INTRODUCTION TO

NCCS

lunchtime series

December, 2013

another one of many lunchtime talks...
György (George) Fekete

What is SLURM

- Resource manager
  - responsible to allocate resources within a cluster
  - What are resources?
    - CPUs
    - memory
    - time
    - GPUs
- Scheduler (limited resources for a lot of work)
  - manages *partitions* (formerly called *queues*)

NCCS History

- Used PBS for at least 10 years
- 2013: Time to reevaluate
  - cost
  - performance
    - scalability
    - “smarts”
    - maintainability
- SLURM – good company: 5 of top 10 in Top 500 list
  - Tianhe-2
    - 16,000 nodes (2 IvyBridge + 3 Phi) = 3.1 Million cores (33.86 Pf/s)
  - Sequoia (LLNL)
    - 98,304 nodes = 1.6 Million cores (17.17 Pf/s)

top500.org

SLURM Endorsements

- Swiss National Supercomputing Centre
- Rensselaer Polytechnic Institute
- EPA
- Lawrence Livermore NL
- Pervasive Technology Labs, Indiana Univ.
- Oak Ridge NL
- *Barcelona Supercomputing Center*
- University of Karlsruhe
- Hewlett Packard

Barcelona Supercomputer Center

What we like about SLURM

- Scalable
  - can handle millions of cores
- Performance
  - accepts over 1,000 job submissions / s
  - execute 500 simple jobs /s
- Flexible
  - add functionality through plugins
    - authentication
    - schedulers

SLURM is Simple Linux Utility for Resource Management

Provides three (3) key functions

1. allocate and provide access to resources
   - CPUs
   - memory
   - time
2. job control
   - setup
   - execute
   - monitor
3. arbitrates access to resources
   - managing partitions (formerly known as queues under PBS)

Where are we?

- Completed initial migration from PBS Pro to SLURM
  - Attempted to make migration as transparent as possible
  - (May not have been completely successful)
- Ready to begin leveraging native SLURM capabilities
- This is the first in a series of talks on native SLURM

Glossary

Term	Description
resource	a mix of CPUs, memory and time
partition	a virtual combination of resources (*queue under PBS*)
batch job	a chain of commands in a script file
interactive session	an allocated resource on which you are the only user
success	a job completes and terminates well (with exit status 0) (cancelled jobs are not considered successful)
failure	anything that lacks success

node	contains one or more sockets
socket	contains one processor
processor	contains one or more cores
core	CPU, performs FLOPs

Discover cluster nodes (Westmere)

2GB/core; 24GB total (shared by all tasks on all CPUs)

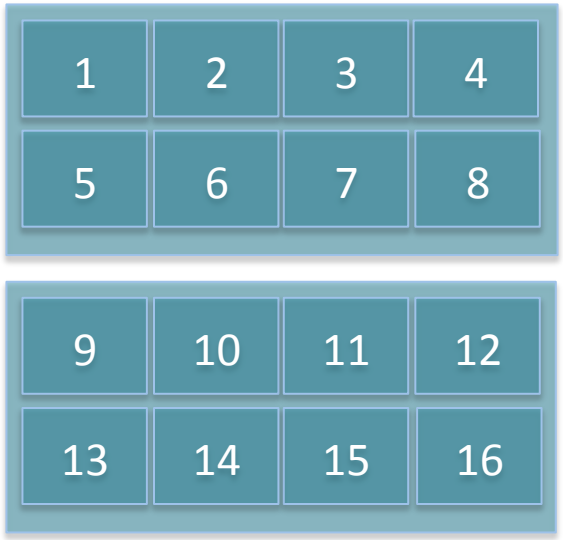
12 tasks
1 task per CPU — 2GB / task

1 task
11 CPUs unused — 24GB / task

4 tasks
8 CPUS unused — 6GB / task

etc.

Discover cluster nodes (Sandy Bridge)



2GB/core; 32GB total (shared by all tasks on all CPUs)

16 tasks
1 task per CPU — 2GB / task

1 task
15 CPUs unused — 32GB / task

4 tasks
12 CPUS unused — 8GB / task

etc.

