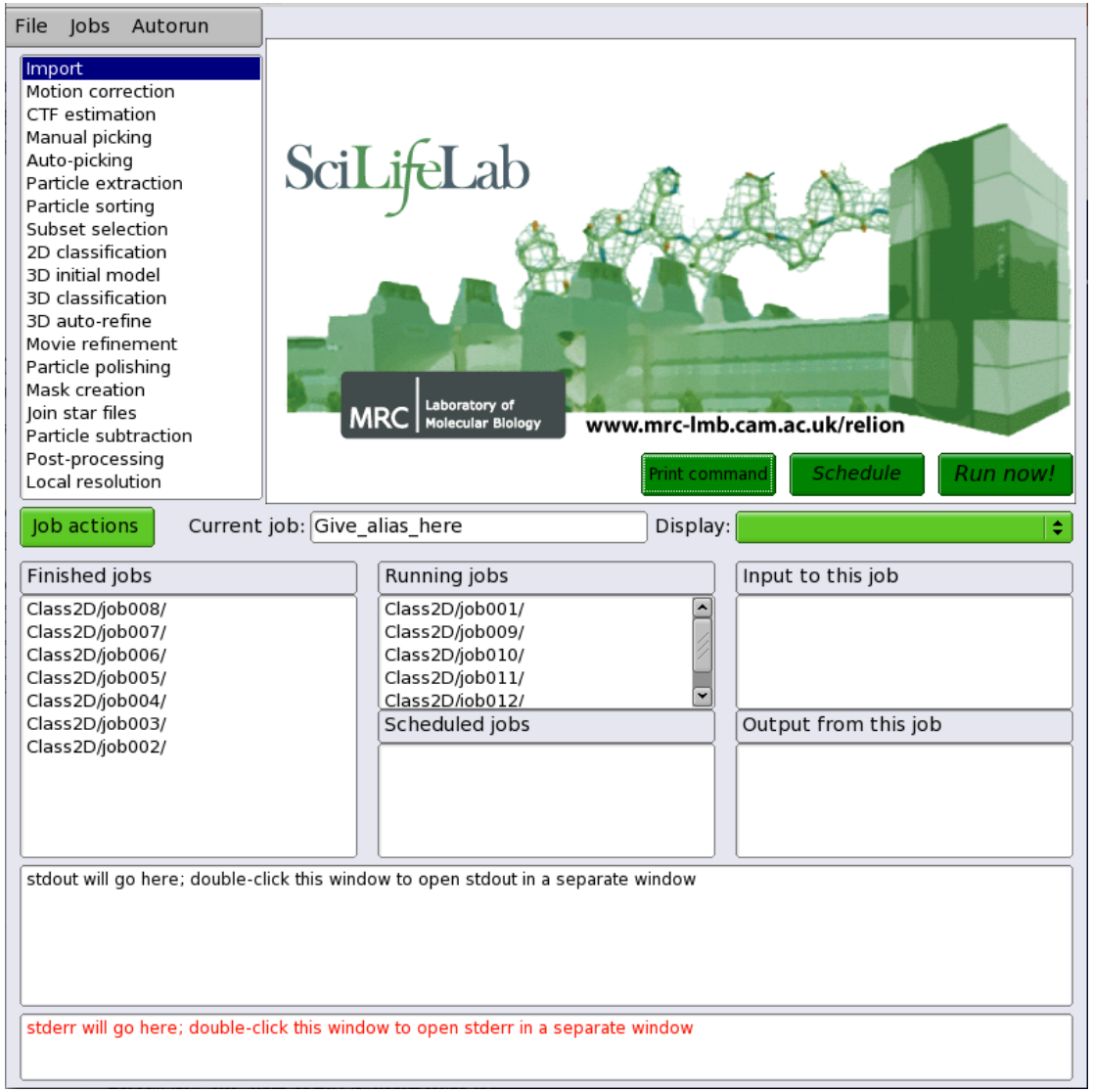
RELION Tips and Tricks

David Hoover, PhD

HPC @ NIH

Using the GUI



Don't run GUI on the login node

- Some job types run quickly
- You will forget you are on the login node

Be aware of options passed to sinteractive

- Default --mem-per-cpu=768

  SLURM_MEM_PER_CPU=768

- Subsequent batch MPI jobs may get confused

  command line > environment variable > within batch script

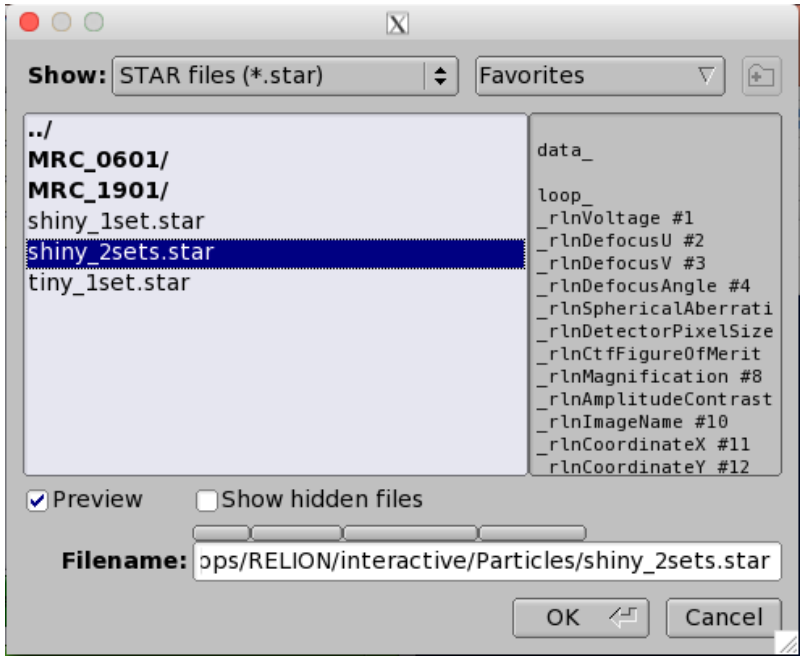
```
SLURM_MEM_PER_CPU=768
SLURM_MEM_PER_NODE=245760
```

```
srun: Force Terminated job step 58421330.0
srun: error: cn2692: task 0: Killed
srun: error: cn2692: task 1: Killed
srun: Force Terminated job step 58421330.1
```

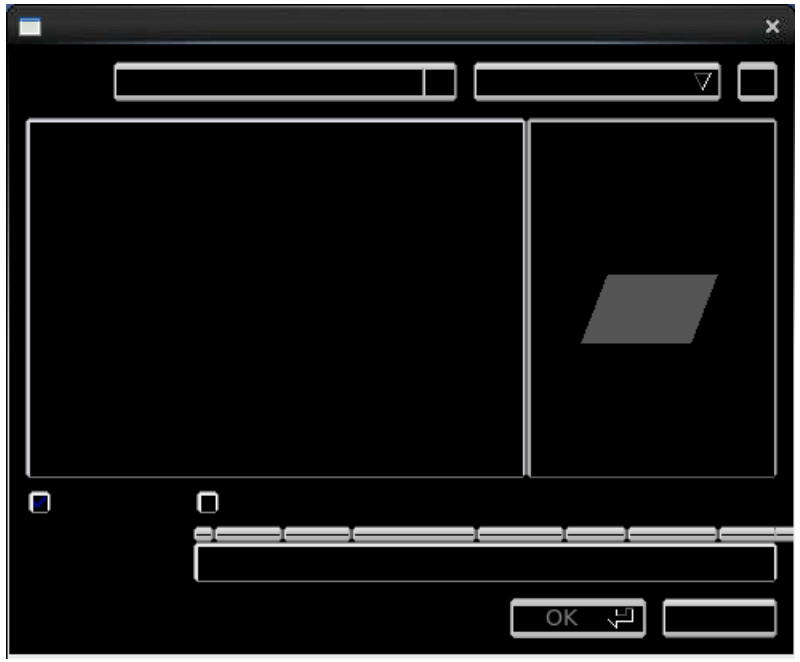
```
#!/bin/bash
#SBATCH --constraint=gpuk80
#SBATCH --ntasks=16
#SBATCH --nodes=4
#SBATCH --cpus-per-task=4
#SBATCH --mem=240g
#SBATCH --partition=XXXqueueXXX
#SBATCH --gres=gpu:k80:4,lscratch:XXXextra1XXX
#SBATCH --error=XXXerrfileXXX
#SBATCH --output=XXXoutfileXXX
#SBATCH --time=XXXextra2XXX

unset SLURM_MEM_PER_CPU
srun --mpi=pmi2 relion_refine_mpi --i Particles
```

NX messes up file browsing

Xquartz:

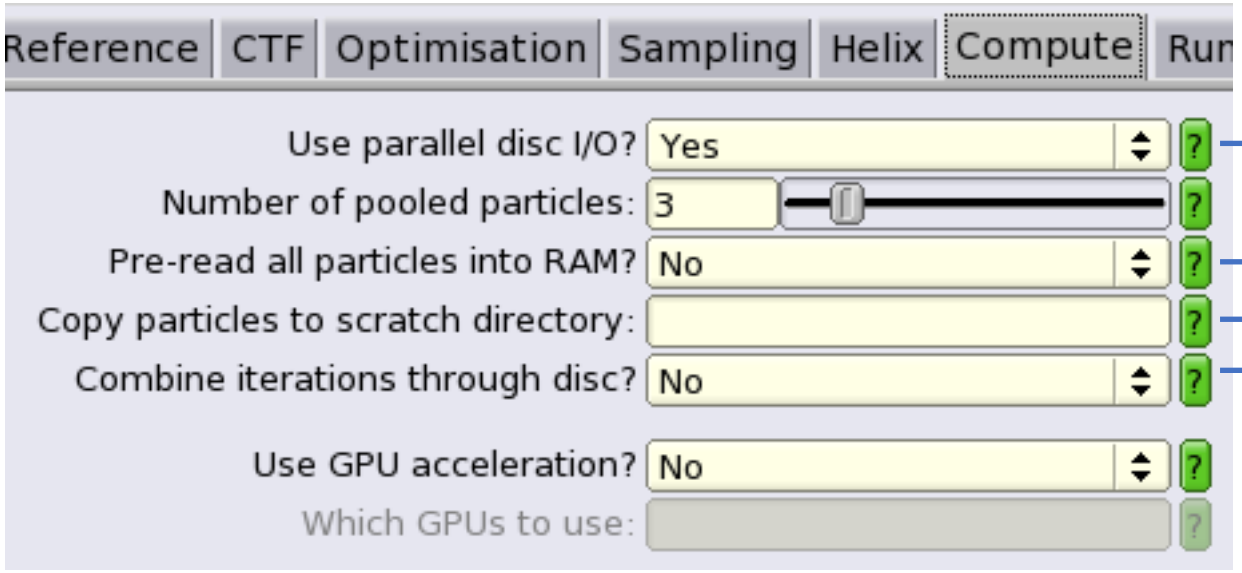


NX:



RELION Compute Options

- Four major choices to be made:



`--no_parallel_disc_io`

always set to Yes

`--preread_images`

depends, exclusive from below

`--scratch_dir=`

usually good idea, exclusive from above

`--dont_combine_weights_via_disc`

always set to No

```
relion_command ... --dont_combine_weights_via_disc --scratch_dir=/lscratch/$SLURM_JOB_ID ...
```

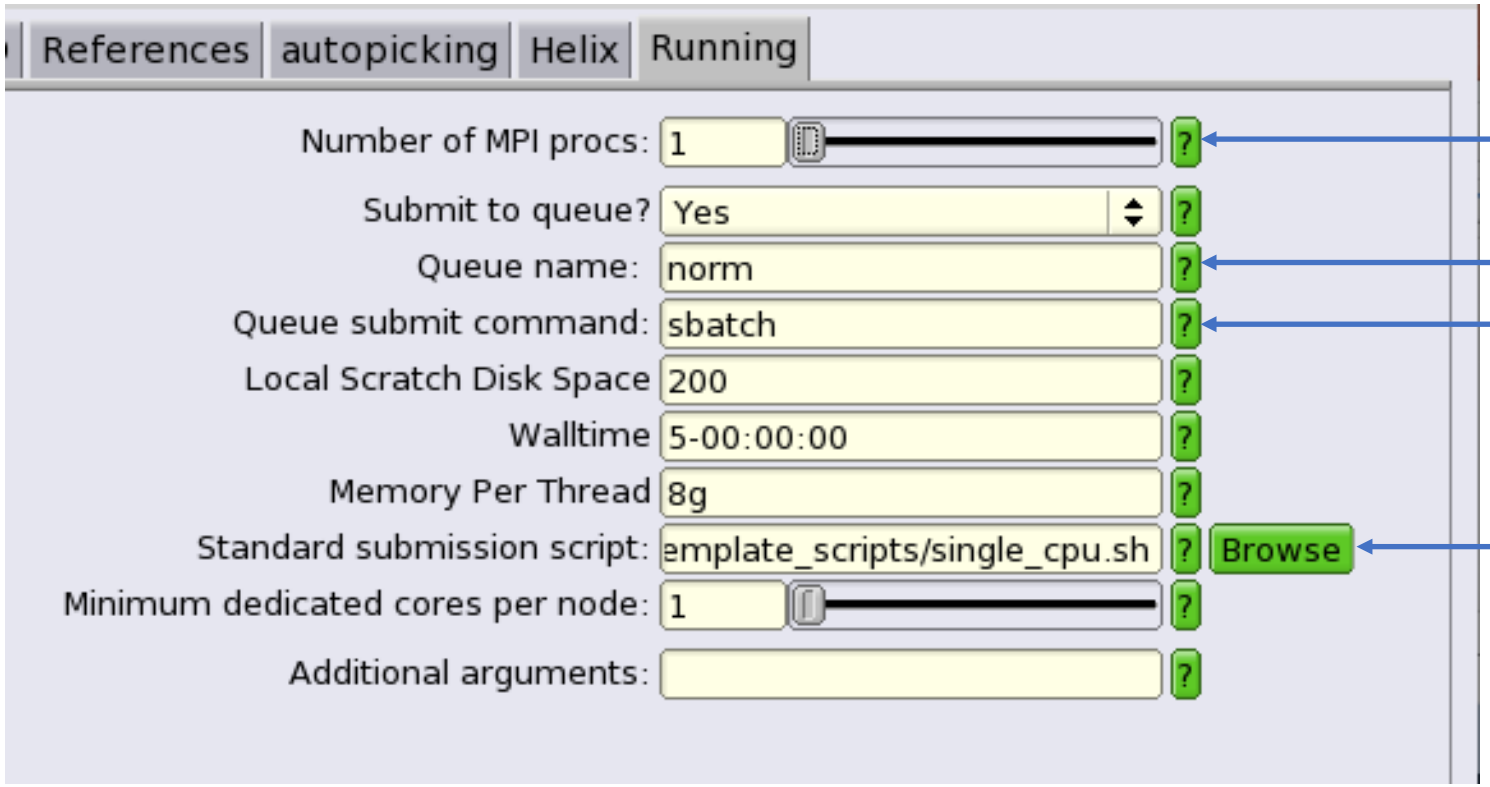
or

```
relion_command ... --dont_combine_weights_via_disc --preread_images ...
```

Choice

- Fastest execution (under ideal conditions) is to NOT use --scratch_dir or --preread_images
- OK for short jobs, but long (>2h) and large jobs will have I/O hassles
- --preread_images is perhaps a little faster
- --scratch_dir most reliable

Running/Batch System Options



1 = non-MPI
>1 = MPI

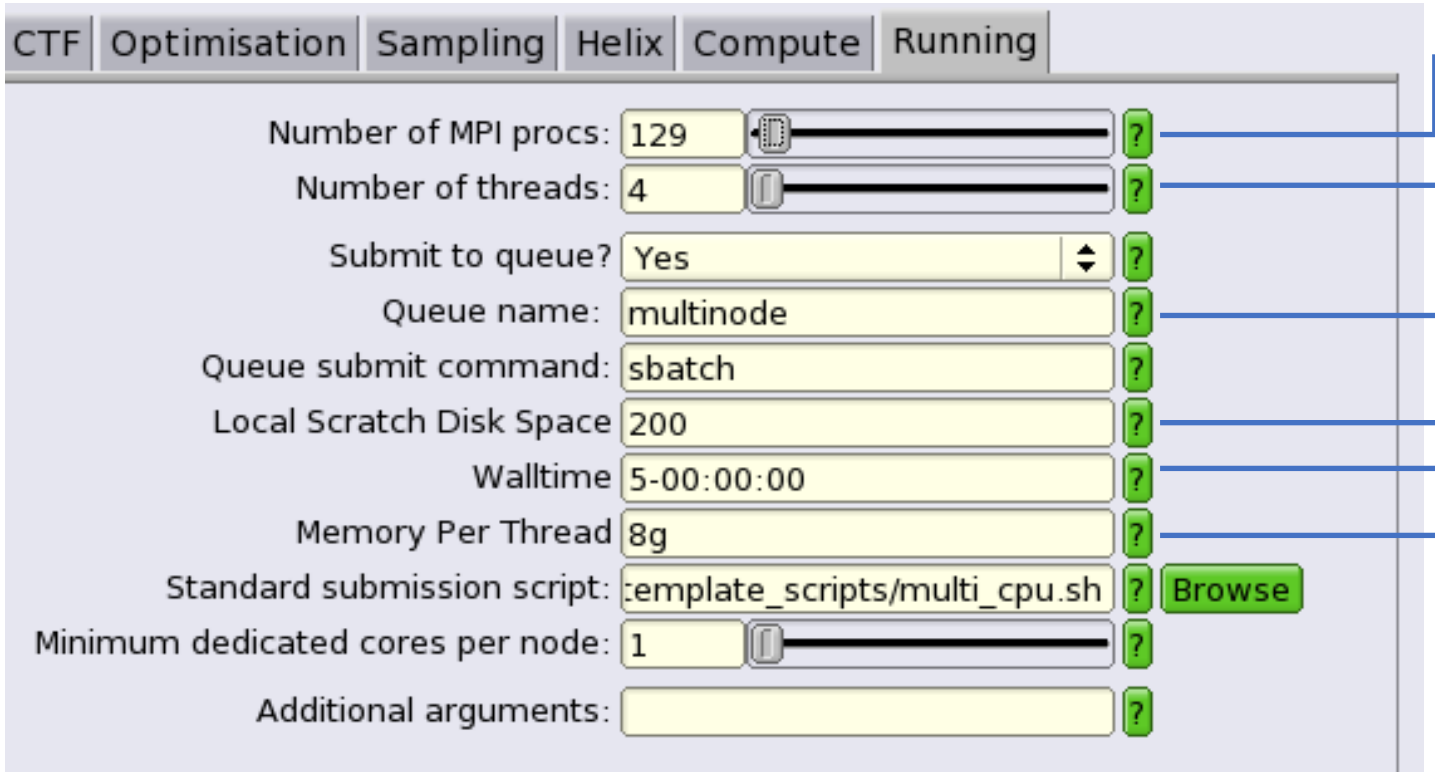
must match job requirements
(norm, multinode, gpu)

