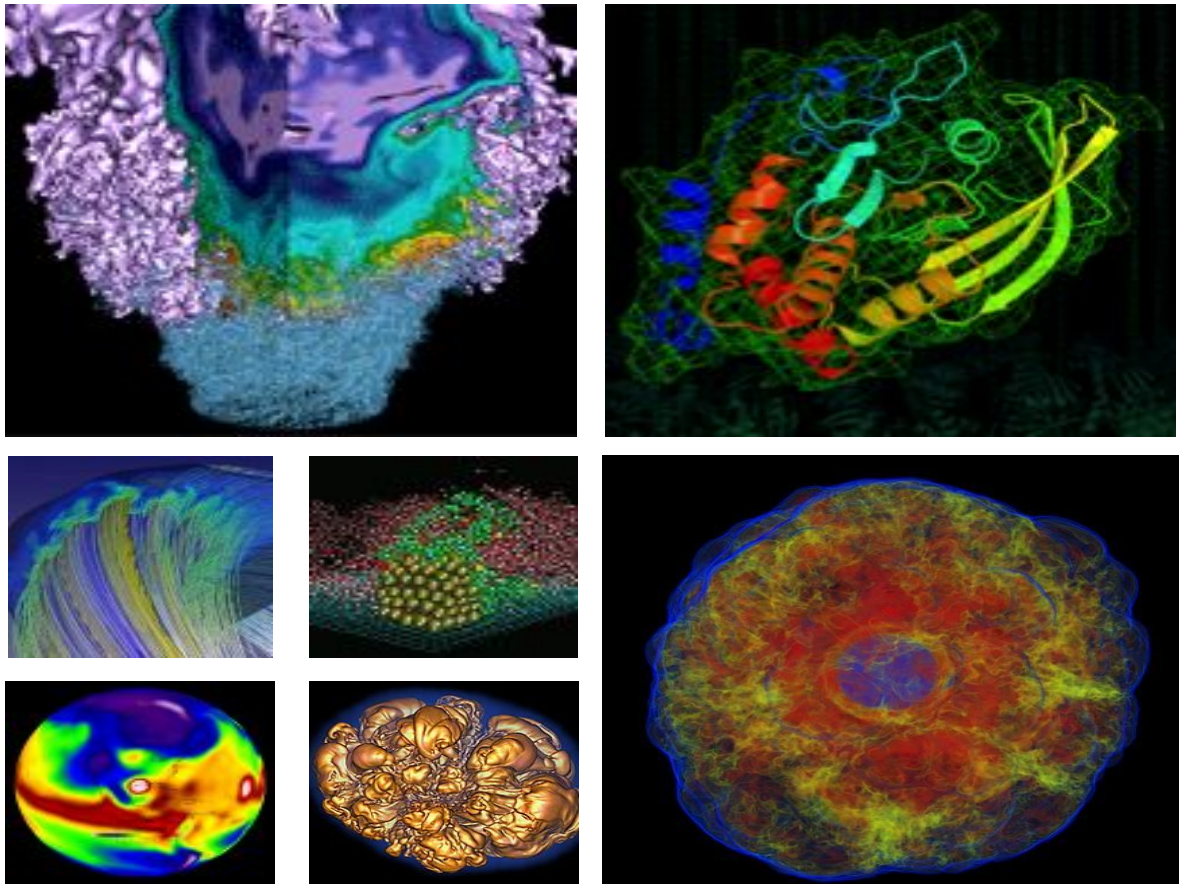
Running Jobs on Cori



Helen He

NERSC User Engagement Group

New User Training

June 21, 2019

Outline

- **Running jobs introductions**
- Batch script examples
- Advanced workflow options
- More on KNL with process/thread/memory affinity considerations
- Monitoring jobs

Jobs at NERSC

- **Most are parallel jobs (10s to 100,000+ cores)**
- **Also a number of "serial" jobs**
  - Typically "pleasantly parallel" simulation or data analysis
- **Production runs execute in batch mode**
- **Our batch scheduler is SLURM**
- **Debug jobs are supported for up to 30 min**
- **Batch interactive jobs are supported for up to 4 hrs**
- **Typical run times are a few to 10s of hours**
  - Limits are necessary because of MTBF and the need to accommodate 7,000 users' jobs

Login Nodes and Compute Nodes

- **Login nodes (external)**
  - Edit files, compile codes, submit batch jobs, etc.
  - Run short, serial utilities and applications
  - Cori has Haswell login nodes
- **Compute nodes**
  - Execute your application
  - Dedicated resources for your job
  - Cori has Haswell and KNL compute nodes
  - Binaries built for Haswell can run on KNL nodes, but not vice versa

Submitting Batch Jobs

- **To run a batch job on the compute nodes you must write a "batch script" that contains**
  - Directives to allow the system to schedule your job
  - An `srun` command that launches your parallel executable
- **Submit the job to the queuing system with the sbatch or salloc command**

