Argonne NATIONAL LABORATORY

# COBALT ENSEMBLE JOBS

**PAUL RICH**
Argonne Leadership Computing Facility

May 3, 2017

# OVERVIEW

- Definitions
- BlueGene/Q
  - Picking The Strategy
  - Basic Script Mode Jobs
  - Subblock Jobs
  - Multi-block Jobs
- Theta
- General Advice

# DEFINITIONS

- Block/Partition – A set of BlueGene nodes and interconnect resources
- Cobalt Job – A job submitted via qsub and controlled via Cobalt commands
- Backend Job – A job as run via the platform's execution command
  - This includes *mpirun*, *runjob* and *aprun*.
- Ensemble Job – A Cobalt job running multiple backend jobs
- Subblock Job – A BlueGene job running on a subset of a BlueGene block
- Multi-block job - A Cobalt job that runs multiple smaller blocks inside of a larger BlueGene block allocated to the job by Cobalt

# BLUE GENE Q – MIRA, CETUS AND VESTA

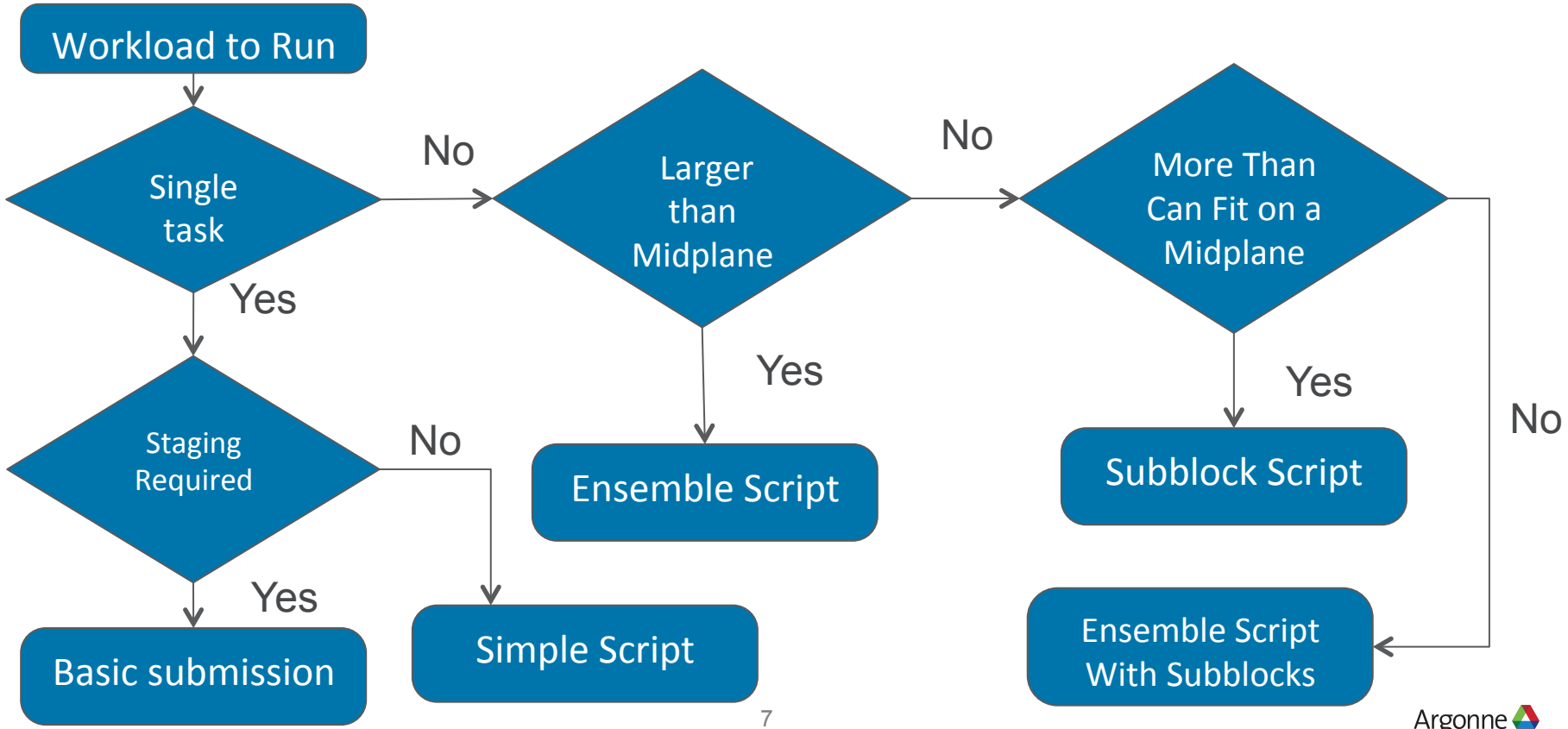
# THE BEST TOOL FOR THE JOB

- Basic Script Jobs
  - You have a task to run and some minor staging that you wish to have occur automatically
  - You need to prompt the system to take extra actions after your run
  - You have a small series of short tasks that can run on the same hardware, and want to minimize boot time
- Ensemble Jobs
  - You want to run multiple simultaneous tasks on smaller blocks within a larger allocation
  - You want to change block size between tasks

# THE BEST TOOL FOR THE JOB

- Subblock Jobs
  - Runjob feature provided by IBM
  - You have a number of small tasks to run
  - All tasks are smaller than the smallest block size on the system
  - More advanced topic
- None of these are MPMD
- Ensemble Jobs and Subblock Jobs are not either-or
  - Advanced topic covered at Ensemble Job videoconference

