대형연구장비 초극저온전자현미경 데이터 분석을 위한 컴퓨팅 팜 구축 및 활용 보고서

Release 1.0

국가슈퍼컴퓨팅본부 대용량데이터허브센터

Nov 14, 2019

CONTENTS:

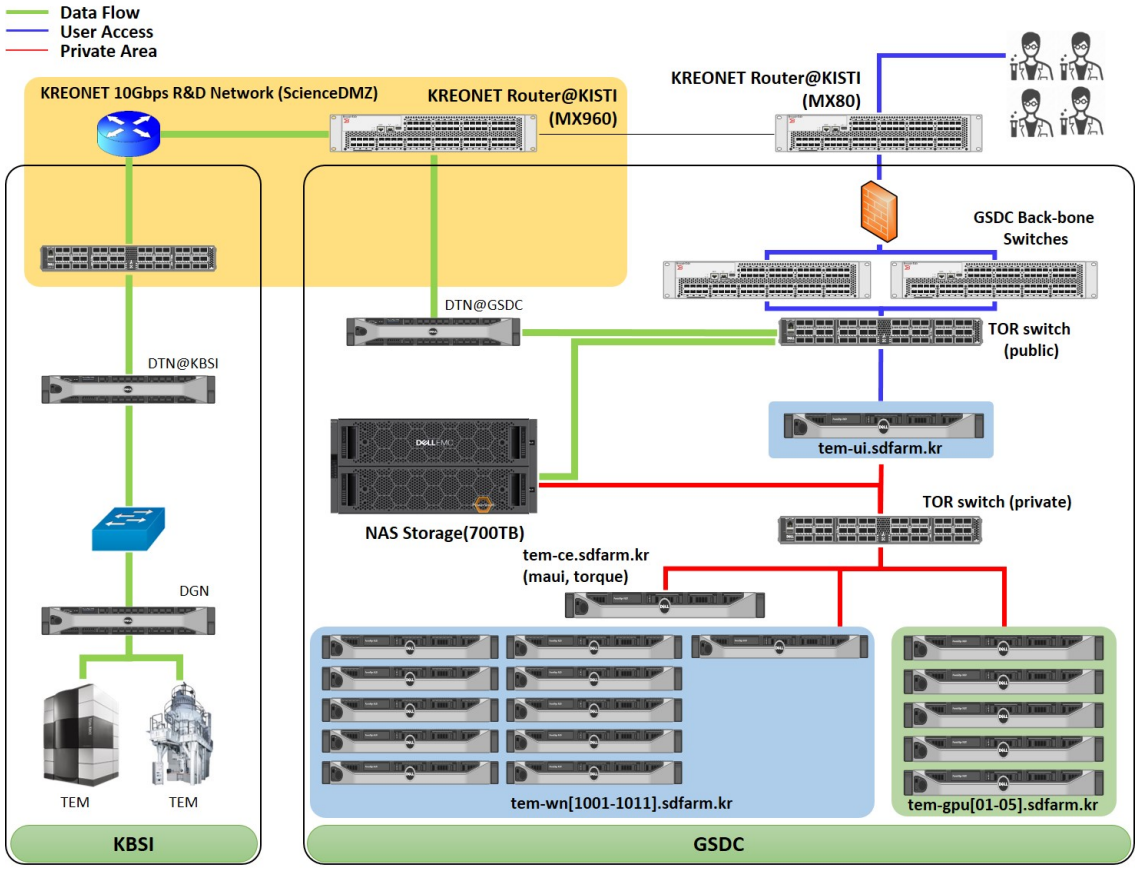
CHAPTER

ONE

TEM INFRASTRUCTURE OVERVIEW

GSDC (Global Science experimental Data hub Center) supports data processing for Structural Biology with Cryo-EM, Lightsource, and X-ray Laser experiments. Cryo-EM instrumentations are operated by KBSI. Cryo-EM facilities are directly connected to GSDC Datacenter through KREONet with 10Gbps dedicated/shared optical fiber links. GSDC provides Peta-bytes scale of storages and GPU equipped computational power. Here is an overview of GSDC's TEM infrastructre for Cryo-EM users.

- Overall architecture between KBSI's Cryo-EM facilities and GSDC's TEM service farm



1.1 Computing and storage resources

- Hardware specification of TEM service farm

Category	Name	Specification	Resources size
Login	tem-ui.sdfarm.kr	• CPU : Intel(R) Xeon(R) CPU E5-2697v3 @ 2.60GHz 14Core * 2 CPUs • RAM : DDR4 8GB * 24 (192GB) • HDD : 12G SAS HDD 1.2TB * 2EA (RAID-1)	28 cores
Computing (master)	tem-ce.sdfarm.kr	• CPU : Intel(R) Xeon(R) CPU E5-2697v3 @ 2.60GHz 14Core * 2 CPUs • RAM : DDR4 8GB * 24 (192GB) • HDD : 12G SAS HDD 1.2TB * 2EA (RAID-1)	28 cores
Computing (workers)	tem-wn[1001-1011].sdfarm.kr	• CPU : Intel(R) Xeon(R) CPU E5-2697v3 @ 2.60GHz 14Core * 2 CPUs • RAM : DDR4 8GB * 24 (192GB) • HDD : 12G SAS HDD 1.2TB * 2EA (RAID-1)	308 cores
	tem-gpu[01-05].sdfarm.kr	• CPU : Intel® Xeon® CPU E5-2690v4 @ 2.60GHz 14Core * 2 CPUs • RAM : DDR4 16GB * 24 (384GB) • SSD : 6G SATA SSD 800GB * 2EA (RAID-1) • GPU : NVIDIA P100 * 2ea (tem-gpu[01-03]) • GPU : NVIDIA P40 * 2ea (tem-gpu[04-05])	140 cores
Storage	Dell EMC Isilon NAS	Network attached storage 700 TB	
Total		504 CPU cores, 10 GPGPUs, 700TB Storage	

1.2 Cluster management softwares

Category	Name	Description	Version (module path)
OS	Scientific Linux	Operating system	6.x
System M/W	Environment module	• Module environment • https://modules.readthedocs.io/en/latest	v3.2.10
	OpenPBS(torque)	• Cluster resources management • http://www.adaptivecomputing.com/products/torque	v6.1.2
	OpenMPI	• Messaging Pass Interface(MPI) • Reference implementation for MPI standard • https://www.open-mpi.org	v1.8.8 (mpi/gcc/openmpi/1.8.8)
	cuda	• Compute Unified Device Architecture(CUDA) • NVIDIA CUDA Runtime & Toolkit • https://developer.nvidia.com/cuda-toolkit	9.1 (cuda/9.1)
	python	• Python runtime	v2.6.6

1.3 Data analysis tools

Category	Name	Description	Version (module path)
Tools	**Relion**	A stand-alone computer program that employs an empirical Bayesian approach to refinement of (multiple) 3D reconstructions or 2D class averages in electron cryo-microscopy (cryo-EM). • https://www3.mrc-lmb.cam.ac.uk/relion/index.php	v3.0.7 (apps/gcc/4.4.7/relion/cpu/3.0.7) (apps/gcc/4.4.7/relion/gpu/3.0.7)
	cisTEM	User-friendly software to process cryo-EM images of macromolecular complexes and obtain high-resolution 3D reconstructions. • https://cistem.org	v1.0.0 (apps/gcc/4.4.7/cistem/1.0.0)
	CryoSPARC	CryoSPARC is the state-of-the-art platform used globally for obtaining 3D structural information from single particle cryo-EM data. • https://cryosparc.com	Not deployed yet (TBD)

CHAPTER

TWO

TEM FARM BASICS

2.1 Accessing TEM service farm

Before you use GSDC's service farm, you should send an application form to TEM service manager and get an user account to access the farm (please see the contact information for the application form). If you already have valid user accounts, you can log into UI (user interface) nodes to access/use various kind of cluster resources and software environments (including data analysis tools, e.g., relion, cisTEM, eman, etc.).

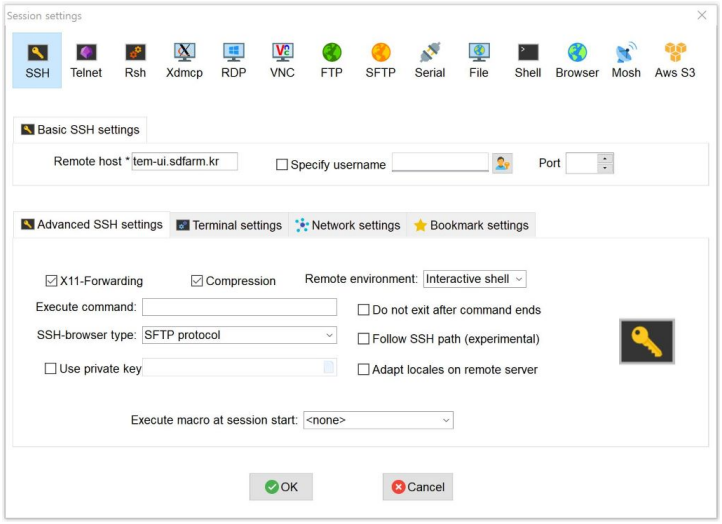
2.1.1 For Linux/Mac users

```
$> ssh -Y -o Port=<port> <userID>@tem-ui.sdfarm.kr
```

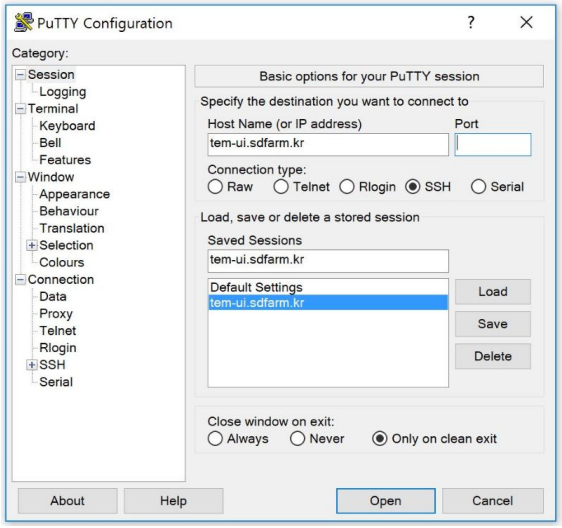
-Y (or -X) options : enable trusted X11 (or untrusted X11) forwarding

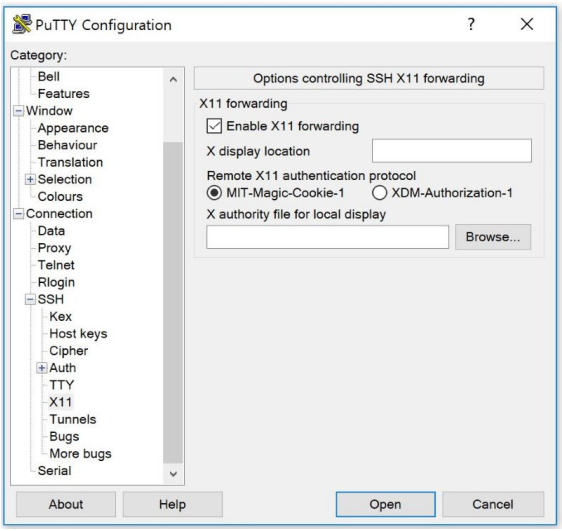
2.1.2 For Windows users

- Using MobaXterm (https://mobaxterm.mobatek.net) : MobaXterm is an enhanced terminal for Windows with self-contained X11 server, tabbed SSH client, network tools and much more.



- Using Putty with Xwindows manager (e.g., Xming, Xmanager, etc.) (https://www.putty.org) : If you use putty terminal application, you must install a 3rd-party Xwindows manager in advance.





2.2 Understanding environment modules

The Environment Modules system is a tool to help users manage their Unix or Linux shell environment, by allowing groups of related environment-variable settings to be made or removed dynamically.

- **Listing available modules**

```
$> module avail
----------- /tem/home/tem/Modules/Modules/default/modulefiles ------------
apps/gcc/4.4.7/cistem/1.0.0     cuda/9.1
apps/gcc/4.4.7/relion/cpu/3.0.7 modules
apps/gcc/4.4.7/relion/gpu/3.0.7 mpi/gcc/openmpi/1.8.8
```

- **Show module details**

```
$> module show apps/gcc/4.4.7/relion/gpu/3.0.7
-------------------------------------------------------------------
/tem/home/tem/Modules/Modules/default/modulefiles/apps/gcc/4.4.7/relion/gpu/3.0.7:
```

(continues on next page)

(continued from previous page)

```
module-whatis    Setups `relion-3.0.7' environment variables
module           load mpi/gcc/openmpi/1.8.8
module           load cuda/9.1
setenv           relion_version 3.0.7
prepend-path     PATH /tem/home/tem/_Applications/relion-3.0.7/gpu/bin
prepend-path     LD_LIBRARY_PATH /tem/home/tem/_Applications/relion-3.0.7/gpu/lib
setenv           LANG en_US.UTF-8
setenv           RELION_QUEUE_NAME tem
setenv           RELION_QSUB_COMMAND qsub
setenv           RELION_QSUB_TEMPLATE /tem/home/tem/_Applications/relion-3.0.7/gpu/
↪bin/qsub-relion3-gpu.bash
setenv           RELION_QSUB_EXTRA_COUNT 3
setenv           RELION_QSUB_EXTRA1 Number of Nodes
setenv           RELION_QSUB_EXTRA2 Number of processes per each node
setenv           RELION_QSUB_EXTRA3 Number of GPUs per node
setenv           RELION_QSUB_EXTRA1_DEFAULT 1
setenv           RELION_QSUB_EXTRA2_DEFAULT 3
setenv           RELION_QSUB_EXTRA3_DEFAULT 2
setenv           RELION_CTFFIND_EXECUTABLE /tem/home/tem/_Applications/ctffind-4.1.13/
↪bin/ctffind
setenv           RELION_GCTF_EXECUTABLE /tem/home/tem/_Applications/Gctf_v1.18_b2/bin/
↪Gctf_v1.18_b2_sm60_cu9.1
setenv           RELION_RESMAP_EXECUTABLE /tem/home/tem/_Applications/ResMap-1.1.4/
↪ResMap-1.1.4-linux64
setenv           RELION_MOTIONCOR2_EXECUTABLE /tem/home/tem/_Applications/MotionCor2/
↪MotionCor2_Cuda9.1_v1.0.5
setenv           RELION_UNBLUR_EXECUTABLE /tem/home/tem/_Applications/unblur_1.0.2/
↪bin/unblur_openmp_7_17_15.exe
setenv           RELION_SUMMOVIE_EXECUTABLE /tem/home/tem/_Applications/summovie_1.0.
↪2/bin/sum_movie_openmp_7_17_15.exe
conflict         apps/gcc/4.4.7/relion
-------------------------------------------------------------------
```

- **Loading modules**

```
$> module load <module_path>
or
$> module add <module_path>
e.g., $> module load apps/gcc/4.4.7/relion/gpu/3.0.7
```

- **Listing loaded modules**

```
$> module load apps/gcc/4.4.7/relion/gpu/3.0.7
$> module list
Currently Loaded Modulefiles:
1) cuda/9.1                          2) mpi/gcc/openmpi/1.8.8            3) apps/gcc/
↪4.4.7/relion/gpu/3.0.7
```

- **Unloading modules**

```
$> module unload <module_path>
or
$> module rm <module_path>
e.g., $> module unload apps/gcc/4.4.7/relion/gpu/3.0.7
```

- **Unloading all the modules**

```
$> module purge
```

- **Module environment help**

```
$> module --help
  Modules Release 3.2.10 2012-12-21 (Copyright GNU GPL v2 1991):

  Usage: module [ switches ] [ subcommand ] [subcommand-args ]

  Switches:
      -H|--help               this usage info
      -V|--version            modules version & configuration options
      -f|--force              force active dependency resolution
      -t|--terse              terse    format avail and list format
      -l|--long               long     format avail and list format
      -h|--human              readable format avail and list format
      -v|--verbose            enable  verbose messages
      -s|--silent             disable verbose messages
      -c|--create             create caches for avail and apropos
      -i|--icase              case insensitive
      -u|--userlvl <lvl>      set user level to (nov[ice],exp[ert],adv[anced])
  Available SubCommands and Args:
      + add|load              modulefile [modulefile ...]
      + rm|unload             modulefile [modulefile ...]
      + switch|swap           [modulefile1] modulefile2
      + display|show          modulefile [modulefile ...]
      + avail                  [modulefile [modulefile ...]]
      + use [-a|--append]     dir [dir ...]
      + unuse                  dir [dir ...]
      + update
      + refresh
      + purge
      + list
      + clear
      + help                   [modulefile [modulefile ...]]
      + whatis                 [modulefile [modulefile ...]]
      + apropos|keyword       string
      + initadd                modulefile [modulefile ...]
      + initprepend            modulefile [modulefile ...]
      + initrm                 modulefile [modulefile ...]
      + initswitch             modulefile1 modulefile2
      + initlist
      + initclear
```

2.3 Job manager (Torque)

2.3.1 Resources manager and job scheduler

- Resource manager : Torque(OpenPBS) v6.1.2
- Job scheduler : Torque default FIFO job scheduler

2.3.2 Directives in Torque job scripts

Torque defines some useful directives (starting with '#PBS') which can be used to describe job's resources requirements. Users must include those directives in job scripts to submit and execute jobs. The order of directives is not important, but the directives must be written prior to job execution commands.

Resource limits

The "-l" option is used to request resources, including nodes, memory, time, etc.

- Nodes and PPN (Processor Per Node)

```
To request a single core on the farm:
#PBS -l nodes=1:ppn=1

To request one whole node on the farm:
#PBS -l nodes=1:ppn=28

To request 4 whole nodes on the farm:
#PBS -l nodes=4:ppn=28

To request 3 whole nodes with 2 GPUs on the farm:
#PBS -l nodes=3:ppn=28:gpus=2

To request 1 node with use of 6 cores and 1 GPU:
#PBS -l nodes=1:ppn=6:gpus=1
```

- Wall clock time

```
To request 20 hours of wall clock time:
#PBS -l walltime=20:00:00
```

If a computational job will have not finished yet until the specified wall clock time, Torque (or maui scheduler) will release the resources that are allocated to the job and stop the job's runnning. If you don't define walltime, the default value is "infinite".

