ALCF Theta/Cooley Quick Start

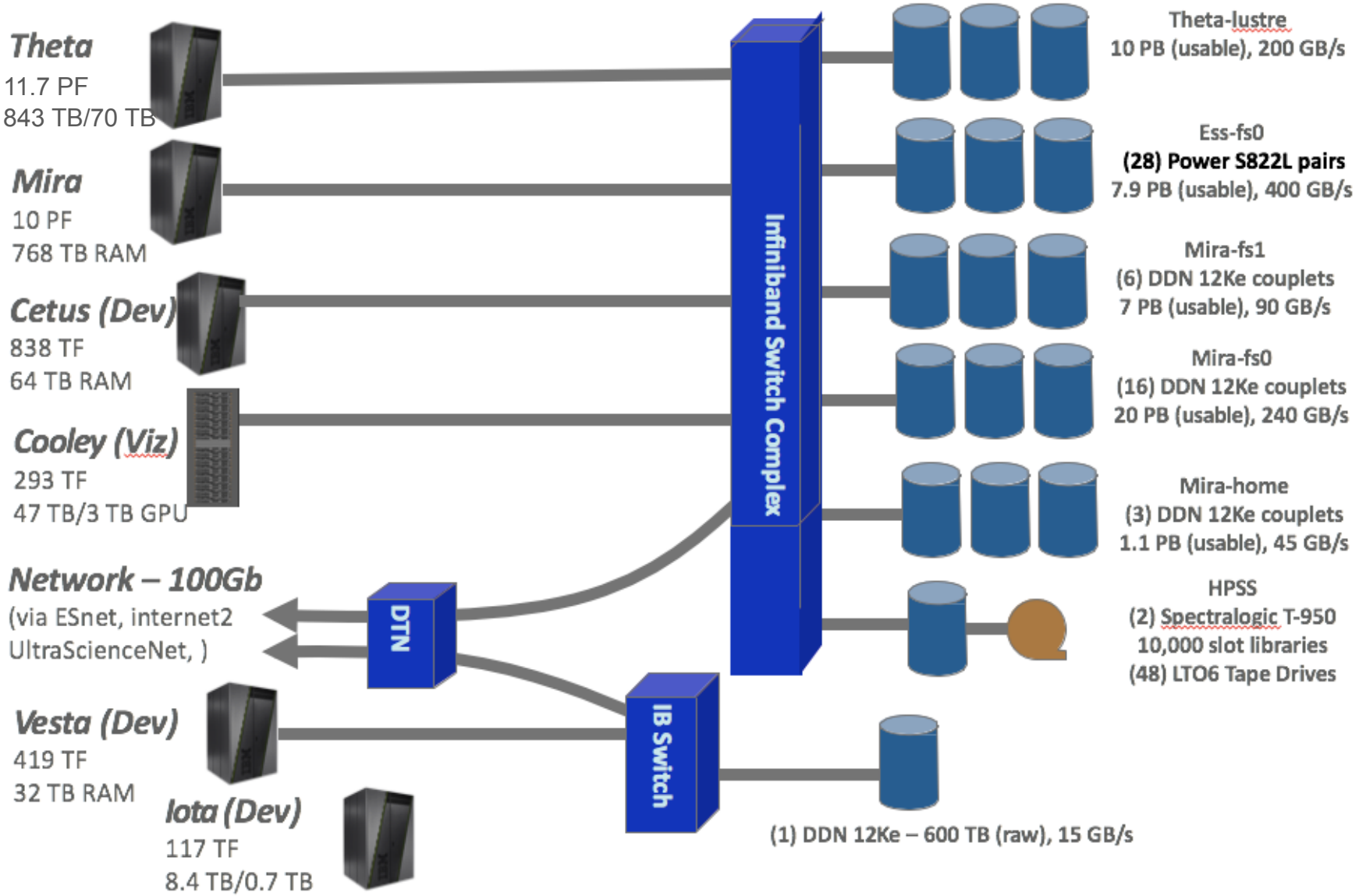
Overview

- Part 1: Hardware and Software
  - Theta Hardware Overview
  - Theta Software Environment
  - Cooley HW and SW Environment

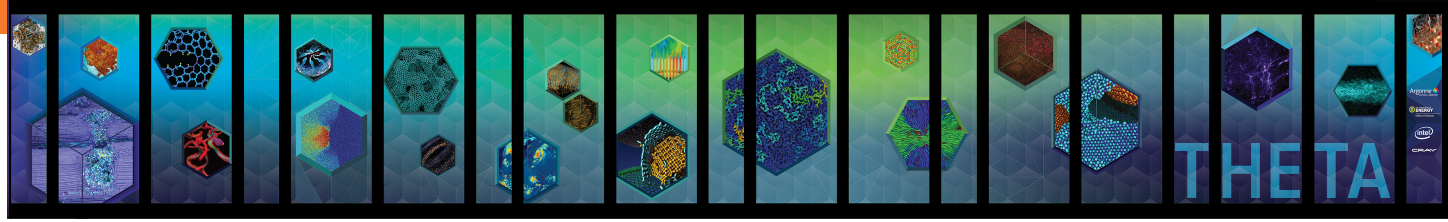
- Part 2: Queuing and Running
  - Theta Job Submission
  - Theta Job Managment and Debugging
  - Cooley Job Submission
  - Live demos / Hands-on
  - Wrap-up / Getting Help

Section:
Theta Hardware Overview

ALCF Resources



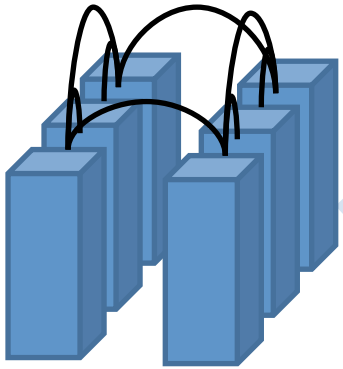
Theta

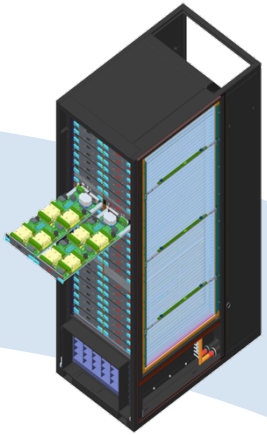


Theta serves as a bridge to the exascale system coming to Argonne

- Serves as a bridge between Mira and Aurora, transition and data analytics system
- Cray XC40 system. Runs Cray software stack
- 11.69 PF peak performance
- 4392 nodes with 2nd Generation Intel® Xeon Phi™ processor
  - codename Knights Landing (KNL), 7230 SKU 64 cores 1.3GHz
  - 4 hardware threads/core
- 192GB DDR4 memory 16GB MCDRAM on each node
- 128GB SSD on each node
- Cray Aries high speed interconnect in dragonfly topology
- Initial file system: 10PB Lustre file system, 200 GB/s throughput

