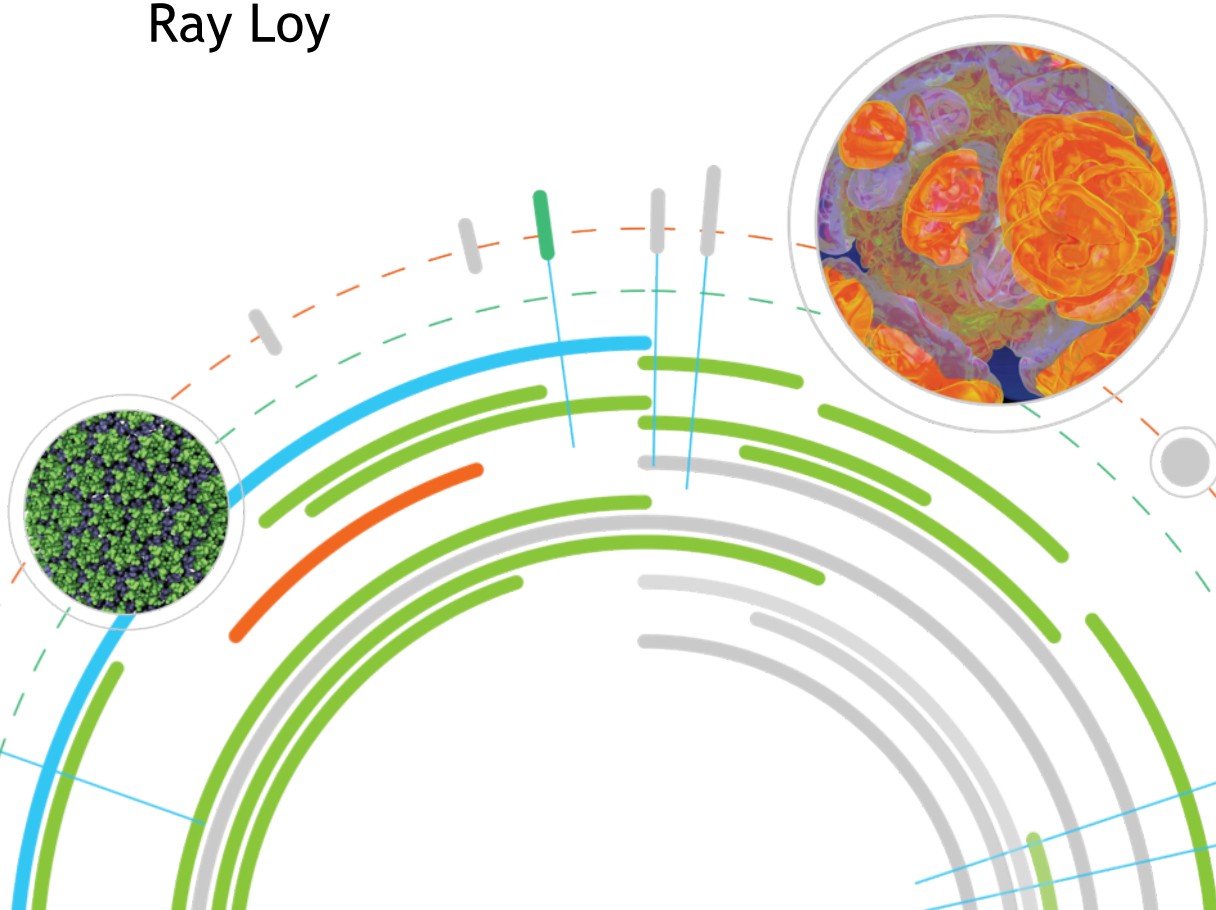
Ensemble Jobs for Better Throughput

Paul Rich
Ray Loy



Argonne Leadership Computing Facility

Argonne NATIONAL LABORATORY

This information only applies to ALCF Blue Gene/Q resources.

Use on other types of systems or at other sites will likely require significant adjustments.