- Memory

```
To request 4GB memory:
#PBS -l mem=4GB
or
#PBS -l mem=4000MB

To request 24GB memory:
#PBS -l mem=24000MB
```

Job name

You can define a job name using "-N" option. If you omit this directive, the default job name is the same as the file name of job script.

```
#PBS -N my_first_job
```

Queue name

In general, a "queue" can be thought of a mapped set of computing resources. You can specify a queue name (using "-q" option) which the job is enqueued to.

```
#PBS -q tem
```

Job log files

When Torque executes an user's job, Torque creates 2 different types of log files (standard output stream and standart error stream) by default. If the job's name is "my_first_job" and the submitted job ID is "123456", you can find 2 files (my_first_job.o123456 and my_first_job.e123456) that are created in the job execution base directory. You can also merge the two streams into one file using "-j oe" option. In that case, my_first_job.o1234567 file contains the standard error stream.

```
#PBS -j oe
```

2.3.3 Torque job script examples

Simple sequential job

```
#PBS -N my_job
#PBS -l walltime=40:00:00
#PBS -l nodes=1:ppn=1
#PBS -q tem

cd $PBS_O_WORKDIR
/usr/bin/time ./mysci > mysci.hist
```

Serial job with OpenMP multithreading

```
#PBS -N my_job
#PBS -l walltime=1:00:00
#PBS -l nodes=1:ppn=28
#PBS -q tem

export OMP_NUM_THREADS=28
cd $PBS_O_WORKDIR
./a.out > my_results
```

Simple MPI parallel job

Here is an example of an MPI job that uses 4 nodes with 4 cores each, running one process per core (16 processes total).

```
#PBS -N my_job
#PBS -l walltime=10:00:00
#PBS -l nodes=4:ppn=4
#PBS -q tem
```

(continues on next page)

(continued from previous page)

```
module load mpi/gcc/openmpi/1.8.8
cd $PBS_O_WORKDIR
mpirun -machinefile $PBS_NODEFILE ./a.out
```

Parallel job with MPI and OpenMP

This example is a hybrid MPI/OpenMP job. It runs one MPI process per node with 28 threads per process. The assumption here is that the code was written to support multi-level parallelism.

```
#PBS -N my_job
#PBS -l walltime=20:00:00
#PBS -l nodes=4:ppn=28
#PBS -q tem

module load mpi/gcc/openmpi/1.8.8
export OMP_NUM_THREADS=28
cd $PBS_O_WORKDIR
mpirun --bynode -machinefile $PBS_NODEFILE ./a.out
```

2.3.4 Job submission

myscript.job : the script file name of a PBS batch job

```
$> qsub myscript.job
```

In response to this command you'll see a line with your job ID:

```
123456.tem-ce.sdfarm.kr
```

2.3.5 Monitoring and managing your jobs

Status of queued jobs

- qstat

Use the qstat command to check the status of your jobs. You can see whether your job is queued or running, along with information about requested resources. If the job is running you can see elapsed time and resources used.

```
### By itself, qstat lists all jobs in the system:
$> qstat

### To list all the jobs belonging to a particular user:
$> qstat -u tem_user

### To list the status of a particular job, in standard or alternate format:
$> qstat 123456
$> qstat -a 123456

### To get all the details about a particular job (full status):
$> qstat -f 123456
```

Managing your jobs

- Deleting (canceling) a job

Situations may arise in which you want to delete one of your jobs from the PBS queue. Perhaps you set the resource limits incorrectly, neglected to copy an input file, or had incorrect or missing commands in the batch file. Or maybe the program is taking too long to run (infinite loop). The PBS command to delete a batch job is qdel. It applies to both queued and running jobs.

```
$> qdel 123456
```

- Altering a queued job

You can alter certain attributes of your job while it's in the queue using the qalter command. This can be useful if you want to make a change without losing your place in the queue. You cannot make any alterations to the executable portion of the script, nor can you make any changes after the job starts running. The options argument consists of one or more PBS directives in the form of command-line options. For example, to change the walltime limit on job 123456 to 5 hours and have email sent when the job ends (only):

```
### The syntax is: qalter [options ...] jobid
$> qalter -l walltime=5:00:00 -m e 123456
```

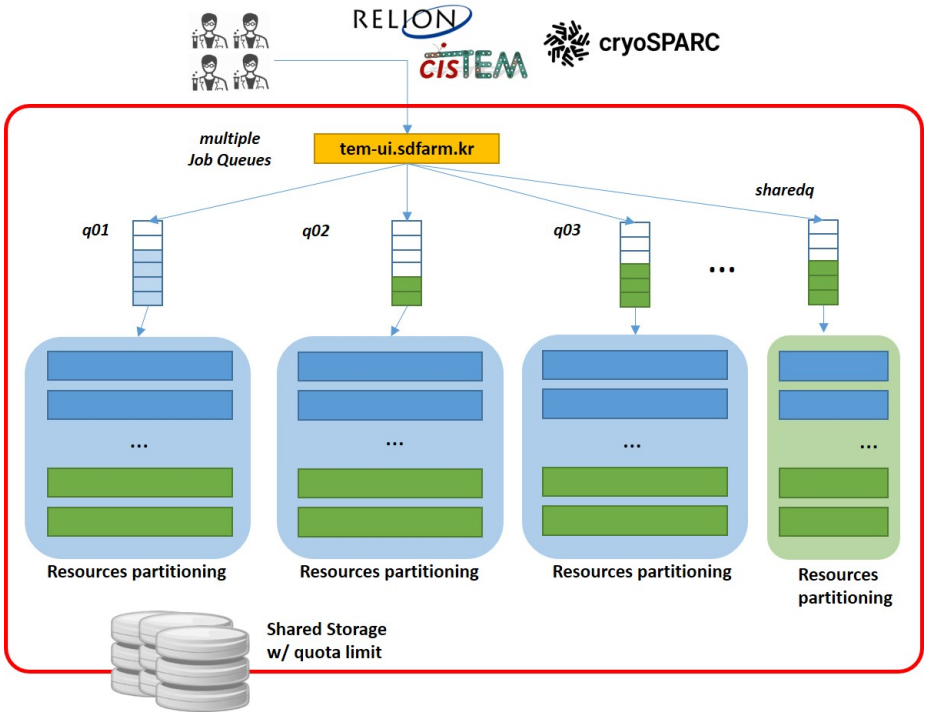
CHAPTER

THREE

BATCH QUEUES

3.1 Batch queue list

TEM farm provides multiple batch queues with different characteristics to users who submit jobs to analyze large-scale Cryo-EM data. A batch queue means a logical set of CPU and GPU computing resources. Users interact with a specific queue to manange their own jobs. Within each queue, submitted jobs are executed in order (First-in-first-out). Note that multiple jobs requiring CPU and/or GPU resources can be executed concurrently if there are enough available resources in the queue.



Category	Queue Name	Assigned Computing Resources	Remarks
Dedicated	**q01**	• tem-wn[1001-1003].sdfarm.kr (28 cores and 192GB memory per node) • tem-gpu01.sdfarm.kr (28 cores, 2 P100 GPGPUs and 384GB memory)	• USER/GROUP Access Control
	q02	• tem-wn[1004-1006].sdfarm.kr (28 cores and 192GB memory per node) • tem-gpu02.sdfarm.kr (28 cores, 2 P100 GPGPUs and 384 GB memory)	
	q03	• tem-wn[1007-1009].sdfarm.kr (28 cores and 192GB memory per node) • tem-gpu03.sdfarm.kr (28 cores, 2 P100 GPGPUs and 384 GB memory)	
Shared	**sharedq** (default)	• tem-wn[1010-1011].sdfarm.kr	• 112 logical CPU cores • hyperthreading(H/T) enabled
Experimental	**exp01**	• tem-gpu04.sdfarm.kr (48 cores, 128GB memory and 2 P40 GPGPUs)	• logical CPU cores, H/T enabled
	exp02	• tem-gpu05.sdfarm.kr (48 cores, 128GB memory and 2 P40 GPGPUs)	• logical CPU cores, H/T enabled

3.2 Time schedule for queue assignment

CHAPTER

FOUR

RELION

RELION (for REgularised LIkelihood OptimisatioN, pronounce rely-on) is a stand-alone computer program that employs an empirical Bayesian approach to refinement of (multiple) 3D reconstructions or 2D class averages in electron cryo-microscopy (cryo-EM). (from Relion official site https://www3.mrc-lmb.cam.ac.uk/relion/index.php?title=Main_Page)

4.1 Executing Relion GUI tools

4.1.1 How to start Relion data analysis tool

1. You can find out relion applications' environment module path by listing all the module available on TEM service farm

```
$> module avail

----------------- /tem/home/tem/Modules/Modules/versions -----------------
3.2.10

----------- /tem/home/tem/Modules/Modules/default/modulefiles ------------
apps/gcc/4.4.7/cistem/1.0.0     cuda/9.1
apps/gcc/4.4.7/relion/cpu/3.0.7 modules
apps/gcc/4.4.7/relion/gpu/3.0.7 mpi/gcc/openmpi/1.8.8
```

2. Check the module details for the specific relion version (e.g., Relion v3.0.7 with GPGPU support or Relion v3.0.7 with CPU cores support only)

```
$> module show apps/gcc/4.4.7/relion/gpu/3.0.7

-------------------------------------------------------------------
/tem/home/tem/Modules/Modules/default/modulefiles/apps/gcc/4.4.7/relion/gpu/3.0.7:

module-whatis    Setups `relion-3.0.7' environment variables
module           load mpi/gcc/openmpi/1.8.8
module           load cuda/9.1
setenv           relion_version 3.0.7
prepend-path     PATH /tem/home/tem/_Applications/relion-3.0.7/gpu/bin
prepend-path     LD_LIBRARY_PATH /tem/home/tem/_Applications/relion-3.0.7/gpu/lib
setenv           LANG en_US.UTF-8
setenv           RELION_QUEUE_NAME tem
setenv           RELION_QSUB_COMMAND qsub
setenv           RELION_QSUB_TEMPLATE /tem/home/tem/_Applications/relion-3.0.7/gpu/
↪bin/qsub-relion3-gpu.bash
```

(continues on next page)

(continued from previous page)

```
setenv           RELION_QSUB_EXTRA_COUNT 3
setenv           RELION_QSUB_EXTRA1 Number of Nodes
setenv           RELION_QSUB_EXTRA2 Number of processes per each node
setenv           RELION_QSUB_EXTRA3 Number of GPUs per node
setenv           RELION_QSUB_EXTRA1_DEFAULT 1
setenv           RELION_QSUB_EXTRA2_DEFAULT 3
setenv           RELION_QSUB_EXTRA3_DEFAULT 2
setenv           RELION_CTFFIND_EXECUTABLE /tem/home/tem/_Applications/ctffind-4.1.13/
↪bin/ctffind
setenv           RELION_GCTF_EXECUTABLE /tem/home/tem/_Applications/Gctf_v1.18_b2/bin/
↪Gctf_v1.18_b2_sm60_cu9.1
setenv           RELION_RESMAP_EXECUTABLE /tem/home/tem/_Applications/ResMap-1.1.4/
↪ResMap-1.1.4-linux64
setenv           RELION_MOTIONCOR2_EXECUTABLE /tem/home/tem/_Applications/MotionCor2/
↪MotionCor2_Cuda9.1_v1.0.5
setenv           RELION_UNBLUR_EXECUTABLE /tem/home/tem/_Applications/unblur_1.0.2/
↪bin/unblur_openmp_7_17_15.exe
setenv           RELION_SUMMOVIE_EXECUTABLE /tem/home/tem/_Applications/summovie_1.0.
↪2/bin/sum_movie_openmp_7_17_15.exe
conflict         apps/gcc/4.4.7/relion
-------------------------------------------------------------------

or

$> module show apps/gcc/4.4.7/relion/cpu/3.0.7

-------------------------------------------------------------------
/tem/home/tem/Modules/Modules/default/modulefiles/apps/gcc/4.4.7/relion/cpu/3.0.7:

module-whatis    Setups `relion-3.0.7' environment variables
module           load mpi/gcc/openmpi/1.8.8
setenv           relion_version 3.0.7
prepend-path     PATH /tem/home/tem/_Applications/relion-3.0.7/cpu/bin
prepend-path     LD_LIBRARY_PATH /tem/home/tem/_Applications/relion-3.0.7/cpu/lib
setenv           LANG en_US.UTF-8
setenv           RELION_QUEUE_USE yes
setenv           RELION_QUEUE_NAME own_queue_name
setenv           RELION_QSUB_COMMAND qsub
setenv           RELION_QSUB_TEMPLATE /tem/home/tem/_Applications/relion-3.0.7/cpu/
↪bin/qsub-relion3-cpu.bash
setenv           RELION_QSUB_EXTRA_COUNT 2
setenv           RELION_QSUB_EXTRA1 Number of Nodes
setenv           RELION_QSUB_EXTRA2 Number of processes per each node
setenv           RELION_QSUB_EXTRA1_DEFAULT 2
setenv           RELION_QSUB_EXTRA2_DEFAULT 16
setenv           RELION_CTFFIND_EXECUTABLE /tem/home/tem/_Applications/ctffind-4.1.13/
↪bin/ctffind
setenv           RELION_GCTF_EXECUTABLE /tem/home/tem/_Applications/Gctf_v1.18_b2/bin/
↪Gctf_v1.18_b2_sm60_cu9.1
setenv           RELION_RESMAP_EXECUTABLE /tem/home/tem/_Applications/ResMap-1.1.4/
↪ResMap-1.1.4-linux64
setenv           RELION_MOTIONCOR2_EXECUTABLE /tem/home/tem/_Applications/MotionCor2/
↪MotionCor2_Cuda9.1_v1.0.5
setenv           RELION_UNBLUR_EXECUTABLE /tem/home/tem/_Applications/unblur_1.0.2/
↪bin/unblur_openmp_7_17_15.exe
setenv           RELION_SUMMOVIE_EXECUTABLE /tem/home/tem/_Applications/summovie_1.0.
↪2/bin/sum_movie_openmp_7_17_15.exe
```

(continues on next page)

(continued from previous page)

```
conflict         apps/gcc/4.4.7/relion
------------------------------------------------------------------
```

3. Load the environment module for the version of relion application which you want to execute. As the module specified is loaded, all the modules with dependency are also loaded (you can check these modules with "module list" command)

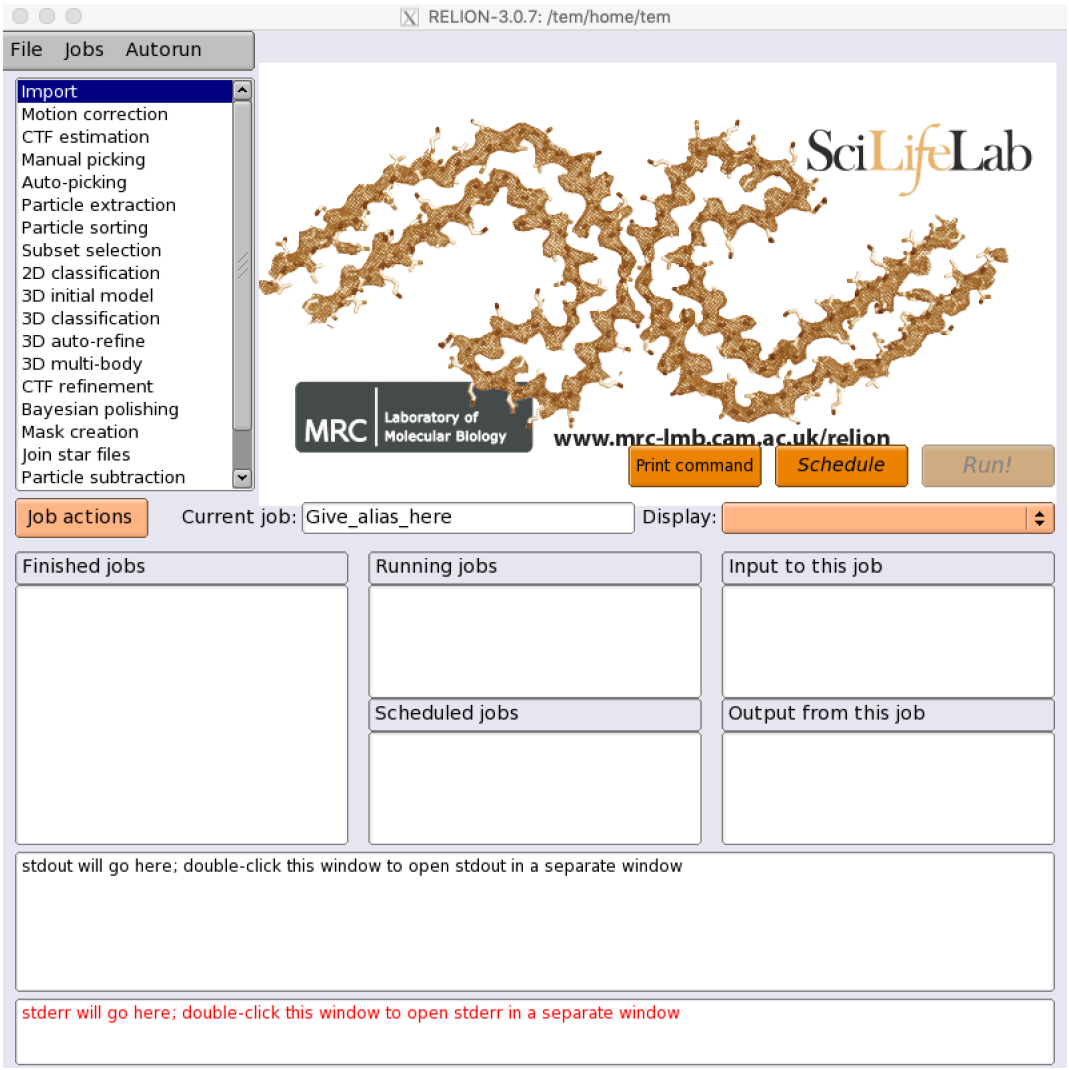
```
$> module load apps/gcc/4.4.7/relion/gpu/3.0.7
$> module list
Currently Loaded Modulefiles:
  1) mpi/gcc/openmpi/1.8.8             2) cuda/9.1                          3) apps/
↪gcc/4.4.7/relion/gpu/3.0.7
```

4. Check the relion application binary path

```
$> which relion
/tem/home/tem/_Applications/relion-3.0.7/gpu/bin/relion
```

5. Execute the relion application (we assume that X11 forwarding is enabled)

```
$> relion
```



4.2 PBS Strings used in Relion

Table 1: torque_strings_of_relion

String	Variable type	Description
XXXcommandXXX	string	relion command + arguments
XXXqueueXXX	string	Name of the queue to submit job to
XXXmpinodesXXX	integer	The number of MPI processes to use
XXXthreadsXXX	integer	The number of threads to use on each MPI process
XXXcoresXXX	integer	The number of MPI processes times the number of threads
XXXdedicatedXXX	integer	The minimum number of cores on each node (use this to fill entire nodes)
XXXnodesXXX	integer	The total number of nodes to be requested
XXXextra1XXX	string	Installation-specific
XXXextra2XXX	string	Installation-specific

Relion, by default, does not use the XXXextra1XXX, XXXextra2XXX, … variables. They provide additional flexibility for queueing systems (like Torque) that require additional variables. They may be activated by first

setting RELION_QSUB_EXTRA_COUNT to the number of fields you need (e.g. 3) and then setting the RELION_QSUB_EXTRA1, RELION_QSUB_EXTRA2, RELION_QSUB_EXTRA3 ... environment variables, respectively. This will result in extra input fields in the GUI, with the label text being equal to the value of the environment variable. Likewise, their default values (upon starting the GUI) can be set through environment variables RELION_QSUB_EXTRA1_DEFAULT, RELION_QSUB_EXTRA2_DEFAULT, etc and their help messages can be set through environmental variables RELION_QSUB_EXTRA1_HELP, RELION_QSUB_EXTRA2_HELP and so on.