Theta system overview





Cabinet: 3 Chassis, 75kW liquid/air cooled

510.72 TF 3TB MCDRAM, 36TB DRAM

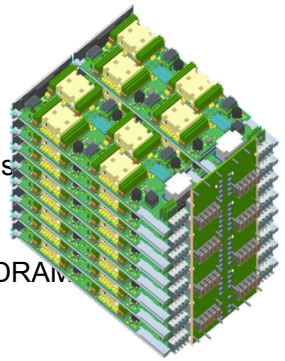
System: 24 Cabinets

4392 Nodes, 1152 Switches

Dual-plane, 12 groups, Dragonfly 12.1 TB/s Bi-Sec

11.7 PF Peak

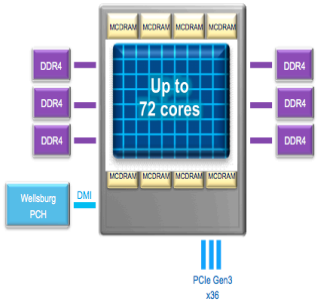
70 TB MCDRAM, 843 TB DRAM

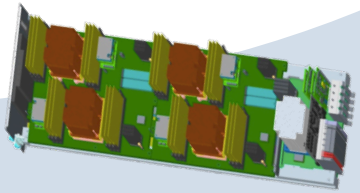


Chassis: 16 Blades, 16 Cards

64 Nodes, 16 Switches

170.24 TF 1TB MCDRAM, 12TB DRAM





Node: KNL Socket

192 GB DDR4 (6 channels) **2.66 TF** 16GB MCDRAM

128 GB SSD

Compute Blade:

4 Nodes/Blade + Aries switch

10.64 TF 64GB MCDRAM

768GB DRAM

Sonexion Storage

4 Cabinets

Lustre file system

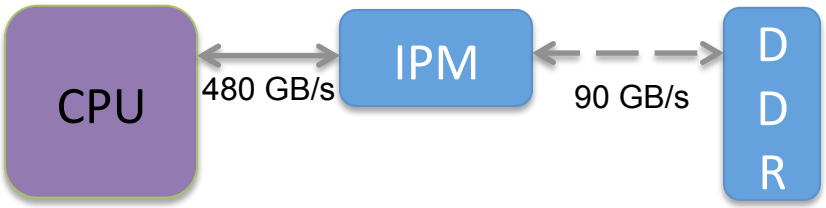
10 PB usable

210 GB/s

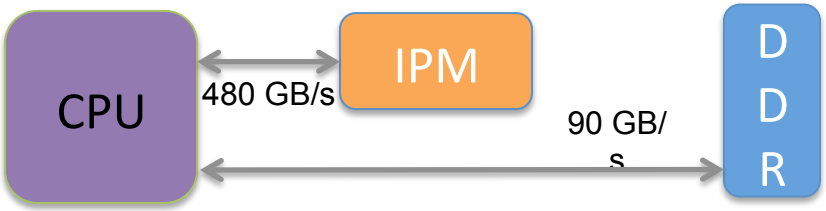
Memory Modes - IPM and DDR

Selected at node boot time

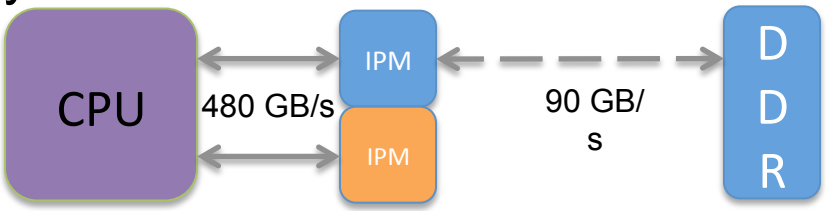
Cache



Flat



Hybrid



- **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

Section:
Theta Software Environment

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

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**

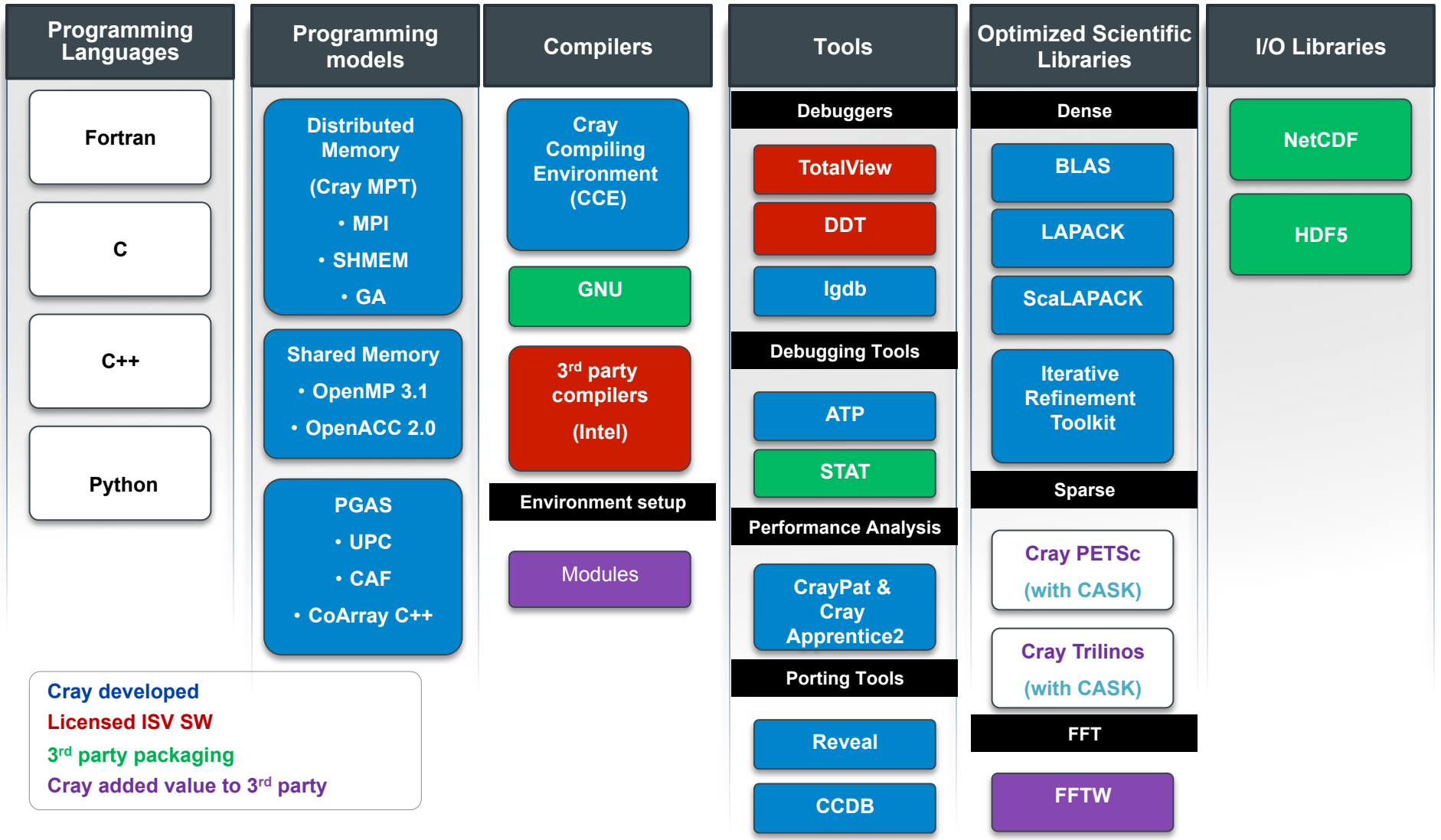
Modules

- 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*

Compilers

- 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

Cray Programming Environment



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.

Section:
Cooley HW and SW Environment

Cooley

Cooley serves as an analysis and visualization resource for projects

- x86+GPU system with 126 compute nodes
- 293 TF peak performance
- Each compute node has
  - two 2.4 GHz Intel Haswell processors (6 cores per CPU, 12 cores total)
  - one NVIDIA Tesla K80 (with two GPUs)
  - 384GB CPU RAM, 12GB RAM per GPU
  - 345GB local scratch space
- FDR Infiniband interconnect
- Mira's GPFS and Theta's Lustre project filesystems mounted

Softenv

- Cooley uses **softenv** instead of **modules**
  - Softenv is similar to modules
  - Keys are read at login time to set environment variables like PATH.
  - Mira, Cetus, Vesta: ~/.soft
  - Cooley: ~/.soft.cooley
- To get started:

```
# Select latest version of mvapich2 with GNU compilers
+mvapich2
@default
# the end – do not put any keys after the @default
```

- After edits to .soft, type "**resoft**" or log out and back in again
- Type "**softenv**" to see list of all available keys

Compilers

- Choose compiler via softenv keys

- Non-MPI
  - GNU: +gcc-4.8.1 (gcc, g++, gfortran)
  - Intel: +intel-composer-xe (icc, ifort)
  - Clang: @clang (clang)

- MPI compiler wrappers
  - mvapich (mpicc, mpicxx, mpifort)
    - GNU: +mvapich2
    - Intel: +mvapich2-intel
    - Clang: @mvapich2-clang (no mpifort)
  - mpich (mpicc, mpicxx, mpif77, mpif90)
    - GNU: +mpich2-1.4.1p1
    - Intel: +mpich2-1.4.1p1-intel

Questions?

Part 2:
Queuing & Running

Section:
Theta Job Submission

Theta Job script

```
#!/bin/bash
#COBALT -t 10
#COBALT -n 2
#COBALT -A Myprojectname
#COBALT --attrs mcdram=cache:numa=quad

# Various env settings are provided by Cobalt
echo $COBALT_JOBID  $COBALT_PARTNAME  $COBALT_JOBSIZE

aprun -n 16 -N 8 -d 1 -j 1 --cc depth ./a.out
status=$?

# could do another aprun here...

exit $status
```

Aprun overview

- Options
  - -n *total_number_of_ranks*
  - -N *ranks_per_node*
  - -d *depth* [number of cpus (hyperthreads) per rank]
  - --cc **depth** [Note: **depth** is a keyword]
  - -j *hyperthreads* [cpus (hyperthreads) per compute unit (core)]

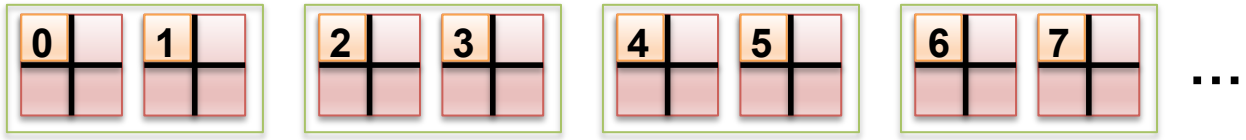
- Env settings you may need
  - -e OMP_NUM_THREADS=*nthreads*
  - -e KMP_AFFINITY=...

- See also **man aprun**

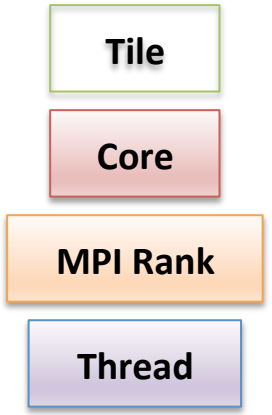
Theta - aprun Overview

- https://www.alcf.anl.gov/user-guides/running-jobs-xc40
  - Theta's KNL nodes have 32 tiles with 2 cores each (4 hardware threads per 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 <app> <app_args>



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}
nname= nid02937 rnk= 5 tid= 0 ht= {5}

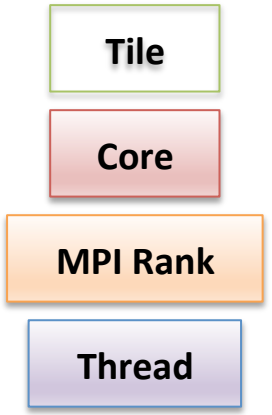


Theta - aprun Overview

- https://www.alcf.anl.gov/user-guides/running-jobs-xc40
  - Theta's KNL nodes have 32 tiles with 2 cores each (4 hardware threads per core)
  - Example #2: 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 <app> <app_args>

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}
nname= nid02937 rnk= 1 tid= 1 ht= {3}



Submitting a Cobalt job

- qsub –A <project> -q <queue> -t <time> -n <nodes> ./jobscript.sh

  E.g.

  qsub –A Myprojname –q debug –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"
- Without "-A", uses environment variable COBALT_PROJ if set
  - export COBALT_PROJ=ALCF_Getting_Started
- **man qsub** for more options

Theta queues and modes

- MCDRAM and NUMA modes can only be set by the system when nodes are rebooted. *Users cannot directly reboot nodes.*
- Submit job with the --attrs flag to get the mode you need. E.g.
  - qsub −n 32 −t 60 −attrs mcdram=cache:numa=quad ./jobscript.sh
  - Other mode choices
    - mcdram: cache, flat, split, equal
    - numa: quad, a2a, hemi, snc2, snc4
  - *When nodes require rebooting, job may be in* **starting** *state for ~15 min*
- Queues
  - Normal jobs use queue named "default"
  - Debugging: debug-cache-quad, debug-flat-quad
    - Note: pre-set for mcdram/numa configuration
  - "qstat −Q" lists all queues

