Cray XC Series Application Programming and Optimization
Student Guide

TR-CPO NERSC

February 12 and 13, 2019

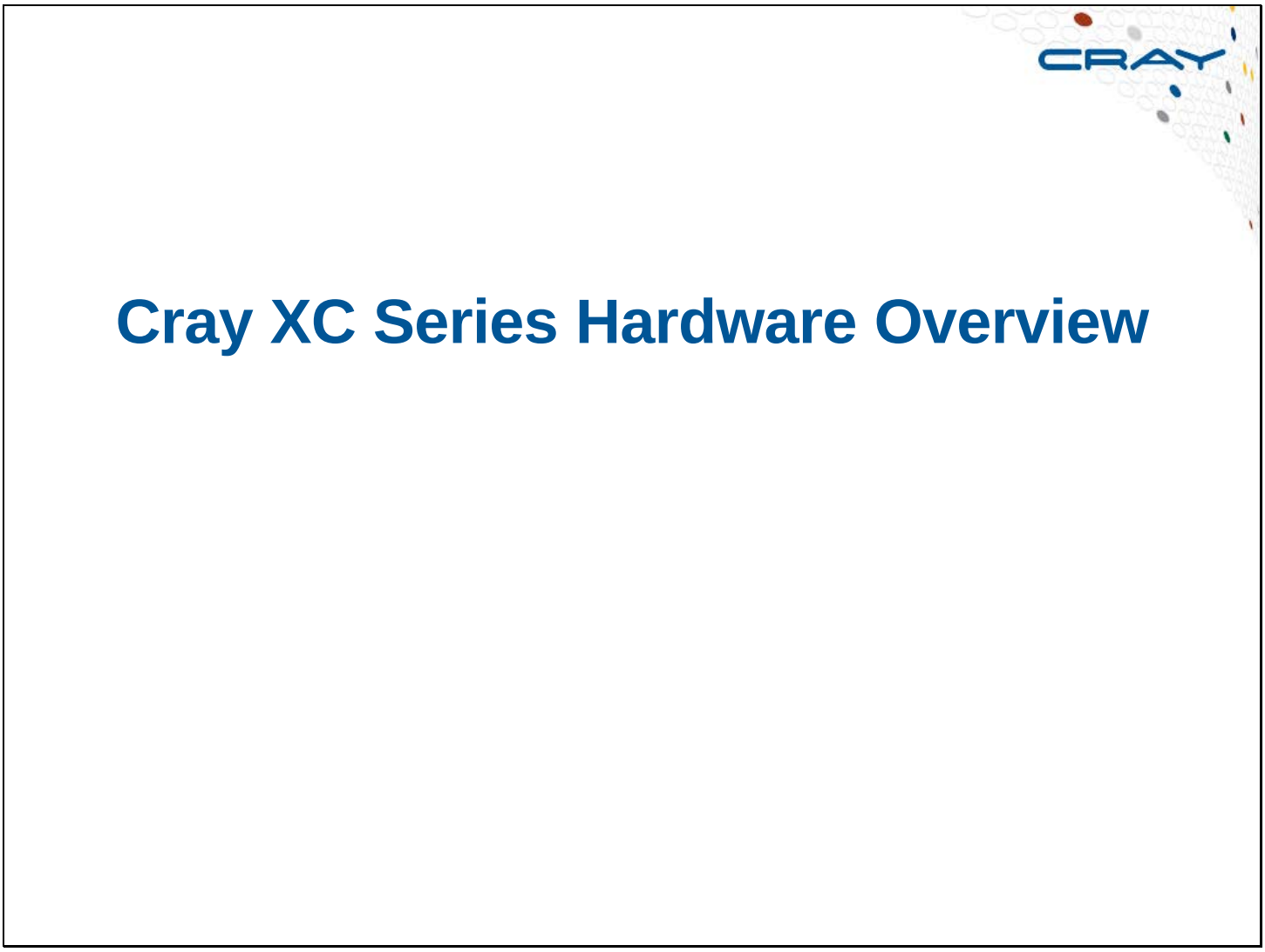
This document is intended for instructional purposes.
Do not use it in place of Cray reference documents

Cray Private

Direct comments about this publication to:

Mail: Cray Inc.
Cray Training
P.O. Box 6000
Chippewa Falls, WI 54729-0080
USA

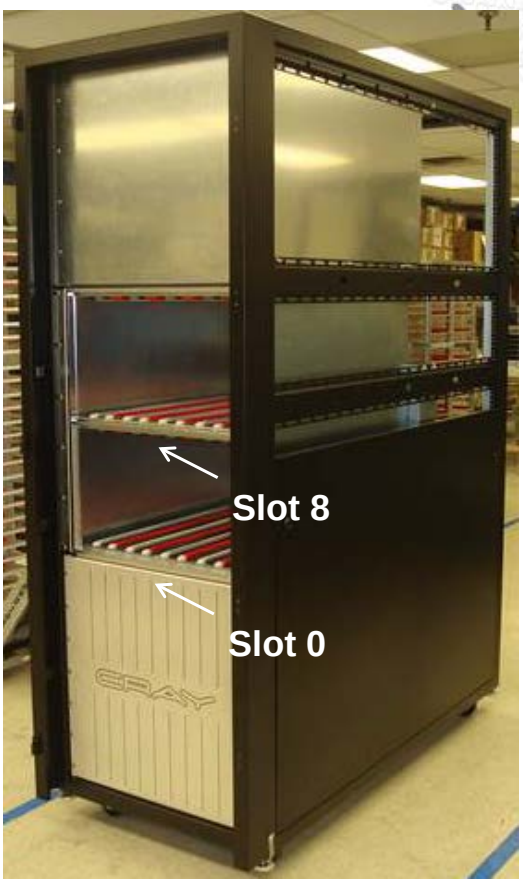
E-mail: ttd_online@cray.com
Fax: +1 715 726 4991



Cray XC Series Hardware Overview

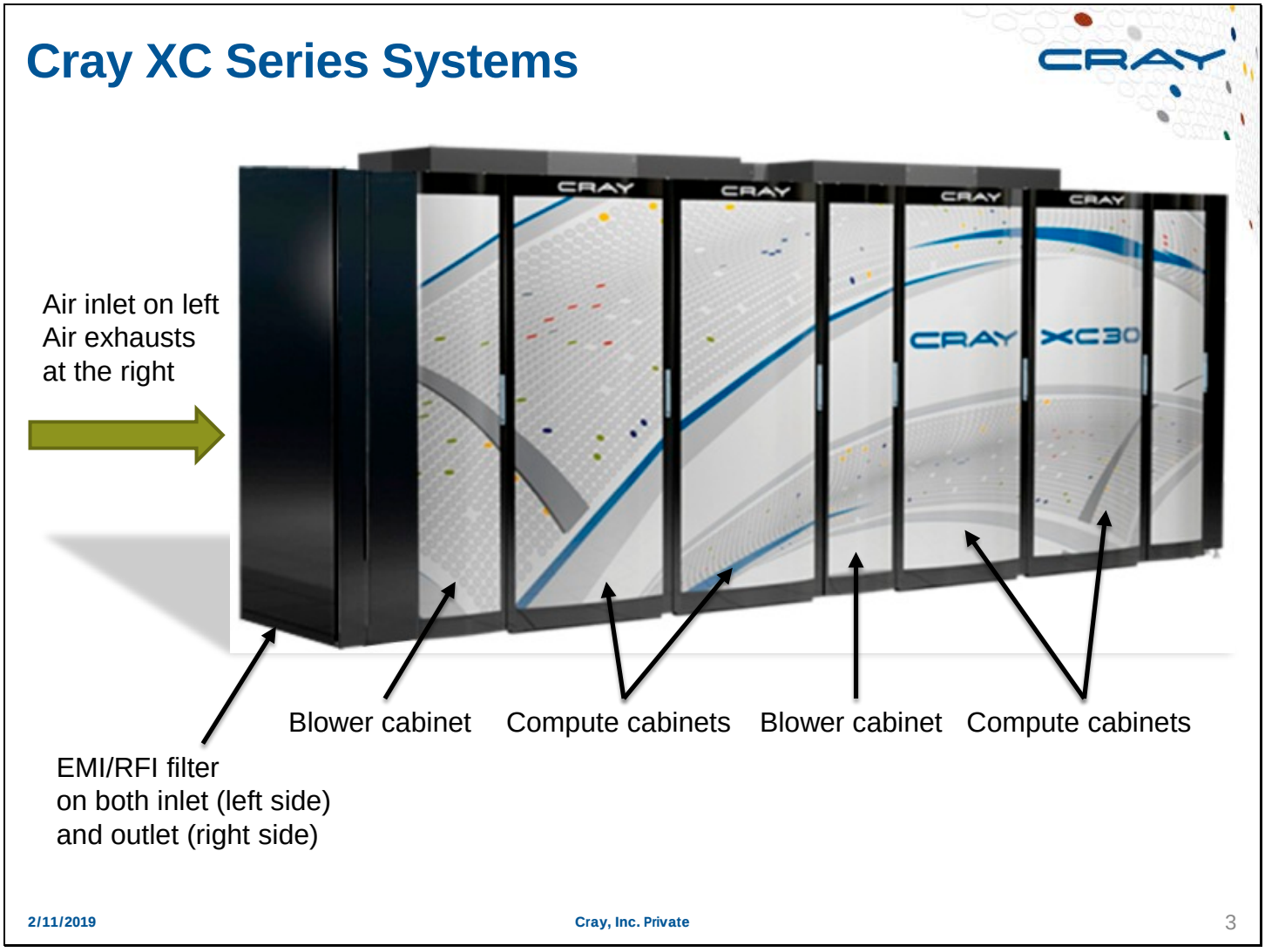
Cray XC-AC System

- **Cray XC-AC (air-cooled) system**
  - Single XC chassis
    - The chassis is rotated 90 degrees counter-clockwise from its orientation in a Cray XC series system
      - Modules stand vertically in cabinet
    - Input power 208VAC or 480VAC
    - Passive Electrical cables (PEC) used in place of the Active Optical Cables (AOCs)
    - Minimum configuration of 4 blades
      - System is expanded by blades
  - Airflow is vertical
    - A single blower in the bottom of the cabinet
  - Maximum of 8 cabinets
    - Cabinets are arranged in a single row



2/11/2019 Cray, Inc. Private 2

The Cray XC Series systems are assumed to be a Liquid-cooled system unless noted by the –AC notation; for example, Cray XC30-AC, Cray XC40-AC, or Cray XC50-AC system. The supported Cray XC–AC configurations range from 1 to 8 cabinets; each cabinet contains a single chassis. The chassis is the same in the XC and XC-AC systems, but are installed 90-degrees counter-clockwise from the chassis orientation in the Cray XC systems.

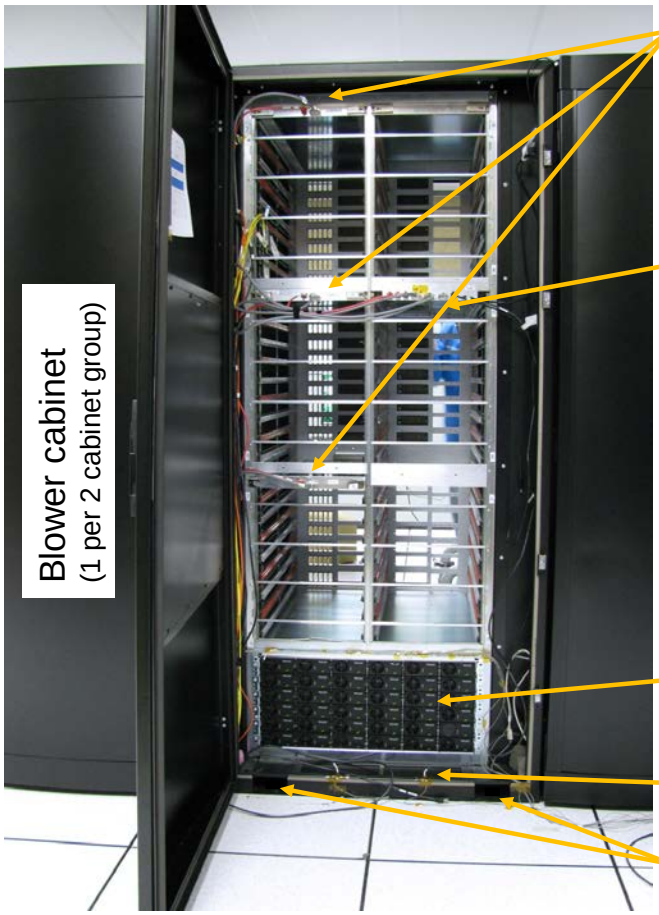


Single air inlet (left side) and exhaust (right side) per row. An optional preconditioning coil (located on inlet side of the first blower cabinet in each row) relaxes inlet air requirements. The exhaust air is conditioned (cooled) air. The exhaust air can be room neutral depending on the environmental conditions.

Cabinet

- **Three chassis per cabinet**
  - Modules install in the left and right columns of the chassis
  - Compute blades are built for left or right column insertion
  - I/O blades are built for left column insertion only
    - I/O blades are distributed throughout the system
      - Normally occupy the lower slots of the chassis
  - If the cabinet was fully populated with compute blades, there would be 192 nodes
- **Cabinet configuration scales:**
  - One or two cabinets: blade level configurations are allowed
  - Three to twenty-six cabinets: last cabinet can be partially populated, but chassis must be full.
  - Twenty-seven to sixty-four cabinets: full cabinet increments.
  - Greater than sixty-four cabinets: increment in two cabinet groups.

Compute Cabinet Front View



2/11/2019 Cray, Inc. Private 5

The following components are part of the system management and do not relate to the "user environment" of the Cray System.

- The cabinet controller (CC) manages the cabinet environment.
- The chassis host module connects the cabinet controller to the backplane of the chassis. Via the backplane the cabinet controller is connected to each of the blade controllers on the each of the blades in the chassis.
- The blade controllers manage the environment of the blade the bladed controller is on. Each of the blades in the chassis can operate independent of the other blades.

Identifying Components

- **The Node ID (NID) is number that identifies each node in the system**
  - The NID is unique and reflects the node location in the network
    - The format is nidnnnnn (decimal), for example: nid00003
- **A component name (cname) or physical ID (physID) is also used**

Component	Format	Description
System	s0, p0 all	All components attached to the SMW.
Cabinet	cX-Y	Cabinet number and row; this is the cabinet controller (CC) host name
Chassis	cX-Yc#	Physical chassis in cabinet: 0 – 2, numbered from bottom to top
Blade or slot	cX-Yc#s#	Physical slot in chassis:0 – 15, numbered from lower left to upper right; this is the Blade controllers (BC) host name
Node	cX-Yc#s#n#	Node on a blade: 0 - 3 for compute,1 - 2 for service
Aries ASIC	cX-Yc#s#a#	Cray Aries ASIC on a blade: 0
Link	cX-Yc#s#a#l#	Link port of a Aries ASIC: 00 – 57 (octal)

2/11/2019 Cray, Inc. Private 6

Examples:

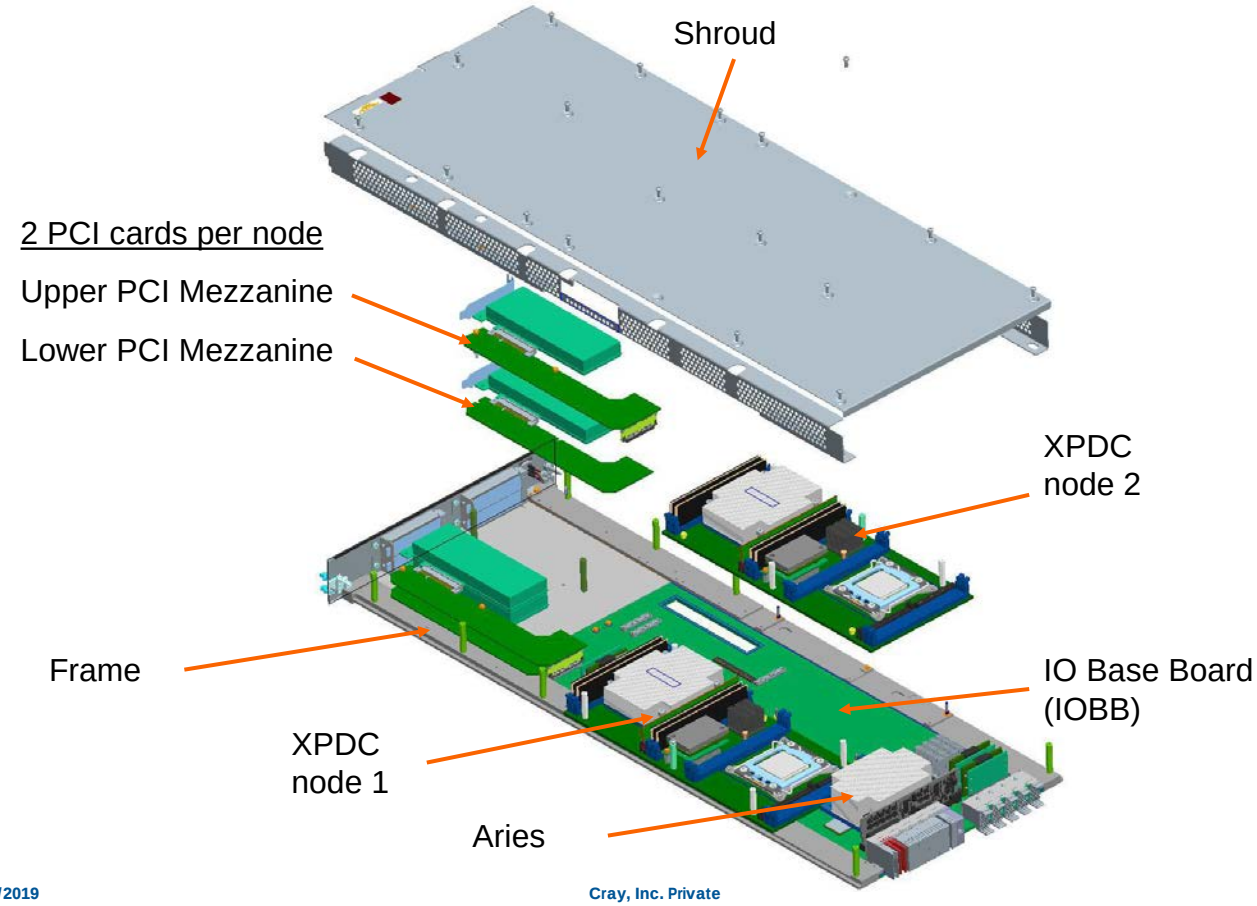
c0-0	Cabinet 0-0, X position 0 within row 0.
c12-3	Cabinet 12-3, X position 12 within row 3.
c*-*	Wildcard: All cabinets within the system.
c0-0c2	Cage 2 of cabinet 0-0.
c0-0c*	Wildcard: All cages within cabinet c0-0.
c0-0c0s4	Slot (module) 4 of cage 0 of cabinet c0-0.
c0-0c0s*	All slots (0..15) of cage 0 of cabinet c0-0.
c0-0c0s0n3	CPU 3 on module slot 0, cage 0, cabinet c0-0.
c0-0c0s0n*	All CPUs (0..3) on module slot 0, cage 0, cabinet c0-0.
c0-0c0s0a0l4	Link 4 of Aries 0 on module slot 0, cage 0, cabinet c0-0.
c0-0c0s0a0l*	Links (0..057) of Aries 0 on module slot 0, cage 0, cabinet c0-0.