4.3 Using CPU cluster (apps/gcc/4.4.7/relion/cpu/3.0.7)

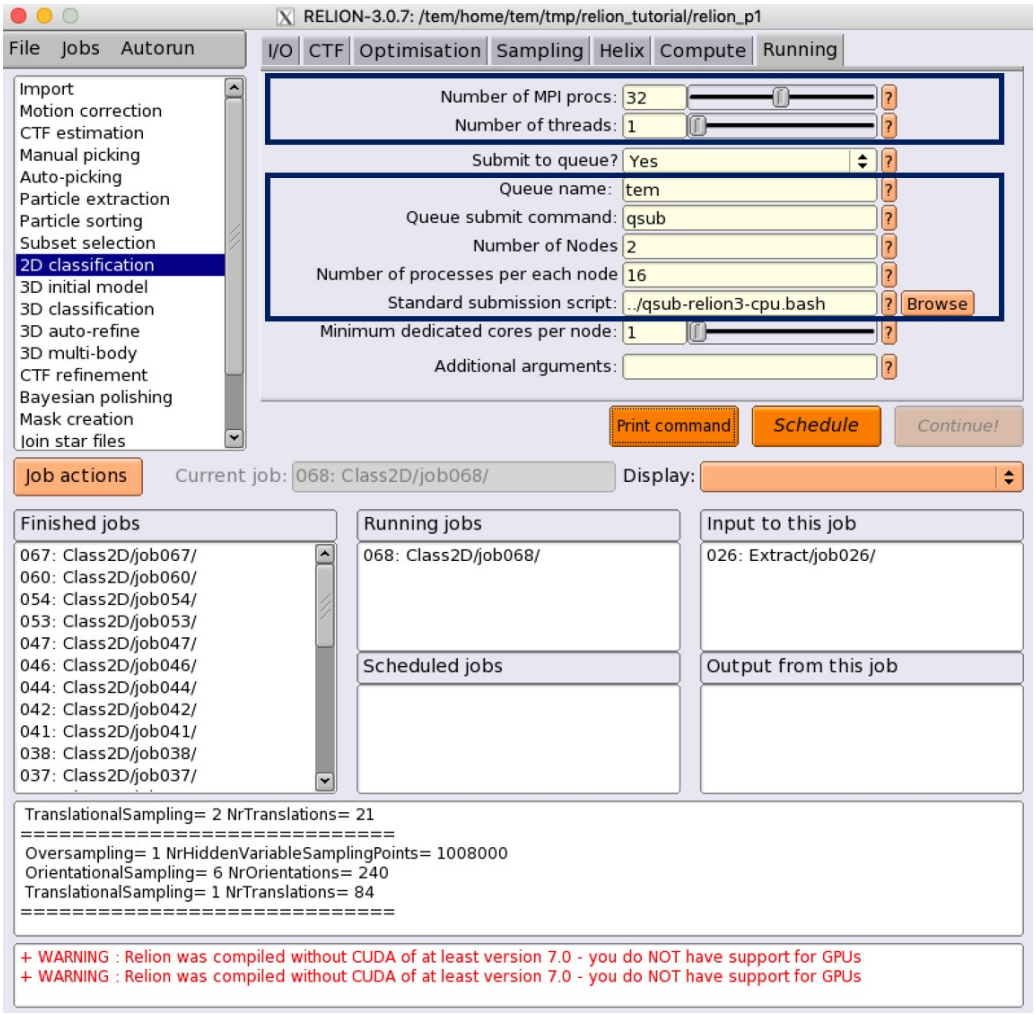
4.3.1 RELION_QSUB_TEMPLATE variable

Relion defines a lot of environment variables that can be used to execute different types of subtasks in the analysis workflows. Among these, "RELION_QSUB_TEMPLATE" describes the location of a proper batch job script template to submit jobs to the farm.

```
(for relion 3.0.7) RELION_QSUB_TEMPLATE /tem/home/tem/_Applications/relion-3.0.7/cpu/
↪bin/qsub-relion3-cpu.bash
```

For the use of CPU cluster nodes, we have set the RELION_QSUB_EXTRA_COUNT to 2. Two extra options describe "Number of Nodes" and "Number of processes per each node", respectively. These values can be referred by XXXextra1, XXXextra2XXX in the following batch job script template.

```
setenv RELION_QSUB_EXTRA_COUNT 2
setenv RELION_QSUB_EXTRA1 "Number of Nodes"
setenv RELION_QSUB_EXTRA2 "Number of processes per each node"
setenv RELION_QSUB_EXTRA1_DEFAULT 2
setenv RELION_QSUB_EXTRA2_DEFAULT 16
```



As shown in above figure, you can browse and select **"standard submission script"** as the location of RELION_QSUB_TEMPLATE for relion 3.0.7 (i.e., /tem/home/tem/_Applications/relion-3.0.7/cpu/bin/qsub-relion3-cpu.bash or its own your copy), and give **"Number of Nodes"** and **"Number of processes per each node"** values instead of default ones to submit a job to Torque based TEM farm. **(NOTE : you MUST use your OWN QUEUEe for "Queue name" and correct "number of MPI procs" which is generally total number of processes (number of nodes x number of processes per each node))**

4.3.2 Job script template (for CPU use)

```
#!/bin/bash

### Inherit all current environment variables
#PBS -V

### Job name
#PBS -N XXXnameXXX

### Queue name
#PBS -q XXXqueueXXX

### CPU cluster use : Specify the number of nodes (XXXextra1XXX) and the number of␣
↪processes per each node (XXXextra2XXX)
```

(continues on next page)

(continued from previous page)

```
#PBS -l nodes=XXXextra1XXX:ppn=XXXextra2XXX:XXXqueueXXX

#PBS -o ${PBS_JOBNAME}/run.out
#PBS -e ${PBS_JOBNAME}/run.err

############################################################
### Print Environment Variables
############################################################
echo ------------------------------------------------------
echo -n 'Job is running on node '; cat $PBS_NODEFILE
echo ------------------------------------------------------
echo PBS: qsub is running on $PBS_O_HOST
echo PBS: originating queue is $PBS_O_QUEUE
echo PBS: executing queue is $PBS_QUEUE
echo PBS: working directory is $PBS_O_WORKDIR
echo PBS: execution mode is $PBS_ENVIRONMENT
echo PBS: job identifier is $PBS_JOBID
echo PBS: job name is $PBS_JOBNAME
echo PBS: node file is $PBS_NODEFILE
echo PBS: current home directory is $PBS_O_HOME
echo PBS: PATH = $PBS_O_PATH
echo ------------------------------------------------------

############################################################
# Switch to the working directory;
cd ${PBS_O_WORKDIR}/${PBS_JOBNAME}
touch run.out
touch run.err
cd $PBS_O_WORKDIR
############################################################

### Run:
module load apps/gcc/4.4.7/relion/cpu/3.0.7
mpirun --prefix /tem/home/tem/_SystemLibs/openmpi-1.8.8 -machinefile $PBS_NODEFILE␣
↪XXXcommandXXX

echo "Done!"
```

4.4 Using GPGPU cluster (apps/gcc/4.4.7/relion/gpu/3.0.7)

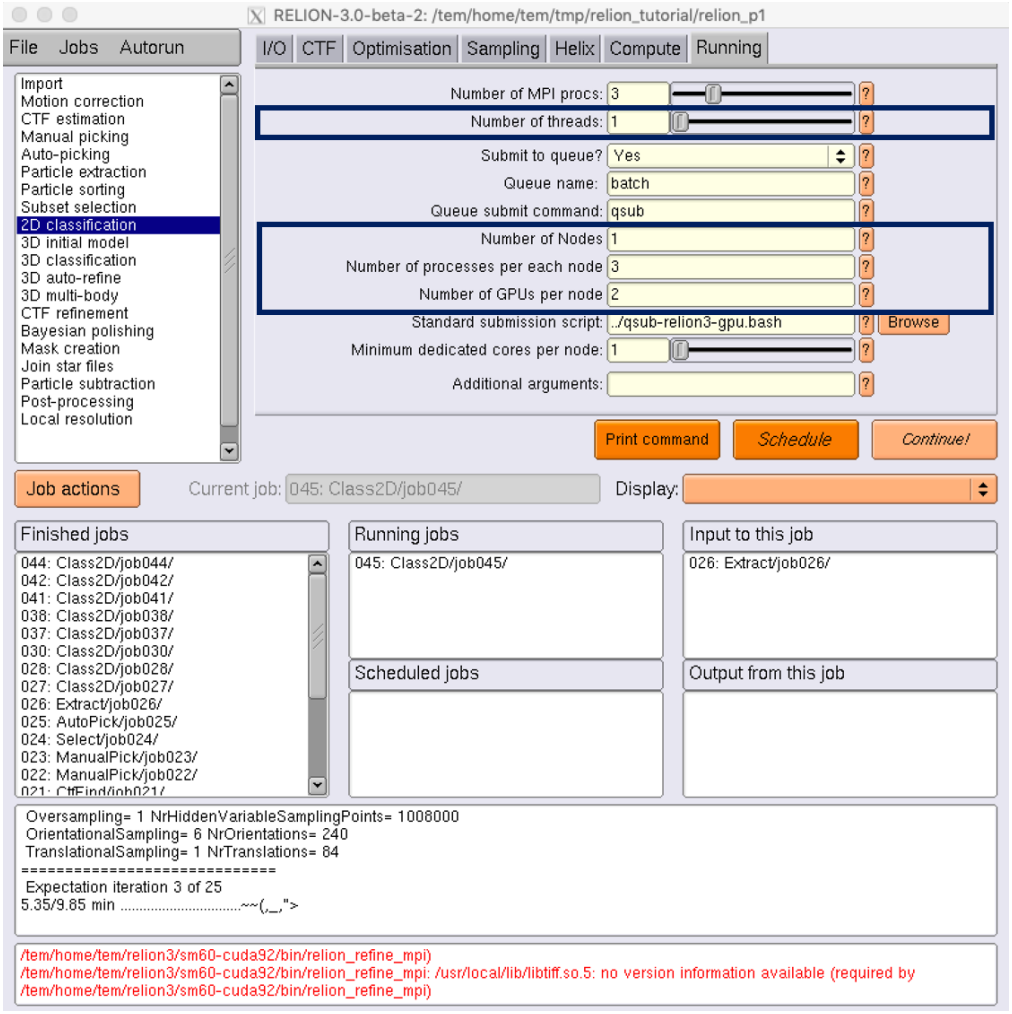
4.4.1 RELION_QSUB_TEMPLATE variable

Relion defines a lot of environment variables that can be used to execute different types of subtasks in the analysis workflows. Among these, "RELION_QSUB_TEMPLATE" describes the location of a proper batch job script to submit jobs to the farm.

```
(for relion 3.0.7 w/ GPU support) RELION_QSUB_TEMPLATE /tem/home/tem/_Applications/
↪relion-3.0.7/gpu/bin/qsub-relion3-gpu.bash
```

Unlike CPU cluster use case, we have set the RELION_QSUB_EXTRA_COUNT to 3 for the use of GPGPU cluster, where each extra option describes "Number of Nodes", "Number of processes per each node", and "Number of GPUs per node", respectively. All these values can be accessed by XXXextra1, XXXextra2XXX, XXXextra3XXX in the batch job script template.

```
setenv RELION_QSUB_EXTRA_COUNT 3
setenv RELION_QSUB_EXTRA1 "Number of Nodes"
setenv RELION_QSUB_EXTRA2 "Number of processes per each node"
setenv RELION_QSUB_EXTRA3 "Number of GPUs per node"
setenv RELION_QSUB_EXTRA1_DEFAULT 1
setenv RELION_QSUB_EXTRA2_DEFAULT 3
setenv RELION_QSUB_EXTRA3_DEFAULT 2
```



4.4.2 Job script template (for GPGPU use)

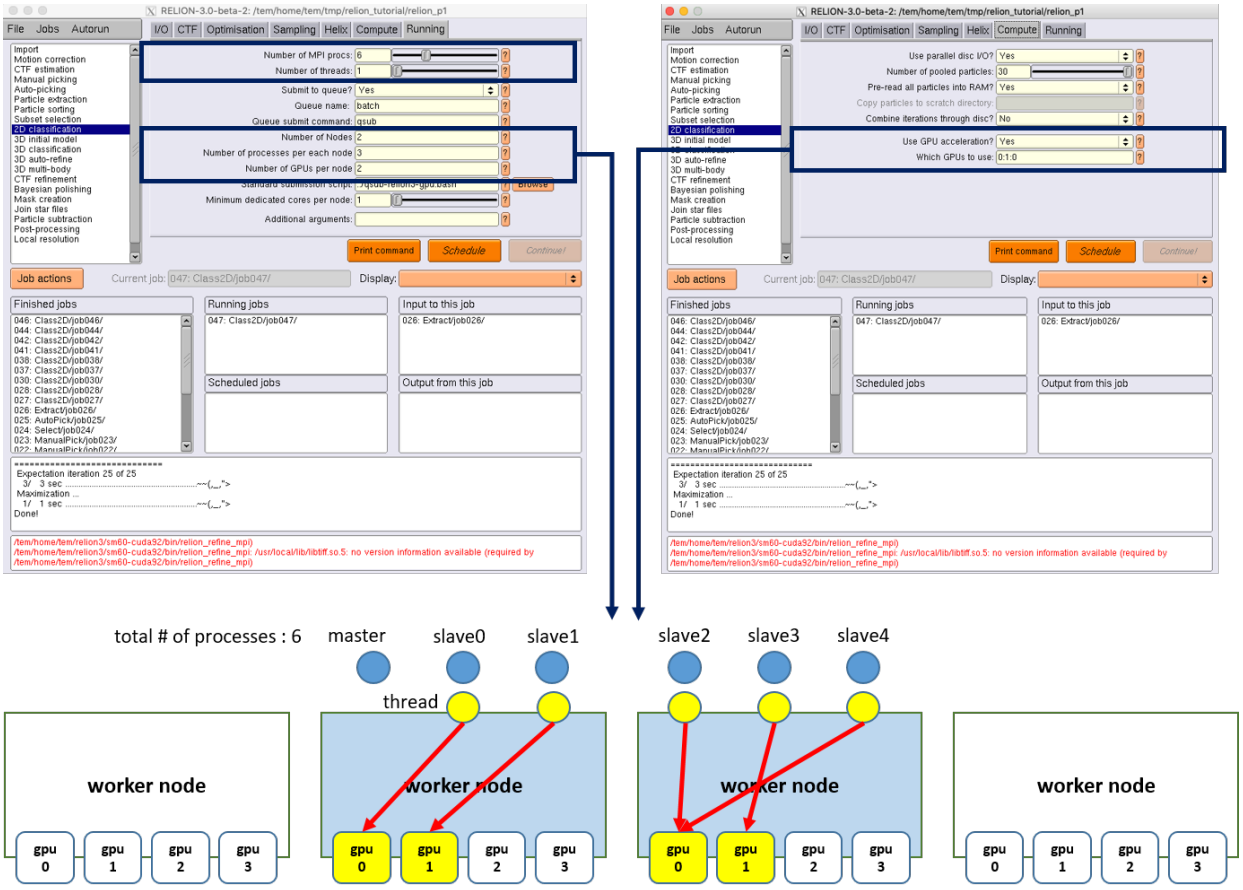
```
#!/bin/bash

### Inherit all current environment variables
#PBS -V

### Job name
#PBS -N XXXnameXXX

### Queue name
#PBS -q XXXqueueXXX
```

(continues on next page)

