

---

**NEC Network Queuing System V (NQSV) User's Guide**

**[Operation]**

---

## **Proprietary Notice**

The information disclosed in this document is the property of NEC Corporation (NEC) and/or its licensors. NEC and/or its licensors, as appropriate, reserve all patent, copyright and other proprietary rights to this document, including all design, manufacturing, reproduction, use and sales rights thereto, except to the extent said rights are expressly granted to others.

The information in this document is subject to change at any time, without notice.

## Preface

This guide explains how to use NEC Network Queuing System V (NQSV) job management system for general users.

The manual of NEC Network Queuing System V (NQSV) is composed by following user's guides.

Name	Contents
NEC Network Queuing System V (NQSV) User's Guide [Introduction]	This guide explains the overview of NQSV and configuration of basic system.
NEC Network Queuing System V (NQSV) User's Guide [Management]	This guide explains the various management functions of the system.
NEC Network Queuing System V (NQSV) User's Guide [Operation]	This guide explains the various functions that used by general user.
NEC Network Queuing System V (NQSV) User's Guide [Reference]	The command reference guide.
NEC Network Queuing System V (NQSV) User's Guide [API]	This guide explains the C programming interface (API) to control NQSV.
NEC Network Queuing System V (NQSV) User's Guide [JobManipulator]	This guide explains about the scheduler component : JobManipulator.
NEC Network Queuing System V (NQSV) User's Guide [Accounting & Budget Control]	This guide explains the functions of accounting.

February 2018	1st edition
January 2023	16 <sup>th</sup> edition
March 2023	17 <sup>th</sup> edition
May 2023	18 <sup>th</sup> edition
June 2023	19 <sup>th</sup> edition
January 2025	20 <sup>th</sup> edition

## Remarks

- (1) This manual conforms to Release 1.00 and subsequent releases of the NQSV.
- (2) All the functions described in this manual are program products. The functions of them conform to the following product names and product series numbers:

Product Name	product series numbers
NEC Network Queuing System V (NQSV) /ResourceManager	UWAF00 UWHAF00 (support pack)
NEC Network Queuing System V (NQSV) /JobServer	UWAG00 UWHAG00 (support pack)
NEC Network Queuing System V (NQSV) /JobManipulator	UWAH00 UWHAH00 (support pack)

- (3) UNIX is a registered trademark of The Open Group.
- (4) Intel is a trademark of Intel Corporation in the U.S. and/or other countries.
- (5) OpenStack is a trademark of OpenStack Foundation in the U.S. and/or other countries.
- (6) Red Hat OpenStack Platform is a trademark of Red Hat, Inc. in the U.S. and/or other countries.
- (7) Linux is a trademark of Linus Torvalds in the U.S. and/or other countries.
- (8) Docker is a trademark of Docker, Inc. in the U.S. and/or other countries.
- (9) InfiniBand is a trademark or service mark of InfiniBand Trade Association.
- (10) Zabbix is a trademark of Zabbix LLC that is based in Republic of Latvia.
- (11) All other product, brand, or trade names used in this publication are the trademarks or registered trademarks of their respective trademark owners.

## About This Manual

### Notation Conventions

The following notation rules are used in this manual:

Omission Symbol	...	This symbol indicates that the item mentioned previously can be repeated. The user may input similar items in any desired number.
Vertical Bar		This symbol divides an option and mandatory selection item.
Brackets	{ }	A pair of brackets indicates a series of parameters or keywords from which one has to be selected.
Braces	[ ]	A pair of braces indicate a series of parameters or keywords that can be omitted.

### Glossary

Term	Definition
Vector Engine (VE)	The NEC original PCIe card for vector processing based on SX architecture. It is connected to x86-64 machine. VE consists of more than one core and shared memory.
Vector Host (VH)	The x86-64 architecture machine that VE connected.
Vector Island (VI)	The general component unit of a single VH and one or more VEs connected to the VH.
Batch Server (BSV)	Resident system process running on a Batch server host to manage entire NQSV system.
Job Server (JSV)	Resident system process running on each execution host to manage the execution of jobs.
JobManipulator (JM)	JobManipulator is the scheduler function of NQSV. JM manages the computing resources and determines the execution time of jobs.
Accounting Server	Accounting server collects and manages account information and manages budgets.
Request	A unit of user jobs in the NQSV system. It consists of one or more jobs. Requests are managed by the Batch Server.
Job	A job is an execution unit of user job. It is managed by Job Server.

Logical Host	A logical host is a set of logical (virtually) divided resources of an execution host.
Queue	It is a mechanism that pools and manages requests submitted to BSV.
BMC	Board Management Controller for short. It performs server management based on the Intelligent Platform Management Interface (IPMI).
HCA	Host Channel Adapter for short. The PCIe card installed in VH to connect to the IB network.
IB	InfiniBand for short.
MPI	Abbreviation for Message Passing Interface. MPI is a standard for parallel computing between nodes.
NIC	Network Interface Card for short. The hardware to communicate with other node.

## Contents

Proprietary Notice.....	ii
Preface.....	iii
Remarks.....	iv
About This Manual .....	v
Contents.....	i
Contents of Figures.....	vi
1. Batch Request Operation .....	1
1.1. Batch Request Create.....	1
1.1.1. Job Script Create.....	1
1.1.2. Job Script Limitation .....	1
1.1.3. How to Specify Input Data.....	1
1.1.4. How to Specify Submit Options .....	2
1.2. Batch Request Submit.....	2
1.2.1. Basic Submitting Method.....	2
1.2.2. Submit Queue Specifying.....	5
1.2.3. Request Name Specifying .....	5
1.2.4. Job Number Specifying .....	5
1.2.5. Submitting of a request with the total number of VEs specified .....	6
1.2.6. HCA Specifying.....	7
1.2.7. Job Type Specifying.....	7
1.2.8. Result Files Options.....	9
1.2.9. Resource Limit Options.....	10
1.2.10. Start Time Specifying.....	13
1.2.11. Mail Notification Options.....	13
1.2.12. Priority Specifying.....	13
1.2.13. Account Code Specifying .....	14
1.2.14. Rerunning Enable/Disable Specifying.....	14
1.2.15. Hold Enable/Disable Specifying.....	14
1.2.16. Use of Environment Variables .....	14
1.2.17. Shell Specifying .....	17
1.2.18. Batch Job Condition Specifying.....	17
1.2.19. Staging Files Options.....	18
1.2.20. Job Migration Options.....	19
1.2.21. Submission Message Changing .....	19
1.2.22. Request Connection Function .....	19
1.2.23. Exclusive execution .....	20

1.2.24.	Hybrid request.....	22
1.2.25.	User Specifying.....	22
1.2.26.	Accept SIGTERM on job execution.....	23
1.3.	Status Check of Batch Request .....	23
1.3.1.	Check of Basic Information.....	23
1.3.2.	Check of Detailed Information.....	24
1.3.3.	Customizing Information .....	26
1.4.	Attribute Change of Batch Request .....	28
1.5.	Batch Request Delete .....	30
1.6.	Batch Request Hold.....	30
1.7.	Batch Request Release .....	30
1.8.	Batch Request Suspend .....	31
1.9.	Batch Request Resume .....	31
1.10.	Batch Request Rerun.....	31
1.11.	Batch Request Move .....	32
1.12.	Batch Request Termination Check.....	32
1.12.1.	Checking by Mail.....	32
1.12.2.	Checking by qwait(1).....	32
1.13.	Batch Request Output File.....	33
1.14.	Run MPI Request.....	33
1.14.1.	Run under the NEC MPI Environment.....	33
1.14.2.	Run under the OpenMPI Environment.....	34
1.14.3.	Run under the Intel MPI Environment.....	35
1.14.4.	Run under the MVAPICH2 Environment .....	37
1.14.5.	Run under the Platform MPI Environment .....	37
1.15.	Display Output file in Request Running.....	38
1.16.	Request log .....	39
1.17.	Attach to Request.....	40
2.	Parametric Request Operation .....	41
2.1.	Parametric Request Submit.....	41
2.1.1.	Submitting Method .....	41
2.1.2.	Parameter and Input File .....	41
2.2.	State Check of Parametric Request .....	42
2.3.	Parametric Request Operation.....	43
2.4.	Parametric Request Start, Termination, and Output File .....	44
2.4.1.	Mail send .....	44
2.4.2.	Waiting Termination .....	44
2.4.3.	Output File .....	44



3.	Interactive Request Operation.....	46
3.1.	Interactive Request Submit.....	46
3.1.1.	Submitting method by qlogin.....	46
3.1.2.	Submit Option of qlogin .....	47
3.2.	State Check of Interactive Request.....	48
3.3.	Attribute Change of Interactive Request.....	50
3.4.	Operations.....	50
3.5.	Interactive Request Start, Termination, and Output File.....	51
3.6.	Run MPI Request .....	51
3.6.1.	Run under the NEC MPI Environment.....	51
3.6.2.	Run under the OpenMPI Environment.....	52
3.6.3.	Run under the Intel MPI Environment.....	52
3.6.4.	Run under the MVAPICH2 Environment .....	53
3.6.5.	Run under the Platform MPI Environment .....	54
4.	Job Operation.....	55
4.1.	Job State Check .....	55
4.1.1.	Check of Basic Information.....	55
4.1.2.	Check of Detail Information.....	55
4.1.3.	Check of Job information of hybrid request .....	57
4.1.4.	Customizing Information .....	57
4.2.	Signal Send .....	58
5.	Network Request Operation .....	59
5.1.	Status Check of Network Request.....	59
5.2.	Network Request Status Transition.....	60
5.3.	Network Request Delete .....	61
6.	System Information Display .....	62
6.1.	How to Display System Information.....	62
6.2.	Customizing Information.....	63
6.3.	Sorting Information.....	69
6.4.	Time Display Format .....	70
6.5.	Displaying All Information .....	71
6.6.	Displaying All Job Serves (Execution Hosts) Information .....	71
7.	Workflow.....	72
7.1.	Environment for the workflow.....	72
7.2.	Workflow description.....	73
7.2.1.	Simple example .....	73
7.2.2.	Execution order .....	73
7.2.3.	Parallel execution .....	74

7.3.	Workflow execution .....	75
7.4.	Reference of workflow .....	75
7.5.	Control of workflow .....	76
7.5.1.	Workflow delete .....	76
7.5.2.	Request cancel by preceding request error.....	76
8.	Group of Request.....	78
8.1.	Group of Request and How to Submit.....	78
8.2.	Display of Information about Group .....	78
8.2.1.	Request Information.....	78
8.2.2.	Queue Information .....	78
9.	Limit per Group and User.....	80
9.1.	Limit per Group and User .....	80
9.2.	Limit Information Reference per Group and User.....	80
10.	Request Submit Using GPU .....	82
11.	Request Submit Using Multi-Instance GPU(MIG).....	83
11.1.	Multi-Instance GPU(MIG) .....	83
11.2.	Display of Information about MIG .....	84
11.3.	Execution of MIG jobs.....	85
12.	Submitting a request to specify a User Pre-Post Script.....	87
12.1.	UserPP script .....	87
12.2.	Specification method when submitting a request .....	88
12.3.	Output of UserPP script .....	89
13.	Submitting a request using a provisioning environment in conjunction with OpenStack... 91	
13.1.	Reference the template information.....	91
13.2.	Submitting a request with a template specified.....	92
13.3.	Information of a request with a template specified.....	92
13.3.1.	Request information.....	92
13.3.2.	Job information .....	93
14.	Submitting a request using a provisioning environment in conjunction with Docker .....	95
14.1.	Referencing the template information .....	95
14.2.	Submitting a request with a template specified.....	95
14.3.	Information of a request with a template specified.....	96
14.3.1.	Request information.....	96
14.3.2.	Job information .....	97
15.	Submitting a request using the Custom Resource function .....	98
15.1.	Referencing the custom resource information .....	98
15.2.	Submitting a request with a custom resource specified .....	99
15.3.	Referencing the custom resource usage of a queue .....	99

16.	Hybrid Request function.....	101
16.1.	About Hybrid Request .....	101
16.2.	Submitting Hybrid Request .....	101
16.3.	Hybrid Request Information .....	105
16.4.	Attribute Change of Hybrid Request.....	106
17.	Submitting a request using a provisioning environment in conjunction with singularity. 108	
17.1.	Job execution method .....	108
17.1.1.	Executing MPI Job .....	108
17.1.2.	Executing distributed Job .....	109
18.	Limitations .....	110
18.1.	Max Value, Max length and Range of Value .....	110
18.2.	Version between Command and Batch Server.....	112
18.3.	User-level checkpoint.....	112
18.3.1.	Preparing to use user-level checkpoint.....	112
18.3.2.	How to submit a request .....	112
18.3.3.	When taking a checkpoint.....	112
	<b>Appendix.A How to submit NQSV Request.....</b>	<b>114</b>
<b>A.1</b>	<b>Request using VEs .....</b>	<b>114</b>
<b>A.2</b>	<b>Request using x86 .....</b>	<b>115</b>
<b>A.3</b>	<b>Request using GPUs .....</b>	<b>115</b>
<b>A.4</b>	<b>Request using Multi-Instance GPU.....</b>	<b>116</b>
	<b>Appendix.B Update history.....</b>	<b>117</b>
	Index.....	118

## Contents of Figures

Figure 1-1 : Execution of a single job .....	8
Figure 1-2 : Execution of distributed jobs.....	8
Figure 1-3 : Execution of MPI jobs .....	9
Figure 7-1 : Execution of a workflow.....	72
Figure 11-1 Example of dividing GI into CIs.....	84

## 1. Batch Request Operation

### 1.1. Batch Request Create

#### 1.1.1. Job Script Create

In order to create a batch request, create a shell script file for batch requests. This shell script is submitted to NQSV batch system and executed as a batch request.

Any combination of commands can be described in this job script as well as usual shell scripts. Additionally, as a shell to interpret this job script can be specified freely, shell scripts for sh, for csh, or for other shells can be used.

An example of a job script for a batch request is shown below.

```
#
# Sample batch request
#

f90 -o prog1 prog1.f90
if [ $?-ne 0 ]
then echo "prog1.f90 compile error"
    exit 1
fi
prog1 < in_data
```

#### 1.1.2. Job Script Limitation

The job scripts for a batch request have the following limits.

- The commands ( e.g. the stty(1) command ) that requires input/output of terminal devices cannot be used, because the shell script is executed by NQSV regardless of any terminal device.
- A shell script that needs interactive operations cannot be executed as a NQSV batch request, because the standard output (stdout) and error output (stderr) are respectively linked to a file during execution of the shell script.

#### 1.1.3. How to Specify Input Data

As above, the data cannot be interactively input by the job script for a batch request. Please use following two ways if there is necessary data for shell commands.

1. Preparing a data file storing input data for the command:

```
sort < input_data
```

In this example, the data to be sorted is stored in a file "input\_data".

2. Using here-document:

```
sort << EOF
Rebert Cohn was
```

```
once middleweight boxing
champion of Princeton
EOF
```

#### 1.1.4. How to Specify Submit Options

NQSV submit options can be embedded in the comment section of shell scripts. Please specify options with the prefix character string "#PBS" in the comment section before the first shell command appears. "#PBS" indicates the line is NQSV submit option. "#PBS" is a default character string and it can be changed according to the option of qsub(1) command.

All options of qsub(1) command which is used for submitting batch requests can be specified in the shell script as embedded options. This embedded option will be regarded as comment lines for ordinary shells, so it does not affect the execution of job scripts.

```
#
# BATCH request script
#
# Starts at 11:30, Limits CPU time to 21 minutes and 10 seconds.
#PBS -a 1130 -l cputim_prc="21:10"
#
# Submits to the queue batch1.
#PBS -q batch1
make
```

As the example above, multiple options can be embedded in one line. If embedded option should be treated as a comment, do not start a line with "#PBS"

```
##PBS -q batch
# Comment Out #PBS -a 1130 -l cputim_prc="21:10"
```

### 1.2. Batch Request Submit

#### 1.2.1. Basic Submitting Method

Submit the batch requests using the qsub(1) command. The main specifications of the qsub command are as follows.

- Options can be specified in the command line. It can be also specified in the job script which named embedded option. The option in the command line will be valid when the same option is specified in the embedded option.
- The job script will be read directly from standard input unless a shell script file is specified in the command line.

An example of submitting batch request is as follows.

```
$ qsub -q batch1 script1
Request 65.host1 submitted to queue: batch1.
```

This example shows a job script "script1" being submitted to "batch1" queue. The message in the next line of qsub command shows that NQSV batch system accepted the submitted batch request. This message indicates following information:

Request 65.host1 submitted to queue: batch1.
(1) (2)

(1) Request ID assigned to the batch request by NQSV

(2) Name of the queue where the batch request is submitted

An example for reading a shell script from standard input is shown as follows.

<pre>\$ qsub -q batch1 make all Ctrl-D (EOF) Request 66.host1 submitted to queue: batch1. \$</pre>
--

Above example submit the job script which execute "make all" command from standard input to the queue batch1. To finish input from standard input, do Ctrl-D.

### **Submitting Options**

The following options can be specified when submitting batch requests:

(Detailed usage for each option is described after the next chapter.)

#### **1. Options related to result files**

This option specifies where to output results of batch requests (results output to standard output and standard error output). Outputs from a batch request are normally output to an NQSV spool file and copied to a specified file when the batch request is terminated. If the destination is not specified, then the output results of batch requests are output to a file in the directly where batch requests are submitted.

Refer to 1.13. Batch Request Output File.

#### **2. Resource Limit Options**

A batch request can limit resources during execution. Resource limits are functions for forcibly terminating execution when a batch request is executed exceeding predefined maximum values for resources such as CPU time, memory size and file size to be used by a batch request.

In NQSV, resource limit values can be specified as submitting options. Unless specified, resource limit values set in a batch queue by the manager are used for the batch request. If resource limit options are specified when submitting, NQSV compares the specified value

with the values set on the submitted batch queue. If a value specified by submitting option is larger than a value set for the batch queue, this batch request will be rejected. This is because this batch queue will not be able to execute a batch request that consumes larger amount of resources.

### 3. Mail Options

Starting and termination of batch request execution are not normally notified in the NQSV. Using the commands explained in 1.3. Status Check of Batch Request, status of a batch request can be checked.

On the other hand, if this option is specified when submitting a batch request, execution start and termination of the batch request will be notified by e-mail. The NQSV sends an e-mail notifying when execution of a batch request specified by this option is started or terminated. The default destination of e-mail is the person who submitted the batch request, but it can be also changed.

If the execution of a batch request terminated abnormally, e-mails containing information on errors are always sent to the person who made the submission whether or not an option is specified.

### 4. Batch Job Option

In NQSV, the number and the type of jobs that a batch request creates can be specified. Single job will be executed as default. Following job types could be selected and distributed job is default.

- Distributed jobs  
Specified batch request shell scripts are executed in each batch job.
- intmpi jobs  
Intel MPI processes are executed in each batch job.
- mvapich jobs  
MVAPICH processes are executed in each batch job.
- necmpi jobs  
NEC MPI processes are executed in each batch job.
- openmpi jobs  
OpenMPI processes are executed in each batch job.
- platform mpi jobs  
Platform MPI processes are executed in each batch job.



## 5. Other options

There are also other options including an option that specifies time to start batch request execution, an option to specify a queue to be submitted and an option to specify priorities of batch requests.

The main options are briefly described below. Refer to Command `qsub(1)` for detailed descriptions and options other than those mentioned below.

### 1.2.2. Submit Queue Specifying

Specify the queue to which a batch request is submitted by the `-q` option of `qsub(1)` command.

`-q destination`

If this option is not specified, the value of environment variable "PBS\_QUEUE" will be used as a queue to be submitted. Even if "PBS\_QUEUE" is not defined, requests will not be submitted to queues and an error message will be output.

### 1.2.3. Request Name Specifying

Specify the request name by the `-N` option of `qsub(1)` command.

`-N name`

If a request name is not specified with this option, request names will be assigned automatically as follows.

1. The request name will be "STDIN" when a script is input from standard input.
2. If a script file is used, the file name will be the request name excluding the directory part.

[Example] "script" will become a request name.

`$ qsub /usr/nqs/script`

A request name starting with a numeral number will be prefixed with "R".

In case of request names with more than 63 characters, up to 63 characters will be included and the rest will be cut off.

### 1.2.4. Job Number Specifying

Specify the number of batch jobs to be executed in a request by the `-b` option of `qsub(1)` command.

`-b job_count`

### 1.2.5. Submitting of a request with the total number of VEs specified

This function is available only for the environment whose execution host is SX-Aurora TSUBASA system.

Specify the total number of VEs required for executing a request by specifying the `--venode` option for `qsub`. If this option is specified, the number of required jobs is automatically calculated based on the default number of incorporated VE nodes specified for the queue. Because of this, you cannot specify the number of jobs (by using the `-b` option) together with this option. This option is used to specify the number of VEs and number of jobs only. Therefore you need to specify other resource limits too (like Elapse time limit and CPU number limit) to execute the job.

The format of this option is as follows:

```
qsub --venode=<venum>
```

For `<venum>`, specify the number of VE nodes required for the request.

For `<venum>`, you can specify an integer between 1 and 2147483647 but within the limited range of the total number of VEs for the queue (to be described later).

If you specify the number of VEs per logical host (by using the `--venum-lhost` option for `qsub`) together with the `--venode` option, you can then specify a value different from the default number of incorporated VE nodes specified for a queue. The format of this option is as follows:

```
qsub --venode=<venum> --venum-lhost=<venum_lhost>
```

When specifying with the `--venode` option, you need to specify a value of 1 or larger for `<venum_lhost>`, regardless of the lower limit value of the resources for the queue. In addition, the value needs to be less or equal to the upper limit value of the resources for the queue.

If the `--venum-lhost` option is specified at the same time, the number of jobs is automatically calculated by using the value specified for `<venum_lhost>` instead of the default number of incorporated VE nodes. The number of logical hosts is calculated in the same way as calculating the default number of incorporated VE nodes for a queue.

No error or warning is output even if the value specified for `<venum_lhost>` is larger than or equal to the number of VEs actually incorporated on the execution host, but the request is not scheduled.

To refer the value of the `--venode` option specified, use `qstat -f`. If you submit a request with `--venode`

option, you can check the number of jobs calculated using the value specified by the `--venum-lhost` option in the line of "Number of Jobs."

### 1.2.6. HCA Specifying

This function is available only for the environment whose execution host is SX-Aurora TSUBASA system.

When execution host has VE and HCA, HCA can be assigned for the job. You can submit a request specifying use for direct communication and number of HCA port using `--use-hca` option.

The format of the `--use-hca` option is as follows.

`qsub --use-hca=[<mode>:]<num>`

<num> is the number of HCA port which is used by VE which is assigned to a logical host.

<mode> is use the HCA. You can specify one of the following. If mode is not specified it is treated as "all".

- io : I/O exclusive use. Only HCA that is specified IO in device resource configuration file is assigned.
- mpi : MPI exclusive use. Only HCA that is specified MPI in device resource configuration file is assigned.
- all : IO and MPI sharing use (initial value). Only HCA that is specified IO and MPI in device resource configuration file is assigned.

This option is only valid for request using VE. This option could be specified to the request that does not use VE and the request which is executed on the execution host that VE not installed. However this option has no meaning for the request. In that case, any HCA may be used.

For details of this function, please refer to [JobManipulator].

### 1.2.7. Job Type Specifying

Specify the type of jobs to be executed in a batch request by the `-T` option of `qsub(1)` command.

`-T job_type`

Parameters for "*job\_type*" can be set as below:

- |           |   |
|-----------|---|
| • distrib | Batch jobs to be executed are distributed jobs. (Default) |
| • intmpi  | Batch jobs to be executed are intmpi jobs.                |
| • mvapich | Batch jobs to be executed are mvapich jobs.               |
| • necmpi  | Batch jobs to be executed are necmpi jobs.                |
| • openmpi | Batch jobs to be executed are openmpi jobs.               |

[Execution of a single job (The number of batch job =1)]

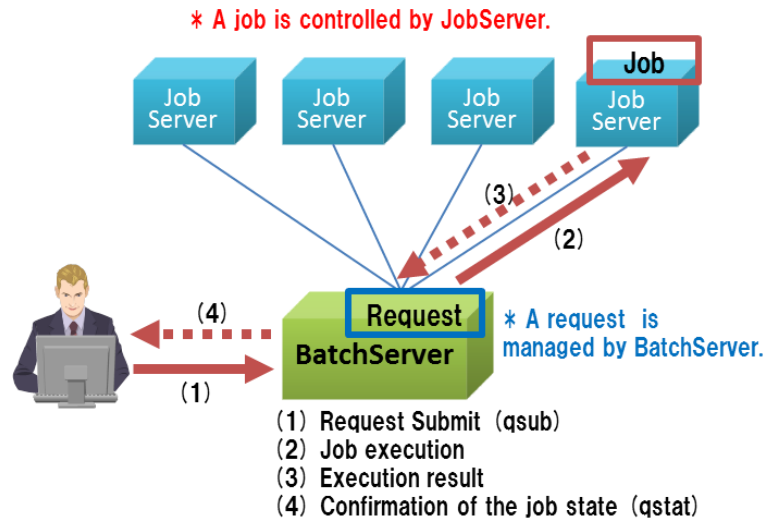


Figure 1-1 : Execution of a single job

[Execution of distributed jobs (The number of batch jobs = 3)]

Request can generate distributed jobs.

