RCCS Reference Manual

■[Last update] 2021-Aug-16

Table of Contents

- Connection
  - Login to RCCS
  - Register SSH Public Key and Password for Web
  - Login Shell
- Whole System of RCCS
- RCCS Resources
  - CPU Points and Queue Factor
  - Checking Resources
  - Individual Limitation of Resources
- Queueing System
  - Overview of Queueing System
  - Queue Classes
  - Show Job Status
  - Submit Your Jobs
  - Delete Jobs
  - Hold/Release Jobs
  - Get Information of Finished Jobs
- Build and Run
  - Command to Build
  - Running Parallel Program
  - Development Tools
  - Environment Modules
- Package Programs
- Other RCCS Special Commands
  - Queueing System Related
  - Job wait time estimator (waitest)
  - Show Resources Used
  - Utility Commands for Batch Jobs
  - Manipulation of Files on Computation Nodes
- Inquiry

Connection

Login to RCCS

- Frontend nodes (ccfep.ims.ac.jp) can be accessed using ssh with public key authentication.
- GPU equipped frontend nodes ('ccgpup', 'ccgpuv') can be accessed from 'ccfep'.
- All computers will stop during 9:00-19:00 on the first Monday of each month because of maintenance. The maintenance time might be extended.
- Access to frontend nodes is allowed only from IPv4 address assigned to Japan or other registered IP addresses. See Application for SSH connection from outside Japan for details.

Register SSH Public Key and Password for Web

Please prepare a public/private key pair of ssh. If you do not know the procedure, please search in internet by yourself.

First registration / Missing your username or password for web

1. Open https://ccportal.ims.ac.jp/en/user/password to request mail for registration in web browser.
2. Fill your email which is written in your application, then press button "E-mail new password".
3. After you recieve an email from RCCS, open URL in the mail to login in web browser.
4. Fill your new password in "Password" and "Confirm password".
5. Paste your public key in "Public key".
6. Press "Save" button.

Using your username and password for web

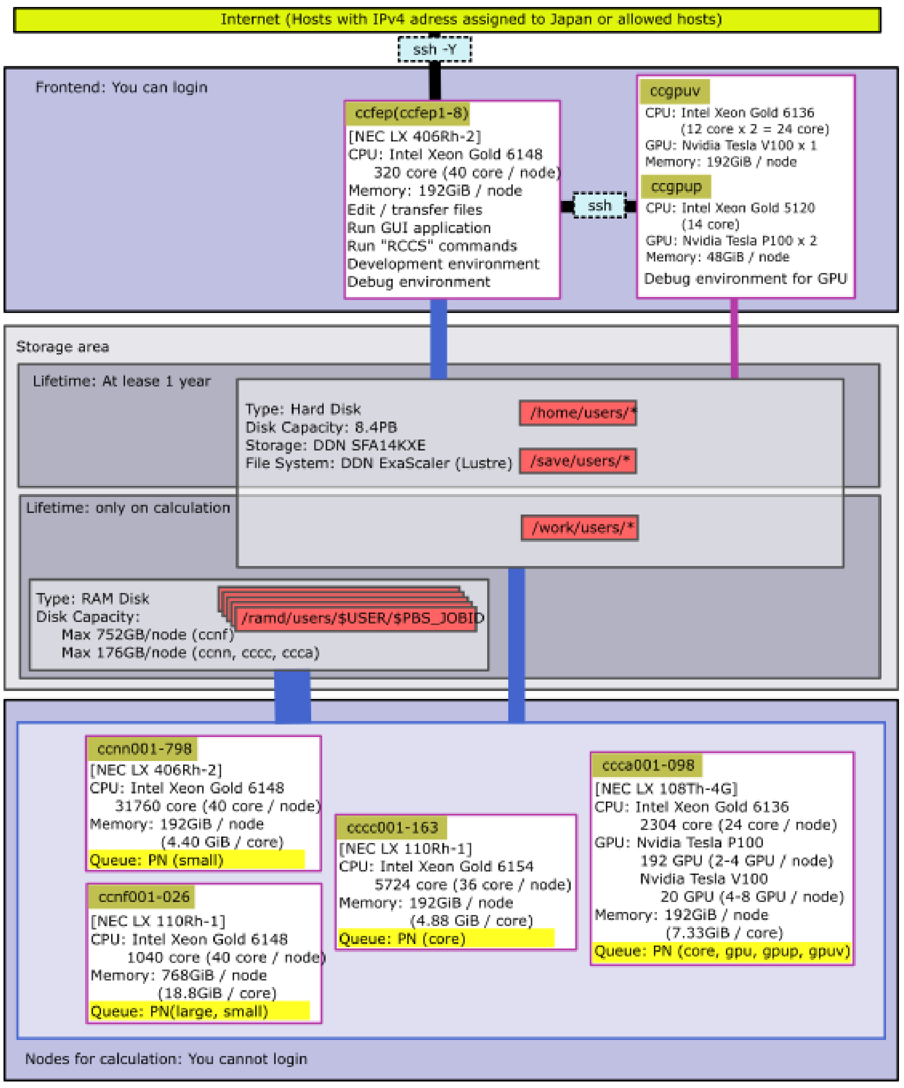
1. Open https://ccportal.ims.ac.jp/en/frontpage in web browser, then fill your username and password and press "Log in" button.
2. Press "My account" which is located in top right corner.
3. Press "Edit" tab.
4. To change password, fill current password and new passwords.
5. Paste your public key.
6. Press "Save" button.