(continued from previous page)

```
### GPU use : Specify the number of nodes (XXXextra1XXX), the number of processes per
↪each node (XXXextra2XXX), and the number of GPGPUs per node (XXXextra3XXX)
#PBS -l nodes=XXXextra1XXX:ppn=XXXextra2XXX:gpus=XXXextra3XXX:XXXqueueXXX

#PBS -o ${PBS_JOBNAME}/run.out
#PBS -e ${PBS_JOBNAME}/run.err

############################################################
### Print Environment Variables
############################################################
echo ------------------------------------------------------
echo -n 'Job is running on node '; cat $PBS_NODEFILE
echo ------------------------------------------------------
echo PBS: qsub is running on $PBS_O_HOST
echo PBS: originating queue is $PBS_O_QUEUE
echo PBS: executing queue is $PBS_QUEUE
echo PBS: working directory is $PBS_O_WORKDIR
echo PBS: execution mode is $PBS_ENVIRONMENT
echo PBS: job identifier is $PBS_JOBID
echo PBS: job name is $PBS_JOBNAME
echo PBS: node file is $PBS_NODEFILE
echo PBS: current home directory is $PBS_O_HOME
echo PBS: PATH = $PBS_O_PATH
echo PBS: PBS_GPUFILE=$PBS_GPUFILE
echo PBS: CUDA_VISIBLE_DEVICES=$CUDA_VISIBLE_DEVICES
echo ------------------------------------------------------

############################################################
# Switch to the working directory;
cd ${PBS_O_WORKDIR}/${PBS_JOBNAME}
touch run.out
touch run.err
cd $PBS_O_WORKDIR
############################################################

### Run:
module load apps/gcc/4.4.7/relion/gpu/3.0.7
mpirun --prefix /tem/home/tem/openmpi-1.8.8 -machinefile $PBS_NODEFILE XXXcommandXXX

echo "Done!"
```

4.4.3 Specifying which GPUs to use



Here, we describe more advanced syntax for restricting RELION processes to certain GPUs on multi-GPU setups. You can use an argument to the –gpu option to provide a list of device-indices. The syntax is then to delimit ranks with colons [:], and threads by commas [,]. Any GPU indices provided is taken to be a list which is repeated if shorter than the total number of GPUs. By extension, the following rules applies

If a GPU id is specified more than once for a single mpi-rank, that GPU will be assigned proprotionally more of the threads of that rank. If no colons are used (i.e. GPUs are only specified for a single rank), then the GPUs specified, apply to all ranks. If GPUs are specified for more than one rank but not for all ranks, the unrestricted ranks are assigned the same GPUs as the restricted ranks, by a modulo rule. For example, if you would only want to use two of the four GPUs for all mpi-ranks, because you want to leave another two free for a different user/job, then (by the above rule 2) you can specify

```
mpirun -n 3 `which relion_refine_mpi` --gpu 2:3
slave 1 is told to use GPU2. slave 2 is told to use GPU3.
```

If you want an even spread over ALL GPUs, then you should not specify selection indices, as RELION will handle this itself. On your hypothetical 4-GPU machine, you would simply say

```
mpirun -n 3 `which relion_refine_mpi` --gpu
## slave 1 will use GPU0 and GPU1 for its threads. slave 2 will use GPU2 and GPU3 for␣
↪its threads
```

One can also schedule individual threads from MPI processes on the GPUs. This would be most useful when available RAM would be a limitation. Then one could for example run 3 MPI processes, each of which spawn a number of threads on two of the cards each, as follows:

```
mpirun -n 3 `which relion_refine_mpi` --j 4 --gpu 0,1,1,2:3
## slave 1 is told to put thread 1 on GPU0, threads 2 and 3 on GPU1, and thread 4 on␣
↪GPU2.  slave 2 is told to put all 4 threads on GPU3.
```

Finally, for completeness, the following is a more complex example to illustrate the full functionality of the GPU-device specification options.

```
mpirun -n 4 ... -j 3 --gpu 2:2:1,3
## slave 1 w/ 3 threads on GPU2, slave 2 w/ 3 threads on GPU2, slave 3 distributes 3␣
↪threads as evenly as possible across GPU1 and GPU3.
```

For more information, please refer to Relion Benchmarks and computer hardware (https://www3.mrc-lmb.cam.ac.uk/relion/index.php/Benchmarks_%26_computer_hardware)

4.5 Executing CPU/GPU jobs in Relion

Basically, with GPU-enabled Relion GUI, users can execute GPU-accelerated built-in subprograms, for examples :

- **refine, refine_mpi** (only the slaves, not the master)
- **autopick, autopick_mpi** (master and slaves)

and 3rd-party GPU-accelerated programs, for examples :

- **Gctf** (/tem/home/tem/_Applications/Gctf_v1.18_b2/bin/Gctf_v1.18_b2_sm60_cu9.1)
- **MotionCor2** (/tem/home/tem/_Applications/MotionCor2/MotionCor2_Cuda9.1_v1.0.5)

However, the GPU-enabled Relion also includes other built-in or 3rd-party CPU-only subprograms which follows :

- **MotionCor2-like alignment algorithm** (by Takanori Nakane)
- **CTFFind 4.1** (/tem/home/tem/_Applications/ctffind-4.1.13/bin/ctffind)
- **autopick** (built-in)
- **particle extraction** (built-in)
- **particle sorting** (built-in)
- **subset selection** (built-in)
- **2D classification** (built-in)
- **3D classification** (built-in)
- **3D refinement, 3D multi-body** (built-in)
- **CTF refinement** (built-in)
- **Particle substraction, etc.** (built-in)

So, for users convenience, we have deployed another Relion application with more generic computational resources requirements (GPU and/or CPU).

4.5.1 Module path

- apps/gcc/4.4.7/relion/gpu/3.0.7p

```
$> module avail

--------------------------- /tem/home/tem/Modules/Modules/versions --------------------
↪--------
3.2.10

---------------------- /tem/home/tem/Modules/Modules/default/modulefiles --------------
↪--------
apps/gcc/4.4.7/cistem/1.0.0      cuda/9.1
apps/gcc/4.4.7/relion/cpu/3.0.7  modules
apps/gcc/4.4.7/relion/gpu/3.0.7  mpi/gcc/openmpi/1.8.8
apps/gcc/4.4.7/relion/gpu/3.0.7p

$> module show apps/gcc/4.4.7/relion/gpu/3.0.7p

-------------------------------------------------------------------
/tem/home/tem/Modules/Modules/default/modulefiles/apps/gcc/4.4.7/relion/gpu/3.0.7p:

module-whatis    Setups `relion-3.0.7' environment variables
module           load mpi/gcc/openmpi/1.8.8
module           load cuda/9.1
setenv           relion_version 3.0.7
prepend-path     PATH /tem/home/tem/_Applications/relion-3.0.7/test/bin
prepend-path     LD_LIBRARY_PATH /tem/home/tem/_Applications/relion-3.0.7/test/lib
setenv           LANG en_US.UTF-8
setenv           RELION_QUEUE_USE yes
setenv           RELION_QUEUE_NAME own_queue_name
setenv           RELION_QSUB_COMMAND qsub
setenv           RELION_QSUB_TEMPLATE /tem/home/tem/_Applications/relion-3.0.7/test/
↪bin/qsub-relion3-gpu.bash
setenv           RELION_QSUB_EXTRA_COUNT 1
setenv           RELION_QSUB_EXTRA1 Resource Requirements
setenv           RELION_QSUB_EXTRA1_DEFAULT nodes=1:ppn=3:gpus=2
setenv           RELION_QSUB_EXTRA1_HELP For the use of GPUs, nodes=<#of nodes>:ppn=<#␣
↪of processes per n# of GPUs per node>. For the use of CPU cores, nodes=<#of nodes>
↪:ppn=<# of processes per node>
setenv           RELION_CTFFIND_EXECUTABLE /tem/home/tem/_Applications/ctffind-4.1.13/
↪bin/ctffind
setenv           RELION_GCTF_EXECUTABLE /tem/home/tem/_Applications/Gctf_v1.18_b2/bin/
↪Gctf_v1.18_b2_sm60_cu9.1
setenv           RELION_RESMAP_EXECUTABLE /tem/home/tem/_Applications/ResMap-1.1.4/
↪ResMap-1.1.4-linux64
setenv           RELION_MOTIONCOR2_EXECUTABLE /tem/home/tem/_Applications/MotionCor2/
↪MotionCor2_Cuda9.1_v1.0.5
setenv           RELION_UNBLUR_EXECUTABLE /tem/home/tem/_Applications/unblur_1.0.2/
↪bin/unblur_openmp_7_17_15.exe
setenv           RELION_SUMMOVIE_EXECUTABLE /tem/home/tem/_Applications/summovie_1.0.
↪2/bin/sum_movie_openmp_7_17_15.exe
conflict         apps/gcc/4.4.7/relion
-------------------------------------------------------------------

$> module load apps/gcc/4.4.7/relion/gpu/3.0.7p
$> module list
Currently Loaded Modulefiles:
  1) mpi/gcc/openmpi/1.8.8             2) cuda/9.1                          3) apps/
↪gcc/4.4.7/relion/gpu/3.0.7p
```

Here, we have set the RELION_QSUB_EXTRA_COUNT to 1 for the statement of more generic resource requirements

(nodes, ppn, gpus) which denote "Number of Nodes", "Number of processes per each node" and "Number of GPUs per node", respectively.

- **nodes** : number of nodes required (default:1)
- **ppn** : number of processes(cores) per node (default:3)
- **gpus** : number of GPU devices per node (default:2)

4.5.2 Job script template

- RELION_QSUB_TEMPLATE : /tem/home/tem/_Applications/relion-3.0.7/test/bin/qsub-relion3-gpu.bash

```
#!/bin/bash

### Inherit all current environment variables
#PBS -V

### Job name
#PBS -N XXXnameXXX

### Queue name
#PBS -q XXXqueueXXX

### Resource requirements : XXXextra1XXX
#PBS -l XXXextra1XXX:XXXqueueXXX

#PBS -o ${PBS_JOBNAME}/run.out
#PBS -e ${PBS_JOBNAME}/run.err

############################################################
### Print Environment Variables
############################################################
echo ------------------------------------------------------
echo -n 'Job is running on node '; cat $PBS_NODEFILE
echo ------------------------------------------------------
echo PBS: qsub is running on $PBS_O_HOST
echo PBS: originating queue is $PBS_O_QUEUE
echo PBS: executing queue is $PBS_QUEUE
echo PBS: working directory is $PBS_O_WORKDIR
echo PBS: execution mode is $PBS_ENVIRONMENT
echo PBS: job identifier is $PBS_JOBID
echo PBS: job name is $PBS_JOBNAME
echo PBS: node file is $PBS_NODEFILE
echo PBS: current home directory is $PBS_O_HOME
echo PBS: PATH = $PBS_O_PATH
echo PBS: PBS_GPUFILE=$PBS_GPUFILE
echo PBS: CUDA_VISIBLE_DEVICES=$CUDA_VISIBLE_DEVICES
echo ------------------------------------------------------

############################################################
# Switch to the working directory;
cd ${PBS_O_WORKDIR}/${PBS_JOBNAME}
touch run.out
touch run.err
cd $PBS_O_WORKDIR
############################################################
```

(continues on next page)

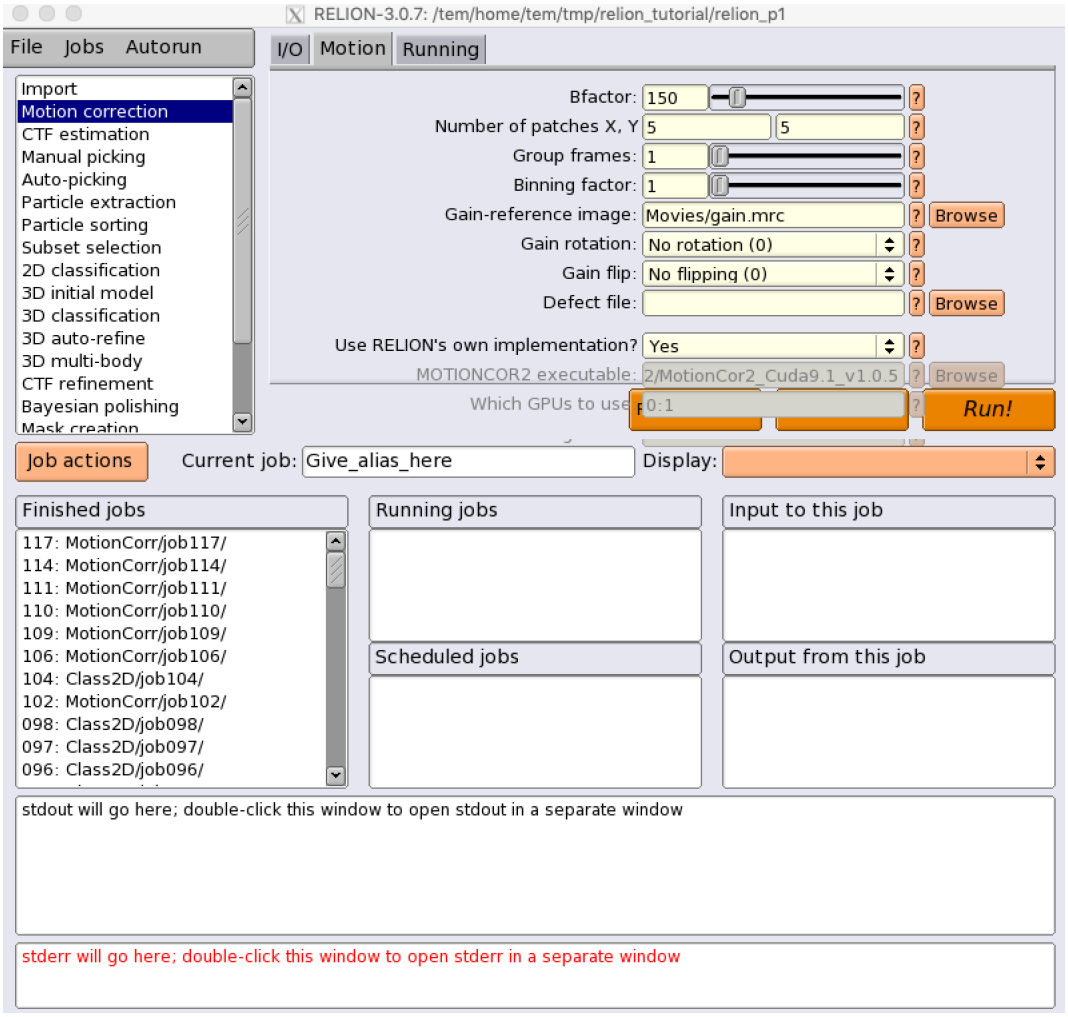
(continued from previous page)

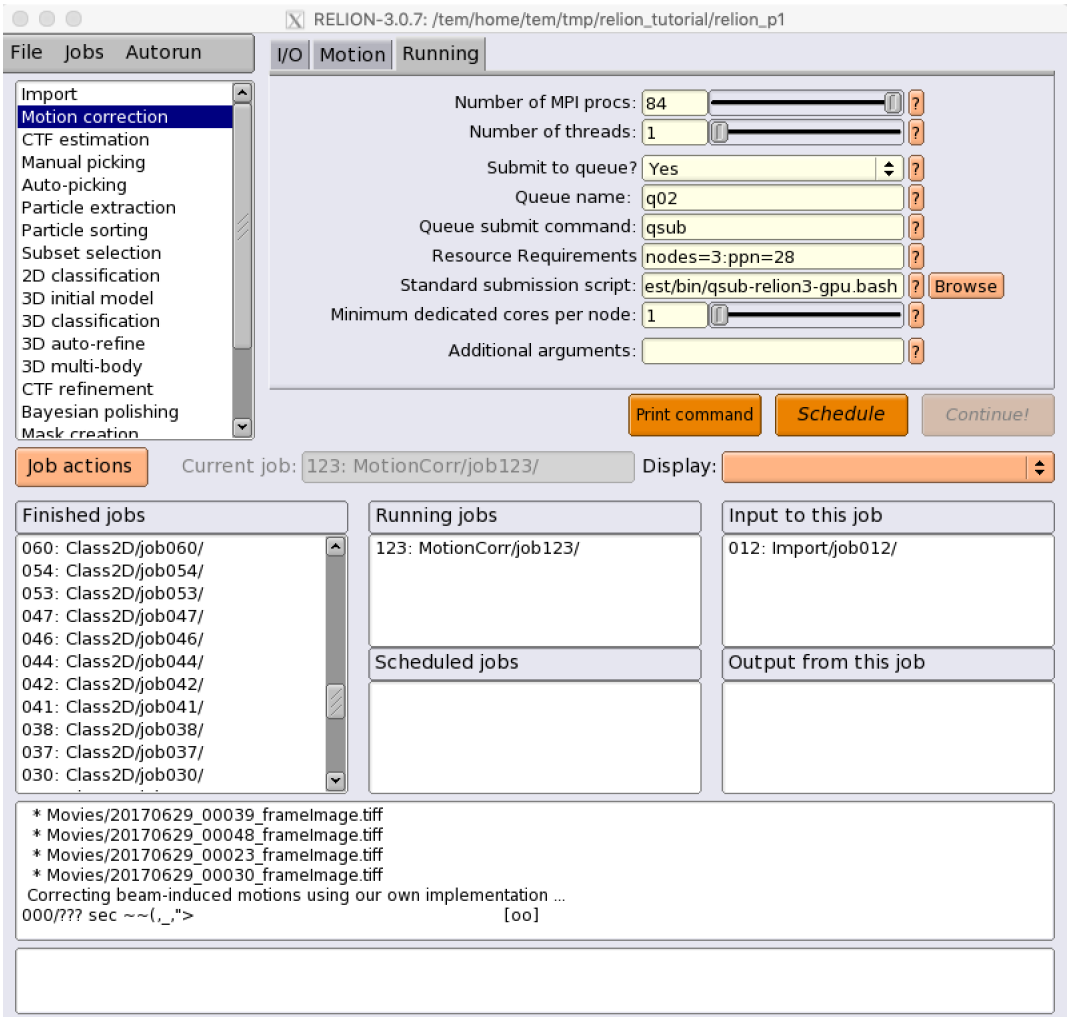
```
### Run:
module load apps/gcc/4.4.7/relion/gpu/3.0.7p
mpirun --prefix /tem/home/tem/openmpi-1.8.8 -machinefile $PBS_NODEFILE XXXcommandXXX

echo "Done!"
```

