University of Wyoming

Using NCAR’s High Performance Computing Resources

With Focus on the New Supercomputer Derecho

Computational & Information Systems Lab (CISL)
High Performance Computing Division (HPCD)

NCAR

September 25, 2023

NSF

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

Agenda

Agendas

10:00 – 10:30 **Welcome, Introduction & Overview of NCAR's HPC Resources – Irfan Elahi**
NCAR HPC Resource Allocations for University of Wyoming Projects – Dave Hart

10:30 – 12:00 **System Access, Storage, & Software Environment – NCAR Consulting Services Group**

Batch Job Submission, Data Analysis Resources

12:00 – 13:00 Lunch

13:00 – 14:00 **Additional Resources & Getting Help – CSG**
Open Discussion / Q&A / Support "Office Hours"

Introduction -

Overall HPCD Resources & NCAR/Wyoming Supercomputing Center

NWSC-3 (*Derecho & Destor*) Project Status

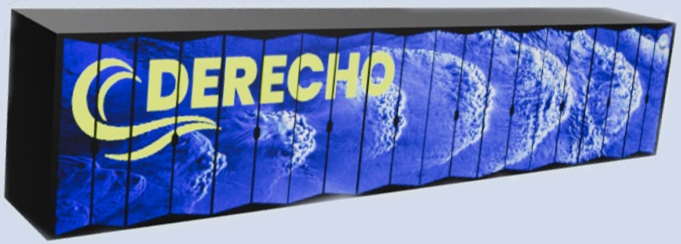
- Planning, Procurement, Facility Modifications
- Acceptance Test Phase (ATP) March-May
  - Functional & Resilience Tests
  - NWSC-3 Benchmark Tests
  - Availability Tests
- System Acceptance
- Accelerated Science Discovery (ASD)
  - June 5, 2023
- ***Open Derecho to all users***
  - ***August 9, 2023+***
- ***Decommission Cheyenne and GLADE -2***

NCAR’s High -Performance Computing, Data, & Analysis Resources: 2023

HPC Systems

2017 — 2023

SGI/HPE
*4032 Nodes, **145,152 Cores**, 313 TB total memory, **4.79 PFlop/s***
#21 Supercomputer in the world at debut



mid-2023 →

Cray/HPE
*2570 Nodes, **323,712 CPU Cores**, 680 TB total memory*
3.5X performance vs Cheyenne
***328 NVidia A100 GPUs** provides 20% overall performance, **19.87 PFlop/s** (projected)*

Data Analysis & Visualization

Casper: heterogeneous system for data analysis & viz.
- 75 High-Throughput Computing nodes
- 9 visualization nodes with accelerated graphics
- 10 dense GPU nodes for AI/ML, Code Development
- 4 nodes for Research Data processing
- 2 1.5TB large memory nodes

jupyterhub

http://projectpythia.org

PROJECT PYTHIA

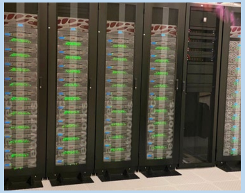
GeoCAT

https://geocat.ucar.edu

CISL develops specialized visualization software & services for Earth Science applications

VAPOR
Visualization & Analysis Platform

High Performance Storage



GLADE & Campaign Storage
- **132PB** long-term, online storage
- 17,464 hard drives
- 56 servers

Derecho ‘scratch’ Storage
- **60PB** short-term storage
- Principally supports HPC jobs
- 5,088 hard drives
- 24 servers

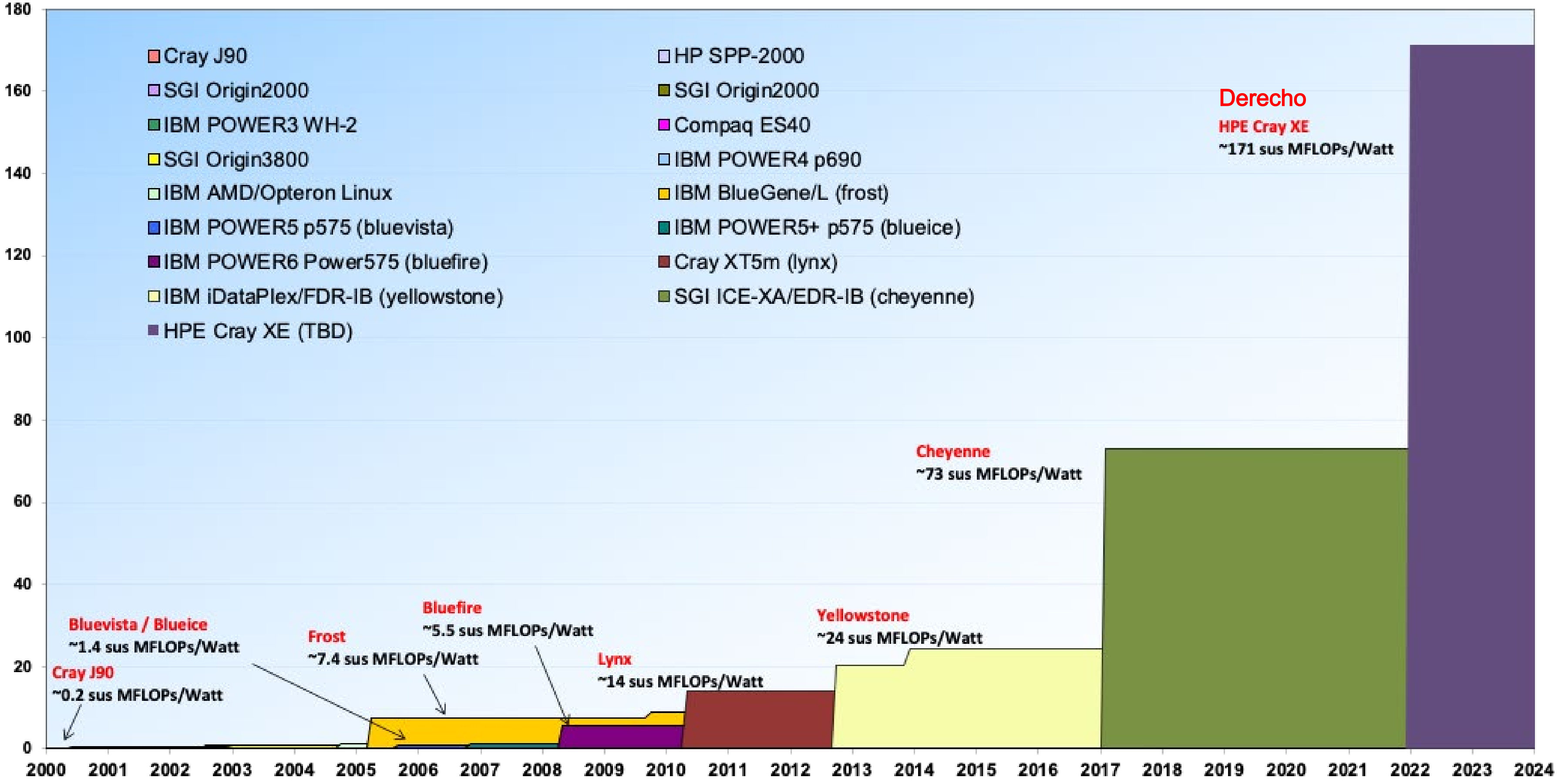
Stratus Object Storage
- **5PB object storage** system
- 588 hard drives
- 6 servers