Distributed jobs are created by a batch request with the number specified by user.

Here is an example of creating three distributed jobs.

```
$ qsub -T distrib -b 3 script1
```

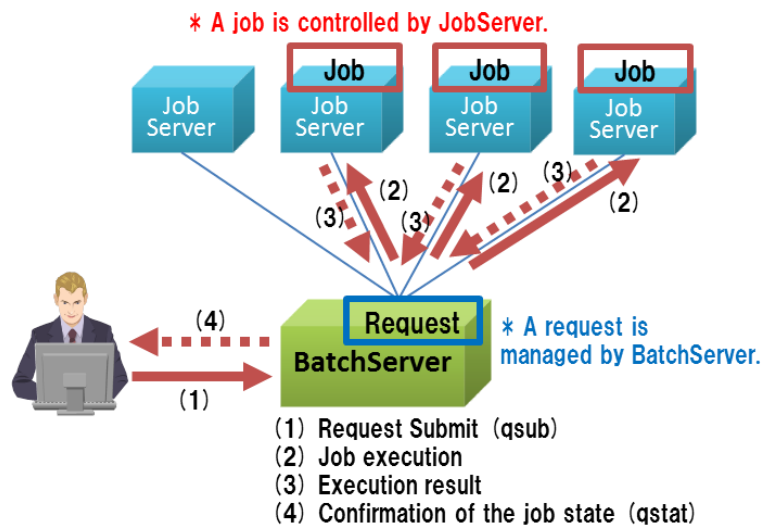


Figure 1-2 : Execution of distributed jobs

[Execution of MPI jobs (The number of batch jobs = 3)]

Request can generate MPI jobs.

MPI jobs are created by a request with the number specified by user and use MPI communication. Here is an example of creating three MPI jobs.

```
$ qsub -T necmpi -b 3 --venum-lhost=1 mpi_script
$ cat mpi_script
mpirun -host 0 -host 1 -host 2 mpi_job
```

Specify a unique number using the `-host` option of `mpirun` command when using the `-T necmpi` option. The number of `-host` option specifications must be equal to the number of batch jobs.

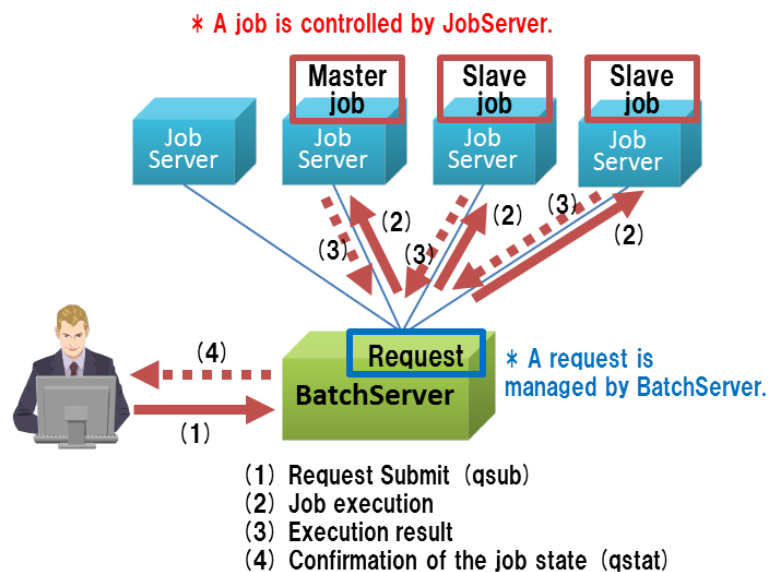


Figure 1-3 : Execution of MPI jobs

### 1.2.8. Result Files Options

Specify the result files of a request by the `-e` option, and `-o` option of `qsub(1)` command.

```
-e file_name
```

Specifies a result file for standard error output from a request.

```
-o file_name
```

Specifies a result file for standard output from a request.

The form of *file\_name* can be set as below:

```
[hostname:] path_name
```

The "*hostname*" is specified a name of a host where results file will be stored. If the "*hostname*" is omitted, results file will be stored to the host where a request was submitted. The "*path\_name*" is specified a name of path where results files will be stored. If the "*path\_name*" is a form of relative path, the path is assumed as the relative path from the current working directory of `qsub`.

It is also possible to merge standard input and output from a batch request using -j option.

### 1.2.9. Resource Limit Options

Specify the resource limits of a request by the -l option of qsub(1) command. Some options are listed below: (For other resource limit, please refer to [Introduction] Resource Limit for Request.)

```
-l filesize_prc=["max_limit",warn_limit"]
```

Sets maximum and warning value of a file size for each process

```
-l memsize_prc=["max_limit",warn_limit"]
```

Sets maximum and warning value of a memory size for each process

```
-l cputime_prc=["max_limit",warn_limit"]
```

Sets maximum and warning value of a CPU time limit for each process

The "*max\_limit*" and "*warn\_limit*" are specified as below:

- Time Limit

Limit values related to time are specified in the following format:

```
[[hours:] minutes:]seconds[.milliseconds]
```

[Example]

1234:58:21.29	1234 hours 58 minutes 21.29 seconds
59:01	59 minutes 1 second
12345	12345 seconds
121.1	121.1 seconds

- Size Limit

Limit values related to size are specified by the following format:

```
[integer][.fraction][units]
```

The following can be specified as *units*:

b	Byte
kb	Kilobyte
mb	Megabyte
gb	Gigabyte
tb	Terabyte
pb	Petabyte
eb	Exabyte

Numbers are interpreted as bytes unless *units* is specified. Numerals that are not greater than or equal to 0 in *integer* even multiplied by 1024 cannot be specified.

[Example]

1234	1234 bytes
------	------------

1234kb	1234 kilobytes
1234.5gb	1234.5 Gigabytes

Some limit parameter can be specified both "max\_limit" and "warn\_limit". The "warn\_limit" has to be below the "max\_limit". If the "warn\_limit" is omitted, it is considered to be the same as the "max\_limit".

When the use resources of the request exceeded the warning value, a signal set by the respective limitation of resources (For the limit per process is according to man of setrlimit(2), for the CPU time limit per logical host is SIGXTERM and the other limit per logical host is SIGTERM) is sent to the request. When exceeding the maximum, execution of a request will be stopped immediately.

NUMA When execution host of a scalar machine for architecture (Linux) is bound, and socket scheduling function puts a request in the queue which becomes on, it is possible to designate socknum\_job (the socket number restriction every job) as qsub -l option instead of cpunum\_job (CPU number restriction every) the job.

I move as the one as which the number of CPU cores according to the equipped resources of the execution host bound by a queue ahead of in case of specified and the investment (the number of CPU cores per socket) designated the number of sockets in cpunum\_job by socknum\_job.

#### [Example]

When a socket of execution host by which 10 cores are be equipped with designated qsub -l socknum\_job=2 as a bound queue and put it in, I move as the one as which qsub -l cpunum\_job=20 was designated.

#### [Note]

When a request is submitted to a queue with socket scheduling enabled, The CPU number limit is automatically set in the OMP\_NUM\_THREADS environment variable. In a job that uses VEs, you have to set the OMP\_NUM\_THREADS environment variables appropriately according to the number of VEs in the job script.

When executing MPI with OpenMP, please specify the value of the OMP\_NUM\_THREADS environment variable with the -v option when submitting a request.

### Resource limits per venode

The resource usage limit can be set to each VE node to be assigned to a job.

```
--< resource name >-venode=resource
resource : ["max_limit",warn_limit"]
```

The following resource limits of a VE node can be specified.

option	meaning	the range of value
--vecputim-lhost= <i>max_limit</i> [, <i>warn_limit</i> ]	VE CPU time per logical host	integer from 0 to 2147483647 or unlimited
--vememsize-lhost= <i>max_limit</i> [, <i>warn_limit</i> ]	VE memory size per logical host	integer from 0 to 2147483647 or unlimited

\*When specifying *max\_limit* and *warn\_limit* in the above table to set the resource limit, use the format of the *-l* option described above.

\*If the resource that is not installed on the execution host is specified, the request cannot be executed.

### Resource limits per logical host

By using a specification method for a job (logical host) that is created with *-b job\_count* specified when inputting a request, the resource usage limit can be set to each logical host to be assigned to a job.

```
--venum-lhost=num
```

Sets the number of VE nodes that must be assigned to each logical host (limit of the number of VE nodes of a logical host). For *num*, an integer from 0 to 256 or "unlimited" can be specified.

```
--< resource name >-lhost=resource
resource : ["max_limit",warn_limit"]
```

Sets the resource amount (resource limit) of a logical host.

This option is equivalent to "*-l <resource name>-job*". The following resource limits of a logical host can be specified.

option	meaning	the range of value
--cpunum-lhost= <i>max_limit</i>	The number of CPUs per logical host	integer from 0 to 2147483647 or unlimited
--cputim-lhost= <i>max_limit</i> [, <i>warn_limit</i> ]	CPU time per logical host	integer from 0 to 2147483647 or unlimited
--gpunum-lhost= <i>max_limit</i>	The number of GPUs per logical host	integer from 0 to 256 or unlimited
--memsize-lhost= <i>max_limit</i> [, <i>warn_limit</i> ]	Memory size per logical host	integer from 0 to 2147483647 or unlimited
--venum-lhost= <i>max_limit</i>	The number of VEs per logical host	integer from 0 to 256 or unlimited
--vecputim-lhost= <i>max_limit</i> [, <i>warn_limit</i> ]	VE CPU time per logical host	integer from 0 to 2147483647 or unlimited
--vememsize-lhost= <i>max_limit</i> [, <i>warn_limit</i> ]	VE memory size per logical host	integer from 0 to 2147483647 or unlimited
--vmemsize-lhost= <i>max_limit</i> [, <i>warn_limit</i> ]	Virtual memory size per logical host	integer from 0 to 2147483647 or unlimited
--stderrsize-lhost= <i>max_limit</i> [, <i>warn_limit</i> ]	Stderr size per logical host	integer from 0 to 2147483647 or unlimited
--stdoutsize-lhost= <i>max_limit</i> [, <i>warn_limit</i> ]	Stdout size per	integer from 0 to 2147483647 or



	logical host	unlimited
--	--------------	-----------

\*When specifying `max_limit` and `warn_limit` in the above table to set the resource limit, use the format of the `-l` option described above.

\*If the resource that is not installed on the execution host is specified, the request cannot be executed.

### 1.2.10. Start Time Specifying

Specify the starting time of a request by the `-a` option of `qsub(1)` command.

```
-a date-time
```

A request will wait for execution until the specified time.

[Example]

1200	12:00 of the day (if it is already past that time, the next day) when a request was submitted
10100900	9:00 on October 10 of year (if it has already passed, the next year) when a request was submitted
200301010900	09:00 on January 1 2003

### 1.2.11. Mail Notification Options

To notify starting or terminating of a request, use the `-m` option, `-M` option of `qsub(1)` command.

```
-m mail_options
```

The following character can be specified to the "*mail\_options*".

- b E-mail is sent when request execution is started.
- e E-mail is sent when request execution is terminated. (includes an abnormal termination)
- a E-mail is sent when request execution is abnormally terminated.  
Abnormal termination indicates the cases when at least one of the running batch job is terminated by signal or when forcibly terminated by trouble of the hardware.
- n E-mail is not sent.

It is possible to combine the character above-mentioned as follows.

`-m eb` E-mail is sent when request execution is started and terminated.

```
-M mail-list
```

Specify mail send-to destinations. The "*mail\_list*" are specified in the following format:

```
mail_address[,mail_address ...]
```

The max length of *mail\_list* is 1023 bytes.

When this option is not specified, a mail is sent to the user on the host who submitted a job.

### 1.2.12. Priority Specifying

Specify the priority of a batch request in a queue by the `-p` option of `qsub(1)` command.

```
-p priority
```

Request priority in a queue is specified. The *priority* must be an integer of the range: [-1024...1023].

The larger this number, the higher the priority. The system will allocate 0 as default if the user does not specify a request priority.

### 1.2.13. Account Code Specifying

Specify the account code of a request by the `-A` option of `qsub(1)` command.

```
-A account_string
```

The account code for requests is specified.

### 1.2.14. Rerunning Enable/Disable Specifying

Specify rerunning enable/disable of a request by the `-r` option of `qsub(1)` command.

```
-r {y | n}
```

The "y" means enable and the "n" means disable. Please refer to 1.10. Batch Request Rerun.

### 1.2.15. Hold Enable/Disable Specifying

Specify hold enable/disable of a request by the `-H` option of `qsub(1)` command.

```
-H {y | n}
```

The "y" means enable and the "n" means disable. Please refer to 1.6. Batch Request Hold.

### 1.2.16. Use of Environment Variables

The options of `qsub(1)` command for each following case are shown below.

- Specify `-V` option in order to export all environment variables in submitting request to the environment for request execution.
- Specify the environment variable to be used to execute a request by `-v` option.

```
-v variable_list
```

[Example]

```
#!/bin/sh

#PBS -v NP=4          #The number of MPI processes
#PBS -T necmpi
#PBS -b $ {NP}
#PBS -I "bin/mpi_prog"
#PBS --venum-lhost=1

cd ${STGDIR}
mpirun -np ${NP} mpi_prog
```

The following environment variables are set by default.

The value can be altered by this option except for `PBS_ENVIRONMENT`,

PBS\_JOBID, PBS\_SUBREQNO, PBS\_JOBNAME and PBS\_NODEFILE.

PBS\_ENVIRONMENT

Sets "PBS\_BATCH" to indicate that the requests is batch processing.

PBS\_JOBID

Sets a batch job identifier of the batch request.

PBS\_SUBREQNO

Sets a sub-request number of the parametric request.

PBS\_JOBNAME

Sets a batch request name

PBS\_NODEFILE

Sets a path to the file of execution host list of all jobs in the request.

PBS\_O\_HOME

Sets Environment Variable "HOME" in a client host.

PBS\_O\_HOST

Sets a client host name.

PBS\_O\_LANG

Sets Environment Variable "LANG" in a client host.

PBS\_O\_LOGNAME

Sets Environment Variable "LOGNAME" in a client host.

PBS\_O\_MAIL

Sets Environment Variable "MAIL" in a client host.

PBS\_O\_PATH

Sets Environment Variable "PATH" in a client host.

PBS\_O\_SHELL

Sets Environment Variable "SHELL" in a client host.

PBS\_O\_TZ

Sets Environment Variable "TZ" in a client host.

PBS\_O\_WORKDIR

Sets a directory for work in a client host.

- Specify the environment variable to be used only in embedded option lines in a script by -w option.

`-w script_variable_list`

[Example]

```
#!/bin/sh
#PBS -N Sample_JOB
```

```
#PBS -w ELAPS=600
#PBS -m b -m e
#PBS -q batch1 #default
#PBS -l elapstim_req=${ELAPS}
```

The environment variable and script variable specified by -v,-w can be referred in embedded option lines. The method to refer is as follows.

`${variable}`

When a same variable name exists, the priority is given in order of the following list.

1. The variable to be used in an embedded option line in a script (specified by -w)
2. The variable specified by qsub command's option (specified by -v)
3. The environment variable

The following environment variables cannot be passed to the job regardless specifying -V or -v option because NQSV generates them and passes to the job.

CUDA_VISIBLE_DEVICES	ENVIRONMENT
FLMOD	HOME
HYDRA_LAUNCHER	HYDRA_LAUNCHER_EXEC
I_MPI_HYDRA_BOOTSTRAP	I_MPI_HYDRA_BOOTSTRAP_EXEC
JMM_COMM_PORT	LAM_MPI_SESSION_PREFIX
LAM_MPI_SESSION_SUFFIX	LOGNAME
MPICH_PROCESS_GROUP	MPI_BATCH_CMD
MPI_MAX_REMSH	MPI_REMSH
NMPI_LAUNCHER_EXEC	NMPI_TTY_COMPAT
NQSII_ELAPSE_OVER_FILE	NQSII_MPINODES
NQSII_MPIOPTS	NQSII_MPI_OUTPUT
NQSII_MPI_OUTPUT_STDERR	NQSII_MPI_OUTPUT_STDOUT
NQSV_MPIOPTS	OMPI_MCA_plm_rsh_agent
OMPI_MCA_plm_rsh_no_tree_spawn	P4_RSHCOMMAND
PATH	PBS_ENVIRONMENT
PBS_EXCLUSIVE	PBS_JOBID
PBS_JOBNAME	PBS_JSVNO
PBS_NODEFILE	PBS_SUBREQNO
QUEUENAME	SHELL
STGDIR	TERM
USER	VE_LIMIT_OPT
_MPI_RNKFL	_NECMPI_JOBTYPE
_NECMPI_MID	_NECMPI_REQID
_NECMPI_VH_NODEFILE	_NECMPI_VH_NODENUM

### 1.2.17. Shell Specifying

Specify the shell to execute the shell script for requests by the -S option of qsub(1) command.

`-S path_name`

An absolute path name of the shell to execute the script of a request is specified to the "*path\_name*".

A shell will be selected by the following method unless an option is specified:

First, the login shell of the user who owns the request is started in executing the request. Next, the login shell selects a suitable shell based on the content of script file for request and this shell (which is not start as login shell) executes a request. (If there is "#!<shell name>" at the top of the script, the login shell starts <shell name> as the shell that executes the script.) Requests are executed in the same procedure as that in interactive processing.

### 1.2.18. Batch Job Condition Specifying

Job condition is a condition for job execution such as which host or job server will execute a job. The job condition is specified when a request is submitted.

For details of job condition function and scheduling, please refer to [JobManipulator].

Specify the job condition of a request by the -B option of qsub(1), qlogin(1), and qrsh(1) command.

`-B ["job_condition", "job_condition"...]`

Specify the conditions for allocating jobs to job servers by the scheduler to *job\_condition*.

The form of *job\_condition* is as follows.

*[job\_number:]condition.*

Specify *job\_number* as target job number that job condition is applied. Following form can be specified.

- |     |  |         |
|-----|--|---------|
| (1) | Single number                            | 0       |
| (2) | Two or more numbers using delimiters "," | 0,2,5   |
| (3) | Consecutive numbers using "-"            | 0-4     |
| (4) | Combination of (2) and (3)               | 0,2,4-6 |
| (5) | All jobs                                 | ALL:    |
| (6) | Not specify equals (5) ALL               |         |

The form of condition is as follows.

`type=condition sentence[:type=condition sentence]`

The condition sentence for each type is as follows.

**Specifying JSV No. :**

JSV=*jsv\_no*

Specify Job Server Number as *jsv\_no*. Following form can be specified.

- |  |         |
|--|---------|
| (1) Single number                            | 0       |
| (2) Two or more numbers using delimiters "," | 0,2,5   |
| (3) Consecutive numbers using "-"            | 0-4     |
| (4) Combination of (2) and (3)               | 0,2,4-6 |

#### Specifying HW Type :

HW=*name[,name....]*

Specify HW type name.

Example) HW=x86\_64

#### Specifying Node Group:

NGRP=*name[,name....]*

Specify Node Group name.

Example) NGRP=grp1,grp2

#### [Notes]

- The format that is not match with above *job\_number* specification, it is treated as *condition* and *job\_number* is treated as ALL.
- If -B *job\_condition* option multiply specified or *job\_condition* for same *job\_number* is specified multiply like -B "*job\_condition*", "*job\_condition*" ..., only the last condition is effective.
- If it needs to specify multiple condition for the same *job\_number*, specify it in the condition by delimit with semi-colon (;).
- Max length of *job\_condition* is 255 bytes.

#### [Example]

JSV No:

-B "0:JSV=100"

-B "1-2:JSV=101-102,105"

HW Type:

-B "ALL:HW=x86\_64"

Node Group:

-B "ALL:NGRP=node\_group"

### 1.2.19. Staging Files Options

Specify the file transferred by the file staging function by the -I option and -O option of *qsub(1)* command.

```
-I "stage_in"["stage_in"...
```

The files related to the job execution are transferred from the client host to the execution host before the job execution.

```
-O "stage_out"["stage_out"...
```

The files which are the output of the job are transferred from the execution host to the client host after the job execution. For details, please refer to [Management] File staging.

### 1.2.20. Job Migration Options

Specify migration enable/disable by the -J option of qsub(1) command.

```
-J { y | n }
```

The "y" means enable, and the "n" means disable.

User files to be routed to a destination host during job migration can be specified by the -G option of qsub(1) command.

```
-G ["migration_file_path"["migration_file_path"...
```

Specify "*migration\_file\_path*" with an absolute path.

For details, please refer to [Management] Job Migration.

### 1.2.21. Submission Message Changing

Specify the following options of the qsub(1) command to change a message for a successful submission of a request.

- -z option            No message is displayed.
- -Z option            Only the request identifier will be displayed.

### 1.2.22. Request Connection Function

Two or more script files can be specified in submitting requests. In this case, these requests are associated and controlled the order of execution by Request connection function. Request connection has two connection types, serial and parallel.

#### Serial type:

The next request starts to execute after finishing of preceding request execution.

In qsub(1) command, specify the script files delimited by space as script-file[ script-file...].

#### Parallel type:

All connected requests start execution at the same time.

In qsub(1) command, specify the script file delimited by ":" as script1[:script2:...].

By request connection function, all the submitted requests to be connected will be in HELD state of Manager privilege and excluded from scheduling target. When submitting of all requests is completed, the first request is released. Therefore, if the error (e.g. option analysis error) occurred while

submitting the requests, the submitted requests remain in HELD state.

In the request connection function, options specified for qsub(1) command are effective to all requests and embedded options specified in the script file are effective only to the corresponding request.

Connected requests cannot be submit to the routing queue that forwards requests to the other batch server. The connection between the requests cannot be kept because they deviate from management by the submitted batch server when forwarding to the other batch server.

### 1.2.23. Exclusive execution

Request can be specified to execute as 1 logical host (job) per 1 host. This execution way calls exclusive execution, and this specified request calls exclusive execution request.

#### (1) Exclusive Execution Request Submit

To submit the exclusive execution request, specify --exclusive option to qsub command. --exclusive take the parameter "host". This "host" parameter can omit. The format of the option is following.

```
qsub --exclusive[=host]
```

This option also can be specified by #PBS line in the job script. The command line option is preferred when it specified with command line option.

This option can't be used with template (--template) option of provisioning function. It will be error when it used at same time.

The resource limit per logical host (CPU, Memory, Virtual Memory, GPU, and VE) can be specified to the exclusive execution request. But it resource limit has no effect that the limit is not used for scheduling and it does not limit the resource usage on the execution host.

The resource limit is treated as following with exclusive execution request.

Resource limit	Treat
CPU number limit per logical host	All CPUs on the execution host is assigned. Socket scheduling feature also assigns all Sockets on the host.
Memory size limit per logical host	All memory (Memory and Virtual memory) on the execution host is assigned. Socket scheduling feature also assigns all memory nodes on the host.
CPU time limit per logical host	It is applied as you specified.
Resource limits per process	It is applied as you specified.
GPU number limit per logical host	All GPUs on the execution host is assigned.
VE and HCA	All the VE and HCA on the execution host is assigned



The RSG of CPuset feature is no meaning for exclusive execution request. If the RSG number of the queue is set to 1 or bigger number, it always assign the RSG0 resources (It means all resources on the execution host).

At the time of use of this function, please set more than 1 as the Queue's Resource Default of CPU number limit, Number of GPU Limit and Number of VE node Limit.

## (2) Job Execution Using Exclusive Execution

It is possible to assign a specific job in the request to high-performance execution host and carry out. And a job can occupy whole resources of an execution host and carry out. If using the exclusive execution function, these can be achieved.

- assignment to a particular execution host of a specific job

In multi-node execution by MPI, we can consider using which gather calculation results of each slave job in rank 0 and process the gathered result with a master job of rank 0

When there are several execution host which have a lot of CPU or memory, rich node, in a cluster, performance of the whole processing and efficient use of resources are expected to improve if above-mentioned high cost calculation of rank 0 is executed on the rich node.

It is possible to do such calculation method using exclusive execution and job condition.

The practical example is as follows

### (1) making node group

First, please make node group to which the nodes, rich nodes, which execute a master job belong.

```
$ qmgr -Pm
Mgr: create node_group = richnode
```

Please add job servers of nodes which execute a master job.

```
Mgr: edit node_group add job_server_id = 100-200 richnode
Add Job_Server to Node_group (richnode).
```

Second, please make node group to which the nodes, non-rich nodes, which execute a slave job belong.

```
$ qmgr -Pm
Mgr: create node_group = slavenode
```

Please add job servers of nodes which execute a slave job.

```
Mgr: edit node_group add job_server_id = 300-400 slavenode
Add Job_Server to Node_group (slavenode).
```

## (2) request submitting

For that, please specify job condition so that the 0th job, master job, is carried out by one of a node which are registered node group richnode and so that the other jobs, slave job, are carried out by one of a node which are registered node group slavenode. In this example, the request has 64 jobs.

```
qsub --exclusive -b 64 -B "0:NGRP=richnode","1-63:NGRP=slavenode"
```

- occupying whole resource of an execution host

We can consider that jobs of a multi-node request occupy whole resource of an execution host and carry out for high-speed running of the job or prevention of fluctuation of performance. If using the exclusive execution function, this can be achieved.