4.5.3 Examples

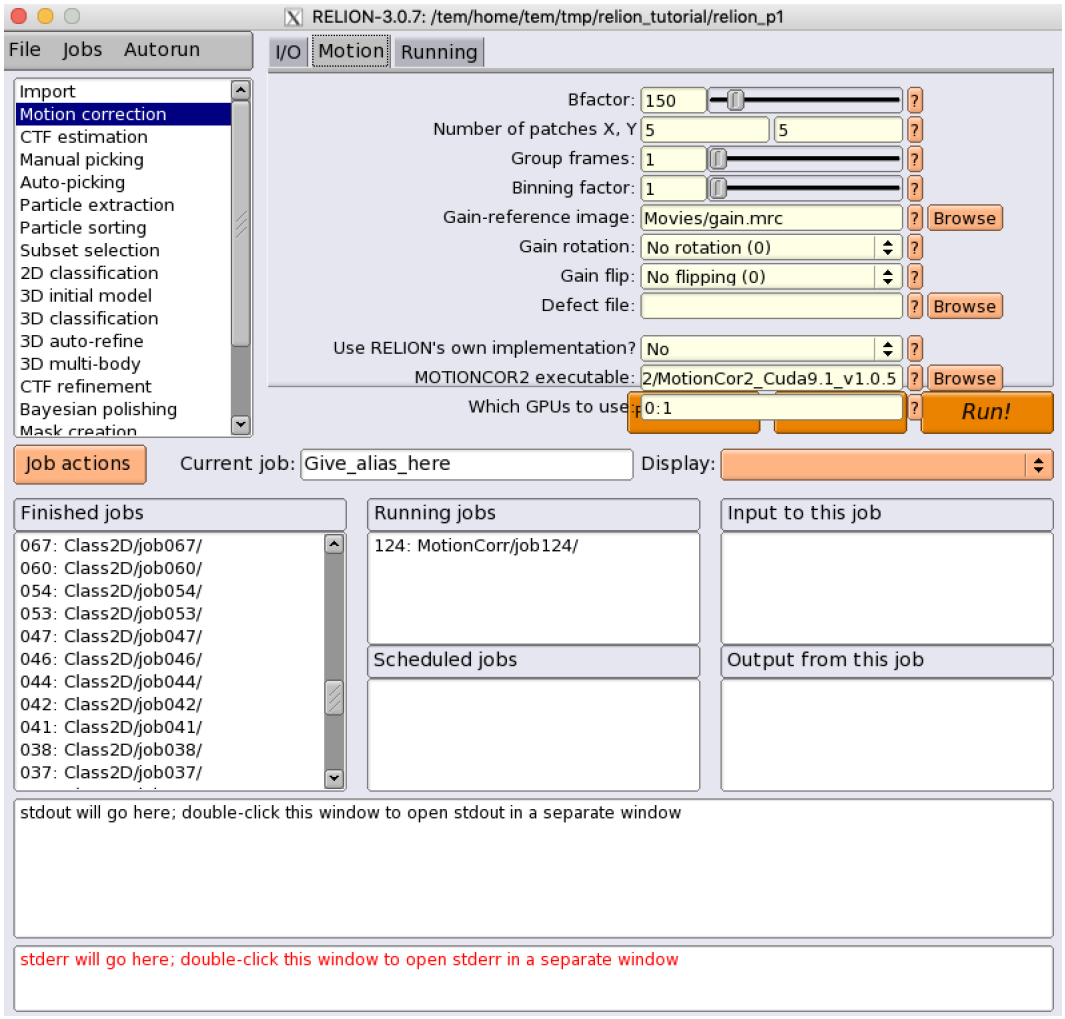
Motion Correction

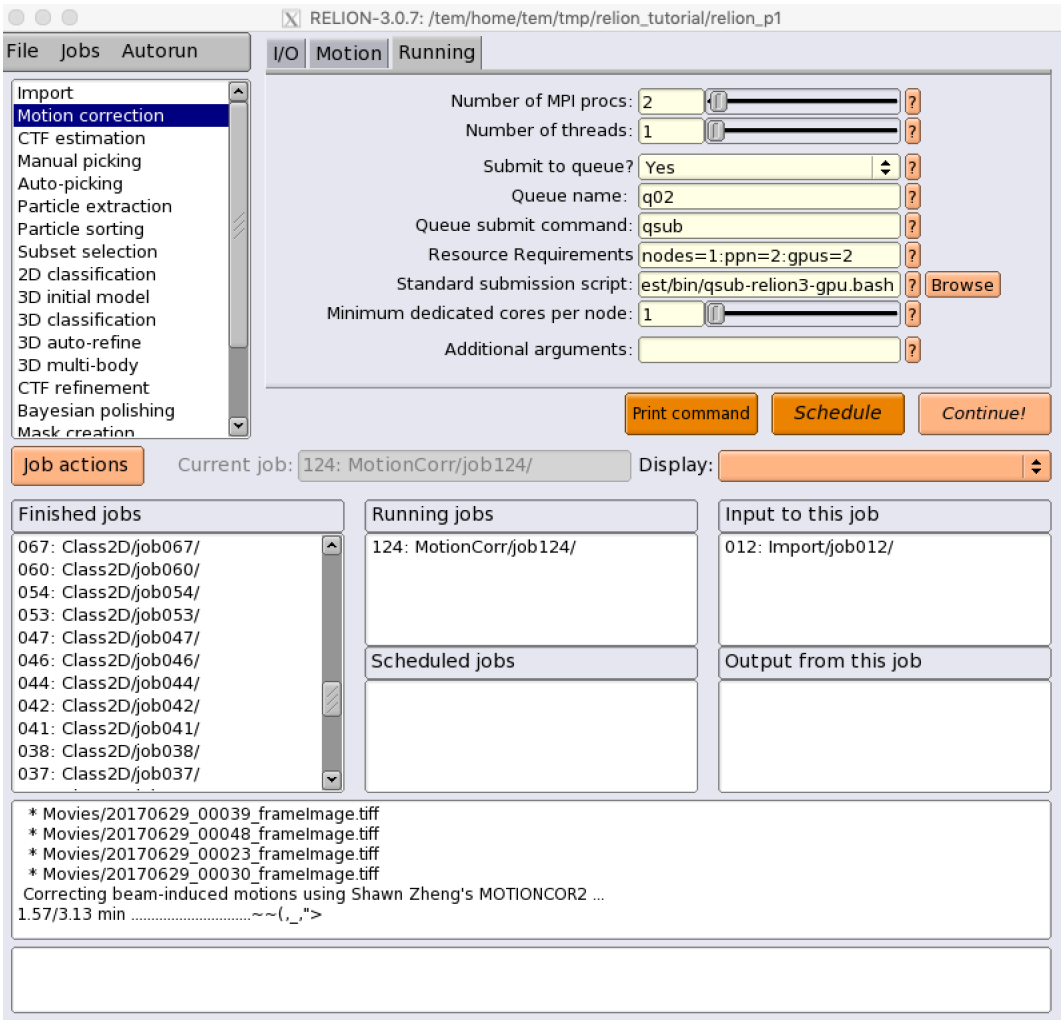
- **MotionCor2-like alignment algorithm** (CPU-only job, relion-own implementation)
  - (Motion) Use RELION's own implementation? : Yes
  - (Running) Number of MPI Procs : 84
  - (Running) Number of threads : 1
  - (Running) Queue name : <your own queue name> (e.g., q02)
  - (Running) Resource Requirements : nodes=3:ppn=28 (e.g., we assume the use of 3 nodes and all 28 cores each node)
  - (Running) Standard submission script : /tem/home/tem/_Applications/relion-3.0.7/test/bin/qsub-relion3-gpu.bash


RELION-3.0.7: /tem/home/tem/tmp/relion_tutorial/relion_p1
File
Jobs
Autorun
I/O
Motion
Running
Import
Motion correction
CTF estimation
Manual picking
Auto-picking
Particle extraction
Particle sorting
Subset selection
2D classification
3D initial model
3D classification
3D auto-refine
3D multi-body
CTF refinement
Bayesian polishing
Mask creation
Bfactor: 150
Number of patches X, Y: 5 5
Group frames: 1
Binning factor: 1
Gain-reference image: Movies/gain.mrc
Browse
Gain rotation: No rotation (0)
Gain flip: No flipping (0)
Defect file:
Browse
Use RELION's own implementation? Yes
MOTIONCOR2 executable: 2/MotionCor2_Cuda9.1_v1.0.5
Browse
Which GPUs to use: 0:1
Run!
Job actions
Current job: Give_alias_here
Display:
Finished jobs
117: MotionCorr/job117/
114: MotionCorr/job114/
111: MotionCorr/job111/
110: MotionCorr/job110/
109: MotionCorr/job109/
106: MotionCorr/job106/
104: Class2D/job104/
102: MotionCorr/job102/
098: Class2D/job098/
097: Class2D/job097/
096: Class2D/job096/
Running jobs
Input to this job
Scheduled jobs
Output from this job
stdout will go here; double-click this window to open stdout in a separate window
stderr will go here; double-click this window to open stderr in a separate window



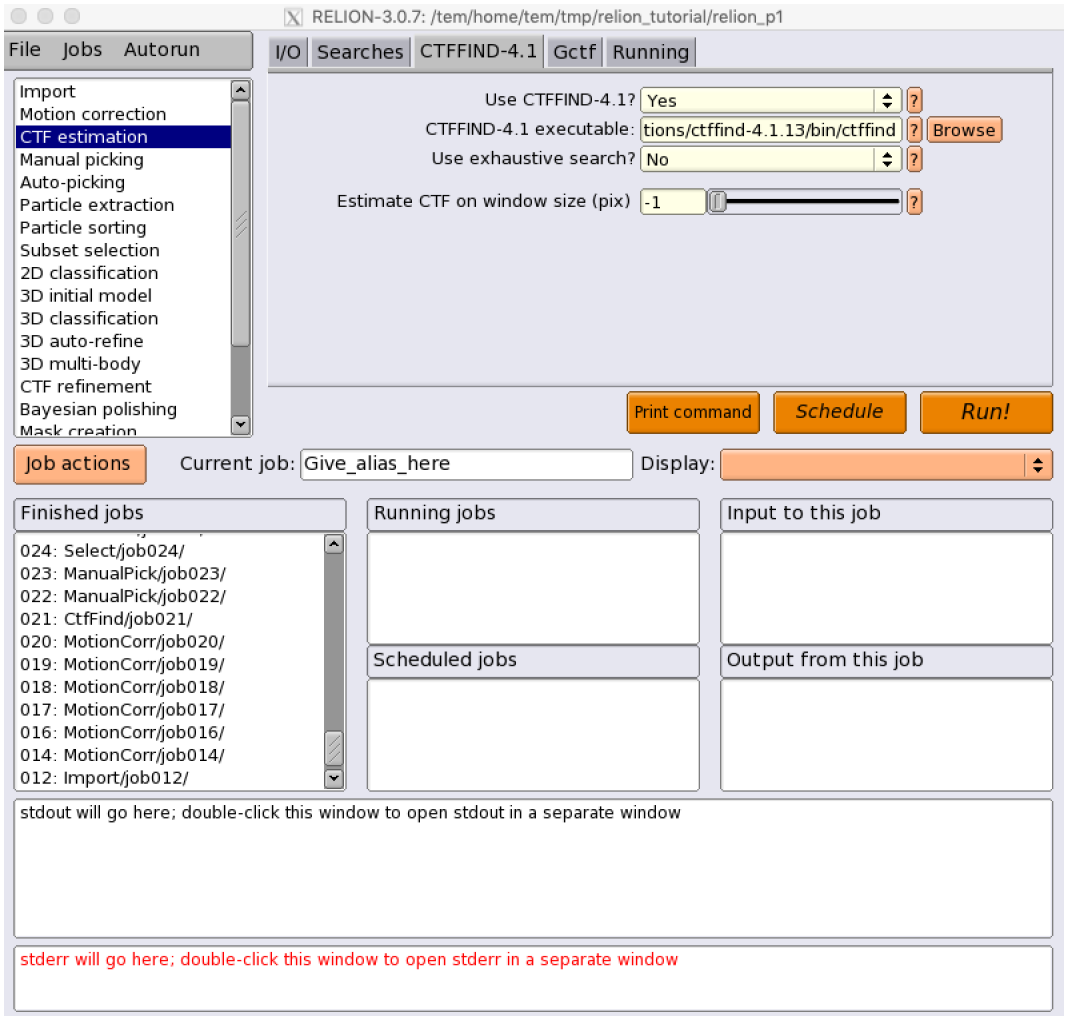
- **MotionCor2** (GPU-accelerated job)
  - (Motion) Use RELION's own implementation? : No
  - (Motion) MOTIONCOR2 executable : /tem/home/tem/_Applications/MotionCor2/MotionCor2_Cuda9.1_v1.0.5
  - (Running) Number of MPI Procs : 2
  - (Running) Number of threads : 1
  - (Running) Queue name : <your own queue name> (e.g., q02)
  - (Running) Resource Requirements : nodes=1:ppn=2:gpus=2 (e.g., we assume the use of 1 gpu node, 2 cpu cores and 2 GPU devices)
  - (Running) Standard submission script : /tem/home/tem/_Applications/relion-3.0.7/test/bin/qsub-relion3-gpu.bash

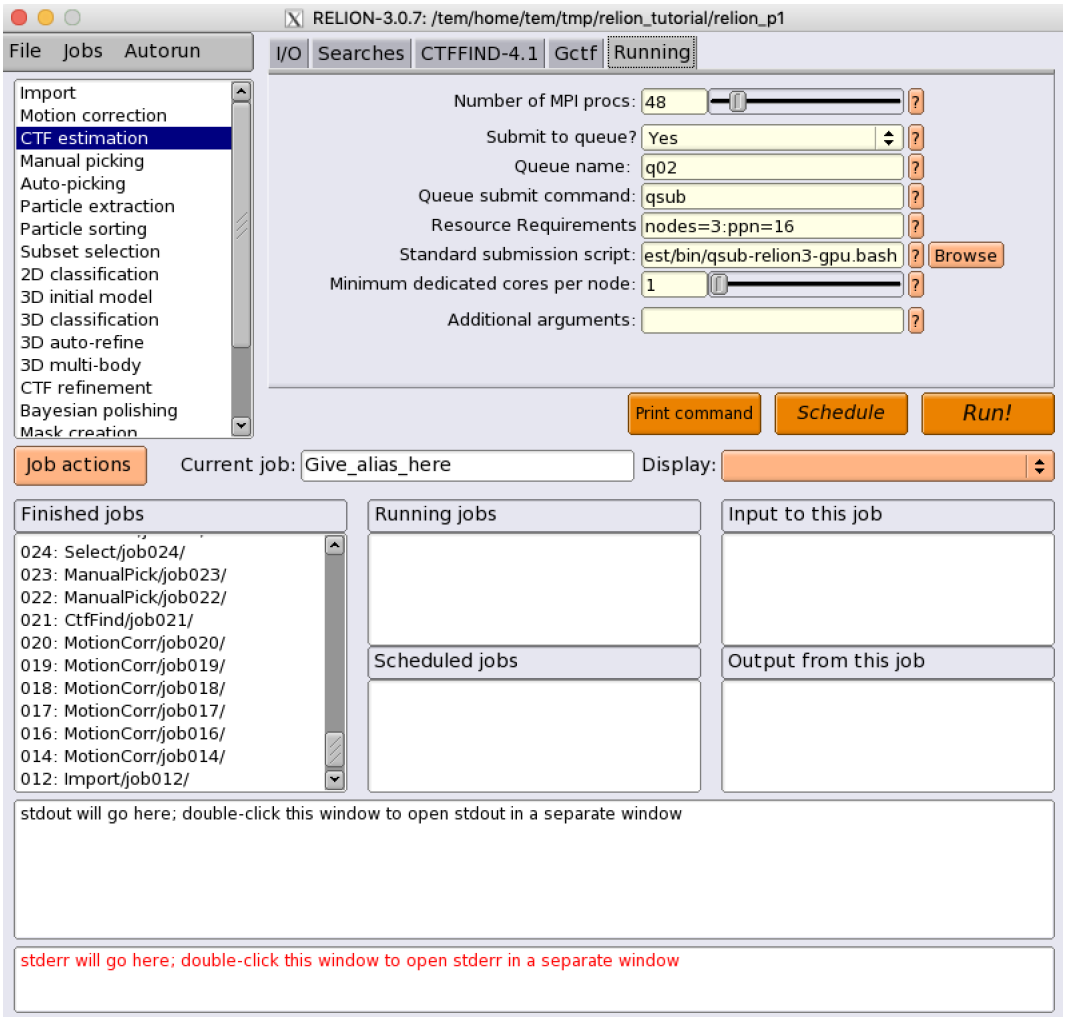

RELION-3.0.7: /tem/home/tem/tmp/relion_tutorial/relion_p1
File Jobs Autorun
I/O Motion Running
Import
Motion correction
CTF estimation
Manual picking
Auto-picking
Particle extraction
Particle sorting
Subset selection
2D classification
3D initial model
3D classification
3D auto-refine
3D multi-body
CTF refinement
Bayesian polishing
Mask creation
Bfactor: 150
Number of patches X, Y: 5 5
Group frames: 1
Binning factor: 1
Gain-reference image: Movies/gain.mrc Browse
Gain rotation: No rotation (0)
Gain flip: No flipping (0)
Defect file: Browse
Use RELION's own implementation? No
MOTIONCOR2 executable: 2/MotionCor2_Cuda9.1_v1.0.5 Browse
Which GPUs to use: 0:1
Run!
Job actions
Current job: Give_alias_here
Display:
Finished jobs
067: Class2D/job067/
060: Class2D/job060/
054: Class2D/job054/
053: Class2D/job053/
047: Class2D/job047/
046: Class2D/job046/
044: Class2D/job044/
042: Class2D/job042/
041: Class2D/job041/
038: Class2D/job038/
037: Class2D/job037/
Running jobs
124: MotionCorr/job124/
Scheduled jobs
Input to this job
Output from this job
stdout will go here; double-click this window to open stdout in a separate window
stderr will go here; double-click this window to open stderr in a separate window



