Getting Started on Theta

Graham Fletcher
ALCF Catalyst Team

ALCF Simulation, Data, and Learning Workshop
February 27, 2018

Outline

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

- Hardware
  - System overview
  - Processor
  - Execution modes
- Software
  - Operating System
  - Programming environment
  - Modules
  - Building Your Code
  - Tools
- Queuing and running jobs
  - aprun
  - Queues
  - Cobalt

Tips for troubleshooting

Theta- Hardware

Theta System Overview

Architecture: Cray XC40

Processor: 1.3 GHz Intel Xeon Phi 7230 SKU

Cores/node: 64

Racks: 24

Nodes: 4,392

Memory/node: 192 GB DDR4 SDRAM

High bandwidth memory/node: 16 GB MCDRAM

SSD/node: 128 GB

Aries interconnect with Dragonfly configuration

Total cores: 281,088

Total MCDRAM: 70 TB

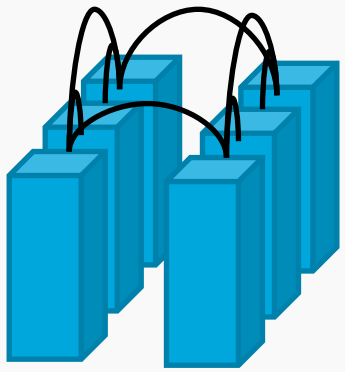
Total DDR4: 843 TB

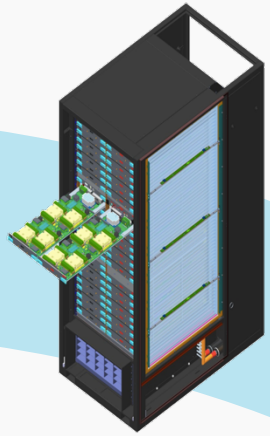
Total SSD: 562 TB

10 PB Lustre file system

Peak performance of 11.69 petaflops

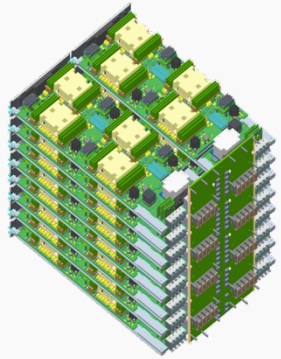
Theta system overview



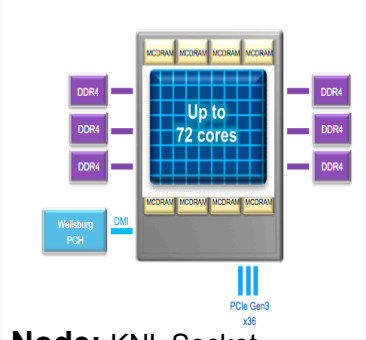


Cabinet: 3 Chassis, 75kW liquid/air cooled
510.72 TF 3TB MCDRAM, 36TB DRAM

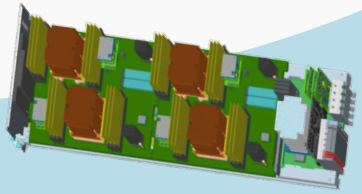
System: 24 Cabinets
4392 Nodes, 1152 Switches
Dual-plane, 12 groups, Dragonfly 7.2 TB/s Bi-Sec
11.69 PF Peak
70 TB MCDRAM, 843 TB DRAM



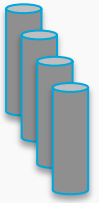
Chassis: 16 Blades, 16 Cards
64 Nodes, 16 Switches
170.24 TF 1TB MCDRAM, 12TB DRAM



Node: KNL Socket
192 GB DDR4 (6 channels) **2.66 TF** 16GB MCDRAM



Compute Blade:
4 Nodes/Blade + Aries switch
128GB SSD
10.64 TF 64GB MCDRAM
768GB DRAM



Sonexion Storage
4 Cabinets
Lustre file system
10 PB usable
210 GB/s

Filesystems

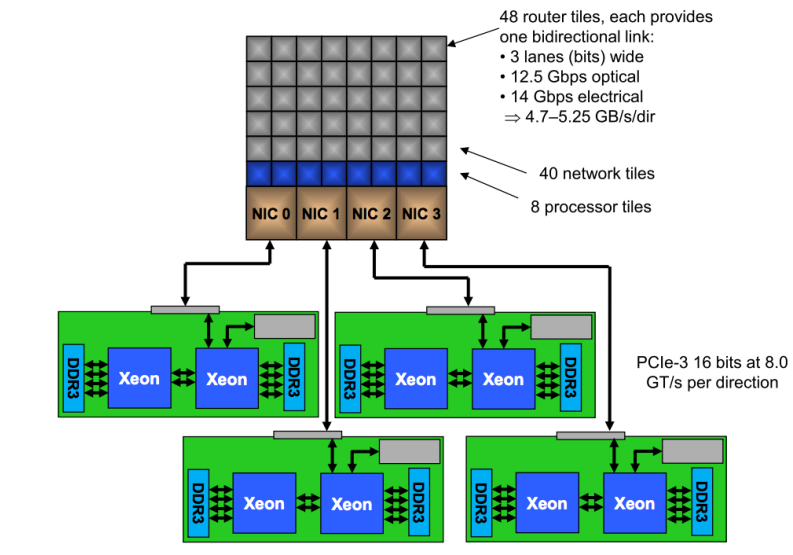
GPFS

– Home directories (/home) are in /gpfs/mira-home
  - Default quota 50GiB
  - Your home directory is backed up