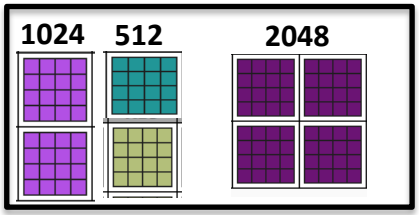
# CHOOSING THE RIGHT TYPE OF SUBMISSION



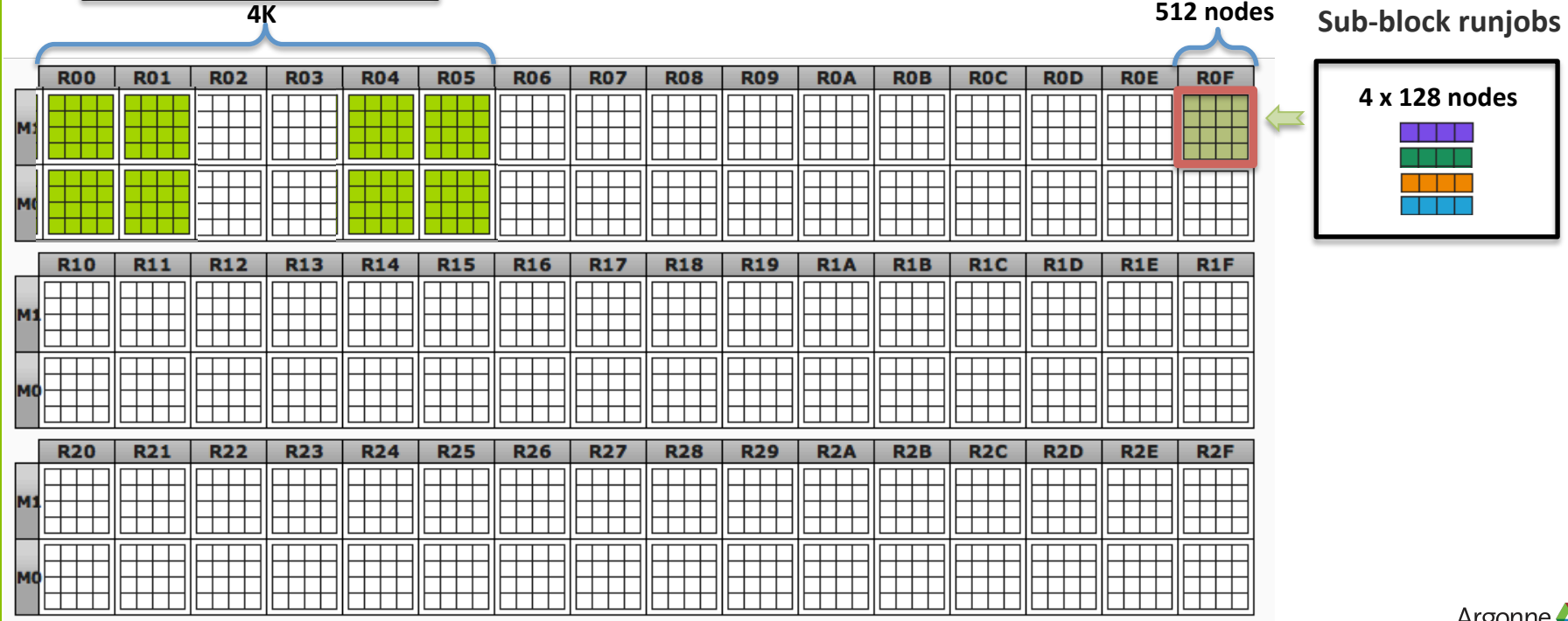
# SETTING UP SCRIPT JOBS

- Submit with *--mode script* on your qsub line
- Script can be anything  executable on a *front end node*
- Allocated block will be booted before the start of the script
- Use Cobalt-provided variables when possible: $COBALT_JOBID, $COBALT_PARTNAME, $COBALT_PARTSIZE, etc.
- Invoke runjob from your script.  You may run multiple tasks on the same block multiple times in series
- You may have to use the *boot-block --reboot* command between runs if:
  - *partlist* shows your block as having a “SoftwareFailure”
  - Your program exited with a non-zero exit status
- If using BG_PERSISTMEMSIZE, remember that contents will not persist past reboots.

# EXAMPLES OF ENSEMBLE JOBS




Multi-block job (one runjob per block)




For jobs with the same characteristics: higher job size = faster score increase

# SUBBLOCK ENSEMBLE JOBS

- Subblock jobs may be used within any script job
- Requires the use of the --corner and --shape flags to runjob
- Corner must be a hardware location
  - Can obtain this from a coordinate from */soft/cobalt/bgq_hardware_mapper/coord2hardware*
  - Use the first 5-tuple of the block name for the origin
  - Groups of corners may be obtained by passing the block name and shape to */soft/cobalt/bgq_hardware_mapper/get-corners.py*
- Shape are the lengths of each dimension
  - *man runjob* has a list of common shapes for valid subblock sizes

# SUBBLOCK ENSEMBLE JOBS

- Must target booted blocks of 512 nodes or smaller
  - Can run down to the single-node level
- Recommended that these be use on the smallest block size for a machine
  - Mira = 512, Cetus = 128, Vesta = 32
- A compute block going into error does not kill previously running jobs
  - Will prevent future jobs from starting
- Watch out for overloading IO nodes

# SUBBLOCK ENSEMBLE JOBS

- If a sub-block runjob exits abnormally, the block it was in may go into an error state