Such host's occupation execution of resources is the exclusive execution function itself.

Therefore if you specify only "--exclusive" at the time of request submitting, you can achieve this.

```
qsub --exclusive script
```

### 1.2.24. Hybrid request

The normal request which has more than one job (= logical host), the resources that can be used by the each job are completely same. On the other hand, the execution form that the different resources are assigned to each job which included in a request is called "hybrid request".

Please refer to 16 Hybrid for detail of Hybrid request feature.

### 1.2.25. User Specifying

You can submit a request with specifying user account by using --user=username option of qsub command, the job will be executed by a specified user. The job is executed by a specified user, and the execution group is the specified user's primary group.

Only users with Group Manager privileges can submit requests with specifying a user. Only users whose primary group is managed by own Group Manager privilege.

If you use user mapping to assign different users to a batch server host and a remote host (a submitting host and an executing host), the --user option cannot be specified.

When you use this option with --group option, the --group option can specify not the supplementary group of the user but the group that matches the following conditions.

- The group must be a supplementary group of the user specified by --user option.
- The group is managed by submit user's Group Manager privilege.

If a user operates a request that submitted with this option such as qalter, qdel, etc. , the -Pg option for a Group Manager privilege must be specified for the command.

The access control of the queue is checked with the user and the primary group of the submitting user regardless of whether this option is specified or not. All other limitation with user-specification are based on the user specified by the --user option.

All files operated in staging function are operated with the permissions of the user specified by the --user option and their primary group.

### 1.2.26. Accept SIGTERM on job execution

SIGTERM can be trapped in the job script if you specify the accept sigterm option on qsub command.

Specify --accept-sigterm option to qsub command as following.

```
qsub --accept-sigterm=<val>
```

If "yes" is specified to <val>, this option is enabled and "no" disables this option.

The default behavior is "no" (disabled) that this option is not specified.

## 1.3. Status Check of Batch Request

### 1.3.1. Check of Basic Information

Check status of a submitted request by the qstat(1) command. A request is specified using request ID.

An example of this direct specification is shown below.

[Example: with request ID]

```
$ qstat 72.host.example.com
RequestID      ReqName  UserName Queue      Pri STT S   Memory      CPU   Elapse R H M Jobs
-----
72.host.example STDIN    user1   batch1      0 RUN -   1.93M      0.23    26 Y Y Y    1
```

[Example: without request ID]

```
$ qstat
RequestID      ReqName  UserName Queue      Pri STT S   Memory      CPU   Elapse R H M Jobs
-----
72.host.example STDIN    user1   batch1      0 RUN -   1.93M      0.23    26 Y Y Y    1
73.host.example STDIN    user1   batch1     20 QUE -   0.00B      0.00     0 Y Y Y    1
```

74.host.example	STDIN	user1	batch2	20	QUE	-	0.00B	0.00	0	N	N	N	2
-----------------	-------	-------	--------	----	-----	---	-------	------	---	---	---	---	---

Column STT will show request status. The meanings of the displayed request states are as follows:

RUN	Executing
QUE	Waiting for execution
WAT	Waiting for start time
HLD	Held
HOL	Holding
SUS	Suspended, Suspending, Resuming (* The same "SUS" is shown for these three states in summary display.)
ARI	Receiving from routing queue
TRS	Sending from routing queue
EXT	Routing execution result file
PRR	Waiting for start of slave request execution (Master request only)
POR	Waiting for end of slave request execution (Master request only)
MIG	Moving of request by dynamic job migration
STG	Creating batch jobs or transferring stage-in target file to execution hosts

See the `qstat(1)` command for more information on items other than STT.

[Note]

The value of the item “Memory” includes file cache if “enable\_memory\_cgroup” is “on” in `/etc/opt/nec/nqsv/nqsd.conf`.

### 1.3.2. Check of Detailed Information

More detailed information on requests can be viewed using the `qstat(1)` command with `-f` option.

```
$ qstat -f 72.host1
Request ID: 72.host1
  Request Name = STDIN
  User  Name = user1
  User  ID  = 111
  Group ID  = 100
  Current State      = Running
  Previous State     = Pre-running
  State Transition Time = Mon Aug 26 19:25:13 2017
  State Transition Reason = PRERUN_SUCCESS
  Queue = batch1@host1 (Execution Queue)
  Job Topology = Distribute Job
  Request Privilege = Level 0
```

```

Request Priority = 0
Request Loglevel = 0
Rerunable = Yes
Holdable = Yes
Hold Type = (none)
Migratable = Yes
Suspend Type = (none)
Account Code = (none)
Stdout = host1:/usr/nec/STDIN.o%s
Stderr = host1:/usr/nec/STDIN.e%s
Reqlog = host1:/usr/nec/STDIN.l%s
Shell = (none)
Mail Address = (none)
Mail Option = (none)
Job Condition:
    Job NO: 0 ""
Number of Jobs = 1
Created Request Time = Mon Aug 26 19:23:30 2017
Entered Queue Time = Mon Aug 26 19:23:30 2017
Planned Start Time = Mon Aug 26 19:23:32 2017
Execute Request Time = (none)
Started Request Time = Mon Aug 26 19:23:32 2017
Ended Request Time = Mon Aug 26 19:25:13 2017
Requested Start Time = (none)
Deadline Time = (none)
UMASK = 022
Checkpoint Interval = 0
Restart File Directory = (none)
Reservation ID = (none)
qattach command = Enable
Attach = No
Cluster Type Select = NONE
UserPP Script = (none)
Exclusive = (none)
VE Node Number = 1
HCA Number = (none)
Accept Sigterm = No
Execution Hosts(JSVNO):
    ehost101(101)
Resources Information:
    Memory = 1.932587MB
    CPU Time = 0.225019S
    Accumulated CPU Time = 0.225019S
    Elapse = 11S
    Remaining Elapse = 446S
    Virtual Memory = 355.789062MB
Logical Host Resources:
    VE Node Number = Max: 1 Warn: ---
    CPU Number = Max: 1 Warn: ---
    GPU Number = Max: 0 Warn: ---
    CPU Time = Max: UNLIMITED Warn: UNLIMITED

```

```

Memory Size           = Max: UNLIMITED Warn: UNLIMITED
Virtual Memory Size   = Max: UNLIMITED Warn: UNLIMITED
VE CPU Time           = Max: UNLIMITED Warn: UNLIMITED
VE Memory Size        = Max: UNLIMITED Warn: UNLIMITED
Stdout Size           = Max: UNLIMITED Warn: UNLIMITED
Stderr Size           = Max: UNLIMITED Warn: UNLIMITED
VE Node Resources:
  VE CPU time          = Max: UNLIMITED Warn: UNLIMITED
  VE Memory Size       = Max: UNLIMITED Warn: UNLIMITED
Resources Limits:
  (Per-Req) Elapse Time Limit = Max: 600S Warn: 600S
  (Per-Job) CPU Time          = Max: UNLIMITED Warn: UNLIMITED
  (Per-Job) CPU Number        = Max: 1 Warn: ---
  (Per-Job) Memory Size       = Max: UNLIMITED Warn: UNLIMITED
  (Per-Job) Virtual Memory Size = Max: UNLIMITED Warn: UNLIMITED
  (Per-Job) GPU Number        = Max: 0 Warn: ---
  (Per-Prc) CPU Time          = Max: UNLIMITED Warn: UNLIMITED
  (Per-Prc) Open File Number  = Max: UNLIMITED Warn: ---
  (Per-Prc) Memory Size       = Max: UNLIMITED Warn: UNLIMITED
  (Per-Prc) Virtual Memory Size = Max: UNLIMITED Warn: UNLIMITED
  (Per-Prc) Data Segment Size = Max: UNLIMITED Warn: UNLIMITED
  (Per-Prc) Stack Segment Size = Max: UNLIMITED Warn: UNLIMITED
  (Per-Prc) Core File Size    = Max: UNLIMITED Warn: UNLIMITED
  (Per-Prc) Permanent File Size = Max: UNLIMITED Warn: UNLIMITED
  (Per-Prc) VE CPU Time       = Max: UNLIMITED Warn: UNLIMITED
  (Per-Prc) VE Memory Size    = Max: UNLIMITED Warn: UNLIMITED
Kernel Parameter:
  Resource Sharing Group = 0
  Nice Value             = 0
User Attributes:
  (none)

```

See the `qstat(1)` command for more information on each item.

### Hybrid request specific

There is some specific feature to check the information of hybrid request.

Please refer to 16.3 Hybrid Request for detail.

[Note]

The value of the item “Memory” in the section “Resources Information:” includes file cache if “enable\_memory\_cgroup” is “on” in `/etc/opt/nec/nqsv/nqsd.conf`.

### 1.3.3. Customizing Information

#### (1) Time format

Time data in the output of `qstat(1)` command is displayed by seconds as default. It is also displayed in the format of `d+hh:mm:ss` by specifying `-d` option. This option is valid in following items.

- CPU time (CPU)

- Accumulated CPU time (ACCPU)
- Elapse time (Elapse)

An example of specifying -d option is as below.

\$ qstat -d													
RequestID	ReqName	UserName	Queue	Pri	STT	S	Memory	CPU	Elapse	R	H	M	Jobs
80536.host1	STDIN	user1	batch1	0	QUE	-	7.62M	1+05:21:10	1+16:30:12	Y	Y	Y	1

## (2) Long format

Sometimes there are cases that information such as execution host name is cut off in the basic information display of qstat(1) command (without -f option) because the viewable number of character is fixed.

By using -l option, it is possible to show all information without being cut.

An example of specifying -l option is as follows.

\$ qstat -l													
RequestID		ReqName						UserName		Queue		Pri STT	
S	Memory	CPU	Elapse	R	H	M	Jobs						
-----													
- - - - -													
80333.host1.example.com				STDIN				nqstest1		que1		0 QUE	
-	0.00B	0.00	0	Y	Y	Y	1						

All information is shown without cutting off. At this time, some displayed contents may run off.

(For details of qstat -l option, please refer to 6.5. Displaying All Information.)

## (3) Selecting Item and Sorting

It is possible to display information by selecting and customizing each item output by -F option of qstat(1) command. However, it is impossible with -f option for long format.

Items which can be specified for request information are as follows.

item	Content	Summary display format
rid	Request identifier	RequestID
reqnm	Request name	ReqName
own	Owner	UserName
quenm	Name of queue resided by request	Queue
pri	Priority	Pri
stt	Request state	STT
stall	Stall state	S
mem	Memory size used by request	Memory
cpu	CPU time of processes during execution	CPU
acpu	Accumulated CPU time	ACCPU
elaps	Elapsed time	Elapse

relaps	Remaining Elapsed time	RElapse
rflg	Re-execution Possible/Not Possible	R
hflg	Hold Possible/Not Possible	H
mflg	Job Migration Possible/Not Possible	M
jobs	Number of jobs residing in request	Jobs
ehost*	Execution host	ExecutionHost
sdate*	Date and time when request was submitted	Date (SUBMIT)
qdate*	Date and time for start of scheduling	Date (QUEUED)
rdate*	Date and time for start of request	Date (RUNNING)
att*	Attached or not	Att

\* This item can be specified in case of displaying information using qstat(1) command -F option.

An example to display batch request information in summary display in order of submitted queue name (quenm), request ID (rid), request status (stt) and execution host (ehost) is as follows.

```
$ qstat -F quenm,rid,stt,ehost
Queue      RequestID      STT ExecutionHost
-----
nqstest    80536.host1    RUN exec1.example.com
nqstest    80542.host1    QUE -
nqstest    80543.host1    QUE -
```

(For details of the qstat -F option, please refer to 6.2. Customizing Information.)

It is possible to sort information by item using -o option or -O option of qstat(1) command.

The following is the example sorting in descending order by the request ID.

```
$ qstat -O rid
RequestID      ReqName  UserName Queue      Pri STT S   Memory      CPU R H Jobs
-----
74.host1      STDIN    user1    batch2    20 QUE -   0.00B      0.00 N N   2
73.host1      STDIN    user1    batch1    20 QUE -   0.00B      0.00 Y Y   1
72.host1      STDIN    user1    batch1     0 RUN -   1.93M      0.23 Y Y   1
```

(For details of qstat -o option and -O option (output at the time of an error), please refer to 6.3. Sorting Information.)

#### 1.4. Attribute Change of Batch Request

Most batch request attributes can be changed after the batch request is submitted.

Request attributes can be changed using the qalter(1) command.

The qalter command can change attribute value by specifying option for attribute and request ID.

Execute as follows when changing the limit value for a CPU time for each process of a batch request whose request ID is 72.host1:



```
$ qalter -l cputim_prc=1000 72.host1
Attribute of Request is altered.
```

Please note the following points concerning attribute changing.

- Some of attributes cannot be changed when a request is executed.
- When change the elapse time limit of running request, all requests that scheduled to after of that request must be rerun.
- Some of attributes cannot be changed when a request is routed in a routing queue.
- It is not possible to change to values that exceed the resource limit value set on an execution queue.
- Resource limit values not supported on the system cannot be changed.

(On Linux system, resource limits per process cannot be changed.)

(About details of the supported restriction value of resources, please refer to [Management]Batch Queue Configuration (Resource Limit, Kernel Parameter).)

The following are some options for changeable attributes by using the qalter command.

-a Changes batch request execution time.

[Example]

```
$ qalter -a 1720 72.host1
Attribute of Request is altered.
```

-e Changes standard-error output result file.

[Example]

```
$ qalter -e host1:/usr/result.e 72.host1
Attribute of Request is altered.
```

-l memsz\_prc

Changes memory-size limit value for each process.

[Example]

```
$ qalter -l memsz_prc=2kb 72.host1
Attribute of Request is altered.
```

-o Changes a standard-output result file.

[Example]

```
$ qalter -o host1:/usr/result.o 72.host1
Attribute of Request is altered.
```

-p Changes a batch request priority.

[Example]

```
$ qalter -p 25 72.host1
```

```
Attribute of Request is altered.
```

**-r n** Changes the batch request rerun enable/disable mode.

[Example]

```
$ qalter -r n 72.host1  
Attribute of Request is altered.
```

**-S** Changes a batch request execution shell.

[Example]

```
$ qalter -S /bin/csh 72.host1  
Attribute of Request is altered.
```

### Attribute change of hybrid request

Please refer to 16.4 Attribute Change of Hybrid Request to change the attribute of hybrid request.

## 1.5. Batch Request Delete

Batch requests can be deleted by the `qdel(1)` command.

```
$ qdel 72.host1  
Request 72.host1 has been deleted.
```

The running batch request will be deleted by SIGKILL signal sent after waiting several seconds specified by the `-g` option (default = 5 seconds).

If a request is specified during transferring result files, also the relative network request of the request will be deleted.

## 1.6. Batch Request Hold

It is possible to keep a request from executing by setting a request to the HELD state.

Request can be held by `qhold(1)` command.

```
$ qhold 72.host1  
Hold request 72.host1 is accepted.
```

Requests can be held in the following four states: (QUEUED, WAITING). It is not possible to hold the request that is not allowed to be held. To execute this command, hold privilege of the target request or higher is necessary.

## 1.7. Batch Request Release

Batch request can be released by the `qrls(1)` command. By releasing the HELD request, the request returns in the state before, and is rescheduled again.

```
$ qrls 72.host1  
Release request 72.host1 is accepted.
```

It is not possible to release a request with a lower privilege than the user who executed holding. However, requests held with Scheduler privilege can be released exceptionally with NQSV Manager privilege.

### 1.8. Batch Request Suspend

For the request running on the execution host, it is possible to suspend the request by sending a SIGSTOP signal to a job. The request will be in the "SUSPEND" state.

Requests can be suspended by the `qsig(1)` command. A request can be suspended only if it is in the RUNNING state.

```
$ qsig -s SIGSTOP 72.host1  
Request 72.host1 was sent signal SIGSTOP.
```

The elapse time limit is valid for the batch request in SUSPENDED state.

To execute this operation, suspend privilege of the target request or higher is necessary.

### 1.9. Batch Request Resume

Requests can be resumed by the `qsig(1)` command by sending a SIGCONT signal.

```
$ qsig -s SIGCONT 72.host1  
Request 72.host1 was sent signal SIGCONT.
```

It is not possible to resume a request with a lower privilege than the user who executed suspending. However, requests suspended with Scheduler privilege can be resume exceptionally with NQSV manager privilege.

### 1.10. Batch Request Rerun

After a request is assigned an execution host and a job was created, the request can be returned to the QUEUE state from any state by rerun operation. If a running request is rerun, the execution jobs will be stopped and the request will be in the QUEUE state. At this time, the request ID is not changed. Request can be rerun by the `qrerun(1)` command.

```
$ qrerun 72.host1  
Request 72.host1 has been rerun.
```

It is not possible to rerun a request that did not create any jobs.

```
$ qrerun 73.host1  
NQSrreq: [BSV EWRNGSTS] Request state isn't suitable. (QUEUED).
```

A request could be rerun SHRT\_MAX (32676) times. It is the max value that the accounting function can record.

### 1.11. Batch Request Move

It is possible to move (re-submit) a request submitted in one batch queue to another batch queue by the `qmove(1)` command. At this time, it is checked whether the resource limit values for the request exceed those for the move-to queue. If exceeded, rerun operation will be failed.

Only batch requests in QUEUED (only if the request did not create any jobs), WAITING and HELD state can be moved. If `-f` option is specified, the batch request that has batch jobs is moved forcibly after deleting the jobs.

```
$ qmove batch1 72.host1
Request 72.host1 has been moved.
```

It is possible to move all the requests in a certain queue to another queue at a time by being specified a certain queue to `qmove(1)` command.

```
$ qmove batch1 -q batch2
Request 72.host1 has been moved.
Request 73.host1 has been moved.
Request 74.host1 has been moved.
```

### 1.12. Batch Request Termination Check

When a request terminates, it is not showed any more by the `qstat` command.

The other ways to check the termination of a request are ways by a mail and by the `qwait(1)` command.

#### 1.12.1. Checking by Mail

An e-mail is sent when the batch request is terminated, if the `"-m e"` option of `qsub` is specified in the submission of a batch request.

#### 1.12.2. Checking by `qwait(1)`

It is possible to wait termination of a batch request by `qwait(1)` command. The `qwait(1)` command waits the termination of a request and displays the termination code of a request.

[Example] When a request has terminated with termination code 45:

```
$ qwait 123.host1
exited 45
```

[Example] When a request was deleted in the QUEUED state:

```
$ qwait 124.host1
deleted in the QUEUED state
```

### 1.13. Batch Request Output File

When the batch request is executed, the following two files are created as the result file.

- Standard output file that stores the contents output to the standard output during the execution of the request.
- Standard error output file that stores the contents output to the standard error output during the execution of the request.

The file name of the standard output file and the standard error file are named as follows.

[Standard output file]

`< request name>. o< sequence number>`

[Example] When a request name is "batreq" and serial request number is 72, "batreq.o72" will be the result file for standard outputs.

[Standard error file]

`< request name>. e< sequence number>`

[Example] When a request name is "batreq" and serial request number is 72, "batreq.e72" will be the result file for standard error outputs.

A request name can be specified when a request is submitted by the -N option of qsub(1) command. The name of the submitted script file will become request name if not specified. The request name will be "STDIN" if a script is input from standard input of qsub(1) command.

### 1.14. Run MPI Request

NQSV supports following MPI.

- NEC MPI
- OpenMPI (OpenMPI Version 4.1.x, Version 5.0.x)
- Intel MPI (Intel(R) MPI Library for Linux OS Version 4.1 Update 3, Version 5.1 Update 3, 2017 Update 1, 2018 Update 3, 2019 Update 7, 2021 Update 5)
- Intel OneAPI (Intel(R) OneAPI Toolkits Release 2021.4)
- MVAPICH2 (MVAPICH2 Version 2.0, Version 2.3.4, Version2.3.6, Version 2.3.7)
- Platform MPI (Platform MPI Version 09.01.04.03)

#### 1.14.1. Run under the NEC MPI Environment

This function is available only for the environment whose execution host is SX-Aurora TSUBASA system.

In NQSV, it is possible to select either Hydra or MPD as process manager to execute NEC MPI jobs for a queue to submit.

\* Queue setting can be confirmed by display of `qstat -Qf` (NEC MPI Process Manager).

To run a job with NEC MPI, submit a request as follows.

- Option in submitting

Specify `necmpi` to `-T` option of the `qsub(1)` command.

An example of the job script to execute MPI program `mpi_prog` executed into 4 logical host which has 8 VE, 1 process per a VE, is shown as follows.

```
#!/bin/sh
#PBS -T necmpi
#PBS -b 4
#PBS -l "bin/mpi_prog,"
#PBS --venum-lhost=8

cd ${STGDIR}

source /opt/nec/ve/mpi/1.0.1/bin/necmpivars.sh
mpirun -nnp 8 -nn 4 -ve 0-7 mpi_prog
```

#### 1.14.2. Run under the OpenMPI Environment

- Submit Method

In NQSV, to execute a job using OpenMPI, submit request as follows.

- Submit Option

Specify `openmpi` with `qsub(1)` command `-T` option.

- `mpirun`

To execute MPI program, use `mpirun` command of OpenMPI.

Specify `#{NQSV_MPIOPTS}` or `#{NQSII_MPIOPTS}` as `mpirun` parameter. (With this parameter, job execution host is provided for MPI.)

[Info]

In case more than one network I/F are available in the jobs, network I/F the jobs use can be designated by `"-mca btl_tcp_if_include"` option.

For example when executing jobs in Docker container, please designate an overlay network I/F by `"-mca btl_tcp_if_include"` option.

For example, to execute OpenMPI program by creating 64 MPI processes, 2 jobs(nodes), as batch request, describe job script as follows.

```
#!/bin/bash
#PBS -T openmpi
#PBS -b 2          # The number of Batch Job
```

```
mpirun ${NQSV_MPIOPTS} -np 64 ${HOME}/mpi_prog
```

- Execution with OpenMP

In case of executing the OpenMPI program with OpenMP, specify multiplicity to OMP\_NUM\_THREADS in a process as job environment variable.

Note that when a request is submitted to a queue with the socket scheduling feature enabled, the OMP\_NUM\_THREADS environment variable is automatically set to the limit value for the number of CPUs per logical host. This environment variable can be overridden by specifying its value with the qsub -v option.

When a job script submits a new request with qsub -V, the environment variables that the first request has are inherited to the second request. Therefore please set the environment variable OMP\_NUM\_THREADS with value used in the second request.

For example, to execute OpenMPI program by creating MPI process on each node, 2 jobs(nodes) and creating 32 threads per each process by OpenMP as batch request, describe job script as follows.

```
#!/bin/bash
#PBS -T openmpi
#PBS -b 2                # The number of Batch Job
#PBS --cpunum-lhost=32   # The number of CPUs per logical host
#PBS -v OMP_NUM_THREADS=32

mpirun ${NQSV_MPIOPTS} -np 2 ${HOME}/mpi_prog
```

#### [Notes]

- Multiple jobs cannot be executed on a single execution host in the case of OpenMPI.
- When the job does not complete normally, such as when the resource limit is exceeded or when a node failure occurs, temporary files may remain under /tmp on the master node. There is no problem with deleting them.
- When using OpenMPI, do not use the -hostfile \${PBS\_NODEFILE} option. Using this option may result in the job using CPUs that have not been allocated by NQSV.

### 1.14.3. Run under the Intel MPI Environment

- Submit Method

In NQSV, to execute a job using Intel MPI, submit request as follows.

- Submit Option

Specify intmpi with qsub(1) command -T option.

- mpirun

mpirun command of Intel MPI is used to execute MPI program.

Specify `{NQSV_MPIOPTS}` or `{NQSII_MPIOPTS}` as mpirun parameter. (With this parameter, job execution host is provided for MPI.)

For example, to execute Intel MPI program by creating 64 MPI processes, 2 jobs(nodes), as batch request, describe job script as follows.

```
#!/bin/bash
#PBS -T intmpi
#PBS -b 2          # The number of Batch Job

mpirun ${NQSV_MPIOPTS} -np 64 ${HOME}/mpi_prog
```

You can also distribute the MPI processes evenly across multiple nodes. For example, to allocate a total of 64 MPI processes on two nodes, 32 processes each, write the job script as follows.

```
#!/bin/bash
#PBS -T intmpi
#PBS -b 2          # The number of Batch Job

mpirun -hostfile ${PBS_NODEFILE} -np 64 -ppn 32 ${HOME}/mpi_prog
```

- Execution with OpenMP

In case of executing Intel MPI program with OpenMP, specify multiplicity to `OMP_NUM_THREADS` in a process as job environment variable.

Note that when a request is submitted to a queue with the socket scheduling feature enabled, the `OMP_NUM_THREADS` environment variable is automatically set to the limit value for the number of CPUs per logical host. This environment variable can be overridden by specifying its value with the `qsub -v` option.

When a job script submits a new request with `qsub -V`, the environment variables that the first request has are inherited to the second request. Therefore please set the environment variable `OMP_NUM_THREADS` with value used in the second request.

