Argonne NATIONAL LABORATORY

ALCF SYSTEM ARCHITECTURES – SOFTWARE AND JOB SUBMISSION

CHRIS KNIGHT
ALCF Catalyst Team

OUTLINE

http://www.alcf.anl.gov/presentations

- Mira (Blue Gene Q)
  - Software & SoftEnv
  - Building your code
  - Queuing and running jobs with qsub & runjob
- Theta (KNL)
  - Software & Environment Modules
  - Building your code
  - Queuing and running jobs with qsub & aprun
- Tips for troubleshooting

MIRA – SOFTWARE & LIBRARIES

http://www.alcf.anl.gov/user-guides/software-and-libraries

- IBM system and provided libraries: /bgsys/drivers/ppcfloor
  - glibc
  - mpi
  - PAMI (Parallel Active Messaging Interface)
- Site supported libraries: /soft/libraries
  - ESSL, PETSc, HDF5, netCDF, Parallel netCDF, Boost
  - ESSL is IBM's optimized Engineering and Scientific Subroutine Library for BG/Q: BLAS, LAPACK, FFT, sort/search, interpolation, quadrature, random number, BLACS
- Additional tuned libraries: /soft/libraries/alcf
  - BLAS, FFTW, LAPACK, PARMETIS, PARPACK, SCALAPACK, SZIP, ZLIB

MIRA – SOFTWARE & LIBRARIES

http://www.alcf.anl.gov/user-guides/software-and-libraries

- Applications: /soft/applications
  - LAMMPS, NAMD, QMCPACK, CP2K, etc…
- Build tools: /soft/buildtools
  - autotools, cmake, doxygen, etc…
- Compilers: /soft/compilers
  - IBM XL, BGCLANG, GNU
- Debuggers: /soft/debuggers
  - DDT
- Performance tools: /soft/perftools
  - TAU, HPCToolkit, PAPI, Autoperf, Scalasca, HPCTW, etc…

MIRA – SOFTENV

http://www.mcs.anl.gov/hs/software/systems/softenv/softenv-intro.html

- A tool for managing a user's environment
  - Sets your paths to access desired front-end tools
  - Your compiler version can be changed here
- Settings
  - Maintained in the file ~/.soft
  - Add/remove keywords from ~/.soft to change environment
  - Make sure @default is at the very end
  - Use .bash_profile for user-specific environment modifications
- Commands
  - List all keywords defined on the system: softenv
  - Reload initial environment from ~/.soft file: resoft
  - Temporarily modify environment: soft add|remove keyword

MIRA – COMPILER WRAPPERS

http://www.alcf.anl.gov/user-guides/overview-how-compile-and-link

- IBM XL cross-compilers
  - SoftEnv key: +mpiwrapper-xl
  - mpixlc_r, mpixlcxx_r, mpixlf77_r, mpixlf90_r, mpixlf95_r, etc...
  - List complete command executed by mpi wrapper: –show
- BGCLANG cross-compilers
  - SoftEnv key: +mpiwrapper-bgclang
  - mpiclang, mpiclang++, mpiclang++11
- GNU cross-compilers:
  - SoftEnv key: +mpiwrapper-gcc
  - mpicc, mpicxx, mpif77, mpif90

MIRA – SOFTENV EXAMPLE

http://www.alcf.anl.gov/user-guides/compiling-and-linking-faq

- A minimal .soft file needed to build code for compute nodes

```
> cat ~/.soft
    +mpiwrapper-xl
    @default

> resoft
> mpixlc_r –qversion
     IBM XL C/C++ for Blue Gene, V12.1
     Version: 12.01.0000.0014
```

- Remember, after editing ~/.soft file, run command resoft to refresh environment

MIRA – IBM XL OPTIMIZATION TIPS

http://www.alcf.anl.gov/user-guides/compiling-and-linking-faq

- Optimization level
  - Best for debugging: –O0
  - Good for correctness check, baseline performance: –O2
  - Can alter program semantics unless used with –qstrict: –O3
- Tips
  - Performance can decrease at higher levels: –O4 or –O5
  - Use –qlistopt to generate a listing of all flags used in compilation
  - Use –qreport to generate a listing showing how code was optimized
  - Compiler option –g must be used to resolve code line numbers