  - May not kill other current sub-block runjobs
    - Other jobs only stay up if a software failure
  - Will prevent subsequent jobs from starting on that block
  - Clear error by rebooting block
- Avoid
  - Starting runjobs too quickly
    - Must use a "sleep 3" after starting each one in background
  - Too many runjobs in total
    - Each runjob uses non-scalable resources that stress the system
    - Maximum of 512 runjobs in *all* your running jobs

# MULTI-BLOCK JOBS

- The Cobalt job's allocated block either must start off unbooted or be freed at the start of the job
  - qsub option (or #COBALT) --disable_preboot
- Boot smaller “child” blocks of the main allocated block
  - Cannot be smaller than the smallest bootable partition
  - May be subject to torus wiring restrictions

# MULTI-BLOCK JOBS

- *get-bootable-blocks* will return all child blocks currently available to boot in a main block
  - Can constrain to particular sizes and geometries
  - Booting one child may block others, they will no longer be available
- *boot-block* can boot, free, or reboot a partition
  - After booting or rebooting, the block is ready for use
  - nonzero exit status means a problem occurred
- Runjob works in the normal way, just using one child block per invocation

# MULTI-BLOCK JOBS

- Some block sizes may have issues running next to each other
  - Use partial mesh 1024 and 4096 node blocks
  - Incremental approach: after booting one block, repeat call to get-bootable-blocks
- Booting a block may fail
  - boot-block will automatically re-try 3 times before giving up
  - Software errors can be cleared by rebooting
  - partlist will show an error as blocked(SoftwareFailure)
- Can mix block sizes and change sizes
  - To change, free children then boot a new set
  - *If using persistent CNK ramdisk (/dev/persist), contents will be erased*
- Some blocks share I/O resources
  - check ALCF system documentation