Section:
Theta Job Management and Debugging

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
- *When nodes require rebooting, job may be in* **starting** *state for ~15 minutes*
- qdel <jobid>
- showres – show reservations currently set in the system
- **man qstat** for more options

Machine status web page

Running Jobs

Starting

Queued Jobs

Reservations

http://status.alcf.anl.gov/theta/activity (a.k.a. The Gronkulator)

Cobalt files 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”
  - jobid can be inserted into your string e.g. "-O myprefix_$jobid"

- **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

Interactive job

- Useful for short tests or debugging
- Submit the job with –I (letter I for Interactive)
  - Default queue and default project
    - qsub –I –n 32 –t 30
  - Specify queue and project:
    - qsub –I –n 32 –t 30 –q R.GSV2018 –A ALCF_Getting_Started
- 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**

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.

Allocation Management

- Every user must be assigned to at least one project:
- Projects are given allocations:
  - Allocations have an amount, start, and end date, and are tracked separately; Charges will cross allocations automatically. The allocation with the earliest end date will be charged first, until it runs out, then the next, and so on.
- NEW: Use '**sbank**' command to query allocation, balance:
  - sbank allocation –p <projectname> # show balance of the project
  - sbank-list-users -p <projectname> –u <user> # charges against this project by this user
  - sbank-list-allocations -S geYYYY-MM-DD # allocations with start greater or equal to date
  - Other useful options:
    - -r <resource> : show results for a specific computer resource, default is current login
    - -E lt<DATE> : show info before this date
    - -S ge<DATE1> -E lt<DATE2> : show info for DATE1 =< date < DATE2
    - -h : list of commands with numerous examples

  Note: sbank is updated once an hour.
- Charges are based on the node count, **NOT the number of cores used**!

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

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)

- Notes on linking your program
  - PrgEnv-cray links everything necessary by default
  - PrgEnv-intel
    - Link with **-Wl,-T/opt/cray/pe/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
  - More info at
    - https://www.alcf.anl.gov/files/Loy-comp_perf_workshop-debugging-2018-v2-template_update.pdf
    - https://www.alcf.anl.gov/files/Hulguin_Arm_Forge_DDT_MAP.pdf

Section:
Cooley Job Submission

Cooley Job Script

- Job script similar to Theta except mpirun instead of aprun
  - Example test.sh:

```
#!/bin/sh
NODES=`cat $COBALT_NODEFILE | wc -l`
PROCS=$((NODES * 12))
mpirun -f $COBALT_NODEFILE -n $PROCS myprog.exe
```

  - Submit on 5 nodes for 10 minutes

```
qsub -q default -n 5 -t 10 -A yourprojectname ./test.sh
```

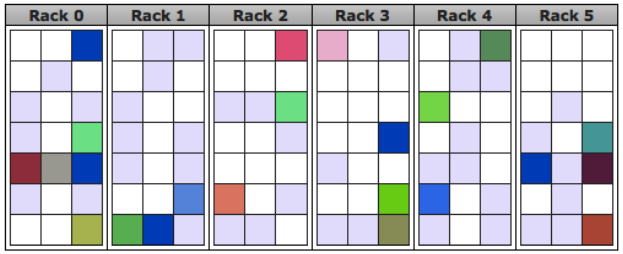
  - Queues
    - **default** for large/long jobs, **debug** for short/small jobs
    - use **pubnet** to get public network visibility
    - use **nox11** queues to suppress X server for CUDA jobs
  - Refer to online user guide for more info

Machine status web page

Argonne NATIONAL LABORATORY | Leadership Computing Facility

Cooley Activity

Home | Cooley | Activity



Running Jobs | Queued Jobs | Reservations

Total Running Jobs: 19

Job Id	Project	Run Time	Walltime	Location
493989	Vendor_Support	07:09:45	12:00:00	cc005,cc007,[cc009-010],cc016,cc018,[cc023-024],cc026,cc028,cc031,[cc033-034],
494070	SDAV	04:52:08	10:00:00	cc084
494093	OSCon	04:38:38	11:59:00	cc021
494092	OSCon	04:38:27	11:59:00	cc039
494095	OSCon	04:38:16	11:59:00	cc045
494096	OSCon	04:38:05	11:59:00	cc091
494097	OSCon	04:37:54	11:59:00	cc117
494094	OSCon	04:37:43	11:59:00	cc126
494098	OSCon	04:37:32	11:59:00	cc040
494099	OSCon	04:37:22	11:59:00	cc081
494100	OSCon	04:37:11	11:59:00	cc064
494101	OSCon	04:37:00	11:59:00	cc120
494156	SDAV	04:00:01	07:00:00	cc012,cc051
494182	visualization	03:29:09	10:00:00	cc003,cc015,cc041,cc075,cc118
494256	OSCon	00:37:46	11:59:00	cc014
494261	OSCon	00:17:03	11:59:00	cc058
494260	OSCon	00:16:41	11:59:00	cc100
494388	visualization	00:09:54	00:10:00	cc087
494389	visualization	00:09:43	00:10:00	cc013

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

Section:
Live demos

Hands-on session

- On Theta or Cooley, copy examples from
  - /projects/ALCF_Getting_Started/examples

- Today's event has its own project and queue names
  - Theta: -A ALCF_Getting_Started -q R.GSV2018
  - Cooley: -A ALCF_Getting_Started -q training

- Theta/Cooley: compile & submit
  - make
  - qsub submit.sh

Hands-on session

- **Compilation** example:
  - Create directory, copy files, and compile program:

    > mkdir -p ~/training/Hands-on
    > cp -r /projects/ALCF_Getting_Started/examples/* ~/training/Hands-on
    > cd ~/training/Hands-on/theta/compilation
    > ls
    > cat hellompi.c
    > cat Makefile
    > cat submit.sh
    > make

  - Submit job and check output:

    > qsub submit.sh
    > qstat -u <username>
    > cat <JobID>.output

  - qsub echoes a number to the screen, which is the Cobalt job id. In the absence of a -o argument, three files are created (say JobID was 227076):

    227076.cobaltlog, 227076.error, 227076.output *(replaced by hellompi.output with -o)*

Hands-on session: Theta

- Example of an **OpenMP** job submission:
  - Change to directory, compile, and submit:

    > cd ~/training/Hands-on/theta/omp

    > make

    > qsub submit.sh

    ***NOTE**: remember that thread affinity is controlled by aprun settings (see slides 22-24 for reference).*

- Example of **Python** job submission:
  - Change to directory, compile, and submit:

    > cd ~/training/Hands-on/theta/python

    > qsub submit.sh

    ***NOTE**: examine submit.sh script for loading python environment on Theta.*

Hands-on session: Cooley

- Example of an **OpenMP** job submission:
  - Change to directory, compile, and submit:

    > cd ~/training/Hands-on/cooley/omp

    > make

    > qsub submit.sh

- Example of **Python** job submission:
  - Edit your ~/.soft.cooley and add "+anaconda" just before **@default** (Note: @default should be the last line in the file)
  - Update your environment to include python paths:

    > resoft

  - Change to directory, compile, and submit:

    > cd ~/training/Hands-on/cooley/python

    > qsub submit.sh

Section:
Wrap-up / Getting Help

When things go wrong… logging in

- Check to make sure it's not a scheduled system maintenance day:
  - Login nodes on ALCF systems are often closed off during system maintenance to allow for activities that would impact users.
  - Look for reminders in the weekly maintenance announcement to users and in the pre-login banner message.
  - An all-clear email will be sent out to users at the close of maintenance.
- Remember that CRYPTOCard passwords:
  - Require a pin at the start
  - Are all hexadecimal characters (0-9, A-F). Letters are all **UPPER CASE**.
- On failed login, try in this order:
  - Try typing PIN + password again (without generating new password)
  - Try a different ALCF host to rule out login node issues (e.g., maintenance)
  - Push CRYPTOCard button to generate a new password and try that
  - Walk through the unlock and resync steps at: http://www.alcf.anl.gov/user-guides/using-cryptocards#troubleshooting-your-cryptocard
  - Still can't login?
    - Connect with **ssh -vvv** and record the output, your IP address, hostname, and the time that you attempted to connect.
    - Send this information in your e-mail to support@alcf.anl.gov

Getting help

Online resources (24/7):

- ALCF web pages:
  - -- Theta: https://www.alcf.anl.gov/user-guides/xc40-system-overview
  - – Cooley: https://www.alcf.anl.gov/user-guides/cooley
  - – Accounts: https://accounts.alcf.anl.gov

Contact us:

e-mail: **support@alcf.anl.gov**

ALCF Help Desk:

Hours: Monday – Friday, 9 a.m. – 5 p.m. (Central time)
Phone: **630-252-3111** or **866-508-9181** (toll-free, US only)

Your Catalyst

News from ALCF:

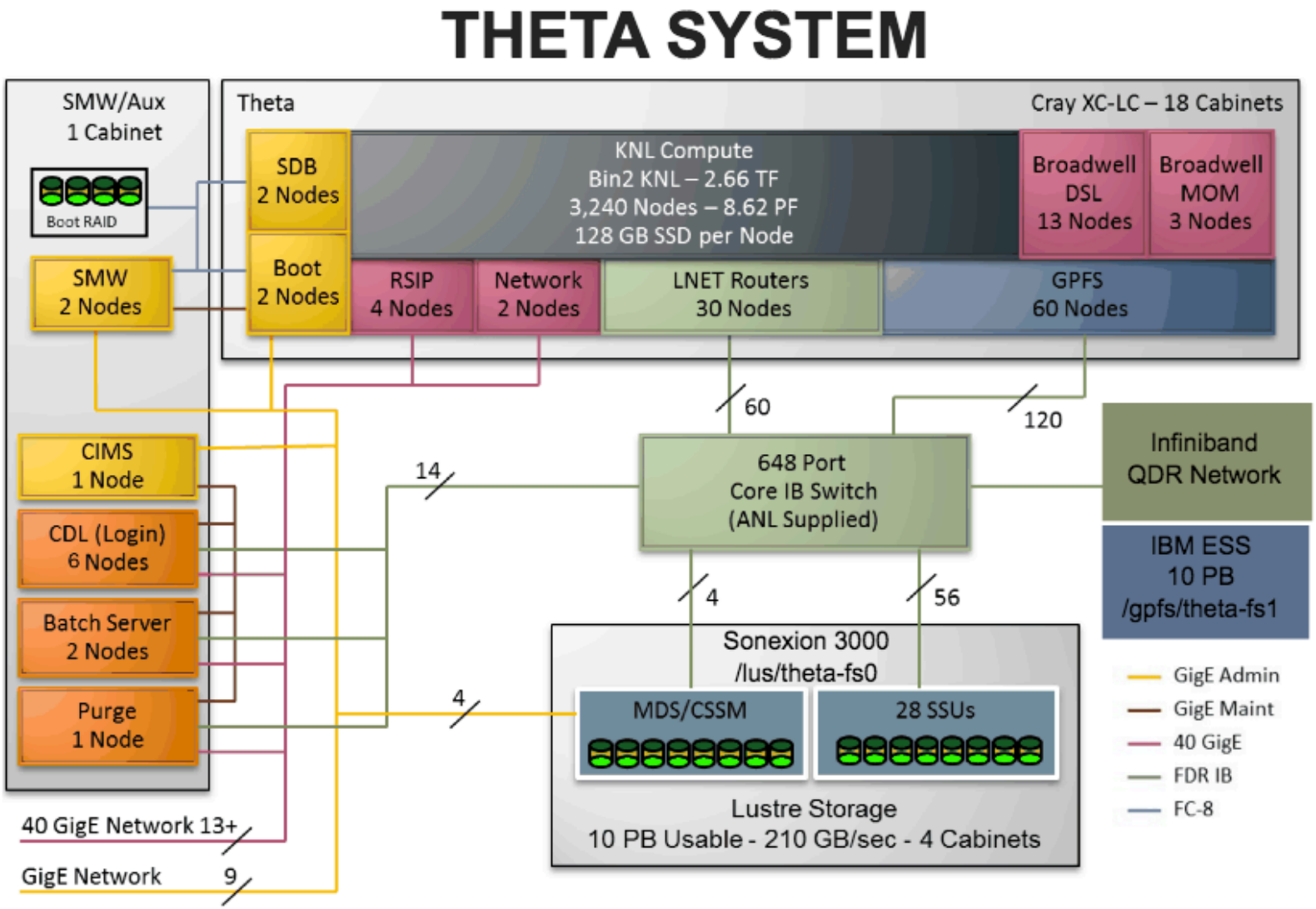
- ALCF Weekly Updates, ALCF newsletters, e-mail via *theta-notify* and *cooley-notify* lists.