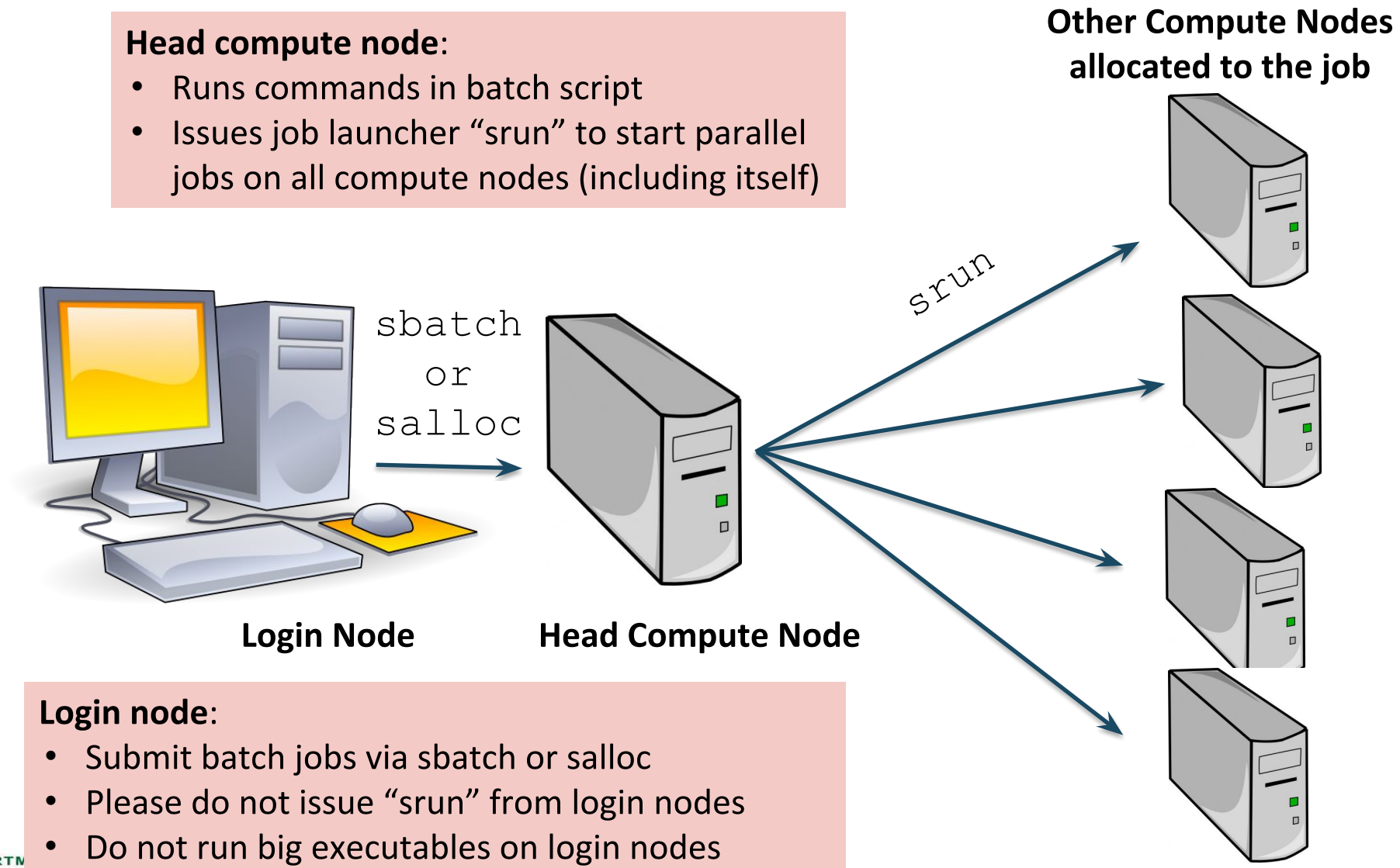
```
% sbatch my_batch_script
```

Or

```
% salloc <command line options>
```

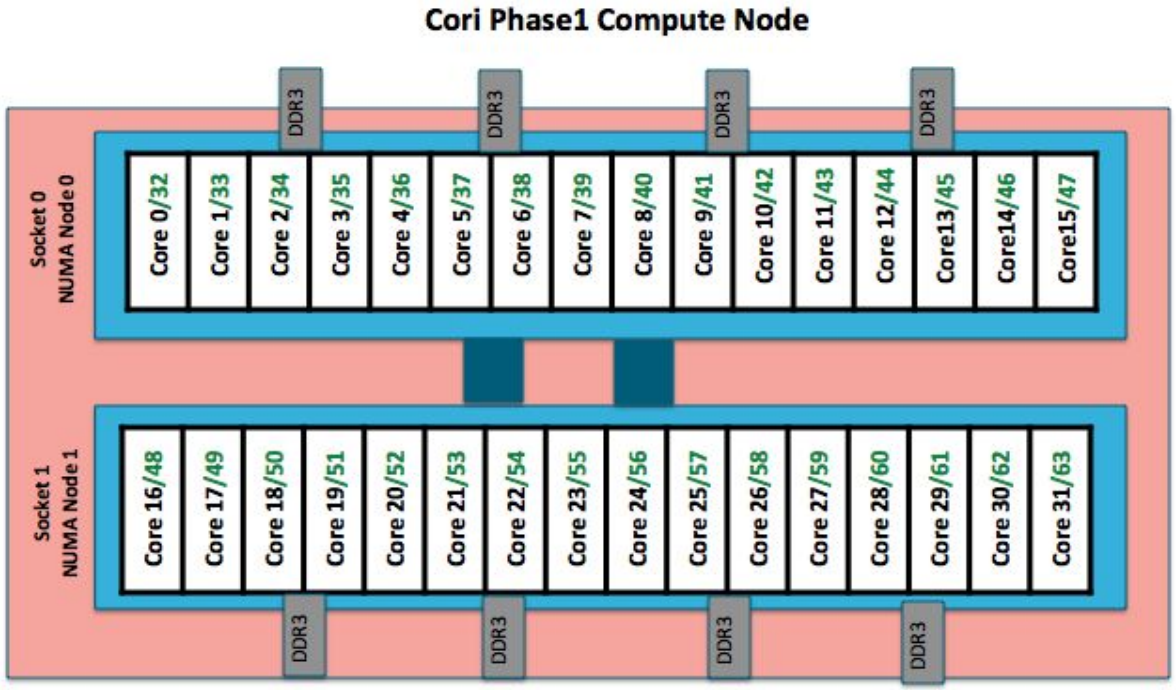
Launching Parallel Jobs with SLURM

Head compute node:
- Runs commands in batch script
- Issues job launcher "srun" to start parallel jobs on all compute nodes (including itself)



Login node:
- Submit batch jobs via sbatch or salloc
- Please do not issue "srun" from login nodes
- Do not run big executables on login nodes

Cori Haswell Compute Nodes



To obtain processor info:

Get on a compute node:
% salloc -N 1 -C …

Then:
% numactl -H
or % cat /proc/cpuinfo
or % hwloc-ls

- **Each Cori Haswell node has 2 Intel Xeon 16-core Haswell processors**
  - **2 NUMA domains (sockets) per node, 16 cores per NUMA domain. 2 hardware threads per physical core.**
  - **NUMA Domain 0: physical cores 0-15 (and logical cores 32-47);**
    NUMA Domain 1: physical cores 16-31 (and logical cores 48-63)
- **Memory bandwidth is non-homogeneous among NUMA domains**

Cori KNL Example Compute Nodes

- **A Cori KNL node has 68 cores/272 CPUs, 96 GB DDR memory, 16 GB high bandwidth on package memory (MCDRAM)**
- **Default mode is: quad, cache**

Arrangement of Hardware Threads for 68 Core KNL

Core # →	0	1	2	3	...	16	17	18	...	33	34	35	...	50	51	52	...	65	66	67
HW Thread #	0	1	2	3	...	16	17	18	...	33	34	35	...	50	51	52	...	65	66	67
	68	69	70	71	...	84	85	86	...	101	102	103	...	118	119	120	...	133	134	135
	136	137	138	139	...	152	153	154	...	169	170	171	...	186	187	188	...	201	202	203
	204	205	206	207	...	220	221	222	...	237	238	239	...	254	255	256	...	269	270	271

- A quad,cache node (default setting) has only 1 NUMA node with all CPUs on the NUMA node 0 (DDR memory). MCDRAM is hidden from the “numactl -H” result since it is a cache.