Login Shell

- /bin/csh(tcsh), /bin/bash and /bin/zsh are available.
- You can select login shell in the sampe page as ssh public key. It will take some time to change login shell.
- You can custumize your .login or .cshrc in your home directory, but be carefully.

Whole System of RCCS

- Whole system of RCCS is shown in the figure below.
- Interactive nodes are ccfep (8 nodes), ccgpup, ccgpuv. You can build or debug applications on them.
- ccfep (8 nodes) is the login node. You can't login to other interactive nodes from internet.
- There are four kinds of disks, namely /work, /ramd, /home and /save. Access speed and data lifetime are different.
- Width of lines between disks and computers represents transfer speed. Wider is faster.
- Disk /work is temporary space for your calculation. All files will be DELETED after the completion of your job.
- Disk /ramd is RAM disk of each node. The size is about 176 GB or 752 GB, depending on node type. The sum of memory used in the job and RAM disk is controlled by the queuing system.
- There are no differences between /home and /save other than the names. (Formerly, there had been a difference regarding backuping policy.)
- Use of /tmp, /var/tmp or /dev/shm is not allowed. Jobs using those dirctories will be killed by the administrator.



RCCS Resources

CPU Points and Queue Factor

CPU points are spent when you use CPU or GPU.
Queue factors are determined as follows on each systems.

System	CPU Queue Factor	GPU Queue Factor
cclx (jobtype=large)	42 / (point/(1 node * 1 hour))	-
cclx (jobtype=small)	28 / (point/(1 node * 1 hour))	-
cclx (jobtype=core)	1.0 / (point/(1 core * 1 hour))	-
cclx (jobtype=gpu, gpup)	1.0 / (point/(1 core * 1 hour))	10 / (point(1 GPU * 1 hour))
cclx (jobtype=gpuv)	1.0 / (point/(1 core * 1 hour))	15 / (point(1 GPU * 1 hour))

- On ccfep and ccgpuv, CPU points will be consumed according to CPU time spent.
- On ccgpup, CPU points won't be required.
- On computation nodes, CPU points will be calculated from the elaps of processes.
- We don't charge money for the supercomputer usage.

If you want to know your current CPU points, run "showlim -c".

Checking Resources

- CPU points used by batch jobs and total disk usage (corresponding to "showlim -c" and "showlim -d", respectively) are updated every 10 minutes.
- CPU points used in interactive nodes are updated on 5:15.
- If CPU points run out, all running jobs of your group will be killed and further job submissions won't be allowed.
- If your disk usage exceed the limit, new job submissions will be rejected.

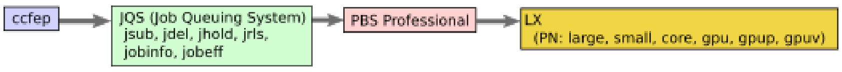
Individual Limitation of Resources

Access to "Limiting Resources Page" with your web browser.

