username@hokusai1 ~]$ sbatch step1.sh
Submitted batch job 1234
[username@hokusai1 ~]$ sbatch --dependency=afterok:1234 step2.sh
Submitted batch job 1235
```

Example 1-4) Submitting a job after the specified job (jobid) ends abnormally

```
[username@hokusai1 ~]$ sbatch step1.sh
Submitted batch job 1234
[username@hokusai1 ~]$ sbatch --dependency=afternotok:1234 step2.sh
Submitted batch job 1235
```

Example 2) Submitting a step job by specifying multiple conditions

```
[username@hokusai1 ~]$ sbatch step1.sh
Submitted batch job 1234
[username@hokusai1 ~]$ sbatch --dependency=afterok:1234 step2.sh
Submitted batch job 1235
[username@hokusai1 ~]$ sbatch --dependency=afterok:1234,afternotok:1236
step3.sh
Submitted batch job 1236
[username@hokusai1 ~]$ sbatch --
dependency=afterany:1234:1235,after:1236+1 step4.sh
Submitted batch job 1237
```

Copyright (C) RIKEN, Japan. All rights reserved.

Table 8 Step job option

Options	Description
afterany:<jobid>	Submits the specified job after completion
afterok:<jobid>	The specified job is submitted after normal completion (completion status is 0).
afternotok:<jobid>	The specified job is submitted after an abnormal end (the end status is not 0).
after:<jobid[+minute]>	Submitted after the specified amount of time has elapsed since the specified job started
singleton	Executes jobs with the same user and job name sequentially

Copyright (C) RIKEN, Japan. All rights reserved.

5.5.3 Array Job

An array job is a job that simultaneously submits and executes multiple identical batch jobs.

For example, if you want to change the parameters of a job and check the execution results of each job, you need to input each job in a normal batch job, but you can use array jobs to input multiple patterns at once.

The format of submitting a bulk job is as follows:

```
sbatch --array 0-2:1 jobscript
```

For array jobs, you must create a job script so that you can change the job input/output files for each subjob. To do this, use the array number set for each subjob. The array number can be referenced by the environment variable SLURM_ARRAY_TASK_ID.

Each array job is given a job ID in the form JobID _ TaskID, as well as a job ID like a regular job. However, because normal job IDs are given in the order in which job execution begins, the order of task IDs may not match the order of job IDs.

5.5.4 Job Output

A batch job's standard output file and standard error output file are written under the job submission directory or to files specified at job submission.

Standard output generated during the job execution is written to a standard output file and error messages generated during the job execution are written to a standard error output file. If no standard output and standard error output files are specified at job submission, the following files are generated for output.

slurm-XXXXX.out --- standard output and error output files
(XXXXX is a job ID assigned at job submission)

Copyright (C) RIKEN, Japan. All rights reserved.

5.5.5 Job script description

To submit a batch job, create a job script using the *vi* command or the *emacs* command.

(1) At the top of a job script, put "#!" followed by a path of shell.

[Sample]

```
#!/bin/bash
```

If your login shell is not bash and you execute the module commands in the job script written in sh, you need to specify "#!/bin/sh -l".

(2) From the second line onward, specify submission options using directives starting with "#PJM".

[Sample]

```
#SBATCH --nodes=1              Specify number of nodes
#SBATCH --time=1:00:00         Specify elapsed time
```

(3) After job submission options, set runtime environment variables and specify program execution. Slurm allocates resources such as nodes and processes according to the job submission option by specifying an executable program(task) with the srun command.

[Sample]

```
export OMP_NUM_THREADS=20      Set environment variable
srun ./a.out                   Run a program
```

5.5.6 Execute MPI Program

srun option

The job input options specified in the job script file are inherited. Therefore, although the srun command has the same options as sbatch, you do not need to specify any options when running an MPI program.

Table 5-13 srun options

Options		Description
--cpu-bind=<type>		Bind tasks to CPUs.
	verbose	Log how tasks are bound to GPU and NIC devices.
	rank	Automatically binds by task rank.
	rank_ldom	Bind to a NUMA locality domain by rank. * Can only be specified when using all cores in the node

Copyright (C) RIKEN, Japan. All rights reserved.

5.6 Example script for batch job

5.6.1 Job Script for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC)

5.6.1.1 Sequential Job Script on Single Node for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC)

The following is a sample script for executing the job below.

- ・Partition : mpc
- ・Number of nodes : 1 node
- ・Number of processes (threads) : 1 process (1 thread)
- ・Elapsed time : 60 min
- ・ProjectID : RB999999