Edison – Cray XC30 System

Cray XC30 and XC40 I/O Blades

- **Two nodes on a blade and each node contains:**
  - One Intel Xeon processor with up to 16 GB of DDR3 memory
    - The processor is a eight-core Xeon (Sandy Bridge)
  - A connection to a Aries ASIC
  - Voltage regulating modules (VRMs)
- **IO Base Blade (IOBB)**
  - Blade controller (BC)
  - One Aries ASIC
  - Four PCIe risers
    - Two risers per node, configured with these supported PCIe cards
      - Gigabit Ethernet
      - 10Gigabit Ethernet
      - Fibre Channel (FC2, FC4, and FC8)
      - InfiniBand
      - SAS (Serial Attached SCSI (Small Computer System Interface))
      - SSD (Solid-state Storage Device)

I/O Blade



2/11/2019 Cray, Inc. Private 9

Cray XC30 and XC40 Blades

- **The system contains these blades:**
  - I/O blades
    - 1 Aries ASIC
    - 2 nodes (nodes 1 and 2)
    - Dual-slot PCIe riser assemblies per node
    - I/O node provide system *services* such as: login, Lustre, lnet, DVS, DSL, or network
    - I/O nodes are often called “service” or SIO nodes
  - Compute blades
    - 1 Aries ASIC
    - 4 nodes
      - Nodes are:
        - CPU-CPU
        - CPU-GPU (Graphics Processing Unit, from Nvidia)
        - CPU-MIC (Many Integrated Cores, from Intel (Xeon-Phi))

More information on system service nodes:

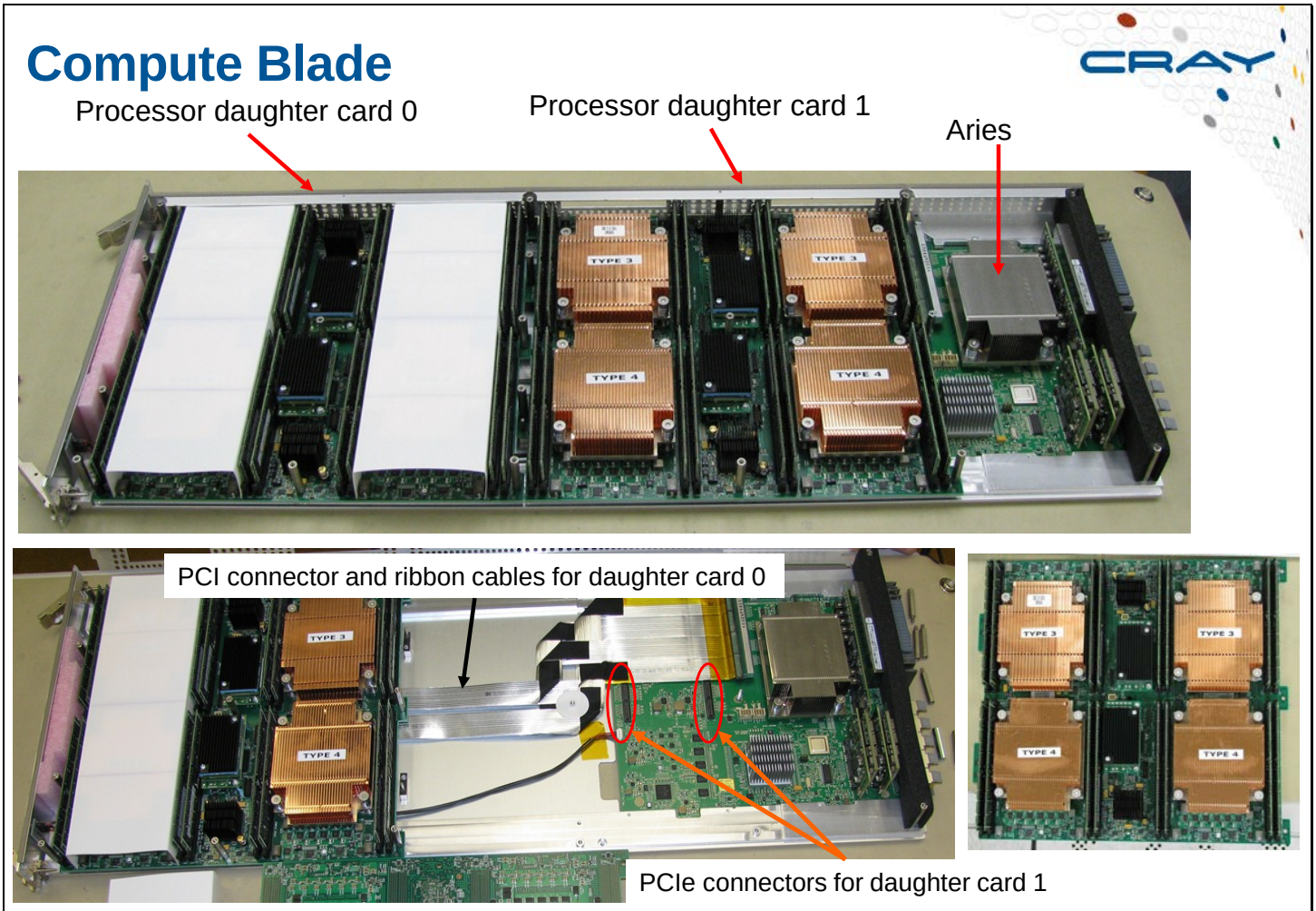
A *login* node provides user access to the system. From the login node users can edit and compile their applications and launch the application to the compute nodes. Optionally systems can be configured with *external login* nodes that perform the same function, but the node itself is external to the Cray system. The external node is a server mounted in an external rack.

Lustre nodes connect the system to a *Lustre Parallel file system*. *lnet* nodes connect the Cray system to an *external Lustre file system*. A basic difference between *lustre* nodes and *lnet* nodes is that lustre nodes are part of the lustre file system configuration and when the Cray system is shut down access to the file system also goes away. With an external lustre file system, the Lustre file system is still available while the Cray system is down.

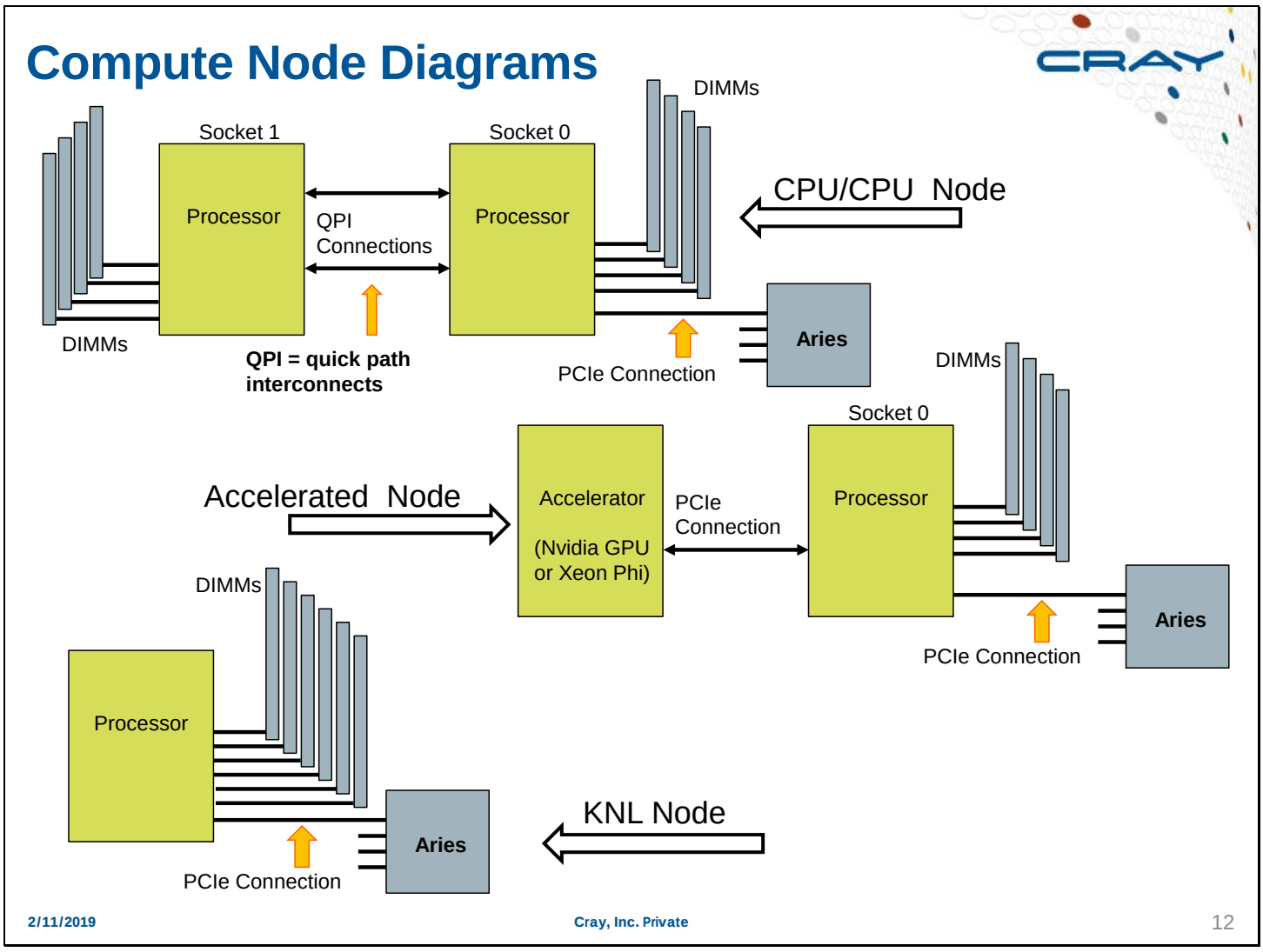
DVS (Data Virtualization Services) nodes provide access to other file systems. A DVS node has a connection to an external file system and then projects the file system across the Cray High Speed Network (HSN) to other service nodes and the compute nodes. Depending on the file system, multiple DVS nodes can be configured to project the file system to improve the bandwidth or accessibility to the file system.

DSL (Dynamic Shared Library) nodes use DVS to project the Cray’s *shared root* to the compute nodes providing dynamic access to application libraries.

Network nodes would be nodes connected to other systems to provide a dedicated access between the Cray system on the other system. These nodes don’t allow users to directly login to them, but instead perform a service for the system.



VIVOC – Vertical intermediate voltage converter
HIVOC – Horizontal IVOC
QPDC – Quad processor daughter card
HPDC – Haswell processor daughter card



The QPI (quick path interconnect) link operates at approximately 6.4 GT/s or 25.6 GB/s. The two socket configuration provides a peak bandwidth of 51.2 GB/s.

The DDR3 memory used in the XC30 systems is either PC3-12800 (1600 MHz) or PC3-14900 (1866 MHz). Each memory channel with a single DIMM per channel operates at approximately 12.8 or 14.9 GB/s peak respectively; with 4 memory channels each socket is capable of approximately 51.2 or 59.6 GB/s and a node has a peak memory bandwidth of approximately 102.4 or 119.2 GB/s respectively.

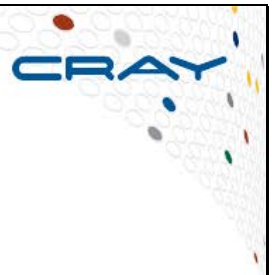
The DDR4 memory used in the XC40 systems is PC4-17000 (2133 MHz). Each memory channel with a single DIMM per channel operates at approximately 17.0 GB/s peak ; with 4 memory channels each socket is capable of approximately 68.2 GB/s and a node has a peak memory bandwidth of approximately 136.5 GB/s.

The PCIe Gen 3 interface has an approximate bandwidth of 15.75 GB/s.

Cori – Cray XC40 System

Cray XC40 KNL Blade

Xeon Phi "*Knights Landing*" Compatibility



- **Runs all existing Xeon x86 64-bit executables**
  - Linux commands
  - ISV applications
  - XC or CS applications built for Xeon processors
- **Existing Xeon-targeted libraries will work**
  - If library is not a critical compute component, recompiling not needed
  - Intel 16 MKL has AVX-512 support enabled at run time
- **Xeon executables can take advantage of all KNL features**
  - Except AVX-512 (requires recompiling)
  - Except moving selected data to MCDRAM (requires source changes)

KNL Acronym and Terminology Reference

DDR	Double Data Rate	Refers to the 6 channels of DDR4-2400 DIMM main memory
MCDRAM	Multi-Channel DRAM	High-bandwidth on-package memory
MCDRAM Cache		MCDRAM configured as a last-level memory-side cache
Flat MCDRAM		MCDRAM configured as addressable memory User-visible as a NUMA node with memory but no CPUs
EDC	Embedded DRAM Controller	Interface to MCDRAM, 8 controllers per processor
Tile		A logic block including two cores sharing an L2 cache Includes an on-chip mesh interface and CHA
CHA	Caching Home Agent	Per-tile block which manages cache coherence (L2 and MCDRAM)
MC or IMC	Integrated (DDR) Memory Controller	
OPIO	On-Package I/O	Interface from KNL processor to MCDRAM
HBM	High Bandwidth Memory	HBM is a memory hardware technology developed by AMD and partners Sometimes used informally to refer to flat MCDRAM on KNL
VPU	Vector Processing Unit	AVX-512 SIMD execution unit, 2 per core
SNC	Sub-NUMA Cluster	Processor mode which divides memory capacity and bandwidth into 2 or 4 NUMA nodes per memory type Also divides the cores and MCDRAM cache among the DDR NUMA nodes