Quasar Tape Library
- **35PB long term archival** storage
- 22 IBM TS1160 tape drives
- 1774 20TB tape cartridges
- 216 hard drives
- 2PB disk cache
- 5 data mover servers

Cheyenne & Derecho side -by-side (Hardware)

Number of Cores	145,152 processor cores	2.3-GHz Intel Xeon E5-2697V4 (Broadwell) processors. 16 flops per clock	323,712 processor cores	3rd Gen AMD EPYC™ 7763 Milan processors
Number of Nodes	4,032 comp nodes	Dual-socket nodes, 18 cores per socket	2,488 CPU-only computation nodes + 82 GPU nodes	Dual-socket nodes, 64 cores per socket 256GB DDR4 memory per node. GPUs are Single-socket nodes, 64 cores per socket 512GB DDR4 memory per node, 4 x NVIDIA 1.41 GHz A100 Tensor Core GPUs per node and 600 GB/s NVIDIA NVLink GPU intercont.
Number of Login Nodes	6 login nodes	Dual-socket nodes, 18 cores per socket. 256 GB memory/node	6 CPU login nodes + 2 GPU develop and testing nodes	Dual-socket nodes with AMD EPYC™ 7763 Milan CPUs, 64 cores per socket, 512GB DDR4-3200 memory. Dual-socket nodes with AMD EPYC™ 7543 Milan CPUs, 32 cores per socket, 2 NVIDIA 1.41 GHz A100 Tensor Core GPUs per node 512GB DDR4-3200 memory
Total Memory	313 TB total system memory	64 GB/node on 3,168 nodes, DDR4-2400. 128 GB/node on 864 nodes, DDR4-2400	692 TB total system memory	637 TB DDR4 memory on 2,488 CPU nodes 42 TB DDR4 memory on 82 GPU nodes 13 TB HBM2 memory on 82 GPU nodes
Interconnect	Mellanox EDR InfiniBand high-speed interconnect	Partial 9D Enhanced Hypercube single-plane interconnect topology. Bandwidth: 25 GBps bidirectional per link Latency: MPI ping-pong < 1 µs; hardware link 130 ns	HPE Slingshot v11 high-speed interconnect	Dragonfly topology, 200 Gb/sec per port per direction. 1.7-2.6 usec MPI latency. CPU-only nodes - one Slingshot injection port. GPU nodes - 4 Slingshot injection ports per node
Sust Equiv Perf (SEP)	3 times Yellowstone computational capacity	Comparison based on the relative performance of CISL High Performance Computing Benchmarks run on each system.	~3.5 times Cheyenne computational capacity	Comparison based on the relative performance of CISL's High Performance Computing Benchmarks run on each system.
Peak Performance	3.5x Yellowstone peak perf	5.34 peak petaflops (vs. 1.504)	3.5x Cheyenne peak perf	19.87 peak petaflops (vs 5.34)

Sustainability: Power Efficiency (Sustained MFLOP/sec per Watt)

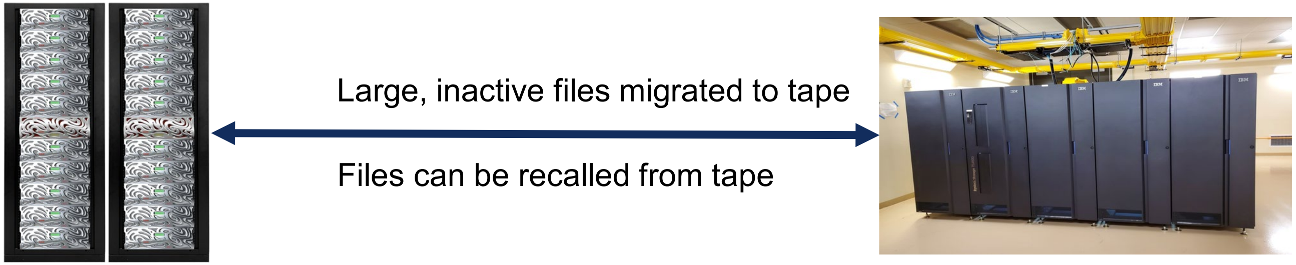


2023 Casper Augmentation

- CISL is currently refreshing components of Casper with a new hardware procurement this fiscal year.
  - We solicited user feedback during the planning phase of this procurement to select the hardware “flavor(s)” included in this purchase.
  - Procurements underway, with initial additional components expected in the coming months.
- We are continuously seeking user feedback on impressions of the current generation of Casper resources as well as desires or hard requirements for the future hardware and software environment.