- Only representative user can limit maximum resources of each members.
- Normal user can only view the values of maximum resources.
- Maximum available number of cpus, point and amount of disks can be limitted.

Queueing System

Overview of Queueing System


ccfep
JQS (Job Queuing System) jsub, jdel, jhold, jrls, jobinfo, jobeff
PBS Professional
LX (PN: large, small, core, gpu, gpup, gpuv)

Queue Classes

Queue class for all users

System	Class	Node	Memory	Limitation for a job	# of cores per group		# of jobs per group	
					Assigned points	# of cores/gpus	Assigned points	# of jobs
cclx	PN (jobtype=large)	ccnf	18.8GB/core	1-10 nodes (40-400 cores)	3,000,000 - 1,000,000 - 300,000 - 100,000 - - 100,000	4000/72 2560/48 1600/30 960/18 320/12	3,000,000 - 1,000,000 - 300,000 - 100,000 - - 100,000	4000 2560 1600 960 320
cclx	PN (jobtype=small)	ccnn ccnf	4.4GB/core	1-32 nodes (40-1280 cores)				
cclx	PN (jobtype=core)	cccc ccca	4.8GB/core	1-36 cores				
cclx	PN (jobtype=gpu, gpup)	ccca	7.3GB/core	1-48 gpus 2-24 cores/node (2 gpus/node) 1-12 cores/node (1 gpu/node)				
cclx	PN (jobtype=gpuv)	ccca	7.3GB/core	1-8 gpus 1-3 cores/gpu (multinode jobs not allowed)				

- Max elaps time for jobtype=core and ncpus>18 jobs must be <= 1 week (168 hours).
- Max elaps time for jobtype=gpuv jobs must be <= 1 week (168 hours).
- Max elaps time for the other jobs is until next maintenance. Only half nodes can be assigned for over 1 week jobs.
- A job which use <= 526 nodes should be assigned in the same Omni-Path group.
- 272 nodes of Node "ccnn" are used only for 1-4 node jobs.
- Jobtype "small" jobs whose walltime is less than 1 day might use Node "ccnf".
- Jobtype "core" jobs whose walltime is less than 1 day and requested cores is from 4 to 18 might use Node "ccca".
- GPU Direct Peer-to-peer communication is available in any ccca node (jobtype = gpup, gpuv, or gpu).
  - jobtype=gpup jobs shall run on NVIDIA Tesla P100 equipped nodes.
  - jobtype=gpuv jobs shall run on NVIDIA Tesla V100 equipped node.
  - jobtype=gpu jobs will run on either of P100 or V100 equipped nodes.

Special queue class

The settings of queue class are following.

System	Class	Wall Time	Memory	# of cores per job	# of cores per group
cclx	(occupy)	7 days	4.4GB/core	ask us	allowed number of cores

Show Job Status

```
ccfep% jobinfo [-h HOST] [-q QUEUE] [-c|-s|-m|-w] [-n] [-g GROUP|-a] [-n]
```

choose additional information type

One of those options can be added.

- -c show the latest result (number of GPUs and jobtype etc. are not available)
- -s show status summary of the queue(s)
- -m show memory information
- -w show working directory of jobs
- -n show node status

jobs of other users

- -g group member's jobs are also shown
  - If you belong to multiple groups, try -g [GROUP name] to specify a group.
- -a show all users' jobs
  - most of information (username etc.) will be hidden.

queue specification

You don't need to specify this usually, since all the user available queues (PN and PNR[0-9]) are the default targets.

- -h HOST: specify host type (only cclx is available)
- -q QUEUE: specify queue name (such as PN or PNR1)