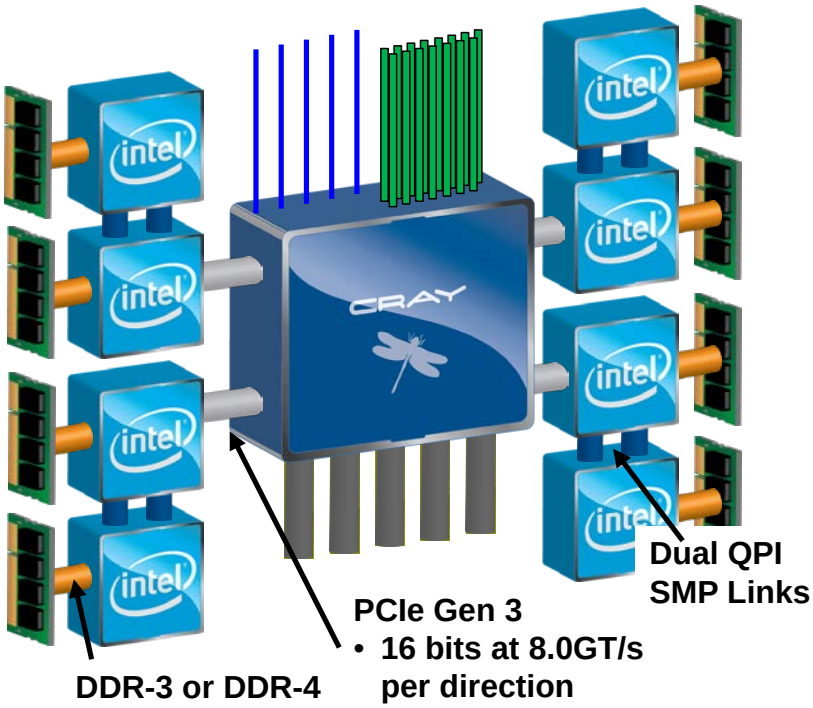
Chassis

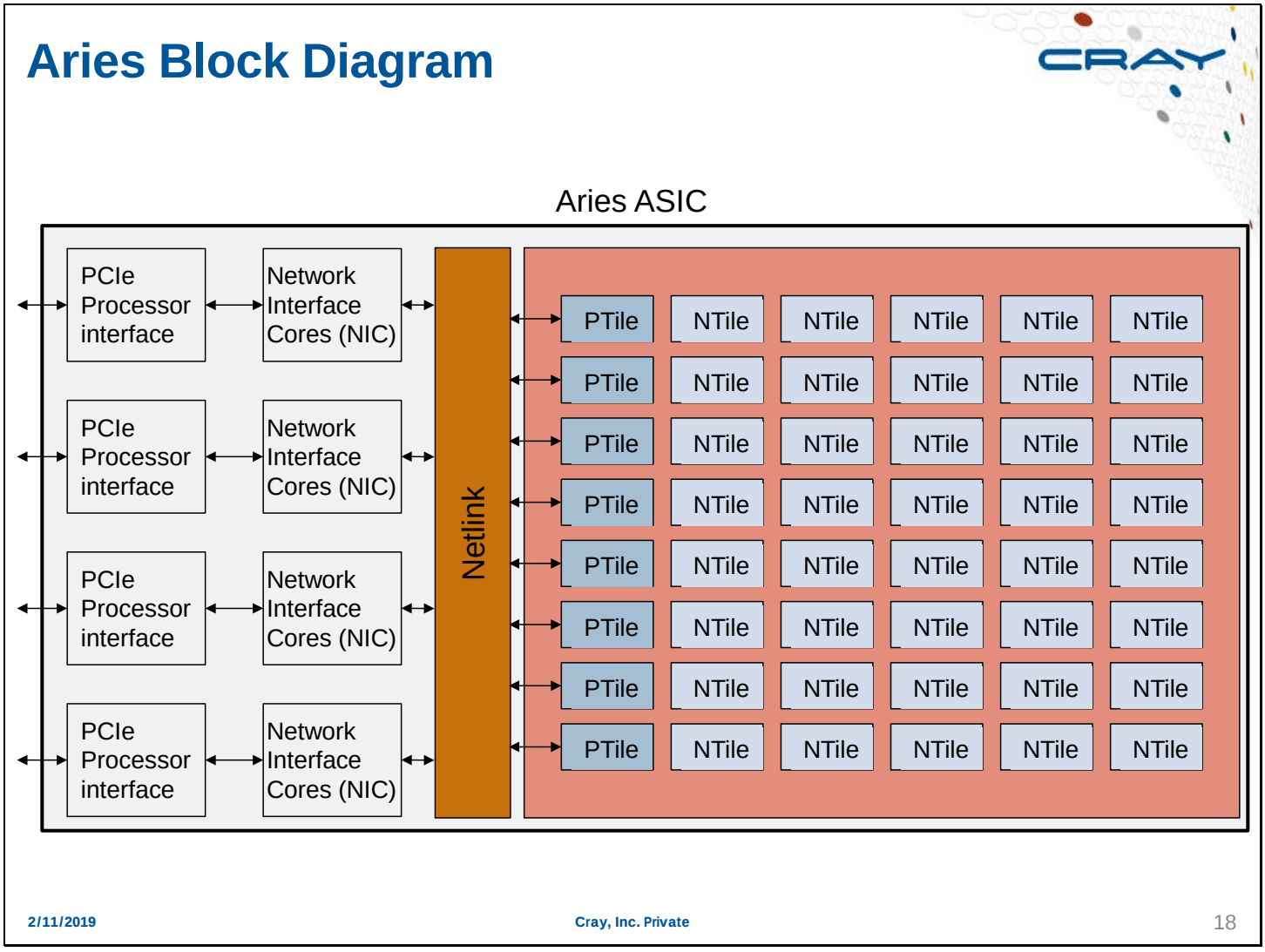
- **The chassis is the primary building block of the system**
  - The four nodes on a blade connect to a single Aries Chip
  - The 16 Aries chips in a chassis are connected via the backplane

Intra-chassis
- **Backplane**
- **15 links in backplane**
- **Rank 1 (green) Network**
- **14 Gbps**

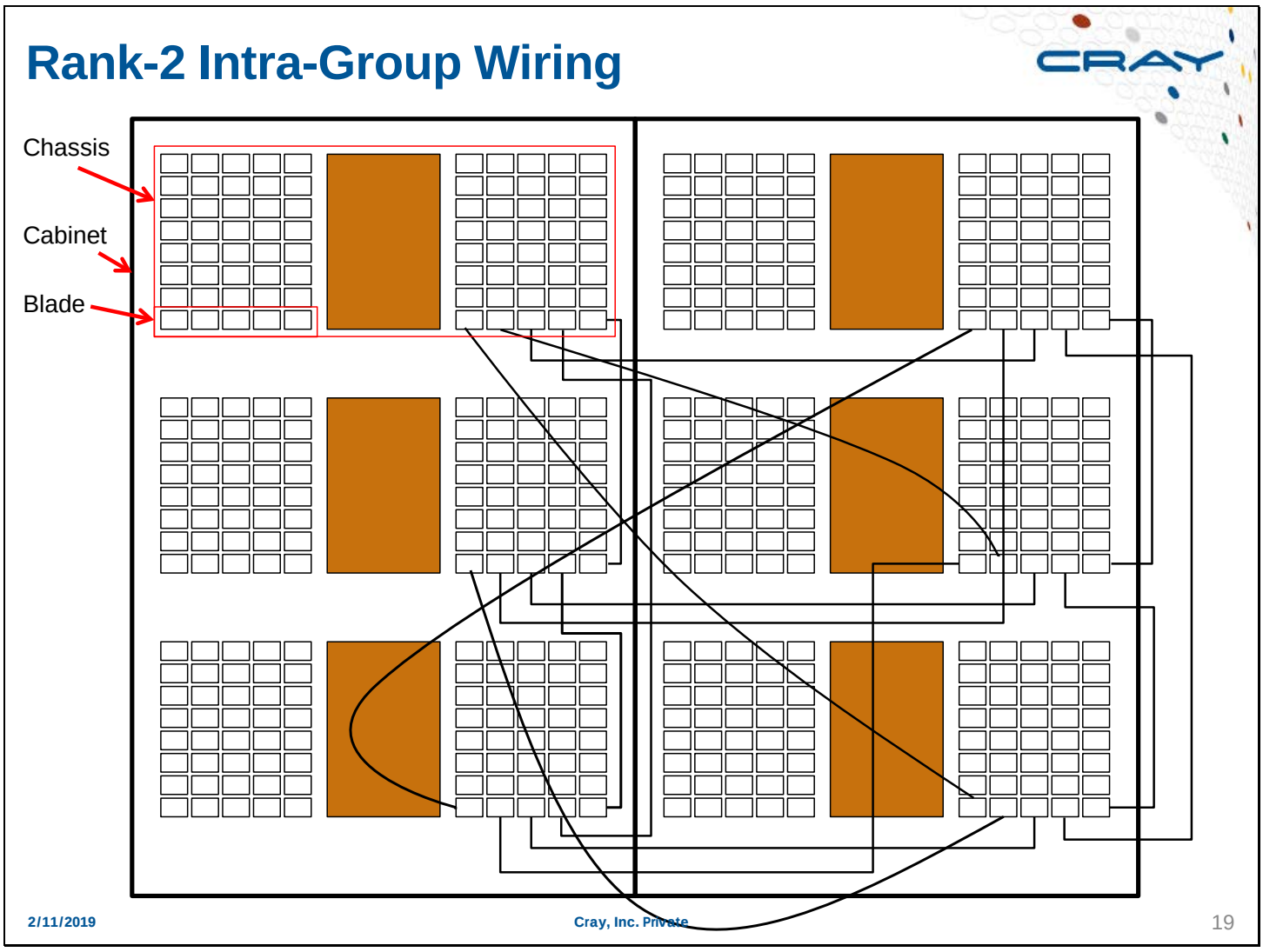
Intra-group
- **Copper cables**
- **15 links in 5 connectors**
- **Rank 2 (black) Network**
- **14 Gbps**

Inter-group links
- **Optical**
- **10 links in 5 connectors**
- **Rank 3 (blue) Network**
- **12.5 Gbps**





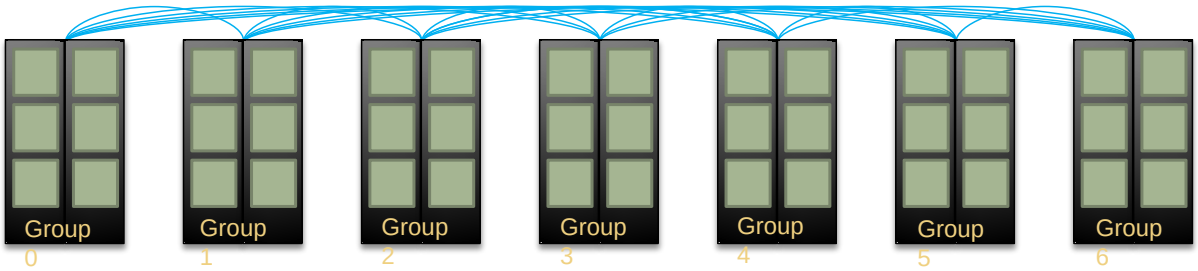
Rank 1 = 15 links
Rank 2 = 15 links (3 links x 5 connectors)
Rank 3 =10 links (2 links x 5 connectors)
Total = 40 links (the number of "N" tiles in the Aries chip)



This slide shows the electrical cable connections between slot 0 in all 6 chassis for a two-cabinet group. This cabling pattern is repeated for the other 15 slots in each chassis.

Rank-3 Topology

- **An all-to-all pattern is cabled between the groups using optical cables**
  - There are 240 AOC ports per 2-cabinet group
    - There are 40 AOC connecters per backplane, times 6 backplanes per two cabinet group
  - On a system the global bandwidth is selected at the time of the system order
    - Bandwidth is configured between 25 and 100%
    - Bandwidth is adjusted by varying the number of cables in the bundles that connect the groups together



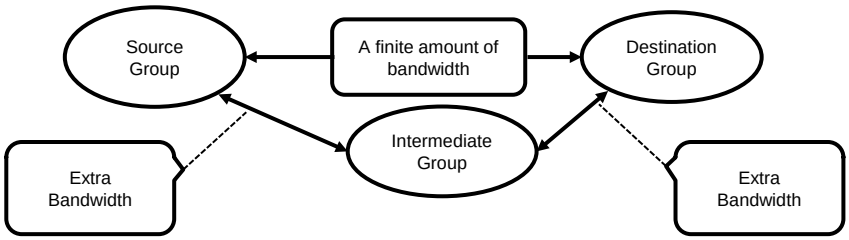
Example: *A 7-group system is interconnected with 21 optical "bundles". Each "bundle" can contain as few as 2 and up to 40 optical cables.*

2/11/2019 Cray, Inc. Private 20

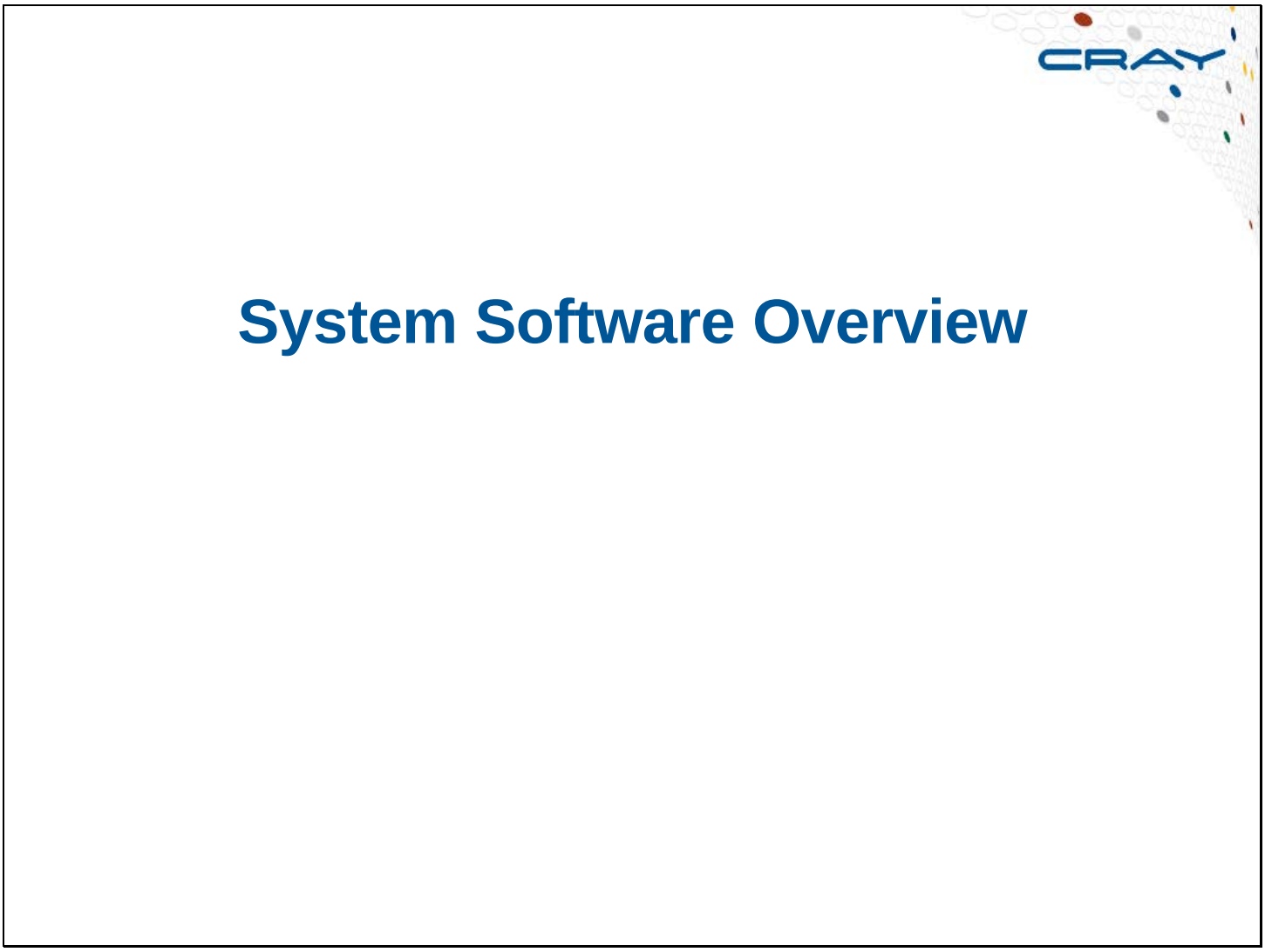
The number of bundles is (n x (n – 1))/2, where n is the number of cabinets in the system.

Routing Modes

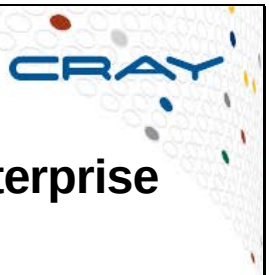
- Minimal
  - Minimal routing is the shortest distance between two nodes, but there is a finite amount of bandwidth.
- Non-minimal
  - Non-minimal uses a longer route that may include more hops
    - Non-minimal routing uses extra bandwidth from the intermediate groups.
- Adaptive
  - Routing begins by identifying two minimal and two non-minimal ports
    - Uses local or global, minimal and non-minimal tables to select ports
      - Uses congestion info and bias table to select whether to use the minimal or non-minimal tables.
      - May start non-minimal and switch to minimal




2/11/2019
Cray, Inc. Private
22

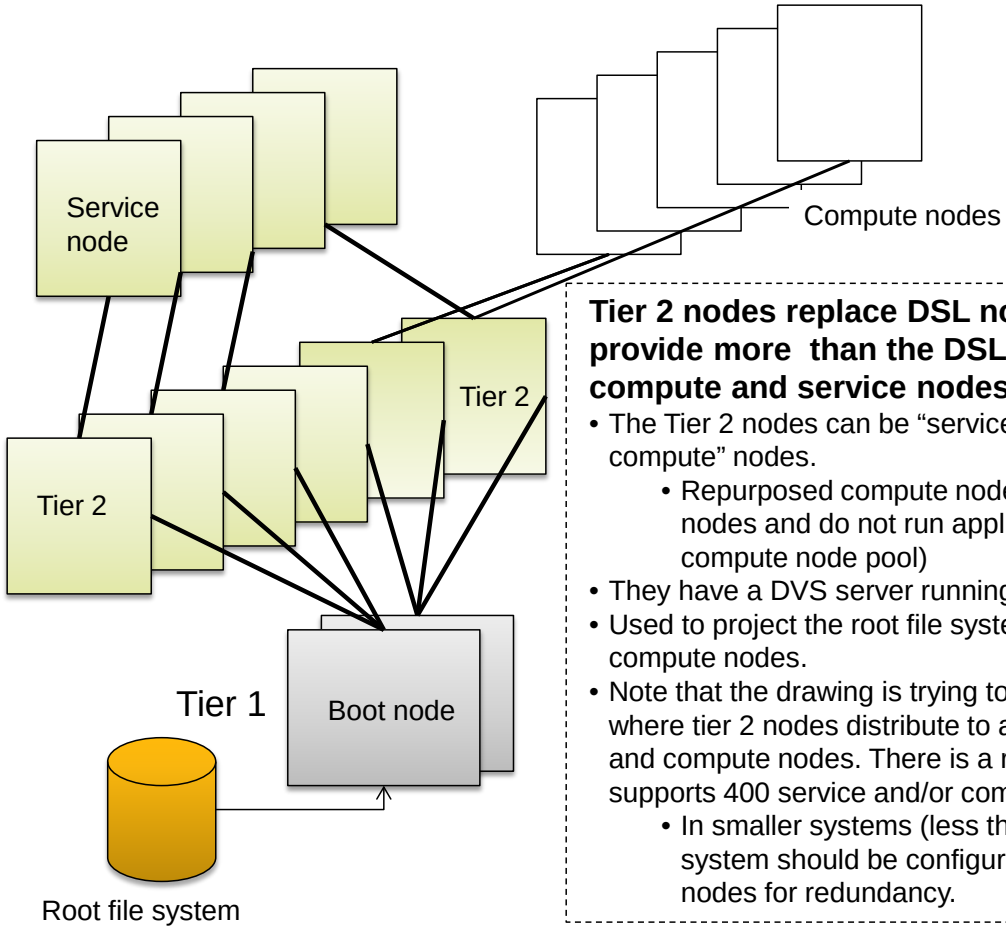

System Software Overview

System Software



- **The system software is based on SuSE Linux Enterprise Server (SLES) version 12**
  - Updates are incorporated as they become available
- **Cray refers to the Software stack as CLE 6.0**
  - CLE is *Cray Linux Environment*
  - CLE 6. is Cray's use of SLES 12
  - CLE 6.0, the .0 is the minor release
  - CLE 6.0.UP0*x*, the UP0*x* is Update Package *x*
  - The System Management Workstation (SMW) software is also tracked by a similar numbering scheme, the current version is SMW 8.0.UP0*x*
- **This provides a common Linux based user interface**

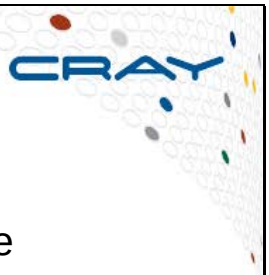
Tiered configuration of nodes



Tier 2 nodes replace DSL nodes and actually provide more than the DSL nodes to the compute and service nodes

- The Tier 2 nodes can be “service” or “repurposed compute” nodes.
  - Repurposed compute nodes are seen as “service” nodes and do not run applications (not part of the compute node pool)
- They have a DVS server running.
- Used to project the root file system (libraries) to the compute nodes.
- Note that the drawing is trying to represent a distribution where tier 2 nodes distribute to a collection of service and compute nodes. There is a ratio where 1 teir-2 node supports 400 service and/or compute nodes.
  - In smaller systems (less than 400 nodes) the system should be configured to have two tier-2 nodes for redundancy.