Lustre

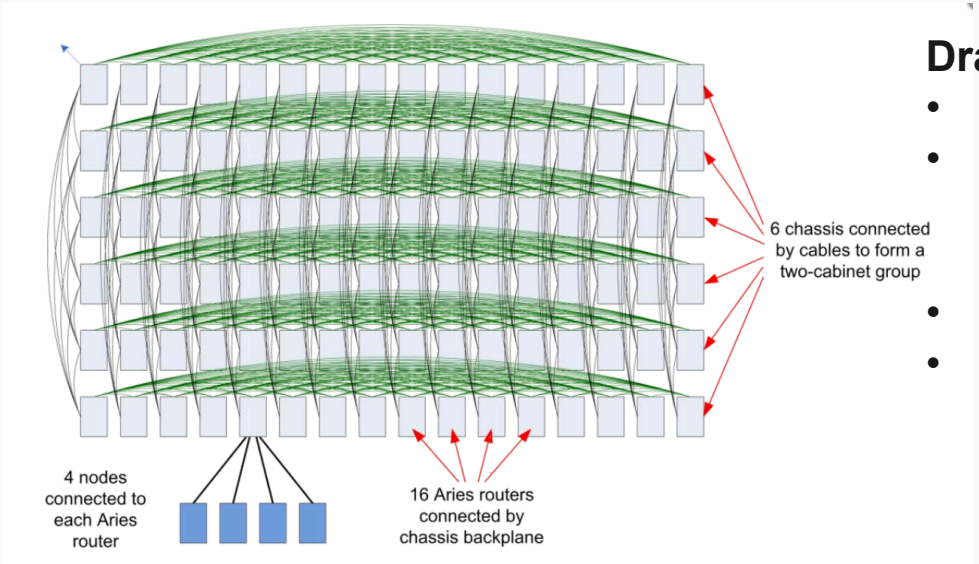
– Project directories (/projects) are in /lus/theta-fs0/projects
  - Access controlled by unix group of your project
  - Default quota 1TiB
  - NOT backed up
– With large I/O, be sure to consider stripe width

Aries Dragonfly Network



Aries Router:

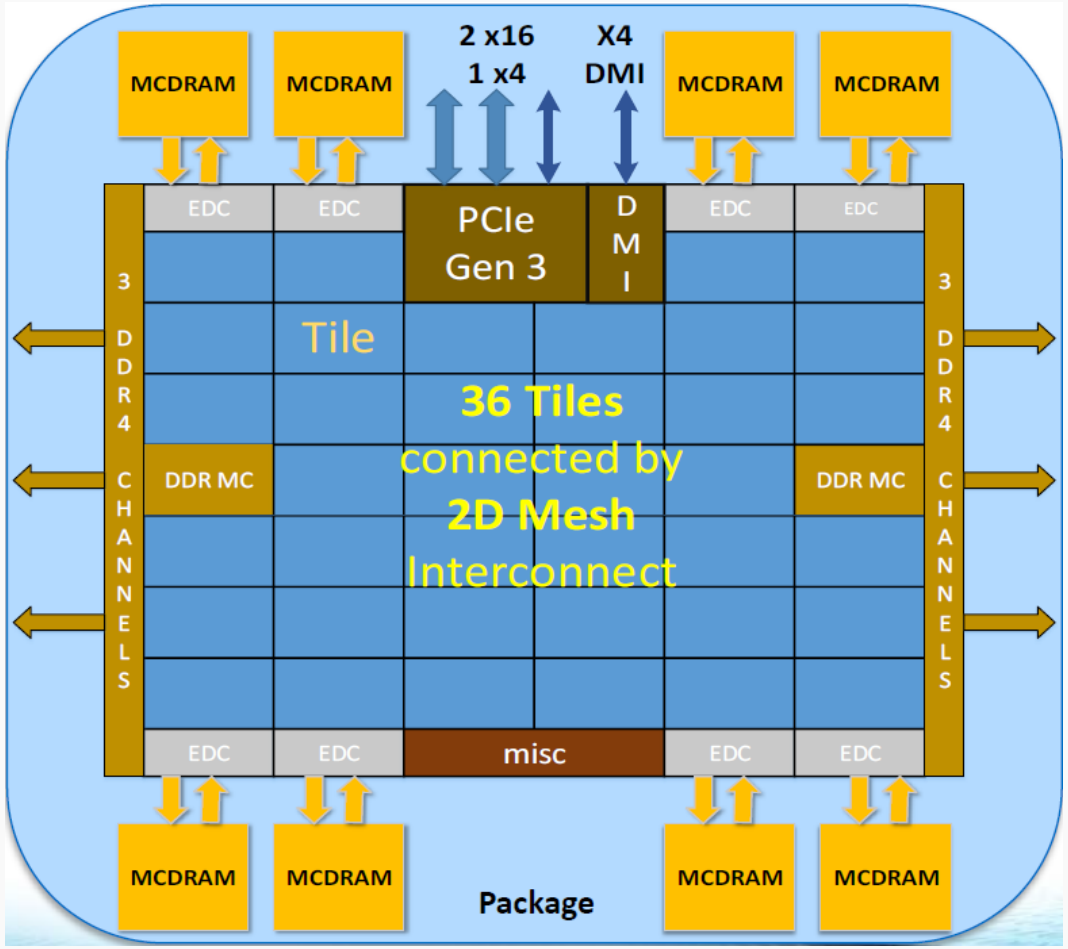
- 4 NIC's connected via PCIe
- 40 Network tiles/links
- 4.7-5.25 GB/s/dir per link