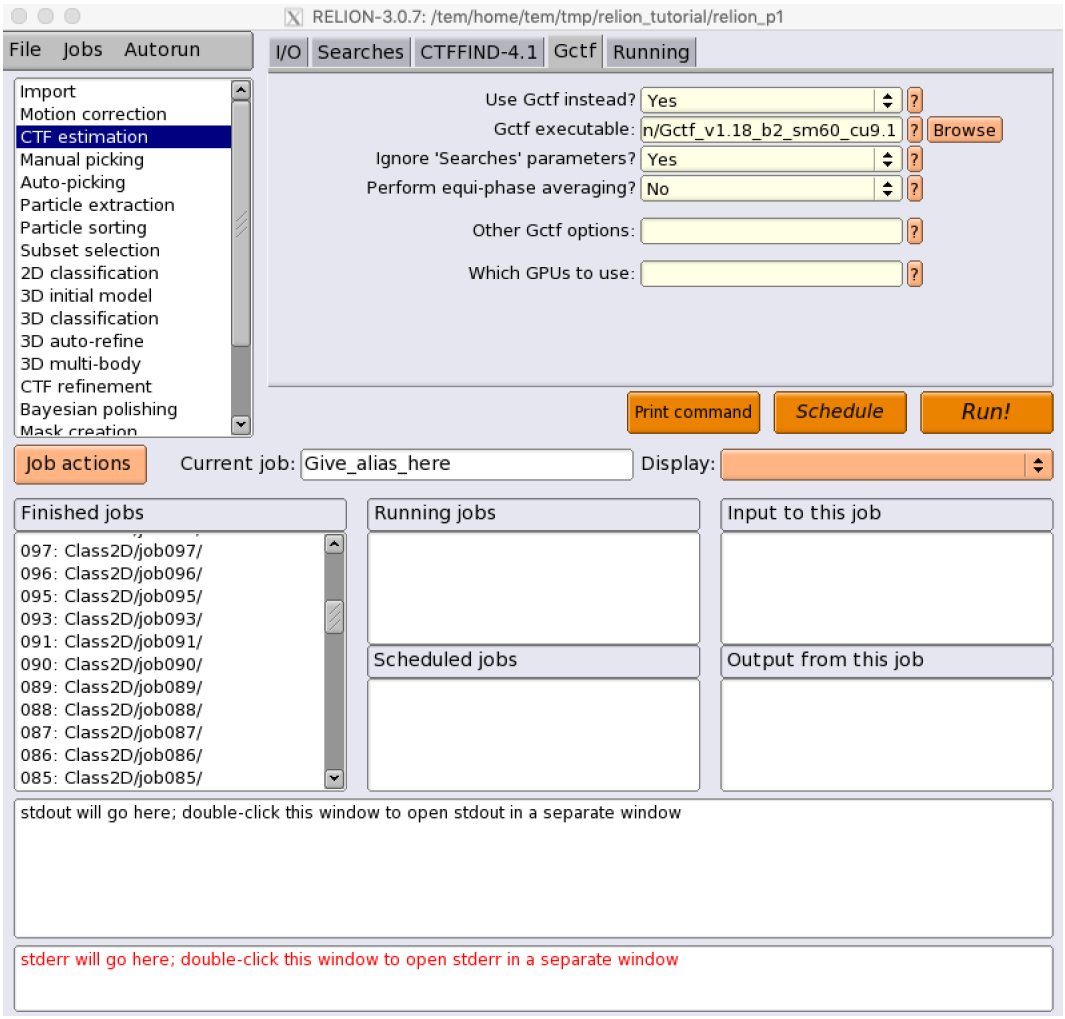
CTF Estimation

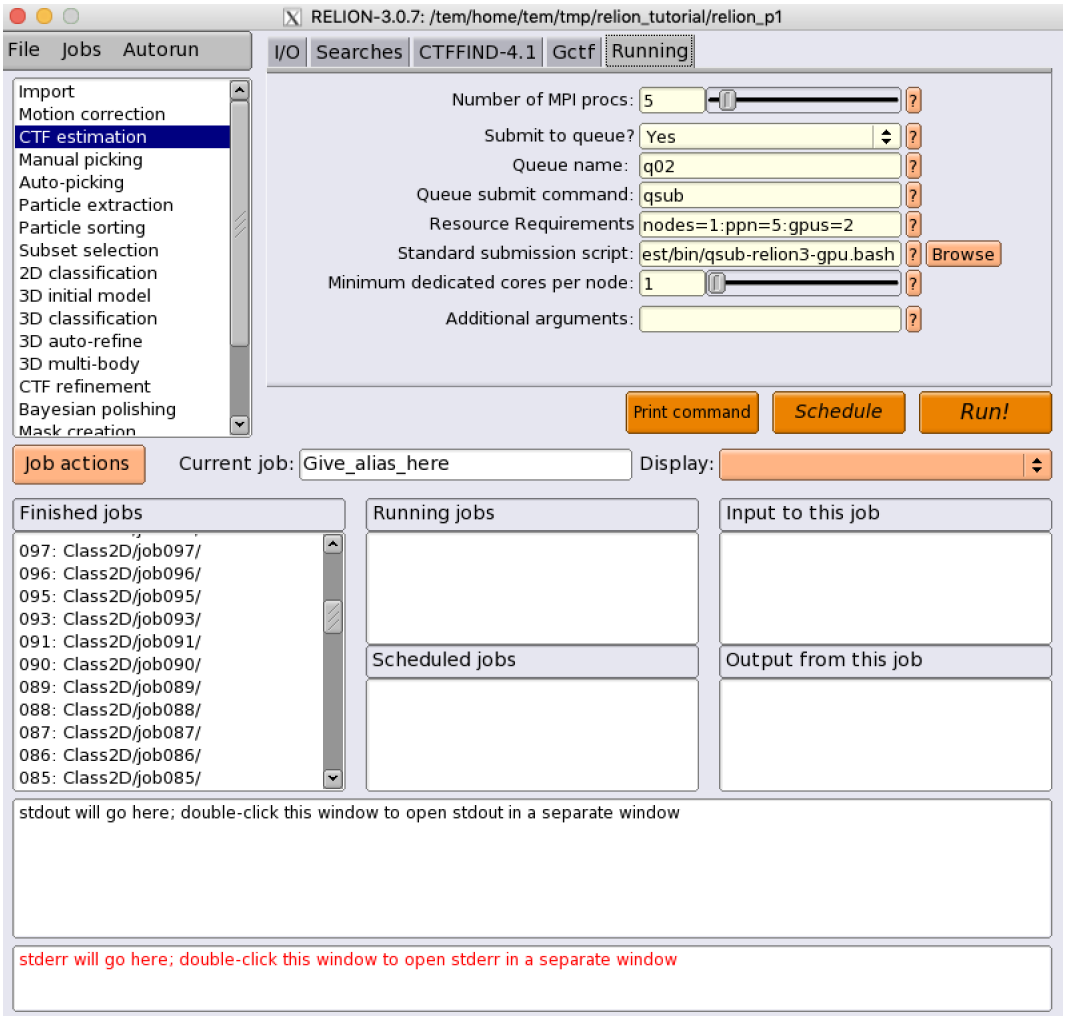
- **CTFFIND-4.1** (CPU-only job)
  - (CTFFIND-4.1) Use CTFFIND-4.1? : Yes
  - (CTFFIND-4.1) CTFFIND-4.1 executable? : /tem/home/tem/_Applications/ctffind-4.1.13/bin/ctffind
  - (Gctf) Use Gctf instead? : No
  - (Running) Number of MPI procs: 48
  - (Running) Submit to queue? : Yes
  - (Running) Queue name : <your own queue name> (e.g., q02)
  - (Running) Resource Requirements : nodes=3:ppn=16 (e.g., we assume the use of 3 nodes, 16 cpu cores per each node)
  - (Running) Standard submission script : /tem/home/tem/_Applications/relion-3.0.7/test/bin/qsub-relion3-gpu.bash


RELION-3.0.7: /tem/home/tem/tmp/relion_tutorial/relion_p1
File Jobs Autorun
I/O Searches CTFFIND-4.1 Gctf Running
Import
Motion correction
CTF estimation
Manual picking
Auto-picking
Particle extraction
Particle sorting
Subset selection
2D classification
3D initial model
3D classification
3D auto-refine
3D multi-body
CTF refinement
Bayesian polishing
Mask creation
Use CTFFIND-4.1? Yes
CTFFIND-4.1 executable: tions/ctffind-4.1.13/bin/ctffind
Browse
Use exhaustive search? No
Estimate CTF on window size (pix) -1
Print command
Schedule
Run!
Job actions
Current job: Give_alias_here
Display:
Finished jobs
024: Select/job024/
023: ManualPick/job023/
022: ManualPick/job022/
021: CtfFind/job021/
020: MotionCorr/job020/
019: MotionCorr/job019/
018: MotionCorr/job018/
017: MotionCorr/job017/
016: MotionCorr/job016/
014: MotionCorr/job014/
012: Import/job012/
Running jobs
Input to this job
Scheduled jobs
Output from this job
stdout will go here; double-click this window to open stdout in a separate window
stderr will go here; double-click this window to open stderr in a separate window



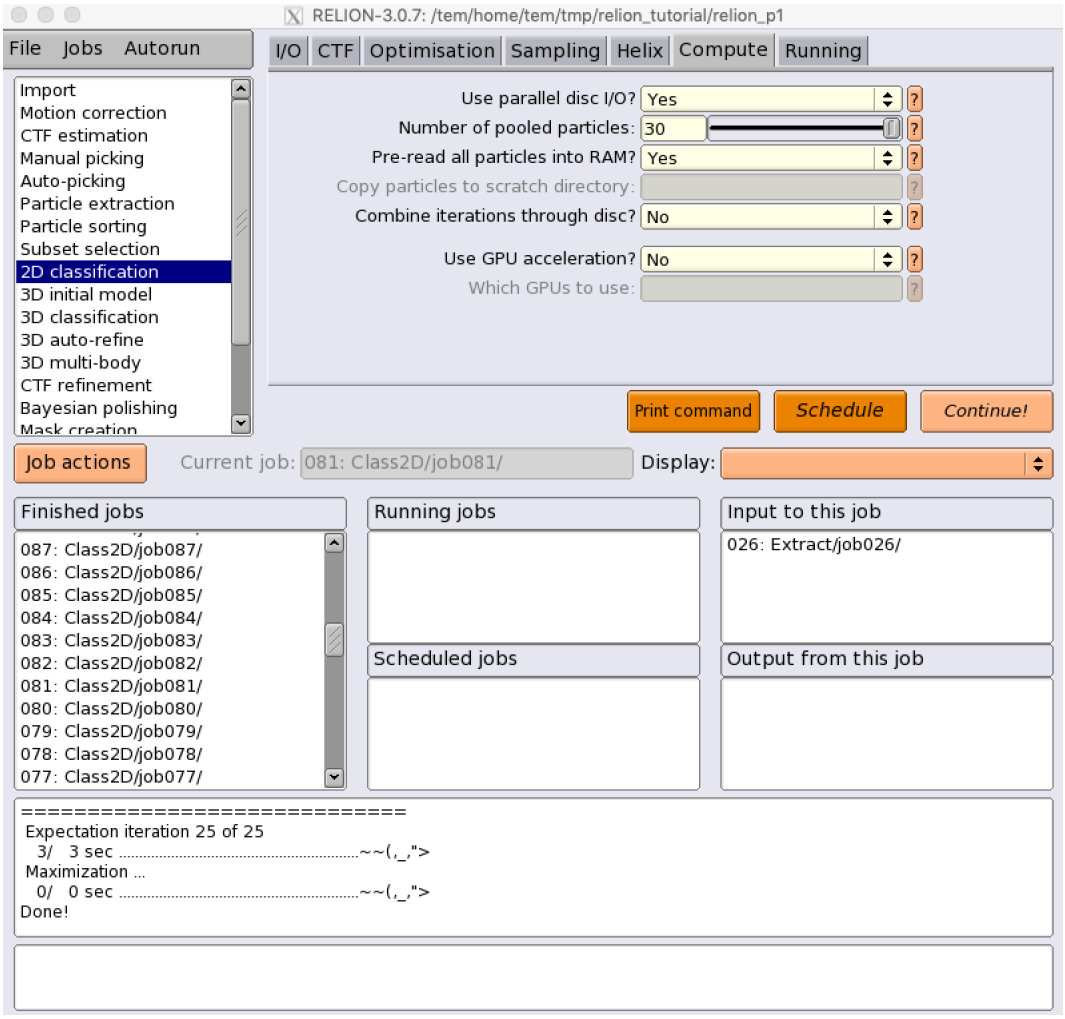
- **Gctf** (GPU-accelerated job)
  - (CTFFIND-4.1) Use CTFFIND-4.1? : No
  - (Gctf) Use Gctf instead? : Yes
  - (Gctf) Gctf executable: /tem/home/tem/_Applications/Gctf_v1.18_b2/bin/Gctf_v1.18_b2_sm60_cu9.1
  - (Gctf) Which GPUs to use: <empty> (i.e., relion automatically assigned available GPU devices to the MPI processes)
  - (Running) Number of MPI procs: 5 (1 master and 4 slave processes)
  - (Running) Submit to queue? : Yes
  - (Running) Queue name : <your own queue name> (e.g., q02)
  - (Running) Resource Requirements : nodes=1:ppn=5:gpus=2
  - (Running) Standard submission script : /tem/home/tem/_Applications/relion-3.0.7/test/bin/qsub-relion3-gpu.bash

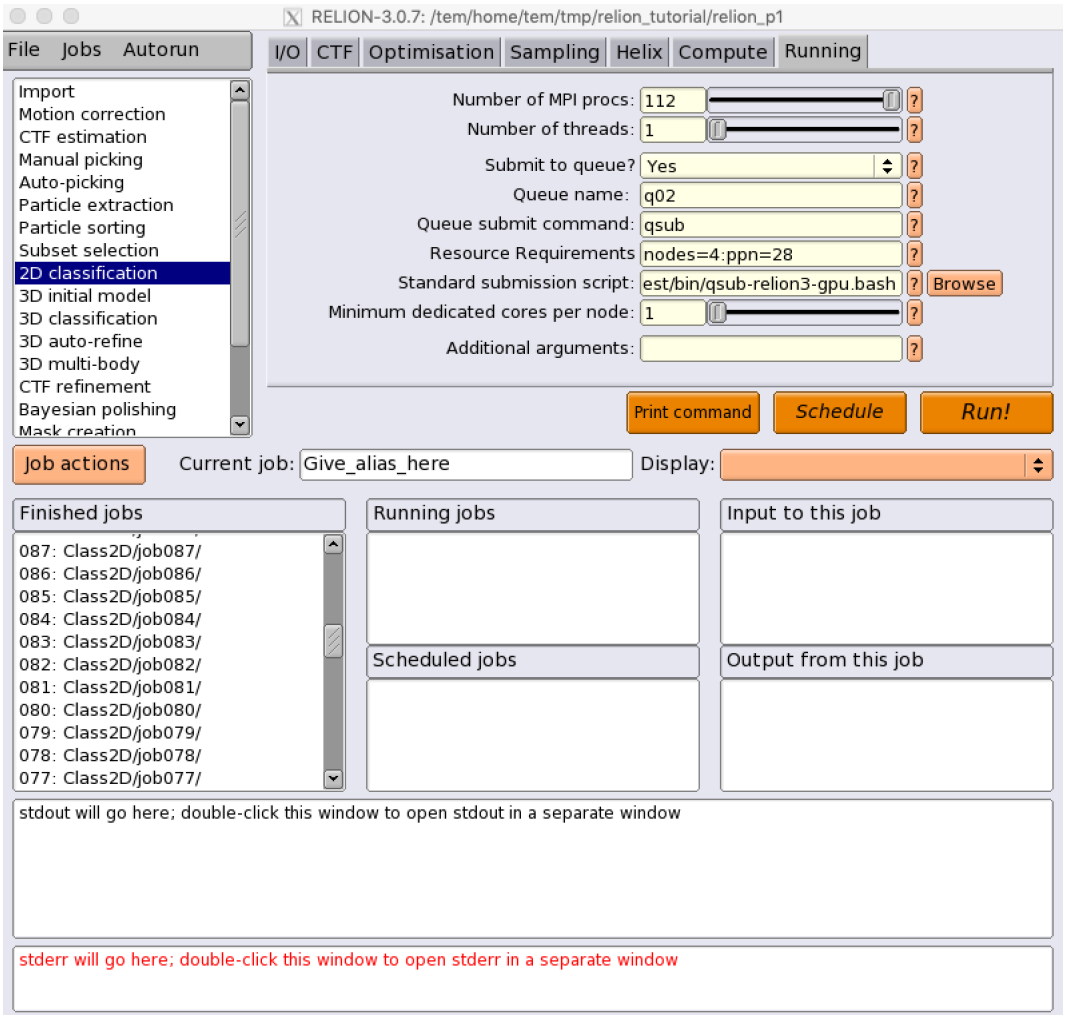

RELION-3.0.7: /tem/home/tem/tmp/relion_tutorial/relion_p1
File
Jobs
Autorun
I/O
Searches
CTFFIND-4.1
Gctf
Running
Import
Motion correction
CTF estimation
Manual picking
Auto-picking
Particle extraction
Particle sorting
Subset selection
2D classification
3D initial model
3D classification
3D auto-refine
3D multi-body
CTF refinement
Bayesian polishing
Mask creation
Use Gctf instead?
Yes
Gctf executable:
n/Gctf_v1.18_b2_sm60_cu9.1
Browse
Ignore 'Searches' parameters?
Yes
Perform equi-phase averaging?
No
Other Gctf options:
Which GPUs to use:
Print command
Schedule
Run!
Job actions
Current job:
Give_alias_here
Display:
Finished jobs
097: Class2D/job097/
096: Class2D/job096/
095: Class2D/job095/
093: Class2D/job093/
091: Class2D/job091/
090: Class2D/job090/
089: Class2D/job089/
088: Class2D/job088/
087: Class2D/job087/
086: Class2D/job086/
085: Class2D/job085/
Running jobs
Input to this job
Scheduled jobs
Output from this job
stdout will go here; double-click this window to open stdout in a separate window
stderr will go here; double-click this window to open stderr in a separate window