Example: show queue summary

In User/Group stat, number of jobs, cpu cores, and gpus can be shown by "jobinfo -s". Limitation about those resources are also shown. In Queue Status, number of waiting jobs and available nodes/cores/gpus will be shown.

```
ccfep% jobinfo -s

User/Group Stat:

--------------------------------------------------------------------------------

 queue: PN                    | user                 | group

--------------------------------------------------------------------------------

   NJob (Run/Queue/Hold/-)      |    0/    0/    0/-    |    0/    0/    0/-

   CPUs (Run/Queue/Hold/RunLim) |    0/    0/    0/3000 |    0/    0/    0/4000

   GPUs (Run/Queue/Hold/RunLim) |    0/    0/    0/48   |    0/    0/    0/72

   core (Run/Queue/Hold/RunLim) |    0/    0/    0/500  |    0/    0/    0/500

--------------------------------------------------------------------------------

Queue Status (PN):

----------------------------------------------------------------------

     job          | free   |     free      | # jobs  |  requested

     type         | nodes  | cores (gpus) | waiting | cores (gpus)

----------------------------------------------------------------------

week jobs

----------------------------------------------------------------------

small 1-4 nodes   |      0 |      0       |     142 |  11400

small 5+  nodes   |      0 |      0       |      17 |  10600

large             |      0 |      0       |       0 |      0

core              |      0 |    200       |      33 |    585

gpu               |      0 |    430 (22)  |      98 |    252 (98)

----------------------------------------------------------------------

long jobs

----------------------------------------------------------------------

small 1-4 nodes   |      0 |      0       |       1 |     40

small 5+  nodes   |      0 |      0       |       3 |   1920

large             |      0 |      0       |       0 |      0

core              |      0 |    133       |       0 |      0

gpu               |      0 |    207 (11)  |       0 |      0 (0)

----------------------------------------------------------------------
```

"core (Run/Queue/Hold/RunLim)" in User/Group Stat is a CPU cores limit for jobtype=core/gpu* jobs, where CPU cores used by jobtype=small/large jobs are not taken into account. For example, in this example, you can use up to 500 cores in total.

Example: show status of jobs

You can see the latest status of jobs by specifying "-c" option. (-l option can be added but is ignored.)

```
ccfep% jobinfo -c

--------------------------------------------------------------------------------

Queue   Job ID Name            Status CPUs User/Grp       Elaps Node/(Reason)

--------------------------------------------------------------------------------

PN      9999900 job0.csh       Run      16  zzz/---     24:06:10 cccc047

PN      9999901 job1.csh       Run      16  zzz/---     24:03:50 cccc003

PN      9999902 job2.sh        Run       6  zzz/---      0:00:36 cccc091

PN      9999903 job3.sh        Run       6  zzz/---      0:00:36 cccc091

PN      9999904 job4.sh        Run       6  zzz/---      0:00:36 cccc090

...

PN      9999989 job89.sh       Run       1  zzz/---      0:00:11 ccca013

PN      9999990 job90.sh       Run       1  zzz/---      0:00:12 ccca010

--------------------------------------------------------------------------------
```

If you don't specify "-c", you can also see some more details (jobtype and number of gpus). The information may be slightly (2-3 minutes usually) old, though.

```
ccfep% jobinfo

--------------------------------------------------------------------------------

Queue   Job ID Name            Status CPUs User/Grp       Elaps Node/(Reason)

--------------------------------------------------------------------------------

PN(c)   9999900 job0.csh       Run      16  zzz/zz9     24:06:10 cccc047

PN(c)   9999901 job1.csh       Run      16  zzz/zz9     24:03:50 cccc003

PN(c)   9999902 job2.sh        Run       6  zzz/zz9      0:00:36 cccc091

PN(c)   9999903 job3.sh        Run       6  zzz/zz9      0:00:36 cccc091

PN(c)   9999904 job4.sh        Run       6  zzz/zz9      0:00:36 cccc090

...

PN(g)   9999989 job89.sh       Run     1+1  zzz/zz9      0:00:11 ccca013

PN(g)   9999990 job90.sh       Run     1+1  zzz/zz9      0:00:12 ccca010

--------------------------------------------------------------------------------
```

Example: show working directory

In case you forget where you ran jobs, try "-w" option. The working directories will be shown like below.

```
ccfep% jobinfo -w

--------------------------------------------------------------------------------

Queue   Job ID Name            Status Workdir
```

```
--------------------------------------------------------------------------------
PN    9999920 PN_12345.sh    Run    /home/users/zzz/gaussian/mol23
PN    9999921 PN_23456.sh    Run    /home/users/zzz/gaussian/mol74
...
```

(You can't use "-c" in this case.)

Submit Your Jobs

Description of the header part

Your have to write a script file which is written in C-shell to submit your job. An example for each system is following.