can include overriding options
(--mail-type=FAIL,
--dependency=12345678,
--qos=XYZ,
--job-name=ABC.try01,
--constraint=x2695)

many choices available

/usr/local/apps/RELION/batch_template_scripts

Running/Batch System Options



XXXmpinodesXXX, --ntasks

XXXthreadsXXX, --cpus-per-task

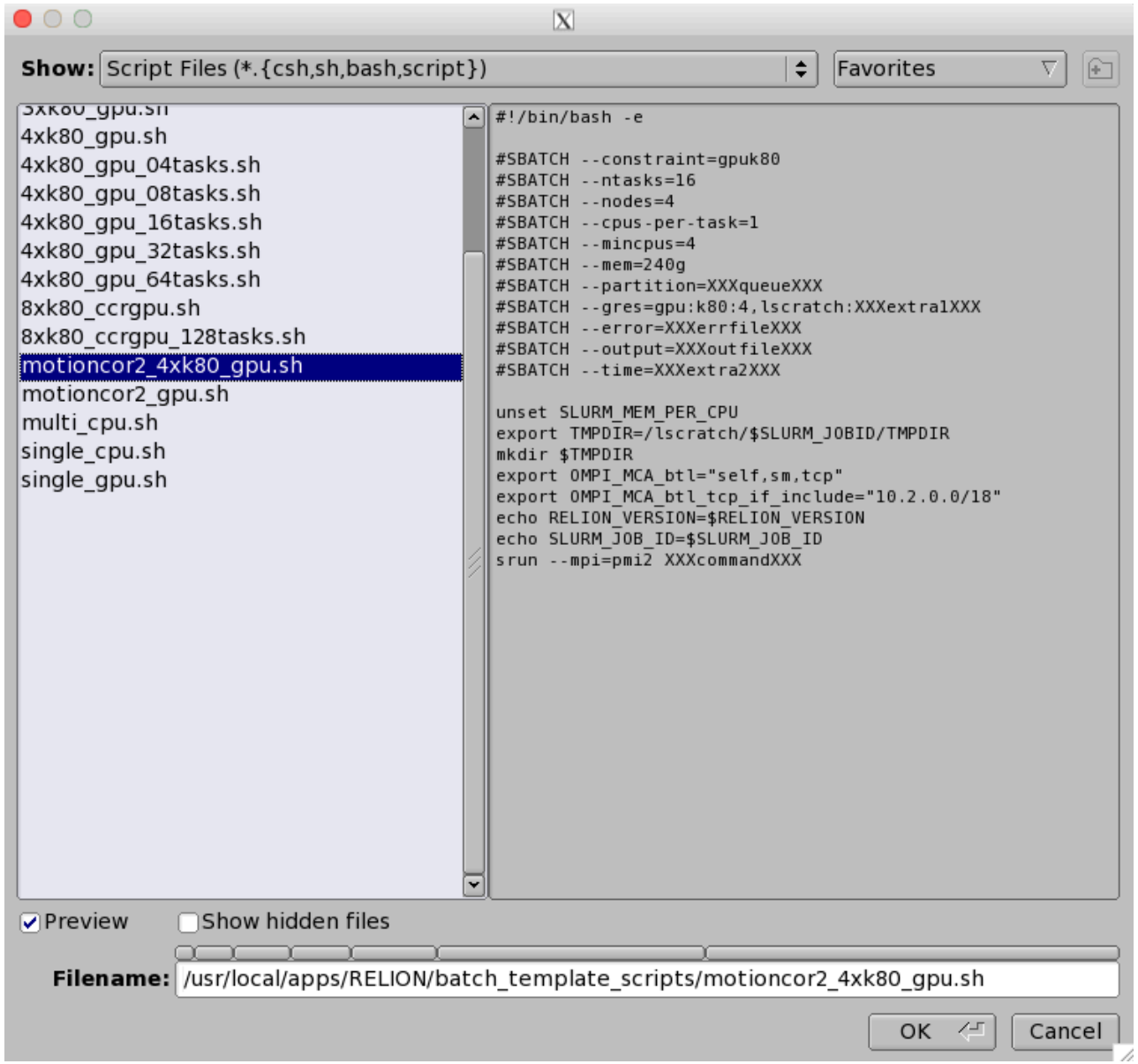
XXXqueueXXX, --partition

XXXextra1XXX, --gres=lscratch:

XXXextra2XXX, --time

XXXextra3XXX, --mem-per-cpu

Understand batch template script



Memory

- Minimum amount of memory (per MPI rank) for --preread_images =

*number of particles * box_size * box_size * 4 / (1024 * 1024 * 1024)*

- E.g. 100K particles, 360 pixel box will need at least 48GB per task
- Memory use will grow 10-20% over time
- Convert to --mem-per-cpu = [*calc RAM required*]/--cpus-per-task
- Same example above with -j 4 (--cpus-per-task=4) needs 14g per cpu

TMPDIR

```
export TMPDIR=/lscratch/$SLURM_JOBID/TMPDIR
mkdir $TMPDIR
```

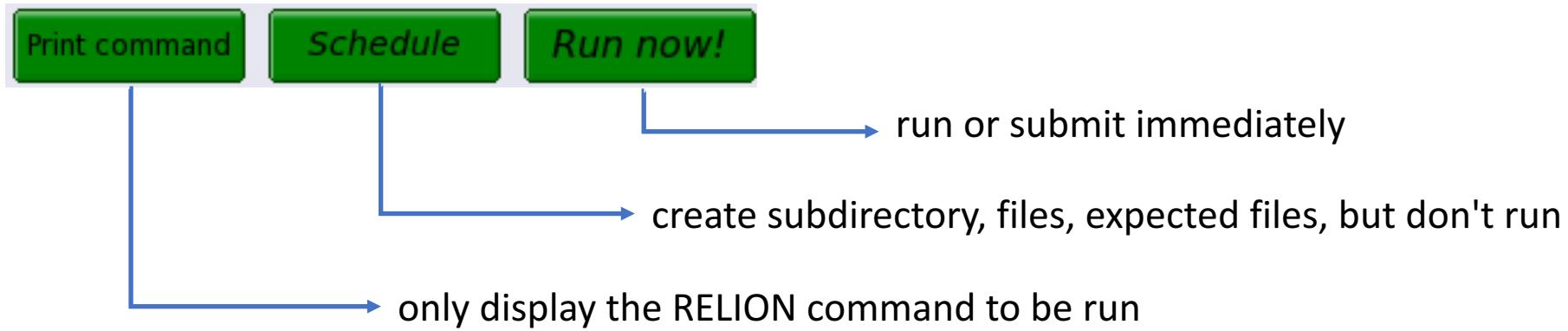
- Always a good idea to allocate local scratch
- MPI requires $TMPDIR for job execution
- /tmp is NOT regularly cleaned up, while /lscratch is
- --scratch_dir=

"Dumb down" the network interface

```
export OMPI_MCA_btl="self,sm,tcp"
export OMPI_MCA_btl_tcp_if_include="10.2.0.0/18"
```

- No difference in execution time between Infiniband and plain 'ol TCP
- MPI RELION can get confused on management network interfaces (10.90)
- Will give warnings when a network interface doesn't exist