```
[username@hokusai1 ~]$ vi bwmpc-seq.sh
#!/bin/sh
#------ slurm option --------#
#SBATCH --partition=mpc
#SBATCH --account=RB999999
#SBATCH --nodes=1
#SBATCH --time=1:00:00
#------- Program execution -------#
module load intel

srun ./a.out
```

Copyright (C) RIKEN, Japan. All rights reserved.

5.6.1.2 Multi-threaded Job Script on Single Node for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC)

The following is a sample script for executing the job below.

- Partition : mpc
- Number of nodes : 1 node
- Number of processes (threads) :1 process (10 threads)
- Elapsed time : 60 min
- ProjectID : RB999999

```
[username@hokusai1 ~]$ vi bwmpc-para.sh
#!/bin/sh
#------ slurm option --------#
#SBATCH --partition=mpc
#SBATCH --account=RB999999
#SBATCH --nodes=1
#SBATCH --cpus-per-task=10
#SBATCH --time=1:00:00
#------- Program execution -------#
module load intel

export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
srun --cpus-per-task=${SLURM_CPUS_PER_TASK} ./a.out
```

When you run a thread parallelized program on the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC), OMP_NUM_THREADS environment variable must be specified. According to specifying amount of memory, a number of allocated cores changes. When you don't specify the number of threads, the program may run with the unintended number of threads and the performance may degrade.

OMP_NUM_THREADS : Number of threads (Specify the ${SLURM_CPUS_PER_TASK} environment variable or the value specified in cpus-per-task)

Copyright (C) RIKEN, Japan. All rights reserved.

5.6.1.3 MPI Parallel Job Script on Single Node for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC)

The following is a sample script for executing the job below.

- Partition : mpc
- Number of nodes : 1
- Number of processes (threads) : 20 process (1 thread)
- Elapsed time : 60 min
- ProjectID : RB999999

```
[username@hokusai1 ~]$ vi bwmpc-single-mpi.sh
#!/bin/bash
#------ slurm option --------#
#SBATCH --partition=mpc
#SBATCH --account=RB999999
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=20
#SBATCH --time=1:00:00
#------- Program execution -------#
module load intel

srun  ./a.out
```

When executing MPI parallel jobs on a Massively Parallel Computing System (BWMPC) / Large Memory Computing Server (BWLMC), the number of cores to be secured may change depending on the amount of memory specified. If omitted, it may be executed with an unintended number of processes.

Copyright (C) RIKEN, Japan. All rights reserved.

5.6.1.4 Hybrid (Multi-thread + MPI) Parallel Job Script on Single Node for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC)

The following is a sample script for executing the job below.

- Partition : mpc
- Number of nodes : 1
- Number of processes (threads) :2 processes (10 threads)
- Number of cores :20 cores (2 x 10)
- Elapsed time : 60 min
- ProjectID : RB999999

```
[username@hokusai1 ~]$ vi bwmpc-single-hybrid.sh
#!/bin/bash
#------ slurm option --------#
#SBATCH --partition=mpc
#SBATCH --account=RB999999
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=2
#SBATCH --cpus-per-task=10
#SBATCH --time=1:00:00
#------- Program execution -------#
module load intel

export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}

srun --cpus-per-task=${SLURM_CPUS_PER_TASK} ./a.out
```

You must specify the OMP_NUM_THREADS environment variable when you run a hybrid parallel job on a Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC). Depending on the amount of memory you specify, the number of cores to allocate may vary, and if omitted, the number of processes/threads may run unintentionally.

OMP_NUM_THREADS : Number of threads (Specify the ${SLURM_CPUS_PER_TASK} environment variable or the value specified in cpus-per-task)

Copyright (C) RIKEN, Japan. All rights reserved.

5.6.1.5 MPI Parallel Job Script on Multinode for the Massively Parallel Computer System (BWMPC) / Large Memory Computing Server (BWLMC)

The following is a sample script for executing the job below.

- ・Partition : mpc
- ・Number of nodes : 2
- ・Number of processes (threads) : 80 process (1 thread)
- ・Number of processes per node :40 processes
- ・Elapsed time : 90 min
- ・ProjectID : RB999999