- csh, bash (/bin/sh), zsh can be used for the job submission script.
- lines started with #PBS are common, regardless of the shell type.
- Sample scripts can be found in `ccfep:/local/apl/lx/(application name)/samples/`.

Meaning	Header part	Importance
The First Line	(csh) #!/bin/csh -f (bash) #!/bin/sh (zsh) #!/bin/zsh	Required (choose one)
Needed Number of CPU	#PBS -l select=[*Nnode*:]ncpus=*Ncore*:mpiprocs=*Nproc*:ompthreads=*Nthread*:jobtype=*Jobtype*[:ngpus=*Ngpu*]	Required
Wall Time	#PBS -l walltime=*72:00:00*	Required
Mail at Start and End	#PBS -m abe	Optional
Prevent Rerun	#PBS -r n	Optional
Change to Submitted Directory	cd ${PBS_O_WORKDIR}	Recommended

- Nnode: # of physical node
- Ncore: # of reserved cores per physical node
- Nproc: # of processes per node
- Nthread: # of threads per process
- Jobtype: large, small, core, gpu, gpup, gpuv
  - large: 18.8GB / core
  - small: 4.4GB / core
  - core: job for less than 18 cores
  - gpu, gpup, gpuv: GPU jobs
- Ngpu: # of GPUs

Example of "Needed Number of CPU": Case of 80 mpi processes on 2 nodes

```
#PBS -l select=2:ncpus=40:mpiprocs=40:ompthreads=1:jobtype=small
```

Example of "Needed Number of CPU": Case of GPGPU

```
#PBS -l select=1:ncpus=6:mpiprocs=1:ompthreads=1:jobtype=gpu:ngpus=1
```

You can find some other examples in this page.

Job submission

After you write the script, type following command to submit.

```
ccfep% jsub -q (PN|PNR[0-9]) [-g  XXX] [-W depend=(afterok|afterany):JOBID1[:JOBID2...]] script.csh
```

If you want to submit your jobs by 'Supercomputing Consortium for Computational Materials Science' group, use -g option. (*XXX* is name of its group)
You can describe dependency of jobs using -W option.
If you want to describe dependency that a job should run after the dependent job exit successfully, use keyword "afterok".
If a job should run after the dependent job including abnormal exit, use keyword "afterany".
Sample script files exist in `ccfep:/local/apl/lx/*/samples/`.

Sequential Jobs (step jobs)

You can easily submit sequential jobs by using --step or --stepany command.

■step job 1 (afterok): run a series of jobs sequentially. If a job exited abnormally, following jobs will be deleted.

```
ccfep% jsub -q (PN|PNR[0-9]) [-g XXX] --step [-W depend=(afterok|afterany):JOBID1[:JOBID2...]] script.csh
script2.csh ...
```

■step job 2 (afterany): run a series of jobs sequentially. If a job finished, next job will run regardless of the exit status.

```
ccfep% jsub -q (PN|PNR[0-9]) [-g XXX] --stepany [-W depend=(afterok|afterany):JOBID1[:JOBID2...]] script.csh
script2.csh ...
```

Example:

```
ccfep% jsub -q PN --stepany job1.csh job2.csh job3.csh
```

Define variables which can be used in jobscript

You can define variables via -v option. The argument to the option is comma-separated list of variable definitions, (variable-name)=(value).

```
ccfep% jsub -v INPUT=myfile.inp,OUTPUT=myout.log script.sh
```

In this example, $INPUT and $OUTPUT will be set to myfile.inp and myout.log in "script.sh", respectively.

Notes:

- if multiple -v is specified, only the last one will be active.
- this can't define ncpus, jobtype etc. values in select= section of the jobscript.

Delete Jobs

First of all, you should get ID of the target jobs using "jobinfo" command. Then, run following command, where the RequestID is the job id.

```
ccfep% jdel [-h cclx]  RequestID
```

Hold/Release Jobs

You can prevent queued job to run by the following command (hold a job, in other words). (Use jobinfo to get target job id.)

```
ccfep% jhold [-h cclx]  RequestID
```

You can release the restiction using jrls command.

```
ccfep% jrls [-h cclx]  RequestID
```

Get Information of Finished Jobs

You can get information of finished jobs, such as finish time, elaps time, parallel efficiency, by *joblog* command.

```
ccfep% joblog [-d ndays] [-o  item1[,item2[,...]]]
```

If the target period is not specified, information about jobs of current FY will be shown. Following options can be used to specify the target period.