In reality you get a bit over 30GB as you loose some to the system

Normally, you want to run 1 task on 1 CPU

Resource request for parallel tasks

request for resources needs to be aware of CPUs in the context of required (therefore requested) memory

7 tasks

1 Westmere
24 GB shared among 7 tasks
3.4 GB per task

7 tasks, 4GB per CPU

requires 28GB
2 Westmeres

-or-

1 Sandy Bridge

show you SLURM directives in about 10 slides

Coming up next:

- Brief introduction to some SLURM commands
- SLURM is very powerful, we focus on typical use patterns at NCCS
- Example use

SLURM commands (partial list)

Task	Command(s)	Notes
submit a job	sbatch	batch mode submits a job for later execution
interactive jobs	salloc	allocates resources and attaches a shell for interactive use
start a job right away	srun	launches a job and creates an allocation if needed
details about job	sattach	connect stdin, stdout, stderr for an existing job

SLURM commands (partial list)

Task	Command(s)	Notes
cancel/delete a submitted job	scancel	
check job state	squeue	see if pending, running, failed, completed, etc...
details about job or partition	scontrol	detailed information about a job or a partition *too much information, use with care*
nodes, partitions	sinfo	view information about nodes and partitions

SLURM commands: general info

man pages available for all commands

```
$ export MANPATH=$MANPATH:/usr/slurm/share/man
% setenv MANPATH=$MANPATH:/usr/slurm/share/man
```

`--help` print brief description

`--usage` print list of options

most commands have options in two formats:

single letter	`-p debug`	
verbose	`--partition=debug`	
time format:	`days-hours:minutes:seconds`	`1-12:00:00 (36 hours)`
	`minutes`	`5 (5 minutes)`

non-zero exit code indicates failure

SLURM commands: general info

man pages available for all commands

```
$ export MANPATH=$MANPATH:/usr/slurm/share/man
% setenv MANPATH=$MANPATH:/usr/slurm/share/man
```

`--help` print brief description

`--usage` print list of options

Note the single and double dash!

most commands have options in two formats:

single letter `-p deb`

verbose `--partition=debug`

time format: `days-hours:minutes:seconds` `1-12:00:00 (36 hours)`

`minutes` `5 (5 minutes)`

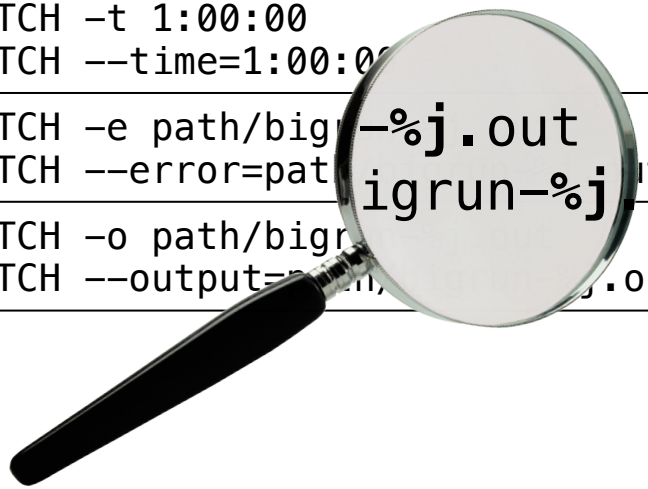
non-zero exit code indicates failure

Directives: PBS and SLURM

What	PBS	SLURM
Name the job	`#PBS –N jobname`	`#SBATCH -J jobname` `#SBATCH --job-name=jobname`
Charge to account	`#PBS –W group_list=s1234`	`#SBATCH –A s1234` `#SBATCH --account=s1234`
Pick a queue(PBS) or partition (SLURM)	`#PBS –q general`	`#SBATCH –p general` `#SBATCH --partition=general`
Request 2 nodes	`#PBS –l select=2`	`#SBATCH –N 2` `#SBATCH --nodes=2`
Request one hour	`#PBS -l walltime=1:00:00`	`#SBATCH -t 1:00:00` `#SBATCH --time=1:00:00`
Send error output	`#PBS -e path`	`#SBATCH -e path/bigrun-%j.out` `#SBATCH --error=path/bigrun-%j.out`
Send stdoutput to a file	`#PBS -o path`	`#SBATCH -o path/bigrun-%j.out` `#SBATCH --output=path/bigrun-%j.out`