```
[username@hokusai1 ~]$ vi bwmpc-multi-mpi.sh
#!/bin/sh
#------ slurm option --------#
#SBATCH --partition=mpc
#SBATCH --account=RB999999
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=40
#SBATCH --time=1:30:00
#------- Program execution -------#
module load intel

srun ./a.out
```

When executing MPI parallel jobs on a Massively Parallel Computing System (BWMPC) / Large Memory Computing Server (BWLMC), the number of cores to be secured may change depending on the amount of memory specified. If omitted, it may
be executed with an unintended number of processes.

Copyright (C) RIKEN, Japan. All rights reserved.

5.6.1.6 Hybrid (Multi-thread + MPI) Parallel Job Script on Single Node for the Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC)

The following is a sample script for executing the job below.

- ・Partition : mpc
- ・Number of nodes : 2
- ・Number of processes (threads) : 4 processes (20 threads)
- ・Number of processes per node : 2 processes
- ・Elapsed time : 1 hour 30 minutes
- ・ProjectID : RB999999

```
[username@hokusai1 ~]$ vi bwmpc-multi-hybrid.sh
#!/bin/sh
#------ slurm option --------#
#SBATCH --partition=mpc
#SBATCH --account=RB999999
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=2
#SBATCH --cpus-per-task=5
#SBATCH --time=1:30:00
#------- Program execution -------#
module load intel

export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}

srun --cpus-per-task=${SLURM_CPUS_PER_TASK} ./a.out
```

You must specify the OMP_NUM_THREADS environment variable when you run a hybrid parallel job on a Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC). Depending on the amount of memory you specify, the number of cores to allocate may vary, and if omitted, the number of processes/threads may run unintentionally.

OMP_NUM_THREADS : Number of threads (Specify the ${SLURM_CPUS_PER_TASK} environment variable or the value specified in cpus-per-task)

Copyright (C) RIKEN, Japan. All rights reserved.

5.6.1.7 Sequential Job Script on Single Node for the GPU Server (BWGPU)

The following is a sample script for executing the job below.

- Partition : gpu
- Number of nodes : 1
- Number of processes (threads) : 1 processes (1 threads)
- Elapsed time : 1 hour
- ProjectID : RB999999

```
[username@hokusai1 ~]$ vi bwgpu-seq.sh
#!/bin/bash
#------ slurm option --------#
#SBATCH --partition=gpu
#SBATCH --gres=gpu:hopper:1
#SBATCH --account=RB999999
#SBATCH --nodes 1
#SBATCH --time 01:00:00

#------- Program execution -------#
module load intel
srun ./a.out
```

Copyright (C) RIKEN, Japan. All rights reserved.

5.7 **Job Status**

Use the squeue command to check the status of submitted jobs and resource information.

```
squeue [option] [JOBID [JOBID …]]
```

Table 5-9. squeue options

Options	Description
None	Displays information about jobs waiting to be executed and jobs currently being executed.
--long	Displays detailed information about jobs that are waiting to be executed and jobs that are currently being executed.
--account <account>	Display information of jobs of all users in the same project.
--jobs <jobid>	Displays job information for the specified job ID.
--partition <partitionname>	Displays job information for the specified partition.
--states <states>	Displays job information for the specified job status.
--nodelist <nodename>	Displays job information assigned to a specified node.
--iterate <seconds>	Displays job information at specified intervals (in seconds).

Copyright (C) RIKEN, Japan. All rights reserved.

5.7.1 Job Status

The squeue command displays status of jobs that are currently running or are in the queue.