Argonne Leadership Computing Facility
Theta/Cooley Quick Start

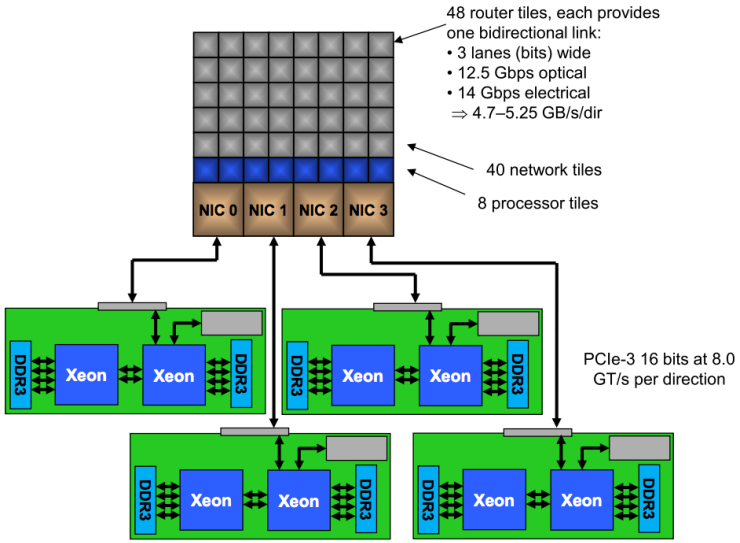
Thank you for attending!

SUPPLEMENTAL TOPICS

Theta System

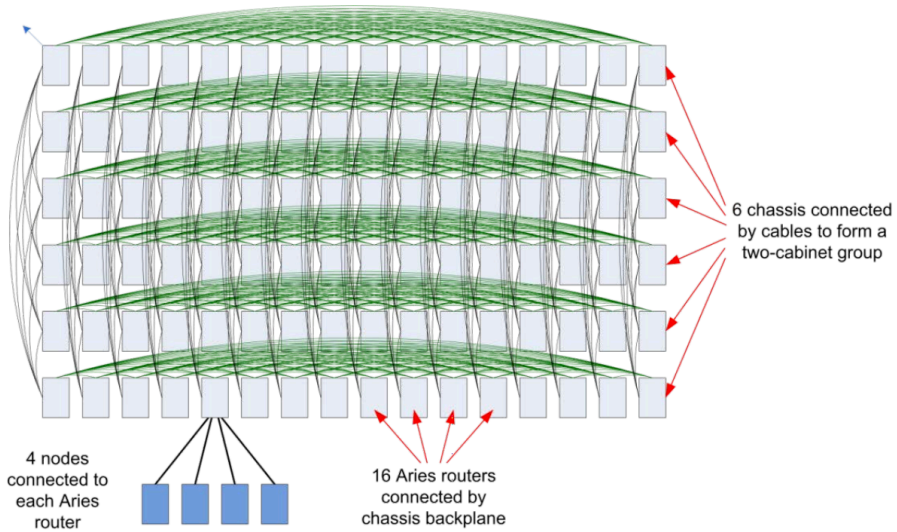


Aries Dragonfly Network



Aries Router:

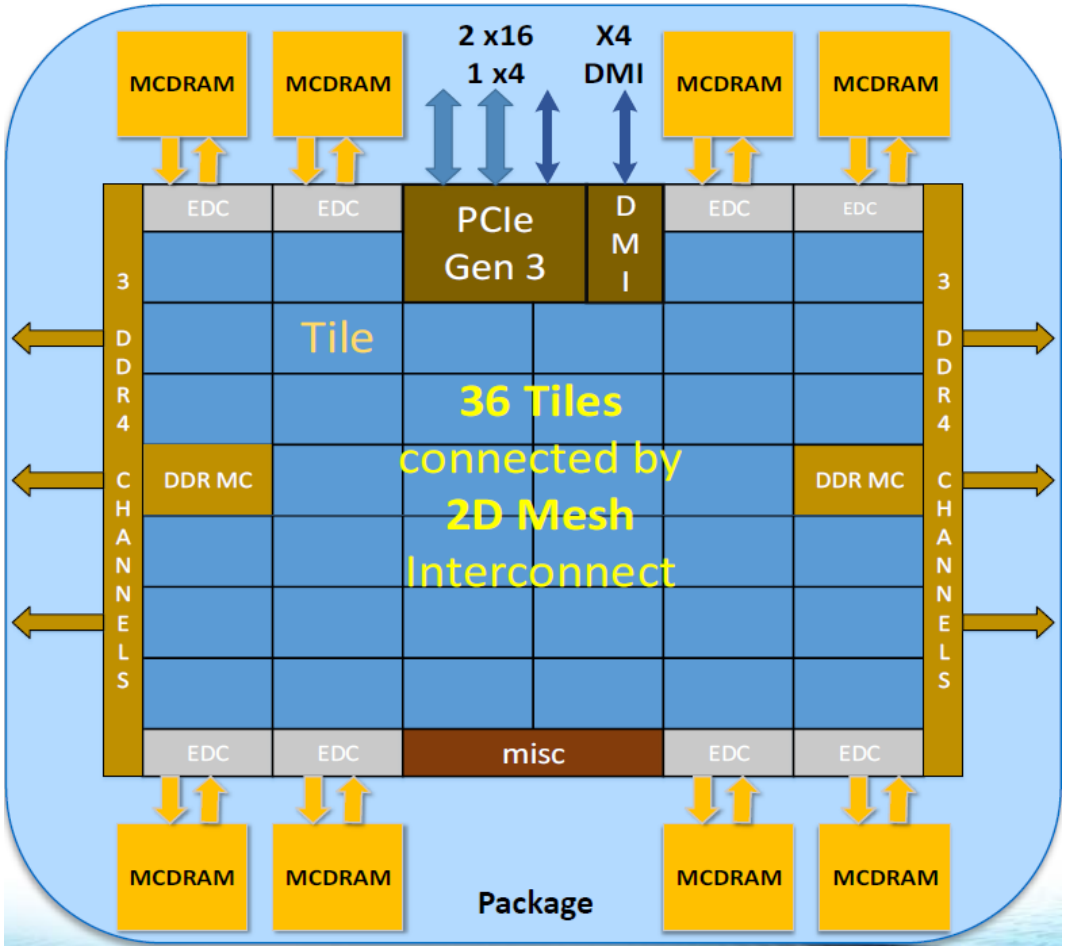
- 4 NIC's connected via PCIe
  - 40 Network tiles/links
  - 4.7-5.25 GB/s/dir per link



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

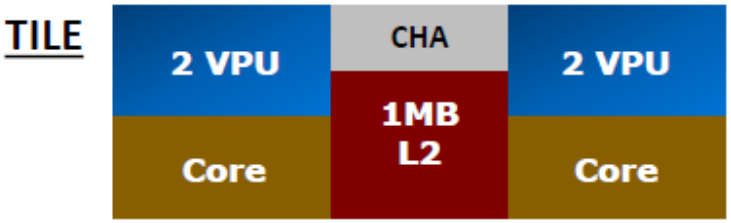
2D Mesh Interconnect
- Tiles connected by 2D mesh