Directives: PBS and SLURM

What	PBS	SLURM
Name the job	`#PBS –N jobname`	`#SBATCH –J jobname` `#SBATCH --job-name=jobname`
Charge to account	`#PBS –W group_list=s1234`	`#SBATCH –A s1234` `#SBATCH --account=s1234`
Pick a queue(PBS) or partition (SLURM)	`#PBS –q general`	`#SBATCH –p general` `#SBATCH --partition=general`
Request 2 nodes	`#PBS –l select=2`	`#SBATCH –N 2` `#SBATCH --nodes=2`
Request one hour	`#PBS –l walltime=1:00:00`	`#SBATCH –t 1:00:00` `#SBATCH --time=1:00:0`
Send error output	`#PBS –e path`	`#SBATCH –e path/big` `#SBATCH --error=pat`
Send stdoutput to a file	`#PBS –o path`	`#SBATCH –o path/bigr` `#SBATCH --output=`



%j is replaced with job ID

Directives: PBS and SLURM

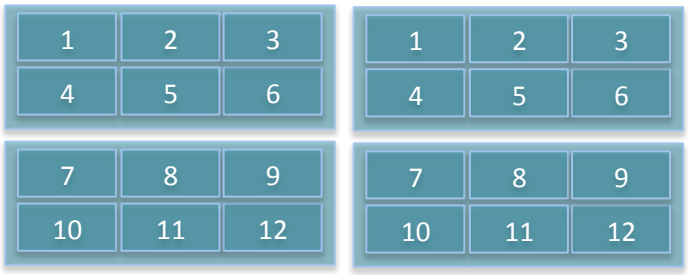
What	PBS	SLURM
Set E-mail recipient	`#PBS -M me@nasa.gov`	`#SBATCH --mail-user=me@nasa.gov`
E-Mail when job completes (fails, starts)	`#PBS -m e (a, b)`	`#SBATCH --mail-type=END (FAIL, BEGIN)`
change to directory from where job was submitted	`cd $PBS_O_WORKDIR`	*nothing. default action for SLURM*

Some command-line directives

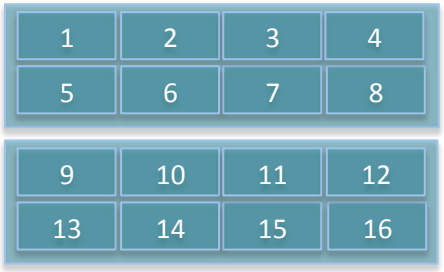
common	
memory needed per CPU (per task, assuming 1 task per CPU)	`--mem-per-cpu=`
number of tasks	`--ntasks=`
number of tasks per node	`--ntasks-per-node`

less common (but available)	
number of tasks per core (usually 1)	`--ntasks-per-core=`
cpus per task	`--cpus-per-task=`
memory required	`--mem=`

Resource request for parallel tasks



2 Westmeres



1 Sandy Bridge

7 tasks

```
--ntasks=7
```

1 Westmere
24 GB shared among 7 tasks
3.4 GB per task

7 tasks, 4GB per CPU

```
--ntasks=7 --mem-per-cpu=4G
```

requires 28GB
2 Westmeres

-or-

```
--ntasks=7 --mem-per-cpu=4G --constraint=sand
```

1 Sandy Bridge

Submit a simple job with `sbatch`

- Hello world job
  - task: print hostname
  - resources: 1 node, 1 task, 100 MB per CPU, 5 minutes
- Options to specify requested resources
  - command line options
  - special comments in the job script
  - command line supersedes in-file

hello.sh

```
#!/bin/bash
hostname
exit $?
```

Command line options

hello.sh

```
#!/bin/bash
hostname
exit $?
```

```
$ sbatch -N1 -n1 --mem-per-cpu=100 -t00:05:00 hello.sh
```