Campaign Storage Tape HSM Tier Concept

- **Goal:** Expand Campaign Storage capacity for users
- **Constraint:** Budgetary limits makes it hard to buy more hard disks and filesystem licenses.
- **Solution:** use a lower cost tape backend
- Manual and automatic migration and recall options



Large, inactive files migrated to tape

Files can be recalled from tape

Campaign Storage

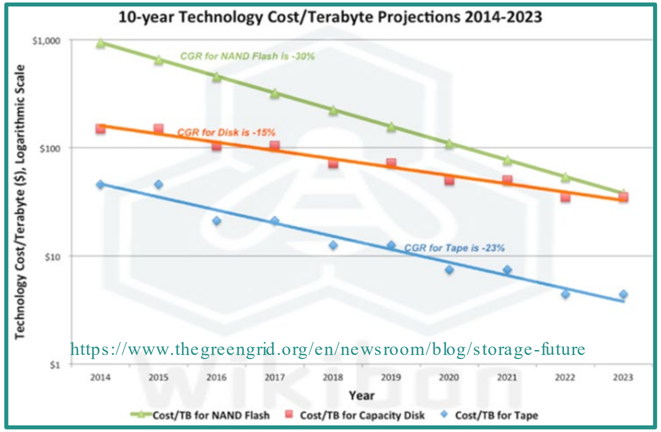
Hard Disk Archival Resource (92 PB capacity)

Quasar Tape Archive

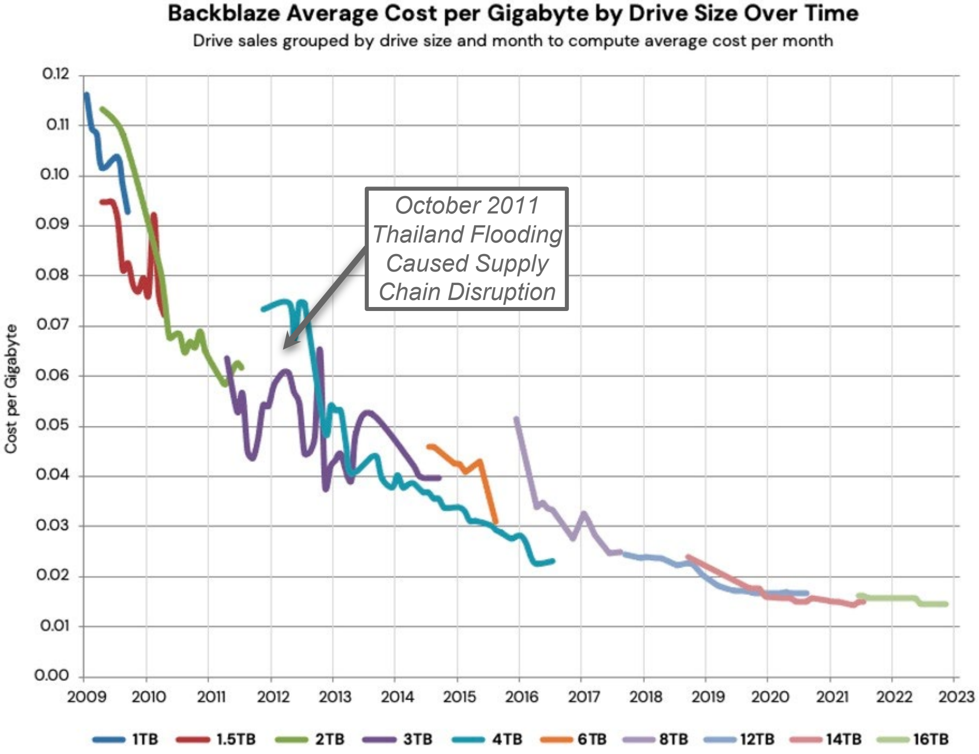
Tape Archive Resource (50 PB capacity)

Hard Disk Storage: Cost Trends

- For decades a constant budget would routinely support massive increases in storage as hard drive capacity quickly grew
  - Unfortunately this “free lunch” phase of disk capacity is over
- Between 2009-2016,
  - Drive Capacity increased ~8X,
  - Cost/GB decreased ~3X
- Between 2016-2023,
  - Drive Capacity increased ~3-4X,
  - Cost/GB decreased ~2X
- To realize more capacity we can no longer rely on larger drives, rather we need many more drives (and servers, network, power)→ **$$$**


