



Running on Blue Gene/Q at Argonne Leadership Computing Facility (ALCF)



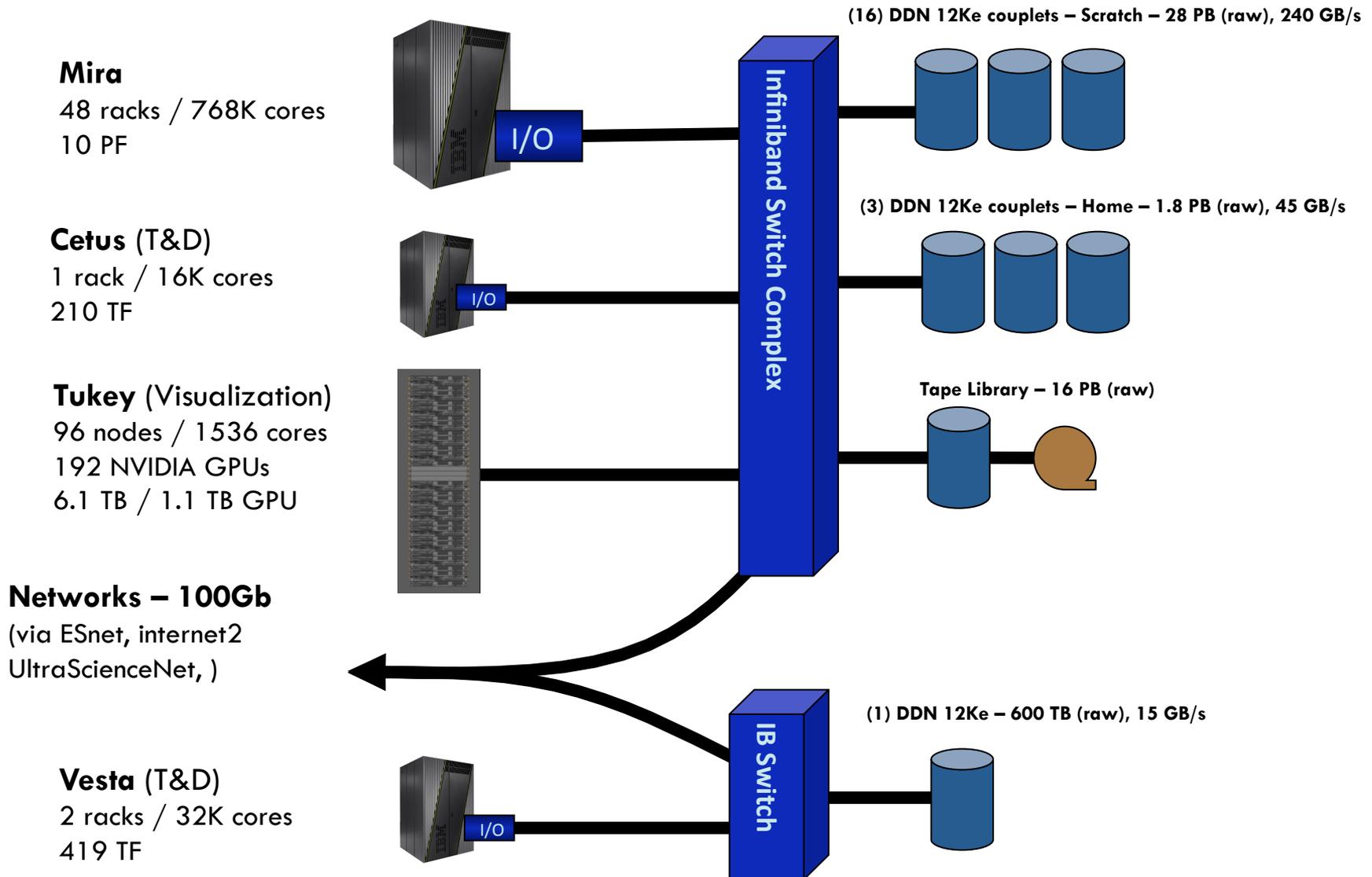
ALCF Resources: Machines & Storage

- **Mira** (Production) – IBM Blue Gene/Q
 - 49,152 nodes / 786,432 cores
 - 768 TB of memory
 - Peak flop rate: 10 PF
 - Linpack flop rate: 8.1 PF
- **Cetus** (Test & Devel.) – IBM Blue Gene/Q
 - 1,024 nodes / 16,384 cores
 - 16 TB of memory
 - 210 TF peak flop rate
- **Vesta** (Test & Devel.) – IBM Blue Gene/Q
 - 2,048 nodes / 32,768 cores
 - 32 TB of memory
 - 419 TF peak flop rate
- **Tukey** (Visualization) – NVIDIA
 - 96 nodes / 1536 x86 cores / 192 M2070 GPUs
 - 6.1 TB x86 memory / 1.1 TB GPU memory
 - 220 TF peak flop rate



- **Storage**
 - Scratch: 28.8 PB raw capacity, 240 GB/s bw (GPFS)
 - Home: 1.8 PB raw capacity, 45 GB/s bw (GPFS)
 - Storage upgrade planned in 2015

ALCF Resources: Networking Diagram



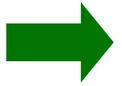
Blue Gene Architectural Features

- **Low speed, low power**
 - Embedded PowerPC core with custom SIMD floating point extensions