MIRA – THREADING

http://www.alcf.anl.gov/user-guides/how-manage-threading

- OpenMP
  - IBM XL compilers: –qsmp=omp:noauto
  - GNU: –fopenmp
  - BGCLANG: –fopenmp
- Pthreads
  - NPTL Pthreads implementation in glibc requires no modification
- Tips
  - Auto thread parallelization with –qsmp=auto not always effective
  - Number of threads controlled with runjob command
  - Each core needs at least 2 (possibly more) threads for peak efficiency

MIRA – PREPARING TO SUBMIT JOB

http://www.alcf.anl.gov/user-guides/allocation-accounting-sbank

- Check that you are a member of a project: projects
- Check available disk space
  - $HOME directory: myquota
  - Project directories: myprojectquotas
  - Project directories should be used for production work
- Check that your project has core-hours available
  - Use sbank command to query allocation details
  - Allocation available to project: sbank l a –p <project_name>
  - Charges against project by user: sbank l u –p <project_name> –u <user>
  - Charges are based on partition size, not number of nodes

MIRA – COBALT

http://www.alcf.anl.gov/user-guides/cobalt-job-control

- Resource management software on all ALCF systems
- Similar to PBS (but not the same)
- Job management commands
  - Submit a job: qsub
  - Query job status: qstat
  - Delete a job: qdel
  - Alter job parameters: qalter
  - Move job to different queue: qmove
  - Place queued job (non-running) on hold: qhold
  - Release hold on job: qrls
- Examples in /soft/cobalt/examples

MIRA – QSUB

http://www.alcf.anl.gov/user-guides/cobalt-job-control

- Standard options
  - Project to charge: –A <project>
  - Queue: –q <queue>
  - Maximum walltime: –t <time_in_minutes>
  - Number of nodes: –n <number_of_nodes>
  - Number of processes: --proccount <number_of_procs>
  - Running mode: --mode <cX | script>
  - Environment variables: --env <VAR1=1:VAR2=1>
  - Prefix for output files: –O <file_prefix>
  - E-mail notifications: –M <email_address>
  - Dependencies: --dependencies <jobid1>:<jobid2>
  - Interactive job: –I or --interactive

MIRA – SUBMITTING SCRIPT JOB

http://www.alcf.anl.gov/user-guides/cobalt-job-control

- Executable is invoked within script (bash, csh, …)
- Example #1 specifies job attributes on command-line

```
> cat myscript.sh
    #!/bin/sh
    echo “Starting Cobalt job script”
    runjob --np 131072 –p 16 --block $COBALT_PARTNAME : <executable> <args>
```


```
> qsub –A myproject –t 10 –n 8192 –O my_fast_code --mode script myscript.sh
    123456
```

MIRA – SUBMITTING SCRIPT JOB

http://www.alcf.anl.gov/user-guides/cobalt-job-control

- Executable is invoked within script (bash, csh, …)
- Example #2 specifies job attributes within script

```
> cat myscript.sh
    #!/bin/sh
    #COBALT –A myproject –t 10 –n 8192 –O my_fast_code
    echo “Starting Cobalt job script”
    runjob --np 131072 –p 16 --block $COBALT_PARTNAME --verbose=INFO
        --env OMP_NUM_THREADS=4 : <executable> <args>

> qsub myscript.sh
    123456
```

MIRA – COBALT FILES FOR A JOB

http://www.alcf.anl.gov/user-guides/cobalt-job-control

- Cobalt will create 3 files per job
  - Basename defaults to jobid if not set with qsub –O argument
- Cobalt log file: <file_prefix>.cobaltlog
  - Created when job is submitted, additional info written while job runs
  - Contains submission information from qsub, runjob, and environment
- Job stderr file: <file_prefix>.error
  - Created at start of job
  - Contains job startup information and any content sent to standard error
- Job stdout file: <file_prefix>.output
  - Created at start of job
  - Contains content sent to standard output

MIRA – NOW THAT YOUR JOB IS QUEUED

http://www.alcf.anl.gov/user-guides/cobalt-job-control

- Check status of submitted jobs

```
> qstat
    JobID   User     WallTime  Nodes  State     Location
    ============================================
    123456  user1 00:30:00  512    dep_fail   None
    123457  user2 01:00:00  1024   running    CET-40400-73771-1024
    123458  user3 00:30:00  2048   queued     None
    123459  user1 01:00:00  512    running    CET-40040-73371-512
    123460  user4 00:30:00  2048   user_hold  None
```