MAMU Nodes



Designed to allow multi-user jobs to share a node

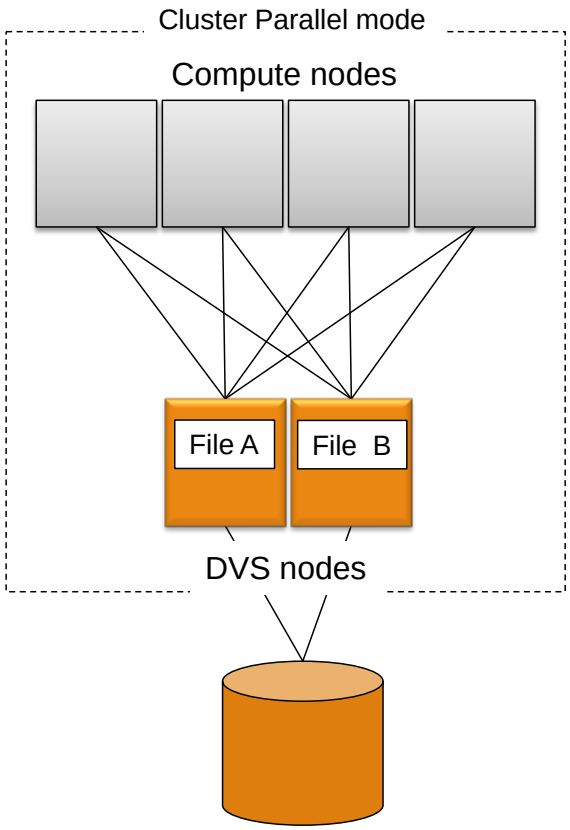
- More efficient for applications running on less than one node
- Possible interference from other users on the node

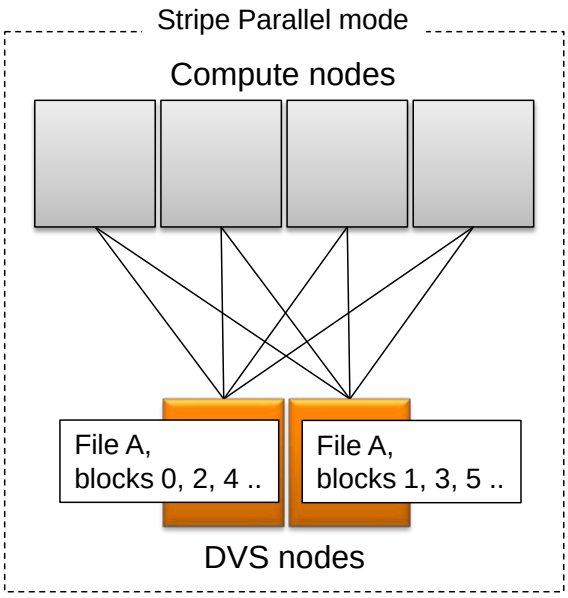
- **Uses the same fully featured OS as service nodes**
- **Multiple use cases, applications can be:**
  - Entirely serial
  - Embarrassingly parallel e.g. fork/exec, spawn + barrier.
  - Shared memory using OpenMP or other threading model.
  - MPI (limited to intra-node MPI only*)
- **Can be referred to as Pre or Post processing nodes**

DVS

- **Data Virtualization Services (DVS)**
  - Projects files system mounted on service nodes to the compute nodes
  - Operating modes
    - Cluster parallel mode
      - A single file is written to a single server
    - Stripe parallel mode
      - A single file is written across multiple DVS servers
    - Load balance mode
      - Work is distributed across the DVS servers
      - Supports failover

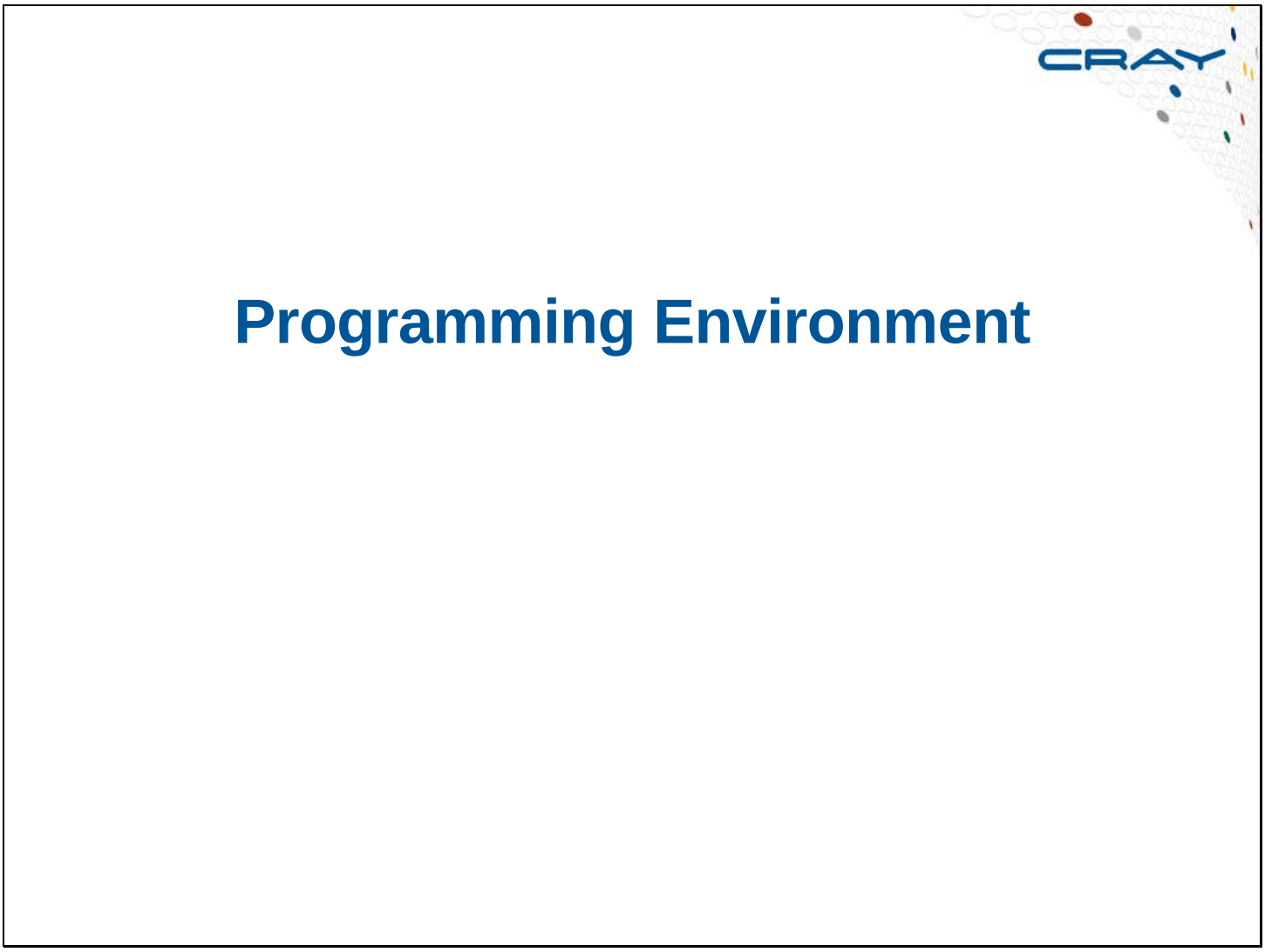
DVS





Depending on the block size and the number of servers, the data is written across all or some of the DVS servers

- All participating compute nodes (application) write the data block to the same server to minimize cache thrashing


CRAY
Programming Environment

Cray Programming Environment

- **A cross-compiler environment**
  - Compiler runs on an internal login node or a Cray Development and Login (CDL) node (external to the Cray system)
  - Executable runs on the compute nodes
- **Cray written compiler driver scripts**
  - CNL compiler options
  - CNL system libraries and header files
  - Compiler specific programming environment libraries
- **Modules utility**
  - Consists of the `module` command and *module files*
  - Initializes the environment for a specific compiler
  - Allows easy swapping of compilers and compiler versions

2/12/2019 (NERSC)

Module files, usually referred to as modules, are written in the Tool Command Language (tcl) . Module files contain commands to configure the shell environment for a particular compiler, library, or utility.

ssh is normally used to connect to the system.

User account information is maintained through an LDAP or Kerberos server.

You can set up passwordless ssh to access a system. You can also set up a pass phrase for a more secure session.

Compiler Driver Scripts

- **Do not call compilers directly; use Cray compile drivers**
  - `ftn`
  - `cc`
  - `CC`
- **Driver actions:**
  - Select compiler version
  - Add system libraries and header files
  - Add compiler-specific programming environment libraries
  - Execute the actual compiler command with added options
    - Without DSL configured, executables are statically linked.
- **Use vendor man pages for details of compiler options**
  - Cray man pages: `crayftn`, `craycc`, `crayc++`
  - GCC man pages: `gfortran`, `gcc`, `g++`
  - Intel man pages: `icc`, `icpc`, `fpp`, and `ifort`

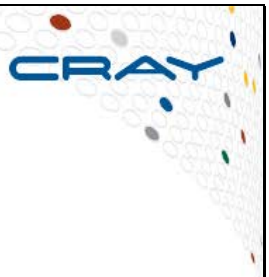
2/12/2019 (NERSC)

If the vendor compiler commands are called and the relevant module is not loaded, a login node binary is produced.

To find version information for the various compilers use -V with the Cray, and Intel compilers and --version with the GCC compilers.

To see all of the information associated with compiling an application using a Fortran compiler you can use the option –show (when using the show option no binary is produced)
For example:
users/rns> ftn –show samp261.F
The above command outputs a long list of information, too long to show here.

Available Compilers



- **GNU Compiler Collection (GCC)**
  - Always included with the system
- **Cray compilers (Cray Compiling Environment (CCE))**
  - Provides additional support for Fortran 2003, CAF (Fortran 2008), and UPC
- **Intel Compilers**
- **All provide Fortran, C, C++, and OpenMP support**
- **So Which Compiler Do I Choose?**
  - If your site offers you a choice, experiment with the various compilers
    - Mixing binaries created by different compilers may cause issues

2/12/2019 (NERSC)

SSE – Streaming SIMD Extensions
AVX – Advanced Vector Extensions
SIMD – Single Instruction, Multiple Data

Module Commands

- **Cray uses modules to control the user environment; use the commands:**

`module`	
`list`	To list the modules in your environment
`avail`	To list available modules For example: To see all available modules: `% module avail` To see all available *PrgEnv* modules: `% module avail PrgEnv`
`load/unload`	To load or unload module
`show`	To see what a module loads
`whatis`	Display the module file information
`swap/switch`	To swap two modules For example: to swap the Intel and Cray compilers `% module swap PrgEnv-intel PrgEnv-cray`
`help`	General help: `$module help` Information about a module: `$ module help PrgEnv-cray`

2/12/2019 (NERSC)

Cray Modules

PrgEnv-cray, PrgEnv-gnu, PrgEnv-intel	CCE, GCC, and Intel compilers All provide: C, C++, Fortran
cray-mpich	MPICH2
cray-shmem	SHMEM
cray-libsci (Cray scientific libraries)	BLAS, LAPACK, BLACS, FFT, FFTW, CRAFFT, IRT, ScaLAPACK, and SuperLU_DIST
Debuggers	cray-lgdb, TotalView, and ddt and ddt-memdebug (DDT (Allinea - Distributed Debugging Tool)
Performance tools	perftools (Includes: CrayPat, Apprentice2, and Reveal), perftools-lite, (perftools-lite-events, perftools-lite-gpu, perftools-lite-hbm, perftools-lite-loops, and perftools-nwpc) perftools-base, and papi (PAPI library)
Other Libraries	cray-trilinos, cray-petsc, cray-petsc-complex-64, cray-netcdf, cray-netcdf-hdf5parallel, cray-parallel-netcdf

2/12/2019 (NERSC)

BLAS (Basic Linear Algebra Subprograms) are used for vector-vector, matrix-vector and matrix-matrix operations and are tuned for a particular architecture. For more information refer to the man pages: intro_blas1, intro_blas2, and intro_blas3

LAPACK (Linear Algebra PACKage) solves systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigen value problems, and singular value problems.

The BLAS and LaPACK libraries include libGoto form the University of Texas. C programmers must use the Fortran interface to these libraries.

FFT (Fast Fourier Transforms) is package of Fortran subprograms for the fast Fourier transform of periodic and other symmetric sequences. For more information refer to the man pages: intro_fft, intro_fftw2 and intro_fftw3

ScaLAPACK (Scalable Linear Algebra PACKage) contains High-performance linear algebra routines for distributed-memory message-passing MIMD computers and networks of workstations that support PVM and/or MPI.

BLACS (Basic Linear Algebra Communication Subprograms) is a message-passing library, designed for linear algebra. The computational model consists of a one- or two-dimensional process grid, where each process stores pieces of the vectors and matrices.

SuperLU is a general purpose library for the direct solution of large, sparse, nonsymmetric systems of linear equations on high-performance machines. Functions are written in C and callable from either C or Fortran. Three different versions exist for various machine architectures; Cray XT systems are distributed memory systems

UPC - Unified Parallel C

Processor and Network Modules

- Cray XC systems
  - `module load craype-x86-skylake`
  - `module load craype-broadwell`
  - `module load craype-haswell`
  - `module load craype-ivybridge`
  - `module load craype-sandybridge`
- Network modules:
  - `module load craype-network-aries`
  - `module load craype-network-none`
- Accelerators
  - `module load craype-mic-knl`
  - `module load craype-intel-knc`
  - `module load craype-accel-nvidia 60 (Pascal GPU)`
  - `module load craype-accel-nvidia 52 (Maxwell GPU)`
  - `module load craype-accel-nvidia35 (Kepler GPU)`
  - `module load craype-accel-nvidia20 (Fermi GPU)`

2/12/2019 (NERSC)

module Example

```
users/rns% module swap PrgEnv-intel PrgEnv-cray
Currently Loaded Modulefiles:
  1) modules/3.2.10.4
 …
 14) Base-opts/2.4.135-6.0.7.0_38.1__g718f891.ari
 15) cce/8.7.8
 16) craype-network-aries
 17) craype/2.5.17
 18) cray-libsci/19.01.1
 19) pmi/5.0.14
 20) rca/2.2.18-6.0.7.0_33.3__g2aa4f39.ari
 21) atp/2.1.3
 22) PrgEnv-cray/6.0.4
 23) totalview-support/1.2.0.81
 24) totalview/2018.3.8
 25) cray-mpich/7.7.4
 26) craype-broadwell
 27) perftools-base/7.0.5
users/rns%
rns/samp264% ftn -o samp264 samp264.f
rns/samp264% file samp264
samp264: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux),
statically linked, for GNU/Linux 3.0.0, not stripped
rns/samp264%
```

Compiler and compiler version

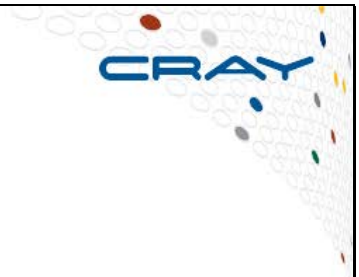
Interconnect network type

Compile driver and version

2/12/2019 (NERSC)

Your site may load the relevant modules during the login shell start-up; issue the command module list to determine what is loaded.