Dragonfly topology

- 4 nodes connected to an Aries
- 2 Local all-to-all dimensions
  - 16 all-to-all horizontal
  - 6 all-to-all vertical
- 384 nodes in local group
- All-to-all connections between groups

Knights Landing Processor



Chip
- 683 mm²
- 14 nm process
- 8 Billion transistors

Up to 72 Cores
- 36 tiles
- 2 cores per tile
- 2.4 TF per node

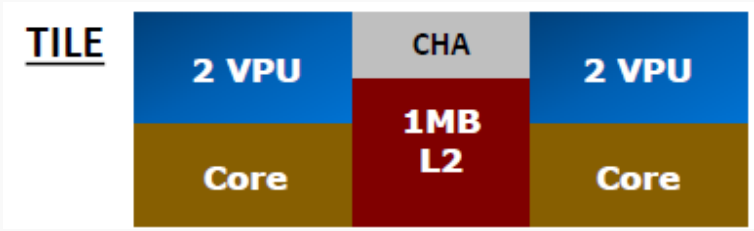
2D Mesh Interconnect
- Tiles connected by 2D mesh

On Package Memory
- 16 GB MCDRAM
- 8 Stacks
- ~450 GB/s bandwidth

6 DDR4 memory channels
- 2 controllers
- up to 384 GB external DDR4
- 90 GB/s bandwidth

On Socket Networking
- Omni-Path NIC on package
- Connected by PCIe

KNL Tile and Core

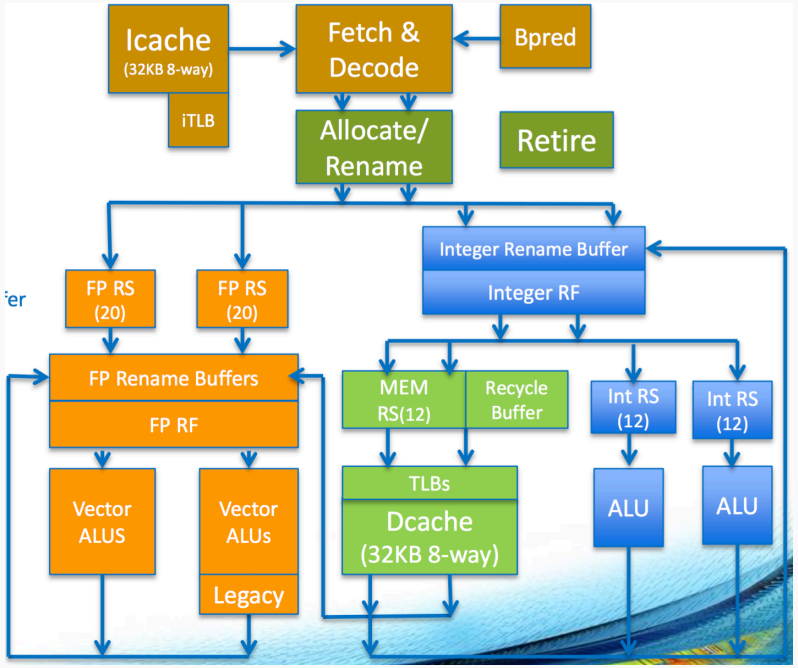


Core

- Based on Silvermont (Atom)
- Functional units:
  - 2 Integer ALUs
  - 2 Memory units
  - 2 VPU's with AVX-512
- Instruction Issue & Exec:
  - 2 wide decode
  - 6 wide execute
  - Out of order
- 4 Hardware threads per core

Tile

- Two CPUs
- 2 VPUs per core
- Shared 1 MB L2 cache (not global)
- Caching/Home agent
  - Distributed directory, Coherence



Theta- Execution Modes

Clustering modes

– All-to-All

– Quadrant/Hemisphere

– Sub-NUMA (SNC-4,2)