10-year Technology Cost/Terabyte Projections 2014-2023

https://www.thegreengrid.org/en/newsroom/blog/storage-future


Backblaze Average Cost per Gigabyte by Drive Size Over Time

Drive sales grouped by drive size and month to compute average cost per month

October 2011 Thailand Flooding Caused Supply Chain Disruption

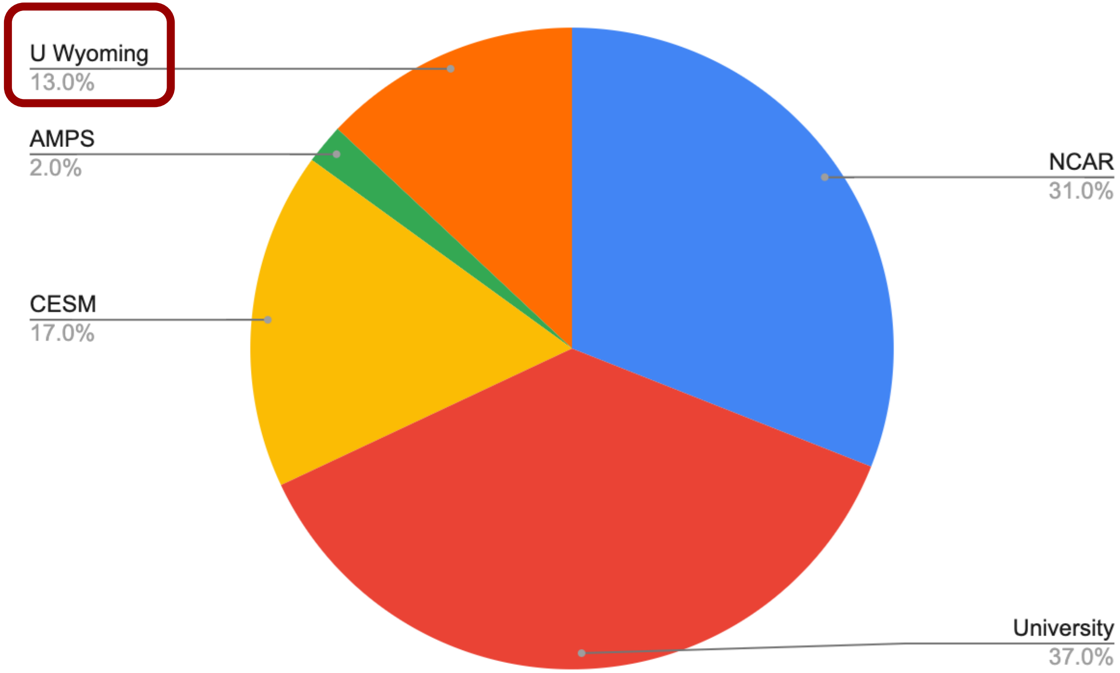
https://www.backblaze.com/blog/hard-drive-cost-per-gigabyte

DERECH
DERECHO

Resource Allocations

University of Wyoming Process

Derecho Community Portions



Community	Annual Portion
University	981 million core-hours
	1 million GPU-hours
NCAR	825 million core-hours
	850,000 GPU-hours
CESM	451 million core-hours
	465,000 GPU-hours
Wyoming	345 million core-hours
	355,000 GPU-hours
AMPS	53 million core-hours
	55,000 GPU-hours

Allocations for the University of Wyoming

- Information on U Wyoming allocations is at https://www.uwyo.edu/nwsc/
- Allocations are available for
  - Classroom activities
  - Small-scale research (up to ~2 million core-hours or ~3,000 GPU-hours)
  - Large-scale research (the sky's the limit, within the amounts available to U Wyoming)
- Small-scale and classroom project requests are accepted anytime
- Large-scale project requests are accepted twice per year for review by the Wyoming Resource Allocation Panel (WRAP)
- Projects are available to support a wide range of domains, in an "Earth system science–first" approach
  - I.e., priority given to Earth system science activities, and other strategic Wyoming activities approved as resources are available— and space is available!
- Projects with collaborators from NCAR or other institutions are particularly encouraged
- For questions, contact wrap@uwyo. edu