Other combinations are by reservation only

- A quad,flat node has only 2 NUMA nodes with all CPUs on the NUMA node 0 (DDR memory). NUMA node 1 has MCDRAM only
- A snc2,flat node has 4 NUMA domains with DDR memory and all CPUs on NUMA nodes 0 and 1

Outline

- Running jobs introductions
- **Batch script examples**
- Advanced workflow options
- More on KNL with process/thread/memory affinity considerations
- Monitoring jobs

Sample Cori Haswell Batch Script - MPI

```
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 4
#SBATCH -t 1:00:00
#SBATCH -C haswell
#SBATCH -L SCRATCH
#SBATCH -J myjob

export OMP_NUM_THREADS=1
srun -n 1280 -c 2 --cpu_bind=cores ./mycode.exe
```

- Need to specify which shell to use for batch script
- Use “-l” as login shell is optional
- Environment is automatically imported

Sample Cori Haswell Batch Script - MPI

```
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 4
#SBATCH -t 1:00:00
#SBATCH -C haswell
#SBATCH -L SCRATCH
#SBATCH -J myjob

export OMP_NUM_THREADS=1
srun -n 1280 -c 2 --cpu_bind=cores ./mycode.exe
```

Job directives: instructions for the batch system

- Submission QOS (default is “debug”)
- How many compute nodes to reserve for your job
- How long to reserve those nodes
- What type of compute nodes to use
- More optional SBATCH keywords

Sample Cori Haswell Batch Script - MPI

```
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 4
#SBATCH -t 1:00:00
#SBATCH -C haswell
#SBATCH -L SCRATCH
#SBATCH -J myjob

export OMP_NUM_THREADS=1
srun -n 1280 -c 2 --cpu_bind=cores ./mycode.exe
```

SBATCH optional keywords:

- What file systems my job depends on (prevent to start when there are file system issues)
- What to name my job
- What to name STDOUT files
- What account to charge
- Whether to notify you by email when your job finishes
- …

Sample Cori Haswell Batch Script - MPI

```
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 40
#SBATCH -t 1:00:00
#SBATCH -C haswell
#SBATCH -L SCRATCH
#SBATCH -J myjob

export OMP_NUM_THREADS=1
srun -n 1280 -c 2 --cpu_bind=cores ./mycode.exe
```

- No need to set this if your application programming model is pure MPI.
- If your code is hybrid MPI/OpenMP, or to prevent from using threaded libraries, set OMP_NUM_THREADS to 1 to run in pure MPI mode.

Sample Cori Haswell Batch Script - MPI

```
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 40
#SBATCH -t 1:00:00
#SBATCH -C haswell
#SBATCH -L SCRATCH
#SBATCH -J myjob

export OMP_NUM_THREADS=1
srun -n 1280 -c 2 --cpu_bind=cores ./mycode.exe
```

32 MPI tasks per node in this example

"srun" command launches parallel executables on the compute nodes

- srun flags overwrite SBATCH keywords
- No need to repeat flags in srun command if already defined in SBATCH keywords.

Sample Cori Haswell Batch Script - MPI

```
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 40
#SBATCH -t 1:00:00
#SBATCH -C haswell
#SBATCH -L SCRATCH
#SBATCH -J myjob

export OMP_NUM_THREADS=1
srun -n 1280 -c 2 --cpu_bind=cores ./mycode.exe
```

32 MPI tasks per node in this example

- There are 64 logical CPUs (the number Slurm sees) on each node
- "-c" specifies # logical CPUs to be allocated to each MPI task
- --cpu_bind is critical especially when nodes are not fully occupied
  - use "--cpu_bind=cores" when #MPI tasks <= # physical cores per node
  - use "--cpu_bind=threads" when #MPI tasks ># physical cores per node
- With 40 nodes, using hyperthreading, up to 40*64=2,560 MPI tasks can be launched: "srun -n 2560 -c 1 --cpu_bind=threads ./mycode.exe" is OK

Sample Cori Haswell Batch Script - Hybrid MPI/OpenMP

```
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 40
#SBATCH -t 1:00:00

export OMP_NUM_THREADS=8
export OMP_PROC_BIND=true
export OMP_PLACES=threads

srun -n 160 -c 16 --cpu_bind=cores ./mycode.exe
```

4 MPI tasks per node in this example

- Set OMP_NUM_THREADS
- Use OpenMP standard settings for process and thread affinity
- Again, “-c” specifies # logical CPUs to be allocated to each MPI task
  - with 4 MPI tasks per node on Haswell, set 64 logical CPUs /4 =16 for ”-c”
  - “-c” value should be >= OMP_NUM_THREADS

Use “shared” QOS to Run Serial Jobs

- **The “shared” QOS allows multiple executables from different users to share a node**
- **Each serial job run on a single core of a “shared” node**
- **Up to 32 (Cori Haswell) jobs from different users depending on their memory requirements**

```
#SBATCH -q shared
#SBATCH -t 1:00:00
#SBATCH --mem=4GB
#SBATCH -C haswell
#SBATCH -J my_job
./mycode.x
```

- Do not specify #SBATCH -N”
- Default “#SBATCH -n” is 1
- Default memory is 1,952 MB for Haswell
- Use -n or --mem to request more slots for larger memory
- **Do not use “srun” for serial executable (reduces overhead)**

- **Only available on Cori Haswell**
- **Small parallel job that use less than a full node can also run in the “shared” partition**
- **https://docs.nersc.gov/jobs/best-practices/#serial-jobs**

How to Run Debug and Interactive Jobs

- **You can run small parallel jobs interactively**
- **Debug**
  - Max 512 nodes, 30 min, run limit 2, submit limit 5

  % salloc -N 20 -q debug -C haswell -t 30:00
- **Interactive (highly recommend to use this!!)**
  - Instant allocation (get nodes in 5 min or reject), run limit 2, submit limit 2
  - max walltime 4 hrs, up to 64 nodes on Cori (Haswell and KNL) per repo

  % salloc -N 2 -q interactive -C knl,quad,cache -t 2:00:00

- https://docs.nersc.gov/jobs/examples/#interactive
- https://docs.nersc.gov/jobs/interactive/

Outline

- Running jobs introductions
- Batch script examples
- **Advanced workflow options**
- More on KNL with process/thread/memory affinity considerations
- Monitoring jobs

Advanced and Workflow Management

- Bundle jobs (multiple “srun”s in one script, sequentially or simultaneously)
- Use Job Arrays for submitting and managing collections of similar jobs
- Use job dependency features to chain jobs that have dependency
- Run variable-time jobs
- Use Burst Buffer for faster IO
- Use Shifter for jobs with custom user environment
- Use “xfer” for transferring to/from HPSS
- Use “bigmem” for large memory jobs