Memory modes

– Cache

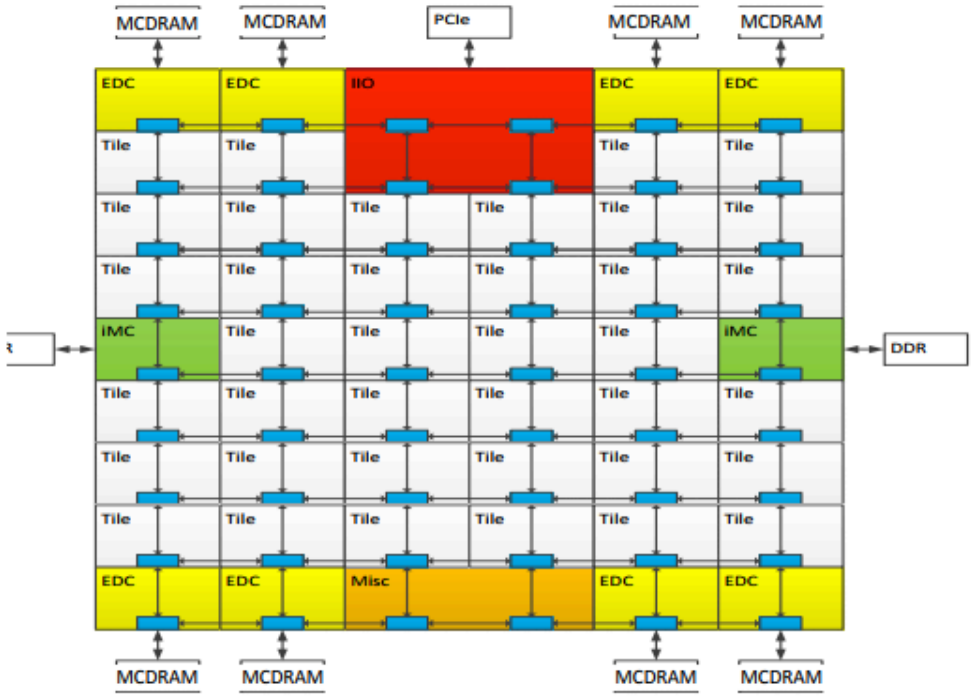
– Flat

– Hybrid

Modes are selected at node boot time (see queues)

Clustering modes

KNL Mesh Interconnect



Mesh of Rings

- Every row and column is a (half) ring
- YX routing: Go in Y → Turn → Go in X
- Messages arbitrate at injection and on turn

Cache Coherent Interconnect

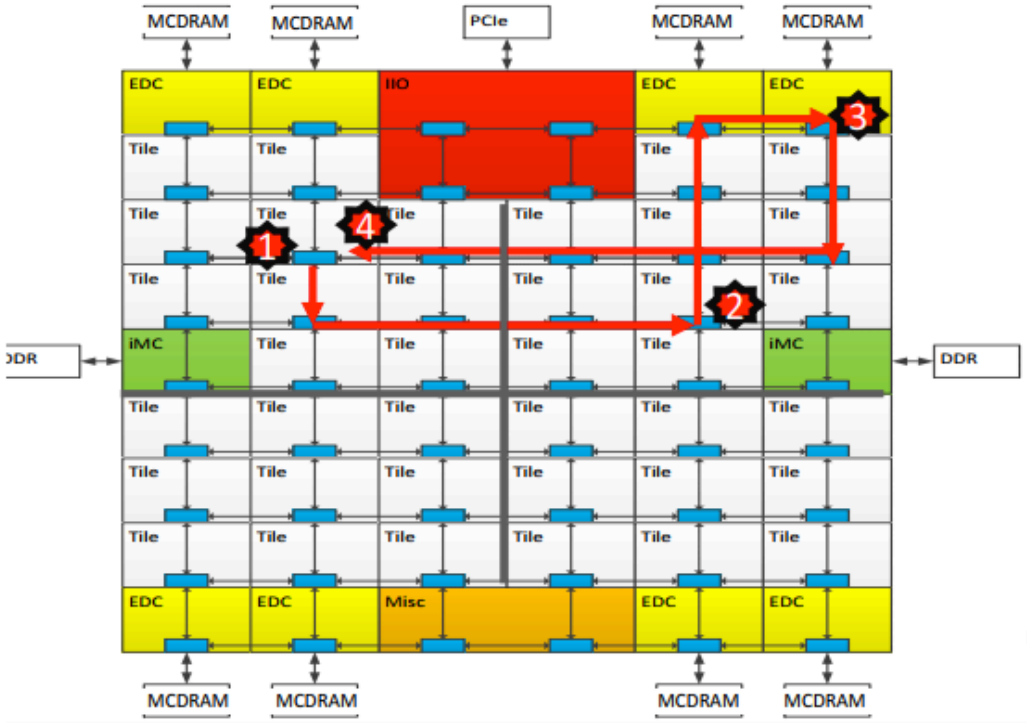
- MESIF protocol (F = Forward)
- Distributed directory to filter snoops