1 node
1 task
at least 100 MB per CPU
5 minutes of wall time
the script

```
Submitted batch job 184917
```

Command line options

hello.sh

```
#!/bin/bash
hostname
exit $?
```

megabytes

```
$ sbatch -N1 -n1 --mem-per-cpu=100 -t00:05:00 hello.sh
```

1 node
1 task
at least 100 MB per CPU
5 minutes of wall time
the script

```
Submitted batch job 184917
```

Command line options(using long names)

```
$ sbatch --nodes=1 --ntasks=1 --time=00:05:00 hello.sh
```

1 node
1 task

5 minutes of wall time
the script

Directives in script

hello.sh

```
#!/bin/bash

#SBATCH --nodes=1 --ntasks=1 --memory-per-cpu=100M --time=00:05:00
#SBATCH --job-name=privyet

hostname
exit $?
```

give it a name other than "hello.sh"

```
$ sbatch -N1 -n1 --mem-per-cpu=100M -t00:05:00 hello.sh
```

```
$ sbatch hello.sh
Submitted batch job 184917
```

View batch job configuration with `scontrol`

```
$ scontrol show job 184917

JobId=184917 Name=privyet
   UserId=gfekete(6064) GroupId=k3001(50501)
   Priority=4294733700 Account=k3001 QOS=normal
   JobState=PENDING Reason=Priority Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 ExitCode=0:0
   RunTime=00:00:00 TimeLimit=00:01:00 TimeMin=N/A
   SubmitTime=2013-11-18T17:40:39 EligibleTime=2013-11-18T17:40:39
   StartTime=Unknown EndTime=Unknown
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   Partition=general AllocNode:Sid=discover16:13607
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=(null)
   NumNodes=1-1 NumCPUs=1-1024 CPUs/Task=1 ReqS:C:T=*:*:*
   MinCPUsNode=1 MinMemoryCPU=100M MinTmpDiskNode=0
   Features=(null) Gres=(null) Reservation=(null)
   Shared=0 Contiguous=0 Licenses=(null) Network=(null)
   Command=/gpfsm/dhome/gfekete/slurm/hello.sh
   WorkDir=/gpfsm/dhome/gfekete/slurm
   Comment=stdout=/gpfsm/dhome/gfekete/slurm/slurm-184917.out
```

Prepare a multi-node parallel MPI job

① Program
② Compilation
③ Script
④ Submission
⑤ Reading output

① hellompi.f90

```
program hellompi

    implicit none
    include 'mpif.h'
    integer rank, ierr, irc

    call mpi_init( ierr )
    call mpi_comm_rank(MPI_COMM_WORLD, rank, ierr)
    print *,"Hello from rank", rank
    call mpi_finalize(irc)

end program hellompi
```

One the simplest possible MPI Fortran programs.

② compilation

```
$ module load comp/intel-12.1.0.233   mpi/impi-4.1.0.024
$ mpif90 hellompi.f90 -o hellompi.exe
```

Prepare job script and submit

③ script in hellompi.sh

```
#!/bin/bash
#SBATCH --job-name=test_mpi --time=00:01:00
#SBATCH --nodes=2 --ntasks-per-node=10
#SBATCH --mail-user=gfekete@me.com --mail-type=ALL
#SBATCH --output=run.stdout
#SBATCH --error=run.stderr

. /usr/share/modules/init/bash
module purge
module load comp/intel-12.1.0.233 mpi/impi-4.1.0.024
set -x

cd  /home/gfekete/slurm/pi
mpirun ./hellompi.exe
exit
```

loads module() into shell

need for mpirun

explicit **cd** encouraged

④ submission

```
$ sbatch hellompi.sh --partition=general
```

Output – standard output

```
Hello from rank 11
Hello from rank  2
Hello from rank 14
Hello from rank  8
Hello from rank 13
Hello from rank  7
Hello from rank 17
Hello from rank  4
Hello from rank 18
Hello from rank  3
Hello from rank  6
Hello from rank  9
Hello from rank  1
Hello from rank  5
Hello from rank  0
Hello from rank 10
Hello from rank 19
Hello from rank 16
Hello from rank 15
Hello from rank 12
```