 - Low frequency: 1.6 GHz on Blue Gene/Q
- **Massive parallelism**
 - Many cores: 786,432 on Mira
- **Fast communication network(s)**
 - 5D Torus network on Blue Gene/Q
- **Balance**
 - Processor, network, and memory speeds are well balanced
- **Minimal system overhead**
 - Simple lightweight OS (CNK) minimizes noise
- **Standard programming models**
 - Fortran, C, C++, and Python languages supported
 - Provides MPI, OpenMP, and Pthreads parallel programming models
- **System-on-a-Chip (SoC) & Custom designed ASIC (Application Specific Integrated Circuit)**
 - All node components on one chip, except for memory
 - Reduces system complexity and power, improves price / performance
- **High reliability**
 - Sophisticated RAS (Reliability, Availability, and Serviceability)
- **Dense packaging**
 - 1024 nodes per rack

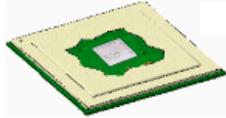


Blue Gene/Q Composition

Chip
16+2 cores



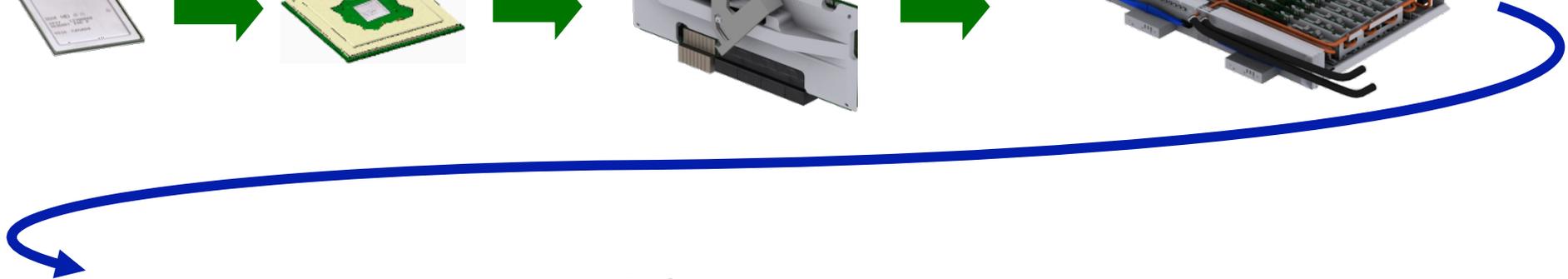
Module
Single chip



Compute card
One single chip module
16 GB DDR3 Memory
Heat Spreader for H₂O Cooling