Three Cluster Modes

(1) All-to-All (2) Quadrant (3) Sub-NUMA Clustering

Clustering modes

Cluster Mode: Quadrant



Chip divided into four virtual Quadrants

Address hashed to a Directory in the same quadrant as the Memory

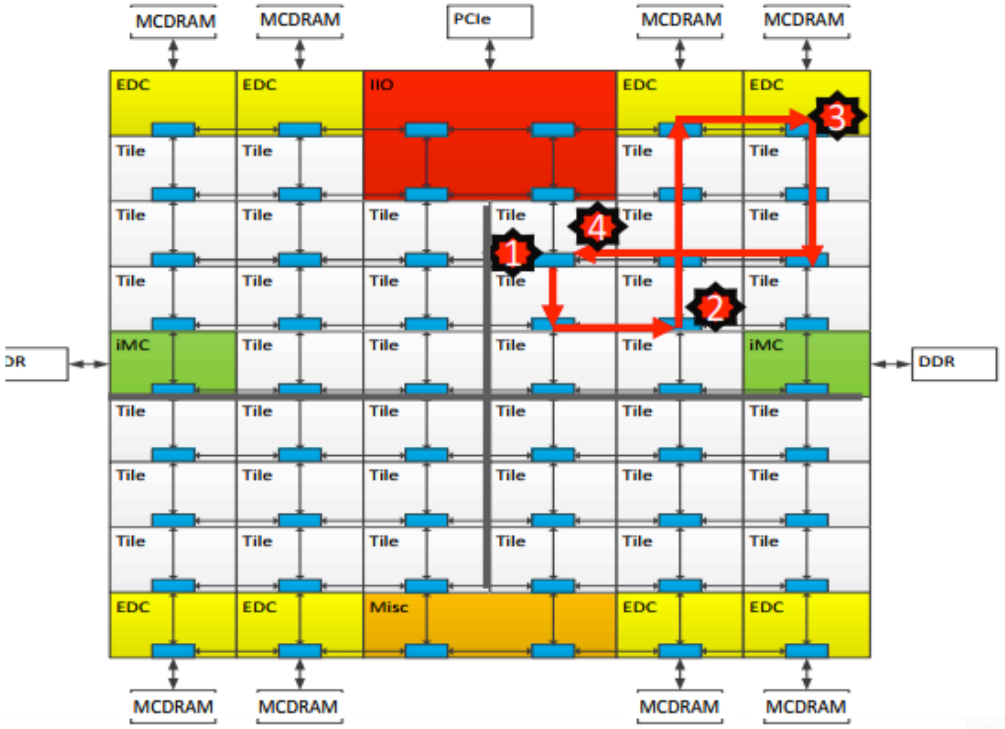
Affinity between the Directory and Memory

Lower latency and higher BW than all-to-all. SW Transparent.

1) L2 miss, 2) Directory access, 3) Memory access, 4) Data return

Clustering modes

Cluster Mode: Sub-NUMA Clustering (SNC)



Each Quadrant (Cluster) exposed as a separate NUMA domain to OS.

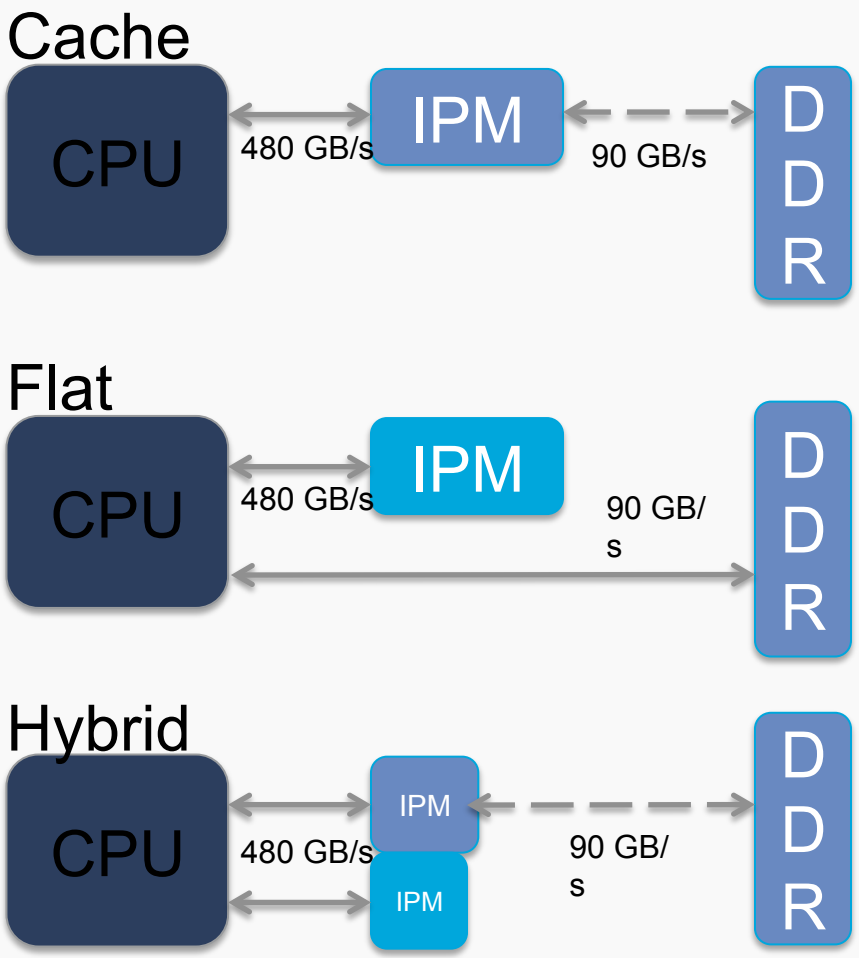
Looks analogous to 4-Socket Xeon

Affinity between Tile, Directory and Memory

Local communication. Lowest latency of all modes.

SW needs to NUMA optimize to get benefit.

Memory Modes - IPM and DDR



- **Two memory types**
  - In Package Memory (IPM)
    - 16 GB MCDRAM
    - ~480 GB/s bandwidth
  - Off Package Memory (DDR)
    - Up to 384 GB
    - ~90 GB/s bandwidth
- **One address space**
    - Possibly multiple NUMA domains
- **Memory configurations**
  - Cached: DDR fully cached by IPM
  - Flat: user managed
  - Hybrid: ¼, ½ IPM used as cache
- **Managing memory:**
  - jemalloc & memkind libraries
  - Pragmas for static memory allocations

Multi-channel DRAM in Flat Mode

Accessing MCDRAM in Flat Mode

- Option A: Using numactl
  - Works best if the whole app can fit in MCDRAM
- Option B: Using libraries
  - Memkind Library
    - Using library calls or Compiler Directives (Fortran*)
    - Needs source modification
  - AutoHBW (interposer library based on memkind)
    - No source modification needed (based on size of allocations)
    - No fine control over *individual* allocations