Migrating Cheyenne allocations to Derecho

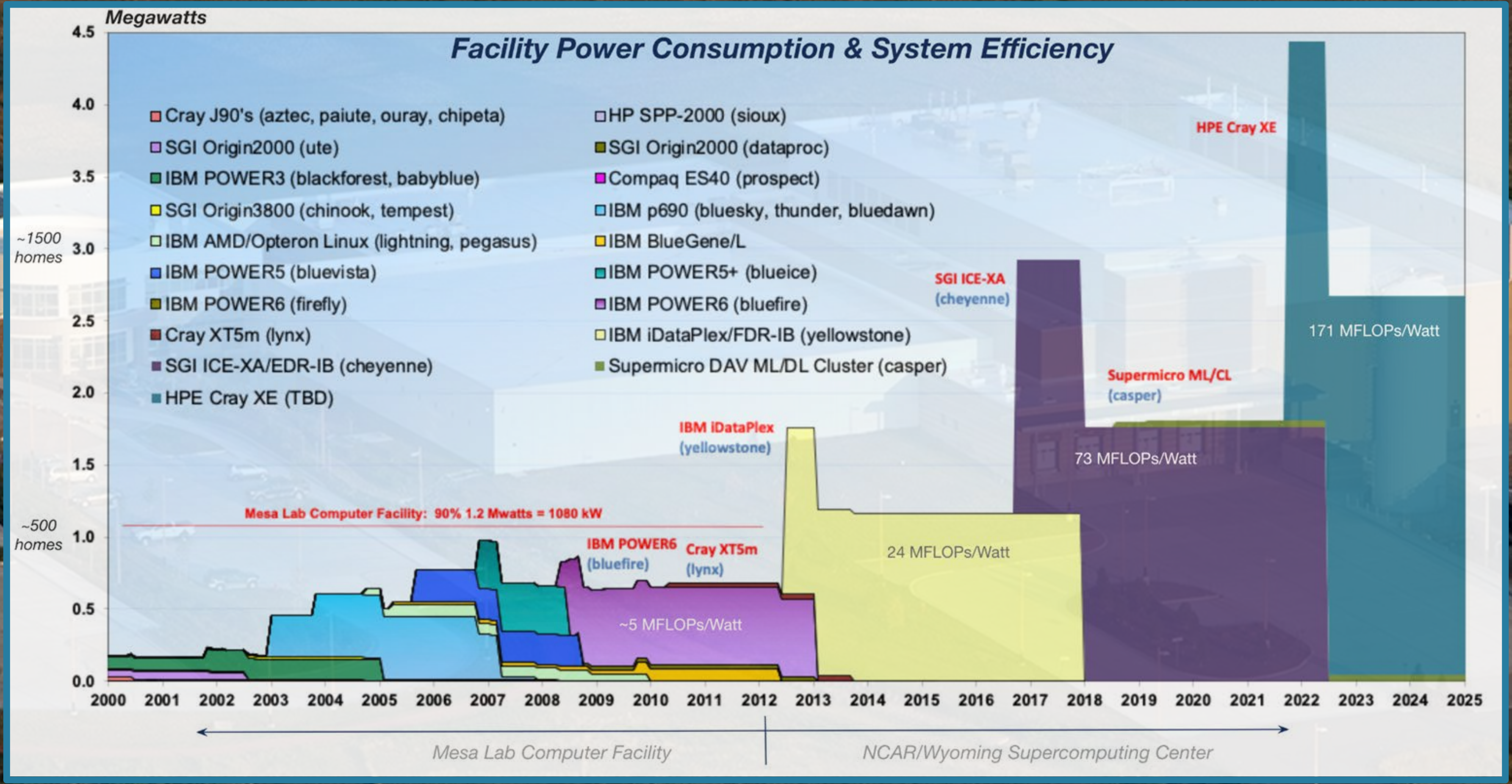
- **All projects currently using Cheyenne need to move their work to Derecho at their earliest convenience**
- Cheyenne ***will*** be retired at the end of 2023
- To move core-hours from Cheyenne, simply email help@ucar.edu and we'll take care of the transfer
- To add GPU-hours to your project, contact wrap @uwyo.edu to see how best to make that request

(Transition to NCAR HPC Training Slides)

The following reference material provides additional Derecho -specific hardware and configuration details that may benefit users familiar with *Cheyenne*

NWSC Facility and Supporting Infrastructure

Preparing a NSF Large Facility for Expanded HPC Operations


Facility Power Consumption & System Efficiency

NCAR
UCAR

Supercomputing System Efficiency - *Historical Overview*

NWSC - Derecho By The Numbers

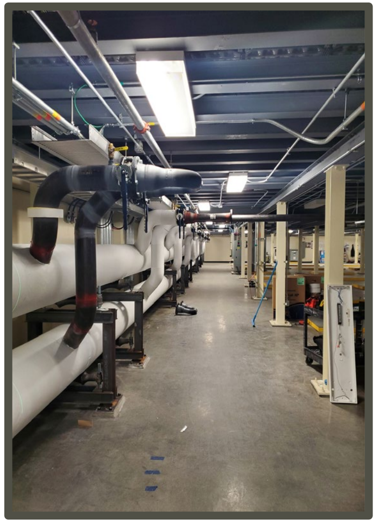
- System Power Requirements
  - (33) 150 Amp 480v Circuits (Compute)
  - (6) 60 Amp 480v Circuits (CDUs)
  - (20) 60 Amp 208v Circuits (Storage and River Racks)
  - 2.32 mW (Compute)
  - 60 kW (Storage)
- System Cooling Requirements
  - (6) 4 Inch 65 deg F Chilled Water Supply and Return Connections (CDUs)
  - (8) 2 Inch 65 deg F Chilled Water Supply and Return Connections (CRACS)
  - 366 kBTU / HR
  - 650 - 800 Gallons Per Minute Chilled Water Flow
- Floor Weight / Dimension Requirements
  - 100,467 Pounds - Derecho Compute System
    - 800 Square Feet
  - 10,320 Pounds - Derecho Storage System
    - 225 Square Feet
- Hardware Components
  - 5,072 CPU sockets across >2,500 nodes with 323,712 processor cores total
  - 332 NVIDIA A100 GPUs with 6912 CUDA cores each, 2,294,784 CUDA cores total
  - >22,500 hard disks in GLADE, Campaign Storage, and Derecho Scratch
  - 5.565 miles of network cables

NWSC-3 Infrastructure Preparation Steps

Four major initiatives required before Derecho installation could begin:

1. **Forecasting Future Loads**
   - 2-3 years before CISL HPC deployment
   - NWSC-3 initial load forecasts were 2-4 additional Megawatts (MW)
     - Module B HPC Power Source 1.7 MW Availability-Module A capacity needed
2. **Capacity Construction**
   - Large Block Load Mechanical and Electrical Components
   - Supports Multiple Generation of HPC Systems
3. **Fit-up**
   - Specific Mechanical and Electrical components for awarded HPC system
   - Potential re-use of infrastructure in future HPC systems
4. **Planned Outages**
   - Utility level alterations
   - Safety Concerns

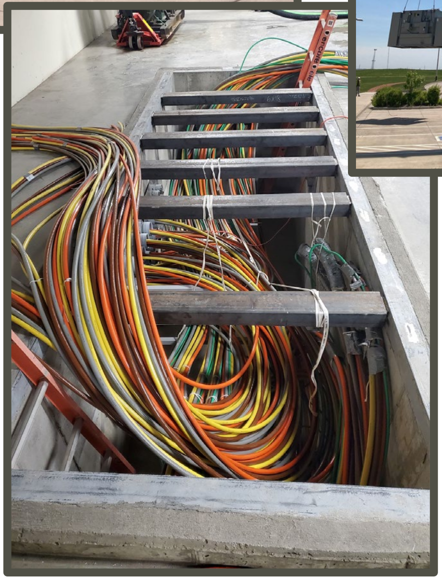
Capacity: Mechanical Construction



18” Chilled Water Piping Header Installation - Lower Module A

These images show the 18” piping being installed from delivery to the NWSC, the welding installation, and finally the pipe being insulated.