⑤ stdout from run.stdout

```
_____________________________________________________________________
Job Resource Usage Summary for 227199
Note: SLURM doesn't account for the exact same resource types as PBS.
      Also, you may see some values missing or zero, typically for
      very short jobs.  However, please contact NCCS USG if you
      think the values below are in error.

  CPU Time Used          : 00:00:24
  Memory Used       : 25152K
  Virtual Memory Used    : 338096K
  Walltime Used          : 00:00:01

  Memory Requested  : 0n (n=per node; c=per core)
  CPUs Requested / Allocated : 20 / 24
  Walltime Requested     : 00:01:00

  Execution Queue   : general
  Head Node         : borg01s100
  Charged to        : k3001

  Job Stopped       : Thu Nov 21 19:04:13 EST 2013
_____________________________________________________________________
```

Output – standard error

⑤ **stderr from run.stderr**

```
+ cd /home/gfekete/slurm/pi
+ mpirun ./hellompi.exe
+ exit
+ cd /home/gfekete/slurm/pi
+ mpirun ./hellompi.exe
+ exit
```

as a result of 'set –x' in script

See all my queued jobs

```
$ squeue -u gfekete
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            227199   general test_mpi  gfekete PD       0:00      2 (Resources)
```

Job is queued and is pending in the "general" partition waiting for resources to become available

View complete state

```
$ scontrol show job 227199
JobId=227199 Name=test_mpi
   UserId=gfekete(6064) GroupId=k3001(50501)
   Priority=4294689606 Account=k3001 QOS=normal
   JobState=PENDING Reason=Resources Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 ExitCode=0:0
   RunTime=00:00:00 TimeLimit=00:01:00 TimeMin=N/A
   SubmitTime=2013-11-21T18:52:17 EligibleTime=2013-11-21T18:52:17
   StartTime=Unknown EndTime=Unknown
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   Partition=general AllocNode:Sid=dali10:30012
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=(null)
   NumNodes=2-2 NumCPUs=20-1024 CPUs/Task=1 ReqS:C:T=*:*:*
   MinCPUsNode=10 MinMemoryNode=0 MinTmpDiskNode=0
   Features=(null) Gres=(null) Reservation=(null)
   Shared=0 Contiguous=0 Licenses=(null) Network=(null)
   Command=/gpfsm/dhome/gfekete/slurm/pi/hellompi.sh
   WorkDir=/gpfsm/dhome/gfekete/slurm/pi
   Comment=stdout=/gpfsm/dhome/gfekete/slurm/pi/run.stdout
```

Interactive session

- allocate resources
- provide access to them
- does not start any application
- environment copied from the calling shell (!)

X11?

Reserve two nodes on the warp-test partition

```
$ salloc --partition=warp-test --nodes=2
salloc: Pending job allocation 235236
salloc: job 235236 queued and waiting for resources
salloc: job 235236 has been allocated resources
salloc: Granted job allocation 235236
srun.slurm: Job step created
```

View interactive job configuration with `scontrol`

```
JobId=235236 Name=sh
   UserId=gfekete(6064) GroupId=k3001(50501)
   Priority=4294680718 Account=k3001 QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=0 ExitCode=0:0
   RunTime=00:04:23 TimeLimit=12:00:00 TimeMin=N/A
   SubmitTime=2013-11-22T09:01:17 EligibleTime=2013-11-22T09:01:17
   StartTime=2013-11-22T09:01:47 EndTime=2013-11-22T21:01:47
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   Partition=warp-test AllocNode:Sid=dali10:30012
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=borg01w[038-039]
   BatchHost=borg01w038
   NumNodes=2 NumCPUs=32 CPUs/Task=1 ReqS:C:T=*:*:*
   MinCPUsNode=1 MinMemoryNode=0 MinTmpDiskNode=0
   Features=(null) Gres=(null) Reservation=(null)
   Shared=0 Contiguous=0 Licenses=(null) Network=(null)
   Command=(null)
   WorkDir=/gpfsm/dhome/gfekete/slurm/pi
   Comment=stdout=/gpfsm/dhome/gfekete/slurm/pi/slurm-235236.out
```

default wall time: 12 hours
may change!