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

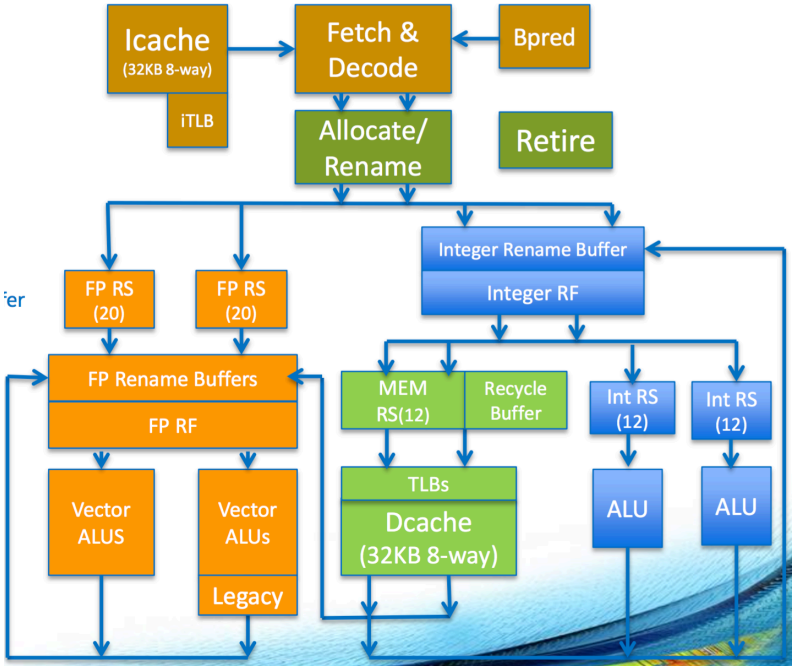


Tile

- Two CPUs
- 2 VPUs per core
- Shared 1 MB L2 cache (not global)

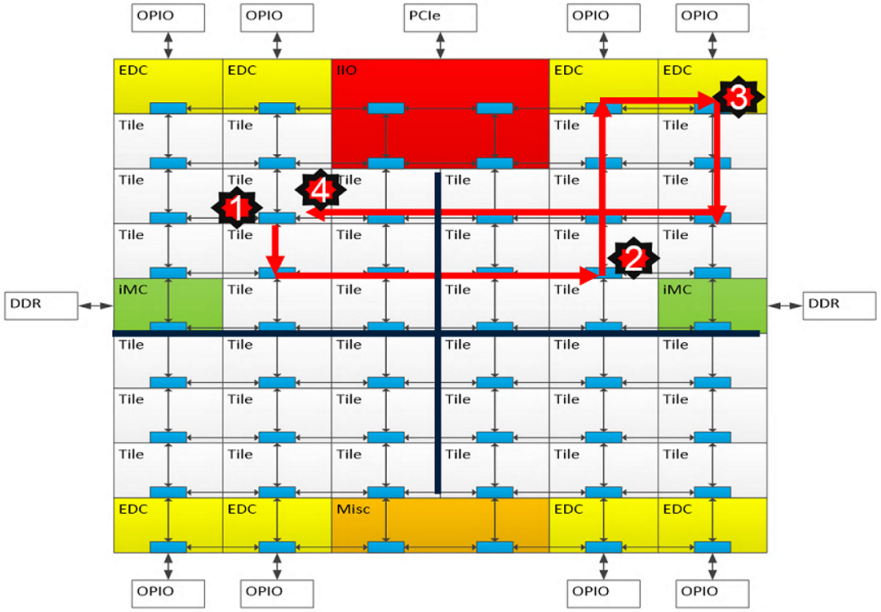
Core

- Based on Silvermont (Atom)
  - Functional units:
    - 2 Integer ALUs
    - 2 Memory units
  - 2 VPU's with AVX-512
  - Instruction Issue & Exec:
    - 2 wide decode