Capacity: Electrical Construction



24kV to 480v Transformer Utility HPC Substation -2 (TUSH-2) Installation

These images show some of the prep work needed to augment the underground for the new transformer, and then the process of setting the new 25,000 pound transformer in place. Finally the secondary side wiring is shown in the vault below Module A.

NCAR
UCAR

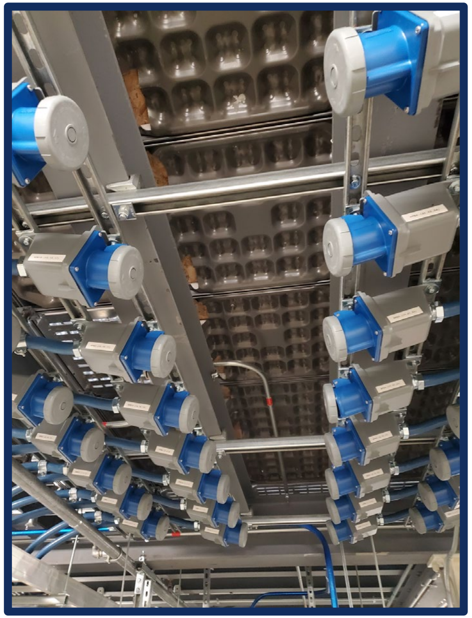
Fit-Up: High-Level Statement of Work

Derecho Fit-up Handled as a Design / Build Project:

- Saunders Construction won procurement May 2021
  - RMH Engineering used to complete the design of the system
  - Encore Electric and Murphy Mechanical sub-contracted partners
  - Same team as the Capacity Construction Project
- Workflow
  - MUS for Derecho and NCAR requirements shared
  - RMH creates Construction Documents (CDs)
    - Saunders / Encore / Murphy participate in design
      - Electrical Component Selection
      - Mechanical Component Selection
    - Many activities happening in parallel
  - Construction starts / completes
  - Derecho Delivery and final infrastructure connections completed

Fit-Up

Production E1000 Pod Pin and Sleeve Receptacles



480v Distribution and Branch Circuit Panels - Derecho Compute

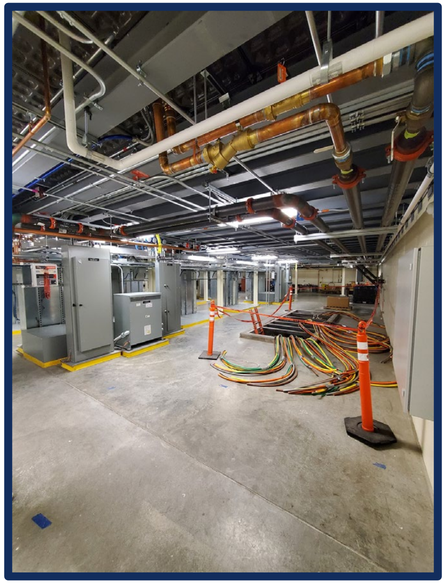
Pre-located J-boxes for HPC Rack Power Whips