Only used by sbatch. Safe to ignore

SLURM-defined environment variables

```
SLURM_CHECKPOINT_IMAGE_DIR=/gpfsm/dhome/gfekete/slurm/pi
SLURM_NODELIST=borg01w[038-039]
SLURMD_NODENAME=borg01w038
SLURM_TOPOLOGY_ADDR=iswC-f02-scu8.iswD-01w-[01-04].iswE-01w-01.borg01w038
SLURM_PRIO_PROCESS=0
SLURM_SRUN_COMM_PORT=36037
SLURM_PTY_WIN_ROW=48
SLURM_TOPOLOGY_ADDR_PATTERN=switch.switch.switch.node
SLURM_CPU_BIND_VERBOSE=quiet
SLURM_CPU_BIND_LIST=
SLURM_NNODES=2
SLURM_STEP_NUM_NODES=1
SLURM_JOBID=235236
SLURM_LAUNCH_NODE_IPADDR=10.1.60.10
SLURM_STEP_ID=0
SLURM_STEP_LAUNCHER_PORT=36037
SLURM_TASKS_PER_NODE=16(x2)
SLURM_JOB_ID=235236
SLURM_STEPID=0
SLURM_SRUN_COMM_HOST=10.1.60.10
SLURM_CPU_BIND_TYPE=cores
SLURM_PTY_WIN_COL=132
SLURM_NODEID=0
SLURM_SUBMIT_DIR=/gpfsm/dhome/gfekete/slurm/pi
SLURM_TASK_PID=4542
SLURM_CPUS_ON_NODE=16
SLURM_DISTRIBUTION=cyclic
SLURM_PROCID=0
SLURM_JOB_NODELIST=borg01w[038-039]
SLURM_PTY_PORT=33766
SLURM_LOCALID=0
SLURM_JOB_CPUS_PER_NODE=16(x2)
SLURM_GTIDS=0
SLURM_SUBMIT_HOST=dali10
SLURM_STEP_NUM_TASKS=1
SLURM_JOB_NUM_NODES=1
SLURM_STEP_TASKS_PER_NODE=1
SLURM_STEP_NODELIST=borg01w038
SLURM_CPU_BIND=quiet,cores
```

Some important ones...

```
SLURM_CHECKPOINT_IMAGE_DIR=/gpfsm/dhome/gfekete/slurm/pi
SLURM_NODELIST=borg01w[038-039]
SLURMD_NODENAME=borg01w038
SLURM_NNODES=2
SLURM_JOBID=235236
SLURM_TASKS_PER_NODE=16(x2)
SLURM_JOB_ID=235236
SLURM_NODEID=0
SLURM_SUBMIT_DIR=/gpfsm/dhome/gfekete/slurm/pi
SLURM_CPUS_ON_NODE=16
SLURM_DISTRIBUTION=cyclic
SLURM_JOB_NODELIST=borg01w[038-039]
SLURM_LOCALID=0
SLURM_JOB_CPUS_PER_NODE=16(x2)
SLURM_SUBMIT_HOST=dali10
SLURM_STEP_NUM_TASKS=1
SLURM_JOB_NUM_NODES=1
SLURM_STEP_TASKS_PER_NODE=1
SLURM_STEP_NODELIST=borg01w038
SLURM_CPU_BIND=quiet,cores
```