Compiling Fortran



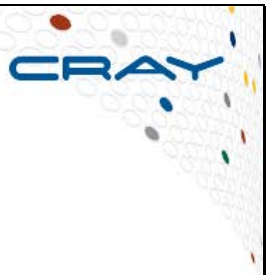
- Compiling and loading (Fortran 90):
  `% ftn –o myprog myprog.f90`
- Four stages:
  - Preprocess: `mpi_where.f90 > mpi_where.f`
  - Compile: `mpi_where.f > mpi_where.s`
  - Assemble: `mpi_where.s > mpi_where.o`
  - Link: `mpi_where.o > mpi_where`
- `% ftn –F mpi_where.f90` produces `myprog.f`
- `% ftn –S mpi_where.f90` produces `myprog.s`
- `% ftn –c mpi_where.f90` produces `myprog.o`

2/12/2019 (NERSC)

Fortran Suffixes

.f	Fixed-format Fortran source; compile
.F	Fixed-format Fortran source; preprocess and compile
.f90	Free-format Fortran source; compile
.f95	Free-format Fortran source; compile
.F90	Free-format Fortran source; preprocess and compile
.F95	Free-format Fortran source; preprocess and compile
.for	Fixed-format Fortran source; compile
.fpp	Fixed-format Fortran source; preprocess and compile

Compiling C



- **Compiling and loading C**
  - `% cc –o mpi_where mpi_where.c`
- **Three stages:**
  - Preprocess and compile: `mpi_where.c > mpi_where.s`
  - Assemble: `mpi_where.s > mpi_where.o`
  - Link: `mpi_where.o > mpi_where`
  - `% cc –S mpi_where.c` produces `mpi_where.s`
  - `% cc –c mpi_where.c` produces `mpi_where.o`

2/12/2019 (NERSC)

OpenMP

- **OpenMP is a shared-memory parallel programming model that application developers can use to create and distribute work using threads**
  - OpenMP provides library routines, Fortran directives, C and C++ pragmas, and environment variables
  - OpenMP applications can be used in hybrid OpenMP/MPI applications, but may not cross node boundaries
    - In OpenMP/MPI applications, MPI calls can be made from master or sequential regions, but not parallel regions

2/12/2019 (NERSC)

OpenMP

- With Cray compilers, it is on by default (`-h omp`)
  - You can disable openmp with `-h noomp`; you can also use the flag `-O omp` or `-O noomp`
- With GCC compilers, use the `-fopenmp` option
- With the Intel compilers, use the `-openmp` option
- To execute OpenMP programs:
  - Set the `OMP_NUM_THREADS` environment variable with an appropriate value
    - When using ALPS (Cray) use the `aprun -d <threads>` option
    - When using `srun` use the `-c, --cpus-per-task <threads>` option
  - Note: The number of threads should not exceed the number of cores (CPU threads) in the node.
  - With Intel compilers, prior to compiling:
    `export KMP_AFFINITY=disabled`
    - Check the `man` page for additional options

2/12/2019 (NERSC)

UPC and Fortran with Coarray Support

- **These are known as PGAS languages (Partitioned Global Address Space), and work on domains similar to MPI**
  - CCE compilers only (Intel CAF has limited capabilities)
    - Fortran 2008 standard, with extensions for additional collectives
    - Cray supports the draft version of UPC version 1.3
    - PGAS support is native to the compiler
      - For UPC, there is no need to translate to C first (preprocess)
    - The CCE implementation uses advanced capabilities of the interconnect, including optimizations that:
      - Automatically convert loops of many scalar remote memory operations to single large transfers
      - Automatic overlap of communication and computation

2/12/2019 (NERSC)

Intel Fortran does support coarrays, but their implementation does not leverage the Gemini PGAS support. Intel C does not provide UPC support.

GNU compilers do not provide UPC or Fortran with coarray support, although there is a third-party effort to add UPC support to GCC (typically based on GASNET).

For more information see the man page **intro_pgas**

Executing Programs on CNL

- **All application executables on the Cray systems are malleable (adaptable)**
  - The number of processors to run on is determined at runtime
  - Must be in a directory accessible by the compute nodes
    - Normally this is a Lustre file system
    - Can be a DVS mounted file system
      - Performance may be an issue
      - This is not a DVS issue, the exact system configuration determines performance

2/12/2019 (NERSC)

The lustre file system mount point can be found by executing the following command:

```
% df -t lustre
Filesystem           1K-blocks      Used Available Use% Mounted on
43@ptl:/work/user  1664914272  60085752 127919700   2% /scratch
```

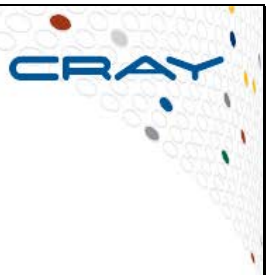
Huge Pages

- **Using Huge pages can improve the performance of your application**
  - Huge pages are:
    - Default or “base” page size is 4 KB
    - On Cray XC system huge pages of 2MB, 4MB, 8MB, 16MB, 32MB, 64MB, 128MB, 256MB, and 512MB are available
  - To use 2MB huge pages load the module and compile your application
    - `$ module load craype-hugepages2M`
    - `$ cc -o my_app my_app.c`
    - When you link (compile) your application with huge pages, run your application with the same module loaded
  - The memory available for huge pages is less than the total amount of memory available to the PE
    - The operating system and I/O buffers reduce available memory
    - Memory fragmentation can reduce available memory
      - Fragmentation usually increases with time
      - This could affect running multiple runs of the application

2/12/2019 (NERSC)

See the intro_hugepages man page for more information.

Memory Allocation: Make it local



- **Touch your memory, or someone else will**
  - Linux has a “first touch policy” for memory allocation
    - *alloc functions don’t actually allocate your memory, it gets allocated when “touched”
  - A code can over allocate the memory
    - Linux assumes “swap space,” we do not have any
    - The applications will not fail until the memory is finally touched
  - Always initialize (touch) your memory immediately after allocating it
    - If you over-allocate, it will fail immediately, rather than at a strange place in your code
    - If every thread touches its own memory, it will be allocated on the proper socket.

2/12/2019 (NERSC)

Using cnselect to Select Nodes

- **If you have a mixture of node types `cnselect` is a convenient MySQL interface to the SDB attributes table**
  - Returns a list of compute nodes based on user-specified criteria
  - Must be run from a login node
  - Nodes used will be selected from the list but only as many as needed will be reserved

```
% module load MySQL
% NODES=$(cnselect numcores.eq.48 .and. availmem.eq.32768)
% echo $NODES
44-55
% export OMP_NUM_THREADS=1
% aprun -n 2 -d 1 -L $NODES ./OMP_where
Rank =         0  Thread =         0 Processor = nid00044
Rank =         1  Thread =         0 Processor = nid00045
```

2/12/2019 (NERSC)

The first example selects any available node. The second two select single- or dual-core nodes. The next one select s clock speed. The remaining ones select on memory size; the final one also chooses the number of cores.

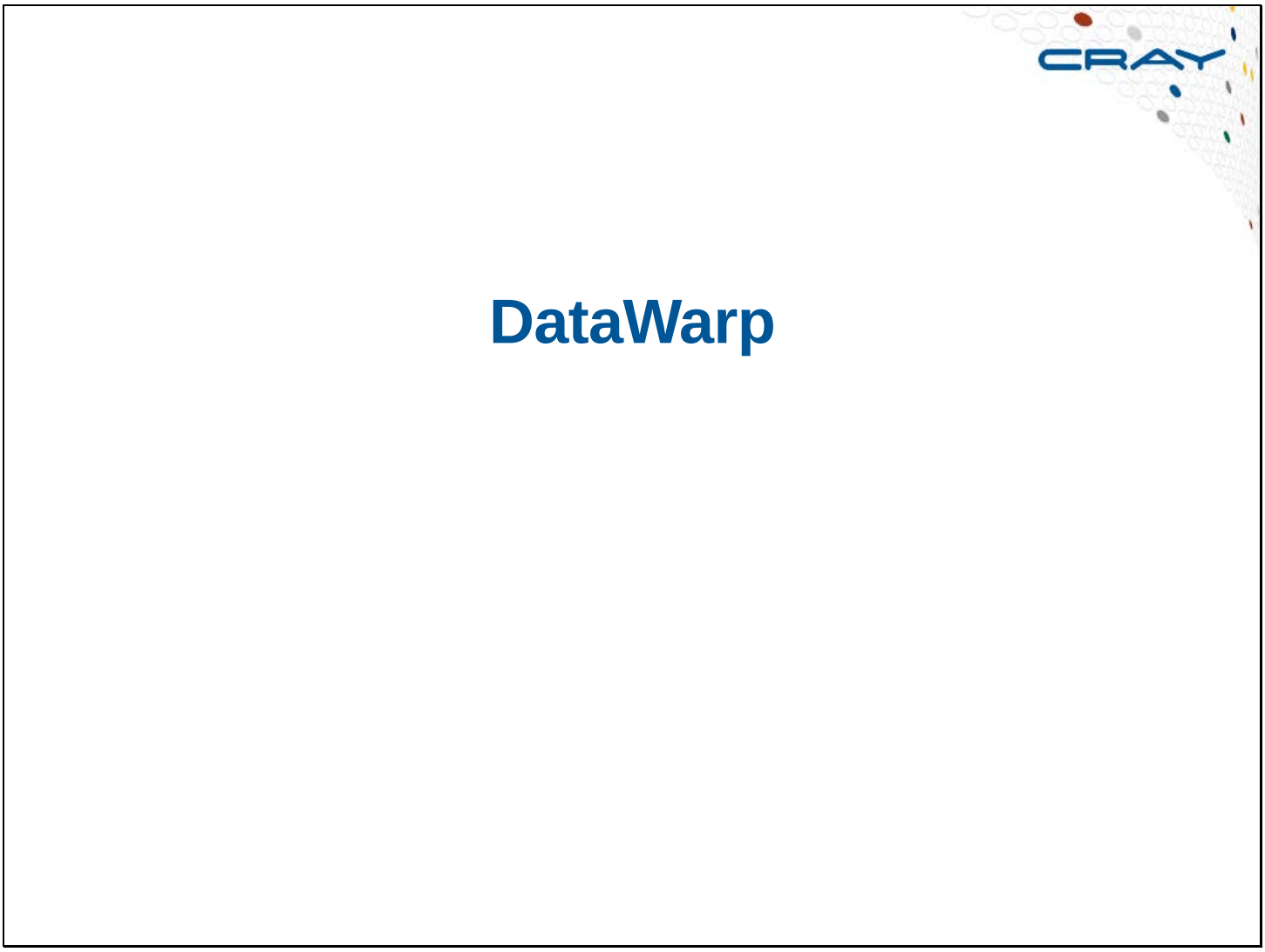
```
% module load MySQL
% cnselect
44-63
% cnselect coremask.eq.1
44-55
% cnselect coremask.gt.1
56-63
% cnselect clockmhz.ge.2400
56-63
% cnselect availmem.lt.2000
44-55
% cnselect availmem.eq.2000
60-63
% cnselect availmem.gt.2000
56-59
% cnselect availmem.lt.3000
```

44-55,60-63
% cnselect availmem.lt.3000.and.coremask.eq.1
44-55

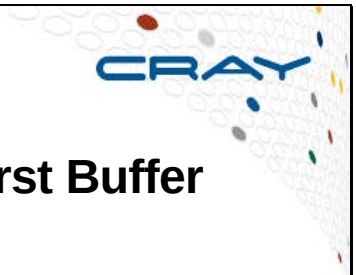
Launching a Compute Node Program

- **CLE supports a number of Workload Managers (WLM)**
  - Each WLM has it's own characteristics
  - Supported WLMs include:
    - PBS Professional
    - TORQUE/Moab
    - Slurm (ALPS is removed from the system)
      - ALPS is the Application Level Placement Scheduler
    - LSF (Load Sharing Facility)
  - WLMs will support Interactive and Batch job launch
    - Interactive: nodes are allocated to the user and the user launches applications against those nodes.
      - Nodes are reserved for a period of time for the user to use
    - Batch
      - User submits job script, job is queued, job runs at a time determined by the WLM
        - Script runs launcher command to start the compute node application

2/12/2019 (NERSC)


DataWarp

What is DataWarp



- **DataWarp is Cray's implementation of the Burst Buffer concept, plus more**
  - Has both Hardware & Software components
  - Hardware
    - XC Service node, directly connected to Aries network
    - PCIe SSD Cards installed on the node
  - Software
    - DataWarp service daemons
    - DataWarp Filesystem (using DVS, LVM, XFS)
    - Integration with WorkLoad Managers (Slurm, MOAB/Torque, PBSpro)

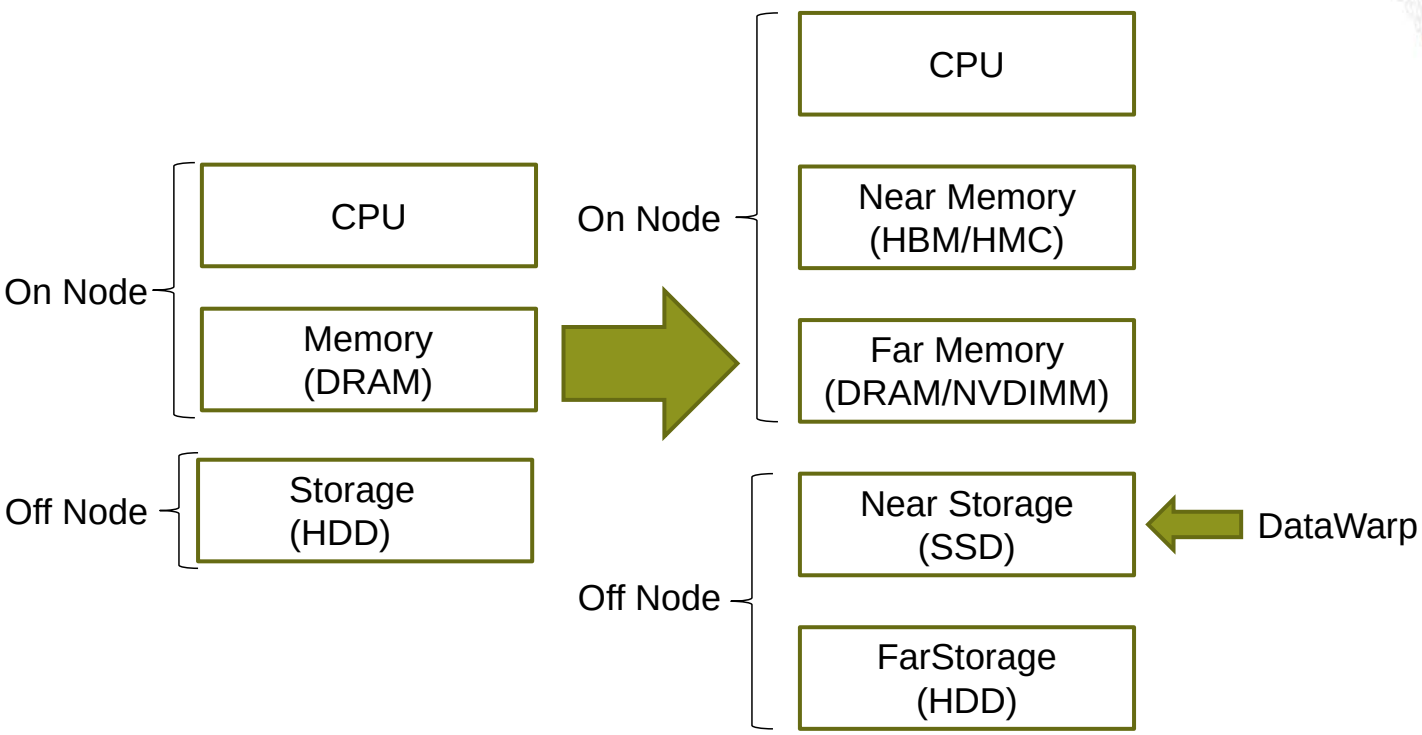
Observations and Trends

- **Many programs do I/O in bursts**
  - Read, Compute, Write, Compute, Write, Compute, etc.
- **Want to have high bandwidth when doing I/O**
  - Compute resources largely idle during I/O
- **Disk-based Parallel Filesystems (PFS) bandwidth is expensive, capacity is cheap**
  - PFS do provide lots of capacity, reliability, permanence, etc.
- **SSD bandwidth is (relatively) inexpensive**
- **Large I/O load at beginning and end of job**
- **Cray Aries network is faster, lower latency, shorter distance than path to the Parallel Filesystem**

Burst Buffer Concepts

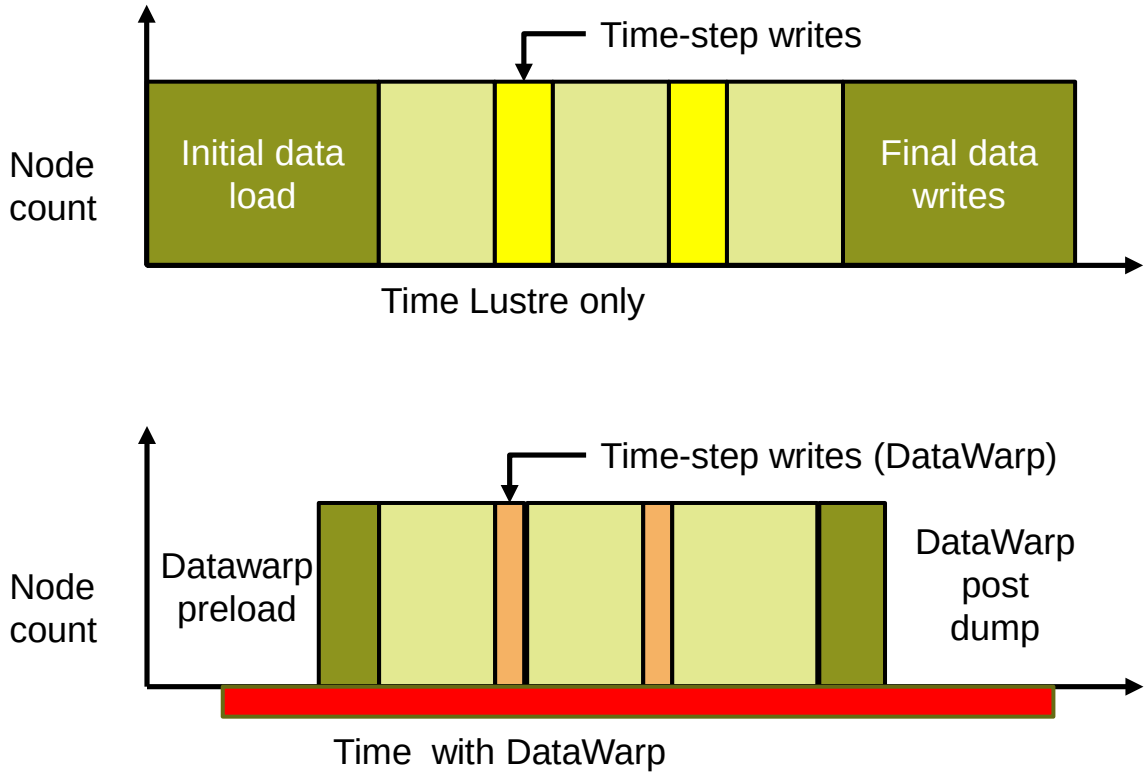
- **Burst Buffer - A high bandwidth, lower capacity, "buffer" space, backed by a disk based Parallel File System**
  - Higher Burst Buffer bandwidth decreases time programs spend on I/O
- **Burst Buffer can interact with the Parallel Filesystem before, during, and after program use**
  - Stage data in to Burst Buffer before computes allocated
  - Stage data back out to Parallel Filesystem after computes deallocated
  - Stage data in or out, using Burst Buffer hardware, while program in computational phase
- **Burst Buffers offer much greater bandwidth per dollar (5x)**
  - So, do I/O to Burst Buffer and write out to Parallel Filesystem over time