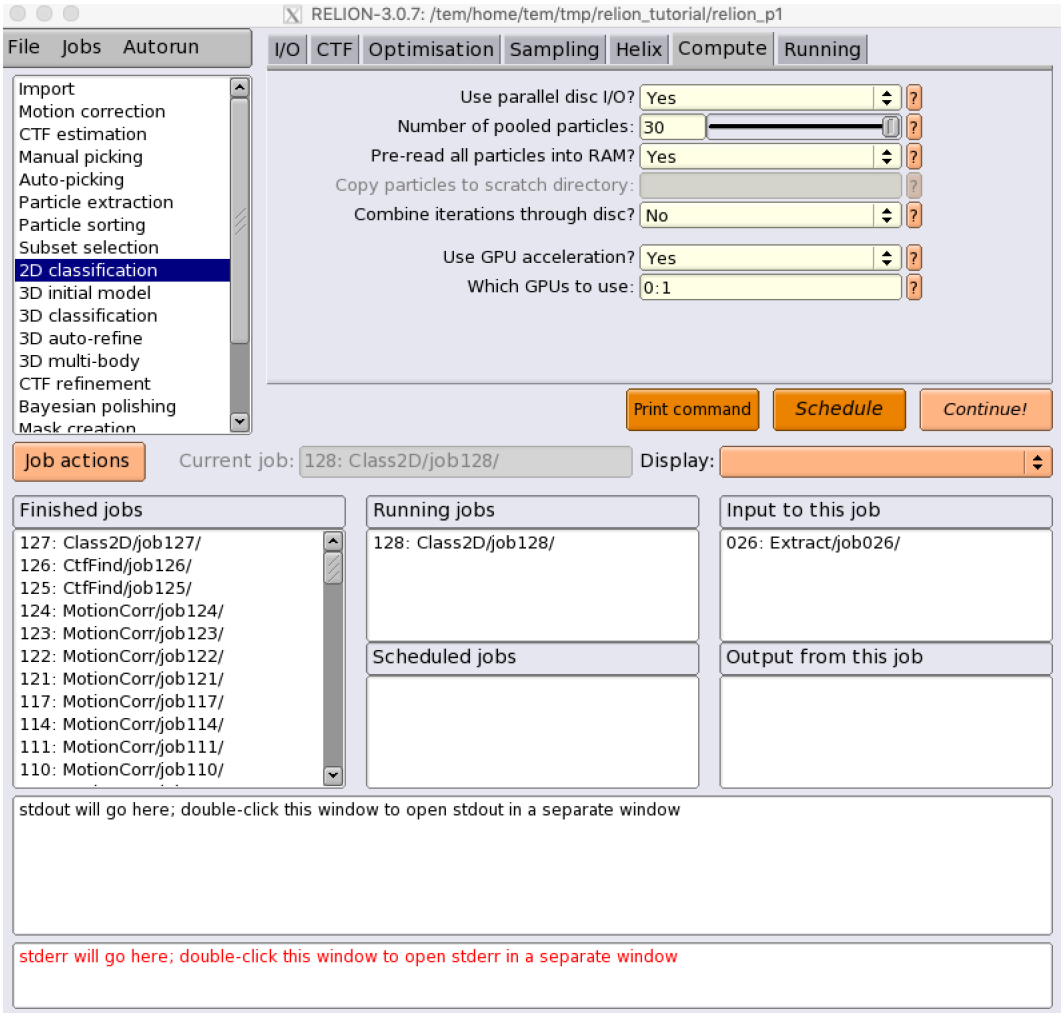
2D Classification

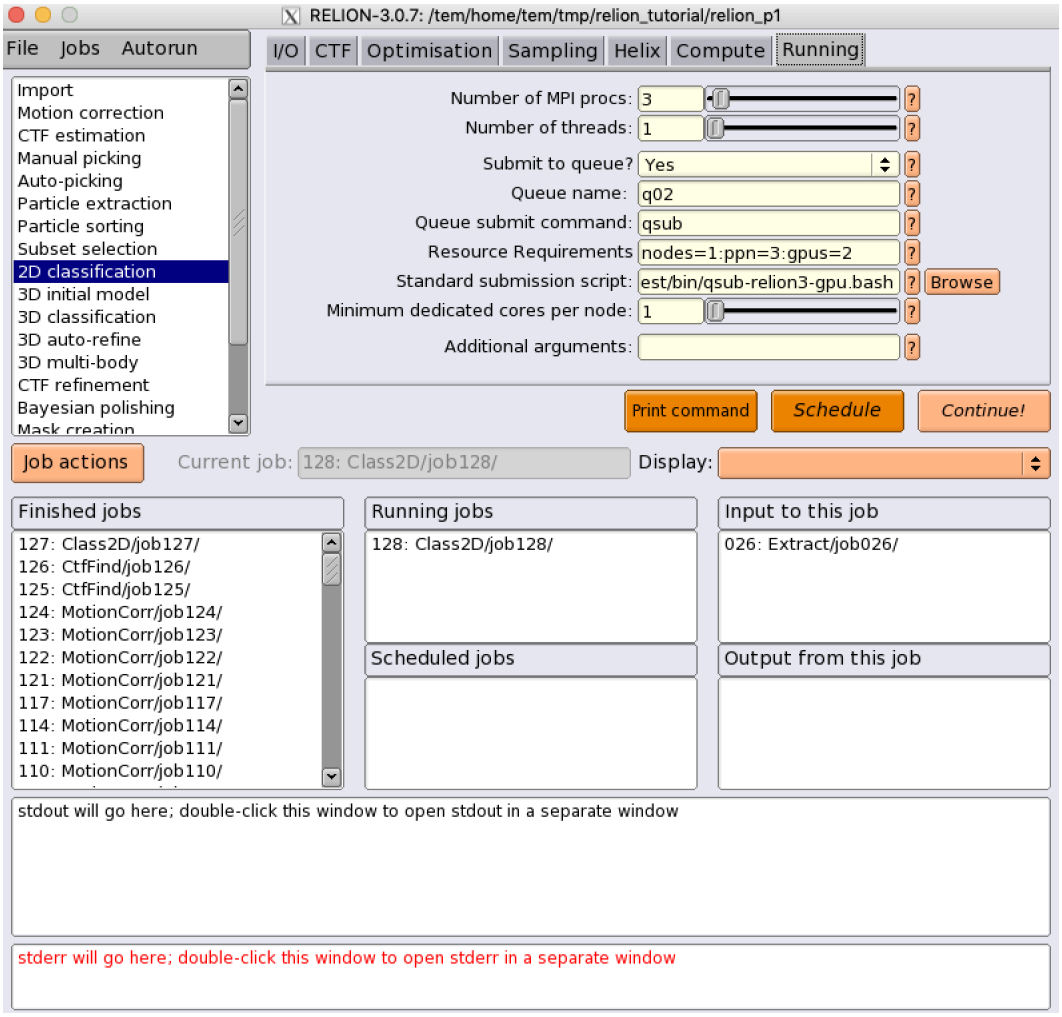
- **relion_refine_mpi** (CPU-only job)
  - (Compute) Use GPU acceleration? : No
  - (Running) Number of MPI procs: 112
  - (Running) Number of threads: 1
  - (Running) Submit to queue? : Yes
  - (Running) Queue name : <your own queue name> (e.g., q02)
  - (Running) Resource Requirements : nodes=4:ppn=28 (e.g., we assume the use of 4 nodes, 28 cpu cores per each node)
  - (Running) Standard submission script : /tem/home/tem/_Applications/relion-3.0.7/test/bin/qsub-relion3-gpu.bash


RELION-3.0.7: /tem/home/tem/tmp/relion_tutorial/relion_p1
File Jobs Autorun
I/O CTF Optimisation Sampling Helix Compute Running
Import
Motion correction
CTF estimation
Manual picking
Auto-picking
Particle extraction
Particle sorting
Subset selection
2D classification
3D initial model
3D classification
3D auto-refine
3D multi-body
CTF refinement
Bayesian polishing
Mask creation
Use parallel disc I/O? Yes
Number of pooled particles: 30
Pre-read all particles into RAM? Yes
Copy particles to scratch directory:
Combine iterations through disc? No
Use GPU acceleration? No
Which GPUs to use:
Print command
Schedule
Continue!
Job actions
Current job: 081: Class2D/job081/
Display:
Finished jobs
087: Class2D/job087/
086: Class2D/job086/
085: Class2D/job085/
084: Class2D/job084/
083: Class2D/job083/
082: Class2D/job082/
081: Class2D/job081/
080: Class2D/job080/
079: Class2D/job079/
078: Class2D/job078/
077: Class2D/job077/
Running jobs
Input to this job
026: Extract/job026/
Scheduled jobs
Output from this job
==========
Expectation iteration 25 of 25
3/ 3 sec ~~(,_,">
Maximization ...
0/ 0 sec ~~(,_,">
Done!



- **relion_refine_mpi** (GPU-accelerated job)
  - (Compute) Use GPU acceleration? : Yes
  - (Compute) Which GPUs to use? : 0:1 (i.e., we will assign each slave process to GPU device index 0 and 1, respectively)
  - (Running) Number of MPI procs: 3 (1 master and 2 slave processes)
  - (Running) Number of threads: 1
  - (Running) Submit to queue? : Yes
  - (Running) Queue name : <your own queue name> (e.g., q02)
  - (Running) Resource Requirements : nodes=1:ppn=3:gpus=2
  - (Running) Standard submission script : /tem/home/tem/_Applications/relion-3.0.7/test/bin/qsub-relion3-gpu.bash


RELION-3.0.7: /tem/home/tem/tmp/relion_tutorial/relion_p1
File
Jobs
Autorun
I/O
CTF
Optimisation
Sampling
Helix
Compute
Running
Import
Motion correction
CTF estimation
Manual picking
Auto-picking
Particle extraction
Particle sorting
Subset selection
2D classification
3D initial model
3D classification
3D auto-refine
3D multi-body
CTF refinement
Bayesian polishing
Mask creation
Use parallel disc I/O? Yes
Number of pooled particles: 30
Pre-read all particles into RAM? Yes
Copy particles to scratch directory:
Combine iterations through disc? No
Use GPU acceleration? Yes
Which GPUs to use: 0:1
Print command
Schedule
Continue!
Job actions
Current job: 128: Class2D/job128/
Display:
Finished jobs
127: Class2D/job127/
126: CtfFind/job126/
125: CtfFind/job125/
124: MotionCorr/job124/
123: MotionCorr/job123/
122: MotionCorr/job122/
121: MotionCorr/job121/
117: MotionCorr/job117/
114: MotionCorr/job114/
111: MotionCorr/job111/
110: MotionCorr/job110/
Running jobs
128: Class2D/job128/
Input to this job
026: Extract/job026/
Scheduled jobs
Output from this job
stdout will go here; double-click this window to open stdout in a separate window
stderr will go here; double-click this window to open stderr in a separate window


RELION-3.0.7: /tem/home/tem/tmp/relion_tutorial/relion_p1
File Jobs Autorun
I/O CTF Optimisation Sampling Helix Compute Running
Import
Motion correction
CTF estimation
Manual picking
Auto-picking
Particle extraction
Particle sorting
Subset selection
2D classification
3D initial model
3D classification
3D auto-refine
3D multi-body
CTF refinement
Bayesian polishing
Mask creation
Number of MPI procs: 3
Number of threads: 1
Submit to queue? Yes
Queue name: q02
Queue submit command: qsub
Resource Requirements nodes=1:ppn=3:gpus=2
Standard submission script: est/bin/qsub-relion3-gpu.bash
Browse
Minimum dedicated cores per node: 1
Additional arguments:
Print command
Schedule
Continue!
Job actions
Current job: 128: Class2D/job128/
Display:
Finished jobs
127: Class2D/job127/
126: CtfFind/job126/
125: CtfFind/job125/
124: MotionCorr/job124/
123: MotionCorr/job123/
122: MotionCorr/job122/
121: MotionCorr/job121/
117: MotionCorr/job117/
114: MotionCorr/job114/
111: MotionCorr/job111/
110: MotionCorr/job110/
Running jobs
128: Class2D/job128/
Input to this job
026: Extract/job026/
Scheduled jobs
Output from this job
stdout will go here; double-click this window to open stdout in a separate window
stderr will go here; double-click this window to open stderr in a separate window

CHAPTER

FIVE

CISTEM

cisTEM is user-friendly software to process cryo-EM images of macromolecular complexes and obtain high-resolution 3D reconstructions from them. It comprises a number of tools to process image data including movies, micrographs and stacks of single-particle images, implementing a complete "pipeline" of processing steps to obtain high-resolution single-particle reconstructions. (from cisTEM official site https://cistem.org)

5.1 Executing cisTEM

5.1.1 How to start cisTEM data analysis tool

1. You can find out cisTEM applications' environment module path by listing all the module available on TEM service farm.

```
$> module avail

----------------- /tem/home/tem/Modules/Modules/versions -----------------
3.2.10

----------- /tem/home/tem/Modules/Modules/default/modulefiles ------------
apps/gcc/4.4.7/cistem/1.0.0     cuda/9.1
apps/gcc/4.4.7/relion/cpu/3.0.7 modules
apps/gcc/4.4.7/relion/gpu/3.0.7 mpi/gcc/openmpi/1.8.8
```

2. Check the module details for cisTEM application

```
$> module show apps/gcc/4.4.7/cistem/1.0.0

-------------------------------------------------------------------
/tem/home/tem/Modules/Modules/default/modulefiles/apps/gcc/4.4.7/cistem/1.0.0:

module-whatis    Setups `cistem-1.0.0' environment variables
module           load mpi/gcc/openmpi/1.8.8
prepend-path     PATH /tem/home/tem/_Applications/cistem-1.0.0-beta
conflict         apps/gcc/4.4.7/cistem
-------------------------------------------------------------------
```

3. Load the environment module for cisTEM application which you want to execute. As the module specified is loaded, all the modules with dependency are also loaded (you can check these modules with "module list" command)

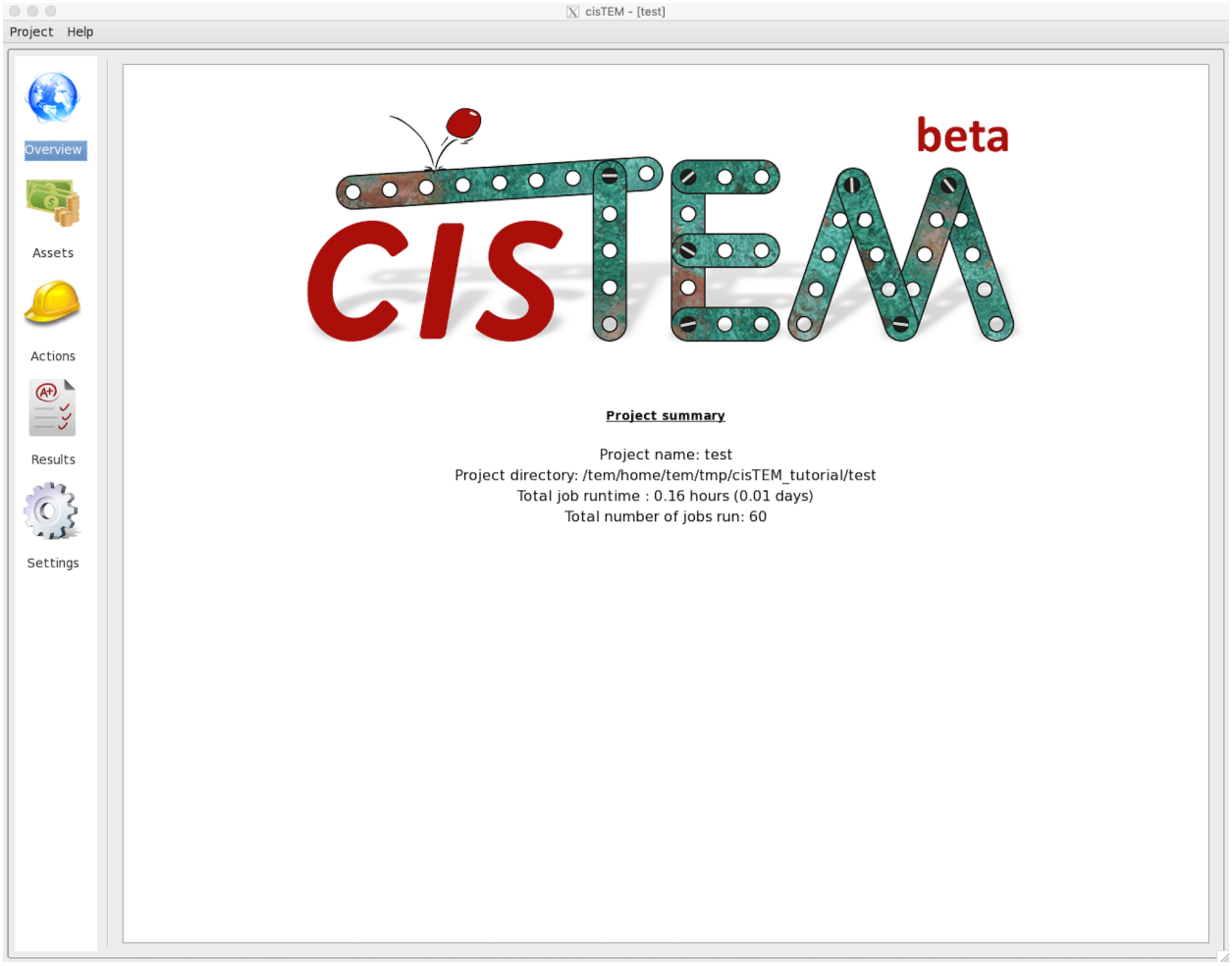
```
$> module load apps/gcc/4.4.7/cistem/1.0.0
$> module list
Currently Loaded Modulefiles:
  1) mpi/gcc/openmpi/1.8.8          2) apps/gcc/4.4.7/cistem/1.0.0
```

4. Check the cisTEM application binary path

```
$> which cisTEM
/tem/home/tem/_Applications/cistem-1.0.0-beta/cisTEM
```

5. Execute the cisTEM application (we assume that X11 forwarding is enabled)

```
$> cisTEM
```



On startup, the GUI presents a list of previously opened projects, as well as options to create a new project or open an existing project. To continue a previous project, click on the provided link.