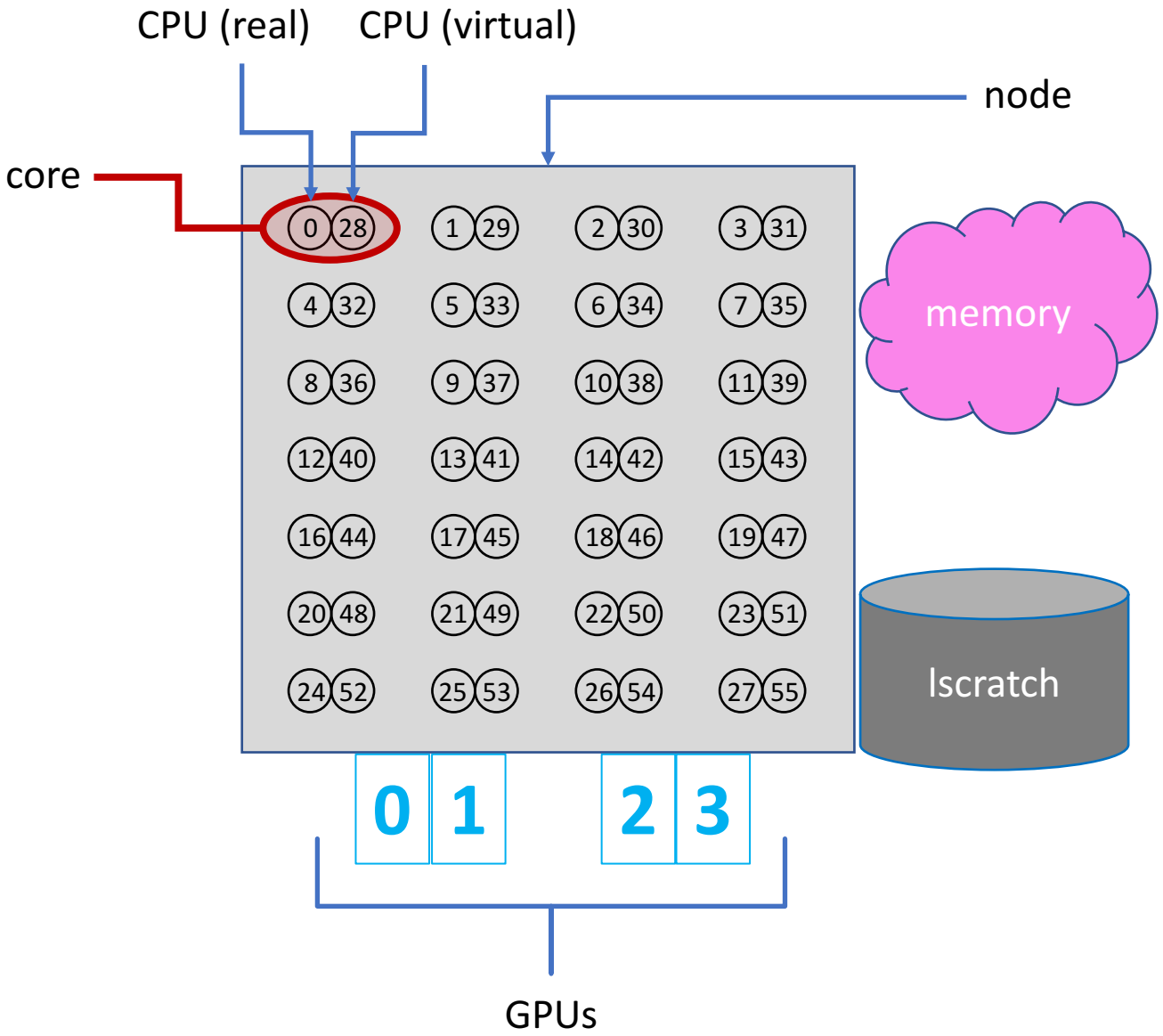
Check twice, run once



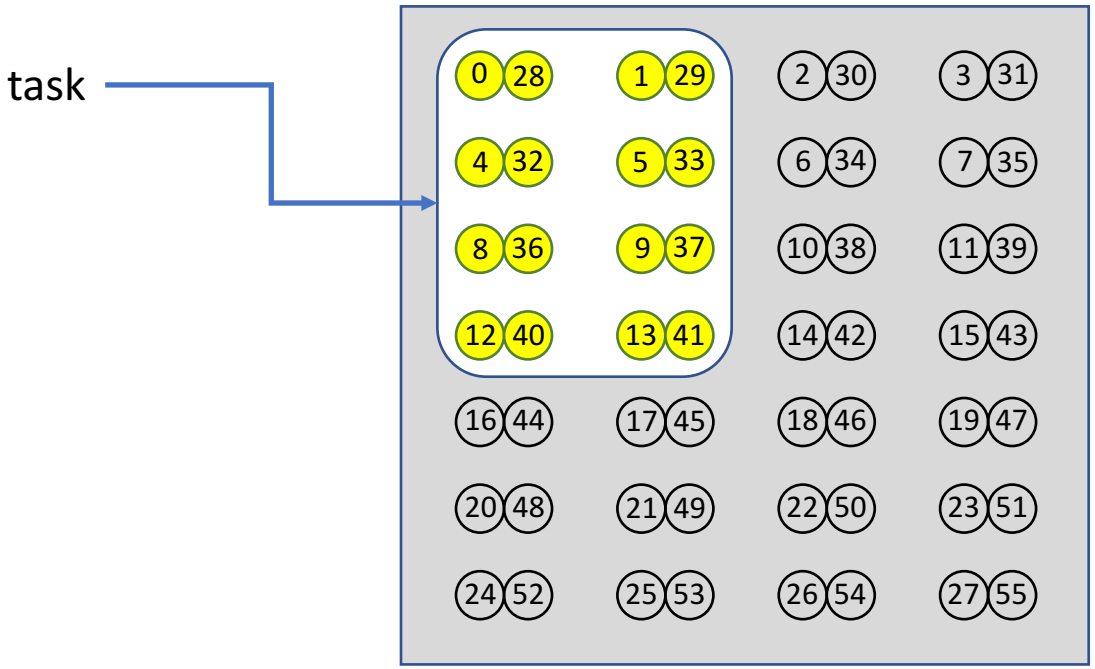
Parallelization on Biowulf using Slurm

- Allocation of resources
- Distribution of resources
- Distribution of tasks on the allocated and distributed resources