- -d ndays: jobs finished within last "ndays"
  - -d 7 : jobs finished within last 7 days
- -y year: jobs in FY "year"
  - -y 2021 : jobs in FY 2021 (2021/4-2022/3)
- -f YYYY[MM[DD[hh[mm]]]] -t YYYY[MM[DD[hh[mm]]]]: from-to specification. (month, day, hour, etc. can be omitted)
  - -f 202107 -t 202108 : jobs finished in July or August of year 2021.

You can customize items which are displayed in *-o* option. Available keywords are:

- queue: Queue name
- jobid: Job ID
- user: User name
- group: Group name
- node: Job head node name
- Node: All node names
- type: jobtype
- start: Start time（YYYY/MM/DD HH:MM）
- Start: Start time（YYYY/MM/DD HH:MM:SS）
- finish: Finish time（YYYY/MM/DD HH:MM）
- Finish: Finish time（YYYY/MM/DD HH:MM:SS）
- elaps: Elaps
- cputime: Total CPU time
- used_memory: Used memory size
- ncpu: Number of reserved cpu cores
- ngpu: Number of reserved gpus
- nproc: Number of MPI processes
- nsmp: Number of threads per process
- peff: Efficiency of job
- attention: Bad efficiency job
- command: Job name
- point: CPU points
- all: show all

■e.g. 1: Show job ID, start and end datetime, CPU points of jobs finished within last 10 days.

```
ccfep% joblog -d 10 -o jobid,start,finish,point
```

■e.g. 2: Show job ID, end datetime, CPU points, and workding directory of jobs in FY2020.

```
ccfep% joblog -y 2020 -o jobid,finish,point,Workdir
```

■e.g. 3: Show all the parameters of jobs finished within last two days.

```
ccfep% joblog -d 2 -o all
```

Build and Run

Command to Build

System	Language	Non-Parallel	Auto-Parallel	OpenMP	MPI
cclx (Intel)	Fortran	ifort	ifort -parallel	ifort -qopenmp	mpiifort
	C	icc	icc -parallel	icc -qopenmp	mpiicc
	C++	icpc	icpc -parallel	icpc -qopenmp	mpiicpc
cclx (PGI)	Fortran	pgfortran	pgfortran -Mconcur	pgfortran -mp	
	C	pgcc	pgcc -Mconcur	pgcc -mp	
	C++	pgcpp	pgcpp -Mconcur	pgcpp -mp	

Please also check the package program list page for the available libraries and MPI environments.

Running Parallel Program

cclx

Please specify correct numbers of MPI processes and OpenMP threads in "mpiprocs" and "ompthreads" values of select line in your job script.
e.g. 12 CPUs, 4 MPI processes and 3 OpenMP threads case -> ncpus=12:mpiprocs=4:ompthreads=3

Please also check sample jobs scripts by RCCS (locating at /local/apl/(system name)/(software name)/samples).

Number of threads specification for OpenMP jobs

When submitting job using "jsub", the value specified by "ompthreads" is considered as number of OpenMP threads.
You can specify or change the value by setting OMP_NUM_THREADS envirnment variable manually.
If you run a command not via "jsub" (on frontend node for example), you should set OMP_NUM_THREADS environment variable manually.

Host list specification for MPI

List of hosts can be found in the file name specified in PBS_NODEFILE environment variable.
MPI environments installed by RCCS (Intel MPI and OpenMPI) automatically consider this environment variable.
Therefore, you can skip specification of machine file when invoking "mpirun" command.

e.g. 4 MPI * 3 OpenMP hybrid parallel

```
#!/bin/sh
#PBS -l select=1:ncpus=12:mpiprocs=4:ompthreads=3:jobtype=core
#PBS -l walltime=24:00:00
cd $PBS_O_WORKDIR
mpirun -np 4 /some/where/my/program options
```

- OpenMP thread number is determined from the value specified by "ompthreads" in select. (equivalent to OMP_NUM_THREADS=3)
- MPI machine file name can be omitted when invoking mpirun (if that mpirun is installed by RCCS).
    - You can specify machine file to filename specified in PBS_NODEFILE env variable. (This doesn't change anything.)