For example, to execute Intel MPI program by creating a process on each node, 2 jobs(nodes), and creating 32 threads per each process by OpenMP as batch request, describe job script as follows.

```
#!/bin/bash
#PBS -T intmpi
#PBS -b 2          # The number of Batch Job
#PBS --cpunum-lhost=32 # The number of CPUs per logical host
#PBS -v OMP_NUM_THREADS=32

mpirun ${NQSV_MPIOPTS} -np 2 ${HOME}/mpi_prog
```



Multiple jobs cannot be executed on a single execution host in the case of IntelMPI when the socket scheduling feature is enabled.

When a job is deleted by qdel(1) command during job execution, the following message may appear in the job execution result file (standard error output). But there is no problem in the operation.

```
[proxy:0:1@host1] proxy_upstream_control_cb (../../../../../src/pm/i_hydra/proxy/proxy_cb.c:69): assert
(!closed) failed
[proxy:0:1@host1]HYDI_dmx_poll_wait_for_event
(../../../../../src/pm/i_hydra/libhydra/demux/hydra_demux_poll.c:80): callback returned error status
[proxy:0:1@host1]main (../../../../../src/pm/i_hydra/proxy/proxy.c:964): error waiting for event
```

#### 1.14.4. Run under the MVAPICH2 Environment

In NQSV, to execute a job using MVAPICH2, submit request as follows.

- Submit Option  
Specify mvapich to with qsub(1) command -T option.
- mpiexec  
mpiexec command of MVAPICH2 is used to execute MPI program.  
Specify \${NQSV\_MPIOPTS} or \${NQSII\_MPIOPTS} as mpiexec parameter. (With this parameter, job execution host is provided for MPI.)

For example, to execute MVAPICH program by creating 64 MPI processes, 2 jobs (nodes), as batch request, describe job script as follows.

```
#!/bin/bash
#PBS -T mvapich
#PBS -b 2          # The number of Batch Job
mpiexec ${NQSV_MPIOPTS} -np 64 ${HOME}/mpi_prog
```

#### 1.14.5. Run under the Platform MPI Environment

- Submit Method

In NQSV, to execute a job using Platform MPI, submit request as follows.

- Submit Option  
Specify pltmpi with qsub(1) command -T option.
- mpirun  
mpirun command of Platform MPI is used to execute MPI program.  
Specify \${NQSV\_MPIOPTS} or \${NQSII\_MPIOPTS} as mpirun parameter. With this parameter, job execution host is provided for MPI. If you need to pass environment variables such as PATH and LD\_LIBRARY\_PATH to each job, specify them in the option

-envlist of mpirun.

For example, to execute Platform MPI program by creating 64 MPI processes, 2 jobs (nodes), as batch request, describe job script as follows.

```
#!/bin/bash
#PBS -T pltmpi
#PBS -b 2          # The number of Batch Job

mpirun ${NQSVMPIOPTS} -np 64 -envlist PATH,LD_LIBRARY_PATH ${HOME}/mpi_prog
```

#### [Notes]

- When a job is deleted by qdel command during job execution, the following message may appear in the job execution result file (standard error output). But there is no problem in the operation.  
-15: -c: line 0: unexpected EOF while looking for matching `"  
-15: -c: line 1: syntax error: unexpected end of file

In addition, the following temporary files created at the time of job execution may remain under /tmp. Please delete them manually.

mpijob\_user1\_0, mpiafQeIuGG

### 1.15. Display Output file in Request Running

The contents of the standard output file/the error output file are displayed by qcat(1) command if the request is running. In addition, qcat(1) can display the job script and the list of running process in the job.

The qcat(1) has the following options.

- |              |  |
|--------------|--|
| -e           | Display the standard error output file.  |
| -i (Default) | Display the script file.                 |
| -o           | Display the standard output file.        |
| -p           | Display running process list of the job. |
| -d           | Display the request log.                 |

How to display can be specified in combination with -e or -o option.

- |    |  |
|----|--|
| -f | Display the appended data when the file grows. |
|----|--|

How to display can be specified in combination with options above.

- n** Display the specified number of line. (If this option is not specified, ten lines are displayed.)
- b** Display from the top of the file. (If this option is not specified, displayed from the last line.)

[Example] display 15 lines from the top file input file (job script) of a running job.

```
$ qcat -b -n 15 123.host1
#!/bin/sh
#
#PBS -N Sample_JOB
#PBS -m b -m e
#PBS -j o
#PBS -q batch1 #default
#PBS -l cputim_prc=10:00:00
#PBS -l cputim_job=20:00:00
#PBS -l memsz_prc=100mb
#PBS -l memsz_job=200mb
#PBS -l cpunum_prc=1
#PBS -l elapstim_req=600

date
make
$
```

[Example] display the process list of a running job (batch job id 3:123.host1).

```
$ qcat -p 3:123.host1
JID PID S  SZ  TIME COMMAND
77 2025 S   33  0:01 ksh
77 2030 R  523  5:02 a.out
```

### 1.16. Request log

The processing state of a batch request can be recorded as request log. The request log is recorded in case of setting the log level to 1 or above by -D option of qsub(1) command. (The default level is 0, and the request log is not output.)

The following three levels can be specified as the log level of the request log.

- 0 The request log is not output. (default)
- 1 The logs about the request are output.
- 2 The logs about the request and the batch job are output.

The request log can be referred by -d option of qcat(1) command. Also, the request log can be staged out to the output location specified by -d option of qsub(1) command after the request is completed.

## 1.17. Attach to Request

It is possible to control a running job from outside program by the `qattach(1)` command.

Specify a program to `-c` option of the `qattach(1)` command and a job ID of the target job to the `qattach(1)` command. It is possible to specify a request ID instead of a job ID. At this case, all of the jobs in the request are the target.

The program specified to `-c` option is executed as the part of the job on the execution host. Therefore, the CPU time or the memory usage of the program is accumulated to the usage of the job.

The standard input/output/error of the program can be handled as those of `qattach(1)` command. However, the program created does not have the control terminal, so is it not possible to specify a program that needs the control terminal or control terminal functions.

[Example] debugging a running job

```
$ qcat -p 0:16303
  SID  PID STAT   VSZ   RSS    TIME COMMAND
10073 10073 Ss    22628  2400 00:00:00 nqs_shpd
10073 10074 S    119772 2276 00:00:00 bash
10073 10113 R+    4164   340 00:00:07 a.out

$ qattach -c "gdb --quiet" 0:16303
(gdb) attach 10113
Attaching to process 10113
:
```

It is possible to attach to each job once at the same time.

## 2. Parametric Request Operation

### 2.1. Parametric Request Submit

#### 2.1.1. Submitting Method

A parametric request is the request that executes the same job script as more than one sub-requests while changing input parameters or input files.

A job script for parametric request is the same as a batch request.

To submit a parametric request, a batch queue specified to `qsub(1)` command. At this time, the number range is also specified to `qsub(1)` command `-t` option.

[Example] Specify the number 1 to 5 as a parameter

```
$ qsub -q batch1 -t 1-5 script1
```

```
Request 65[]. bsv1 submitted to queue: batch1.
```

The ways of the `-t` option are as follows.

delimited by comma(,) (ex. `-t 1,3,5`)

Using hyphen(-) (ex. `-t 1-5` It is same as "`-t 1,2,3,4,5`")

Using hyphen(-) and skipping number (:*n*) (ex. `-t 1-5:2` It is same as "`-t 1,3,5`")

When submission of a parametric request succeeds, a message including the request ID and a queue name is shown to standard output like submission of a batch request. The request ID is shown as parametric request ID's, which is sequence number of a usual batch request added "[ ]" marker.

In above example, five sub-requests are created and each sub-request is scheduled and executed same as usual batch requests. The number of jobs and resource limits, etc. specified to submitting parametric request are set to each sub-request.

#### 2.1.2. Parameter and Input File

The "PBS\_SUBREQNO" environment variable can be used to refer to the sub-request number specified by `-t` option in submitting a parametric request as the input parameter when each sub-request is executed.

In specifying a transmission file for file staging by `-I` or `-O` option of `qsub(1)` command, it is possible to include sub-request number in the transmission file path by "%t". For example, each sub-request's transmission files are respectively `/DATA/IN_DATA.001`, `/DATA/IN_DATA.002`, `/DATA/IN_DATA.003` ... as follows.

```
$ qsub -q batch1 -t 1-5 -I ¥"/DATA/IN-DATA.%03t,filein¥" script1
```

```
Request 65[]. bsv1 submitted to queue: batch1.
```

It is possible to refer to the path name as "\$STGDIR/filein" in script file.

## 2.2. State Check of Parametric Request

Check status of a submitted parametric request by the `qstat(1)` command.

\$ qstat													
RequestID	ReqName	UserName	Queue	Pri	STT	S	Memory	CPU	Elapse	R	H	M	Jobs
71. bsv1	STDIN	user1	batch1	0	RUN	-	1.93M	0.23	46	Y	Y	Y	1
72[.]. bsv1	STDIN	user1	batch1	0	QUE	-	-	-	-	Y	Y	Y	1

When a request and a parametric request exist, without options in a request reference by the `qstat` command, the information indicates a parametric request. For a certain parametric request, specify the parametric request ID that is a sequence number added "[ ]".

\$ qstat 72[]. bsv1													
RequestID	ReqName	UserName	Queue	Pri	STT	S	Memory	CPU	Elapse	R	H	M	Jobs
72[]. bsv1	STDIN	user1	batch1	0	QUE	-	-	-	-	Y	Y	Y	1

When checking the state of the sub-request executed actually as a parametric request, specify `-s` option of the `qstat(1)` command. The information of sub-request instead of its parametric request is shown.

\$ qstat -s													
RequestID	ReqName	UserName	Queue	Pri	STT	S	Memory	CPU	Elapse	R	H	M	Jobs
71. bsv1	STDIN	user1	batch1	0	RUN	-	1.93M	0.23	46	Y	Y	Y	1
72[1]. bsv1	STDIN	user1	batch1	0	RUN	-	3.87M	0.00	29	Y	Y	Y	1
72[2]. bsv1	STDIN	user1	batch1	0	QUE	-	0.00B	0.00	0	Y	Y	Y	1
72[3]. bsv1	STDIN	user1	batch1	0	QUE	-	0.00B	0.00	0	Y	Y	Y	1
72[4]. bsv1	STDIN	user1	batch1	0	QUE	-	0.00B	0.00	0	Y	Y	Y	1
72[5]. bsv1	STDIN	user1	batch1	0	QUE	-	0.00B	0.00	0	Y	Y	Y	1

Request ID of sub-request is form that is parametric request ID and sub-request number clipped by "[" and "]".

The upper limit is set as the number of the sub-request for parametric request created on the batch server (Default 100). Therefore, all the sub-requests specified by the `-t` option of the `qsub(1)` command may not be shown. In addition, parametric requests suspended by the `-h` or `-a` option of the `qsub(1)` command may not be shown because sub-requests have not created yet.

For the summary information of parametric requests, use the `qstat(1)` command with the `-R` option.

\$ qstat -R												
RequestID	ReqName	UserName	Queue	Pri	STT	R	H	M	Jobs	TOTAL	ACTIVE	DONE
72[]. bsv1	STDIN	user1	batch1	0	RUN	Y	Y	Y	1	5	2	3

The number of all sub-requests is shown to TOTAL, the number of sub-requests on the batch server is shown to ACTIVE and the number of terminated sub-requests (include deleted sub-requests) is shown to DONE.

[Note]

The value of the item “Memory” includes file cache if “enable\_memory\_cgroup” is “on” in /etc/opt/nec/nqsv/nqsd.conf.

### 2.3. Parametric Request Operation

There are operation to the whole parametric request including all sub-requests and operation by the sub-request unit. For the operation to the whole parametric request, specify parametric request ID. For the operation by the sub-request unit, specify sub-request ID.

To delete the whole parametric request, specify as below.

```
$ qdel 72[].bsv1
Request 72[].bsv1 was deleted.
```

To delete the certain sub-request 72[2], specify as below.

```
$ qdel 72[2].bsv1
Request 72[2].bsv1 was deleted.
```

Each operation works as below.

Operation	The operation to the whole parametric request	The operation by the sub-request unit
Delete (qdel)	Delete all sub-requests and parametric request.	Delete a specified sub-request.
Alter Attribute (qalter)	Alter common attributes among all sub-requests. * The change result is effective to sub-requests executed after altering.	Unable
Hold (qhold)	Hold all sub-requests. * It is not assure to hold each sub-request.	Hold a specified sub-request.
Hold Release (qrls)	Release all held sub-requests.	Release a specified held sub-request.
Suspend (qsig -s STOP)	Suspend all sub-requests.	Suspend a specified sub-request.
Resume (qsig -s CONT)	Resume all sub-requests.	Resume a specified sub-request.
Rerun (qrerun)	After deleting all sub-requests, submit the parametric request again.	Rerun a specified sub-request.
Move queue (qmove)	Change submit queue for the whole parametric request. It is possible only when all sub-requests have not started. * When a sub-request has a job, the "force" option is needed.	Unable
Attach (qattach)	Unable	Attach to a job in a specified sub-request.

[Notes]

Hold operation for a parametric request is performed for each sub-request. So the hold processing can

be failed according to the states of sub-requests. Therefore, the HELD state of the parametric request does not mean that the states of sub-requests are all HELD.

## 2.4. Parametric Request Start, Termination, and Output File

### 2.4.1. Mail send

An e-mail is sent when the batch request is starting or terminated, if the "-m e" option of qsub(1) command.

It is possible to specify it so that a mail may be sent at the time of start or termination of the parametric request and at the time of start or termination of the each sub-request.

Mail option	At the timing of a mail send
b	Parametric request starts.
e	Parametric request terminates.
a	Sub-request aborts abnormally.
s	Each sub-request starts.
d	Each sub-request terminates.

### 2.4.2. Waiting Termination

It is possible to wait termination of a request or each sub-request by qwait(1) command. For waiting a parametric request, specify the request ID. For waiting a sub-request, specify the sub-request ID. For the operation to the whole parametric request, specify parametric request ID. For the operation by the sub-request unit, specify sub-request ID.

To delete the whole parametric request, specify as below.

[Example] waiting parametric request

```
$ qwait 72[]. bsv1
exited (3 subrequests)
```

[Example] waiting a certain sub-request

```
$ qwait 72[2]. bsv1
exited 0
```

### 2.4.3. Output File

To execute a parametric request by the sub-request unit, an output file is also made by each sub-request. The file name of the standard output file is named as follows.

```
<request name>.o<sequence number>.< sub-request number>
```

[Example] When a request name is "preq", serial request number is 72 and sub-request number is 2, "preq.o72.2" will be the result file for standard output.

When a standard output file is specified by the "-o" option of the qsub(1) command, the file name is named as follows.



`<specified file name>.<sub-request number>`

The file name of the standard error file is named as follows.

`<request name>.e<sequence number>.<sub-request number>`

[Example] When a request name is "preq", serial request number is 72 and sub-request number is 2, "preq.e72.2" will be the result file for standard errors.

When a standard error file is specified by the "-e" option of the qsub(1) command, the file name is named as follows.

`<specified file name>.<sub-request number>`

For a parametric request, sub-request number can be expanded in file name specified with qsub -o, -e and -O options by using "%t".

And the %r in the file name specified with qsub -o, -e, -I and -O options is expanded for each sub-request as

`<sequence-number>[<sub-request-number>].<BSV host name>`

When an external staging is used, %r expansion might cause a problem that the "[" and "]" characters are not recognized as file name characters. In that case, please use %P expansion that uses "\_" instead of "[" and "]" for request ID.

%P is expanded in the file name specified with -o, -e, -I and -O options as

`<sequence-number>_<sub-request-number>.<BSV host name>`

### 3. Interactive Request Operation

#### 3.1. Interactive Request Submit

To submit interactive request, there are two types of way. One is to use the qlogin command and the other is to use the remote execution function (the qrsh command). The qlogin command, which is a session connection type, is explained in this section.

About the remote execution function (the qrsh command), please refer to [Management] Remote Execution by Interactive Function.

##### 3.1.1. Submitting method by qlogin

To submit interactive request of a session connection type by the qlogin(1) command. Some options can be specified to the qlogin(1) command. The options which can be specified are the same as the options of qsub(1) command basically. About the options for only qlogin(1) is explained at 3.1.2. Submit Option of qlogin.

Below is an example of interactive request by the qlogin command.

```
$ qlogin -q ique --cpunum-lhost=4 -l elapstim_req=1800
Request 111.bsv1 submitted to queue: ique.          <-(1)
Waiting for 111.bsv1 to start.                    <-(2)
-sh-4.1$                                           <-(3)
-sh-4.1$ hostname
ehost001
```

The above example designates resource limits as interactive queue ique and submits interactive request.

- (1) The message display next to the qlogin command means that NQSV accepted the interactive request, request ID assigned by NQSV and the queue name are displayed.
- (2) Showing a message of "Waiting for ...", and wait for a connection of a session with execution host. After the connection of both standard input/output and error output is established, interactive request starts.
- (3) A prompt of a shell on the execution host is displayed, and it becomes possible to use execution host interactively.

When an interactive request is submitted with qlogin command, interactive session is connected from a job server on execution host to the qlogin command. At this time, the host name on which the qlogin command is executed is used as the destination host.

If the host name of the client host on which qlogin executed cannot be referenced on execution hosts, you can use NQSV\_INTERACTIVE\_IP or NQSII\_INTERACTIVE\_IP environment variable to specify the qlogin's client host.

There are no difference for using NQSV\_INTERACTIVE\_IP and NQSII\_INTERACTIVE\_IP. If both of these environment variables are specified, the value of NQSV\_INTERACTIVE\_IP is valid. Please use NQSV\_INTERACTIVE\_IP as follows.

```
$ export NQSV_INTERACTIVE_IP=192.168.0.1
$ qlogin -q ique --cpunum-lhost=4 -l elapstim_req=1800
Request 111.bsv1 submitted to queue: ique.
:
```

### 3.1.2. Submit Option of qlogin

#### (1) Queue Name

```
-q queue
```

Specify an interactive queue.

#### (2) Number of Jobs

```
-b N
```

When the type of jobs specified by -T option is "distrib", which means distributed job, "N" has to be "1". Even if more than "1" is specified, it is assumed that "1" is set.

#### (3) Waiting of Execution Host Allocation

```
-W
```

When waiting setting of execution host allocation for interactive queue is set to "manual", you can wait for the allocation by the -W option of qlogin command. When not specifying -W option and there is no execution host to be assigned immediately, the submission will be canceled.

When waiting setting of execution host allocation for interactive queue is not set to "manual", the -W option is no effect.

(For details of waiting setting of interactive queue, please refer to [Management] Interactive Queue Configuration (Waiting Option).)

#### (4) Idle Timer

```
-l idle_timer=max_limit
```

If input dose not arrive, after waiting *max\_limit* minutes the shell disconnects its session and deletes request automatically.

(For details of idle timer of interactive queue, please refer to [Management] Interactive Queue Configuration (Idle Timer).)

#### (5) Others

Interactive request has the following features.

1. It does not do staging because input/output of a job is connected with stdin/stdout/stderr of qlogin.
2. It does not have priority, rerun, and parametric request functions like the batch execution.

Therefore, there are not specifications for following functions.

- Embedded option for script file
- Request log
- Standard output/stderr file
- Migration
- Hold, rerun
- Designation related to
- Staging
- Request priority

### 3.2. State Check of Interactive Request

Check status of a submitted interactive request by the qstat(1) command.

[request ID is specified]

```
$ qstat 82.host
```

RequestID	ReqName	UserName	Queue	Pri	STT	S	Memory	CPU	Elapse	R	H	M	Jobs
82.host.example	QLOGIN	user1	ique	0	RUN	-	1.93M	0.23	128	N	N	N	1

[request ID is not specified]

```
$ qstat.
```

RequestID	ReqName	UserName	Queue	Pri	STT	S	Memory	CPU	Elapse	R	H	M	Jobs
73.host.example	STDIN	user1	batch1	20	QUE	-	0.00B	0.00	0	Y	Y	Y	1
74.host.example	STDIN	user1	batch2	20	QUE	-	0.00B	0.00	0	N	N	N	2
82.host.example	QLOGIN	user1	ique	0	RUN	-	1.93M	0.23	128	N	N	N	1

[Detailed information]

```
$ qstat -f 82.host
```

Request ID: 82.host.example.com	
Request Name	= QLOGIN
User Name	= user1
User ID	= 100
Group ID	= 100
Current State	= Running
Previous State	= Pre-running
State Transition Time	= Sat Feb 8 15:26:35 2017
State Transition Reason	= PRERUN_SUCCESS
Queue	= ique@ host.example.com (Interactive Queue)

```

Job Topology = Distribute Job
Request Privilege = Level 0
Request Priority = 0
Rerunable = No
Holdable = No
Migratable = No
Suspend Type = (none)
Account Code = (none)
Shell = (none)
Restrict shell = (none)
Mail Address = user1@client.example.com
Mail Option = (none)
Job Condition:
    Job NO: 0 ""
Number of Jobs = 1
Created Request Time = Sat Feb 8 15:26:22 2017
Entered Queue Time = Sat Feb 8 15:26:22 2017
Planned Start Time = Sat Feb 8 15:26:35 2017
Execute Request Time = (none)
Started Request Time = Sat Feb 8 15:26:35 2017
Ended Request Time = (none)
Requested Start Time = (none)
UMASK = 022
Interactive Host = client.example.com
Interactive Port = 54467
Idle Timer = 0
qattach command = Enable
Attach = No
Cluster Type Select = NONE
UserPP Script = (none)
Exclusive = (none)
HCA Number = (none)
Accept Sigterm = No
Execution Hosts(JSVNO):
    ehost100(100)
Resources Information:
    Memory = 3.011719MB
    CPU Time = 0.010000S
    Accumulated CPU Time = 0.010000S
    Elapse = 766S
    Remaining Elapse = 2834S
    Virtual Memory = 355.789062MB
Logical Host Resources:
    VE Node Number = Max: 1 Warn: ---
    CPU Number = Max: 1 Warn: ---
    GPU Number = Max: 0 Warn: ---
    CPU Time = Max: UNLIMITED Warn: UNLIMITED
    Memory Size = Max: UNLIMITED Warn: UNLIMITED
    Virtual Memory Size = Max: UNLIMITED Warn: UNLIMITED
    VE CPU Time = Max: UNLIMITED Warn: UNLIMITED
    VE Memory Size = Max: UNLIMITED Warn: UNLIMITED

```

#### VE Node Resources:

VE CPU Time = Max: UNLIMITED Warn: UNLIMITED

VE Memory Size = Max: UNLIMITED Warn: UNLIMITED

#### Resources Limits:

(Per-Req) Elapse Time Limit = Max: 3600S Warn: 3600S

(Per-Job) CPU Time = Max: UNLIMITED Warn: UNLIMITED

(Per-Job) CPU Number = Max: 1 Warn: —

(Per-Job) Memory Size = Max: UNLIMITED Warn: UNLIMITED

(Per-Job) Virtual Memory Size = Max: UNLIMITED Warn: UNLIMITED

(Per-Job) GPU Number = Max: 0 Warn: —

(Per-Prc) CPU Time = Max: UNLIMITED Warn: UNLIMITED

(Per-Prc) Open File Number = Max: UNLIMITED Warn: —

(Per-Prc) Memory Size = Max: UNLIMITED Warn: UNLIMITED

(Per-Prc) Virtual Memory Size = Max: UNLIMITED Warn: UNLIMITED

(Per-Prc) Data Segment Size = Max: UNLIMITED Warn: UNLIMITED

(Per-Prc) Stack Segment Size = Max: UNLIMITED Warn: UNLIMITED

(Per-Prc) Core File Size = Max: UNLIMITED Warn: UNLIMITED

(Per-Prc) Permanent File Size = Max: UNLIMITED Warn: UNLIMITED

(Per-Prc) VE CPU Time = Max: UNLIMITED Warn: UNLIMITED

(Per-Prc) VE Memory Size = Max: UNLIMITED Warn: UNLIMITED

#### Kernel Parameter:

Resource Sharing Group = 0

Nice Value = 0

#### User Attributes:

(none)

Please refer to qstat (1) about the contents of each item.

#### [Note]

The value of the item "Memory" when basic information and the item "Memory" in the section "Resources Information:" when detail information includes file cache if "enable\_memory\_cgroup" is "on" in /etc/opt/nec/nqsv/nqsd.conf.

### 3.3. Attribute Change of Interactive Request

It is possible to change the attribute of the interactive request like a batch request by the qalter(1) command. Attribute that cannot be specified to interactive request cannot be changed.

### 3.4. Operations

The following operations are also possible like a batch request to interactive request.

Operation	Command
Delete	qdel
Attribute Change	qalter
Suspend	qsig -s STOP

Resume	qsig -s CONT
Attach	qattach

It is not possible to do Move, Hold and Rerun of interactive request.

And it is not possible to move a batch request to interactive queue.

#### [Notes]

When suspend/resume is performed on an interactive request that is running in an OpenMPI environment, the running MPI process may be forced to stop.