Multi-channel DRAM in Flat Mode

Option A: Using numactl to Access MCDRAM

- MCDRAM is exposed to OS/software as a NUMA node
- Utility numactl is standard utility for NUMA system control
  - See "man numactl"
  - Do "numactl --hardware" to see the NUMA configuration of your system
- If the total memory footprint of your app is smaller than the size of MCDRAM
  - Use numactl to allocate all of its memory from MCDRAM
  - numactl --membind=*mcdram_id* <*your_command*>
    - Where *mcdram_id* is the ID of MCDRAM "node"
- If the total memory footprint of your app is larger than the size of MCDRAM
  - You can still use numactl to allocate *part* of your app in MCDRAM
    - numactl --preferred=*mcdram_id* <*your_command*>
      - Allocations that don't fit into MCDRAM spills over to DDR
    - numactl --interleave=*nodes* <*your_command*>
      - Allocations are interleaved across all *nodes*

intel | 22

Multi-channel DRAM in Flat Mode

```
user@theta% numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 ... rest of the cores ...
node 0 size: 98207 MB
node 0 free: 94141 MB
node 1 cpus:
node 1 size: 16384 MB
node 1 free: 15923 MB
```

NUMA node 0 is the on-platform DDR4 memory with all the cores
NUMA node 1 is the on-package MCDRAM with no cores associated with it

```
user@theta% numactl --membind 1 ./run-app
OR
user@theta% numactl --m 1 ./run-app
```

Using MCDRAM in Flat Mode, Option B

Flat MCDRAM SW Usage: Code Snippets

C/C++ (*https://github.com/memkind)

Allocate into DDR

```
float   *fv;
fv = (float *)malloc(sizeof(float)*100);
```

Allocate into MCDRAM

```
float   *fv;
fv = (float *)hbw_malloc(sizeof(float) * 100);
```

Intel Fortran

Allocate into MCDRAM

```
c      Declare arrays to be dynamic
       REAL, ALLOCATABLE :: A(:)

!DEC$ ATTRIBUTES, FASTMEM :: A

       NSIZE=1024
c      allocate array 'A' from MCDRAM
c
       ALLOCATE (A(1:NSIZE))
```


ANY QUESTIONS?

Logging into Theta

ssh [your username]@theta.alcf.anl.gov ⏎

press the button on your cryptocard

enter your 4-digit PIN followed by the cryptocard sequence (upper-case!) ⏎

Theta- Software

Operating System

Cray Linux Environment (CLE)

- Login node: SUSE Enterprise Linux based full CLE OS
- Compute Node Linux (CNL)
  - Subset of CLE Linux distribution
  - Reduced OS noise and jitter, <3% runtime variability
  - Provides standard Linux services and interfaces
  - Doesn't restrict services as much as a Light Weight Kernel
  - Configurable from Extreme Scaling Mode to Cluster Compatibility Mode
  - OS activity largely confined to OS cores
  - LD_PRELOAD and shared libraries supported
  - MPMD jobs supported
  - Interfaces for controlling thread placement and affinity
  - POSIX signals
  - Memory utilization information
  - Core file generation and management via ATP