Development Tools

Some tools can be used in command line, but it is better to use X Window edition's.

Intel Inspector XE

- Memory / Thread inspector
- (GUI command) inspxe-gui
- (CUI command) inspxe-cl

Intel Vtune Amplifier XE

- Hotspot analizer
- (GUI command) amplxe-gui
- (CUI command) amplxe-cl

Allinea Forge

- Debugger
- (GUI command) ddt

Environment Modules

Environment Modules ("module" command) is available from July 2018. See this page for detailed information.

Package Programs

- The installed pacakge programs for each systems are listed in https://ccportal.ims.ac.jp/en/installed_applications.
- Sample script files are located in `ccfep:/local/apl/lx/`*`appname`*`/samples/`.
- Installed directories are located in `/local/apl/lx/`*`appname`*`/` on each systems.
- Applications compiled by center are listed in https://ccportal.ims.ac.jp/en/how_to_configure with detail description.

Request installation you want to use

Please fill the following items and send it rccs-admin[at]ims.ac.jp.

- Software name and version that you want to use
- Overview of the software and its feature
- Necessity of installation to supercomputers in RCCS
- URL of the software development

Special Commands of RCCS

Related Queueing System

You can find descriptions about "jobinfo", "jsub", "jdel", "jhold", "jrls", and "joblog" above in this page.

Submitting Gaussian Jobs

Case of Gaussian 16

```
ccfep% g16sub [-q "QUE_NAME"] [-j "jobtype"] [-g "XXX"] [-walltime "hh:mm:ss"] [-noedit] \
          [-rev "g16xxx"] [-np "ncpus"] [-ngpus "n"] [-mem "size"] [-save] [-mail] input_files
```

- Command "g09sub" are also available to use Gaussian 09.
- Default walltime is set to 72 hours. Please set excepted time for calculation and extra to do long job or run early.
- If you want to know the meaning of options and examples, please just type "g16sub".
- %mem, %nproc, %cpu in the original input files will be overwritten by g16sub / g09sub. Please specify those values with -mem, -np etc. command line options.
  - You can prevent this automatic overwrite by -noedit option. But this is not recommended.
  - Safe maximum %mem value will be automatically assigned by g09sub/g16sub. You may not need to set that manually unless you need to reduce the amount of memory.

example case: g16c01, timelimit 24 hours, 12 cores job (Gaussian input file name is my_gaussian_input.gjf)

```
ccfep% g16sub -rev g16c01 -walltime 24:00:00 -np 12 my_gaussian_input.gjf
```

Job wait time estimator (waitest)

On ccfep, you can estimate start time of job with "waitest" command, where waitest assumes jobs will run for the time specified with "walltime" parameter". Therefore, the estimated datetime will be the worst case estimation. On the other hand, other user's jobs submitted later might run earlier than your ones. (Many parameters, such as job priority, jobtype, and remedies for (small) groups, are involved.) You shouldn't expect high accuracy for the estimation. You may also need to check queue status (jobinfo -s) additionally.

Basic Usage

Estimate start time of submitted job(s):

```
$ waitest [jobid1] ([jobid2] ...)
```

Estimate start time of not yet submitted job(s):

```
$ waitest -s [job script1] ([jobscript2] ...)
```

Example 1: estimate start time of a submitted job

```
[user@ccfep2]$ waitest 4923556
Current Date   : 2019-10-15 14:32:30
2019-10-15 14:32:30 ...
2019-10-15 16:40:44 ...
2019-10-15 22:26:07 ...
2019-10-16 00:43:43 ...
2019-10-16 03:03:11 ...
2019-10-16 05:58:00 ...
2019-10-16 11:34:12 ...
Job 4923556 will run at 2019-10-16 13:03:11 on ccnn500,ccnn496,ccnn497,ccnn494,ccnn495,ccnn489.
Estimation completed.
```