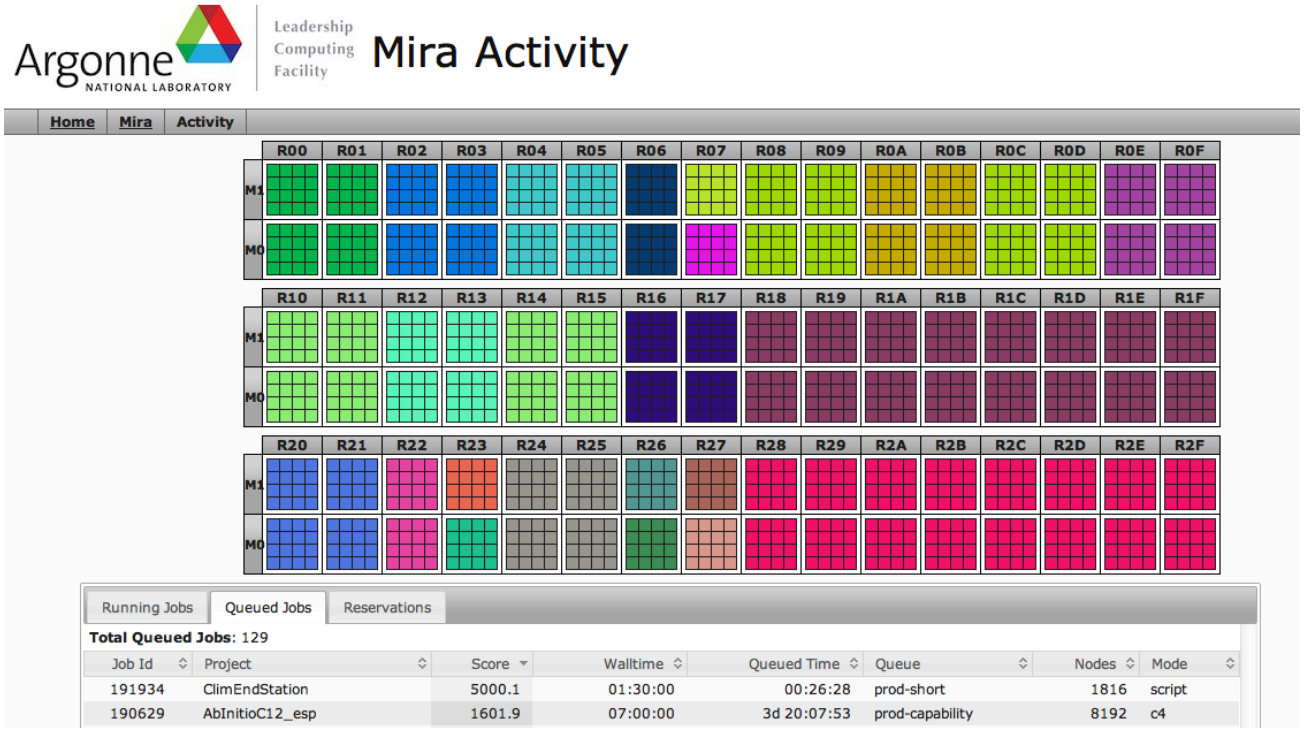
MIRA – NOW THAT YOUR JOB IS QUEUED

http://www.alcf.anl.gov/user-guides/cobalt-job-control

- Additional qstat queries
  - Show more job details: qstat –f <jobid>
  - Show all job details: qstat –fl <jobid>
  - Show all jobs from user: qstat –u <username>
  - Show information about queues: qstat –Q
- Delete job from queue: qdel <jobid>
- Alter properties of queued job
  - Change walltime: qalter –t <new_time> <jobid>
  - Change number of nodes: qalter –n <new_nodes> <jobid>
  - Change to different queue: qmove <new_queue> <jobid>

MIRA – CHECKING STATUS OF JOB

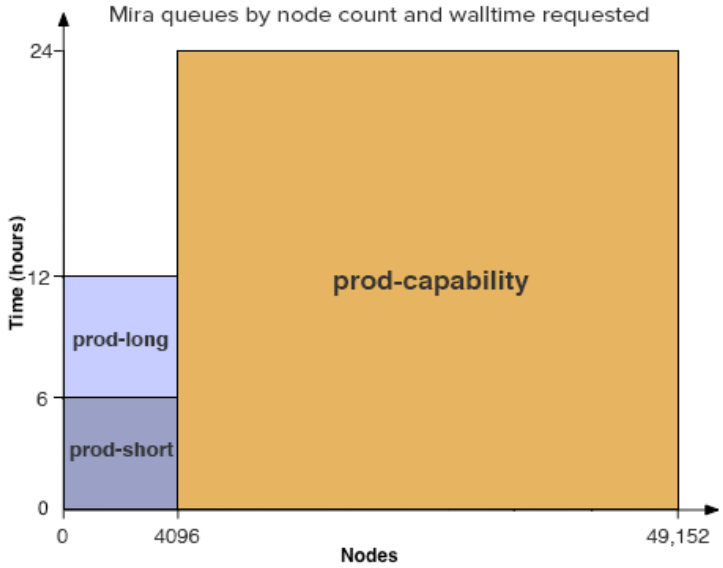
http://status.alcf.anl.gov/mira/activity