Overview

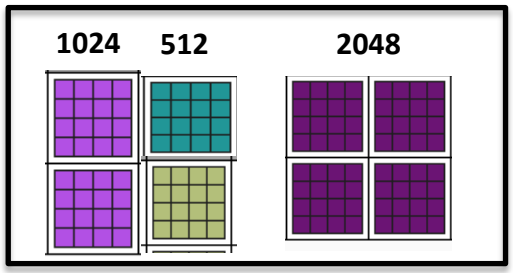
- Definitions
- Picking the right type of job
- Basic Script mode jobs
- Ensemble jobs
  - Sub-block runjobs
  - Multi-block jobs
  - Hybrid multi-block + sub-block jobs
- Error checking in job scripts

Definitions and Disambiguation

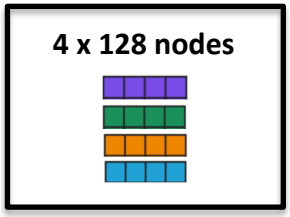
- Block – A Blue Gene partition
- Cobalt Job - A job submitted to Cobalt via qsub. Shows up in qstat.
  - Non-script job
  - Script Job - A Cobalt job submitted with the *--mode script* option
    - Can do many things a non-script job can't
- Blue Gene job - A task run on the compute nodes via runjob
  - runjob is BG equivalent of mpirun or mpiexec
- Ensemble job – A Cobalt job with >1 simultaneous runjob
- Sub-block runjob – Runjob only uses part of a booted block. The block can be shared with other sub-block runjobs
- Multi-block job - A Cobalt job that takes the overall block assigned by the scheduler and separately boots smaller blocks within it. Simultaneous runjobs may be run on these smaller blocks.

Examples of multi-block and sub-block jobs

Multi-block job (one runjob per block)