Bundle Jobs

Multiple Jobs Sequentially:

```
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 100
#SBATCH -t 12:00:00
#SBATCH -J my_job
#SBATCH -o my_job.o%j
#SBATCH -L project,SCRATCH
#SBATCH -C haswell

srun -n 3200 ./a.out
srun -n 3200 ./b.out
srun -n 3200 ./c.out
```

- Need to request largest number of nodes needed
- https://docs.nersc.gov/jobs/examples/#multiple-parallel-jobs-sequentially

Multiple Jobs Simultaneously:`

```
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 9
#SBATCH -t 12:00:00
#SBATCH -J my_job
#SBATCH -o my_job.o%j
#SBATCH -L project
#SBATCH -C haswell

srun -n 44 -N 2 -c2 --cpu_bind=cores ./a.out &
srun -n 108 -N 5 -c2 --cpu_bind=cores ./b.out &
srun -n 40 -N 2 -c2 --cpu_bind=cores ./c.out &
wait
```

- Need to request total number of nodes needed
- No applications are shared on the same nodes
- Make sure to use “&” (otherwise run in sequential) and “wait” (otherwise job exit immediately)
- https://docs.nersc.gov/jobs/examples/#multiple-parallel-jobs-simultaneously

Job Arrays

```
# Submit a job array with index values between 0 and 31:
$ sbatch --array=0-31 -N1
# Submit a job array with index values of 1, 3, 5 and 7:
$ sbatch --array=1,3,5,7 -N1 -n2
# Submit a job array with index values between 1 and 7 with a step size of 2 (i.e. 1, 3, 5, 7)
$ sbatch --array=1-7:2 -N1 -q regular
```

```
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 1
#SBATCH -t 00:05:00
#SBATCH --array=1-10
#SBATCH -L SCRATCH
#SBATCH -C haswell

cd test_$SLURM_ARRAY_JOB_ID
srun ./mycode.exe
```

- Better managing jobs, not necessary faster turnaround
- Each array task is considered a single job for scheduling
- Use $SLURM_ARRAY_JOB_ID for each individual array task

https://docs.nersc.gov/jobs/examples/#job-arrays

Dependency Jobs

```
cori% sbatch job1
Submitted batch job 165547

cori06% sbatch --dependency=afterok:5547 job2
or
cori06% sbatch --dependency=afterany:5547 job2
```

https://docs.nersc.gov/jobs/examples/#dependencies

```
cori06% sbatch job1
submitted batch job 165547

cori06% cat job2
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 1
#SBATCH -t 00:30:00
#SBATCH -d afterok:165547
#SBATCH -C haswell
srun -n 16 -c 4 ./a.out

cori06% sbatch job2
```

Variable-time Jobs

```
#!/bin/bash
#SBATCH -q regular
#SBATCH -C haswell
#SBATCH -N 2
#SBATCH --comment=96:00:00
#SBATCH --time-min=2:00:00
#SBATCH --time=48:00:00
#SBATCH --signal=B:USR1@60
#SBATCH –requeue

max_timelimit=48:00:00
ckpt_overhead=60
. /usr/common/software/variable-time-job/setup.sh
srun -n 8 -c 16 --cpu_bind=cores ../test.exe &
wait
```

- Allows to run multiple jobs with accumulated run time longer than max allowed wall time
- You may get run time longer than 2 hrs but shorter than 48 hrs at a time in this example.
- Job needs to have checkpoint/restart capability
- Individual jobs will be terminated with signal USR1 before time limit is reached
- Pre-terminated jobs will be requeued

https://docs.nersc.gov/jobs/examples/#variable-time-jobs

Use “flex” QOS to Run Variable-Time Jobs

- For user jobs that can produce useful work with a relatively short amount of run time before terminating, such as jobs capable of checkpointing and restarting where left off.
- Helps to improve throughput by submitting jobs that can fit into cracks in Slurm job scheduling
- Requires to use “--time-min” of <= 2hrs, max “--time” is 48 hrs
- 75% charging discount as of June 2019 (subject to change)
- Available for KNL only. More info at
  - https://docs.nersc.gov/jobs/examples/#using-the-flex-qos-for-charging-discount-for-variable-time-jobs-on-knl
  - https://docs.nersc.gov/jobs/policy/#flex

Use Burst Buffer for Faster IO

- Cori has 1.8PB of SSD-based "Burst Buffer" to support I/O intensive workloads
- Jobs can request a job-temporary BB filesystem, or a persistent (up to a few weeks) reservation
- More info at
  - http://www.nersc.gov/users/computational-systems/cori/burst-buffer/
  - https://docs.nersc.gov/jobs/examples/#burst-buffer
- See this afternoon's Burst Buffer talk for usage examples

Use Shifter for Custom Environment

- Shifter is a open-source software stack that enables users to run custom environments on HPC systems
- Compatible with the popular Docker container format so users can easily run Docker containers on NERSC systems
- More info at
  - https://docs.nersc.gov/development/shifter/how-to-use/
- See this afternoon's Shifter talk for usage examples

xfer Jobs

```
#!/bin/bash -l
#SBATCH -M escori
#SBATCH -q xfer
#SBATCH -t 12:00:00
#SBATCH -J my_transfer

#Archive run01 to HPSS
htar -cvf run01.tar run01
```

- Configured for the purpose of **staging data from HPSS before run or archive result to HPSS after run**
- Avoid large NERSC hours if done within large runs
- Runs on external login nodes, via Slurm Server ”escori”.
- Can submit jobs to the xfer QOS from inside another batch script:
  - Add to the end of batch script: “sbatch -M escori -q xfer myarchive.sl”

https://docs.nersc.gov/jobs/examples/#xfer-queue

bigmem Jobs

```
#!/bin/bash –l
#SBATCH -M escori
#SBATCH -q bigmem
#SBATCH -N 1
#SBATCH -t 01:00:00
#SBATCH -J my_big_job
#SBATCH -L SCRATCH
#SBATCH --mem=250GB
srun -N 1 -n 1 my_big_executable
```

- Runs on external login nodes, via Slurm Server “escori”
- https://docs.nersc.gov/jobs/examples/#large-memory

Outline

- Running jobs introductions
- Batch script examples
- Advanced workflow options
- **More on KNL with process/thread/memory affinity considerations**
- Monitoring jobs

Process / Thread / Memory Affinity

- **Correct process, thread and memory affinity is the basis for getting optimal performance on Cori Haswell and KNL. It is also essential for guiding further performance optimizations.**
  - Process Affinity: bind MPI tasks to CPUs
  - Thread Affinity: bind threads to CPUs allocated to its MPI process
  - Memory Affinity: allocate memory from specific NUMA domains
- **Our goal is to promote OpenMP standard settings for portability. For example, OMP_PROC_BIND and OMP_PLACES are preferred to Intel specific KMP_AFFINITY and KMP_PLACE_THREADS settings.**
- **https://docs.nersc.gov/jobs/affinity/**

“numactl -H” displays NUMA info

68-core KNL Quad Cache node:

NUMA Domain 0: all 68 cores (272 logic cores)

```
yunhe@cori01:> salloc -N 1 --qos=interactive -C knl,quad,cache -t 30:00
salloc: Granted job allocation 5291739

yunhe@nid02305:> numactl -H
available: 1 nodes (0)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83
84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147
148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177
178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207
208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237
238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267
268 269 270 271
node 0 size: 96762 MB
node 0 free: 93067 MB
node distances:
node   0
  0:  10
```

- The quad,cache mode has only 1 NUMA node with all CPUs on the NUMA node 0 (DDR memory)
- The MCDRAM is hidden from the numactl -H command (it is a cache).