MIRA – OPTIMIZING FOR QUEUE THROUGHPUT

http://www.alcf.anl.gov/user-guides/job-scheduling-policy-bgq-systems

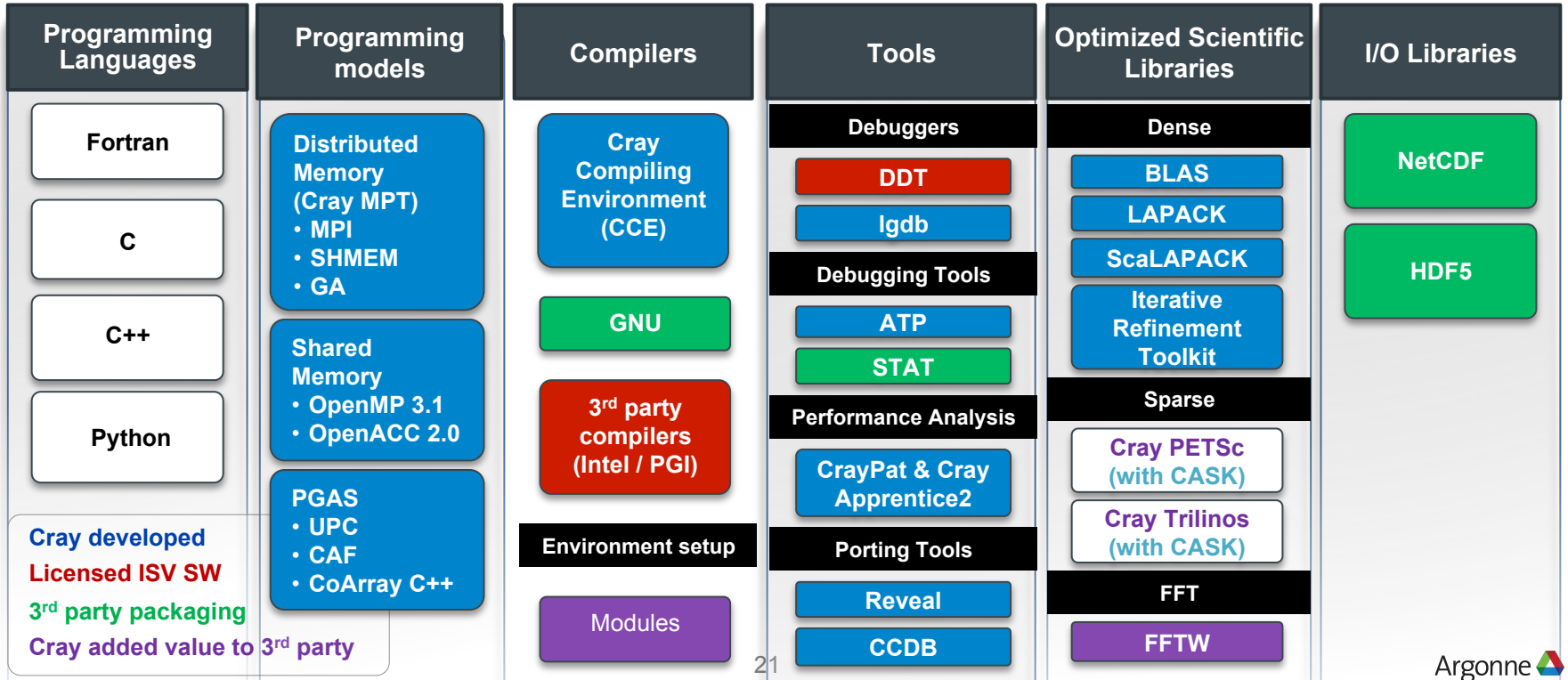
- Small (≤ 4K), long (6-12h) jobs redirected to prod-long queue, which is restricted to row 0
- Consider instead
  - Small (≤ 4K), short (≤6h) jobs redirected to prod-short queue, which run anywhere.
  - Large (> 4K) jobs redirected to prod-capability, which run anywhere.
- For long sequences of jobs, chain them together with dependencies
  - Dependent jobs inherit score boost



ANY QUESTIONS?