Node Anatomy

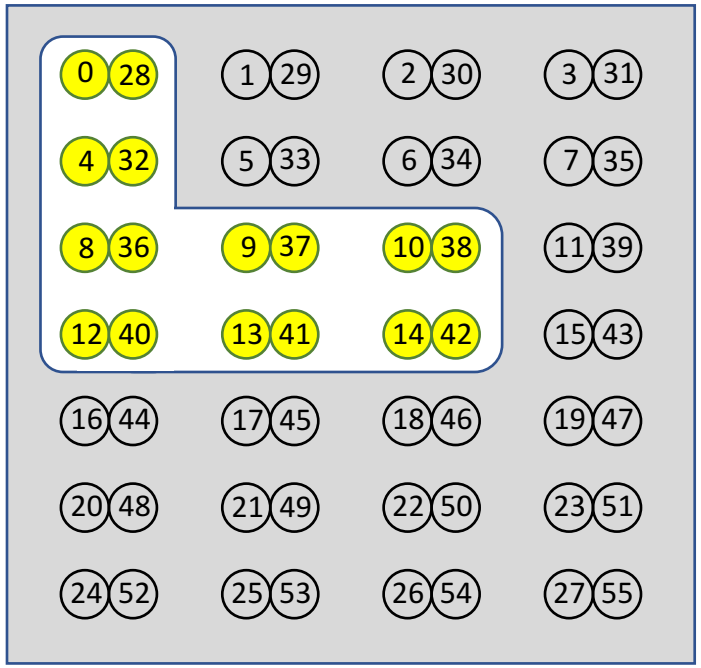


Multithreaded Job



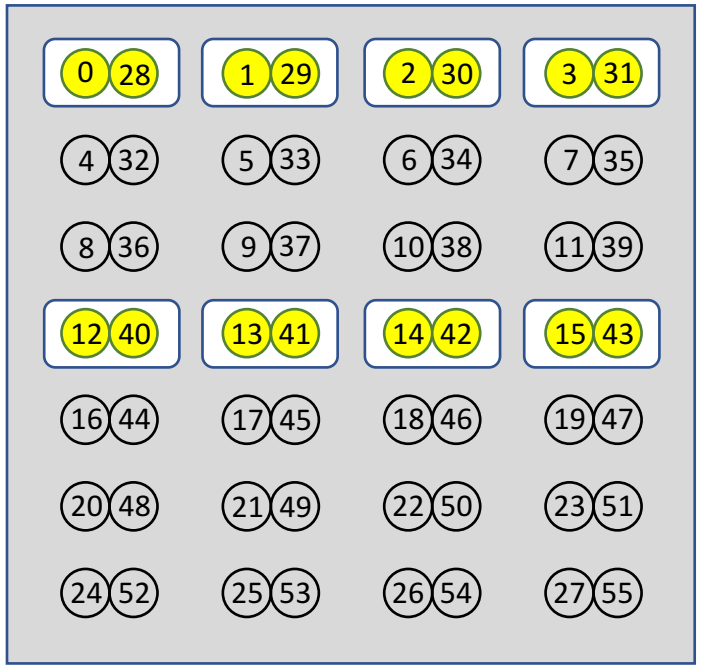

```
sbatch
  --cpus-per-task=16
```

Multithreaded Job




```
sbatch
  --cpus-per-task=16
```

MPI Job – single node




```
sbatch
  --cpus-per-task=2
  --ntasks=8
```

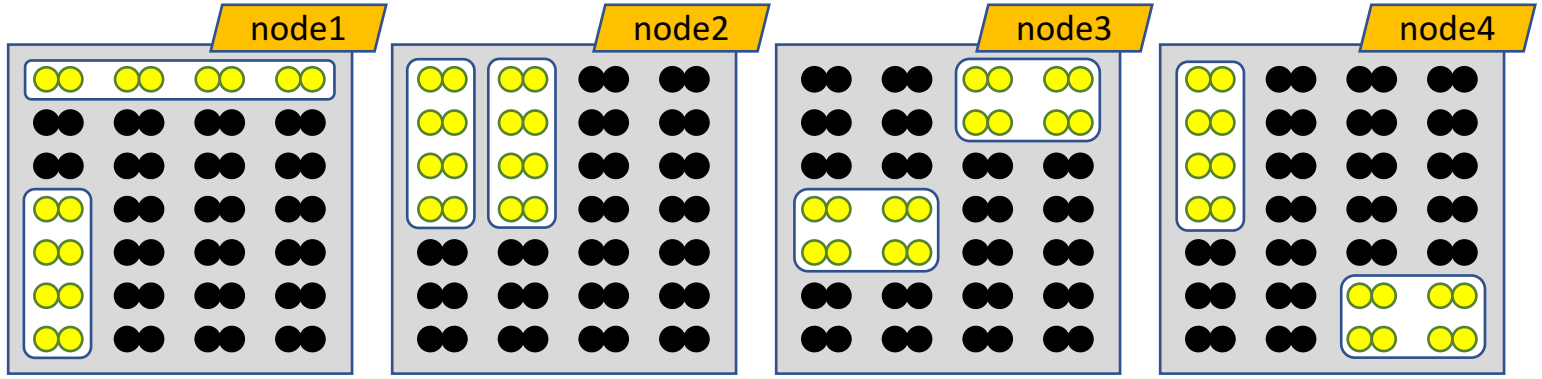
(--partition=norm is implied)

```
srun --mpi=pmi2 executable arg arg arg
```

srun inherits allocation settings from sbatch

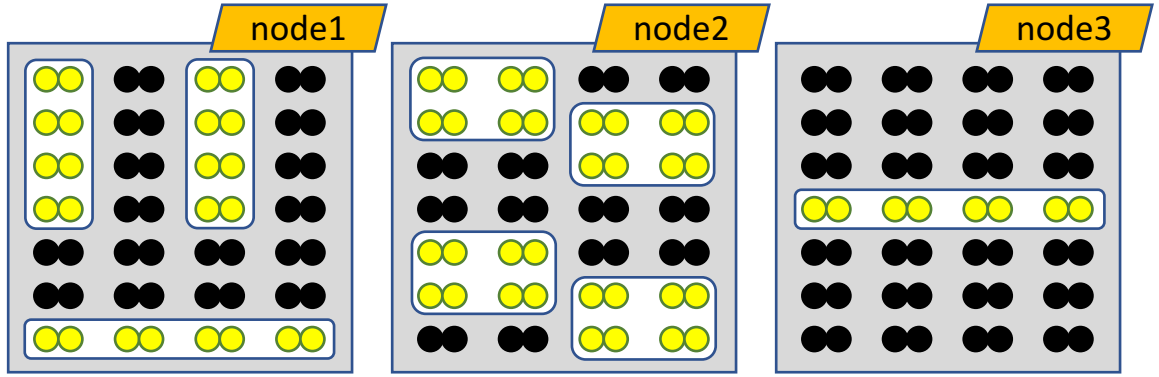
MPI Job – multiple nodes

```
sbatch
  --cpus-per-task=8
  --ntasks=8
  --partition=multinode
```



MPI Job – multiple nodes

```
sbatch
  --cpus-per-task=8
  --ntasks=8
  --partition=multinode
```



RELION Parallelization

- Default method is MPI
- Subset use both MPI and pthreads (hybrid-parallel)
- Master-Slave method for hybrid-parallel
- Master = MPI rank 0, all others are slaves

Standard CPU-based RELION Job

- ~50 nodes

```
#!/bin/bash -e
#SBATCH --gres=lscratch:200
#SBATCH --partition=multinode
#SBATCH --ntasks=513
#SBATCH --cpus-per-task=2
#SBATCH --mem-per-cpu=8g
#SBATCH --error=Class2D/job004/run.err
#SBATCH --output=Class2D/job004/run.out
#SBATCH --time=1-00:00:00

export TMPDIR=/lscratch/$SLURM_JOBID/TMPDIR
mkdir $TMPDIR
export OMPI_MCA_btl="self,sm,tcp"
export OMPI_MCA_btl_tcp_if_include="10.1.0.0/16,10.2.0.0/18"
echo RELION_VERSION=$RELION_VERSION
echo SLURM_JOB_ID=$SLURM_JOB_ID

srun --mpi=pmi2 `which relion_refine_mpi` --o Class2D/job004/run --i Particles/shiny_2sets.star
--dont_combine_weights_via_disc --scratch_dir=/lscratch/$SLURM_JOB_ID --pool 100 --ctf --iter 25
--tau2_fudge 2 --particle_diameter 360 --K 200 --flatten_solvent --zero_mask --oversampling 1
--psi_step 6 --offset_range 5 --offset_step 2 --norm --scale --j 2
```

Standard CPU-based RELION Job

- 129 nodes

```
#!/bin/bash -e
#SBATCH --gres=lscratch:200
#SBATCH --partition=multinode
#SBATCH --ntasks=513
#SBATCH --cpus-per-task=2
#SBATCH --mem-per-cpu=30g
#SBATCH --error=Class2D/job011/run.err
#SBATCH --output=Class2D/job011/run.out
#SBATCH --time=1-00:00:00

export TMPDIR=/lscratch/$SLURM_JOBID/TMPDIR
mkdir $TMPDIR
export OMPI_MCA_btl="self,sm,tcp"
export OMPI_MCA_btl_tcp_if_include="10.1.0.0/16,10.2.0.0/18"
echo RELION_VERSION=$RELION_VERSION
echo SLURM_JOB_ID=$SLURM_JOB_ID

srun --mpi=pmi2 `which relion_refine_mpi` --o Class2D/job011/run --i Particles/shiny_2sets.star
--dont_combine_weights_via_disc --preread_images --pool 100 --ctf --iter 25 --tau2_fudge 2
--particle_diameter 360 --K 200 --flatten_solvent --zero_mask --oversampling 1 --psi_step 6
--offset_range 5 --offset_step 2 --norm --scale --j 2
```

Tidbits

- RELION does not scale much beyond 2000 CPUs
- It takes a LONG time to read images into memory of every single MPI rank
- 50GB images took ~60 minutes to read into memory for ~500 ranks

Understanding GPU Jobs

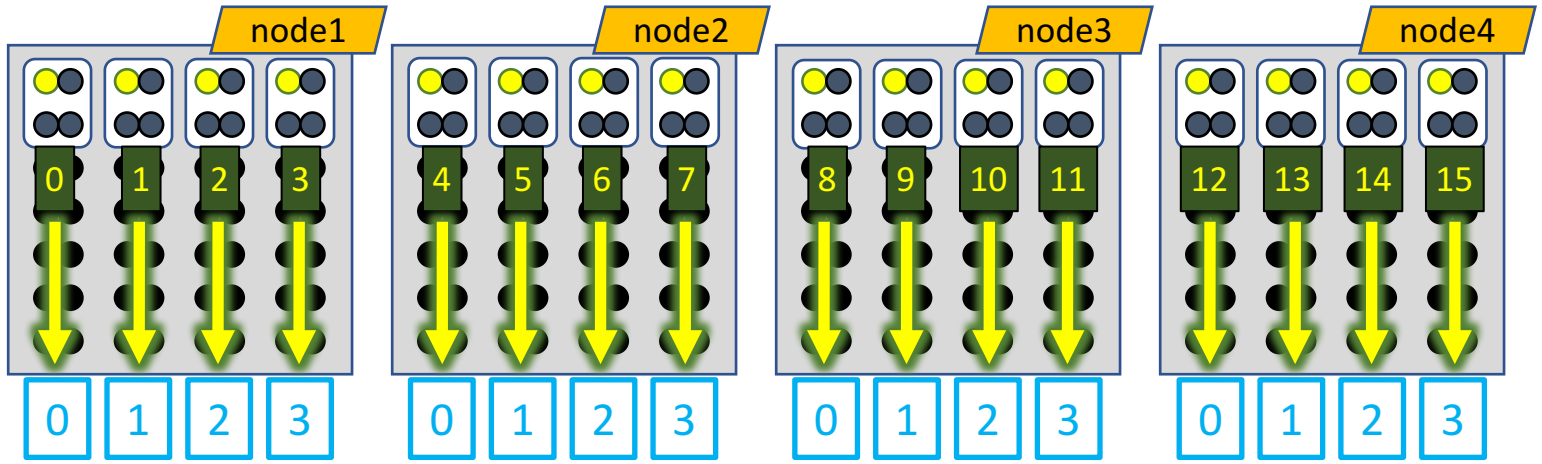
- GPUs accelerate particular execution steps 80x-100x
- Master – Slave model of parallelization conflicts with Slurm resource allocation and MPI rank distribution for GPU nodes
- GPU devices mapped to slaves on the basis of MPI rank id and device id