Fit-Up

CRAC Installation for Derecho Air Cooled Load and CDU Hose Mock-up

CDU Lineset with Flowset and Hose Mock-up

Completed CRAC Lineset

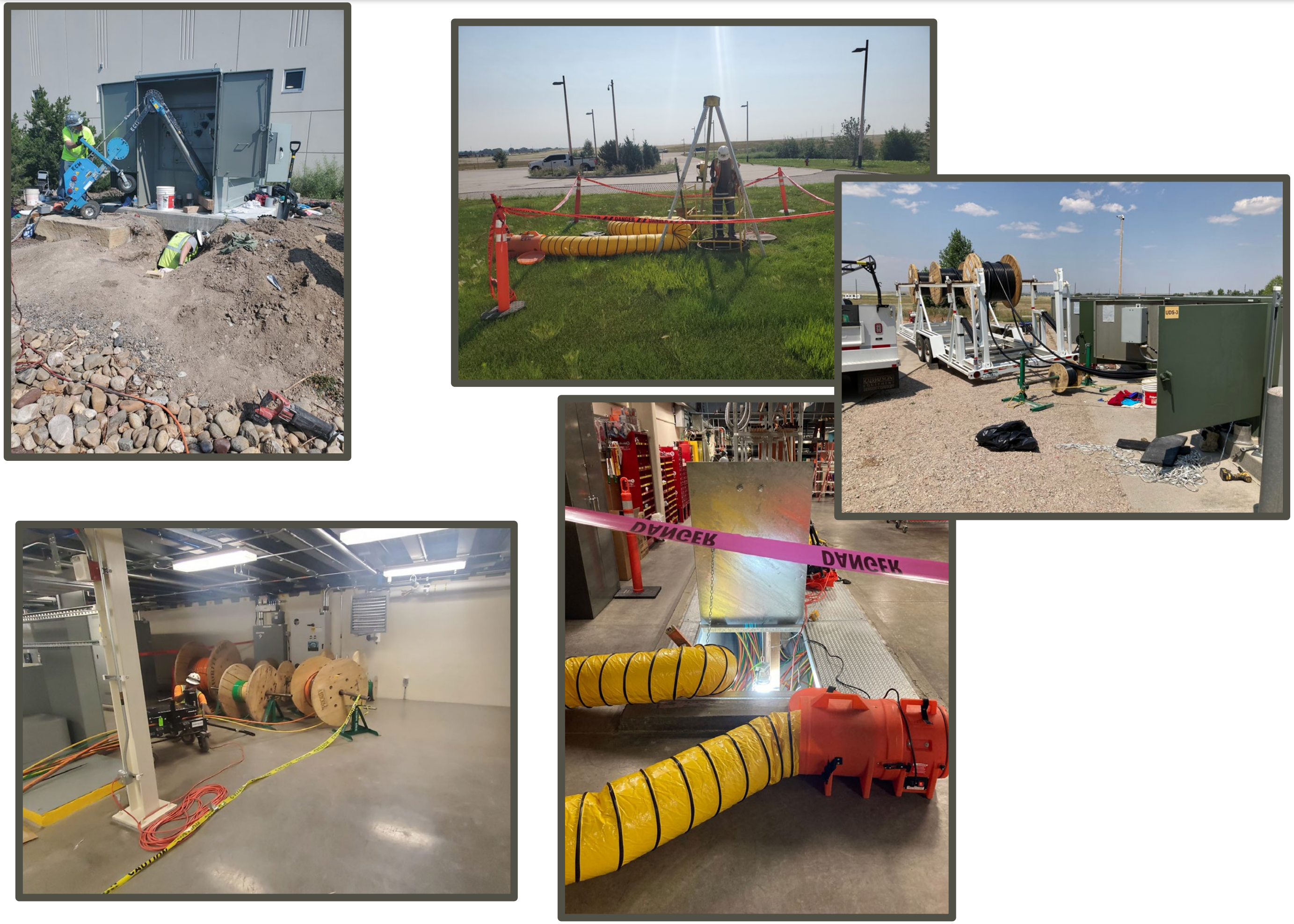


Derecho Preparation Outages

Three types of outages were completed throughout the Derecho facility preparations:

1. **Small Shutdowns** (5 total)
   - Single or Multiple Branch Circuits- No impact to computer room operations
2. **Medium Shutdowns** (4 total)
   - Larger block loads that can potentially impact computer room operations
     - Essential Power Modifications- Mechanical Redundancy
     - Essential UPS Power Modifications- Networking Equipment at the NWSC
     - Mechanical Equipment Start-ups and Commissioning
3. **Large Shutdown** (1 total)
   - July 26th - July 30th 2021
   - Medium Voltage (24,900 V) system augmentations
   - Computer Room UPS alterations
   - Mechanical Power Alterations
   - Module A Mechanical Header Commissioning

Preparation Outages



Photos From Large Shutdown

These images show some of the work completed during the NWSC large outage. The top (3) photos are of the Medium Voltage Crew pulling in the 24.9kV cables.

The pictures on the bottom half show the UPS wiring being pulled through the vaults.

Note the air purifying systems being used when working in vaults inside or outside the facility.

NWSC Virtual Tour Links

For NWSC Virtual Tour visit:

https://bit.ly/NCAR360

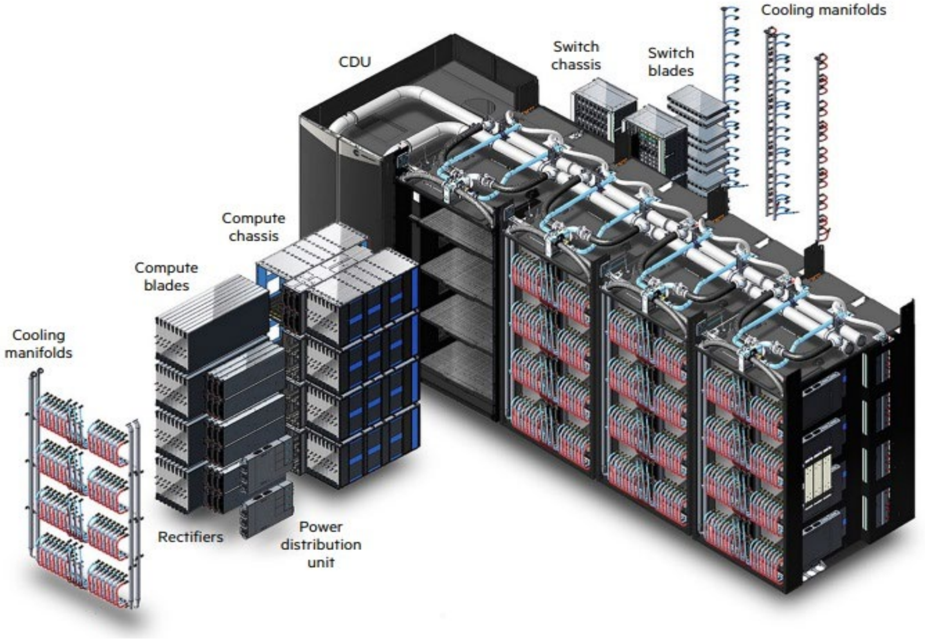
Derecho Construction Virtual Tour is available for viewing at:

https://www.thinglink.com/mediacard/1510396772545986561

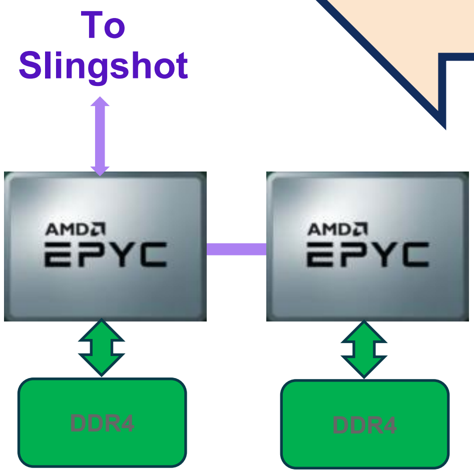
Derecho Architecture & Technology Deep Dive