Can We Just Do a Naïve Srun?

Example: 16 MPI tasks x 8 OpenMP threads per task on a single 68-core KNL quad,cache node:

% export OMP_NUM_THREADS=8

% export OMP_PROC_BIND=spread (other choice are “close”,”master”,”true”,”false”)

% export OMP_PLACES=threads (other choices are: cores, sockets, and various ways to specify explicit lists, etc.)

```
% srun -n 16  ./xthi |sort -k4n,6n
Hello from rank 0, thread 0, on nid02304. (core affinity = 0)
Hello from rank 0, thread 1, on nid02304. (core affinity = 144)      (on physical core 8)
Hello from rank 0, thread 2, on nid02304. (core affinity = 17)
Hello from rank 0, thread 3, on nid02304. (core affinity = 161)      (on physical core 25)
Hello from rank 0, thread 4, on nid02304. (core affinity = 34)
Hello from rank 0, thread 5, on nid02304. (core affinity = 178)      (on physical core 42)
Hello from rank 0, thread 6, on nid02304. (core affinity = 51)
Hello from rank 0, thread 7, on nid02304. (core affinity = 195)      (on physical core 59)
Hello from rank 1, thread 0, on nid02304. (core affinity = 0)
Hello from rank 1, thread 1, on nid02304. (core affinity = 144)
```

It is a mess!

Importance of -c and --cpu_bind Options

- **The reason: 68 is not divisible by #MPI tasks!**
  - Each MPI task is getting 68x4/#MPI tasks of logical cores as the domain size
  - MPI tasks are crossing tile boundaries
- **Set number of logical cores per MPI task (-c) manually by wasting extra 4 cores on purpose: 256/#MPI_tasks_per_node.**
  - Meaning to use 64 cores only on the 68-core KNL node, and spread the logical cores allocated to each MPI task evenly among these 64 cores.
  - Now it looks good!
  - % srun -n 16 -c 16 --cpu_bind=cores ./xthi

```
Hello from rank 0, thread 0, on nid09244. (core affinity = 0)
Hello from rank 0, thread 1, on nid09244. (core affinity = 136)      (on physical core 0)
Hello from rank 0, thread 2, on nid09244. (core affinity = 1)
Hello from rank 0, thread 3, on nid09244. (core affinity = 137)      (on physical core 1)
```

Now It Looks Good!

Process/thread affinity are good! (Marked first 6 and last MPI tasks only)

- MPI rank 0
- MPI rank 1
- MPI rank 2
- MPI rank 3
- MPI rank 4
- MPI rank 5
- ....
- MPI rank 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85
136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153
204																220	221
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
86																102	103
154	155	156	157	158	159	160	161									170	171
222																238	239
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51		
104																	
172																	
240																	
52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67		
120																	
188								196	197	198	199						
256																	

And so on for other MPI tasks and threads ….

Essential Runtime Settings for Process/Thread Affinity

- **Use srun -c and --cpu_bind flags to bind tasks to CPUs**
  - **-c <n> (or --cpus-per-task=n) allocates n CPUs per MPI task (process). It helps to evenly spread MPI tasks, can use up to n OpenMP threads per MPI task.**
  - Use **--cpu_bind=cores** (no hyperthreads) or **--cpu_bind=threads** (if hyperthreads are used)
- **Use OpenMP envs: OMP_PROC_BIND, OMP_PLACES to fine pin each thread to a subset of CPUs allocated to the host task**
  - Different compilers may have different implementations
  - The following provide compatible thread affinity among Intel, GNU and Cray compilers:

OMP_PROC_BIND=true # Specifying threads may not be moved between CPUs

OMP_PLACES=threads # Specifying a thread should be placed on a single CPU

Sample Job Script to Run on KNL quad,cache Nodes

Sample Job script (MPI+OpenMP)

```
#!/bin/bash -l
#SBATCH -N 2
#SBATCH -q regular
#SBATCH -t 1:00:00
#SBATCH -L SCRATCH
#SBATCH -C knl,quad,cache

export OMP_NUM_THREADS=4
srun -n 128 -c 4 --cpu_bind=cores ./a.out
```

This job script requests 2 KNL quad, cache nodes to run 128 MPI tasks, 64 MPI tasks per node, allocating 4 CPUs per task, and binds each task to the 4 CPUs allocated within each core. The 4 OpenMP threads per MPI task are bound to the 4 CPUs in the core.