5.2 Run profiles for job submission

5.2.1 Profile templates

If you need cisTEM to work on multiple computing servers in a cluster which is managed with Torque, you should check out (or create) a "Run Profile" in cisTEM's settings tab. You can find a shell script available in following file paths.

```
(cisTEM with job outputs and errors) /tem/home/tem/_Applications/cistem-1.0.0-beta/
↪qsub-cisTEM-cpu.sh
(cisTEM without outputs and errors)  /tem/home/tem/_Applications/cistem-1.0.0-beta/
↪qsub-cisTEM-cpu-noout.sh
```

For qsub-cisTEM-cpu.sh,

```
#!/bin/bash
queue=
while getopts ":q:" OPTION
do
  case "${OPTION}" in
    q) queue="${OPTARG}";;
  esac
done
shift $((OPTIND-1))

cat - <<EOF | qsub
#!/bin/bash
#PBS -N cisTEM.${1}
${queue:+#PBS -l nodes=1:ppn=1:${queue}}
${queue:+#PBS -q ${queue}}

module load apps/gcc/4.4.7/cistem/1.0.0
${@}
EOF
```

For qsub-cisTEM-cpu-noout.sh,

```
#!/bin/bash
queue=
while getopts ":q:" OPTION
do
  case "${OPTION}" in
    q) queue="${OPTARG}";;
  esac
done
shift $((OPTIND-1))

cat - <<EOF | qsub
#!/bin/bash
#PBS -N cisTEM.${1}
#PBS -e /dev/null
#PBS -o /dev/null
${queue:+#PBS -l nodes=1:ppn=1:${queue}}
${queue:+#PBS -q ${queue}}

module load apps/gcc/4.4.7/cistem/1.0.0
```

(continues on next page)

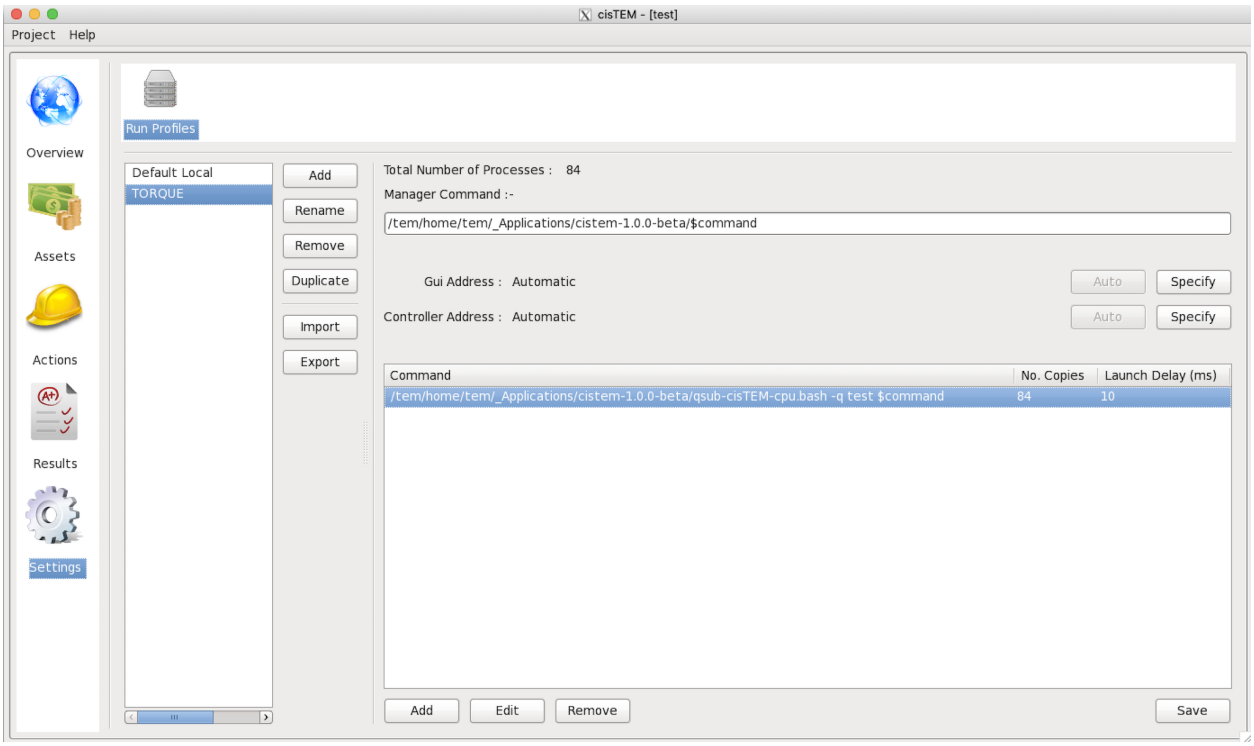
(continued from previous page)

```
${@}
EOF
```

5.2.2 Adding a new Run Profile

In cisTEM settings, add a new "Run Profile" (called TORQUE here) with the following parameters :

- Manager Command: /tem/home/tem/_Applications/cistem-1.0.0-beta/$command
- Gui Address: Automatic
- Controller Address: Automatic
- Command -> Edit:
  - Command: /tem/home/tem/_Applications/cistem-1.0.0-beta/qsub-cisTEM-cpu.sh **-q** <**your_own_queue_name**> $command
  - No. Copies: 84
  - Delay (ms): 10

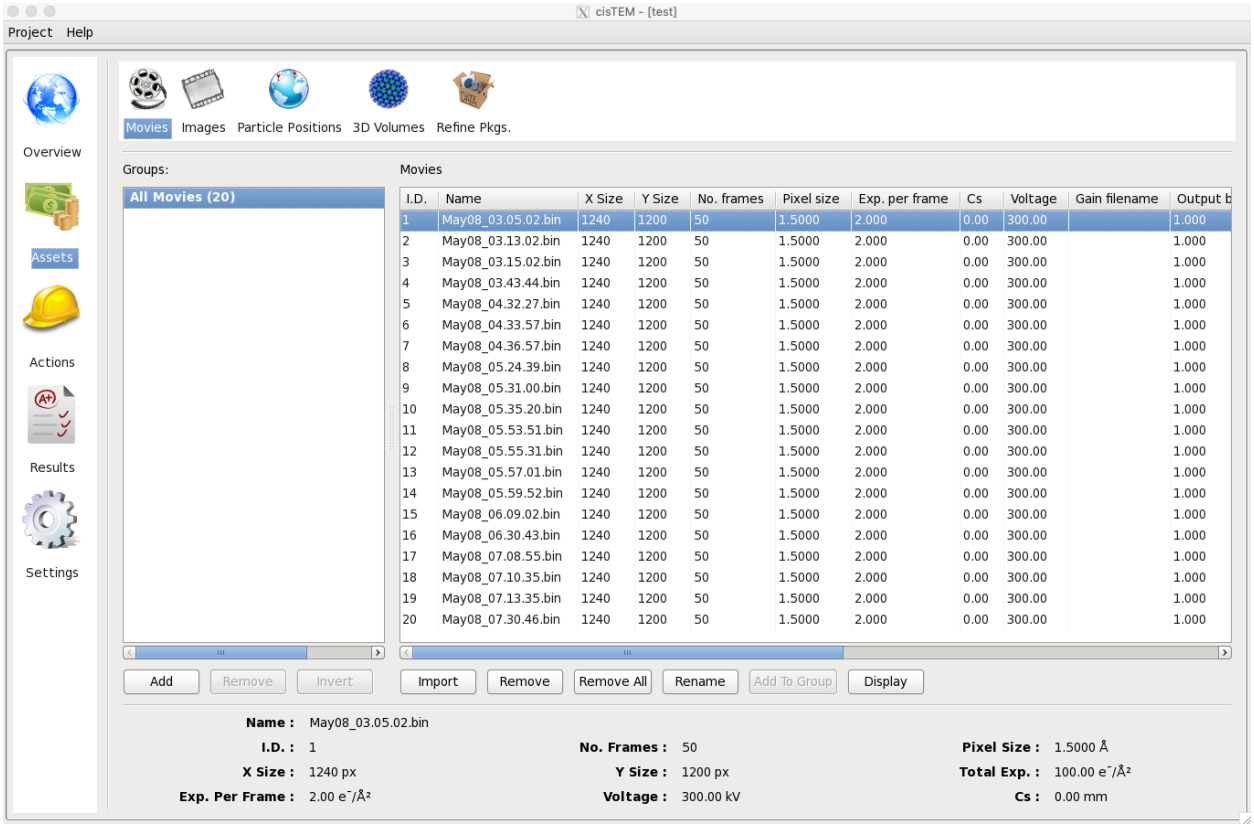


5.3 Exampels of running cisTEM jobs

With the above cisTEM setting, here, we provide some examples of running cisTEM jobs with cisTEM GUI tools.

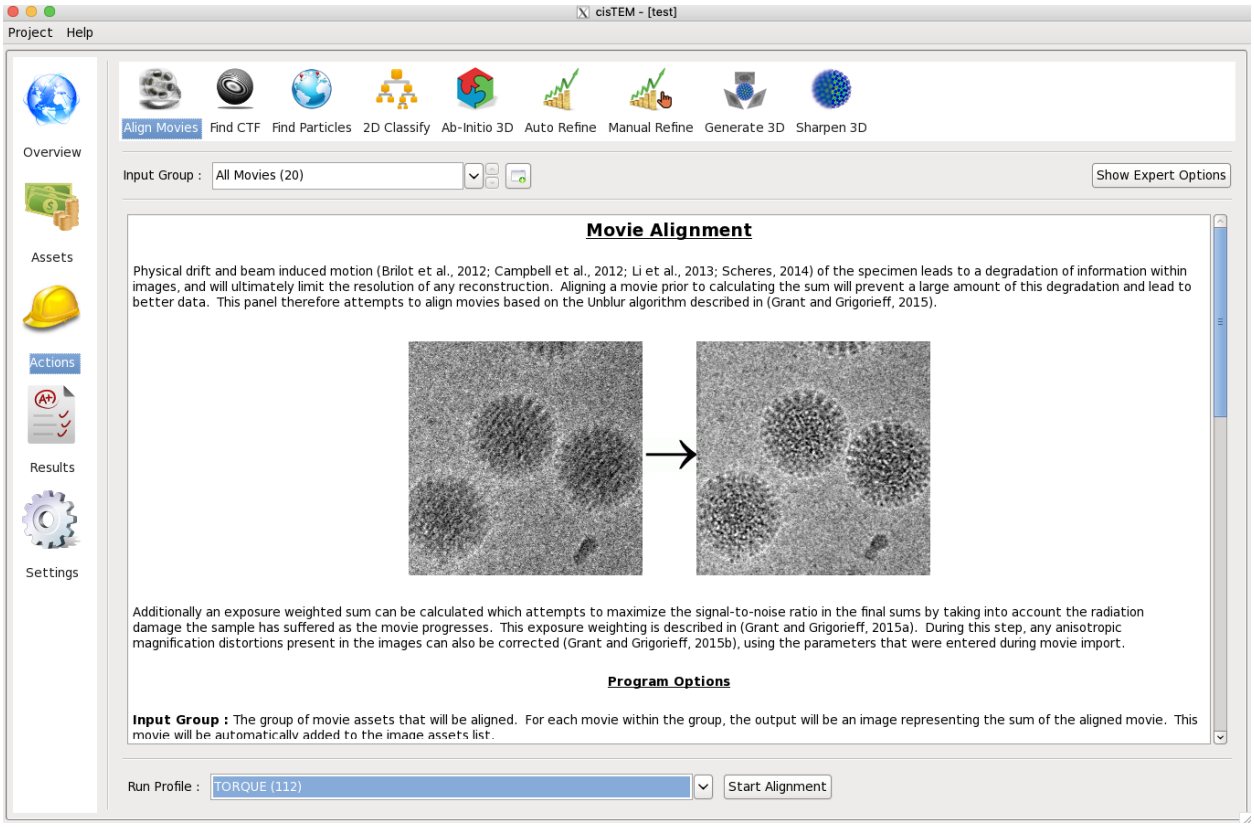
5.3.1 Importing Movies and images

Once a project is open or has been newly created, Assets can be imported. These will usually be Movies or Images but can also be Particle Positions, 3D Volumes and Refinement Packages. Click on Assets, then Movies and Import. In the dialog, select “Add Directory” and navigate to the directory containing your own movies. The movies are all part of a group called “All Movies”. Additional groups can be created using “Add” to select subsets of a dataset for further processing. You should continue with all the data for now. If images are available instead of movies, these can be imported as Image Assets in the same way as Movies, by clicking “Images”.



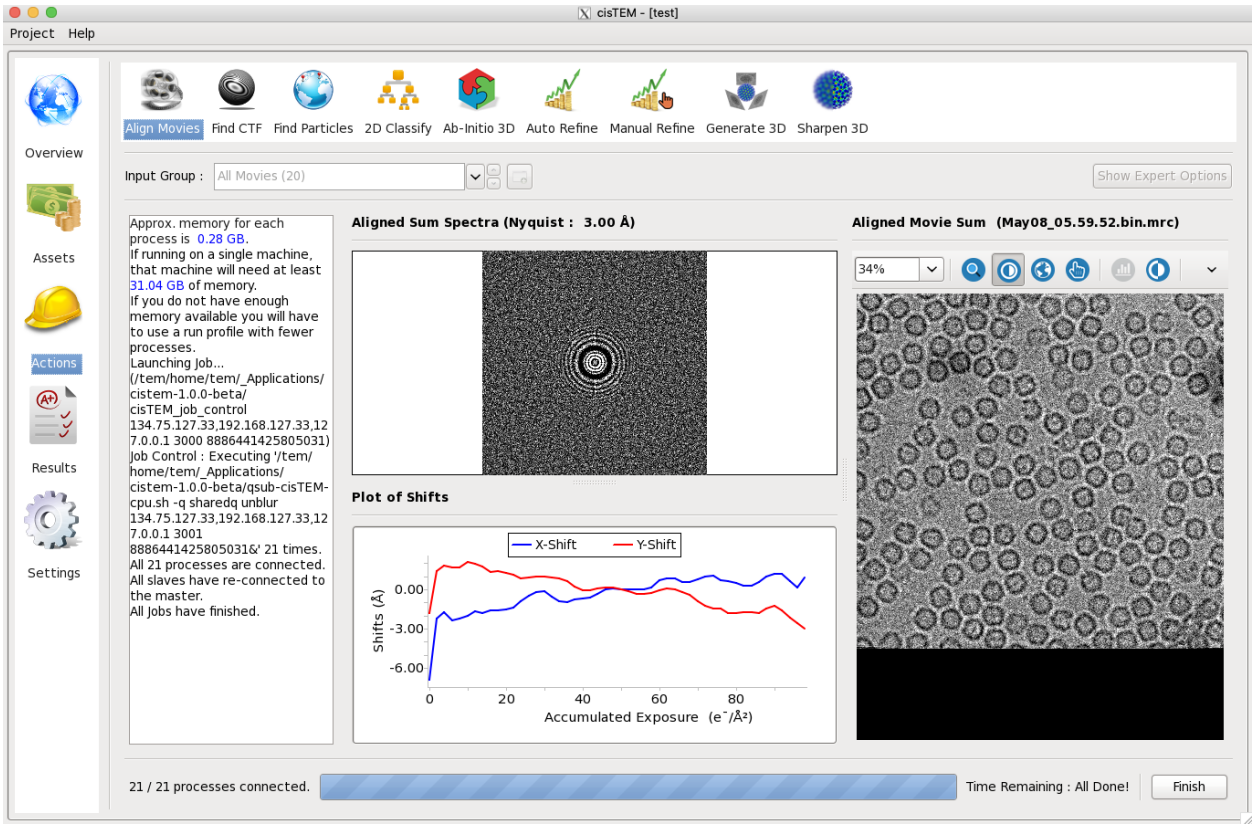
5.3.2 Movie Alignment

Movie data collection and frame alignment have been part of the single-particle image processing pipeline since it was first introduced by Brilot et al. in 2012. The original software **Unblur** was developed further by Grant & Grigorieff (2015) when exposure weighting was added to take into account the radiation-dependent signal loss when adding movie frames, yielding signal-optimized frame sums. cisTEM implements the Unblur algorithm in the Align Movies panel, which also provides some background to the method. Click “Actions” and select “Align Movies” to call up the panel.



Actions panels display parameters that you can change. Some of these are shown on the main panel while others are only accessible when “Show Expert Options” is selected. Movie alignment usually works with the default parameters and you should simply click “Start Alignment” near the bottom of the panel. You will notice that next to the start button a menu is shown that allows you to select different run profiles. The Local profile should **NOT** be selected because it will launch alignment jobs onto the login node but you should change to other profiles (for example, TORQUE profile) if these were previously set up under Settings.

The alignment of all the movies takes less than a minute. While the job is running, X,Y traces are displayed for some of the movies and a progress bar indicates the time left until completion of the job. After termination (you must click on “Finish” at the end of all jobs), you can inspect the results by clicking “Results”


cisTEM - [test]
Project
Help
Overview
Assets
Actions
Results
Settings
Align Movies
Find CTF
Find Particles
2D Classify
Ab-Initio 3D
Auto Refine
Manual Refine
Generate 3D
Sharpen 3D
Input Group :
All Movies (20)
Show Expert Options
Approx. memory for each process is 0.28 GB.
If running on a single machine, that machine will need at least 31.04 GB of memory.
If you do not have enough memory available you will have to use a run profile with fewer processes.
Launching Job...
(/tem/home/tem/_Applications/cistem-1.0.0-beta/cisTEM_job_control 134.75.127.33,192.168.127.33,127.0.0.1 3000 8886441425805031)
Job Control : Executing '/tem/home/tem/_Applications/cistem-1.0.0-beta/qsub-cisTEM-cpu.sh -q sharedq unblur 134.75.127.33,192.168.127.33,127.0.0.1 3001 8886441425805031&' 21 times.
All 21 processes are connected.
All slaves have re-connected to the master.
All Jobs have finished.
Aligned Sum Spectra (Nyquist : 3.00 Å)
Plot of Shifts
X-Shift
Y-Shift
Shifts (Å)
0.00
-3.00
-6.00
0
20
40
60
80
Accumulated Exposure (e⁻/Å²)
Aligned Movie Sum (May08_05.59.52.bin.mrc)
34%
21 / 21 processes connected.
Time Remaining : All Done!
Finish