# GENERAL ADVICE FOR MIRA

- Using Partial-mesh 1024 node and 4096 node blocks
  - 1024: MIR-XXXXX-YYYYY-1-1024
  - 4096: MIR-XXXXX-YYYYY-2-4096 (Not in normal queues)
- If using mesh blocks to pack, *all blocks of that size must be mesh*.
- Certain other size blocks may have alternate shapes defined
  - May need to filter output of *get_bootable_blocks*
- When packing, work from largest size to smallest
- No more than 512 simultaneous runjob invocations
  - More in series is fine
  - Limitation of control system resources

# CRAY - THETA

# SCRIPTING FOR THETA

- All jobs on Theta are either “script” mode jobs or interactive jobs
- Nodes are not normally rebooted between jobs
- Aprun blocks until job completion
  - Background for simultaneous runs
- Cobalt provides the overall allocation of nodes for a run
  - $COBALT_PARTLIST provides a list of nodes.
  - Same list format as used for “--attrs location” as well as Cray commands
- Aprun provides subsetting
  - See documentation on “-l”, “-n” and “-N” flags

# SCRIPTING FOR THETA: LIMITATIONS

- When running simultaneous apruns, a maximum of 1000 per cobalt job
  - System limit to prevent front-end resource starvation
- When starting multiple apruns, include a short sleep (<1 second)
- You may end up on any “mom” node for your run
- Apruns may be backgrounded but must not be paused (SIGSTOP)
  - Disrupts communication to the aprun front-end and will kill the aprun
- Memory mode changes (Coming Soon)
  - Jobs may request memory modes by the “mcdram” and “numa” attributes
  - If this causes a mode switch at startup, can take up to 45 minutes to complete
  - Mode changes are not currently permitted during a job

# GENERAL SCRIPT ADVICE

- Scripts may be any file executable on a front-end node
  - Shell scripts and python are common
- The job is charged for the set of allocated compute resources for the entire runtime.
  - Do not run expensive operations like compiles if you can help it.
- Check Exit and Block Statuses between runs.
- Do not delete Cobalt-generated files as a part of the script.
  - This includes the .cobaltlog, and .error files.

# ERROR HANDLING

- Always check exit statuses
  - Non-zero usually indicates a failure
  - Some codes do not follow this convention!
- Overall script exit status if usually the last command that completes
  - Save important status/statuses and use in an explicit exit
  - Masked exit statuses will impact job dependencies
- Consider using the ‘-e’ flag if using a shell script.
- When possible test script mechanics on a debug queue/resource

# USEFUL RESOURCES

- Example scripts may be found on ALCF systems at:
  - /soft/cobalt/examples
- Cobalt Manpages may be found on all ALCF systems and on:
  - https://trac.mcs.anl.gov/projects/cobalt/wiki/CommandReference
- Advanced Bash Scripting reference:
  - http://www.tldp.org/LDP/abs/html/

# QUESTIONS?

# PARTITION DIMENSIONS: MIRA

| Nodes | A | B | C | D | E |
|---|---|---|---|---|---|
| 512 | 4 | 4 | 4 | 4 | 2 |
| 1024 | 4 | 4 | 4 | 8 | 2 |
| 2048 | 4 | 4 | 4 | 16 | 2 |
| 4096 | 4/8 | 4 | 8/4 | 16 | 2 |
| 8192 | 4 | 4 | 16 | 16 | 2 |
| 12288 | 8 | 4 | 12 | 16 | 2 |
| 16384 | 4/8 | 8/4 | 16 | 16 | 2 |
| 24576 | 4 | 12 | 16 | 16 | 2 |
| 32768 | 8 | 8 | 16 | 16 | 2 |
| 49152 | 8 | 12 | 16 | 16 | 2 |

**Command: partlist**

http://www.alcf.anl.gov/user-guides/machine-partitions

# PARTITION DIMENSIONS: CETUS AND VESTA

## Cetus

| Nodes | A | B | C | D | E |
|---|---|---|---|---|---|
| 128 | 2 | 2 | 4 | 4 | 2 |
| 256 | 4 | 2 | 4 | 4 | 2 |
| 512 | 4 | 4 | 4 | 4 | 2 |
| 1024 | 4 | 4 | 4 | 8 | 2 |
| 2048 | 4/8 | 4 | 4/8 | 4/8 | 2 |