Process and thread affinity

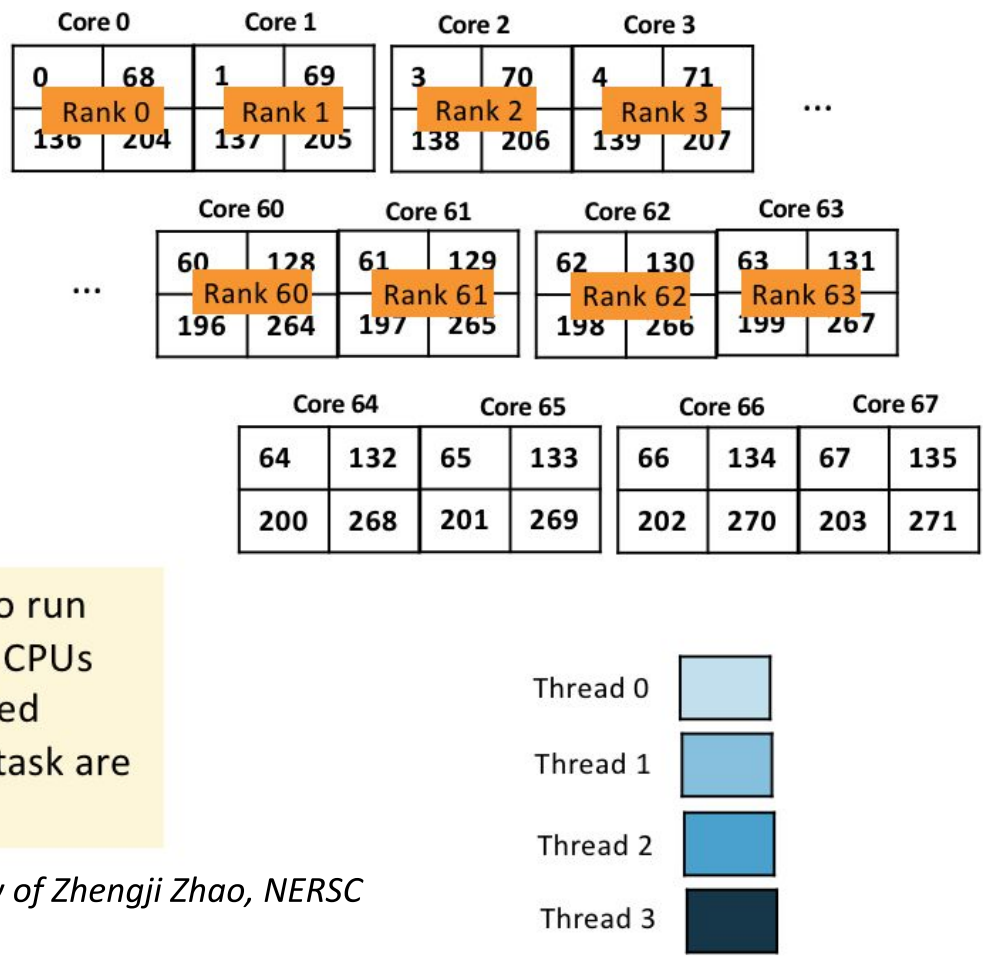


Illustration Courtesy of Zhengji Zhao, NERSC

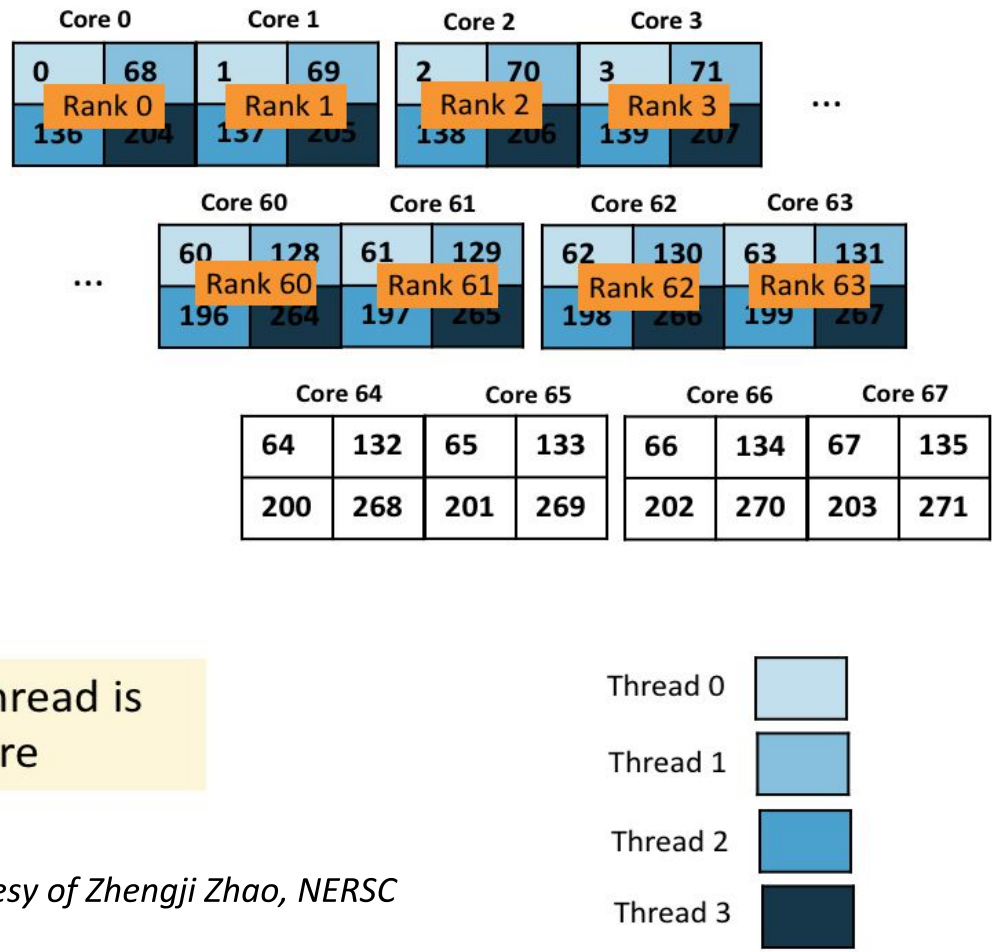
Sample Job Script to Run on KNL quad,cache Nodes

Sample Job script (MPI+OpenMP)

```
#!/bin/bash -l
#SBATCH -N 2
#SBATCH -q regular
#SBATCH -t 1:00:00
#SBATCH -L SCRATCH
#SBATCH -C knl,quad,cache

export OMP_PROC_BIND=true
export OMP_PLACES=threads
export OMP_NUM_THREADS=4
srun -n 128 -c 4 --cpu_bind=cores ./a.out
```

Process and thread affinity



With the above two OpenMP envs, each thread is now pinned to a single CPU within each core

Illustration Courtesy of Zhengji Zhao, NERSC

Affinity Verification Methods

- **NERSC has provided pre-built binaries from a Cray code (xthi.c) to display process thread affinity: check-mpi.intel.cori, check-mpi.cray.cori, check-hybrid.intel.cori, etc.**

```
% srun -n 32 -c 8 --cpu_bind=cores check-mpi.intel.cori | sort -nk 4
Hello from rank 0, on nid02305. (core affinity = 0,1,68,69,136,137,204,205)
Hello from rank 1, on nid02305. (core affinity = 2,3,70,71,138,139,206,207)
```

- **OpenMP 5.0 has OMP_DISPLAY_AFFINITY and OMP_AFFINITY_FORMAT**
  - Available in Intel compiler >= 18.0.5, gcc >= 9.0, and CCE >= 9.0.0

```
% export OMP_DISPLAY_AFFINITY=true
% export OMP_AFFINITY_FORMAT="host=%H, pid=%P, thread_num=%n, thread affinity=%A”
host=nid02496, pid=150147, thread_num=0, thread affinity=0
host=nid02496, pid=150147, thread_num=1, thread affinity=4
```


U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC Job Script Generator

https://my.nersc.gov/script_generator.php

- yunhe
- Dashboard
- Jobs
  - Jobscript Generator
  - Completed Jobs
  - Cori Queues
  - Edison Queues
  - PDSF Queues
  - Queue Backlog
  - Job Completion Stats
- Center Status
- File Browser
- My Tickets
- Data Dashboard

Jobscript Generator

Job Information

This tool generates a batch script template which also realizes specific process and thread binding configurations.