Sub-block runjobs



4K

512 nodes

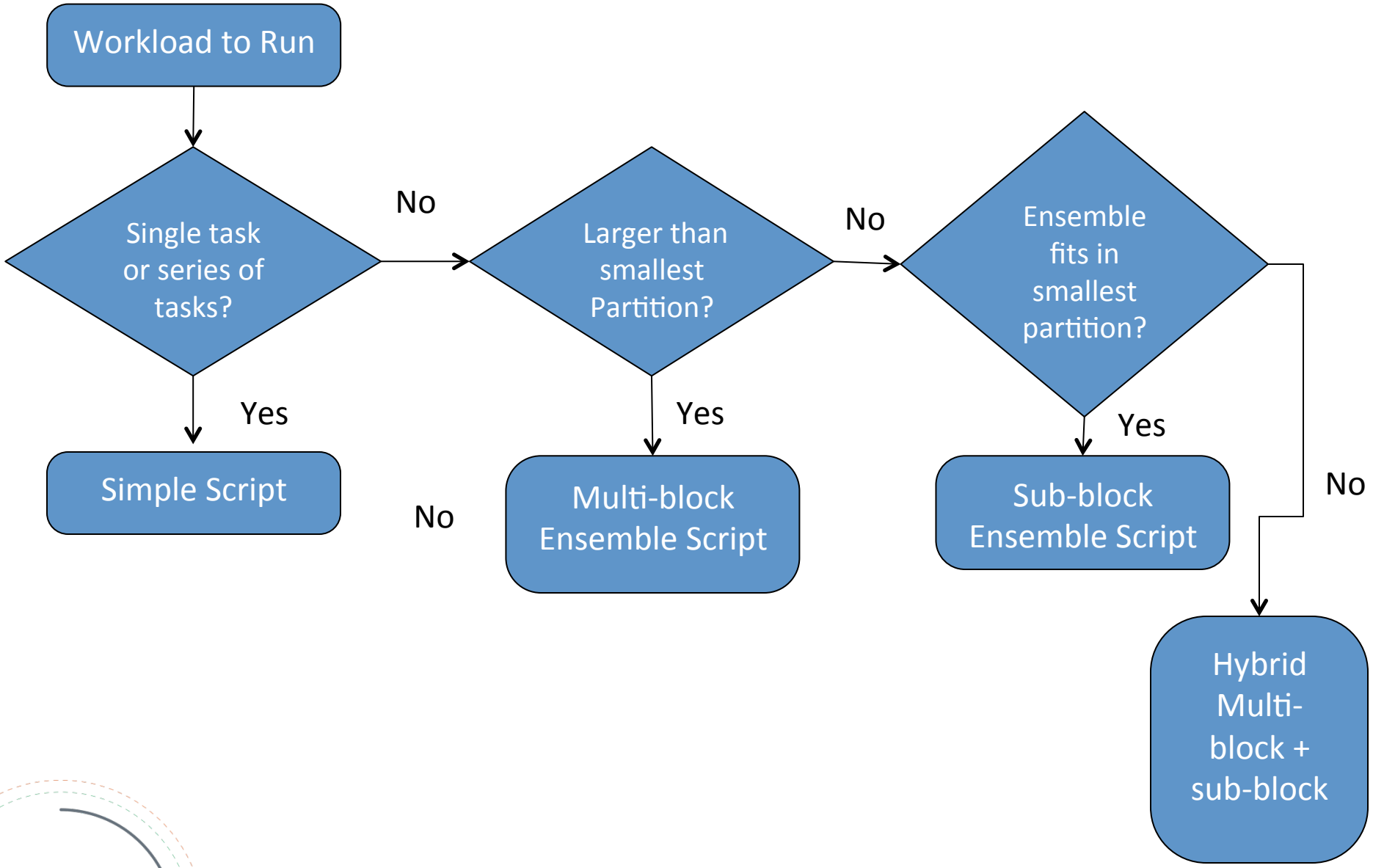
Minimum partition size on Mira

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

Types of script jobs and the best tool for the job

- Basic Script Jobs
  - Can run commands both before and after the runjob
  - Can run a series of runjobs one after the other
- Ensemble Jobs
  - You want to run multiple simultaneous tasks
  - Single-block with sub-block runjobs
    - All tasks are smaller than the smallest block size on the system
      - Mira=512, Cetus=128, Vesta=32
  - Multi-block
    - Boot smaller blocks within the overall Cobalt allocation
    - Two possibilities
      - Run one runjob on each block
      - OR run multiple sub-block runjobs in each block
  - Advanced: Either sub-block or multi-block can change runjob size between tasks
  - NOTE: None of these methods are MPMD runs

Choosing the Right Type of Submission



Equivalent Cobalt jobs

- Non-script
  - qsub –t 10 –n 32 --mode c16 --proccount 512 a.out arg1 arg2
- Script
  - qsub –t 10 –n 32 --mode script job.sh

    ```
    #!bin/bash
    runjob –p 16 –np 512 --block $COBALT_PARTNAME : a.out arg1 arg2
    ```
  - qsub job.sh

    ```
    #!bin/bash
    #COBALT –t 10 –n 32
    runjob –p 16 –np 512 --block $COBALT_PARTNAME : a.out arg1 arg2
    ```
  - qsub job.sh

    ```
    #!bin/bash
    #COBALT –t 10 –n 32 --disable_preboot
    boot-block --block $COBALT_PARTNAME
    runjob –p 16 –np 512 --block $COBALT_PARTNAME : a.out arg1 arg2
    ```

Script job basics

- Script can be any executable for a *front-end node* (e.g. shellscript, python, ...) if you submit with --mode script
  - Shell scripts containing #COBALT are implicitly script mode
- The job script runs on a front-end node but the set of allocated compute nodes is charged for the entire runtime.
  - Avoid running long serial (non-runjob) commands e.g. compilation
- By default, the entire block allocated by Cobalt will be booted before starting the script
- Cobalt sets: $COBALT_JOBID, $COBALT_PARTNAME, $COBALT_PARTSIZE, ...
- runjob starts execution on compute nodes
  - Multiple runjobs may be run in series
    - Advanced: check status of block in between runjobs
- Be careful about last command in a script
  - “echo done” will cause exit status of 0 regardless of anything else!
  - Very important if you’re using job dependencies
- The Cobalt job's .output/.error are the stdout/stderr from your job script.
  - Do not delete these files (or the .cobaltlog) – help us help you

Consecutive runjobs

```
#!bin/bash
#COBALT –t 10 –n 32

runjob –p 16 --np 512 --block $COBALT_PARTNAME : a.out arg1 arg2
status=$?

if [ $status –ne 0 ] ; then
  echo "Error on first run, quitting"; exit 1
fi

runjob –p 8 --np 256 --block $COBALT_PARTNAME : a.out foo bar
status=$?

if [ $status –ne 0 ] ; then
  echo "Error on second run"
fi
exit $status
```

Sub-block runjobs

- Sub-block runjobs may be used within any script job
  - e.g. a simple one-block job, or a multi-block job
- Recommended use is only within smallest hardware partition
  - Mira=512, Cetus=128, Vesta=32
  - Can run down to the single-node level
  - Only supported for booted blocks of 512 nodes or smaller
- Use runjob --corner and --shape flags
  - Shape gives the extents of a 5D patch e.g. "2x2x4x2x2" (=64 nodes)
    - *man runjob* has a list of common shapes for small sub-block sizes
    - Size **must** be a power of 2
  - Corner is a hardware location
    - A disjoint set of corners may be obtained by passing the block name and a shape to */soft/cobalt/bgq_hardware_mapper/get-corners.py*