### 3.5. Interactive Request Start, Termination, and Output File

It is possible to wait for starting of a request, mail when a request terminates by specified at the time of request submission by qlogin(1) command. In addition, it is possible to wait for a termination of by the qwait (1) command.

However, it is not possible to create output file, because the output of the interactive request is displayed as the output of the qlogin.

### 3.6. Run MPI Request

#### 3.6.1. Run under the NEC MPI Environment

This function is available only for the environment whose execution host is SX-Aurora TSUBASA system.

In NQSV, it is possible to select either Hydra or MPD as process manager to execute NEC MPI jobs for a queue to submit.

\* Queue setting can be confirmed by display of qstat -Qf (NEC MPI Process Manager).

To run a job with NEC MPI, submit a request as follows.

- Option in submitting  
Specify necmpi to -T option of the qlogin(1) command.

An example to execute MPI program, mpi\_prog executed into 2 logical host which has 4 VE, 1 processes per a VE, is shown as follows.

```
client $ qlogin -q ique -T necmpi -b 4 --venum-lhost=8
Request 28.host1 submitted to queue: ique.
Waiting for 28.host1 to start.
ehost $ source /opt/nec/ve/mpi/1.0.1/bin/necmpivars.sh
ehost $ mpirun -nnp 4 -nn 2 -ve 0-3 ./mpi_prog
```

#### [Notes]

It is assumed that mpi\_prog is located on the user's home directories on the execution host.

### 3.6.2. Run under the OpenMPI Environment

On interactive request, in case to execute program using OpenMPI, request is submitted and mpirun is executed as follows.

- Submit Option

Specify openmpi with qlogin(1) command -T option.

- mpirun

To execute MPI program, use mpirun command of OpenMPI.

Specify \${NQSVMPIOPTS} or \${NQSII\_MPIOPTS} as mpirun parameter. (With this parameter, job execution host is provided for MPI.)

[Info]

In case more than one network I/F are available in the jobs, network I/F the jobs use can be designated by "-mca btl\_tcp\_if\_include" option.

For example when executing jobs in Docker container, please designate an overlay network I/F by "-mca btl\_tcp\_if\_include" option.

For example, to execute OpenMPI program "mpi\_prog" by creating 64 MPI processes, 2 jobs(nodes), execute as follows.

```
client $ qlogin -q iq1 -T openmpi -b 2
Request 173.bsv submitted to queue: iq1.
Waiting for 173.bsv to start.
-bash-$ mpirun ${NQSVMPIOPTS} -np 64 ${HOME}/mpi_prog
```

#### [Notes]

- Multiple jobs cannot be executed on a single execution host in the case of OpenMPI.
- When the job does not complete normally, such as when the resource limit is exceeded or when a node failure occurs, temporary files may remain under /tmp on the master node. There is no problem with deleting them.
- When using OpenMPI, do not use the -hostfile \${PBS\_NODEFILE} option. Using this option may result in the job using CPUs that have not been allocated by NQSV.

### 3.6.3. Run under the Intel MPI Environment

On interactive request, in case to execute program using Intel MPI, request is submitted and mpirun is executed as follows.

- Submit Option



Specify `intmpi` with `qlogin(1)` command `-T` option.

- `mpirun`

The `mpirun` command of Intel MPI is used to execute MPI program.

Specify `#{NQSV_MPIOPTS}` or `#{NQSII_MPIOPTS}` as `mpirun` parameter. (With this parameter, job execution host is provided for MPI.)

For example, to execute Intel MPI program "`mpi_prog`" by creating 64 MPI processes, 2 jobs(nodes), as batch request, execute as follows.

```
client $ qlogin -q iq2 -T intmpi -b 2
Request 174.bsv submitted to queue: iq2.
Waiting for 174.bsv to start.
-bash-$ mpirun #{NQSV_MPIOPTS} -np 64 ${HOME}/mpi_prog
```

#### [Notes]

- When a job is deleted by `qdel(1)` command during job execution, the following message may appear in the job execution result file (standard error output). But there is no problem in the operation.

```
[proxy:0:1@host1] proxy_upstream_control_cb (../../../../src/pm/i_hydra/proxy/proxy_cb.c:69):
assert (!closed) failed
[proxy:0:1@host1]HYDI_dmxdmx_poll_wait_for_event
../../../../src/pm/i_hydra/libhydra/demux/hydra_demux_poll.c:80): callback returned error
status
[proxy:0:1@host1]main (../../../../src/pm/i_hydra/proxy/proxy.c:964): error waiting for event
```

### 3.6.4. Run under the MVAPICH2 Environment

On interactive request, to execute a job using MVAPICH2, submit request as follows.

- Submit Option

Specify `mvapich` to with `qlogin(1)` command `-T` option.

- `mpiexec`

`mpiexec` command of MVAPICH2 is used to execute MPI program.

Specify `#{NQSV_MPIOPTS}` or `#{NQSII_MPIOPTS}` as `mpiexec` parameter. (With this parameter, job execution host is provided for MPI.)

For example, to execute MVAPICH program by creating 64 MPI processes, 2 jobs (nodes), execute as follows.

```
client $ qlogin -q iq3 -T mvapich -b 2
Request 175.bsv submitted to queue: iq3.
Waiting for 175.bsv to start.
mpiexec ${NQSV_MPIOPTS} -np 64 ${HOME}/mpi_prog
```

### 3.6.5. Run under the Platform MPI Environment

On interactive request, in case to execute program using Platform MPI, request is submitted and mpirun is executed as follows.

- Submit Option

Specify pltmpi with qlogin(1) command -T option.

- mpirun

In case to submit request to queue is set, mpirun command of Platform MPI is used to execute MPI program.

Specify \${NQSV\_MPIOPTS} or \${NQSII\_MPIOPTS} as mpirun parameter. (With this parameter, job execution host is provided for MPI.)

For example, to execute Platform MPI program "mpi\_prog" by creating 64 MPI processes, 2 jobs(nodes), as batch request, execute as follows.

```
client $ qlogin -q iq2 -T pltmpi -b 2
Request 174.bsv submitted to queue: iq2.
Waiting for 174.bsv to start.
-bash-$ mpirun ${NQSV_MPIOPTS} -np 64 ${HOME}/mpi_prog
```

#### [Notes]

- When a job is deleted by qdel(1) command during job execution, the following message may appear in the job execution result file (standard error output). But there is no problem in the operation.

-15: -c: line 0: unexpected EOF while looking for matching ``"

-15: -c: line 1: syntax error: unexpected end of file

In addition, the following temporary files created at the time of job execution may remain under /tmp. Please delete them manually.

mpijob\_user1\_0, mpiafQeIuGG

## 4. Job Operation

### 4.1. Job State Check

Job is an execution unit of user jobs, and is created on job servers when an execution host is assigned to a request and staging is started.

#### 4.1.1. Check of Basic Information

The `qstat(1)` command with `-J` option is used to check batch job status. A job is specified as job ID(request ID + Job number).

An example is shown below.

[Specify Job ID]

\$ qstat -J 0:72.host.example.com								
JNO	RequestID	EJID	Memory	CPU	JSVNO	ExecutionHost	UserName	Exit
0	72.host.example	11366	1.93M	0.23	10	exechost.exempl	user1	-

[Not Specify Job ID]

\$ qstat -J								
JNO	RequestID	EJID	Memory	CPU	JSVNO	ExecutionHost	UserName	Exit
0	72.host.example	11366	1.93M	0.23	10	exechost.exempl	user1	-
1	72.host.example	11366	2.23M	3.15	11	exechost.exempl	user1	-
2	72.host.example	11366	1.01M	2.45	12	exechost.exempl	user1	-

If the job is executed on the SX-Aurora TSUBASA system, by specifying the `-e` option for the `qstat -J` command, you can check the VE information in the job. When the `-e` option is specified, only the job on the VH are displayed. In the following descriptions, the CPU on VEs is referred to as "VECPU" and the memory on VEs are referred to as "VE memory."

An example is shown below.

\$ qstat -Je								
JNO	RequestID	EJID	VEMemory	VECPU	JSVNO	VectorIsland	UserName	Exit
0	72.host.example	11366	1.00GB	2.00	10	exechost.exempl	user1	-

[Note]

The value of the item "Memory" includes file cache if "enable\_memory\_cgroup" is "on" in `/etc/opt/nec/nqsv/nqsd.conf`.

#### 4.1.2. Check of Detail Information

Execute the `qstat(1)` command with `-f` option when more detailed information of a batch job is required.

\$ qstat -J -f 0:72.host.example.com								
--------------------------------------	--	--	--	--	--	--	--	--

```
Request ID : 72.host.example.com
Batch Job Number = 0
Execution Job ID = 11366
User Name = user1
User ID = 111
Group ID = 100
Job Server Number = 10
Job Server Name = JobServer0033
Execution Host = exechost.example.com
Exit Code = (none)
Resources Information:
Memory = 1.932587MB
CPU Time = 0.225019S
Accumulated CPU Time = 0.225019S
Remaining CPU Time = UNLIMITED
```

By specifying the `-f` option for the `qstat -Je` command, you can check the detailed VE information in the job.

An example is shown below.

```
$ qstat -J -f 0: 72.host.example.com
Request ID : 72.host.example.com
Batch Job Number = 0
Execution Job ID = 11366
User Name = user1
User ID = 111
Group ID = 100
Job Server Number = 10
Job Server Name = JobServer0033
Vector Island = exechost.example.com m
VE Node = 0-3, 5
Exit Code = (none)
Resources Information:
VEMemory = 1.000000GB
VECPU Time = 2.000000S
```

Display Scheduler Message in `qstat(1)` with `-f` option. This message shows the pending reason of a job.

An example is shown below.

```
$ qstat -f [Request-ID]
:
Scheduler Message:
  Not enough resources for 1 of 2 jobs in a request on available 1 hosts.
  1 of 2 hosts cannot be used due to the scheduling condition mismatching.
  Reason:
    - JSV LINKDOWN
```

[Note]

The value of the item “Memory” in the section “Resources Information:” includes file cache if

“enable\_memory\_cgroup” is “on” in /etc/opt/nec/nqsv/nqsd.conf.

#### 4.1.3. Check of Job information of hybrid request

There is some specific feature to check the information of hybrid request.

Please refer to 16.3 Hybrid Request for detail.

#### 4.1.4. Customizing Information

##### (1) Time format

Time data in the output of qstat(1) command is displayed by seconds as default. It is also displayed in the format of d+hh:mm:ss by specifying -d option. This option is valid in following items.

- CPU time (CPU)
- Accumulated CPU time (with -c option, ACCPU)
- Rest CPU time (with -m option, RCPU)

An example of specifying -d option is as below.

\$ qstat -J -d								
JNO	RequestID	EJID	Memory	CPU	JSVNO	ExecutionHost	UserName	Exit
0	1249. host1. exam	19710	1.91M	1+05:21:10	1020	host1. example. c	user1	-
1	1249. host1. exam	19710	1.90M	1+05:01:10	1030	host2. example. c	user1	-

(About details of qstat -d option, please refer to 6.4. Time Display Format.)

##### (2) Long format

Sometimes there are cases that information such as execution host name is cut off in the basic information display of qstat(1) command (without -f option) because the viewable number of character is fixed. By using -l option, it is possible to show all information without being cut.

All information are shown without cutting off. At this time, some displayed contents may run off.

(About details of qstat -l option, please refer to 6.5. Displaying All Information.)

##### (3) Selecting Item and Sorting

It is possible to display information by selecting and customizing each item output by -F option of qstat(1) command. However, it is impossible with -f option for long format.

Items which can be specified for request information are as follows.

item	Contents	Summary display format
jno	Job number	JNO
rid	Request identifier	RequestID
jid	Job identifier	JobID
ejid	Execution job identifier (job identifier on an execution	EJID

	host)	
mem	Memory size used by job	Memory
cpu	CPU time of processes during execution	CPU
acpu	Accumulated CPU time (include exited processes)	ACCPU
rcpu	Rest CPU time	RCPU
jsvno	Job server number	JSVNO
ehost	Execution host	ExecutionHost
euser	Execution user	UserName
ecode	Exit code (hexadecimal format)	Exit

About details of the `qstat(1)` -F option, please refer to 6.2. Customizing Information.

It is possible to sort information by item using -o option or -O option of `qstat(1)` command. About details of the `qstat(1)` -o option and -O option (output at the time of an error), please refer to 6.3. Sorting Information.

## 4.2. Signal Send

It is possible to send a signal to a job by the `qsig(1)` command specified a job ID. The signal is sent to all processes in the specified job.

```
$ qsig -s SIGKILL 0:72.host1
Job 0:72.host1 was sent signal SIGKILL.
```

When "-s *signal*" is not specified, SIGTERM will be sent as default.

It is available in case of forcibly aborting a batch job.

## 5. Network Request Operation

Network Request is a request that transmits input/output files or result files of batch jobs between client hosts and execution host.

### 5.1. Status Check of Network Request

Check status of a network request by the `qstat(1)` command with `-T` option. A network request is specified using batch request ID that is a file staging target, or parent request ID.

Specify `-T` option and parent request ID to the `qstat (1)` command. An example is shown below.

```
$ qstat -T 92.host1
```

RequestID	UserName	Pri	Dir	STGNO	StagingFile	STT
92.host1	user1	0	IN	0	host1:/home/user1/dir1/infile1	QUE
92.host1	user1	0	IN	1	host1:/home/user1/dir1/infile1	QUE

Column `STT` will show request statuses. The meanings of the displayed request states are as follows:

- `RUN.....` Executing
- `QUE.....` Waiting for execution
- `WAT.....` Waiting for start time

See the `qstat(1)` command for more information on items other than `STT`.

When specifying only `-T` option to `qstat(1)` command, information on all network requests registered at present is displayed.

```
$ qstat -T
```

RequestID	UserName	Pri	Dir	STGNO	StagingFile	STT
90.host1	user1	0	IN	0	host1:/home/user1/dir0/infile0	QUE
90.host1	user1	0	IN	1	host1:/home/user1/dir0/infile0	QUE
91.host1	user1	0	IN	0	host1:/home/user1/dir1/infile1	QUE
92.host1	user1	0	IN	0	host1:/home/user1/dir1/infile1	QUE
92.host1	user1	0	IN	1	host1:/home/user1/dir1/infile1	QUE

Execute the `qstat(1)` command with `-T -f` option when more detailed information of a network request is required.

```
$ qstat -T -f 92.host1
Request ID: 92.host1
  User Name = user1
  User ID   = 500
  Group ID  = 501
  Network Request Priority = 0
  Staging Direction = STAGE_IN
  Staging Number   = 0
```

Staging File	= host1:/home/user1/dir1/infile1.txt
Staging Method	= Internal
State	= Queued

See the `qstat(1)` command for more information on each item.

It is possible to display information by selecting and customizing each item output by `qstat(1)` command. Please specify items to be displayed with `-F` option of `qstat(1)` command. Items that can be specified for network request information are as follows.

Item	Contents	Summary display style
rid	Parent request identifier	RequestID
own	Owner	UserName
pri	Network request priority	Pri
stgdir	Staging direction	Dir
stgno	Staging file number	STGNO
stgf	Staging file	StagingFile
stt	Network request state	STT

About details of `-F` option, please refer to 6.2. Customizing Information.

It is also possible to sort information by any item using `-o` option or `-O` option of `qstat(1)` command. About details of the `qstat(1)` command `-o` option and `-O` option (output at the time of an error), please refer to 6.3. Sorting Information.

Sometimes there are cases that information such as execution host name is cut off in the basic information display of `qstat(1)` command (without `-f` option) because the viewable number of character is fixed. By using `-l` option, it is possible to show all information without being cut. About details of `-l` option, please refer to 6.5. Displaying All Information.

## 5.2. Network Request Status Transition

A network request is automatically created when the batch request becomes in `STAGING` or `EXITING` state and is submitted to the network queue. A network request is in `QUEUED` state immediately after it is submitted. Then, it is scheduled by the batch server, and becomes in `RUNNING` state. The file staging is executed when a network request becomes in `RUNNING` state. (Please refer to [Management] Network Queue Configuration (Forwarding Host).)

A network request will be in `WAITING` state if the file staging is failed. The staging-file will be staged again after an interval set to the batch server or the network queue when a network request is in `WAITING` state. (Please refer to [Management] Network Queue Configuration (Waiting Interval).)



### **5.3. Network Request Delete**

A network request is deleted along with the parent request deletion. Delete the parent request in order to delete a network request. It is not possible to delete only network requests. (Please refer to 1.5. Batch Request Delete.)

## 6. System Information Display

### 6.1. How to Display System Information

Execute the `qstat(1)` command to see NQSV system information.

Each information is displayed by execution of the `qstat` command with the following options.

Information	Option	Additional option
Batch server host	-B	-l, -F, -o, -O, -d, -x
Scheduler	-D	-l, -F, -o, -O, -x
Job server	-S	-l, -F, -o, -O, t, -x
Execution host	-E	-l, -F, -o, -O, -t
Node group	-G	-l, -F, -o, -O, -x
Queue	-Q	-l, -F, -o, -O, -d, -t, --group= <i>group_name</i>
Request	No option	-l, -F, -o, -O, -d, -x, --group[= <i>group_name</i> ]
Parametric request	-R	-l, -F, -o, -O, -x
Job	-J	-l, -F, -o, -O, -d
Network request	-T	-l, -F, -o, -O
Limit per Group and User	--limit	--group= <i>group_name</i>
Custom Resource	--custom	
Template	--template	-l
VE node	--venode	

Execute the `qstat(1)` command with each option and `-f` option when more detailed information is required.

According to the type of information, the following options can be specified

- The `-F` option which customizes the output item (6.2. Customizing Information)
- The `-o, -O` option which sorts output information by an item (6.3. Sorting Information)
- The `-d` option which changes the format of the time data (6.4. Time Display Format)
- The `-l` option which cancels the restriction of number of characters (6.5. Displaying All Information)
- The `-t` option which displays information about job server (execution host) that is down (6.6. Displaying All Job Serves (Execution Hosts) Information)
- The `--group` option about group (8.2. Display of Information about Group)

Special User privilege or higher is necessary to check request information of other users. User

privilege or higher is necessary to check other information.

Please refer to the reference of the qstat(1) command about a peculiar option of each information on other than above.

## 6.2. Customizing Information

It is possible to display information by selecting and customizing each item output by qstat(1) command. Please specify items to be displayed with -F option of qstat(1) command as following format. Each item will be displayed according to the order specified with -F option.

```
qstat< option> -F item[,item, ...]
```

An example to display batch request information in summary display in order of submitted queue name (quenm), request ID (rid), request status (stt) and execution host (ehost) is as follows.

```
$ qstat -F quenm,rid,stt,ehost
Queue      RequestID      STT ExecutionHost
-----
nqstest    80536.host1    RUN exec.example.co
nqstest    80542.host1    QUE -
nqstest    80543.host1    QUE -
```

Items which can be specified are as follows.

Display option	Item	Summary display format	Contents
-B (Batch server Host information)	bsvhost	BatchServer	Batch server host name
	mid	MachineID	Mechanical ID
	umax	UMAX	Request submit limit per user
	gmax	GMAX	Request submit limit per group
	rrlm	RRL	Routing queue run limit
	nrlm	NRL	Network queue run limit
	tot	TOT	Number of batch requests managed by the batch server
	arr	ARR	Number of requests for each state
	wai	WAI	
	gqd	GQD	
	que	QUE	
	run	RUN	
	ext	EXT	

Display option	Item	Summary display format	Contents
	hld	HLD	
	sus	SUS	
	fwd	FWD	
	bstt	Status	Batch server status
-D (Scheduler information)	schid	SCHID	Scheduler ID
	schnm	SCHName	Scheduler name
	schost	ExecutionHost	Execution host
	ques	BindQueue	Number of bound execution queue
	quenm*	BindQueueName	Bound execution queue name
-S (Job server information)	jsvno	JSVNO	Job server number
	jsvnm	JobServerName	Job server name
	bsvhost	BatchServer	Batch server host name
	ehost	ExecutionHost	Execution host
	link	LINK	Link state
	bind	BIND	Binding state
	quenm	Queue	Queue name
	jobs	Jobs	Number of batch jobs
	ldavg1	Load	Load average of most recent one minute
	cpuavg1	CPU	CPU average of most recent one minute
-E (Running host information)	ehost	ExecutionHost	Executionhost
	bsvhost	BatchServer	Batch server host name
	osnm	OS	Name of operating system
	rel	Release	OS release number
	hwnm	Hardware	Hardware name
	ldavg1	Load	Load average of most recent one minute
	cpuavg1	CPU	CPU average of most recent one minute
	ucpu*	Used_CPUs	Used CPU number
	fcpu*	Free_CPUs	Free CPU number
	umem1*	Used_MEM1	Used memory size
	fmem1*	Free_MEM1	Free memory size
	uswap1*	Used_SWAP1	Used swap size
	fswap1*	Free_SWAP1	Free swap size
	quenm*	QueueName	Queue name

Display option	Item	Summary display format	Contents
	stt	STT	Status of the execution host
-G (Node group Information)	ngrpnm	NodeGroup	Node group name
	ngrptype	Type	Type of node group
	bsvhost	BatchServer	Batch server host
	comment	Comment	Comment
	jsvs	JSVs	Number of job servers
	quenm	BindQueue	Queue name
-Q [-e] (Execution queue information)	quenm	QueueName	Queue name
	schid	SCH	Scheduler ID bound to the queue
	jsvs	JSVs	Number of job servers bound to the queue
	stt1	ENA	Queue state (Submit Possible/Not Possible)
	stt2	STS	Queue state (Execute Possible/Not Possible)
	pri	PRI	Queue priority
	qtot	TOT	Number of batch requests submitted in a queue
	arr	ARR	Number of batch requests for each state
	wai	WAI	
	que	QUE	
	prp	PRP	
	run	RUN	
	por	POR	
	ext	EXT	
	hld	HLD	
	hol	HOL	
	rst	RST	
	sus	SUS	
	mig	MIG	
	stg	STG	
	chk	CHK	
	ehost*	ExecutionHost	Execution host bound to the queue
	attbl*	ATTBL	Attach request Possible/Not Possible
-Q -r	quenm	QueueName	Queue name

Display option	Item	Summary display format	Contents
(Routing queue information)	stt1	ENA	Queue state (Submit Possible/Not Possible)
	stt2	STS	Queue state (Execute Possible/Not Possible)
	pri	PRI	Queue priority
	rqlm	RLM	Run limit for each queue
	qtot	TOT	Number of batch requests submitted in a queue
	arr	ARR	Number of batch requests for each state
	wai	WAI	
	que	QUE	
	hld	HLD	
	trs	TRS	
-Q -N (Network Queue information)	quenm	QueueName	Queue name
	stghost	StagingMachine	Staging host name
	stt1	ENA	Queue state (Submit Possible/Not Possible)
	stt2	STS	Queue state (Execute Possible/Not Possible)
	pri	PRI	Queue priority
	nqlm	RLM	Run limit for each queue
	qtot	TOT	Number of batch requests submitted in a queue
	wai	WAI	Number of batch requests for each state
	que	QUE	
	run	RUN	
-Q -i (Interactive queue information)	quenm	QueueName	Queue name
	schid	SCH	Scheduler ID bound to the queue
	jsvs	JSVs	Number of job servers bound to the queue
	stt1	ENA	Queue state (Submit Possible/Not Possible)
	stt2	STS	Queue state (Execute Possible/Not Possible)
	pri	PRI	Queue priority

Display option	Item	Summary display format	Contents
	qtot	TOT	Number of interactive requests submitted in a queue
	arr	ARR	Number of interactive requests for each state
	wai	WAI	
	que	QUE	
	prp	PRR	
	run	RUN	
	por	POR	
	ext	EXT	
	hld	HLD	
	sus	SUS	
	ehost*	ExecutionHost	Execution host bound to the queue
	attbl*	ATTBL	Attach request Possible/Not Possible
no option (Request Information)	rid	RequestID	Request ID
	reqnm	ReqName	Request name
	own	UserName	Owner
	group	GrpName	Group name
	quenm	Queue	Queue name
	pri	Pri	Request priority
	stt	STT	Request state
	stall	S	Stall state
	mem	Memory	Memory size used by batch request
	cpu	CPU	CPU time of processes during execution
	acpu	ACCPU	Accumulated CPU time (include exited processes)
	elaps	Elapse	Elapsed time
	relaps	RElapse	Remaining Elapsed time
	rflg	R	Re-execution Possible/Not Possible
	hflg	H	Hold Possible/Not Possible
	mflg	M	Job Migration Possible/Not Possible
	jobs	Jobs	Number of jobs residing in request
	ehost*	ExecutionHost	Execution host
	sdate*	Date (SUBMIT)	Date and time when request was submitted
	qdate*	Date (QUEUED)	Date and time for start of scheduling

Display option	Item	Summary display format	Contents
	rdate*	Date (RUNNING)	Date and time for start of request
	att*	Att	Attach connection Exist/Not exist
-R (Parametric Request information)	rid	RequestID	Parametric request ID
	reqnm	ReqName	Request name
	own	UserName	Owner
	group	GrpName	Group name
	quenm	Queue	Queue name
	pri	Pri	Request priority
	stt	STT	Request state
	rflg	R	Re-execution Possible/Not Possible
	hflg	H	Hold Possible/Not Possible
	mflg	M	Job Migration Possible/Not Possible
	jobs	Jobs	Number of jobs residing in sub-request
	tot	TOTAL	Number of sub-requests
	active	ACTIVE	Number of existing sub-requests
	done	DONE	Number of sub-requests which have ended
-J (Batch job information)	jno	JNO	Job number
	rid	RequestID	Request ID
	jid*	JobID	Job identifier (Job number + Request ID)
	ejid	EJID	Memory size used by batch jobs
	mem	Memory	Memory size used by batch jobs
	cpu	CPU	CPU time by batch jobs
	acpu	ACCPU	Accumulated CPU time by batch jobs
	rcpu	RCPU	Remaining CPU time for batch jobs
	jsvno	JSVNO	Job server number
	ehost	ExecutionHost	Execution host
	euser	UserName	Batch job executing user
	ecode	Exit	Batch job exit code used hexadecimal For detailed values, refer to 1.16 qstat(1) in NEC Network Queuing System V (NQSV) User's Guide [Reference].
-T	rid	RequestID	Parent request identifier



Display option	Item	Summary display format	Contents
(Network Request information)	own	UserName	Owner
	pri	Pri	Network request priority
	stgdir	Dir	Staging direction
	stgno	STGNO	Staging file number
	stgf	StagingFile	Staging file
	stt	STT	Network request state

\* This item can be specified in case of displaying information using qstat(1) command -F option.