Operating System

This paper by ALCF staff-

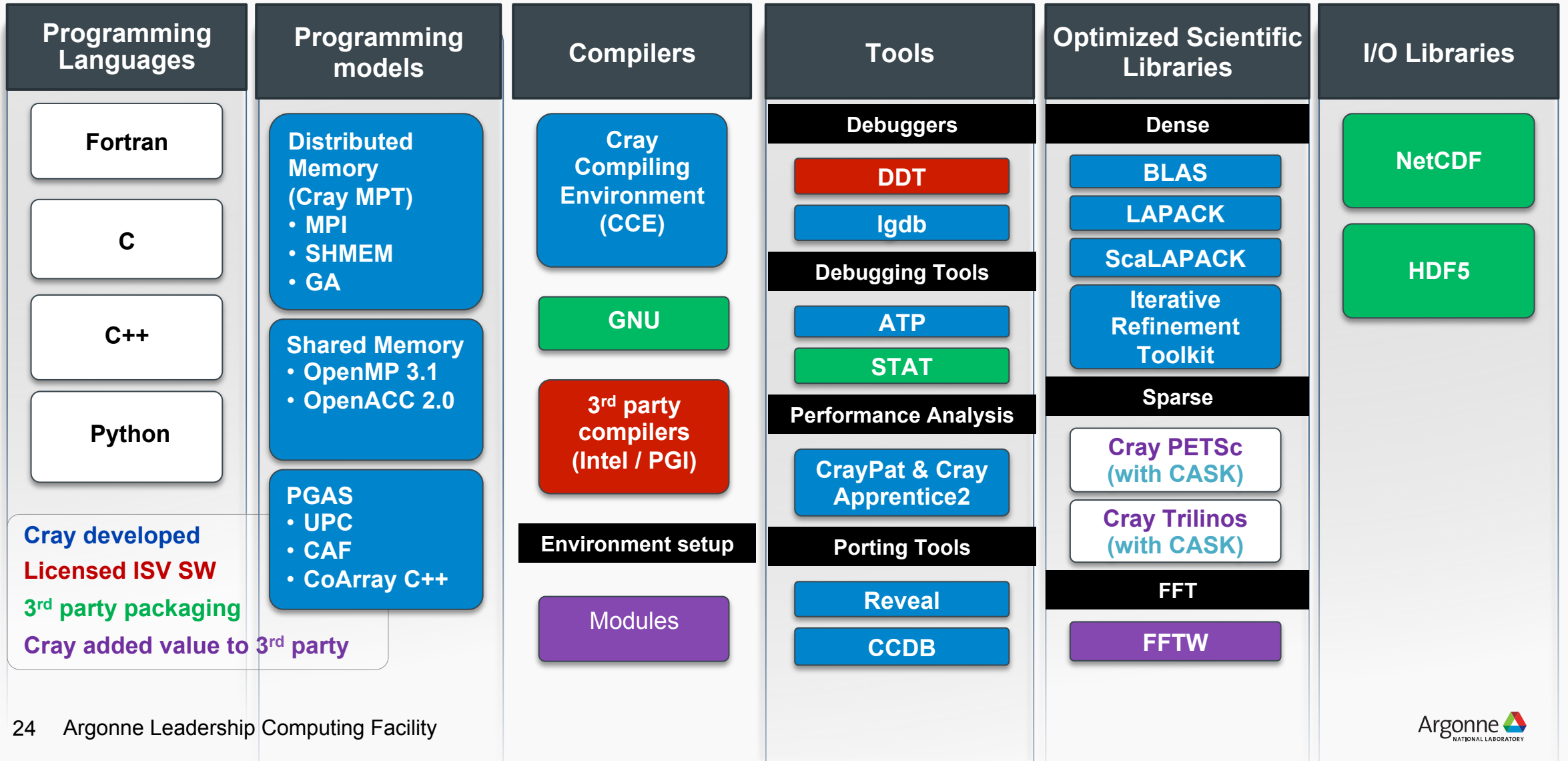
Run-to-run variability on Xeon Phi based cray XC systems
Sudneer Chunduri, Kevin Harms, Scott Parker, Vitali Morozov, Samuel Oshin*, Naveen Cherukuri*, Kalyan Kumaran,
SC'17 Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis
*Intel Corporation

Available at: https://dl.acm.org/citation.cfm?id=3126926

- deals with the sources of run time variability in KNL and Cray systems.

Cray programming environment

http://modules.sourceforge.net



Argonne NATIONAL LABORATORY

Modules

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

- help
- list ← what is currently loaded
- avail
- load
- unload
- switch|swap
- use ← add a directory to MODULEPATH
- display|show ← information about module

Building Your Code

Compiler wrappers

For all compilers (Intel, Cray, Gnu, etc):

- Use: cc, CC, ftn
- Do not use mpicc, MPICC, mpic++, mpif77, mpif90
  - they do not generate code for the compute nodes

Selecting the compiler you want using "module swap" or "module unload" followed by "module load"

- Intel
  - PrgEnv-intel This is the default
- Cray
  - module swap PrgEnv-intel PrgEnv-cray
  - NOTE: links libsci by default
- Gnu
  - module swap PrgEnv-intel PrgEnv-gnu
- Clang/LLVM
  - module swap PrgEnv-intel PrgEnv-llvm

Tools: performance, profiling, debugging

Non-system libraries and tools are under the /soft directory, module setup is in progress

- /soft/applications – applications
- /soft/compilers – site installed compilers
  - llvm and intel beta releases
- /soft/debuggers - debuggers
  - DDT
- /soft/libraries - libraries
  - argobots, bolt, breakpad
- /soft/perftools - performance tools
  - darshan, hpctoolkit, memlog, TAU, etc.

Performance Tools

CrayPat/Cray Apprentice2: Profiling, tracing, and performance visualization tool

Cray Reveal: Combines performance information with Cray compiler optimization feedback

Intel Vtune: Detailed processor level performance analysis utilizing sampling and hardware counters

Intel Trace Analyzer and Collector: MPI profiling and tracing

Intel Advisor: Provides guidance for vectorizing and threading

PAPI: Library providing API to access hardware performance counters

TAU: Profiling and tracing toolkit

HPCToolkit: Performance measurement and analysis toolkit utilizing sampling

Vampir/Score-P: Performance analysis tools providing large scale tracing and visualization

Darshan: IO characterization tool

Debugging Tools

DDT
– Full featured parallel debugger
TotalView
– Full featured parallel debugger
Cray LGDB & CCDB
– Parallel command line debugger with comparative debugging
ATP (Cray Abnormal Termination tool)
– Stack traces on exit for application failures
STAT
– Stack traces for hung applications
Intel Inspector
– Memory and thread error checking
Valgrind
– Memory debugging, memory leak detection, thread debugging with data race detection


ANY QUESTIONS?