Sub-block runjob example

```
#!/bin/bash
#COBALT –n 32 –t 10

SHAPE=1x2x2x2x2   # 16 nodes
CORNERS=`get-corners.py $COBALT_PARTNAME $SHAPE`

for C in CORNERS; do
   runjob –-block $COBALT_PARTNAME --corner $C –shape $SHAPE –p 1 --np 16 :
a.out >RUNJOB.$C.output 2>&1 &
   sleep 3
done

wait
exit 0       # Need to do more coding to collect runjob statuses
```

Sub-block runjob Caveats

- 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
  - However, will prevent future jobs from starting
  - When this happens, wait for sub-block runjobs to complete (or kill them), then reboot block.
- Avoid
  - Starting runjobs too quickly
    - Must use a "sleep 3" after starting each one in background
  - Overloading I/O nodes
  - 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
- *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
  - Advanced: you can also run a set of sub-block runjobs on each child

Example Multi-block Script

```
#!/bin/bash
#COBALT -n 1024 -t 10 --disable_preboot

BLOCKS=`get-bootable-blocks --size 512 $COBALT_PARTNAME`

for BLOCK in $BLOCKS ; do
  boot-block --block $BLOCK &
done
wait

for BLOCK in $BLOCKS ; do
  runjob --block $BLOCK : ./my_binary >RUNJOB.$BLOCK 2>&1 &
  sleep 3  # Important
done
wait

# More code required to check for runjob success/fail
exit 0
```

Based on http://trac.mcs.anl.gov/projects/cobalt/wiki/BGQUserComputeBlockControl

Multi-block Caveats

- Some block sizes may have issues running next to each other
  - Adjacent 4096- and 1024-node full-torus blocks (due to physical wiring)
    - Use partial mesh versions of these blocks
  - Incremental approach: after booting one block, repeat call to get-bootable-blocks
- Booting a block may fail
  - File systems may fail to mount. Hardware may die during boot.
  - boot-block will automatically re-try 3 times before giving up
  - partlist will show an error as blocked(SoftwareFailure)
    - Software errors can be cleared by rebooting
- 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 by a block reboot.*
- Once a block is booted, can run multiple runjobs on it
- Some blocks share I/O resources
  - check ALCF system documentation
- Test your script on Cetus, if possible
  - Adjust block sizes for test

Considerations for Mira

- Adjacent 1024 node and 4096 node blocks have potential torus wiring conflicts
  - Avoid by using blocks with partial mesh dimensions
  - 1024: MIR-XXXXX-YYYYY-1-1024 (same blocks used for the prod-short/ prod-long queues)
  - 4096: MIR-XXXXX-YYYYY-2-4096 (Not in any normal queues)
- Certain other size blocks may have alternate shapes defined
  - You may have to use grep to filter the output of get-bootable-blocks
- If using mesh blocks to pack, **all** blocks of that size must be mesh
  - Cannot mix torus and mesh due to wiring
- When packing different sizes, start with largest block and work down in sizes
  - This will result in the most efficient packing
- No more than 512 simultaneous runjob invocations
  - More in series is fine, this is a limit for simultaneous runs
  - This is based on a global Blue Gene control system limit

Hybrid Multi-block boot + Sub-block runjobs

```
#!/bin/bash
#COBALT –n 1024 –t 10 --disable_preboot

SHAPE=2x2x2x2x2  # 32 nodes
BLOCKS=`get-bootable-blocks --size 512 $COBALT_PARTNAME`

for B in $BLOCKS ; do
  boot-block --block $B &
done
wait

for B in $BLOCKS ; do
  CORNERS=`get-corners.py $COBALT_PARTNAME $SHAPE`
  for C in CORNERS; do
    runjob --block $B --corner $C -shape $SHAPE -p 1 -np 32 : a.out >LOG.$B.$C.output 2>&1 &
    sleep 3
done
wait

# More code required to check for runjob success/fail
exit 0
```

Handling Errors

- Check Exit Statuses
  - Non-zero means something went wrong
  - Check boot-block, runjob
- Blocks may encounter errors that cause a boot to fail but are recoverable.
  - Try to boot the block again
  - Recommend no more than 3 retries. At that point there is likely a hardware problem
  - Contact support@alcf.anl.gov if you see problems booting a particular block
- If runjob exits with a nonzero status the block may have had a “software failure”
  - Check partlist for the block, if the status is “hardware offline (SoftwareFailure)” you can reboot the block to clear the error.
    - boot-block --reboot $BLOCK_NAME
  - If the block states "hardware offline" with a different error, the block is not recoverable due to an actual hardware failure.