When -F option and -f option (Detailed information) are specified at the same time, an error is not output, and -f option is given priority and -F option is ignored. (Detailed information is output as a result.)

### 6.3. Sorting Information

It is possible to sort information by any item using -o option or -O option of qstat(1) command. The items for sorting are same as the items shown on the table in 6.2. Customizing Information. More than one item can be specified with delimited by comma ",".

`-o item[,item, ...]`

Information will be sorted in ascending order with the key specified in *item*.

`-O item[,item, ...]`

Information will be sorted in descending order with the key specified in *item*.

If these options are specified more than once, the sorted results are further sorted according to the order of items specified. It is also possible to specify like ascending sort for the first key and descending sort for the second key by combination of -o option and -O option. It can be used with -F option at the same time, too.

As a sort method, small and large between the numbers are compared if the target information is numerical number and alphabetical order is used in case of comparing characters.

[Example]

- (1) `qstat -o rid,quenm,stt`
- (2) `qstat -o rid -o quenm -o stt`
- (3) `qstat -o rid -O quenm -o stt`

The display result of (1) and (2) are same and sorted with the order of "rid", "quenm", "stt" in ascending. In (3), the display result is sorted with the key "rid" by ascending order, next sorted with "quenm" in

descending order, and then sorted with "stt" in ascending order.

An error will be shown in the cases below.

- No target items of sorting are specified.
- Items not supported are specified for the target of sorting.
- Items which do not match to each display option are specified for the target of sorting.

When -o, -O option and -f option (Detailed information) are specified at the same time, an error is not output, and -f option is given priority and -o, -O option is ignored. (Detailed information is output as a result.)

#### 6.4. Time Display Format

Time data in the output of qstat(1) command is displayed by seconds as default. It is also displayed in the format of d+hh:mm:ss by specifying -d option. This feature can be used in case digit number of the time data is very large and it is difficult to understand the time immediately by seconds.

The -d option is valid in following items.

- Elaps time (Elaps)
- CPU time (CPU) and Accumulated CPU time (ACCPU) in batch request information (No option)
- CPU time (CPU) and Accumulated CPU time (ACCPU) in batch job information (-J option)

Items above are also displayed in d+hh:mm:ss format with detailed information display with -f option.

[Example]

CPU time (CPU) and Elaps time(Elapse) in batch request information

\$ qstat - d													
RequestID	ReqName	UserName	Queue	Pri	STT	S	Memory	CPU	Elapse	R	H	M	Jobs
80536.host1	STDIN	nqstest	que1	0	QUE	-	7.62M	1+05:21:10	1+16:30:12	Y	Y	Y	1

CPU time (CPU) in batch job information

\$ qstat -J -d									
JNO	RequestID	EJID	Memory	CPU	JSVNO	ExecutionHost	UserName	Exit	
0	1249.bsvhost	19710	1.91M	1+05:21:10	1020	host1	nqstest	-	
1	1249.bsvhost	28094	1.90M	1+05:01:10	1030	host2	nqstest	-	

## 6.5. Displaying All Information

Sometimes there are cases that information such as execution host name is cut off in the basic information display of `qstat(1)` command (without `-f` option) because this information is a summary and its viewable number of character is fixed. By using `-l` option, it is possible to show all information. For example, it enables scripts which make use of output of `qstat` command to be executed without problems.

If specified with `-f` option (Detailed information) of `qstat` command, it is just ignored without error message. The `-l` options is ignored and information with `-f` option will be displayed. (Detailed information is output as a result.)

Examples of specifying `-l` option and without `-l` option are as follows.

\$ qstat													
RequestID	ReqName	UserName	Queue	Pri	STT	S	Memory	CPU	Elapse	R	H	M	Jobs
80333.host1	STDIN	nqstest	que1	0	QUE	-	0.00B	0.00	0	Y	Y	Y	1

\$ qstat -l													
RequestID	ReqName			UserName			Queue						
Pri	STT	S	Memory	CPU	Elapse	R	H	M	Jobs				
80333.host1.example.com	STDIN			nqstest1			que1						
0	QUE	-	0.00B	0.00	0	Y	Y	Y	1				

When the `-l` option specified, though displayed contents will run off the fixed number of characters, all information are shown without cutting off.

## 6.6. Displaying All Job Serves (Execution Hosts) Information

Information of job servers or execution hosts in the output of `qstat(1)` is about linked up job servers. However, when the `-t` option is specified, linked down job servers are displayed.

The `-t` option is valid with the `-S` option (for job server information) and the `-E` option (for execution host information). For job server information (`-E` option), job server number and job server state information are also displayed. When specified together with `-f` option, linked down job servers' information is also displayed, which is limited to the information.

For queue information (`-Q` option), when specified together with `-t` option, queues that is bound to linked down job servers are also displayed.

## 7. Workflow

NQSV workflow is made as a shell script described some job execution flow including request submission. By executing the shell script (workflow script) on a NQSV client host, NQSV treats the series of requests as a set of requests for the workflow.

For the NQSV workflow, the following functions are enabled.

- Sequential order can be set to the requests in a workflow at the submission. The requests are executed as specified order.
- In a workflow script, loops and conditionals can be described as a shell script. By the termination code of a request, the execution flow can be controlled.
- By creating a workflow script, the same execution flow of the requests can be repeated.
- The requests can be referenced or deleted by a workflow unit.

### 7.1. Environment for the workflow

There are some conditions for NQSV environment to use NQSV workflow.

- In a workflow script, all requests are submitted to the same batch server.
- All queues to which the requests in a workflow script are submitted must be bound to the same JobManipulator.

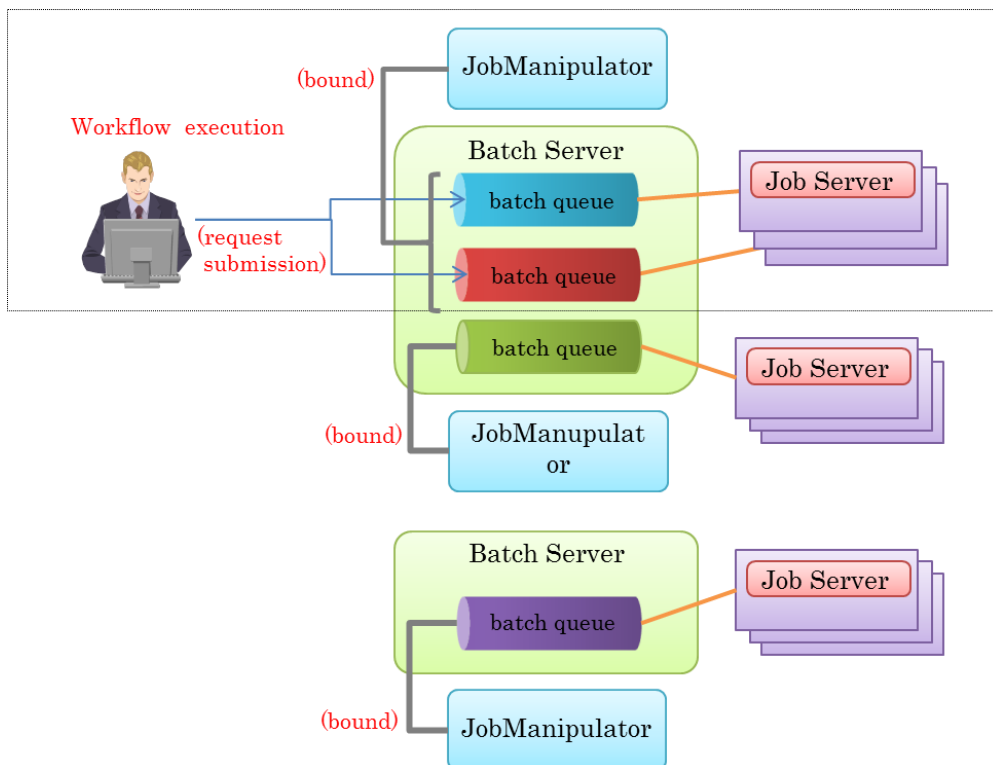


Figure 7-1 : Execution of a workflow

## 7.2. Workflow description

NQSV workflow is described as a shell script on a client host. The shell script described as an NQSV workflow is called as "workflow script".

In a workflow script, NQSV CUI commands as qsub, qstat, qdel, etc. can be used to submit, reference control the requests.

### 7.2.1. Simple example

For example, the workflow script that submit a request and waits its termination can be described as follows.

```
#!/bin/bash
qsub -q batch1 ./req_A
qwait req_A
```

In the example above, req\_A shows the job script file that is submitted to the batch queue, batch1. In a job script file, qsub options can be specified as #PBS lines. (Please refer to 1.1. Batch Request Create.)

In a workflow script, request name can be used to control a request in place of the request ID. In the example above, the request name "req\_A" is used at the qwait command. (When a request name is not specified explicitly by qsub -N option, the specified job script file name is used as the request name.)

The commands that a request name can be used in place of the request ID in a workflow script are as follows.

- qdel
- qhold
- qrls
- qalter
- qmove
- qrerun
- qstat
- qwait
- qwait2

When a request name is used to control a request in a workflow script, all requests in the workflow must have unique request names.

### 7.2.2. Execution order

To specify an execution order in a workflow, please use qsub --after option.

For example, to start request B (req\_B) after termination of request A (req\_A), the requests are submitted as follows.

```
#!/bin/bash
qsub -q batch1 ./req_A
qsub -q batch2 --after req_A ./req_B
```

In a workflow script, the requests are submitted as described above, the execution order of the request A and B is scheduler by scheduler (JobManipulator).

At the qsub --after option, request name can be specified in place of request ID.

When you want to execute a request after terminations of two or more other requests, please specify requests separated by comma (,) to the qsub --after option.

```
#!/bin/bash
qsub -q batch1 ./req_A
qsub -q batch2 ./req_B
qsub -q batch1 --after req_A, req_B ./req_C
```

In the example above, the request C (req\_C) is scheduled to start after termination of both of the requests, req\_A and req\_B.

When two or more requests are submitted by specifying the execution order and the output files of the preceding request are used as the input files of the following request, the files must be shared by the requests using shared file system among execution hosts.

### 7.2.3. Parallel execution

In a workflow, to start two or more requests simultaneously, use qsub --parallel option.

```
#!/bin/bash
qsub -q batch1 --parallel ./req_A ./req_B
```

When --parallel option is used with qsub command and two or more job script files are specified as the qsub's last arguments, the specified job scripts are submitted as the requests and they are scheduled to start simultaneously.

In above case, the specified "req\_A" and "req\_B" are submitted as the parallel requests and they are scheduled to start simultaneously.

When the --parallel option is specified with qsub command, the request connection function (please refer to 1.2.22 Request Connection Function) cannot be used.

If a hybrid request (a request submitted using the --job-separator option or --- option with the qsub command and requests a different resource for each job) is submitted as a concurrent request (a request submitted using the --parallel option with the qsub command), the request will not be scheduled.

### 7.3. Workflow execution

The workflow script described above must be executed by wstart command on a client host. To execute the wstart command, specify a workflow script file as follows.

```
$ /opt/nec/nqsv/bin/wstart wfl.sh
Workflow 84_bsvhost is start.
Request 163.bsvhost submitted to queue: batch1.
Request 164.bsvhost submitted to queue: batch2.
:
```

By executing the wstart command, wstart makes connection with the batch server and notices the start of a workflow to the batch server. The batch server returns a workflow ID to the wstart and the wstart prints the workflow ID to STDOUT and then executes the specified workflow script. The requests submitted in a workflow script that is executed by wstart are managed as related to the workflow created by the wstart.

The workflow ID can be referenced by the environment variable NQSV\_WFID or NQSII\_WFID (they have same value) in the workflow script.

The STDOUT and STDERR of the execution of the workflow script are output to the terminal in which the wstart has been invoked. And the STDIN of the terminal is passed to the workflow script.

Only batch request can be managed as the request related to a workflow in the batch server. Interactive request cannot be a part of a workflow.

The workflow created by wstart command is managed as a workflow in the batch server while the workflow script is executing or a request that is submitted in the workflow script exists. And while a workflow exists, workflow can be referenced and controlled as follows.

### 7.4. Reference of workflow

A workflow created by wstart command can be referenced by wstat command. By executing wstat command without arguments, the list of the workflows which the execution user owns is displayed.

```
$ wstat
WFL-ID      Request Owner  Stat
-----
84_bsvhost      3 user1  RUN
86_bsvhost      2 user1  RUN
```

wstat displays workflow ID, number of requests in the workflow, the owner of the workflow and state of the workflow (RUN).

By executing wstat command with a workflow ID as an argument, wstat displays the requests related

to the workflow.

```
$ wstat 84_bsvhost
WFL-ID      RequestID  RequestName  Stat  Exit
-----
84_bsvhost  163. bsvhost  Req_A        EXT   0
84_bsvhost  164. bsvhost  Req_B        EXT   9
84_bsvhost  165. bsvhost  Req_C        RUN   -
```

In this case, wstat displays workflow ID, request ID, request name, state of the request and exit status of the request for each request which is related to the specified workflow.

The displayed exit status is the exit status of the job whose job number is 0. For the request which does not terminated yet, "-" is displayed.

## 7.5. Control of workflow

### 7.5.1. Workflow delete

A workflow created by wstart command can be deleted by wdel command. By executing the wdel command specifying the target workflow ID, the specified workflow is deleted.

```
$ wdel 84_bsvhost
Workflow 84_bsvhost was deleted.
```

By deleting a workflow, all requests submitted in the workflow script are deleted and workflow script is also terminated if it is running.

### 7.5.2. Request cancel by preceding request error

If a preceding request terminated abnormally, following requests can be canceled automatically. Specify "qsub --cancel-after" option for preceding request submitting.

Request is treated as abnormal termination in the following case.

- Exit code is not 0.
- Request has been deleted by qdel(1).
- Request terminated because HW error.

For example, you can describe a workflow script as follows, in case that you want to execute the request B (req\_B) after termination of the request A (req\_A), and that you want to cancel the request B (req\_B) if the preceding request A (req\_A) terminates abnormally.

```
#!/bin/bash
qsub -q batch1 --cancel-after ./req_A
qsub -q batch2 --after req_A ./req_B
```

In the example above, the workflow performs as follows.

1. Submit req\_A and req\_B.
2. Start req\_A.
3. Wait termination of req\_A.



- in case of normal termination : Start req\_B.
- **in case of abnormal termination** : **Delete req\_B.**

Further, requests canceled automatically are all following requests as well as just after error request.

Also wstart command with "--cancel-after" option can specify "--cancel-after" to qsub command all together in workflow script.

Below is an example.

```
$ /opt/nec/nqsv/bin/wstart --cancel-after wfl.sh
```

```
wfl.sh
```

```
#!/bin/bash
qsub -q batch1 ./req_A          # "--cancel-after" is specified implicitly.
qsub -q batch2 --after req_A ./req_B  # "--cancel-after" is specified implicitly.
```

## 8. Group of Request

### 8.1. Group of Request and How to Submit

When submitting a request, information of group as well as user are retained in the request information. This group usually becomes primary group of the user who submitted the request. In addition, jobs are executed by primary group on execution host.

In case that Designated Group Execution Function of request is set to **ON** in batch server setting, the group on executing request submission command such as qsub, qlogin and qrsh becomes the group of request. In addition, jobs are executed by the group. This is also applicable to the case that execution group is switched to supplementary group by newgrp command and request submission command is executed.

It is also possible to specify group name with --group=<group\_name> option for submission command.

Please note that only the group to which the user belongs can be specified.

### 8.2. Display of Information about Group

#### 8.2.1. Request Information

When displaying request information by qstat, all of requests owned by execution user of qstat are displayed. In this case, execution group of qstat or group of request doesn't matter.

In case of specifying --group=<group\_name> for request information reference, only requests of specified group among all of owned request are selectively displayed.

In addition, if --group option is specified on displaying summary information of request by qstat, the group of the request is displayed.

\$ qstat --group													
RequestID	ReqName	UserName	GrpName	Queue	Pri	STT	S	Memory	CPU	Elapse	R	H	M Jobs
46. bsv1	a. sh	user1	grpA	bq0	0	HLD	-	0.00B	0.00		0	Y	Y Y 1
47. bsv1	a. sh	user1	grpA	bq0	0	QUE	-	0.00B	0.00		0	Y	Y Y 1
52. bsv1	STDIN	user1	grpX	bq2	0	RUN	-	1.93M	0.23	26	Y	Y	Y Y 1

#### 8.2.2. Queue Information

When displaying queue information by qstat -Q, only queues for which user has access privilege are displayed. For this access privilege check, user name and primary group are usually checked on executing qstat.

In case that Designated Group Execution Function of request is set to **ON** in batch server setting, the user and group on executing qstat command are checked about access privilege. In addition, jobs

are executed by the group. It is also possible to display queue information specifying group name with `--group=<group_name>` option.

## 9. Limit per Group and User

### 9.1. Limit per Group and User

On submitting request, request is checked against limits set for batch server and queue to see if any of limits is not exceeded. Among each limit set for batch server and queue, the following limits can be set specifying group or user name individually.

- Submit number limit of batch server
  - Submit number limit for entire batch server
  - Submit number limit per group
    - Submit number limit per one group (in case of no individual designation)
    - Submit number limit per individually designated group name
  - Submit number limit per user
    - Submit number limit per one user (in case of no individual designation)
    - Submit number limit per individually designated user name
- Submit number limit of batch, interactive and routing queue
  - Submit number limit for entire queue
  - Submit number limit per group
    - Submit number limit per one group (in case of no individual designation)
    - Submit number limit per individually designated group name
  - Submit number limit per user
    - Submit number limit per one user (in case of no individual designation)
    - Submit number limit per individually designated user name
- Limitation of the job number of batch and interactive queue
  - Limitation of the job number per queue
  - Limitation of the job number specifying group name individually
  - Limitation of the job number specifying user name individually
- Elapse time limit of batch and interactive queue
  - Elapse time limit per queue
  - Elapse time limit specifying group name individually
  - Elapse time limit specifying user name individually

If any of the above limit is exceeded, submit error occurs. As for limit per group, group of request is checked. For detail of group of request, please refer to 8.1. Group of Request and How to Submit.

### 9.2. Limit Information Reference per Group and User

The above limit values can be displayed by `qstat --limit`. As for batch server, batch queue, interactive

queue and routing queue, limit information is displayed for user and primary group on executing qstat command. (However, limit information of queue for which access privilege is not granted are not displayed.)

In case that Designated Group Execution Function of request is set to **ON** in batch server setting, the limit information of user and group on executing qstat command are displayed.

It is also possible to display limit information specifying group name with --group=<group\_name> option.

Example:

In case of user name, user1 and group name, grpA, limit are checked against the values in bold type.

```
$qstat --limit
Batch Server: bsv1
  Submit Number Limitation Value      = 1000
  Submit User  Number Limitation Value = UNLIMITED
    user1      : Limit = 200
  Submit Group Number Limitation Value = 300
    grpA       : Limit = 100
Execution Queue: bq0
  Submit Number Limit      = 100
  Submit User  Number Limit = 10
    user1      : Limit = 30
  Submit Group Number Limit = 50
    grpA       : Limit = 40
Range of Jobs Limit per Batch Request (min,max) = 1, 10240
  User Limit:
    user1      : Limit = 1, 512
  Group Limit:
    grpA       : Limit = 512, 1024
Resources Limits:
  (Per-Req) Elapse Time Limit = Max: UNLIMITED Warn: UNLIMITED
  User Limit:
    user1      : Limit = Max:      3000S Warn:      2940S
  Group Limit:
    grpA       : Limit = Max:      600S Warn:      540S
Interactive Queue: iq
  Submit Number Limit      = UNLIMITED
  Submit User  Number Limit = UNLIMITED
  Submit Group Number Limit = UNLIMITED
Range of Jobs Limit per Batch Request (min,max) = 1, 10240
Resource Limits:
  (Per-Req) Elapse Time Limit = Max: UNLIMITED Warn: UNLIMITED
Routing Queue: rq
  Submit Number Limit      = UNLIMITED
  Submit User  Number Limit = UNLIMITED
  Submit Group Number Limit = UNLIMITED
```

## 10. Request Submit Using GPU

It is possible to set concurrent GPU number limit by specifying "gpunum\_job=*max\_limit*" as qsub -l option. No warning value is to set to concurrent GPU number limit. If concurrent GPU number limit is not set with this option, default value of concurrent GPU number limit of queue to submit is set to the request.

[Example]

```
$ qsub -l gpunum_job=1 -q bq -l elapstim_req=1000
```

When starting execution of request to which concurrent GPU number limit greater than or equal to 1 is set, JSV selects GPU device which each job can use in execution host, and then job is started with the environment variable, CUDA\_VISIBLE\_DEVICES to which the device number of GPU selected is set.

In case that concurrent GPU number limit is 0 or UNLIMITED, the environment variable, CUDA\_VISIBLE\_DEVICES is not set.

To use the GPU-CPU Affinity feature, set the GPU-CPU Affinity feature to ON using qmgr(1M) command, and submit a request with the GPU limit specified using the --gpunum-lhost option or the -l gpunum\_job option. In this case, the CPU limit is calculated automatically, so do not specify it when submitting a request. For details on the GPU-CPU Affinity feature, refer to [Management] 18.3 GPU-CPU Affinity function.

## 11. Request Submit Using Multi-Instance GPU(MIG)

### 11.1. Multi-Instance GPU(MIG)

When submitting a request to use a multi-instance GPU (hereinafter referred to as MIG), use the `qsub --mig` option. If `qsub` both `--mig` and `--gpunum-lhost` are specified at the same time, `--gpunum-lhost` is ignored.

Format

```
$ qsub --mig=gi-name[:ci-slice-count[,ci-slice-count...]]
```

By specifying the “*gi-name*” argument to the `qsub --mig` option, a GPU instance (GI) corresponding to the “*gi-name*” is allocated.

By specifying the “*ci-slice-count*” argument to the `qsub --mig` option, the GI can be further divided into multiple compute instances (CI), and one GI can be used by multiple processes (when CI is divided, the memory area in the GI is shared by processes).

Multiple GIs can also be assigned to a request by specifying multiple `--mig` options.

Examples of execution are as follows.

- **Submitting a request that uses 1 GI**

[Example] 1 GI profile 4g.20gb is used.

```
$ qsub --mig=4g.20gb -l cpunum_job=1,elapstim_req=1000
```

- **Submitting a request that uses multiple GIs**

Multiple GIs designation can be set at the time of request.

[Example] Using two GI profiles 4g.20gb

```
$ qsub --mig=4g.20gb --mig=4g.20gb -l cpunum_job=1,elapstim_req=1000
```

[Example] Use GI profiles 4g.20gb and 3g.20gb

```
$ qsub --mig=4g.20gb --mig=3g.20gb -l cpunum_job=1,elapstim_req=1000
```

- **Submitting a request with a specification to split the GI into CI**

A GI can be divided into multiple compute instances (CI). Specify the number of CIs to be divided in the “*ci-slice-count*” of the qsub --mig option.