MPI rank - GPU Mapping

- Default –gpu ""
- GPUs are mapped to MPI ranks in order across ALL MPI ranks, not just those on a node
- You could set --gpu manually, but good luck, stockpile aspirin
- --gpu 0,0,0,0:1,1,1,1:2,2,2,2:3,3,3,3

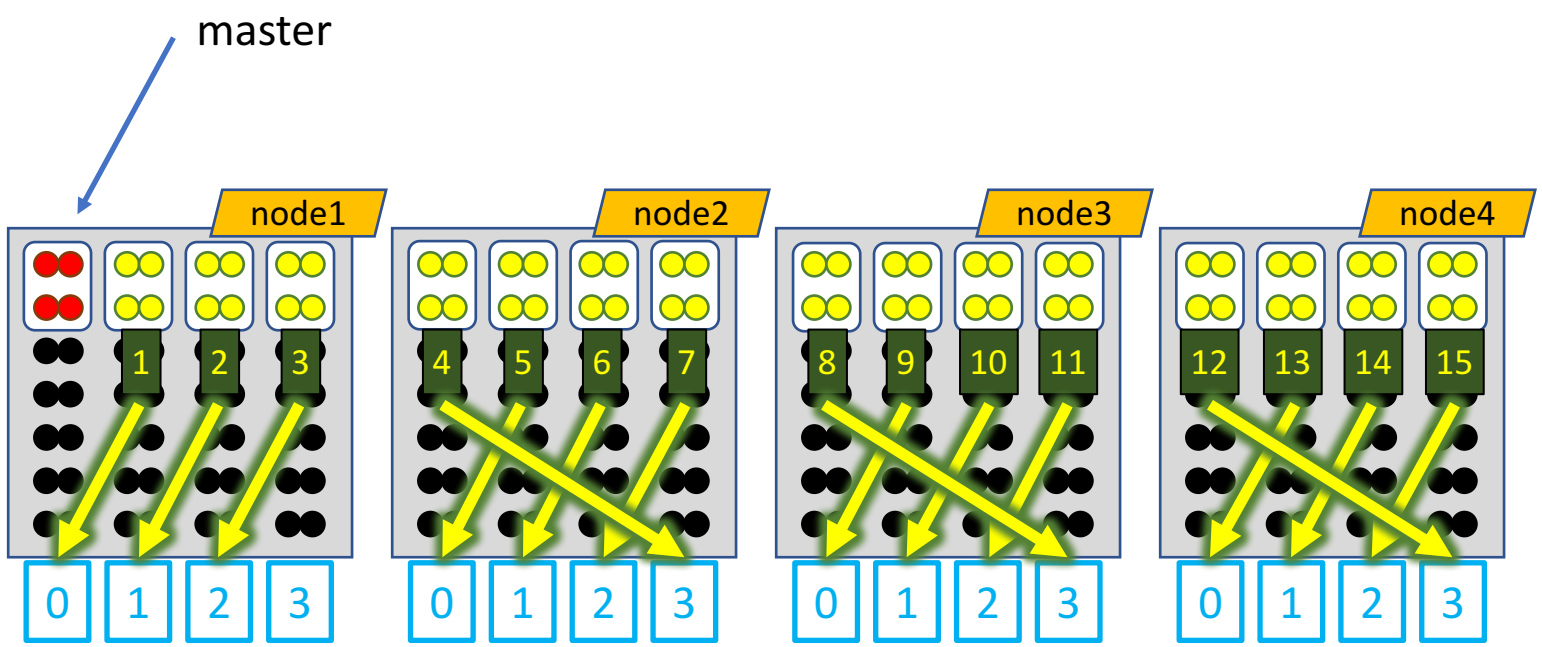
MotionCorr and AutoPick – Ideal Cases

```
sbatch
  --cpus-per-task=4
  --ntasks=16
  --ntasks-per-node=4
  --mem=240g
  --partition=gpu
  --gres=gpu:k80:4,lscratch:100
  --constraint=gpuk80
```



* MotionCorr and AutoPick only run 1 thread per MPI rank
* /lscratch is not used by RELION, but is used by OpenMPI

Class2D, Class3D, Refine – Problems




```
sbatch
  --cpus-per-task=4
  --ntasks=16
  --ntasks-per-node=4
  --mem=240g
  --partition=gpu
  --gres=gpu:k80:4,lscratch:100
  --constraint=gpuk80
```

Class2D, Class3D, Refine – Problems

```
sbatch
  --cpus-per-task=4
  --ntasks=17
  --ntasks-per-node=4
```

not possible due to GPU limit (16 per job)

```
sbatch
  --cpus-per-task=4
  --ntasks=17
  --nodes=4
```

random distribution of tasks on nodes

```
sbatch
  --cpus-per-task=4
  --ntasks=17
  --nodes=4
  --distribution=plane=1
```

complete overutilization of GPUs

```
sbatch
  --cpus-per-task=4
  --ntasks=17
  --nodes=4
  --mincpus=16
```

unpredictable which node receives 5 tasks

Arbitrary Distribution

- Over allocate on purpose

```
sbatch
  --cpus-per-task=4
  --ntasks=20
  --nodes=4
  --ntasks-per-node=5
  --distribution=arbitrary
```

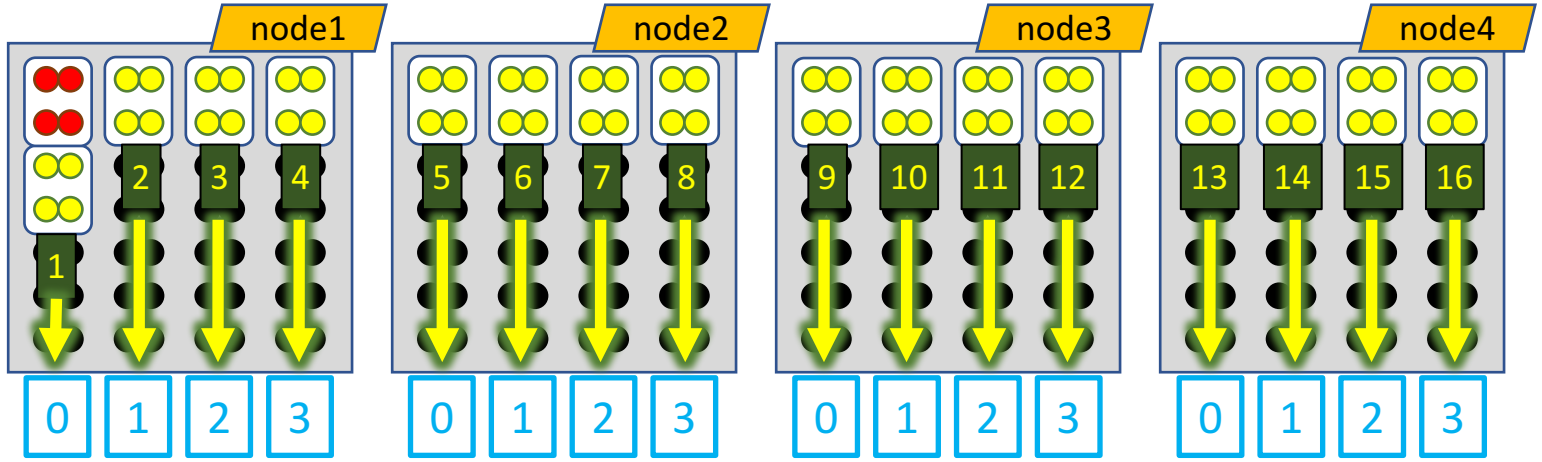
- Create $SLURM_HOSTFILE and unset $SLURM_NTASKS_PER_NODE

```
array=( $( scontrol show hostname $SLURM_JOB_NODELIST) )
file=$(mktemp)
echo ${array[0]} > $file
for ((i=0;i<${SLURM_JOB_NUM_NODES};i++)); do
  for ((j=0;j<$((SLURM_NTASKS_PER_NODE-1));j++)); do
    echo ${array[${i}]} >> $file
  done
done
export SLURM_HOSTFILE=$file
unset SLURM_NTASKS_PER_NODE
```