Example 2: estimate start time of not yet submitted jobs

```
[user@ccfep2]$ waitest -s run_small_4N.sh run_small_6N.sh run_small_8N.sh
Current Date   : 2019-10-15 14:34:39

Job Mapping "run_small_4N.sh" -> jobid=1000000000
Job Mapping "run_small_6N.sh" -> jobid=1000000001
Job Mapping "run_small_8N.sh" -> jobid=1000000002

2019-10-15 14:34:39 ...
2019-10-15 16:40:52 ...
2019-10-15 22:26:15 ...
2019-10-16 00:43:51 ...
2019-10-16 03:03:18 ...
2019-10-16 05:58:08 ...
2019-10-16 11:34:10 ...
Job 1000000001 will run at 2019-10-16 13:03:18 on ccnn500,ccnn496,ccnn497,ccnn494,ccnn495,ccnn489.
2019-10-16 13:28:41 ...
2019-10-16 16:00:36 ...
2019-10-16 20:52:10 ...
2019-10-17 01:08:27 ...
Job 1000000002 will run at 2019-10-17 03:03:18 on
ccnn458,ccnn329,ccnn515,ccnn520,ccnn373,ccnn437,ccnn380,ccnn352.
2019-10-17 03:33:56 ...
```

```
2019-10-17 06:43:51 ...
2019-10-17 08:50:03 ...
2019-10-17 11:34:10 ...
2019-10-17 13:03:18 ...
2019-10-17 16:08:16 ...
2019-10-17 18:35:08 ...
2019-10-17 20:36:56 ...
2019-10-17 22:38:49 ...
Job 1000000000 will run at 2019-10-18 00:18:34 on ccnn760,ccnn789,ccnn791,ccnn787.
Estimation completed.
```

(In some case, larger jobs will run first due to the job priority and other parameters.)

Example 3: show periodically estimated wait time for general types of jobs

For some of basic types of jobs, wait time for those jobs are estimated periodically. The result can be accessed via the following command.

```
[user@ccfep2]$ waitest --showref
```

Showing Used Resources

```
ccfep% showlim (-cpu|-c|-disk|-d) [-m]
```

- -cpu|-c: Show used point and limited value.
- -disk|-d: Show current disk size and limited value.
- -m: Show values of each members.

■example 1: show CPU points (approved and used) of YOU and WHOLE YOUR GROUP

```
ccfep% showlim -c
```

■example 2: show CPU points (approved and used) of YOU, YOUR GROUP MEMBERS, and WHOLE GROUP

```
ccfep% showlim -c -m
```

■example 3: show disk status (approved and used) of YOU, YOUR GROUP MEMBERS, and WHOLE GROUP

```
ccfep% showlim -d -m
```

Utility Commands for Batch Jobs

Limit the walltime of command

```
/local/apl/lx/ps_walltime -d duration -- command [arguments...]
```

- -d *duration*: Duration to execute command is described as "-d 72:00:00".
- Command will be killed after specified duriation.

Showing statistic of current job

```
/local/apl/lx/jobstatistic
```

- Show similar statistic information in the mail notification of end of your job when you describe in PBS header line.
- This statistic information is the information when the command is executed.

■Output Items

- resources_used.cpupercent: Efficiency of CPU usage. Maximum value is multiplication the number of threads and 100. Illegal value will be shown when multi nodes are used.
- resources_used.cput: Sum of actual time for calculation in each CPUs.
- resources_used.mem: Amount of actual memory size.
- resources_used.ncpus: Number of actually used CPUs.
- resources_used.walltime: Real time for calculation.

Manipulation of Files on Computation Nodes

You can access local files on computation nodes which cannot be directly accessed from the frontend nodes (ccfep) via "remsh" command.

```
remsh hostname command options
```

- hostname: Hostname such as cccc???, cccca???, ccnn???, or ccnf???.
- command: Command to be executed on the node. Acceptable commands are: ls, cat, cp, and find.
- options: Commandline arguments to the comannd.

Example: how to access ramdisk of a computation node, ccnnXXX, by user "zzz"

```
remsh ccnnXXX ls /ramd/users/zzz
```

```
remsh ccnnXXX cat /ramd/users/zzz/99999/fort.10 | tail
```

Host names and jobids of your jobs can be found in the output of "jobinfo" command. (see above for the usage)

Inquiry

See https://ccportal.ims.ac.jp/en/contact.