Exascale Computing Memory Trends

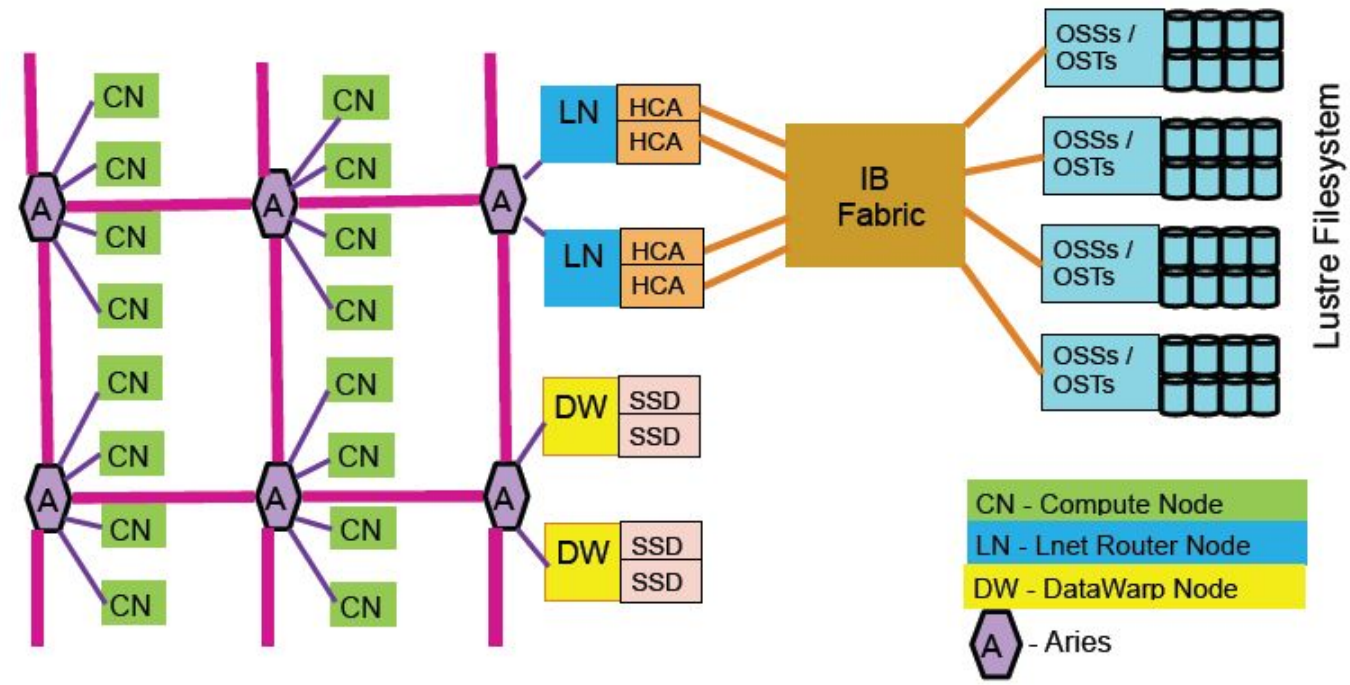


2/14/2017 (NERSC) Cray, Inc. Private 51

DataWarp - Minimize Compute Residence Time

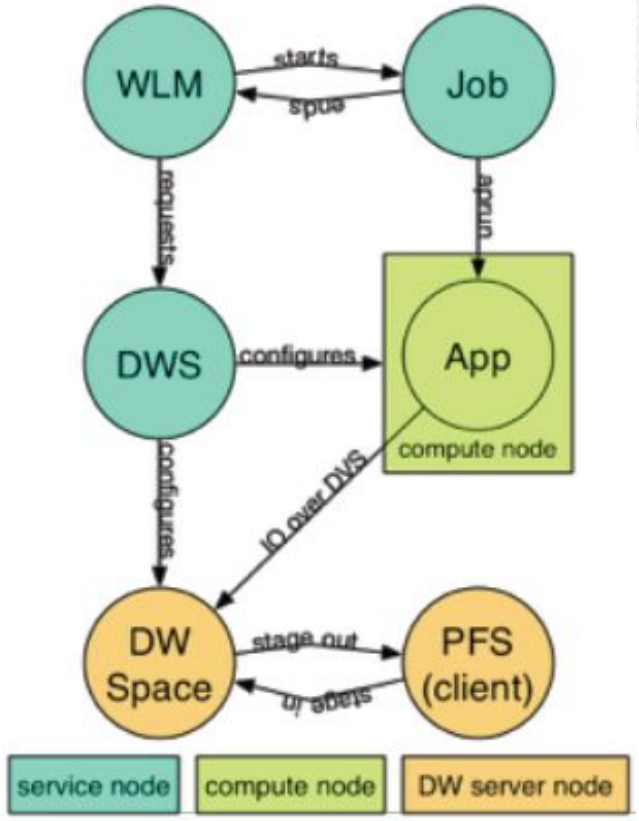


DataWarp Hardware Architecture



DataWarp Job Flow

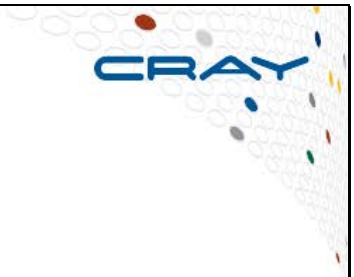
- WLM queues job, requests DWS set up job for using DW
- DataWarp Service (DWS) configures DW space, compute node access to DW
- DataWarp Filesystem handles stage interactions with PFS
- Compute nodes access DW via a mount point



Two kinds of DataWarp Instances (Space)

- **Two kinds of DataWarp Instances (Space)**
  - Can it be shared? What is its lifetime?
    - Job Instance
      - Can only be used by job that creates it
      - Lifetime is the same as the creating job
    - Persistent Instance
      - Can be used by any job (subject to permissions)
      - Lifetime is controlled by creator
- **Two DataWarp Access Modes**
    - Striped (“Shared”)
      - Files are striped across all DataWarp nodes assigned to an instance
      - Files are visible to all compute nodes using the instance
    - Private
      - Files are assigned to one DataWarp node
      - File are visible to only the compute node that created them

How to Utilize DataWarp



- **Job script directives - `#DW` ...**
  - Allocate job DataWarp space
  - Access persistent DataWarp space
  - Stage files or directories in from PFS to DW; out from DW to PFS
  - Supported by Slurm, Moab/TORQUE so far, PBSPro support soon
- **User library API – `libdatawarp`**
  - Allows direct control of staging files asynchronously
  - C library interface
- **Mount points**
  - Perform POSIX I/O through mount points on compute nodes
- **Command line**
  - "`dwstat`" command for users and admins to see status of their spaces
  - Other commands, like dwcli, mostly for system admins

Job Script commands: Without & With DataWarp

Without DataWarp

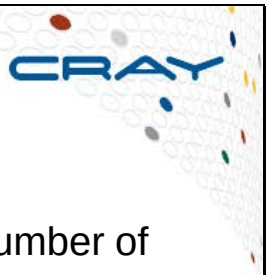
```
#!/bin/bash
#SBATCH -n 3200 -t 2000
export JOBDIR=/lustre/my_dir
srun –n 3200 a.out
```

With DataWarp

```
#!/bin/bash
#SBATCH -n 3200 -t 2000
#DW jobdw type=scratch access_mode=striped capacity=1TiB
#DW stage_in type=directory source=/lustre/my_dir destination=$DW_JOB_STRIPED
#DW stage_out type=directory destination=/lustre/my_dir source=$DW_JOB_STRIPED
export JOBDIR=$DW_JOB_STRIPED
srun –n 3200 a.out
```

More detailed examples on the NERSC user web pages at:
http://www.nersc.gov/users/computational-systems/cori/burst-buffer/example-batch-scripts/

Design differences - Lustre vs DataWarp



- **Lustre**
  - User can guarantee the number of servers (by setting the number of OSTs, stripe sizes)
  - ...but cannot guarantee amount of space
  - If an OST fills up, user is out of luck
- **DataWarp**
  - User can guarantee the amount of space
  - ...but cannot guarantee the number of servers
  - Same request may get assigned different numbers of servers on different runs, depending on DW activity and configuration on the system

Job Script commands: Without and With DataWarp

Without

```
#!/bin/bash
#SBATCH -n 3200 -t 2000
export JOBDIR=/lustre/my_dir
srun –n 3200 a.out
```

With

```
#!/bin/bash
#SBATCH -n 3200 -t 2000
#DW jobdw type=scratch access_mode=striped capacity=1TiB
#DW stage_in type=directory source=/lustre/my_dir destination=$DW_JOB_STRIPED
#DW stage_out type=directory destination=/lustre/my_dir source=$DW_JOB_STRIPED
export JOBDIR=$DW_JOB_STRIPED
srun –n 3200 a.out
```

More detailed examples on the NERSC user web pages at:
http://www.nersc.gov/users/computational-systems/cori/burst-buffer/example-batch-scripts/

Using dwstat - your view of DataWarp state

- **Command line client to view DataWarp state**
- **Most useful `dwstat` commands for users**
  - `dwstat pools`

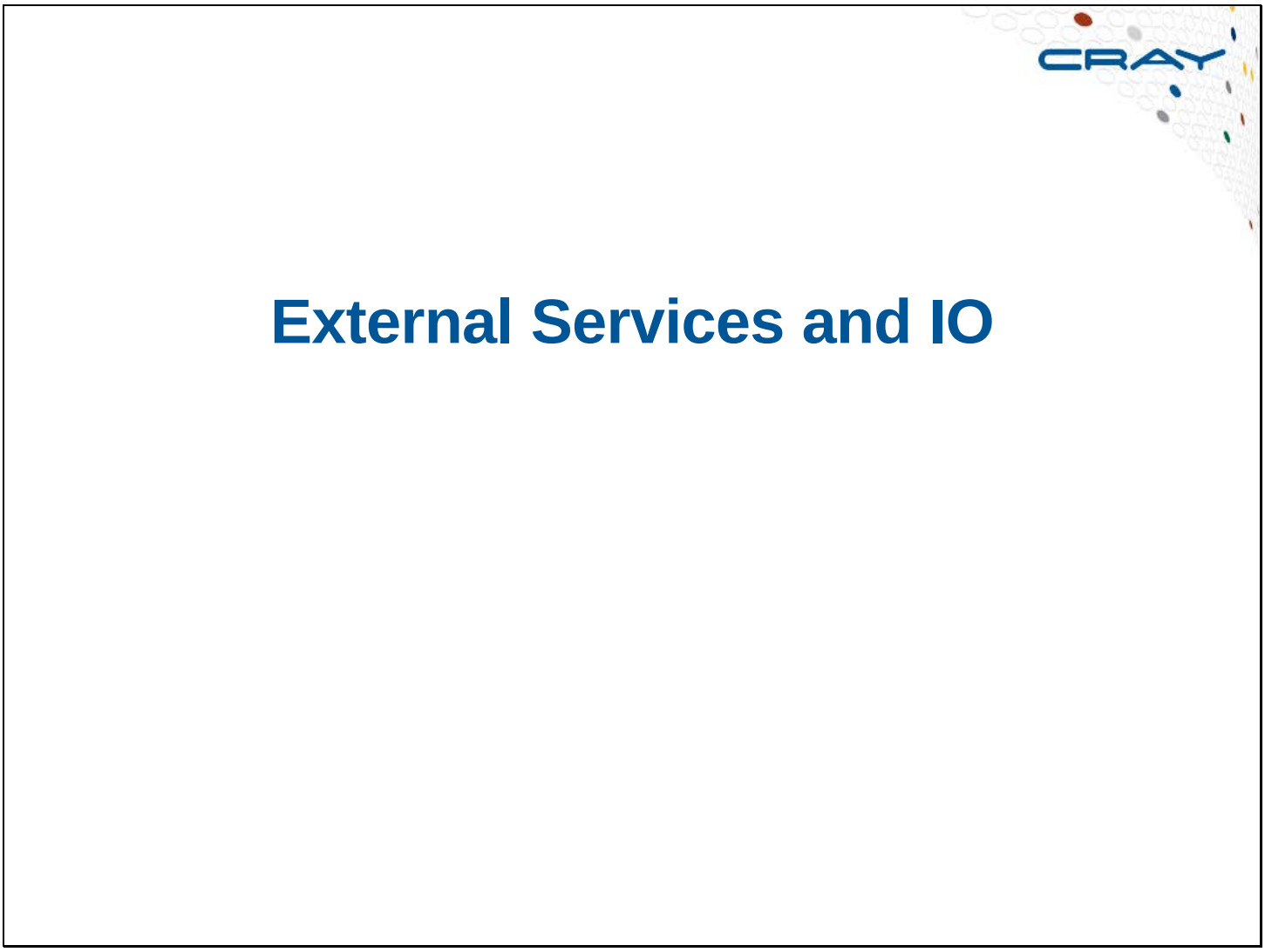
```
prompt:#> module load dws
prompt:#> dwstat pools
         pool     units    quantity free              gran
  test_pool       bytes    5.82TiB  5.82TiB  16MiB
   wlm_pool       bytes    832.5TiB          755.98TiB         212.91GiB
```

  - `dwstat instances`

```
prompt:#> dwstat instances
Inst state sess  bytes  nodes            created  expiration intact label public confs
  29 CA--- 36    16MiB  1     2015-08-21T13:10:05   never     true   blast true       1
  37 CA--- 44    16MiB  1     2015-08-26T08:51:28   never     true   I44-0 false      2
```

  - `dwstat fragments`

```
prompt:#> dwstat fragments
Frag     state    inst     capacity     node
  38     CA--     29       16MiB             nid00009
 108     CA--     37       16MiB             nid00010
```



External Services and IO

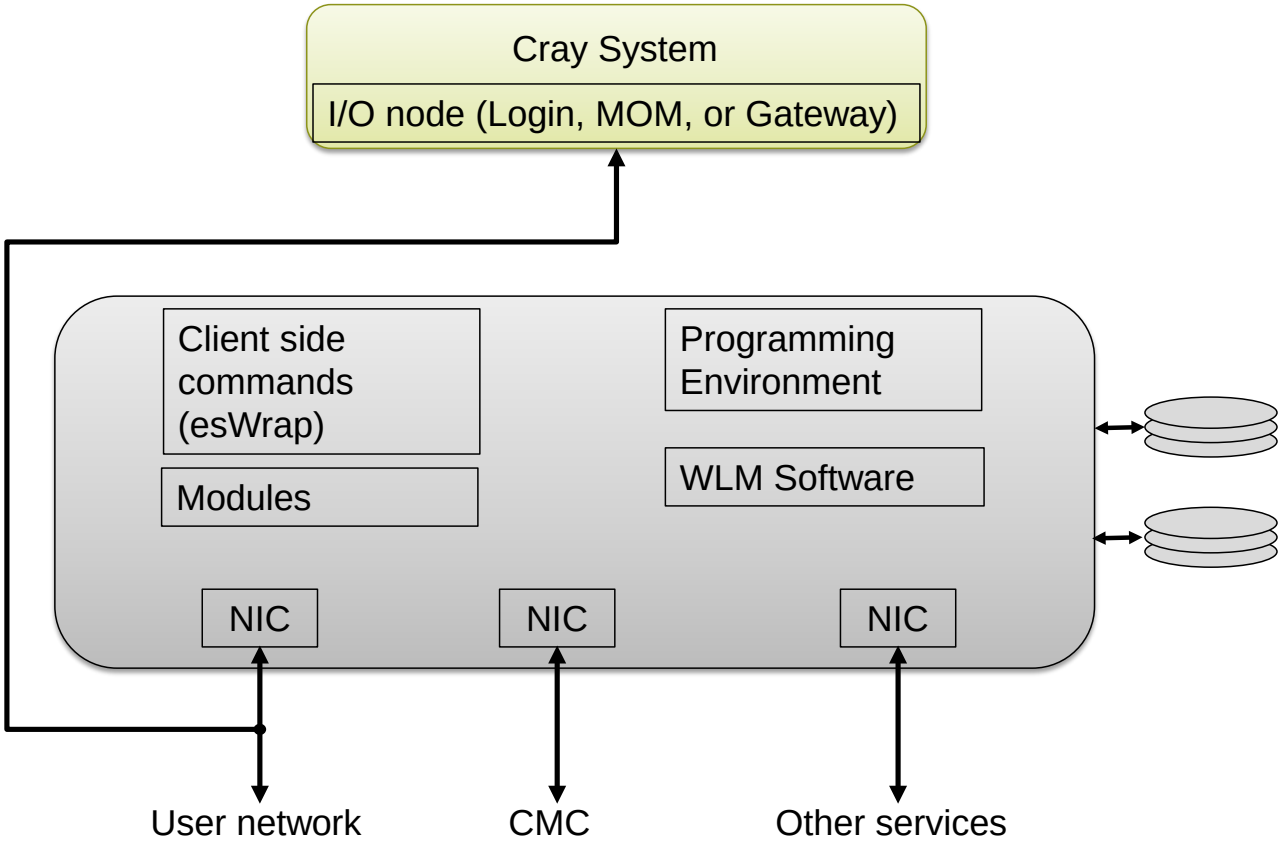
eLogin Nodes

- **External servers that are configured with the Cray Programming Environment (PE)**
- **Why eLogin nodes**
  - To address customer requirements:
    - More flexible user access
    - More options for data management, data protection
    - Leverage commodity components in customer-specific implementations
    - Provide faster access to new devices and technologies
    - Repeatable solutions that remain open to custom configuration
    - Enable each solution to be used, scaled, and configured independently
  - Provides the same environment as an internal Login or Gateway node
    - Compile and launch a user application
    - Monitor and control the application