Queuing and Running Jobs

aprun overview

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 with -r <num_threads>

– Offload OS and MPI services to unused hyperthreads

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)

Tile

Core

MPI Rank

Thread

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)

Tile
Core
MPI Rank
Thread

Submitting a Cobalt job

qsub –A <project> -q <queue> -t <time> -n <nodes> ./jobscript.sh
Example:
qsub –A Myprojname –q cache-quad t –t 10 –n 32 ./jobscript.sh

If you specify your options in the script via #COBALT, then just:
– qsub jobscript.sh

Make sure jobscript.sh is executable

Without "-q", submits to the queue named "default"

Other options:
Dependencies: --dependencies <jobid1>:<jobid2>
Place job on hold: -h

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. TWO modes are supported:
– Cache-quad mode: –q cache-quad
– Flat-quad mode: –q flat-quad

Alternative mode specification:

qsub –n 32 –t 60 –attrs mcdram=cache:numa=quad ./jobscript.sh

Users cannot directly reboot nodes.
If you need nodes in a particular mode, please contact ALCF support.
When nodes require rebooting, job may be in **starting** *state for ~15 min*

Production Queues, policy

There is a single submission queue for the entire system: default

Priority is given to jobs using at least 20% of Theta (878 nodes)

There is a global limit of ten (10) jobs running per user

There is a global limit of twenty (20) jobs in queue per user

There is a minimum job time of thirty (00:30:00) minutes for the default queue

There is a minimum allocation of 8 nodes

While shorter jobs may accumulate priority faster, all requested wall-clock times (job durations) greater than or equal to 12 hours are treated equivalently.

https://www.alcf.anl.gov/user-guides/job-scheduling-policy-xc40-systems#queues

Production Queues, policy

Wall-clock limits are a step-wise function designed to encourage scaling:

- node count >= 8 nodes : maximum 2:00:00 hours
- node count >= 16 nodes : maximum 4:00:00 hours
- node count >= 128 nodes : maximum 6:00:00 hours
- node count >= 384 nodes : maximum 12:00:00 hours
- node count >= 648 nodes : maximum 24:00:00 hours

https://www.alcf.anl.gov/user-guides/job-scheduling-policy-xc40-systems#queues

Debugging Queues, policy

There are two 16-node debugging queues:
debug-cache-quad
debug-flat-quad

Hardware is dedicated to each queue

Nodes are not rebootable to another mode

Job wall-clock time is limited to 1:00:00 (1 hour).

The maximum running job count is one (1) job per user.

https://www.alcf.anl.gov/user-guides/job-scheduling-policy-xc40-systems#queues

Submitting a 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 job_name –q cache-quad
echo “Starting Cobalt job script”
aprun -n 16 –N 8 –d 1 –j 1 --cc depth <executable> <args>
```

MPI ranks

Ranks/Node

Affinity

Queue

```
> qsub myscript.sh
```

Cobalt files for a job

Cobalt will create 3 files per job

Cobalt log file: <prefix>.cobaltlog
- created by Cobalt when job is submitted, additional info written during the job
- 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

The basename <prefix> defaults to the jobid, but can be set with “qsub -O myprefix”
- jobid can be inserted into your string e.g. "-O myprefix_$jobid"

Managing your job

qstat – show what's in the queue

- qstat –u <username> # Jobs only for user
- qstat <jobid> # Status of this particular job
- qstat –fl <jobid> # Detailed info on job

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

man qstat for more options

showres – show reservations currently set in the system

To delete a job from the queue-
qdel <jobid>

Managing your job

Other cobalt commands

Alter parameters of a queued job

qalter [most qsub options] <jobid1> …

, except the queue itself-

qmove <destination_queue> <jobid>

Place a hold on a job-

qhold <jobid>

Release a job-

qrls <jobid>

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

Interactive job

Useful for short tests or debugging
Submit the job with –I (letter I for Interactive)
Example:

- qsub –I –n 32 –t 30 –q cache-quad –A Myprojname

Wait for job's shell prompt
– This is a new shell with env settings e.g. COBALT_JOBID
– Exit this shell to end your job

From job's shell prompt, run just like in a script job, e.g.
– aprun –n 512 –N 16 –d 1 –j 1 –cc depth ./a.out

After job expires, apruns will fail. Check qstat $COBALT_JOBID

Reservations

Reservations allow exclusive use of a set of nodes for a specified group of users for a specific period of time

- a reservation prevents other users' jobs from running on that resource
- often used for system maintenance or debugging
- R.pm (preventive maintenance), R.hw* or R.sw* (addressing HW or SW issues)
- maintenance reservations appear idle

Requesting

- See: http://www.alcf.anl.gov/user-guides/reservations
- Email reservation requests to support@alcf.anl.gov
- View reservations with showres
- Release reservations with userres

When working with others in a reservation, these qsub options are useful:

- --run_users <user1>:<user2>:… All users in this list can control this job
- --run_project <projectname> All users in this project can control this job


ANY QUESTIONS?

Reasons why a job may not be running yet

Job is in state "queued”

– There is a reservation which interferes with your job
  - showres shows all reservations currently in place

– There are no available nodes for the requested queue
  - nodelist shows idle nodes

– Job was submitted to a queue that is restricted from running at this time

Job is in state "starting”

– If no nodes are currently booted in the cache/numa mode requested (via --attrs), your job may be in the state "starting" for up to 15 minutes while the nodes are rebooted.

Core files and debugging

Abnormal Termination Processing (ATP)

- Set environment ATP_ENABLED=1 in your job script before aprun
- On program failure, generates a merged stack backtrace tree in file atpMergedBT.dot
- View the output file with the program stat-view (module load stat)

Other debugging tools

- You can generate STAT snapshots asynchronously
- Full-featured debugging with DDT
- More info at
  - https://collab.cels.anl.gov/display/ESP/ATP+and+STAT
  - https://collab.cels.anl.gov/display/ESP/Allinea+Forge+(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!