$SLURM_HOSTFILE

```
cn4000
cn4000
cn4000
cn4000
cn4000
cn4001
cn4001
cn4001
cn4001
cn4002
cn4002
cn4002
cn4002
cn4003
cn4003
cn4003
cn4003
```

Class2D, Class3D, Refine




```
sbatch
  --cpus-per-task=4
  --ntasks=20
  --ntasks-per-node=5
  --nodes=4
  --distribution=arbitrary
  --mem=240g
  --partition=gpu
  --gres=gpu:k80:4,lscratch:100
  --constraint=gpuk80
```

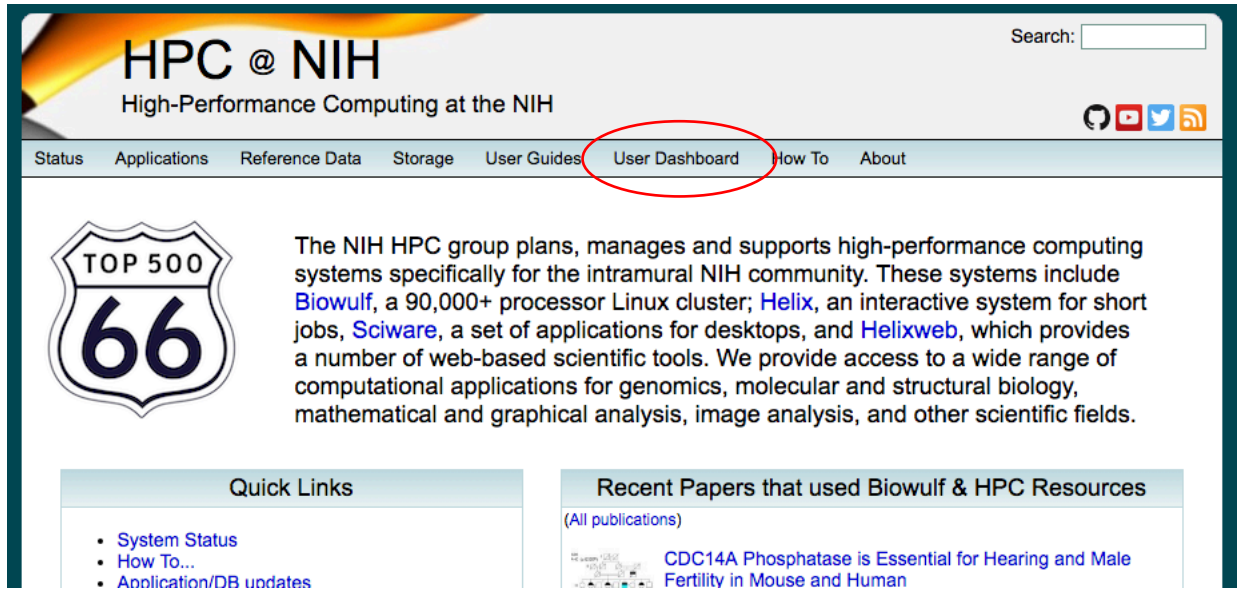
Tidbits

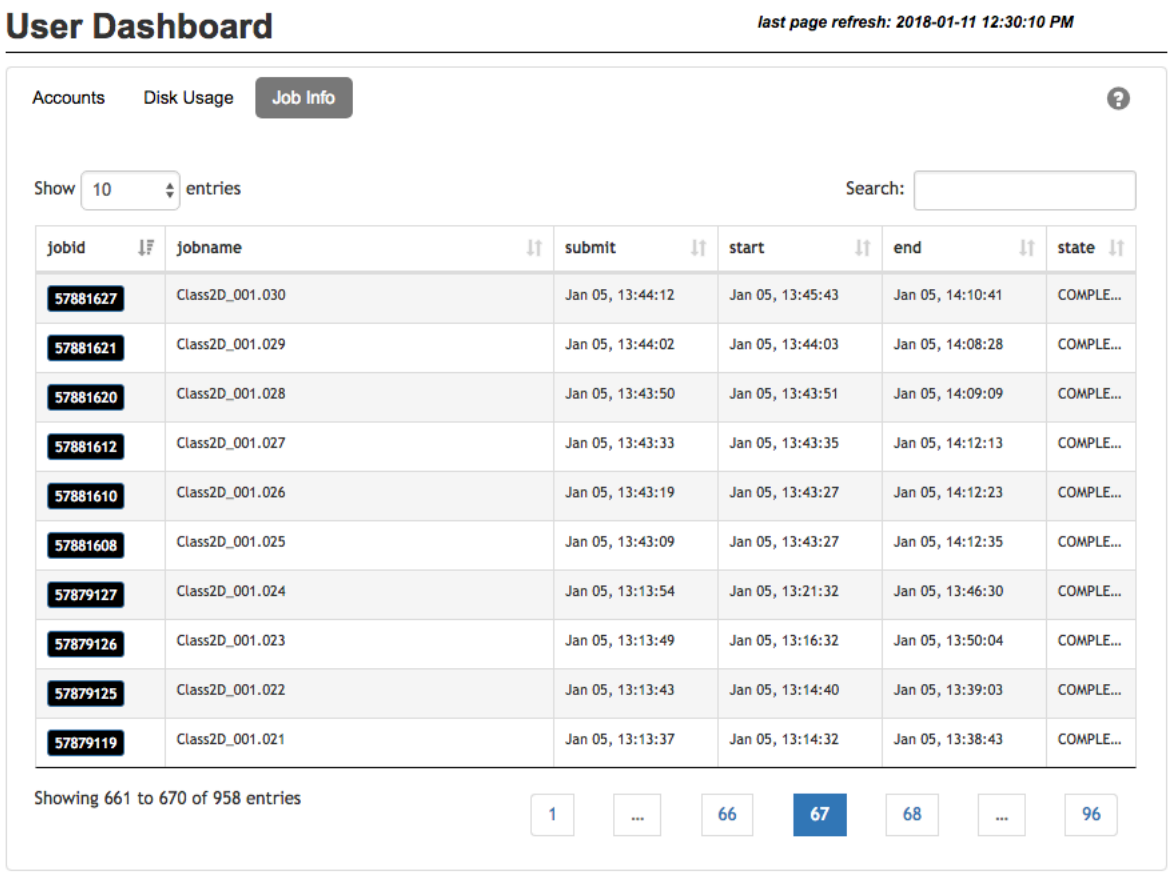
- Oversubscription of GPUs generally NOT a good idea
- Maintain 1 MPI rank per GPU, but 4 or 8 threads per GPU not always bad
- The more GPUs, the faster
- Run like Goldilocks

RELION Job Monitoring

- run.out and run.err files
- jobhist, jobload, jobdata
- dashboard
- nvidia-smi

Dashboard



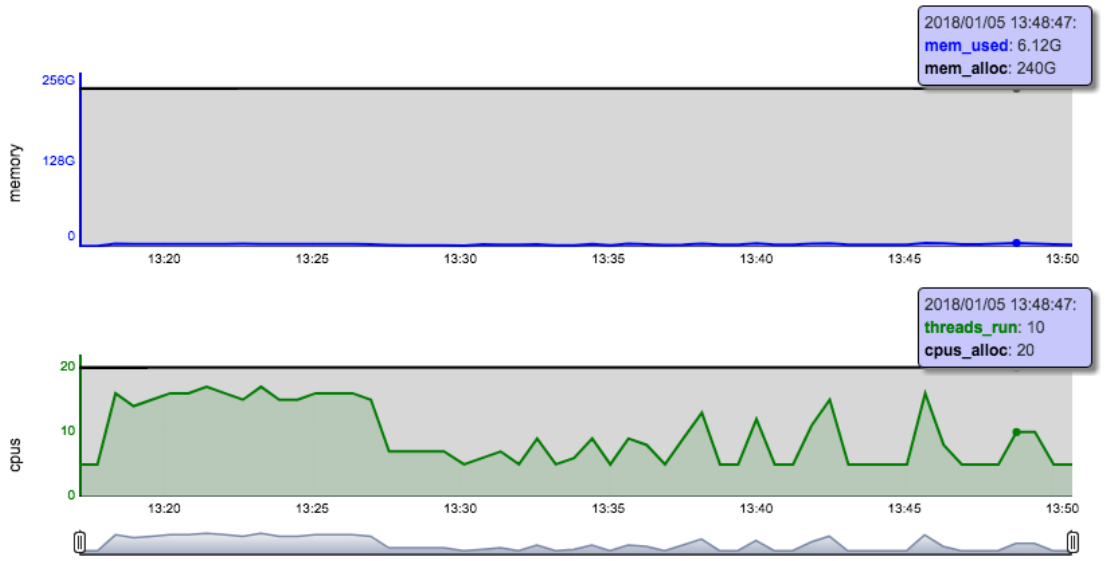


Jobs remain listed on the dashboard for 10 days after ending

Search by jobid, jobname, timestamp, state

Dashboard

Biowulf Job 57879126



jobname	Class2D_001.023
submitted	Jan 05, 13:13:49
state	COMPLETED
submission script	
work directory	/data/hooverdm/private/apps/RELION/benchmarks/Class2D
STDIN	/dev/null
STDOUT	/data/hooverdm/private/apps/RELION/benchmarks/Class2D/Class2D_001/023/run.out
STDERR	/data/hooverdm/private/apps/RELION/benchmarks/Class2D/Class2D_001/023/run.err
alloc nodes	4
total alloc CPUs	80