```
[username@hokusai1 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              1234       mpc   job.sh username  R       0:04      1 bwmpc185
```

Table 5-15 Job status

Field	Description
JOBID	Job ID
PARTITION	Queue name of the job
NAME	Job name
USER	User name
ST	Current job status (see Table 5-16 Job state)
TIME	Job execution time
NODES	Number of nodes used for job execution
NODELIST	List of host names on which the job runs

Table 5-16 Job state

Status	Description
PD	Waiting for job execution
CF	Acquiring resources for job execution
R	Job Running
CG	Waiting for job to finish
CD	Job termination
CA	Termination due to job cancellation
F	Exit with a non-zero exit code or other failure condition
NF	Termination due to failure of one of the assigned nodes
TO	Termination due to execution time limit
RJT	Failure to accept input

Copyright (C) RIKEN, Japan. All rights reserved.

5.7.2 Detailed Job Status (sacct --format option)

The *--format* option of the sacct command displays job information for the specified item. The items to display can be checked with the --helpformat option.

```
[username@hokusai1                ~]$              sacct                -X                --
format=JobID,Partition,Account,Submit,Start,End,CPUTimeRAW,AllocTres%40,State -j 1440
JobID          Partition     Account               Submit               Start                  End
CPUTimeRAW                                  AllocTRES       State
------------ ---------- ---------- ------------------- ------------------- ------------------- ---
------- ---------------------------------------- ----------
1234                   mpc      RB999999 2023-11-16T13:04:05 2023-11-16T13:04:18 2023-11-16T13:09:19
301          billing=8,cpu=1,mem=8800M,node=1  COMPLETED
```

Table 5-17 sacct options

Options	Description
-X	Show only statistics related to the job assignment itself
--accounts	View information for a specified assignment
--jobs	Displays information about the specified job ID
--helpformat	Checking the items displayed with the --fomat option
--fomat	Display specified information
--starttime	Displays job information from the specified date and time (YYYY-MM-DD [THH: MM [: SS]])

Table 5-18 job detailed information

Field		Description
Account		Project Number
Partition		Partition
Submit		Job submission date and time
Start		Job start time
End		Job End Time
Elapsed		elapsed execution time
CPUTimeRAW		Elapsed time (CPU seconds)
AllocTres		Assigned Resources
	billing	Number of cores used in calculation time (number of cores actually used)
	cpu	Number of assigned cores
	mem	Amount of allocated memory
	node	Number of nodes allocated
State		Job Status

Copyright (C) RIKEN, Japan. All rights reserved.

5.7.3 Ended Job Status (sacct command)

When you execute the *sacct* command, jobs that have already been executed are displayed in addition to submitted jobs. The --starttime option can be specified to display information prior to the command execution date.

```
[username@hokusai1 ~]$ sacct
JobID           JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
12345          test_job        mpc   username          4  COMPLETED      0:0
```

Table 5-19 Ended job status

Field	Description
JOBID	Job ID
JOBNAME	Job name
Partition	Partition on which the job was executed
Account	Project ID
AllocCPUS	Number of allocated CPUs
State	Job end status
ExitCode	Job end code

Copyright (C) RIKEN, Japan. All rights reserved.

5.7.4 Viewing partitions information

The sinfo command displays partition available for the user.

```
[username@hokusai1 ~]$ sinfo
PARTITION AVAIL  TIMELIMIT   NODES(A/I/O/T) NODELIST
mpc*         up 1-00:00:00      0/312/0/312 bwmpc[001-312]
mpc_l        up 3-00:00:00      0/156/0/156 bwmpc[001-156]
lmc          up 1-00:00:00          0/2/0/2 bwlmc[1-2]
```

Table 5 Partition status

Field	Description
PARTITION	Queue name (partition name)
AVAIL	Queue Status
TIMELIMIT	Maximum execution time
NODES	Displays the status of the node. A:allocated I:idle O:other T:total
NODELIST	Node assigned to a queue (partition)

Copyright (C) RIKEN, Japan. All rights reserved.

5.7.5 Resource limit of job submission (sinfo command)

The --format option of the sinfo command displays the number of nodes available per partition, the number of CPUs per node, and the maximum age.