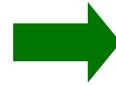
Node board
32 compute cards,
optical Modules,
link chips; 5D Torus



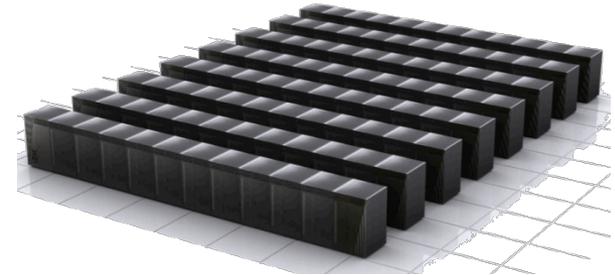
I/O drawer
8 I/O cards w/16 GB
8 PCIe Gen2 x8 slots
3D I/O Torus



Rack
1 or 2 midplanes
0, 1, 2, or 4 I/O drawers



Multi-rack system
Mira: 48 racks, 10 PF/s

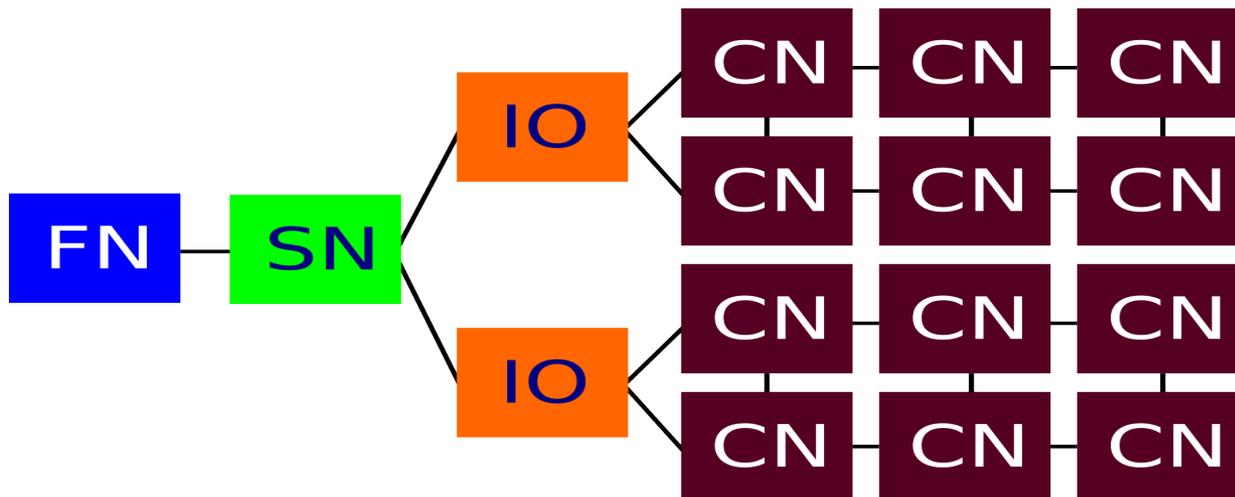


Midplane
16 node boards



Blue Gene/Q System Components

- **Front-end nodes** – dedicated for user's to login, compile programs, submit jobs, query job status, debug applications. **RedHat Linux OS**.
- **Service nodes** – perform partitioning, monitoring, synchronization and other system management services. Users do not run on service nodes directly.
- **I/O nodes** – provide a number of Linux/Unix typical services, such as files, sockets, process launching, signals, debugging; run Linux.
- **Compute nodes** – run user applications, use simple **compute node kernel (CNK)** operating system, ships I/O-related system calls to I/O nodes.



Partition Dimensions on Vesta

Vesta

Command: `partlist`

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

Do not request more than 32 nodes during afternoon labs!!!

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

Vesta
2 racks

32 nodes = minimum partition size on Vesta

	R00				R01			
M1								
M0								

<http://status.alcf.anl.gov/vesta/activity>

(*) Partition not active.