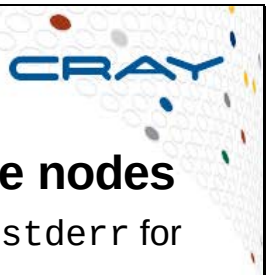
eLogin

- **eLogin**
  - Was called Cray Development and Login node (CDL) (earlier esLogin)
  - Increases availability of data and system services to users
    - An enhanced user environment
      - Larger memory, swap space, and more horsepower
    - Often still available to users when Cray system is unavailable
  - eLogin hardware configuration:
    - Multi socket, multi-core processors; match compute nodes
    - Internal memory of 128 GB or more
    - Local disks for local `root`, `swap`, and `tmp`
    - NICs for connection to Cray system and customer network
  - eLogin software configuration:
    - Connections to file systems, including CLFS
      - Workload Manager (WLM) for job submission
        - WLM could be PBS Professional, TORQUE/Moab, or Slurm
    - Cray libraries, build tools, performance tools
    - Third-party compiler(s) and debugger(s)

eLogin Node

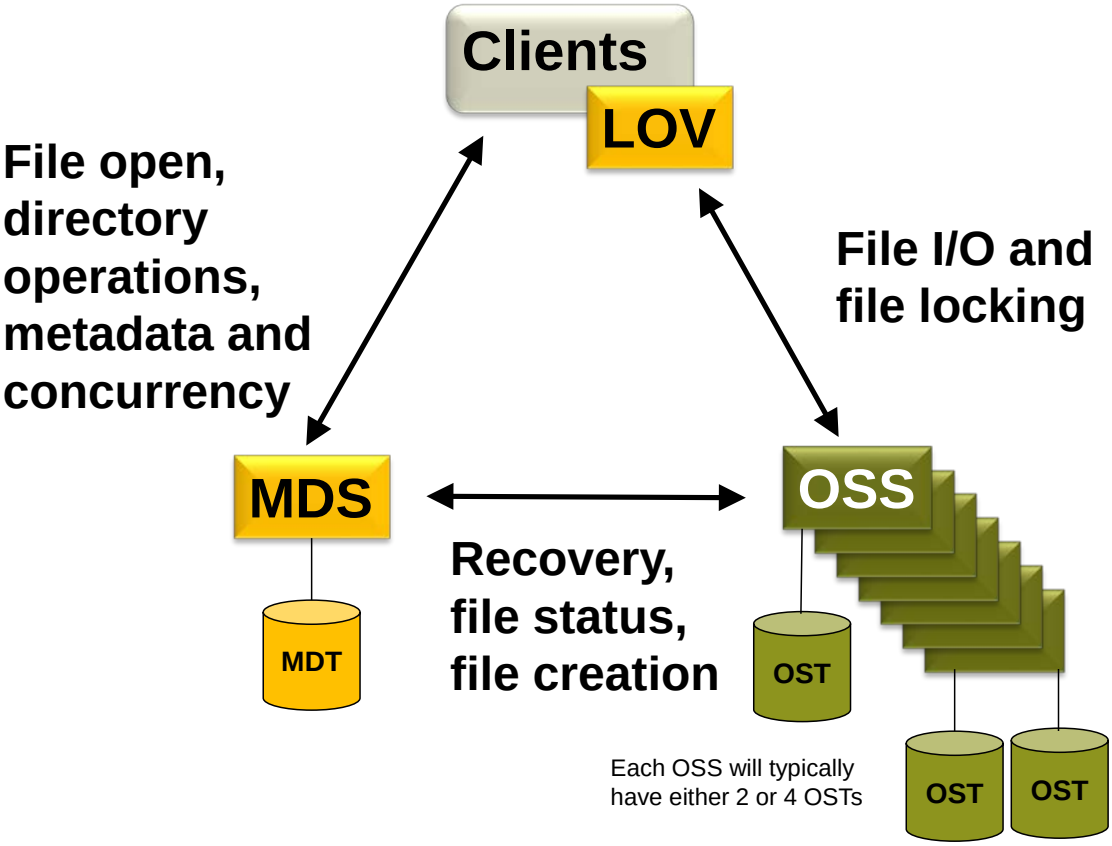

Cray System
I/O node (Login, MOM, or Gateway)
Client side commands (esWrap)
Modules
Programming Environment
WLM Software
NIC
NIC
NIC
User network
CMC
Other services

I/O Support



- **The compute nodes will hand off I/O to the service nodes**
  - The `aprun` application launcher handles `stdin`, `stdout`, and `stderr` for the application
- **Lustre provides a parallel file system for application use**
  - A Lustre file system consists of a Metadata server (MDS) and one or more Object Storage Targets (OSTs)
    - If you want to create another Lustre file system, you must configure it on separate service nodes and disk devices
- **Other file system may be available to the compute nodes, but may not provide the performance of Lustre**
  - Other file systems are provided for convenience

Node Interaction



Using the df Command

- **Use the standard `df` or `mount` command to locate the mount point for a Lustre file system**

```
users/rns> df -t lustre
Filesystem                                  1K-blocks        Used   Available Use% Mounted on
10.149.0.2@o2ib:10.149.0.3@o2ib:/scratch 93685609608 12869871220 76128850632  15% /lus/scratch


users/rns> mount -t lustre
10.149.0.2@o2ib:10.149.0.3@o2ib:/scratch on /lus/scratch type lustre (rw,flock)
users/rns>
```

Lustre Commands

- **`lfs` is a Lustre utility that can:**
  - Provide file system configuration information
    - `lfs df`
  - Create a file or directory with a specific striping
    - `lfs setstripe`
  - Display file striping patterns
    - `lfs getstripe [directory | file name]`
  - Find file locations
    - `lfs find [directory | file name]`
      - For example, to find directories or files on a particular OST
      - `lfs find –r –obd ost5_UUID /work/rns`
  - Display quota information
    - `lfs quota -u|g <name> file system`

The stripe utility, lfs, enables you to stripe files after Lustre is configured. Normally, you stripe files by using the lmc command when you create the configuration file.

Client View of File System Space

- **To view the individual OSTs, use: `lfs df`**

```
users/rns> lfs df
UUID                   1K-blocks        Used   Available Use% Mounted on
scratch-MDT0000_UUID   878145652    18574060   801018060   2% /lus/scratch[MDT:0]
scratch-OST0000_UUID 15614268268  2330265896 12502854904  16% /lus/scratch[OST:0]
scratch-OST0001_UUID 15614268268  2367066484 12466053784  16% /lus/scratch[OST:1]
scratch-OST0002_UUID 15614268268  2540145500 12292975156  17% /lus/scratch[OST:2]
scratch-OST0003_UUID 15614268268  1869094548 12964023432  13% /lus/scratch[OST:3]
scratch-OST0004_UUID 15614268268  1813158756 13019961664  12% /lus/scratch[OST:4]
scratch-OST0005_UUID 15614268268  1950155424 12882965000  13% /lus/scratch[OST:5]

filesystem summary:  93685609608 12869886608 76128833940  14% /lus/scratch

users/rns>
```

File Striping and Inheritance

- **Lustre distributes files across all OSTs**
  - The default stripe count is set in the configuration file
    - Striping is at the system (MDS) level, not user level
  - Users can create files and directories with various striping characteristics
    - New files inherit the striping of the parent directory
    - Striping across more OSTs generally leads to higher peak performance on large files, but may not be best for small files
      - Maximum file size per OST is 2TB, you must stripe a file greater than 2TB
      - Maximum number of OSTs per file is 160, maximum file size is 320TB
    - CANNOT change the stripe information on an existing file
    - CAN change the stripe information on a directory
  - Improper striping, such as in the following list, may result in inefficient use of your Lustre file system:
    - Writing a many large files to a single OST
    - Creating a directory where files do not circle through the OSTs
    - Striping a small file across many OSTs

lfs Command

- **Use the `lfs setstripe` command to manage striping characteristics**
  - To define striping for a file or directory:

`lfs setstripe [--count | -c]` `[--pool | -p] [--offset | -o][--size | -s]` **`<dir|filename>`**	
`--count|-c`	stripe count, 0 means use the default
`--pool|-p`	name of OST pool
`--offset|-o`	starting OST, -1 means use the default (round robin)
`--size|-s`	stripe-size, 0 means use the default

    - Defaults are defined in the Lustre configuration file
- **To view striping for a file or directory:**

```
lfs getstripe <file-name|dir-name>
```

lfs Command Example

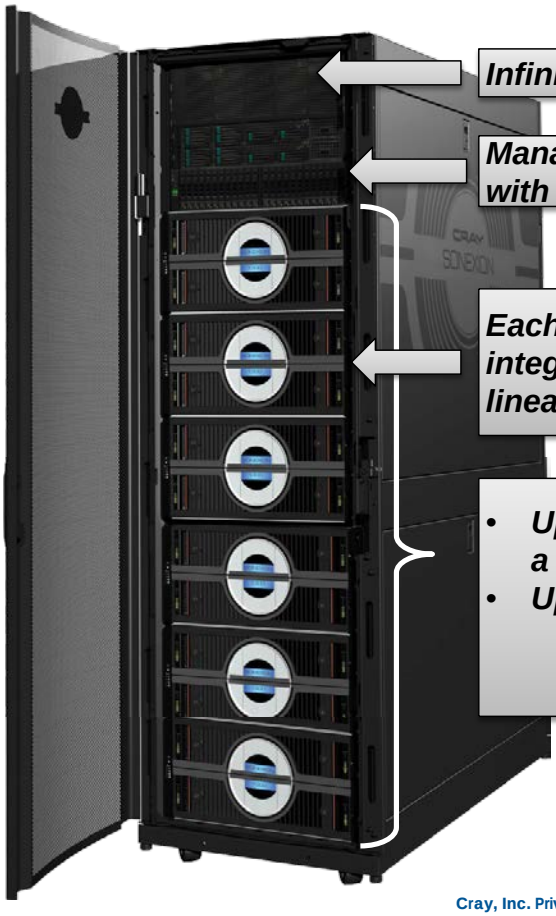
```
rns> mkdir /lus/scratch/rns/lustre_test
rns> lfs getstripe /lus/scratch/rns/lustre_test
    lustre_test
    stripe_count:   1 stripe_size:    1048576 stripe_offset:  -1
rns> cd /lus/scratch/rns/lustre_test
rns/lustre_test> touch file_one
rns/lustre_test> lfs getstripe file_one
    file_one
    lmm_stripe_count:   1
    lmm_stripe_size:    1048576
    lmm_pattern:        1
    lmm_layout_gen:     0
    lmm_stripe_offset:  0
            obdidx           objid           objid           group
                 0        53623103      0x332393f               0
rns/lustre_test> lfs setstripe -c 3 file_two
rns/lustre_test> lfs getstripe file_two
    file_two
    lmm_stripe_count:   3
    lmm_stripe_size:    1048576
    lmm_pattern:        1
    lmm_layout_gen:     0
    lmm_stripe_offset:  5
            obdidx           objid           objid           group
                 5        53611349      0x3320b55               0
                 4        53600944      0x331e2b0               0
                 1        53631902      0x3325b9e               0
```

SONEXION (ClusterStor) Storage Systems

- **The ClusterStor (Sonexion) is a high performance storage solution**
  - The system is an integrated system that optimizes Lustre file system performance, based on best practices
  - The system has three major hardware components:
    - One or more 42U racks
      - The first rack is the base rack, additional racks are expansion racks
        - Each rack contains two InfiniBand (IB) and two GigE switches
      - The base rack contains
        - One Metadata Management Unit (MMU), located in the first (base) rack
          - Lustre management server (MGS)
          - Lustre metadata server (MDS)
        - Cray Sonexion Management Servers (CSMS)
        - Disk enclosure(s) to store metadata and management data
        - One or more Scalable Storage Units (SSUs)
          - Implemented as a 5U high, 84-slot disk enclosure that includes:
          - Each SSU includes two embedded server modules (ESMs) that function as Lustre Object Storage Servers (OSSs)
      - Each expansion rack contains InfiniBand and Ethernet switches and addition SSUs

Each SSU contains eight logical storage devices referred to as Object Storage Targets (OSTs) and two Object Storage Servers (OSSs) that provide file I/O service and network request handling for the client(s). Each OST is a RAID6 (8+2) array of disk drives. The SSU also contains two global hot spares and a mirrored pair of SSDs to store Journaling information.

An Integrated Solution



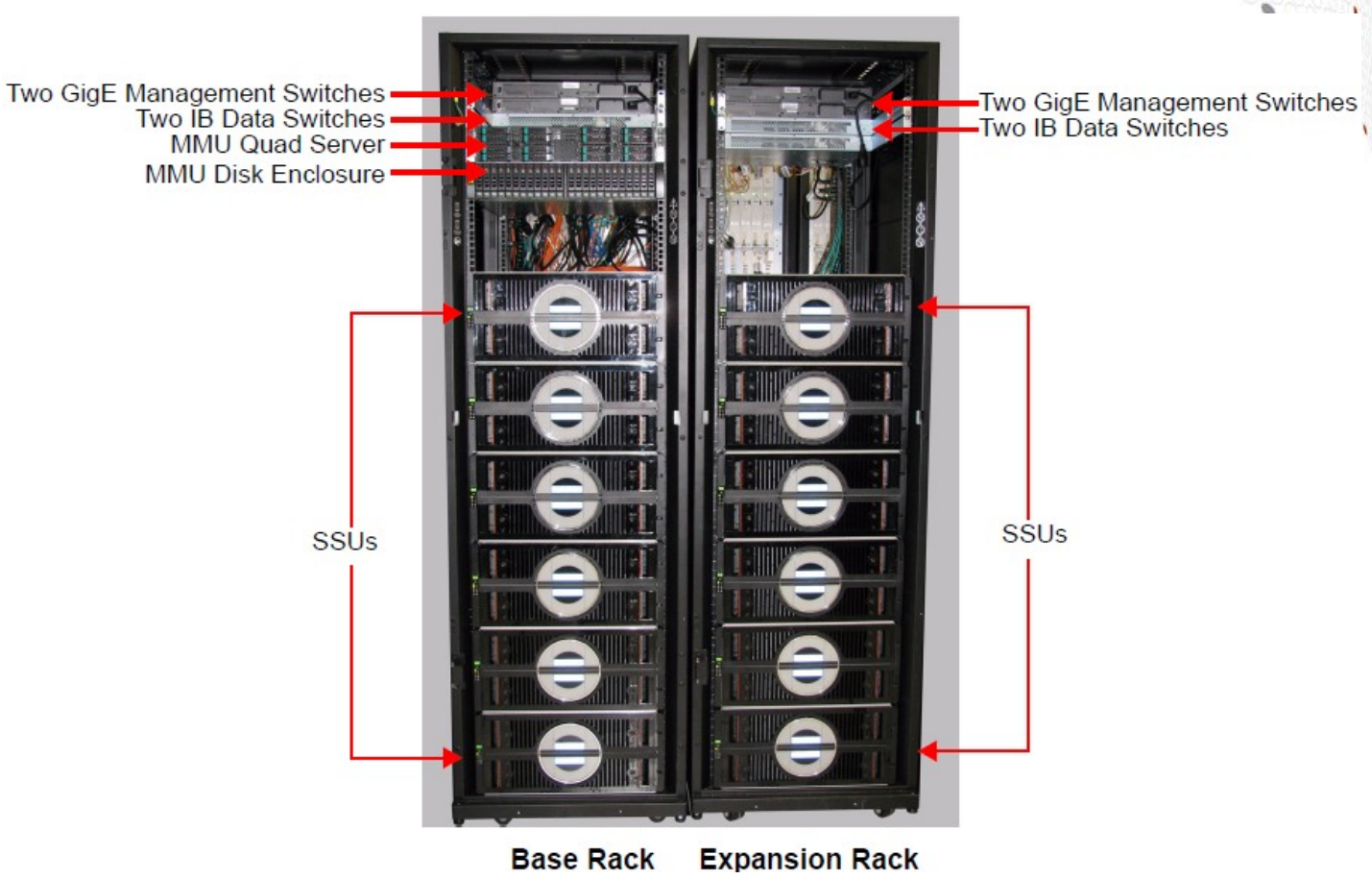
InfiniBand & GigE switches

Management & Metadata servers with related storage enclosure

Each Scalable Storage Unit (SSU) integrates storage and servers for linear scaling of speed & capacity

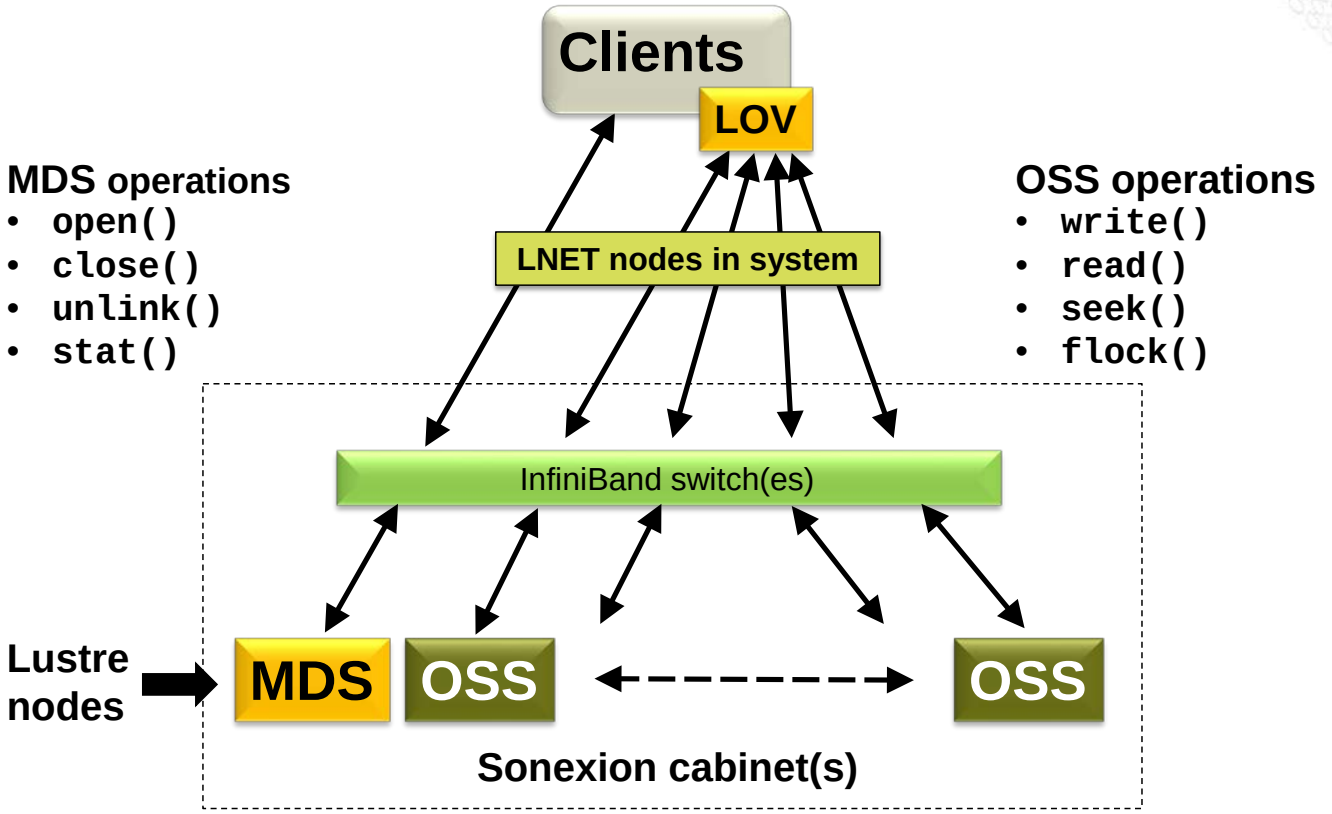
- ***Up to 6 Scalable Storage Units (SSUs) in a base rack***
- ***Up to 6 SSUs in each expansion rack***
  - ***(Seven SSUs are possible, normally systems are configured with up to 6)***