Machine

Select the machine on which you want to submit your job.

Cori - KNL

Application Name

Specify your application including the full path.

myapp.x

Job Name

Specify a name for your job.

mytest_KNL

Email Address

Specify your email address to get notified when the job enters a certain state.

```
#!/bin/bash
#SBATCH -N 150
#SBATCH -C knl
#SBATCH -q regular
#SBATCH -J mytest_KNL
#SBATCH -t 02:30:00

#OpenMP settings:
export OMP_NUM_THREADS=8
export OMP_PLACES=threads
export OMP_PROC_BIND=spread


#run the application:
srun -n 1200 -c 32 --cpu_bind=cores myapp.x
```

Outline

- Running jobs introductions
- Batch script examples
- Advanced workflow options
- More on KNL with process/thread/memory affinity considerations
- **Monitoring jobs**

Monitoring Your Job

- **Once your job is submitted, it enters the queue and will start when resources are available**
- **Overall job priorities are a combination of QOS, queue wait time, job size, wall time request (and fair share).**
- **You can monitor it with:**
  - `sqs`
  - `squeue`

  On the web:

  https://my.nersc.gov

  https://www.nersc.gov/users/live-status/ □ "Queue Look"

  https://www.nersc.gov/users/job-logs-and-analytics/completed-jobs/

SLURM User Commands

- **sbatch:** submit a batch script
- **salloc:** request nodes for an interactive batch session
- **srun:** launch parallel jobs
- **scancel:** delete a batch job
- **sqs**: NERSC custom queue display with job priority ranking info
- **squeue:** display info about jobs in the queue
- **sinfo:** view SLURM configuration about nodes and partitions
- **scontrol**: view and modify SLURM configuration and job state
- **sacct:** display accounting data for jobs and job steps
- https://docs.nersc.gov/jobs/#submitting-jobs
- https://docs.nersc.gov/jobs/#monitoring-jobs

sqs: NERSC Custom Batch Queue Script

```
yunhe@cori05:~> sqs -u fxxxx |more
JOBID      ST  USER    NAME       NODES REQUESTED USED  SUBMIT               QOS        SCHEDULED_START      FEATURES REASON
110901xx   PD  fxxxx   mxxx       1536    5:00    0:00  2018-03-20T10:49:23  regular_0  2018-03-22T06:30:00  haswell  Resources
110901xx   PD  fxxxx   run.xxx*   1537   20:00    0:00  2018-03-20T10:51:03  regular_0  2018-03-22T06:30:00  haswell  Resources
110823xx   PD  fxxxx   gxxx        300   30:00    0:00  2018-03-19T23:05:24  regular_1  avail_in_~1.6_days   haswell  Priority
110823xx   PD  fxxxx   run-xx      768   20:00    0:00  2018-03-19T23:05:33  regular_1  avail_in_~1.6_days   haswell  Priority
110901xx   PD  fxxxx   Jxxx        768   10:00    0:00  2018-03-20T10:51:33  regular_1  avail_in_~2.1_days   haswell  Priority
110921xx   PD  fxxxx   pxxxxx-xx*  150   30:00    0:00  2018-03-20T12:42:38  regular_1  avail_in_~2.1_days   haswell  Priority
111046xx   PD  fxxxx   hxxxxx       32   20:00    0:00  2018-03-21T01:20:09  regular_1  avail_in_~2.7_days   haswell  Priority
110823xx   PD  fxxxx   rxxxx      1536   20:00    0:00  2018-03-19T23:05:04  regular_0  N/A                  haswell  JobHeldUser
110823xx   PD  fxxxx   axxxxxxxx* 1536   30:00    0:00  2018-03-19T23:05:16  regular_0  N/A                  haswell  JobHeldUser
111152xx   PD  fxxxx   run.xxx     769 2:00:00    0:00  2018-03-21T09:39:29  regular_1  avail_in_~3.0_days   knl&quad&cache  None
<omitted...>

==========================================================================================
Note: The SCHEDULED_START column shows either the estimated start time provided by Slurm,
    or that an estimated start time will be "available in xxx hrs/days", meaning your job
    won't be scheduled by then (due to priority) unless via backfill, and the actual job
    start time could still be some period of time after then.

Second Note: Upcoming system maintenance on 03-22-2018 07:00 AM, SCHEDULED_START will be
        updated after the maintenance is completed.
==========================================================================================
```

- sqs combines information from squeue and sinfo, provides formatted output with various options
- It has many options, such as no running jobs, yes or no for a specific qos, wider with more fields, compact format, custom fields, expand array jobs, csv and json output, etc. See “sqs --help” or “man sqs” for more details.
- The scheduled_start gives users an estimate for when their jobs will start (or when will be considered for scheduling based on distance to priority threshold).

sinfo: Show Available Nodes

- % sinfo --format="%.20F %.20b" for features of available nodes

A/I/O/T (allocated/idle/other/total)

```
yunhe@cori11:~> sinfo -o "%.20F %.20b"
       NODES(A/I/O/T)     ACTIVE_FEATURES
                0/0/3/3                 knl
  7972/1702/11/9685       knl,cache,quad
    2030/351/7/2388              haswell
```

all these are knl,quad,cache nodes

scontrol: Show Job Details

- % scontrol show job <jobid> for details of a job

```
yunhe@cori11:~> scontrol show job 18193352
JobId=18193352 JobName=mpi4py-import-cori-haswell-common-150
   UserId=fbench(42034) GroupId=fbench(42034) MCS_label=N/A
   Priority=67065 Nice=0 Account=nstaff QOS=regular_1
   JobState=PENDING Reason=Priority Dependency=(null)
   Requeue=0 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:00 TimeLimit=00:30:00 TimeMin=N/A
   SubmitTime=2019-01-23T12:42:07 EligibleTime=2019-01-23T12:42:07
   AccrueTime=2019-01-23T12:42:07
   StartTime=Unknown EndTime=Unknown Deadline=N/A
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   LastSchedEval=2019-01-25T02:28:44
   Partition=regular AllocNode:Sid=cori03:31944
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=(null)
   NumNodes=150-150 NumCPUs=4800 NumTasks=4800 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=4800,node=150,billing=4800
   Socks/Node=* NtasksPerN:B:S:C=32:0:*:* CoreSpec=*
   MinCPUsNode=32 MinMemoryNode=0 MinTmpDiskNode=0
   Features=haswell DelayBoot=00:00:00
   OverSubscribe=NO Contiguous=0 Licenses=cscratch1:1 Network=(null)
   Command=/global/cscratch1/sd/fbench/nersc-python-bench/scripts/mpi4py-import-cori-haswell-common-150.sh
   WorkDir=/global/cscratch1/sd/fbench/nersc-python-bench/scripts
   StdErr=/global/cscratch1/sd/fbench/nersc-python-bench/scripts/logs/mpi4py-import-cori-haswell-common-150-18193352.out
   StdIn=/dev/null
  StdOut=/global/cscratch1/sd/fbench/nersc-python-bench/scripts/logs/mpi4py-import-cori-haswell-common-150-18193352.out
   Power=
   TresPerNode=craynetwork:1
```