2x16 = 32 cores

MPI program from the interactive session

```
borg $ mpirun ./hellompi.exe
```

```
Hello from rank          0
Hello from rank         22
Hello from rank          8
Hello from rank         30
Hello from rank          6
Hello from rank         27
Hello from rank         26
Hello from rank          2
Hello from rank         24
Hello from rank          1
Hello from rank         25
Hello from rank         28
Hello from rank          4
Hello from rank          3
Hello from rank         21
Hello from rank          9
```

```
Hello from rank         23
Hello from rank         10
Hello from rank         17
Hello from rank         15
Hello from rank         16
Hello from rank         19
Hello from rank         11
Hello from rank         31
Hello from rank         14
Hello from rank         18
Hello from rank         13
Hello from rank         12
Hello from rank          7
Hello from rank         20
Hello from rank          5
Hello from rank         29
```

mpirun by default will use all available cores

```
SLURM_NNODES=2
SLURM_CPUS_ON_NODE=16
```

MPI program from the interactive session

```
borg $ mpirun -perhost 1 ./hellompi.exe
 Hello from rank           0
 Hello from rank           1
```

1 per host = 2

```
borg $ mpirun -perhost 3 ./hellompi.exe
 Hello from rank           0
 Hello from rank           4
 Hello from rank           5
 Hello from rank           1
 Hello from rank           2
 Hello from rank           3
```

3 per host = 6

Next...

- Simple parallel jobs without MPI
- View information about queued jobs
  - working with squeue
  - getting only the information you need with custom formats

Simple parallel jobs (no MPI)

```
$ srun --label --ntasks=24 --nodes=2 --partition=debug hostname
```

--label causes task numbers to output lines

```
srun.slurm: job 208384 queued and waiting for resources
srun.slurm: job 208384 has been allocated resources
08: borgb107
17: borgb113
18: borgb113
20: borgb113
21: borgb113
23: borgb113
13: borgb113
14: borgb113
00: borgb107
02: borgb107
06: borgb107
10: borgb107
11: borgb107
16: borgb113
22: borgb113
04: borgb107
15: borgb113
09: borgb107
19: borgb113
12: borgb113
03: borgb107
05: borgb107
01: borgb107
07: borgb107
```

Working with squeue

```
$ squeue –u gfekete
JOBID PARTITION      NAME    USER ST    TIME  NODES NODELIST(REASON)
18497   general  privyet  gfekete PD    0:00      1 (Priority)
```

```
$ squeue ––user=gfekete –all
JOBID PARTITION      NAME    USER    STATE    TIME  TIMELIMIT NODES NOD
18497   general  privyet  gfekete PENDINGD    0:00       5:00     1 (Pri
```

customize output with –o *(see man squeue for many options)*

```
$ squeue –u gfekete ––format="Your job %j needs %m Megs of memory"
Your job NAME needs MIN_MEMORY Megs of memory
Your job privyet needs 100 Megs of memory
```

Jobs with dependencies

submit job1

```
$ sbatch job1.sh
Submitted batch job jobid1
```

run job2 only after job1 finishes successfully

```
$ sbatch job2.sh --dependency=afterok:jobid1
```

Suppose job1 may run so fast that neither I nor my script has enough time to capture the job ID so by the time job2 is submitted job1 is all gone…

Jobs with dependencies: hold and release

submit job1, but hold it

```
$ sbatch --hold job1.sh
Submitted batch job jobid1
```

submit job2

```
$ sbatch job2.sh --dependency=afterok:jobid1
```

release job1 so it can run

```
$ scontrol release jobid1
```

Good to know...

Some default behavior with SLURM may be not what you expect because
local configuration is customized
ten years of PBS behavior may have introduced some expectations

1. The SLURM stdout (or stderr) file will be appended, not overwritten (if exists)
2. SLURM processes launched with srun are not run under a shell, so none of the following are not executed: `~/.profile ~/.baschrc ~/.login ~/.cshrc,` etc.
3. Accounting is by node count, not by CPU count

Thank You!

As always, feel free to contact the NCCS User Services Group with questions or problems

301-286-9120

support@nccs.nasa.gov