Questions?

Partition dimensions on BG/Q systems

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

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

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

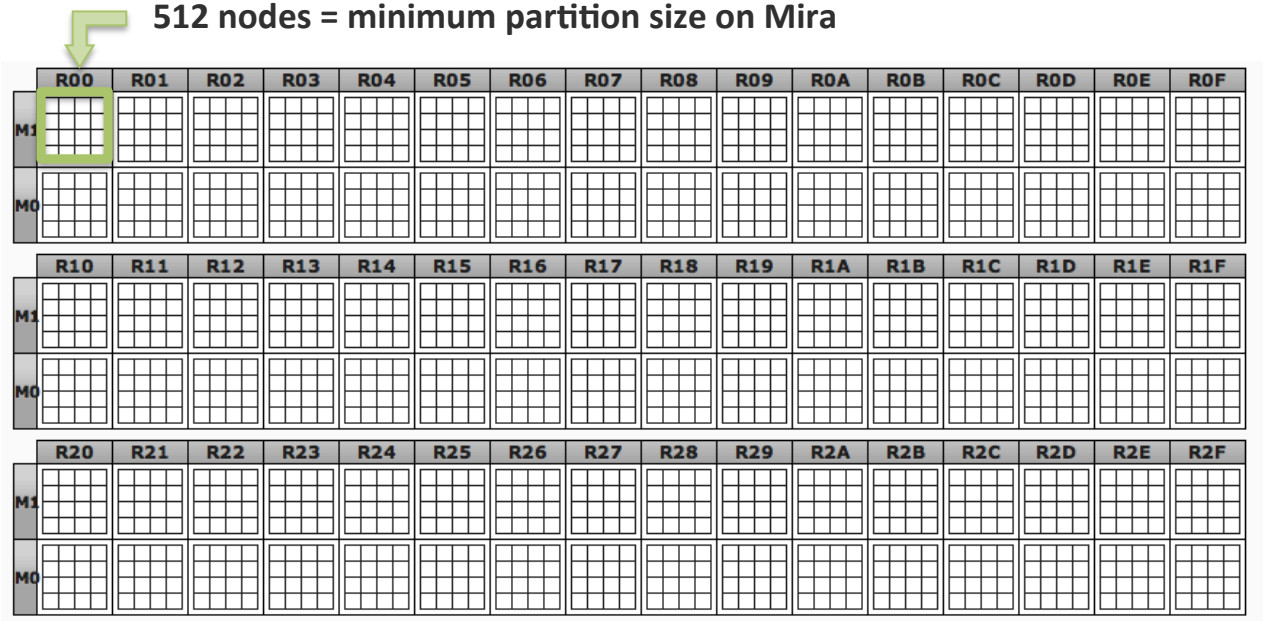
Command: partlist

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

Minimum partition sizes on BG/Q machines

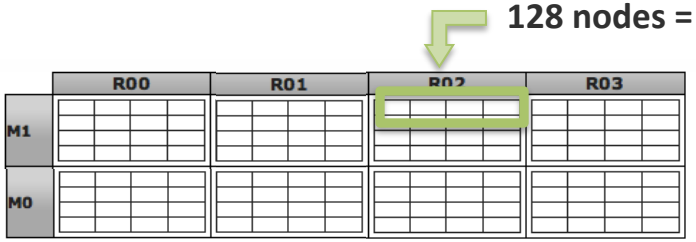
Mira

48 racks



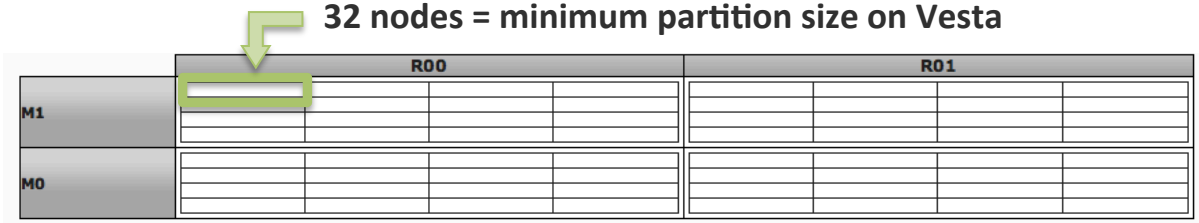
Cetus

4 racks



Vesta

2 racks



Error checking – background processes

```
# a handy bash function
. /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
```

Example: array of runjob args

```
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
```

Advanced: Block 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