sacct: Query Completed and Pending Jobs Information

```
yunhe@cori02:~> sacct -u fbench -S 2019-01-09 -E 2019-01-10 -o user,jobid,start,end,elapsed,timelimit,nnodes,exitcode,
state -X|more
     User        JobID               Start                 End    Elapsed  Timelimit   NNodes ExitCode      State
--------- ------------ ------------------- ------------------- ---------- ---------- -------- -------- ----------
   fbench 17708834     2019-01-09T06:19:41 2019-01-09T06:34:32   00:14:51   00:45:00      769      0:0  COMPLETED
   fbench 17708835     2019-01-09T06:39:40 2019-01-09T06:40:40   00:01:00   00:10:00      768      0:0  COMPLETED
   fbench 17708836     2019-01-09T03:09:28 2019-01-09T03:22:41   00:13:13   00:30:00      384      0:0  COMPLETED
   fbench 17708838     2019-01-09T02:09:38 2019-01-09T02:38:32   00:28:54   00:30:00       47      1:0     FAILED
   fbench 17708839     2019-01-09T03:03:47 2019-01-09T03:42:48   00:39:01   02:00:00      150      0:0  COMPLETED
   fbench 17708840     2019-01-09T06:43:42 2019-01-09T06:49:42   00:06:00   00:30:00      768      0:0  COMPLETED
   fbench 17708841     2019-01-09T06:55:25 2019-01-09T07:00:23   00:04:58   00:15:00      768      0:0  COMPLETED
   fbench 17711327     2019-01-09T00:39:49 2019-01-09T00:52:14   00:12:25   00:30:00        3      0:0  COMPLETED
   fbench 17711328     2019-01-09T00:55:08 2019-01-09T01:25:31   00:30:23   00:30:00        3      0:0    TIMEOUT
```

- **Maximum query duration is one month (subject to change)**
- **Detailed job steps info will be displayed without "-X" flag**
- **Many more job fields can be queried. See "man sacct"**

How Your Jobs Are Charged (1)

- **Unit: NERSC Hours**
- **Each architecture has a base charge per node hour used:**
  - Cori Haswell: 90
  - Cori KNL: 90
- **Modification to base charge by QOS used:**
  - premium: 2.0
  - normal: 1.0 (default)
  - low: 0.75
  - flex: 0.25
  - scavenger: 0
  - shared: fraction of the node used
- **On Cori KNL:**
  - Jobs requesting 1024 or more nodes get a 20% discount

How Your Jobs Are Charged (2)

- **Your repository is charged for each node your job was allocated for the entire duration of your job.**
  - The minimum allocatable unit is a node (*except for the "shared" QOS*).
  - Example: 4 Cori Haswell nodes, run for 1 hour with "premium" QOS
    NERSC hours = 4 * 1 hour * 90 * 2 = 720
  - "shared" jobs are charged with # of physical CPUs used instead of entire node.
- **If you have access to multiple repos, pick which one to charge in your batch script**

```
#SBATCH -A repo_name
```

Cori Haswell Queue Policy (as of June 2019)

QOS	Max nodes	Max time (hrs)	Submit limit	Run limit	Priority	Charge
regular	1932	48	5000	-	4	90
shared [1]	0.5	48	10000	-	4	90
interactive [4]	64	4	2	2	-	90
debug	64	0.5	5	2	3	90
premium	1772	48	5	-	2	180
scavenger [2]	1772	48	5000	-	5	0
xfer	1 (login)	48	100	15	-	0
bigmem	1 (login)	72	100	1	-	90
realtime [3]	custom	custom	custom	custom	1	custom
special [5]	custom	custom	custom	custom	-	custom

Cori KNL Queue Policy (as of June 2019)

QOS	Max nodes	Max time (hrs)	Submit limit	Run limit	Priority	Charge
regular	9489	48	5000	-	4	90 [6]
interactive [4]	64	4	2	2	-	90
debug	512	0.5	5	2	3	90
premium	9489	48	5	-	2	180 [6]
low	9489	48	5000	-	5	67.5 [7]
flex	256	48	5000	-	6	22.5 [8]
scavenger [2]	9489	48	5000	-	7	0
special [5]	custom	custom	custom	custom	-	custom

Where to Run My Jobs

- **Queue configuration and policies are subject to further tuning for max throughput and system utilization**
- **Factors to consider: queue wait time (much shorter on KNL than on Haswell), throughput, charging, code readiness on KNL**
  - Cori Haswell (also known as the "Cori Data Partition") system is designed to accelerate data-intensive applications; 2388 total compute nodes
  - Cori KNL: Large capability and performance; 9688 total compute nodes
  - Jobs use 1024+ nodes on Cori KNL get 20% charging discount
  - “flex” and “low” qos with discounts only available on Cori KNL
  - “shared” and ”realtime” available on Cori Haswell only
  - “interactive” nodes available on Cori Haswell and KNL; 192 nodes each
  - “bigmem” and “xfer” available on Cori (run on external login nodes)
- **Remember to compile separately for each type of compute nodes**

Tips for Getting Better Throughput

- **Line jumping is allowed, but it may cost more (with "premium" QOS)**
- **Submit shorter jobs, they are easier to schedule**
  - Checkpoint to break up long jobs, use variable time
  - Short jobs can take advantage of 'backfill' opportunities
  - Run short jobs just before maintenance
  - Run variable-time jobs; use "flex" QOS
- **Make sure the wall clock time you request is accurate**
  - Larger shorter jobs are easier to schedule than long smaller jobs
  - Many users unnecessarily enter the largest wall clock time possible as default
- **Queue wait time statistics**
  - https://my.nersc.gov/queuewaittimes.php
  - https://www.nersc.gov/users/queues/queue-wait-times/

More Information

NERSC Web pages:
https://docs.nersc.gov/jobs/

Contact NERSC Consulting:

- File a service ticket via Help Portal
  - https://help.nersc.gov
- Talk to a consultant
  - Toll-free 800-666-3772, option #3 or 510-486-8611

NERSC 40 YEARS at the FOREFRONT

Thank you.