**Command: partlist**

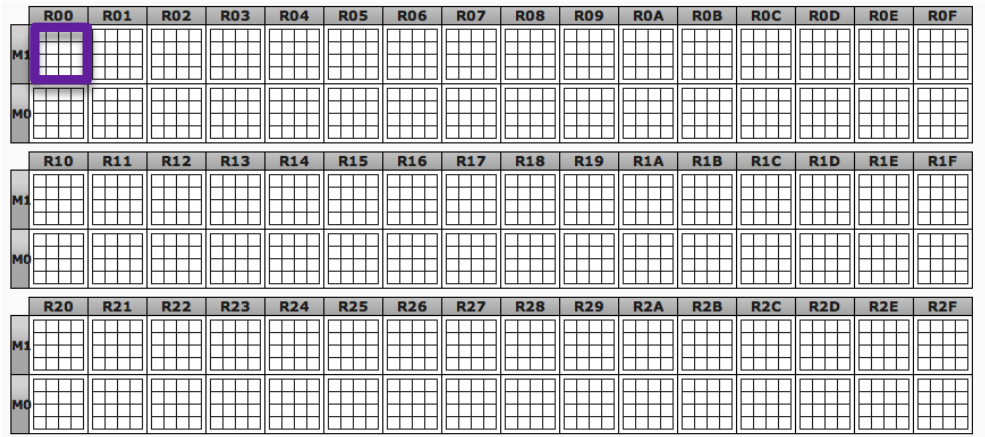
http://www.alcf.anl.gov/user-guides/machine-partitions

## Vesta

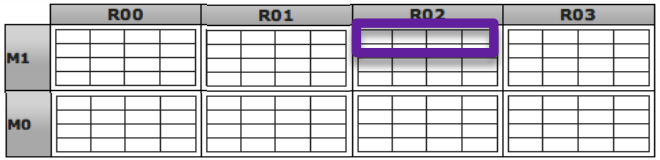
| Nodes | A | B | C | D | E |
|---|---|---|---|---|---|
| 32 | 2 | 2 | 2 | 2 | 2 |
| 64 | 2 | 2 | 4 | 2 | 2 |
| 128 | 2 | 2 | 4 | 4 | 2 |
| 256 | 4 | 2 | 4 | 4 | 2 |
| 512 | 4 | 4 | 4 | 4 | 2 |
| 1024 | 4 | 4 | 4/8 | 8/4 | 2 |
| 2048 | 4 | 4 | 8 | 8 | 2 |

# MINIMUM BGQ PARTITION SIZES

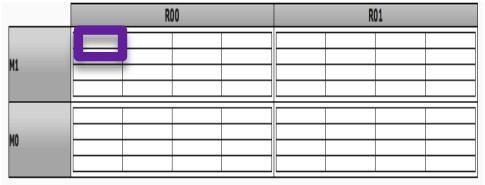
**512 nodes = minimum partition size on Mira**




**128 nodes = minimum partition size on Cetus**




**32 nodes = minimum partition size on Vesta**




Argonne NATIONAL LABORATORY

# ERROR CHECKING: EXAMPLE

```
# Bash function for waiting for exit statuses
. /soft/cobalt/examples/ensemble/script/wait-all
pids=""
for B in $BLOCKS ; do
  boot-block –block $B &
  pids+=" $!"
done
wait-all "boot" $pids                  # bash function from above
[ $? –ne 0 ] && exit 1                  # quit if any of the boots fail
# Can use the same method for any backgrounded commands
# E.g. runjob
```

# ARRAY OF ARGUMENTS: EXAMPLE

```
rootdir=`pwd`
dir[0]=$rootdir/subdir_a
dir[1]=$rootdir/subdir_b
...
cmd[0]="-p 1 --np 16 : a.out"
cmd[1]="-p 16 --np 256 : b.out"
...
i=0
for B in $BLOCKS ; do
  cd ${dir[$i]}
  runjob –-block $B ${cmd[$i]} >LOG.output 2>LOG.error &
  sleep 3
  ((i++))
done
wait
```

# BLOCK NAME TRANSLATION

- */soft/cobalt/bgq_hardware_mapper* contains basic helper scripts
- *hardware2coord* -- take a hardware location and translate to ABCDE
- *coord2hardware* -- take an ABCDE location and translate to a hardware location
- get-corners.py experimental -- given a block name and a shape, generate every valid *--corner* argument for that shape on that block.
  - Must be used on a block of 512 nodes or smaller