THETA– CRAY PROGRAMMING ENVIRONMENT

http://modules.sourceforge.net



THETA– NON-SYSTEM SOFTWARE & LIBRARIES

- Compilers: /soft/compilers
  - llvm and intel beta releases
- Debuggers: /soft/debuggers
  - DDT
- Libraries: /soft/libraries
  - argobots, bolt, breakpad
- Performance tools: /soft/perftools
  - Darshan, HPCToolkit, memlog, TAU
- Visualization: /soft/visualization
  - Paraview, R

THETA– MODULES

http://modules.sourceforge.net

- A tool for managing a user's environment
  - Sets your PATH to access desired front-end tools
  - *Your compiler version can be changed here*
- Module Commands
  - List available module commands: help
  - List currently loaded modules: list
  - List all available modules: avail
  - Add module to environment: load
  - Remove module from environment: unload
  - Swap loaded module with new one: switch|swap
  - List information about module: display|show

THETA– COMPILER WRAPPERS

- For all compilers (Intel, Cray, GNU, etc…)
  - Use cc, CC, ftn
  - Do not use mpicc MPICC, mpic++, mpif77, mpif90, etc…
    - They do not generate code for compute nodes
- Select compiler you want: module swap <old_env> <new_env>
  - Intel (default): PrgEnv-intel
  - Cray: module swap PrgEnv-intel PrgEnv-cray
  - GNU: module swap PrgEnv-intel PrgEnv-gnu
  - BGClang/LLVM: module swap PrgEnv-intel PrgEnv-llvm
- Cray wrappers
  - List complete command executed: –craype-verbose
  - Disable automatic linking with libsci: –mkl

THETA– SUBMITTING SCRIPT JOB

- Executable is invoked within script (bash, csh, …)
- aprun is used to launch executables on compute nodes

```
> cat myscript.sh
    #!/bin/sh
    #COBALT –A myproject –t 10 –n 2 –O my_fast_code –q cache-quad
    echo “Starting Cobalt job script”
    aprun -n 16 –N 8 –d 1 –j 1 --cc depth <executable> <args>
```

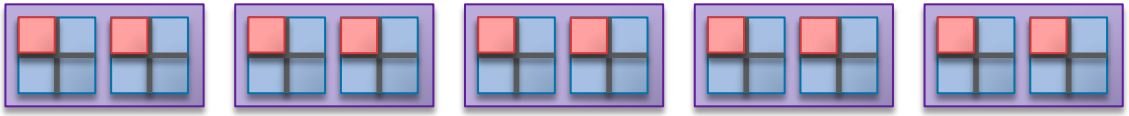

```
> qsub myscript.sh
    123456
```

THETA– APRUN OVERVIEW

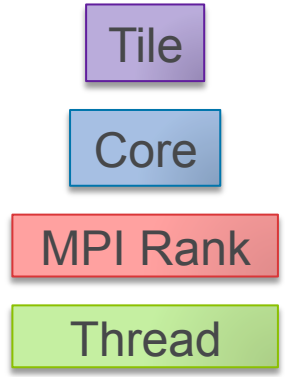
- Aprun options
  - Total number of MPI ranks: –n <total_num_ranks>
  - Number of MPI ranks per node: –N <num_ranks_per_node>
  - MPI rank and thread placement: --cc depth
  - Number of hyperthreads per core: –j <num_threads>
  - Number of hyperthreads per MPI rank (depth): –d <num_threads>
  - Environment variables: -e <env_var>
  - Core specialization: -r <num_threads>
- See also man aprun

THETA– APRUN EXAMPLES

- Theta KNL nodes have 32 tiles with 2 cores each (4 hyperthreads/core)
- Example #1: 2 nodes, 64 ranks/node, 1 thread/rank, 1 rank/core
  - aprun -n 128 –N 64 –d 1 –j 1 --cc depth –e OMP_NUM_THREADS=1 <exe>



nname= nid02937 rnk=0 tid= 0: ht= (0)
nname= nid02937 rnk=1 tid= 0: ht= (1)
nname= nid02937 rnk=2 tid= 0: ht= (2)
nname= nid02937 rnk=3 tid= 0: ht= (3)
nname= nid02937 rnk=4 tid= 0: ht= (4)