[Example] Using one GI profile 4g.20gb, divided the GI into 3 CIs as shown in the red box below. In this case, specify 2,1,1 for “*ci-slice-count*”.

```
$ qsub --mig=4g.20gb:2,1,1 -l cpunum_job=1,elapstim_req=1000
```

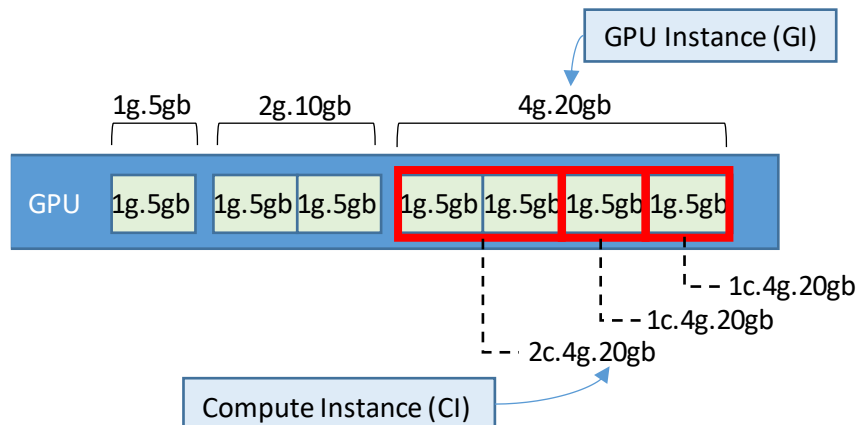


Figure 11-1 Example of dividing GI into CIs

## 11.2. Display of Information about MIG

GIs that can submit requests can be checked with the qstat -Ef command (information on the execution host).

```
$ qstat -Ef
Execution Host: mig1.example.local
Batch Server = bsv.example.local
:
Resource Information:
Memory      = Assign: 255846400 Using: 889856 Maximum: 255846400
Swap        = Assign: 1047552 Using: 0 Maximum: 1047552
Number of Cpus = Assign: 128 Using: 0 Maximum: 128
Multi Instances GPU = {
GPU0:
MIG 1g.5gb    = Using: 0 Total : 0
MIG 1g.5gb+me = Using: 0 Total : 0
MIG 2g.10gb   = Using: 0 Total : 0
```



```

    MIG 3g. 20gb    = Using: 0  Total : 1 (4:4)
    MIG 4g. 20gb    = Using: 0  Total : 1 (0:4)
    MIG 7g. 40gb    = Using: 0  Total : 0
GPU1:
    MIG 1g. 5gb     = Using: 0  Total : 0
    MIG 1g. 5gb+me  = Using: 0  Total : 0
    MIG 2g. 10gb    = Using: 0  Total : 0
    MIG 3g. 20gb    = Using: 0  Total : 1 (4:4)
    MIG 4g. 20gb    = Using: 0  Total : 1 (0:4)
    MIG 7g. 40gb    = Using: 0  Total : 0
}
...

```

### 11.3. Execution of MIG jobs

When the job server executes the job, it sets the specified MIG device in the `CUDA_VISIBLE_DEVICES` environment variable and starts the job script.

The `CUDA_VISIBLE_DEVICES` environment variable is set to multiple MIG devices separated by commas.

However, due to limitations of the CUDA library on MIG-enabled GPU environments, only the first MIG device is used by CUDA applications. If multiple MIG devices are to be used in a single request, each separated item by commas must be reassigned to `CUDA_VISIBLE_DEVICES` environment variable and each process executes with reassigned `CUDA_VISIBLE_DEVICES` environment variable.

Example:

If `CUDA_VISIBLE_DEVICES=device1,device2,device3`, reassign each item as following to `CUDA_VISIBLE_DEVICES` and execute each process.

```

export CUDA_VISIBLE_DEVICE=device1
program1

export CUDA_VISIBLE_DEVICE=device2
program2

export CUDA_VISIBLE_DEVICE=device3
program3

```

Example job script:

```
#!/bin/bash
#PBS -q bq
#PBS -l elapstim_req=1000
#PBS -l cpunum_job=1
#PBS --mig=4g.20gb
#PBS --mig=4g.20gb

function cudamprun() {
    pids=""
    # Replace ", " with a space so that it can be used in the for statement
    tmpcuda="${CUDA_VISIBLE_DEVICES//,/ }"
    # Redefine the CUDA_VISIBLE_DEVICES environment variable
    for CUDA_VISIBLE_DEVICES in $tmpcuda;do
        "$@" &
        pids="$pids $!"
    done
    wait $pids
}

cudamprun cudaprogram
```

## 12. Submitting a request to specify a User Pre-Post Script

The User Pre-Post script function is the function to execute script (It is called a UserPP script.) which is specified to the request. It is executed when before executing the job (PRE-RUNNING) or after executing the job (POST-RUNNING).

### 12.1. UserPP script

For example, using this function, you can check health of a node before starting a job execution and also delete temporary files in an execution host after the job execution.

A UserPP script can reference the following environment variables.

Environment variable	Explanation	Value
UPP_LOCATION	Location (state) where a UserPP script was started	"PRERUNNING" or "POSTRUNNING"
UPP_JOBID	Job ID	<job-number>:<sequence-number>:<host-name>
UPP_NJOBS	Number of jobs	Integer of 1 or larger
UPP_USER	User name of a request owner to be used on an execution host	Character string
UPP_GROUP	Group name of a request owner to be used on an execution host, or group name specified when a request is submitted if the function to specify a group of a request is enabled	Character string
UPP_UID	User ID of a request owner to be used on an execution host	Integer of 0 or larger
UPP_GID	Group ID of a request owner to be used on an execution host, or group ID of the group specified when a request is submitted if the function to specify a group of a request is enabled	Integer of 0 or larger
UPP_HOME	Absolute path name of a home directory	Character string
UPP_STGDIR	Staging file storage directory	Absolute path
UPP_RSTPREV	Previous request state	"QUEUED" (QUEUED state) "RUNNING" (RUNNING state) "CHKPNTING" (CHKPNTING state)

		"HOLDING" (HOLDING state) "SUSPENDING" (SUSPENDING state) "SUSPENDED" (SUSPENDED state) "RESUMING" (RESUMING state)
UPP_RSTRSON	State transition reason	"RUN" (execution start) "RERUN" (rerun) "DELETE" (delete) "EXIT" (execution end) "SYSTEM_FAILURE" (system failure)
UPP_JTPLGY	Job topology	"JTPLGY_DISTRIB" (distribute job) "JTPLGY_INTMPI" (intmpi job) "JTPLGY_MVAPICH" (mvapich job) "JTPLGY_NECMPI" (necmpi job) "JTPLGY_OPENMPI" (openmpi job)
Other environment variables to be set to a job		

The current directory of a UserPP script is a home directory of a job execution user (user of a request owner on an execution host).

Whether executing a UserPP script is successful is determined by its exit status. When the exit status is 0, it is determined that the script execution succeeded. When the exit status is other than 0, it is determined that the script execution failed. If executing a UserPP script fails in PRE-RUNNING, the script execution does not start and the relevant request enters QUEUED. If executing a UserPP script fails in POST-RUNNING, the subsequent processes (such as staging out) are executed and the relevant request ends.

## 12.2. Specification method when submitting a request

Use the `--userpp-script` option of the `qsub(1)`, `qlogin(1)`, and `qrsh(1)` command to specify a UserPP script.

```
--userpp-script=<loc>:<script_path>[,<loc>:<script_path>]
```

For `<loc>`, specify the location (state) in which to execute a UserPP script. The following states can be specified.

- `pr` Executes a UserPP script in PRE-RUNNING.
- `por` Executes a UserPP script in POST-RUNNING.

For `<script_path>`, specify the path of a script to be executed.

The following shows an example to specify test.sh in the current directory to be executed before executing a job.

```
$ qsub --userpp-script=pr: test.sh
```

Use the -f option of the qstat(1) command to check the UserPP script that was specified when submitting a request.

[Display example]

```
$ qstat -f 123.bsv1
Request ID:123.bsv1
  Request Name = STDIN
  User Name = user1
  :
  Attach = No
  Cluster Type Select = NONE
  UserPP Script = {
    Pre-running = /home/user1/test.sh
  }
  :
```

Use the --userpp-script option of qalter(1) to change or delete a UserPP script. However, a script cannot be deleted after the execution of the relevant request has started.

### 12.3. Output of UserPP script

Use the --userpp-script option of qcat(1) to reference the execution result (standard output and standard error output) of a UserPP script.

For a batch request, the detailed result information (such as an execution result, standard output, standard error output) can be referenced in a request log by setting the log level of the request log to 2.

The following shows a request log output example.

```
... OMIT ...
----- UserPP Information -----
Location: PRE-RUNNING
Path: clhost:/home/user1/test.sh
===== Job 0 =====
Result: Success (exited 0)
Session ID: 8739
Stdout:
<The contents of standard output of a UserPP script>
```

**Stderr:**

<The contents of stderr of a UserPP script>

Location shows the state in which a UserPP script was executed. (PRE-RUNNING or POST-RUNNING)

Path shows the path on the submission host to which a UserPP script that was specified when submitting a request.

Result shows the UserPP script execution result. Any of the following is displayed.

Output of Result	Meaning
Success (exited 0)	The UserPP script execution succeeded.
Failure (exited+exitcode)	The UserPP script execution failed.
Killed by signal + signal number	The UserPP script execution was terminated by a signal.
Deleted	The UserPP script execution was interrupted by qdel.
Timeout	The UserPP script execution was terminated due to a timeout.
Not Running	The UserPP script could not be executed because the job server entered LINKDOWN state.
System Error	An NQSV system error occurred.

Session ID shows the process session ID of a UserPP script. (This is displayed only when the UserPP script execution succeeded.)

Stdout and Stderr show the contents of the standard output and standard error output of a UserPP script.

## 13. Submitting a request using a provisioning environment in conjunction with OpenStack

This function is NOT available for the environment whose execution host is SX-Aurora TSUBASA system.

NQSV defines and manages information of an OS image and resources used by a job in a provisioning environment in conjunction with OpenStack as a template.

A job can be executed in a provisioning environment in conjunction with OpenStack by specifying the template for the `--template` option of a submit command (`qsub(1)`, `qlogin(1)`, or `qssh(1)`).

### 13.1. Reference the template information

Use the `--template` option of the `qstat(1)` command to reference the template information defined in a system.

[Indication example]

<b>\$qstat --template</b>							
[OpenStack Template]							
Template	L Image	Flavor	CPU	Memory	GPU	Custom	Comment
os_70_small	- rhel70	small	1	1.0G	0	(none)	RHEL7 Small.
os_70_medium	- rhel70	medium	2	2.0G	0	(none)	(none)
:							

Use `qstat --template -f` to reference the more detailed information.

[Indication example]

```
$qstat --template -f
OpenStack Template: os_70_small
  Lock State = UNLOCK
  OS Image   = rhel70
  Flavor     = small
  CPU Number = 1
  Memory Size = 1GB
  GPU Number = 0
  Boot Timeout = 900
  Stop Timeout = 900
  Custom      = (none)
  Comment     = RHEL7 Small.
  Requests    = 0

OpenStack Template: os_70_medium
  Lock State = UNLOCK
  :
```

### 13.2. Submitting a request with a template specified

Use the `--template` option of `qsub(1)`, `qlogin(1)`, and `qrsh(1)` to submit a request with a template specified.

[Example]

```
$qsub --template= os_70_small -q bq -l elapstim_req=1000
```

When a request is submitted with the `--template` option specified, the number of CPUs, memory size, and number of GPUs that are defined in the specified template are used as the limits on resources (number of CPUs, memory size, number of GPUs) per job of the submitted request. These limits are used to check the resources (number of CPUs, memory size, and number of GPUs per job) of a queue when the queue is submitted. However, the queue standard values are not applied.

### 13.3. Information of a request with a template specified

#### 13.3.1. Request information

When submitting a request, if a template is specified by using the `--template` option, `qstat -f` shows the following.

[Example]

```
$ qstat -f
Request ID: 3.bsvhost
  Request Name = STDIN
  User Name = user1
  :
  Attach = No
  OpenStack Template = os_70_small
    CPU Number = 1
    Memory Size = unused
    GPU Number = 0
    :
Resources Information:
  Memory = 0.000000B
  :
Resources Limits:
  (Per-Req) Elapse Time Limit = Max: 1000S Warn: 1000S
  :
  (Per-Job) CPU Number = Max: 1 Warn: —
  :
  (Per-Job) Memory Size = Max: 1G Warn: 1G
  :
  (Per-Job) GPU Number = Max: 0 Warn: —
  :
```

OpenStack Template shows information of the specified template. The limits on the number of CPUs,



memory size, and number of GPUs that are defined in the specified template are used as those (number of CPUs, memory size, number of GPUs) per job.

### 13.3.2. Job information

When a job of the request that was submitted with --template specified is running on a virtual machine, qstat -J displays an asterisk (\*) at the beginning of the ExecutionHost item. This information is displayed while the request state is from PRERUNNING to POSTRUNNING and a virtual machine is running.

[Example]

\$qstat -J								
NO	RequestID	EJID	Memory	CPU	JSVNO	ExecutionHost	UserName	Exit
0	3. bsvhost	100	0.00B	0.00	100	*exec1	user1	-

When the details of the above job is displayed by qstat -Jf, a virtual machine host name is displayed after an execution host name as follows.

[Example]

\$qstat -Jf	
Request ID: 3. bsvhost	
Batch Job Number = 0	
Execution Job ID = (none)	
User Name = user1	
User ID = 111	
Group ID = 111	
Job Server Number = 100	
Job Server Name = JobServer0100	
Execution Host = exec1 (vmhost1)	
:	

When a job of the request that was submitted with --template specified is running on a bare metal server, qstat -J displays [B] at the beginning of the ExecutionHost item.

[Example]

\$qstat -J								
NO	RequestID	EJID	Memory	CPU	JSVNO	ExecutionHost	UserName	Exit
0	4. bsvhost	100	0.00B	0.00	1000	[B]bhost	user1	-

When the details of the above job is displayed by qstat -Jf, [Baremetal] is displayed after an execution host name.

[Example]

\$qstat -Jf	
Request ID: 4. bsvhost	
Batch Job Number = 0	

```
Execution Job ID = (none)
User Name = user1
User ID = 111
Group ID = 111
Job Server Number = 1000
Job Server Name = JobServer1000
Execution Host = bhost [Baremetal]
:
```

## 14. Submitting a request using a provisioning environment in conjunction with Docker

NQSV defines and manages information of an OS image and resources used by a job of a provisioning environment in conjunction with Docker as a template.

A job can be executed in a provisioning environment in conjunction with Docker by specifying the template for the `--template` option of a submit command (`qsub(1)`, `qlogin(1)`, or `qrsh(1)`).

### 14.1. Referencing the template information

Use the `--template` option of the `qstat(1)` command to reference the template information defined in a system.

[Display example]

```
$qstat --template
[Container Template]
```

Template	L	Image	CPU	Memory	GPU	Custom	Comment
App_A	-	App_A_Img	2	1.0G	0	(none)	For App_A

Use `qstat --template -f` to reference the more detailed information.

[Display example]

```
$qstat --template -f
Container Template: App_A
  Lock State   = UNLOCK
  Image        = App_A_Img
  CPU Number   = 2
  Memory Size  = 1GB
  GPU Number   = 0
  VE Number    = 2
  HCA Number   = {
    For I/O = 0
    For MPI = 0
    For ALL = 1
  }
  Boot Timeout = 900
  Stop Timeout = 900
  Custom       = (none)
  Comment      = For App_A
  Requests     = 5
```

### 14.2. Submitting a request with a template specified

Use the `--template` option of `qsub(1)`, `qlogin(1)`, and `qrsh(1)` to submit a request with a template specified.

[Example]

```
$qsub --template=App_A -q batch1 -l elapstim_req=1000
```

When a request is submitted with the `--template` option specified, the number of CPUs, memory size, and number of GPUs that are defined in the specified template are used as the limits on resources (number of CPUs, memory size, number of GPUs) per job of the submitted request. These limits are used to check the resources (number of CPUs, memory size, and number of GPUs per job) of a queue when the queue is submitted. However, the queue standard values are not applied.

### 14.3. Information of a request with a template specified

#### 14.3.1. Request information

When submitting a request, if a template is specified by using the `--template` option, `qstat -f` shows the following.

[Example]

```
$ qstat -f
Request ID: 443.bsvhost
  Request Name = STDIN
  User Name = user1
  :
  Container Template = App_A
    CPU Number = 2
    Memory Size = 1GB
    GPU Number = 0
    VE Number = 2
    HCA Number = {
      For I/O = 0
      For MPI = 0
      For ALL = 1
    }
  :
Resources Information:
  :
Logical Host Resources:
  VE Node Number = Max:      2 Warn:      ---
  CPU Number     = Max:      2 Warn:      ---
  :
Resources Limits:
  (Per-Req) Elapse Time Limit = Max:    1000S Warn:    1000S
  :
  (Per-Job) CPU Time          = Max: UNLIMITED Warn: UNLIMITED
  (Per-Job) CPU Number       = Max:      2 Warn:      ---
  (Per-Job) Tape Number      = Max: UNLIMITED Warn:      ---
  :
  (Per-Job) Process Number   = Max: UNLIMITED Warn:      ---
```

(Per-Job) Memory Size	= Max:	1G	Warn:	1G
(Per-Job) Virtual Memory Size	= Max:	UNLIMITED	Warn:	UNLIMITED
(Per-Job) GPU Number	= Max:	0	Warn:	—
(Per-Prc) CPU Time	= Max:	UNLIMITED	Warn:	UNLIMITED
:				

Container Template shows information of the specified template. The limits on the number of CPUs, memory size, and number of GPUs that are defined in the specified template are used as those (number of CPUs, memory size, number of GPUs) per job.

### 14.3.2. Job information

When a job of the request that was submitted with `--template` specified is running in a container, `qstat -J` displays an asterisk (\*) at the beginning of the ExecutionHost item. This information is displayed while the request state is from PRERUNNING to POSTRUNNING and a container is running.

[Example]

\$qstat -J								
NO	RequestID	EJID	Memory	CPU	JSVNO	ExecutionHost	UserName	Exit
-----	-----	-----	-----	-----	-----	-----	-----	-----
0	443.bsvhost	100	0.00B	0.00	100	*exec1	user1	-

When the details of the above job is displayed by `qstat -Jf`, a container host name is displayed after an execution host name as follows.

[Example]

```
$qstat -Jf
Request ID: 443.bsvhost
  Batch Job Number = 0
  Execution Job ID = (none)
  User Name = user1
  User ID = 111
  Group ID = 111
  Job Server Number = 100
  Job Server Name = JobServer0100
  Execution Host = exec1 (NQSV-0-443-10)
  :
```

## 15. Submitting a request using the Custom Resource function

The custom resource function is a function to control the custom resource amount that is used concurrently in scheduling according to the defined custom resource information.

In an environment on which the custom resource function is used, use the `--custom` option of a job submit command (`qsub(1)`, `qlogin(1)`, `qrsh(1)`) to specify a custom resource name and its usable amount. In addition, the custom resource usage limit information is set to a queue. This information includes the default usage, usage specification range, and setting of whether it is possible to specify the usage of a request to be uncontrolled.

### [Notes]

The custom resource function is available only for a batch request (local request) and interactive request.

### 15.1. Referencing the custom resource information

Use the `--template` option of the `qstat(1)` command to reference the custom resource information defined in a system.

[Example]

```
$ qstat --custom
Custom Resource : Power
  Consumer = job
  Type = bsv : Available Resource Limit = 5000
  Type = host: Target = (default) Available Resource Limit = 100
                Target = host_a Available Resource Limit = 120

Custom Resource : License
  Consumer = request
  Type = bsv : Available Resource Limit = 50

Custom Resource : Virtual
  Consumer = job
  Type = bsv : Available Resource Limit = 100
  Type = host: Target = (default) Available Resource Limit = 1
```

Use `qstat -Qf` to check the custom resource usage limit information of a queue. The displayed information includes the default usage, usage specification range, and setting of whether it is possible to specify the usage of a request to be uncontrolled.

[Example]

```
$ qstat -Qf
Execution Queue: bq@bsv
  Run State = Active
  Submit State = Enable
```

```

:
UserExit Script:
  (none)
Custom Resources:
  Power      : Range (min,max) = 10, 50      Std = 30      : Permit Unused = No
  License    : Range (min,max) = 1, 20      Std = unused   : Permit Unused = Yes
  Virtual    : Range (min,max) = 1, 1      Std = 1      : Permit Unused = No
Resources Limits:
  (Per-Req) Elapse Time Limit      = Max: UNLIMITED Warn: UNLIMITED Std: 3600S
  (Per-Job) CPU Time              = Max: UNLIMITED Warn: UNLIMITED Std: UNLIMITED
:

```

## 15.2. Submitting a request with a custom resource specified

The following shows the details of how to specify a custom resource at the request submit commands `qsub(1)`, `qlogin(1)` and `qrsh(1)`.

```
--custom cr_name=n[,cr_name=n...]
```

For *cr\_name*, specify a custom resource name. For *n*, specify the custom resource amount to be used concurrently. The usage of multiple custom resources can be specified by delimiting with a comma. Do not enter any space after a comma.

For *n*, specify an integer from 1 to 2147483647 or **unused** (or 0). **unused** (or 0) means that the specified custom resource will not be used (the usage is 0) and its usage will be uncontrolled. However, unused (or 0) cannot be specified if it is not possible for a submission host to specify the usage of a request to be uncontrolled. If the usage specified with an integer exceeds the specification range set to the submission queue, a submit error occurs.

If the usable amount of the custom resource is not specified when submitting a request, the default resource usage set to the submission queue is used.

[Example]

```
$ qsub --custom Power=20,License=unused -q bq script
Request 6.bsv submitted to queue: bq.
```

## 15.3. Referencing the custom resource usage of a queue

Use `qstat -f` to reference the custom resource usage set to a queue.

[Example]

```
$ qstat -f
Request ID: 6.bsv
  Request Name = STDIN
  User Name = user1
  :
Custom Resources:
  Power      = 20
```

```
License          = unused
Virtual          = 1
Execution Hosts(JSVNO):
ehost100(100)
Resources Information:
Memory   = 1.932587MB
CPU Time = 0.225019S
:
```

For the custom resource usage information of a request, set the environment variables of a job as follows. There are 2 kinds of naming rule for the environment variable but both of them have same value. This allows to reference the usage of all custom resources of a request in a job or UserPP script. (For details, see "12. Submitting a request to specify a User Pre-Post Script".)

```
NQSV_CR_<custom resource name>=<amount>
```

or

```
NQSII_CR_<custom resource name>=<amount>
```

[Setting example of an environment variable]

```
NQSV_CR_Power = 20
NQSV_CR_License = unused
NQSV_CR_Virtual = 1
```



## 16. Hybrid Request function

### 16.1. About Hybrid Request

A normal (non-hybrid) request consists of one or more jobs (= logic hosts). But the resources assigned for each job were the same. In a hybrid request, different amount of resources can be used for each job.

For example a request with the following six jobs can be realized by hybrid request.

- Job 0 and Job 1 : Number of CPU : 1 and number of VEs : 2
- Job 2 and Job 3 : Number of CPU : 1 and number of GPUs : 4
- Job 4 and Job 5 : Number of CPU : 8

The MPI which can be executed by hybrid request is only NEC MPI.

### 16.2. Submitting Hybrid Request

A hybrid request has multiple jobs and it composes two or more group of job. Each group has one or more jobs. All jobs in one group has the same resource limitation value. A different group can have the different resource limitation value. A request which has only one group is treated as normal (non-hybrid) request.

A hybrid request can be submitted in the same way as a normal request. The batch request is submitted by qsub command, and an interactive request is submitted by qlogin command or qrsh command. The **--job-separator** option is used to specify the different options to every job group. (It is possible to omit --job-separator by using "---".)

It can be specified like as follows.

```
qsub <common options for whole jobs> ¥
--job-separator <the option for job group0> ¥
--job-separator <the option for job group1> ¥
:
<job script>
```

The option which can be used to specify different values for each job group is as follows. It is possible to specify an individual option to each group divided in --job-separator option.

Option	Use
--cpunum-lhost	The number of CPUs per logical host
--cputim-lhost	CPU time per logical host
--gpunum-lhost	The number of GPUs per logical host
--memsz-lhost	Memory size per logical host
--venum-lhost	The number of VE nodes per logical host
--vmemsz-lhost	Virtual memory size per logical host
--use-hca	HCA port number
--venode	total number of VEs
-l cpunum_job	The number of CPUs per job (logical host)
-l cputim_job	CPU time per job (logical host)
-l gpunum_job	The number of GPUs per job (logical host)
-l memsz_job	Memory size per job (logical host)
-l vmemsz_job	Virtual memory size per job (logical host)
-l coresz_prc	Core file size per process
-l cputim_prc	CPU time per process
-l datasz_prc	Data segment size per process
-l filenum_prc	The number of open file per process
-l filesz_prc	File size per process file size
-l stacksz_prc	Stack size per process
-l vmemsz_prc	Virtual memory size per process
-b	The number of jobs
-B	Job Condition
-v	Environment Variables
-y	Reservation ID (*)

(\*) All job groups in the hybrid request must be specified the Reservation ID if the hybrid request is submitted into the reservation area. Some job in the hybrid request is specified the reservation id and the others is not specified could not submit.

The options described above are able to specify as an individual option to each group separated by the --job-separator option, and they are also able to specify as common options for whole request. If they are specified as common options, they are treated as default values for every job group that individual option is not specified.

The other options that listed above can be specified as common options only. If they are specified to each job group, they are erroneous.

A concrete instance is mentioned below. The area of the option with **orange** background is the common option, and the area of the option with **light-blue** background is the individual option.