Accounts, Projects, Logging In

- ALCF Account
 - Login username
 - `/home/username`
 - Access to at least one machine
 - CRYPTOCard token for authentication
 - PIN
 - Must call ALCF help desk to activate
- Project
 - Corresponds to allocation of core-hours on at least one machine
 - User can be member of one or more projects (QMC_2014_training for this workshop)
 - `/projects/ProjectName`
- Logging in
 - `ssh -Y username@vesta.alcf.anl.gov`
 - Click button on CRYPTOCard
 - Password: PIN + CRYPTOCard display

Manage your account at
<http://accounts.alcf.anl.gov>
(password needed)

<http://www.alcf.anl.gov/user-guides/accounts-access>

Blue Gene/Q File Systems at ALCF

Name	Accessible from	Type	Path	Production	Backed up	Uses
vesta-home	Vesta	GPFS	/home or /gpfs/vesta-home	Yes	No	General use
vesta-fs0	Vesta	GPFS	/projects	Yes	No	Intensive job output, large files
mira-home	Mira Cetus Tukey	GPFS	/home or /gpfs/mira-home	Yes	Yes	General use
mira-fs0	Mira Cetus Tukey	GPFS	/projects	Yes	No	Intensive job output, large files

Disk Quota Management

- **Disk storage**

- **/home** directories:

- Default of **100 GB** for Mira and **50 GB** for Vesta
- Check your quota with the '**myquota**' command

- **/projects** directories:

- Default of **1000 GB** for Mira and **500 GB** for Vesta
- Check the quota in your projects with the '**myprojectquotas**' command

- See <http://www.alcf.anl.gov/user-guides/data-policy#data-storage-systems>

Vesta Job Scheduling

User Queue	Partition Sizes in Nodes	Wall-clock Time (hours)	Max. Running per User	Max. Queued Node-hours
default	32, 64, 128, 256, 512, 1024	0 - 2	5	1024
singles	32, 64, 128, 256, 512, 1024	0 - 2	1	1024
low	32, 64, 128, 256, 512, 1024	0 - 2	None	4096

- I/O to compute node ratio 1:32

Do not request more than 32 nodes during afternoon labs!!!

Cobalt Resource Manager & Job Scheduler

- Cobalt is resource management software on all ALCF systems
 - Similar to PBS but not the same
- Job management commands:
 - qsub**: submit a job
 - qstat**: query a job status
 - qdel**: delete a job
 - qalter**: alter batched job parameters
 - qmove**: move job to different queue
 - qhold**: place queued (non-running) job on hold
 - qrls**: release hold on job
- For reservations:
 - showres**: show current and future reservations
 - userres**: release reservation for other users

qsub: Submit a Job

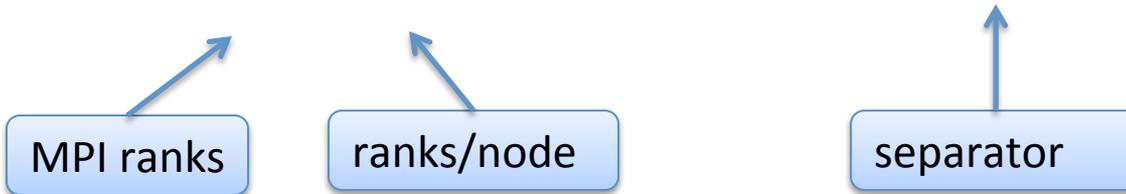
- Script: submit a script (bash, csh,)

```
qsub --mode script ... path/script
```

- Script example:

```
qsub -t 10 -n 8192 -A myproject --mode script myscript.sh
```

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



- New #COBALT syntax:

```
qsub -t 10 -n 8192 --mode script myscript.sh
```

```
#!/bin/bash  
#COBALT -A myproject -O My_Run_$jobid  
runjob --np 131072 -p 16 --block $COBALT_PARTNAME --verbose=INFO : <executable>  
<args...>
```

expands to Cobalt jobid

qstat: Show Status of a Batch Job(s)