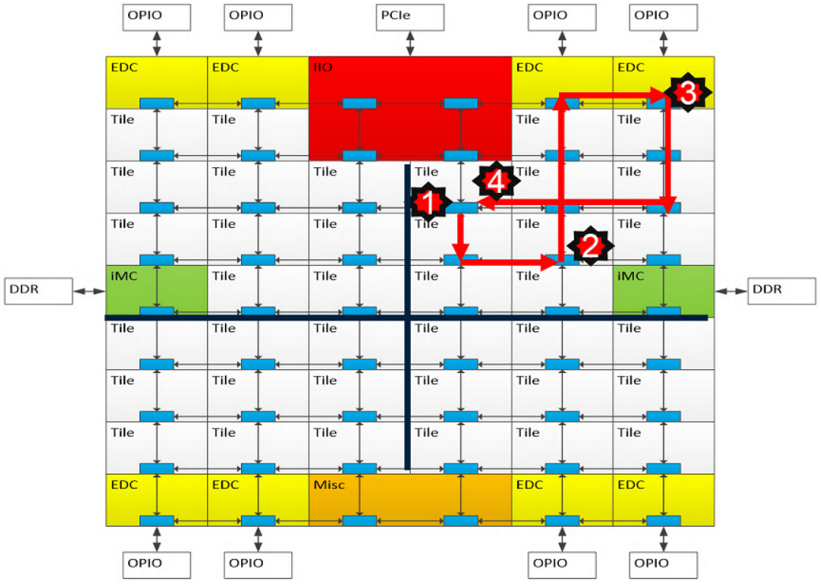


Clustering Modes

Quadrant Mode



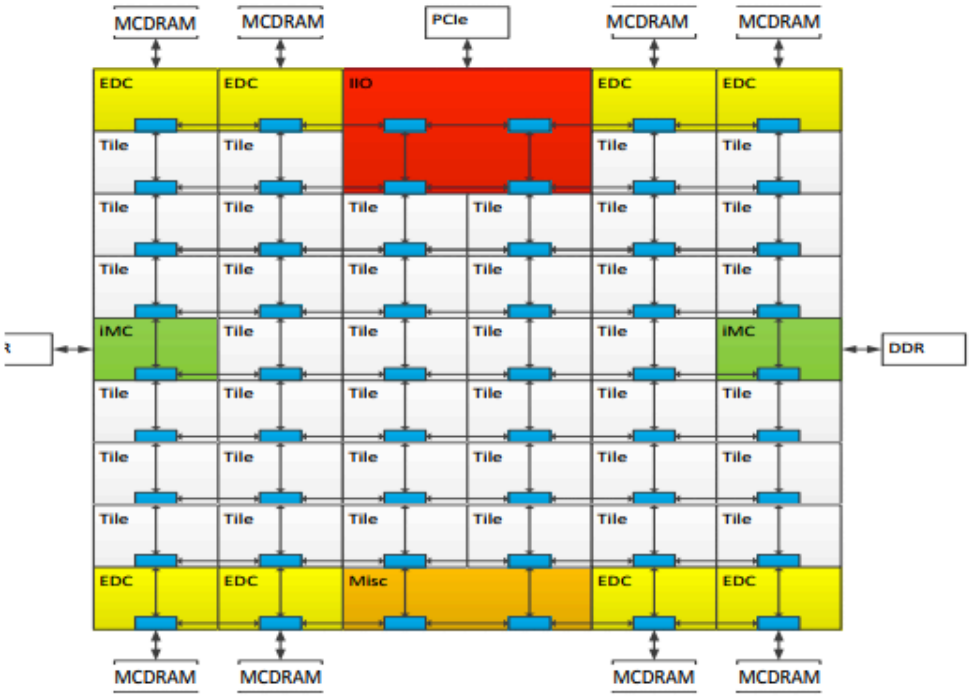
SNC-4 Mode



1. Tile has Cache Miss
2. CHA selected
3. Memory Controller
4. Memory Received

Knights Landing overview

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

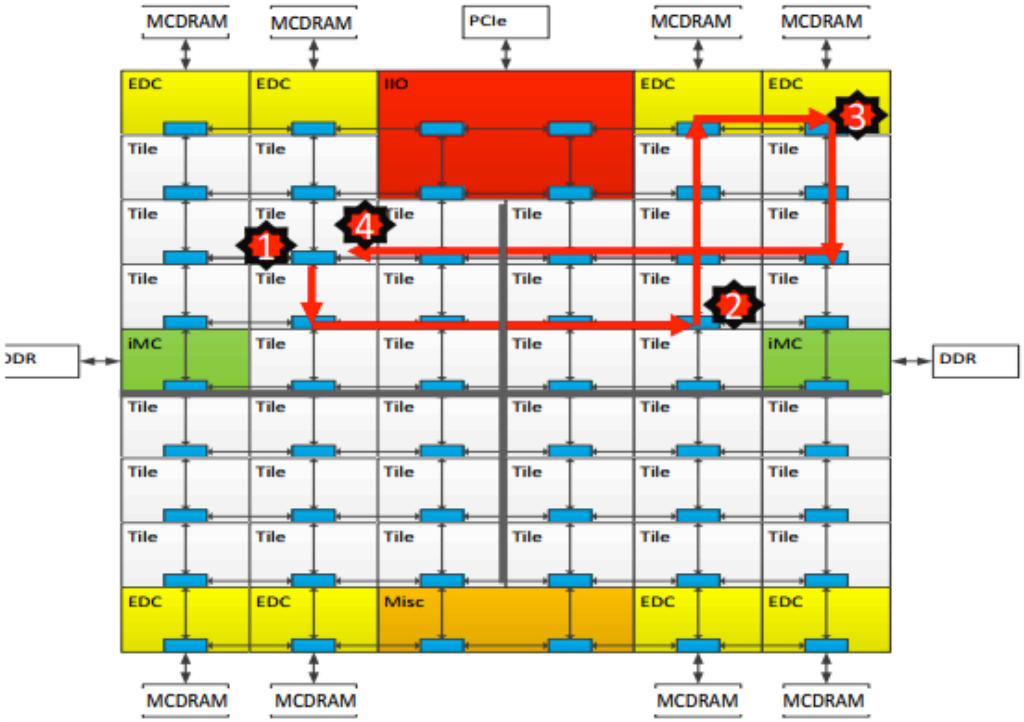
- MESIF protocol (F = Forward)
- Distributed directory to filter snoops

Three Cluster Modes

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

Knights Landing overview

Cluster Mode: Quadrant



Chip divided into four virtual Quadrants

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

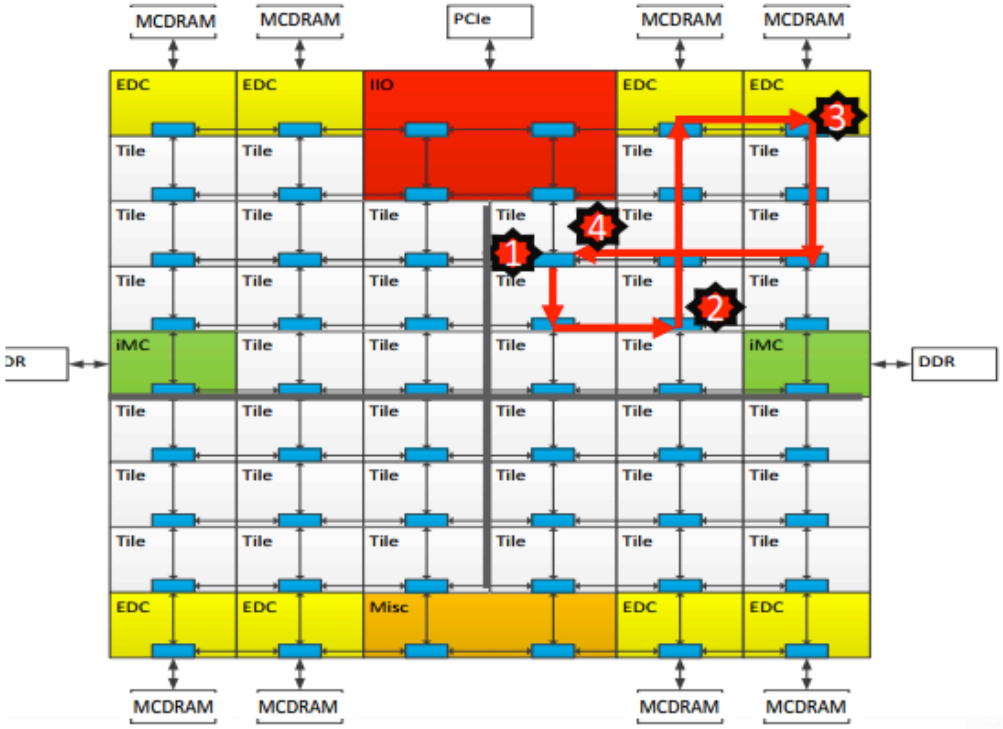
Affinity between the Directory and Memory

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

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

Knights Landing overview

Cluster Mode: Sub-NUMA Clustering (SNC)



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

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.

Knights Landing overview

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))
```

Multi-channel DRAM overview

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
- Option C: Direct OS system calls
  - mmap(1), mbind(1)
  - Not the preferred method
    - Page-only granularity, OS serialization, no pool management

Multi-channel DRAM overview

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

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

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)
  - reservations are sometimes idle, but still block other users' jobs from running on a partition
  - should be the exception not the rule

- 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