Example 1:

```
qsub -q exeq -l elapstim_req=600 -Tnecmpi -b2 ¥
--job-separator --venum-lhost=2 --cpunum-lhost=1 --use-hca=1 ¥
--job-separator --cpunum-lhost=1 --gpunum-lhost=4 ¥
--job-separator --cpunum-lhost=8 ¥
job_script
```

This example submits the job script named "job\_script" to "exeq" queue. It is elapse time limit is 600 seconds and job topology is necmpi. -b 2 that is specified as common option is applied to each individual job group because the number of job (-b) isn't specified to individual job group.

Therefore following 3 kinds of job group is submitted on this example.

- Number of CPU : 1 , number of VE : 2 and number of HCA:1      2 jobs
- Number of CPU : 1 and number of GPU : 4      2 jobs
- Number of CPU : 8      2 jobs

Example 2:

```
qsub -q exeq -l elapstim_req=600 --cpunum-lhost=2 -Tnecmpi ¥
--job-separator --memsz-lhost=1GB ¥
--job-separator --venum-lhost=3 --use-hca=1 -b4 ¥
job_script
```

This example submits the job script named "job\_script" to "exeq" queue. It is elapse time limit is 600 seconds and job topology is necmpi. --cpunum-lhost=2 that is specified as a common option is applied to each individual job group, because per logical host CPU number limit (--cpunum-lhost) isn't specified to individual job group.

Therefore following 2 kinds of job group is submitted on this example.

- Number of CPU : 2 and memory size : 1GB      1 job
- Number of CPU : 2 , number of VE : 3 and number of HCA:1      4 jobs

Example 3:

```
qsub -q exeq -l elapstim_req=600 --cpunum-lhost=2 -Tnecmpi --venum-lhost=2 --use-hca=1 \
-b4 job_script
```

This example submits the job script named "job\_script" to "exeq" queue. It is elapse time limit is 600 seconds and job topology is necmpi. This request is a normal (not hybrid) request, because --job-separator option is not specified.

Therefore this request submits 4 jobs with 2 CPUs and 2 VEs.

Example 4:

```
qsub -q exeq -l elapstim_req=600 --cpunum-lhost=1 -Tnecmpi ¥
--job-separator --cpunum-lhost=4 --venum-lhost=1 --use-hca=1 ¥
--job-separator --use-hca=1 --venode=8 --use-hca=1 ¥
job_script
```

This example submits the job script named "job\_script" to "exeq" queue. It is elapse time limit is 600 seconds and job topology is necmpi. The number of job isn't specified to common option, and --venode option is specified to the 2nd individual job group. Therefore, the number of job and number of VE for the 2nd individual job group depends on the default number of incorporated VE nodes of the queue "exeq".

Therefore following 2 kinds of job group is submitted on this example.

- Number of CPU : 4, number of VE : 1 and number of HCA : 1 1 job
- Number of CPU : 1, number of HCA : 1 and number of VE and number of job is depended on the default number of incorporated VE nodes of the queue "exeq".

Example 5:

```
qsub -q exeq -Tnecmpi ¥
--job-separator --cpunum-lhost=4 --venum-lhost=1 -l --use-hca=1 elapstim_req=300 ¥
--job-separator --cpunum-lhost=1 -b2 ¥
job_script
```

This example specifies -l elapstim\_req option to the 1st individual job group. This operation is erroneous, because the option cannot be specified to individual job group.

If a hybrid request is submitted as a concurrent request (a request submitted using the --parallel option with the qsub command), the request will not be scheduled.

### 16.3. Hybrid Request Information

It is possible to find the information of hybrid requests by the -f option of the qstat(1) command. If a request is a parametric request, qstat -Rf is also available. In case of a hybrid request, the following items are displayed for the job group number 0.

Reservation ID	
HCA Number	
Resources Limits per Logical Host	
	VE Node Number
	CPU Number
	GPU Number
	CPU Time
	GPU Number
	Memory Size
	Virtual Memory Size
Resources Limits per Process	
	CPU Time
	Open File Number
	Memory Size
	Virtual Memory Size
	Data Segment Size
	Stack Segment Size
	Core File Size
	Permanent Files Size

The item "VE Node Number" indicates the total of --venode option value for all individual job group. Please note that the value is not --venode value of job group that job 0 belongs to.

Use qstat -Jt to display the values other job group than number 0. An example is shown below.

JNO	RequestID	VE	CPU	GPU	CPUTime	Memory	VMemory
0	117. host. exampl	1	1	0	ULIM	ULIM	ULIM
1	117. host. exampl	0	4	0	ULIM	1G	ULIM
2-3	117. host. exampl	0	1	1	ULIM	ULIM	ULIM

The items listed above are displayed in the resource limits. In case of the request which isn't a hybrid request, the same values are displayed to each job.

It can be used with -f option and -t option at the same time, too. In this case, resource limits are displayed in detail.

```
$ qstat -Jft
Request ID: 117.host.example.com
Batch Job Number = 0
HCA Number = (none)
Reservation ID = (none)
Resources Limits per Logical Host:
VE Node Number      = Max:          1 Warn:      ---
CPU Number           = Max:          1 Warn:      ---
GPU Number           = Max:          0 Warn:      ---
CPU Time             = Max: UNLIMITED Warn: UNLIMITED
Memory Size          = Max: UNLIMITED Warn: UNLIMITED
Virtual Memory Size  = Max: UNLIMITED Warn: UNLIMITED

Resources Limits per Process:
CPU Time             = Max: UNLIMITED Warn: UNLIMITED
Open File Number     = Max: UNLIMITED Warn:      ---
Memory Size          = Max: UNLIMITED Warn: UNLIMITED
Virtual Memory Size  = Max: UNLIMITED Warn: UNLIMITED
Data Segment Size    = Max: UNLIMITED Warn: UNLIMITED
Stack Segment Size   = Max: UNLIMITED Warn: UNLIMITED
Core File Size       = Max: UNLIMITED Warn: UNLIMITED
Permanent File Size  = Max: UNLIMITED Warn: UNLIMITED
:
```

#### 16.4. Attribute Change of Hybrid Request

It is possible to change the attribute of a hybrid request in the same way as a normal batch request by the `qalter(1)` command. Attribute that specified to interactive request cannot be changed. When changing the attribute that has the different values for each job group like resource limits, specify the target job number with request ID like following.

```
[job_number:]<request_identifier>
```

All jobs which belongs to same job group is altered to same value.

For example, in order to change the number of VEs to 4 of job 1 of 117.host.example.com, execute following command. On this situation, the number of VEs of all job that belongs to same job group of job 1 is changed to 4.

```
$ qalter --venum-lhost=4 1:117.host.example.com  
Attribute of Request is altered.
```

The job group number can be omitted, and if it is omitted, all job in the request is altered. To change the attribute which is set to whole of request (common option), a job group number cannot be specified. If a job group number is specified for it, it is erroneous.

## 17. Submitting a request using a provisioning environment in conjunction with singularity

NQSV allows you to execute jobs in conjunction with singularity container. By preparing a singularity container image file in advance and calling the singularity command in the job script when executing the job and specifying the image file or job program, the job by the singularity container can be executed.

For NQSV, the available version is singularity version 3.6.4.

NEC MPI and OpenMPI 4.0.3 are supported.

Please refer to the following page for how to build a singularity container that uses NEC MPI.

<https://github.com/veos-sxarr-NEC/singularity>

### 17.1. Job execution method

#### 17.1.1. Executing MPI Job

For NEC MPI

When executing an NEC MPI job in a singularity container, describe it in the job script as follows.

```
source /opt/nec/ve/mpi/<nec mpi version in the image>/bin64/necmpivars.sh
mpirun -np <number_of_processes> singularity exec --bind /var/opt/nec/ve/veos <container
image name> <program name>
```

Job script example:

```
#!/bin/bash
#PBS -q bq
#PBS -T necmpi
#PBS -b 2
#PBS -l elapstim_req=300
#PBS -venum-lhost=8 --cpunum-lhost=2

source /opt/nec/ve/mpi/2.11.0/bin64/necmpivars.sh
mpirun -np 16 /usr/bin/singularity exec --bind /var/opt/nec/ve/veos ~/necmpi.sif ~/a.out
```

When executing an interactive request with qlogin (1) or qrsh (1), it is possible to execute a NEC MPI job with a singularity container with the same image as a batch request.

For OpenMPI



When executing an OpenMPI job in a singularity container, describe it in the job script as follows.

```
mpirun -np <number_of_processes> singularity exec <container image name> <program name>
```

Job script example:

```
#!/bin/bash
#PBS -q bq
#PBS -T openmpi
#PBS -b 2
#PBS -l elapstim_req=300
#PBS --cpunum-lhost=2

mpirun -np 2 /usr/bin/singularity exec ~/openmpi.sif ~/a.out
```

When executing an interactive request with qlogin (1) or qrsh (1), it is possible to execute an OpenMPI job with a singularity container with the same image as a batch request.

### 17.1.2. Executing distributed Job

When executing a distributed job in a singularity container, describe it in the job script as follows.

```
singularity exec <container image name> <program name>
```

Job script example:

```
#!/bin/bash
#PBS -q bq
#PBS -T distribute
#PBS -b 2
#PBS -l elapstim_req=300
#PBS --cpunum-lhost=2

/usr/bin/singularity exec ~/distribute.sif ~/a.out
```

When executing an interactive request with qlogin (1) or qrsh (1), it is possible to execute a distributed job with a singularity container with the same image as a batch request.

## 18. Limitations

### 18.1. Max Value, Max length and Range of Value

The max value, the max length and the range of value in NQSV system are defined as follows. Some of them are defined in nqsv.h.

1. Job server
  - The number of job server that can connect to one batch server is at most 10240.
  - The range of specified job server number is from 0 to 10239.  
NQS\_MAX\_JSVNO is defined as 10239 (the maximum of job server number)
  - The max length of job server name is 15.  
NQS\_LEN\_JSVNAME is defined as 15.
2. Scheduler (For details, please refer to [JobManipulator].)
  - The range of specified scheduler number is from 0 to 15.  
NQS\_MAX\_SCHNO is defined as 15 (the maximum of scheduler number)
  - The max length of scheduler name is 15.  
NQS\_LEN\_SCHNAME is defined as 15.
3. The host name
  - The max length of the host name in NQSV is 255.  
NQS\_LEN\_HOSTNAME is defined as 255.
4. Node group
  - The number of node group that can be created is at most 2048.
  - The max length of the node group name is 15.  
NQS\_LEN\_NGRPNAME is defined as 15.
  - The max length of the comment text is 63.  
NQS\_LEN\_COMMENT is defined as 63.
5. Request name
  - The max length of request name in NQSV is 63.  
NQS\_LEN\_REQNAME is defined as 63.
6. Job number
  - The range of job number is from 0 to 10239.  
NQS\_MAX\_JOBNO is defined as 10239. (the maximum of job number)
7. User name
  - The max length of user name in NQSV is 47.  
NQS\_LEN\_USERNAME is defined as 47.
8. Group name

- The max length of group name in NQSV is 47.  
NQS\_LEN\_GROUPNAME is defined as value 47.
9. Queue name
- The max length of queue name in NQSV is 15.  
NQS\_LEN\_QUENAME is defined as 15.
10. Pathname
- The max length of the pathname in NQSV is 1023.  
NQS\_LEN\_PATHNAME is defined as 1023.
11. File name
- The max length of the file name in NQSV is 255.  
NQS\_LEN\_FILENAME is defined as 255.
12. Account code
- The max length of the account code b in NQSV is 15.
13. Mail address
- The max length of the mail address in NQSV is 1023.  
NQS\_LEN\_MAILADDR is defined as 1023.
14. Job execution conditional
- The max length of the job condition in NQSV is 255.  
NQS\_LEN\_JOBCOND is defined as 255.
15. Template
- The max length of the template name in NQSV is 47.  
NQS\_LEN\_TEMPLATENAME is defined as 47
  - The max length of the OS image name in NQSV is 47  
NQS\_LEN\_VMIMGNAME is defined as 47
  - The max length of the flavor name in NQSV is 47  
NQS\_LEN\_FLAVORNAME is defined as 47
  - The max length of the custom define in NQSV is 400  
NQS\_LEN\_TEMPLATECUSTOM is defined as 400
  - The max length of the comment in NQSV is 255  
NQS\_LEN\_TEMPLATECOMMENT is defined as 255
16. Custom resource
- The number of custom resource that can be created is at most 20.  
NQS\_MAX\_CRNUM is defined as 20
  - The max length of the custom resource name in NQSV is 15  
NQS\_LEN\_CRNAME is defined as 15

## 18.2. Version between Command and Batch Server

When the version of user command or administrator command, match versions the command and a Batch Server. The Batch Server version which are connected to a batch server to refer to information or do setting, is older than the version of NQSV/Client, an error can occur.

## 18.3. User-level checkpoint

Checkpoint is a function that saves the internal state of a running job. The saved job can be restarted by the paired restart function. User-level checkpoints are saved by the user process. NQSV provides support for using user-level checkpoints.

### [Notes]

This function is an experimental function. Please contact NEC Support if you want to use it.

### 18.3.1. Preparing to use user-level checkpoint

Create a script for checkpointing on the user side. This script is not included in NQSV. This script is called a checkpoint script. The checkpoint script is stored on the execution host, but it must be the same path on all execution hosts.

### 18.3.2. How to submit a request

For requests that use checkpointing, please specify the checkpoint script when submitting with `qsub(1)`. The checkpoint script is specified with the `--use-custom-suspend` option. The format is as follows.

```
--use-custom-suspend = chkpnt:script
```

In *script*, specify the checkpoint script on the execution host with an absolute path. For example, in the case of `/home/user1/checkpoint.sh`, specify as follows.

```
--use-custom-suspend = chkpnt:/home/user1/checkpoint.sh
```

For details, refer to the `qsub(1)` in [Reference].

### 18.3.3. When taking a checkpoint

The timing to take a checkpoint is when a normal request is interrupted by an urgent request or a special request. For details on the urgent request and special request, refer to Chapter 4 Advanced Scheduling Features of [JobManipulator]. If an interrupted normal request specifies a checkpoint

script with the `--use-custom-suspend` option, NQSV executes this checkpoint script. After execution, the interrupted normal request ends.

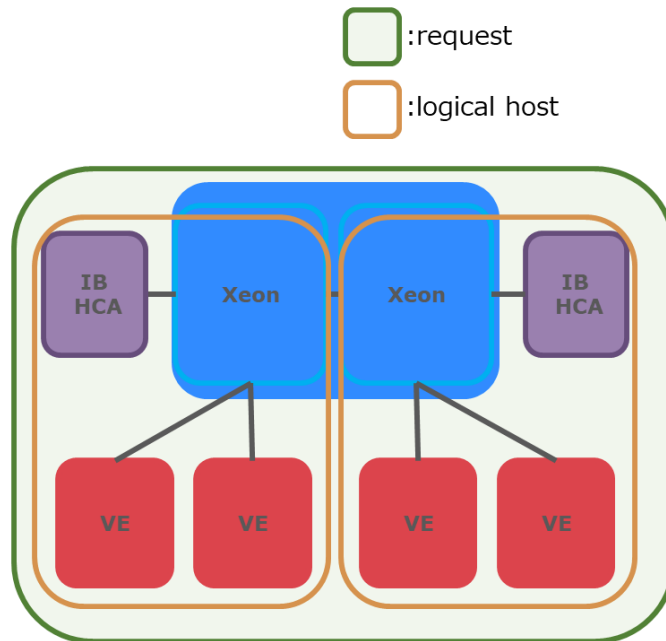
## Appendix.A How to submit NQSV Request

### A.1 Request using VEs

The following example shows the job script of the MPI program with 8 processes, on two logical hosts, two VEs each logical hosts.

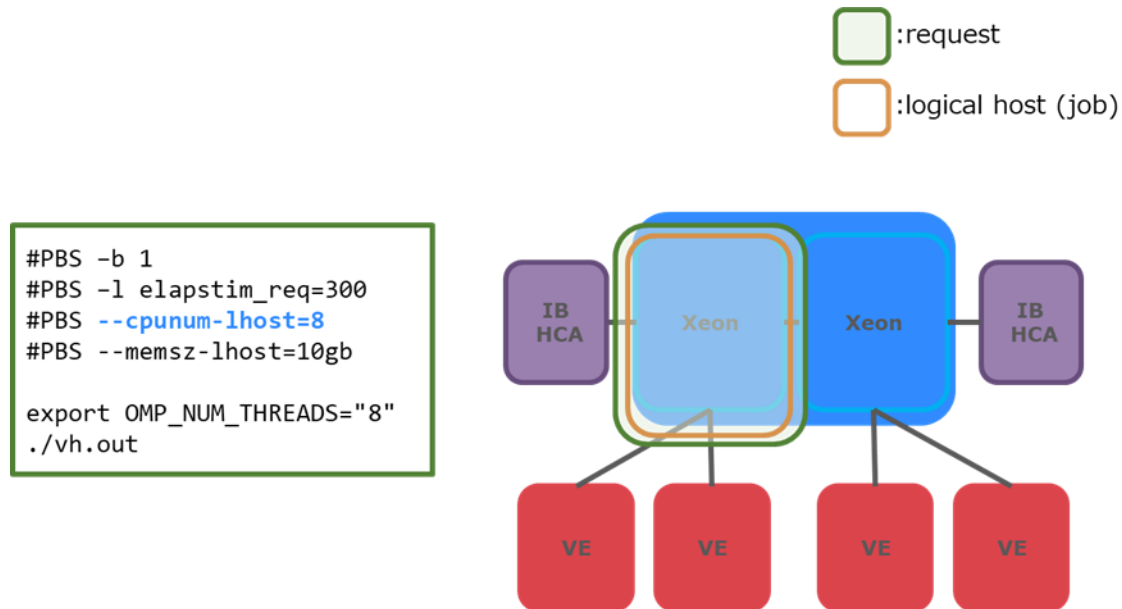
```
#PBS -b 2
#PBS -T necmpi
#PBS -l elapstim_req=300
#PBS --cpunum-lhost=2
#PBS --memsz-lhost=10gb
#PBS --use-hca=1
#PBS --venum-lhost=2

mpirun -nn 2 -np 8 ./ve.out
```



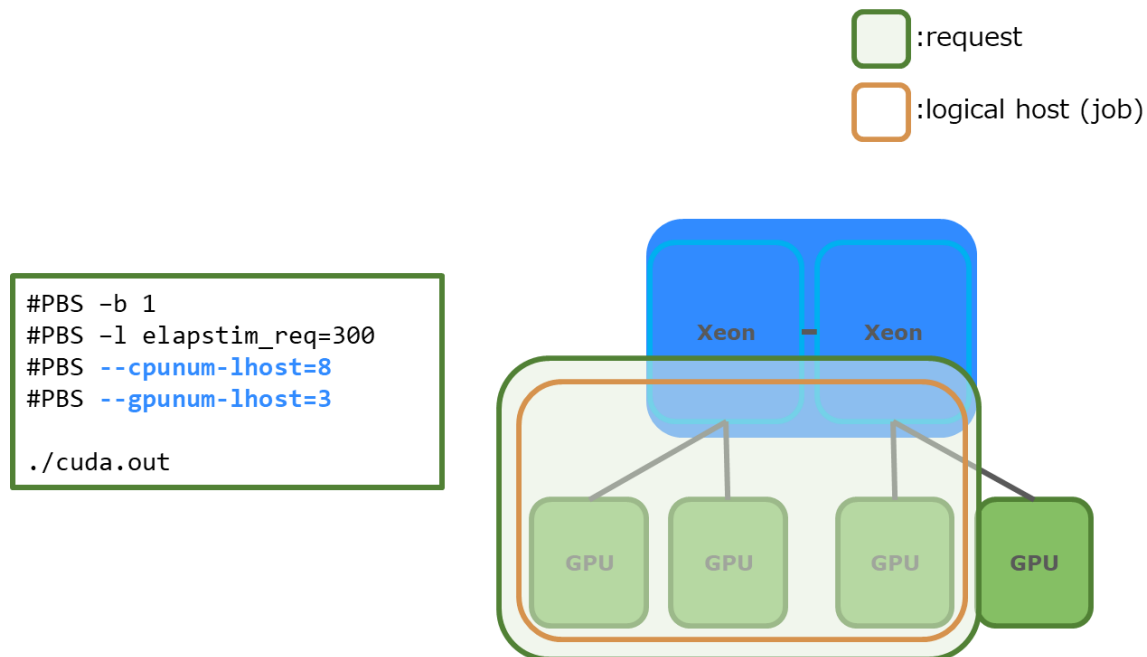
## A.2 Request using x86

The following example shows the job script of the OpenMP program which uses only x86 CPUs of VH.



## A.3 Request using GPUs

The following example shows the job script of the program using three GPUs.




#### A.4 Request using Multi-Instance GPU

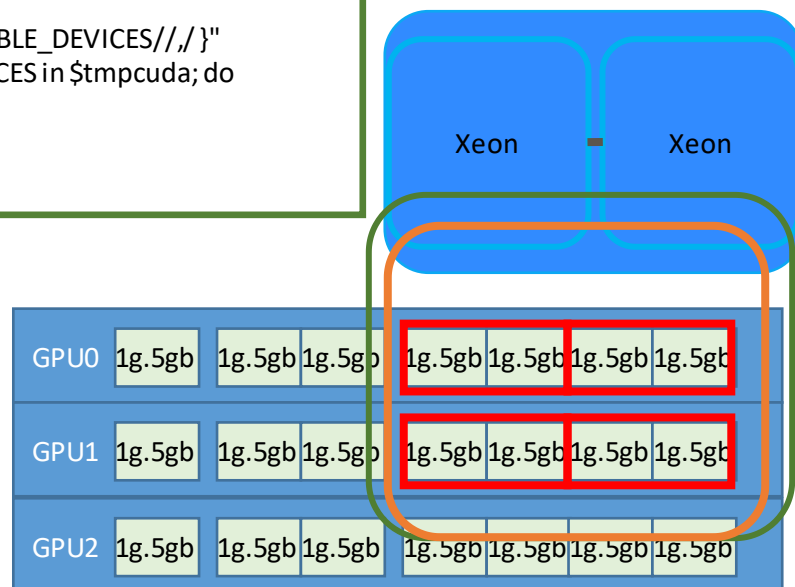
An example job script for a request that uses the resources of a multi-instance GPU is shown below.

```
#PBS -l elapstim_req=1000
#PBS --cpunum-lhost=8
#PBS --mig=4g.20gb:2,2
#PBS --mig=4g.20gb:2,2
```

```
tmpcuda="{CUDA_VISIBLE_DEVICES//,/}"
for CUDA_VISIBLE_DEVICES in $tmpcuda; do
    cuda.out &
done
```

 :request

 :logical host(job)





## Appendix.B Update history

### 16<sup>th</sup> edition

- Added Request Submit Using Multi-Instance GPU in chapter 11.
- Added Request using Multi-Instance GPU in appendix A.4.

### 17<sup>th</sup> edition

- Added Stdout size limit and Stderr size limit in chapter 1.2 and 1.3.

### 18<sup>th</sup> edition

- Added note when executing MPI with OpenMP in chapter 1.2.9.
- Corrected an error in the description of environment variables that can be changed with the `-v` option in chapter 1.2.16.
- Added explanation for execution with OpenMP in chapter 1.14.2 and chapter 1.14.3.
- Added note on when NQSV's socket scheduling feature is enabled in chapter 1.14.3.

### 19<sup>th</sup> edition

- Added list of the environment variables that cannot be passed to the jobs by `-V` or `-v` option in chapter 1.2.16.

### 20<sup>th</sup> edition

- Updated supported OpenMPI version in chapter 1.14.
- Added note on when using OpenMPI in chapter 1.14.2 and chapter 3.6.2.

## Index

### A

Attach to Request ..... 38

### B

Batch Request ..... 1

BMC ..... vi

BSV ..... v

### C

Custom Resource ..... 96

Customizing Information ..... 62

### D

Docker ..... 93, 106

### G

GPU ..... 81

Group of Request ..... 77

### H

HCA ..... vi, 7

Hold ..... 29

### I

IB ..... vi

Idle Timer ..... 46

Interactive Request ..... 45

### J

JM ..... v

Job Script ..... 1

JSV ..... v

### L

Limit per Group and User ..... 79

### M

Move ..... 31

MPI ..... vi

MPI Request ..... 32, 50

### N

NEC MPI ..... 50

Network Request ..... 58

NIC ..... vi

<b>O</b>	
OpenStack .....	89
Output File.....	32, 43
<b>P</b>	
Parametric Request .....	40
provisioning.....	89, 93, 106
<b>Q</b>	
qlogin.....	45
<b>R</b>	
Request log .....	38
Resume .....	30
<b>S</b>	
Sorting Information .....	68
Suspend.....	30
<b>U</b>	
User Pre-Post Script.....	85
UserPP script.....	85
<b>V</b>	
VE.....	v, 6
VH .....	v
VI.....	v
<b>W</b>	
Workflow .....	71
workflow script .....	72

**NEC Network Queuing System V (NQS-V) User's Guide**  
**[Operation]**

January 2025 20<sup>th</sup> edition

**NEC Corporation**

Copyright: NEC Corporation 2018

No part of this guide shall be reproduced, modified or transmitted without a written permission from NEC Corporation.

The information contained in this guide may be changed in the future without prior notice.