```
sinfo --format="%.6P %.5a %.11l %.10s %.6D %.18B %.8m"
```

If you see the following, it indicates that the mpc partition of a Massively Parallel Computer (BWMPC) has 1 to 12 nodes available, that the job can run for up to 24 hours (1 day), and that the CPU per node can run on 112 cores.

It is possible to submit a job that specifies a value that exceeds the displayed upper limit, but the job will not be executed. Therefore, execute the scancel command to delete the job.

```
[username@hokusai1 ~]$ sinfo --format="%.6P %.5a %.11l %.10s %.6D %.18B %.8m"
PARTIT AVAIL    TIMELIMIT   JOB_SIZE  NODES  MAX_CPUS_PER_NODE    MEMORY
  mpc*    up  1-00:00:00       1-12    260                112    128300
 mpc_l    up  3-00:00:00        1-6    156                112    128300
   lmc    up  1-00:00:00          1      2                 96   3095859
```

Table 510 Resource limit status

Field	Description
PARTIT	Partition
AVAIL	Viewing Partition Status
TIMELIMIT	Maximum elapsed time
JOB_SIZE	Number of assigned nodes
NODES	Number of nodes
MAX_CPUS_PER_NODE	Maximum CPUs per node
MEMORY	Memory per node (MB)

Copyright (C) RIKEN, Japan. All rights reserved.

5.8 Cancel jobs

Use the scancel command to cancel submitted jobs.

```
scancel JOBID [JOBID…]
```

Specify job IDs to cancel to the argument on the scancel command line.

```
[username@hokusai1 ~]$ scancel 12345
```

5.9 Environment Variable

5.9.1 environment variables for thread parallelism and MPI execution

This section explains main environment variables specified to execute thread parallelized programs or MPI programs.

Table 5-22 Runtime environment variables

Environment Variable	Description
OMP_NUM_THREADS	When executing a multi-threaded program by OpenMP or auto parallelization, set the number of threads to the OMP_NUM_THREADS environment variable. If this variable is not specified, the number of cores available for the job is set.
KMP_AFFINITY	Control to bind threads to physical processors.
I_MPI_PIN_DOMAIN	Control to bind MPI processes to physical processors.

Copyright (C) RIKEN, Japan. All rights reserved.

5.9.2 Environment Variables for jobs

The job management system configures the following environment variables for jobs.

Table 5-23 Available environment variables in jobs

Environment Variable	Description
SLURM_JOB_ID	Job ID
SLURM_JOB_NAME	Job name
SLURM_SUBMIT_DIR	Current directory when the *sbatch* command is executed
SLURM_ARRAY_JOB_ID	Array job ID (set only for array jobs)
SLURM_ARRAY_TASK_ID	Array job task number (set only for array jobs)

In addition, some of the resource amounts specified as options when submitting a job are set in the following environment variables within the job. Items that are not specified when submitting a job will be automatically set to appropriate values.

Table 5-24 Available environment variables in jobs

Environment Variable	Description
SBATCH_PARTITION	Partition name (value of -p)
SLURM_JOB_NUM_NODES	Number of nodes (value of -N)
SLURM_JOB_CPUS_PER_NODE	number of cores per node Number of cores per node, separated by commas
SLURM_MEM_PER_NODE	Memory per node --mem value
SLURM_MEM_PER_CPU	Memory per core --mem-per-cpu value

You should use the --export option on the sbatch command line to pass environment variables set before job submission on a login node to a job. Otherwise, no environment variables are passed.

Copyright (C) RIKEN, Japan. All rights reserved.

6. Development Tools

6.1 Massively Parallel Computer (BWMPC) / Large Memory Computing Server (BWLMC)

The following tools for the Massively Parallel Computer (BWMPC) / Lrage Memory Computing Server (BWLMC) are available.

- Intel VTune Amplifier XE (Performance profiler)
- Intel Inspector XE (Memory and Thread Debugger)
- Intel Advisor XE (Thread design and prototype)
- Intel Trace Analyzer & Collector (MPI Communications Profiling and Analysis)

Copyright (C) RIKEN, Japan. All rights reserved.