nodelist	cn4186,cn4187,cn4188,cn4189

ended	Jan 05, 13:50:04
CPU hours	44.711113
alloc mem per cpu	12.0 GB
max mem per cpu	296 MB
exit code	0

script

```
#!/bin/bash
mkdir /lscratch/${SLURM_JOB_ID}/TMPDIR
export TMPDIR=/lscratch/${SLURM_JOB_ID}/TMPDIR
export OMPI_MCA_btl="self,sm,tcp"
export OMPI_MCA_btl_tcp_if_include="10.2.0.0/18"
echo SLURM_JOB_ID=${SLURM_JOB_ID}
echo RELION=2.1.0
echo NT=20
echo CPT=4
echo TPN=5
echo SLURM_JOB_NUM_NODES=${SLURM_JOB_NUM_NODES}
echo GPN=4
echo date=$(date)

array=( $( scontrol show hostname $SLURM_JOB_NODELIST) )
file=$(mktemp)
echo ${array[0]} > $file
for ((i=0;i<${SLURM_JOB_NUM_NODES};i++)); do
  for ((j=0;j<$((SLURM_NTASKS_PER_NODE-1));j++)); do
    echo ${array[${i}]} >> $file
  done
done
export SLURM_HOSTFILE=$file

unset SLURM_NTASKS_PER_NODE
srun --mpi=pmi2 relion_refine_mpi --o Class2D_001/023/run --i Particles/tiny_1set.
star --pool 100 --ctf --iter 10 --tau2_fudge 2 --particle_diameter 360 --K 200 --f
latten_solvent --zero_mask --oversampling 1 --psi_step 6 --offset_range 5 --offset
_step 2 --norm --scale --j 4  --no_parallel_disc_io --scratch_dir /lscratch/${SLUR
M_JOB_ID}  --random_seed 0 --gpu

for i in $(ls Class2D_001/023/*.{star,mrcs}) ; do hollow_file.sh $i ; done

echo date=$(date)
```

nvidia-smi

- While job is running, display snapshot of GPU utilization

```
$ jobload -j 11111111
          JOBID            TIME            NODES  CPUS  THREADS   LOAD       MEMORY
                     Elapsed / Wall                Alloc   Active           Used /     Alloc
       11111111    00:52:53 /  1-00:00:00 cn4235    20       11    55%     4.0 /  240.0 GB
                   00:52:53 /  1-00:00:00 cn4236    16       15    94%     3.7 /  240.0 GB
                   00:52:53 /  1-00:00:00 cn4237    16       16   100%     3.6 /  240.0 GB
                   00:52:53 /  1-00:00:00 cn4238    16       14    88%     3.6 /  240.0 GB
                Nodes:    4    CPUs:   68  Load Avg:  84%
$ ssh cn4236 nvidia-smi
Thu Jan 11 12:39:16 2018
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 375.26                 Driver Version: 375.26                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K80           On   | 0000:83:00.0     Off |                  Off |
| N/A   78C    P0   137W / 149W |  11481MiB / 12205MiB |    100%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K80           On   | 0000:84:00.0     Off |                  Off |
| N/A   55C    P0   148W / 149W |  11481MiB / 12205MiB |     86%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla K80           On   | 0000:8A:00.0     Off |                  Off |
| N/A   79C    P0   120W / 149W |  11481MiB / 12205MiB |     89%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla K80           On   | 0000:8B:00.0     Off |                  Off |
| N/A   65C    P0   148W / 149W |  11481MiB / 12205MiB |     90%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|    0     37242    C   .../local/relion_2.1.0/bin/relion_refine_mpi 11477MiB |
|    1     37243    C   .../local/relion_2.1.0/bin/relion_refine_mpi 11477MiB |
|    2     37240    C   .../local/relion_2.1.0/bin/relion_refine_mpi 11477MiB |
|    3     37241    C   .../local/relion_2.1.0/bin/relion_refine_mpi 11477MiB |
+-----------------------------------------------------------------------------+
$
```

nvidia-smi dmon –d 10

- Continuously monitor GPU utilization for defined periods of time
- dmon is highly configurable

```
$ ssh cn4236 nvidia-smi dmon -d 10
# gpu   pwr  temp    sm   mem   enc   dec  mclk  pclk
# Idx     W     C     %     %     %     %   MHz   MHz
    0   146    77   100     4     0     0  2505   810
    1   143    54    15     1     0     0  2505   797
    2    70    79     6     0     0     0  2505   810
    3   147    65    12     1     0     0  2505   875
    0   143    77   100     1     0     0  2505   810
    1   148    55   100     1     0     0  2505   810
    2   144    80   100     2     0     0  2505   823
    3   125    66    44     2     0     0  2505   797
    0   146    78    64     4     0     0  2505   810
    1   148    55   100     1     0     0  2505   810
    2   142    80   100     2     0     0  2505   810
    3    87    66    81     4     0     0  2505   875
    0    87    78   100     1     0     0  2505   810
    1    87    55   100     4     0     0  2505   810
    2   141    80   100     2     0     0  2505   810
    3   145    67   100     4     0     0  2505   797
    0   145    78   100     4     0     0  2505   810
    1   148    56   100     2     0     0  2505   810
    2   141    80   100     2     0     0  2505   810
    3   147    67   100     4     0     0  2505   797
^C$ ^C
```

Troubleshooting

- RELION touches every weak and sensitive point in the cluster

- Many different ways things can go south

Insufficient Memory

- Insufficient memory causes individual MPI ranks to be killed, leading to "zombie" RELION jobs
- run.err shows something like this:

```
srun: error: cn3068: task 3: Killed
```

- Job continues to run, but run.out does not progress
- Other causes for rank failure include network hiccups, quota overrun, generic hardware failure

Insufficient GPU Memory

- K80 GPUs have access to 24GB VRAM

```
                                ***WARNING***

With the current settings and hardware, you will be able to
use an estimated image-size of 533 pixels during the last iteration...

...but your input box-size (image_size) is however 659. This means that
you will likely run out of memory on the GPU(s), and will have to then re-start
from the last completed iteration (i.e. continue from it) *without* the use of GPUs.

You are also splitting at least one GPU between two or more mpi-slaves, which
might be the limiting factor, since each mpi-slave that shares a GPU increases the
use of memory. In this case we recommend running a single mpi-slave per GPU, which
will still yield good performance and possibly a more stable execution.
```

- Downscale images
- Recompile in single-precision

Mixing RELION Versions

```
PipeLine::read: cannot find name or type in pipeline_nodes table
```

- The GUI creates hidden files for maintaining state between sessions under .Nodes
- Running a different version of RELION on top of .Nodes may cause corruption

```
$ tree .Nodes
.Nodes
├── 0
│   ├── Import
│   │   └── job001
│   │       └── movies.star
│   └── MotionCorr
│       └── job002
│           └── corrected_micrograph_movies.star
├── 1
│   ├── CtfFind
│   │   └── job003
│   │       └── micrographs_ctf.star
│   ├── ManualPick
│   │   └── job004
│   │       └── micrographs_selected.star
│   └── MotionCorr
│       └── job002
│           └── corrected_micrographs.star
├── 10
│   └── Refine3D
│       └── job012
│           └── run_half1_class001_unfil.mrc
├── 11
│   ├── LocalRes
│   │   └── first3dref
│   │       └── relion_locres_filtered.mrc
│   └── PostProcess
│       └── first3def
│           ├── postprocess_masked.mrc
│           └── postprocess.mrc
├── 12
│   └── LocalRes
│       └── first3dref
│           └── relion_locres.mrc
├── 13
│   ├── MotionCorr
│   │   └── job002
```

Inappropriate RELION Options

- Almost always diagnosable from .out and .err files
- Contact RELION developers or CCPEM mailing list if warning is unclear
- Mixing --preread_images and --scratch_dir
- Too many classes (--K)
- --write_fom_maps from within AutoPick

I/O

- Chronic problem nowadays
- Certain steps (copying to /lscratch, reading into memory) take inordinately long
- Don't launch 1000's of nodes simultaneously – stagger
- Use GPUs whenever possible
- ALWAYS SET --dont_combine_weights_via_disc

Bad Input

- Filter and prune your data!
- Choose only the best images, particles, and classes
- More data does NOT mean better results

Questions? Comments?

staff@hpc.nih.gov