- **qstat** # list all jobs

JobID	User	WallTime	Nodes	State	Location
301295	smith	00:10:00	16	queued	None
- About jobs
 - JobID is needed to kill the job or alter the job parameters
 - Common states: queued, running, user_hold, maxrun_hold, dep_hold, dep_fail
- **qstat -f <jobid>** # show more job details
- **qstat -fl <jobid>** # show all job details
- **qstat -u <username>** # show all jobs from <username>
- **qstat -Q**
 - Instead of jobs, this shows information about the queues
 - Will show all available queues and their limits
 - Includes special queues used to handle reservations

Machine Status Web Page



Leadership
Computing
Facility

Vesta Activity

[Home](#) [Vesta](#) [Activity](#)

	R00				R01			
M1								
M0								

Running Jobs							
Queued Jobs							
Reservations							
Total Running Jobs: 3							
Job Id	Project	Run Time	Walltime	Location	Queue	Nodes	Mode
148459	Excited_States_CNTs	01:06:53	02:00:00	VST-02440-13751-64	default	40	c16
148460	Performance	01:04:27	02:00:00	VST-22460-33571-32	default	32	script
148461	Excited_States_CNTs	00:56:10	02:00:00	VST-02660-13771-32	default	32	c16

<http://status.alcf.anl.gov/vesta/activity>

Cobalt Files Created for a Job

- Cobalt will create 3 files per job, the basename **<prefix>** defaults to the jobid, but can be set with “qsub -O myprefix”
- Cobalt log file: **<prefix>.cobaltlog**
 - first file created by Cobalt after a job is submitted
 - contains submission information from qsub command, runjob, and environment variables
- Job stderr file: **<prefix>.error**
 - created at the start of a job
 - contains job startup information and any content sent to standard error while the user program is running
- Job stdout file: **<prefix>.output**
 - contains any content sent to standard output by user program

qdel: Kill a Job

- **qdel <jobid1> <jobid2>**
 - delete the job from a queue
 - terminated a running job

qalter, qmove: Alter Job Parameters

- Allows to alter the parameters of queued jobs without resubmitting
 - Most parameters may only be changed before the run starts
- Usage: **qalter** [options] <jobid1> <jobid2> ...
- Example:
> **qalter** -t 60 123 124 125
(changes wall time of jobs 123, 124 and 125 to 60 minutes)
- Type '**qalter -help**' to see full list of options
- qalter cannot change the queue; use **qmove** instead:
> **qmove** <destination_queue> <jobid>

qhold, qrls: Hold, Release Jobs

- **qhold** - Hold a submitted job (will not run until released)
`qhold <jobid1> <jobid2>`
- To submit directly into the hold state, use `qsub -h`
- **qrls** - Release a held job (in the *user_hold* state)
`qrls <jobid1> <jobid2>`
- Jobs in the *dep_hold* state released by removing the dependency
or `qrls --dependencies <jobid>`
or `qalter -dependencies none <jobid>`
- Jobs in the *admin_hold* state may only be released by a system administrator

Why a Job May Not Be Running Yet

- There is a reservation, which interferes with your job
 - **showres** shows all reservations currently in place
- There are no available partitions for the requested queue
 - **partlist** shows all partitions marked as functional
 - **partlist** shows the assignment of each partition to a queue

```
Name                Queue                State                //
=====
...
MIR-04800-37B71-1-1024  prod-short:backfill  busy                //
MIR-04880-37BF1-1-1024  prod-short:backfill  blocked (MIR-048C0-37BF1-512) //
MIR-04C00-37F71-1-1024  prod-short:backfill  blocked (MIR-04C00-37F31-512) //
MIR-04C80-37FF1-1-1024  prod-short:backfill  idle                //
...
```

- Job submitted to a queue which is restricted to run at this time