Base Rack with Expansion Rack



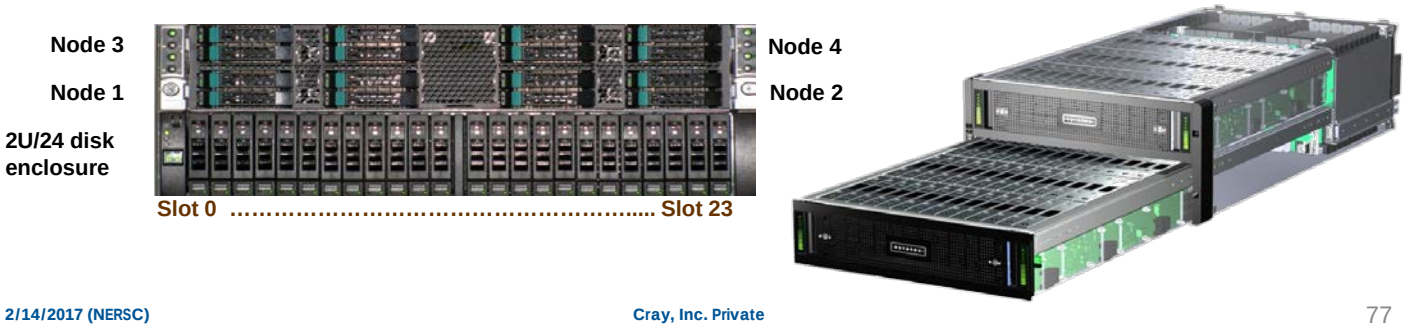
2/14/2017 (NERSC) Cray, Inc. Private 75

External Lustre Node Interaction



Metadata Management Unit (MMU)

- **The MMU is a 2U quad-server that includes:**
  - An active MDS (secondary MGS) in Node 1
  - An active MGS (secondary MDS) in Node 2
  - A primary CSMS in Node 4 (with a secondary CSMS in Node 3)
- One or two 2U/24-bay (shown) or 5U/84-bay disk enclosure(s)
  - The 2U/24-bay contains either 450GB, 600GB, or 900GB disks
    - 600GB or 900GB disks are required if the optional 2U/24 expansion chassis is used
  - The 5U 84-slot chassis uses 2.5” 15K 300GB disks
    - Note: these disks different from the disks used in the SSU
    - The 5U/84 enclosure is available for very large file systems



Scalable Storage Units (SSUs)

- **Scalable Storage Units are based on a 5U 84-slot 3.5” disk enclosure that includes two Storage Bridge Bay (SBB) slots for OSS or expander modules**
  - The SSU enclosures include two OSS controller modules configured as an active/active failover pair.
    - Adding SSUs to the configuration provides increased capacity and performance
    - Each OSS connects to an InfiniBand switch at the top of the rack
  - Expansion Storage Enclosures (ESUs) include two EBOD modules that connect to the OSS controllers in the SSU.
    - ESUs increase capacity without additional bandwidth
    - An SSU can connect to a single ESU
  - All SSUs in a file system must be configured identically

SSU enclosures

Sonexion 1600 and 2000 Differences

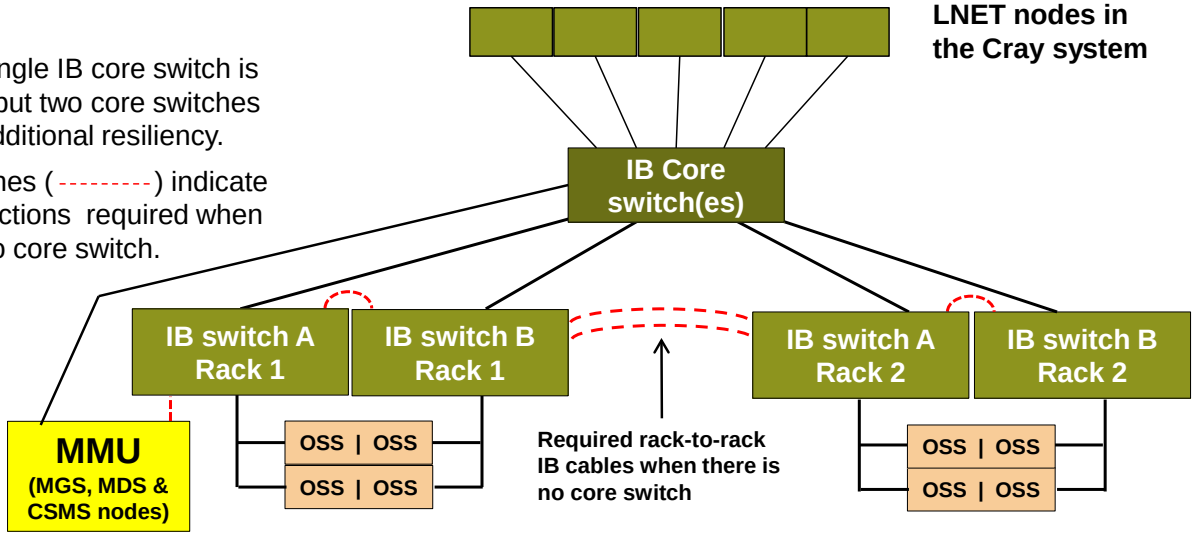
- **The Sonexion 1600 configuration**
  - Each OSS provides a primary path to four of the eight MDRAID, RAID6 8+2 arrays and a failover path to the other four arrays
    - The arrays are also referred to as Object Storage Targets (OSTs)
    - The SSU includes two hot spares for enhanced reliability & availability and has two SSDs for Journaling
- **The Sonexion 2000 configuration**
  - Each OSS provides a primary path to one of the GridRAID RAID6 arrays and a failover path to the other array
    - The arrays are also referred to as Object Storage Targets (OSTs)
    - The SSU includes two SSDs for journaling

InfiniBand Data Network Overview

- **An InfiniBand "Core" switch is recommended whenever:**
  - Multiple hosts are to be connected to a Sonexion file system
  - The Sonexion system comprises more than two racks
  - Multiple Sonexion systems are connected to the system

Note: A single IB core switch is possible, but two core switches provide additional resiliency.

Dashed lines (---------) indicate the connections required when there is no core switch.

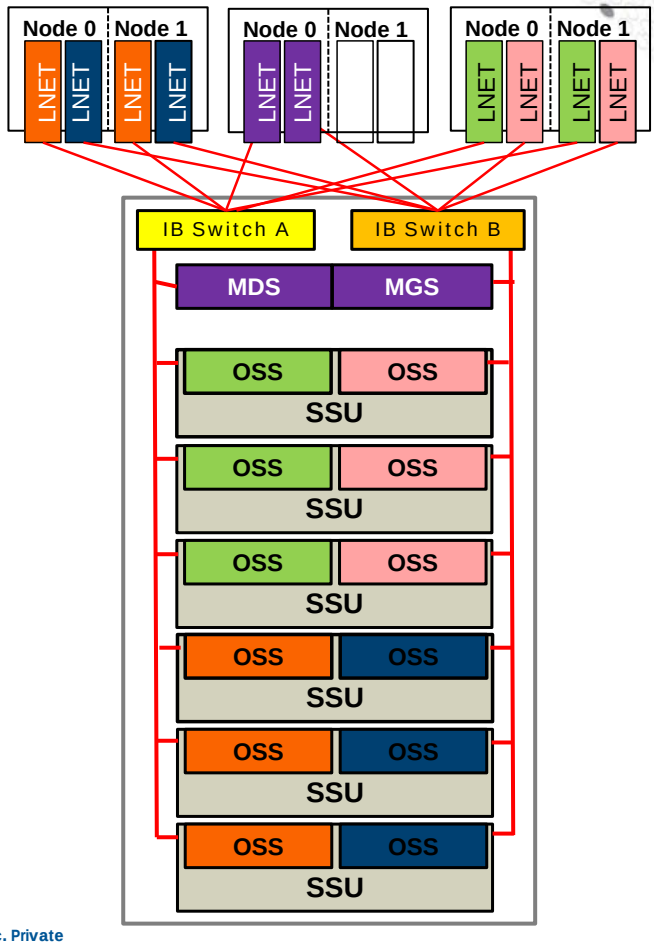


InfiniBand Routing

- **Without a core switch in the configuration of the Sonexion certain bad routing can occur in the IB switches in the top of the Sonexion racks**
- **Fine grained routing (FGR) resolves this issue**
  - FGR is applied to both the Cray system and the Sonexion
  - FGR confines the communication between sets of LNET nodes and OSS in the Sonexion system
    - Communications from the compute nodes (Lustre clients) is directed over the Cray High Speed Network (HSN)
    - Based on the system a recommended ratio of I/O nodes to OSS is provided
      - For the Sonexion 1600 the ration is 2 I/O connection to 3 OSS's
        - The OSSs on either the right or left of the SSUs
        - InfiniBand cabling in the Sonexion rack provides a HA configuration by routing all the cabling from the OSSs on the right or left of the rack to one of the top or bottom IB switch in the top-of-rack
      - For the Sonexion 2000, it is one-to-one, but implemented in 6-to-6

Fine Grain Routing Ratios

- **Sonexion 1600**
  - The ratio is 2:3
    - Two LNET nodes connecting to 3 OSSs in each group.
      - The 2 IB HCA cards within a node cannot be assigned to the same LNET group
- **Sonexion 2000**
  - The ratio is 6:6 (1:1)
    - Six LNET nodes connecting to 6 OSSs in each group.
      - The 2 IB HCA cards within a node cannot be assigned to the same LNET group

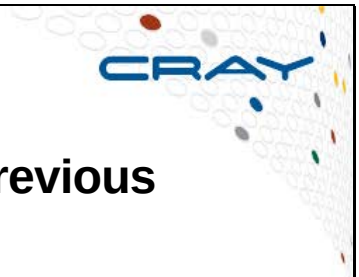


2/14/2017 (NERSC) Cray, Inc. Private 83

Two InfiniBand HCAs installed on the same I/O node cannot be connected to the same LNET group.

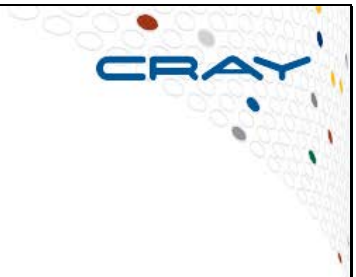
SLURM
(Simple Linux Utility for Resource Management)

Slurm



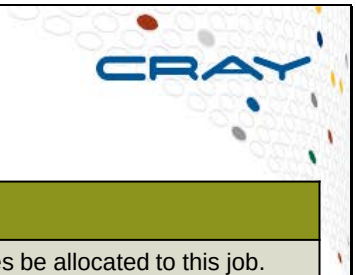
- **At NERSC Slurm replaces Torque/Moab on previous systems**
  - More information related to NERSC can be found at: https://www.nersc.gov/users/computational-systems/cori/running-jobs/

Advantages of using SLURM



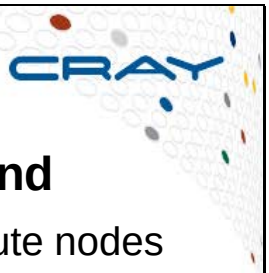
- **Fully open source.**
- **Extensible (plugin architecture).**
- **Low latency scheduling.**
- **Integrated "serial" or "shared" queue.**
- **Integrated Burst Buffer (DataWarp) support.**
- **Good memory management.**
- **Specifics about Cray System**
  - Runs without Cray ALPS (Application Level Placement Scheduler)
    - No `aprun`, `apstat`, nor `apsys`
  - Batch scripts run on the head compute node directly.
  - Easier to use
    - Less chance for contention compared to shared MOM node.

srun options



Full Name	Abbreviated	Definition
`--nodes`	`-N`	Request that a minimum number of nodes be allocated to this job.
`--ntasks`	`-n`	Specify the number of tasks to run.
`--cpus-per-task`	`-c`	Request that *ncpus* be allocated **per process**. This may be useful if the job is multithreaded and requires more than one CPU per task for optimal performance.
`--mem-per-cpu`		Minimum memory required per allocated CPU in MegaBytes.

Interactive Jobs



- **To run an *interactive* job, use the `salloc` command**
  - The job will allocate resources from a list of available compute nodes
    - If there are no compute nodes available the command line will hang waiting for resources.
  - Once resources are allocated, use the `srun` command to launch your application
    - You cannot exceed the resources you requested with `salloc`
    - To request an interactive session with 2 nodes, 8 tasks and 4 tasks per node, and to run on $SCRATCH system, do:

```
% salloc -N 2 -n 8 -p debug -t 00:30:00 -L SCRATCH
```

  - One can request the same options above via the **`srun`** command. **`srun`** will automatically bind 4 MPI tasks per node

```
% salloc -N 2 -p debug -t 00:30:00 -L SCRATCH
salloc: Granted job allocation 5313
salloc: Waiting for resource configuration
salloc: Nodes nid000[29-30] are ready for job
% srun -n 8 ./myprogram
```

Batch Jobs

- **Batch jobs are jobs that run non-interactively under the control of a "batch script"**
  - A batch script is a text file containing a number of job directives and Linux commands or utilities.
  - Batch scripts are submitted to the "batch system," where they are queued awaiting free resources.
    - The simplest case is running the same number of MPI tasks as there are physical cores on each node. The example below uses 2 KNL nodes in quad cache mode, 68 MPI ranks per node for a total of 136 MPI tasks.

```
#!/bin/bash –l
#SBATCH -p debug
#SBATCH -N 2
#SBATCH –t 00:30:00
#SBATCH -C knl,quad,cache
#SBATCH -L SCRATCH
export OMP_NUM_THREADS=1 # only needed for hybrid MPI/OpenMP codes
                         # built with “-qopenmp“ flag
srun -n 136 ./mycode.exe
```

2/14/2017 (NERSC) Cray, Inc. Private 90

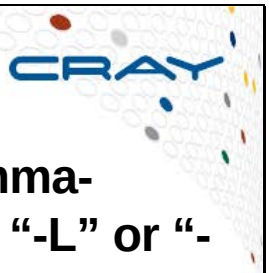
https://www.nersc.gov/users/computational-systems/cori/running-jobs/example-batch-scripts-for-knl/

Batch Jobs

- **The next example uses 2 KNL nodes in quad cache mode, with 32 MPI ranks per node.**
  - This is not fully packed MPI (the number of MPI ranks per node is not evenly divisible by 68) so the following two options are needed on the `srun` command line: "`-c`" and "`--cpu_bind=cores`".
  - The value following the "`-c`" option sets the number of logical cores to allocate per MPI task.
  - For this example it should be set as floor(68/32)*4=8 which allocate 8 logical cores per MPI task. Since there are 4 logical cores per physical core, this will bind each MPI task to 2 physical cores.

```
#!/bin/bash -l
#SBATCH –p debug
#SBATCH –N 2
#SBATCH -C knl,quad,cache
#SBATCH -S 2
#SBATCH -t 00:30:00
#SBATCH -L project
export OMP_NUM_THREADS=1
srun -n 64 –c 8 --cpu_bind=cores ./mycode.exe
```

File System Licenses



- **The syntax for specifying the file system is a comma-separated list of file systems with the keyword of "-L" or "--license="**
  - `#SBATCH -L scratch1`
  - `#SBATCH --license=scratch1`
  - `#SBATCH -L scratch1,project`
  - With this new SLURM license feature, a batch job will not start if the specified file system is unavailable due to maintenance, an outage, or if administrators have detected performance issues.
    - This will protect your jobs from failures or decreased performance from known issues.

Monitoring

Command	Description
sqs	NERSC custom script lists jobs in the queue with job ranking
squeue	Lists jobs in the queue
sinfo	Prints queue information about nodes and partitions
sbatch batch script	Submits a batch script to the queue
scancel jobid	Cancel a job from the queue
scontrol hold jobid	Puts a job on hold in the queue.
scontrol release	Releases a job from hold
scontrol update	Change attributes of submitted job.
scontrol requeue	Requeue a running, suspended or finished Slurm batch job into pending state.
scontrol show job jobid	Produce a very detailed report for the job.
sacct -k,--timelimit-min	Only send data about jobs with this time limit.
sacct -A account_list	Display jobs when a comma separated list of accounts are given as the argument.
sstat	Display information about CPU, Task, Node, Resident Set Size and Virtual Memory
sprio	Display information about a job's scheduling priority from multi-factor priority components.

https://www.nersc.gov/users/computational-systems/cori/running-jobs/monitoring-jobs/