THETA– APRUN EXAMPLES

- Theta KNL nodes have 32 tiles with 2 cores each (4 hyperthreads/core)
- Example #1: 2 nodes, 32 ranks/node, 4 thread/rank, 2 threads/core
  - aprun -n 64 –N 32 –d 4 –j 2 --cc depth –e OMP_NUM_THREADS=4 <exe>

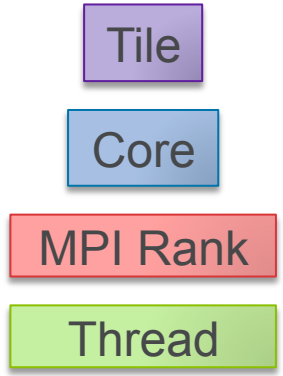
nname= nid02937 rnk=0 tid= 0: ht= (0)

nname= nid02937 rnk=0 tid= 1: ht= (1)

nname= nid02937 rnk=0 tid= 2: ht= (64)

nname= nid02937 rnk=0 tid= 3: ht= (65)

nname= nid02937 rnk=1 tid= 0: ht= (2)



THETA– APRUN TIPS

- Affinity
  - Use –d and --cc depth to let ALPS control affinity
  - Aprun and KMP affinity shouldn't be used together
  - Use --cc none if you want to use KMP_AFFINITY
- Core specialization with -r <num_threads>
  - Offload OS and MPI to unused hyperthreads
- Allocating memory in flat mode
  - Default memory allocation in DDR (NUMA 0)
  - Only allocate memory to HBM (NUMA 1): numactl -m 1
  - Prefer memory allocation to HBM: numactl -p 1
  - Example: aprun -n 128 –N 64 –d 1 –j 1 --cc depth numactl -m 1 <exe>

THETA– QUEUES

- Check available queues: qstat –Q
- Check available nodes: nodelist
- Always specify queue when submitting jobs
  - Default queue randomly selects nodes regardless of memory configuration
- Submit to queue that has nodes booted in mode you need
  - Cache-quad mode: –q cache-quad
  - Flat-quad mode: –q flat-quad
- If you need nodes in a particular mode, please contact ALCF support.

ANY QUESTIONS?

WHY HASN'T MY JOB STARTED?

- There is a reservation which interferes with your job
  - List all reservations currently in place: showres
- There are no available nodes for the requested queue
  - Nodes may be down, busy running other jobs, or draining for next job
    - Queue status: qstat
    - Machine status: http://status.alcf.anl.gov
  - Not enough nodes booted in mode you requested (Theta)
- List idle resources
  - List status of partitions on Mira: partlist
  - List status of nodes on Theta: nodelist

MIRA– CORE FILES AND DEBUGGING

http://www.alcf.anl.gov/user-guides/debugging-profiling

- Examining core files
  - Core files are in text format (e.g. readable with more command)
  - List call stack trace from single core file: bgq_stack <exe> core.#
  - List call stack trace from multiple core files: coreprocessor.pl
    - Example: coreprocessor.pl –c=<dir> -b=<exe>
    - Can also connect to running job.
- Environment variables
  - Create core dump when application exits: BG_COREDUMPONEXIT=1
  - Disables creation of any core files: BG_COREDUMPDISABLED=1
- Full-featured debugging with DDT

THETA– CORE FILES AND DEBUGGING

- Abnormal Termination Processing (ATP)
  - Set environment ATP_ENABLED=1 in job script before aprun
  - Upon failure, generates merged stack backtrace tree in atpMergedBT.dot file
  - View output file with program stat-view: module load stat
- Notes on linking your program
  - PrgEnv-cray links everything necessary by default
  - PrgEnv-intel requires –Wl,-T/opt/cray/pc/cce/8.5.2/craylibs/x86-64/2.23.1.cce.ld
- Other debugging tools
  - You can generate STAT snapshots asynchronously
  - Full-featured debugging with DDT

WHEN THINGS GO WRONG…RUNNING

- Examine core files (see previous slides)
- Best to save all three files generated by cobalt
  - *.cobaltlog, *.error, *.output
- Retain important information
  - Jobid, machine name, copy/location of all files, exact error message
- Contact us
  - Your ALCF contact
  - Email: support@alcf.anl.gov
  - Call the ALCF Help Desk
    - Hours: Monday-Friday, 9am-5pm CT
    - Phone: 630-252-3111 or 866-508-9181 (toll-free,US only)

HAPPY COMPUTING!