Derecho (NWSC -3) HPE/Cray Solution

- Complete proposal received from HPE/Cray
  - Includes HPC and PFS
  - Peak: 19.87 PetaFlops
  - 60PB usable file system
- HPE CSEP Exceeds RFP requirement
  - 3.51 CSEP
    - CPU – 2.84 CSEP
    - GPU – 0.67 CSEP
- Large installation base
- Includes onsite 3x FTE support



CPU Node



CPU Cabinet

4 nodes per compute blade

1 slingshot injection

64 blades per cabinet

256 nodes per cabinet

210.7 kW

0.65 tons of mech cooling

~1.3 PFLOPS

0.29 CSEP

GPU Cabinet

2 nodes per compute blade

4 x GPU per node

4 Slingshot Injections

64 blades per cabinet

128 nodes per cabinet

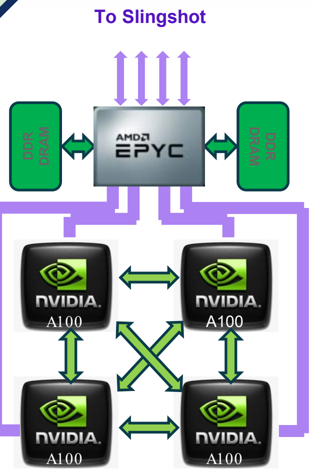
190 kW

0.59 tons of mech cooling

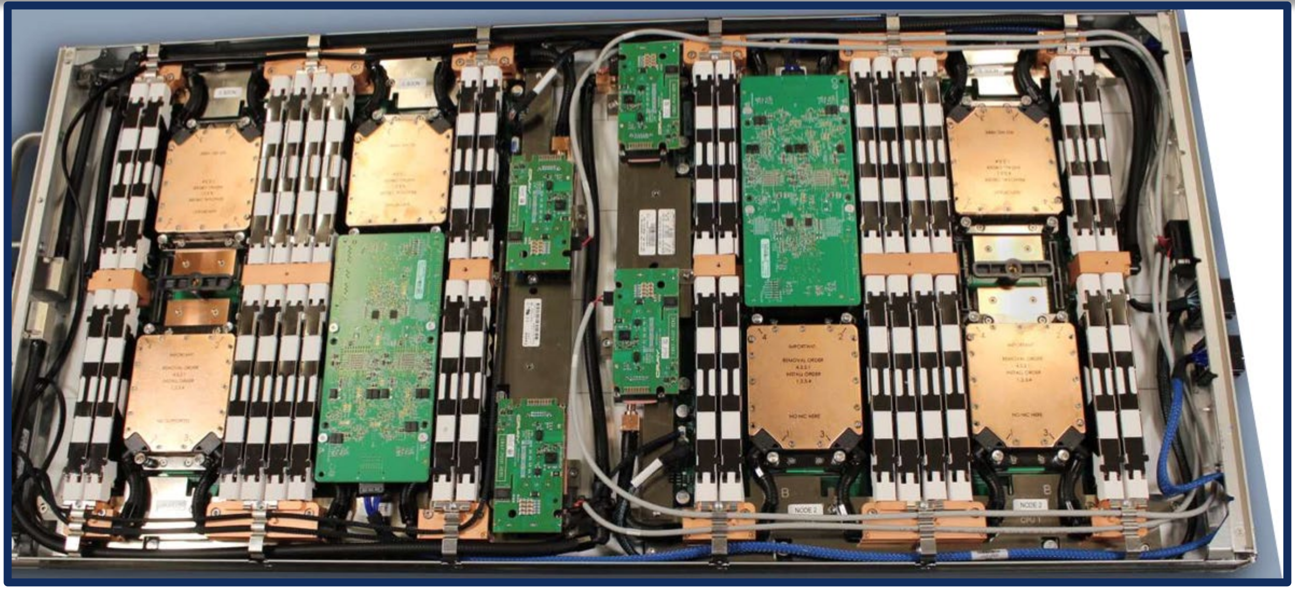
~10.3 PFLOPS

1.04 CSEP

GPU Node

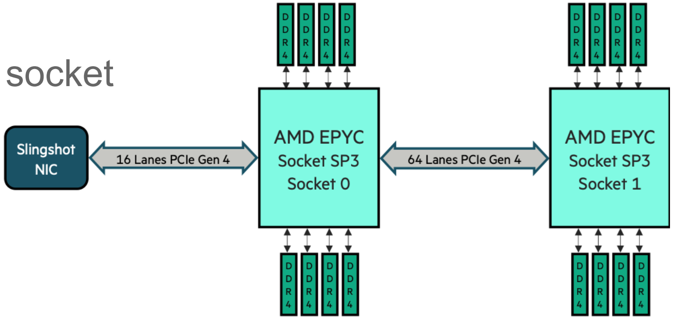


Derecho CPU Node Anatomy

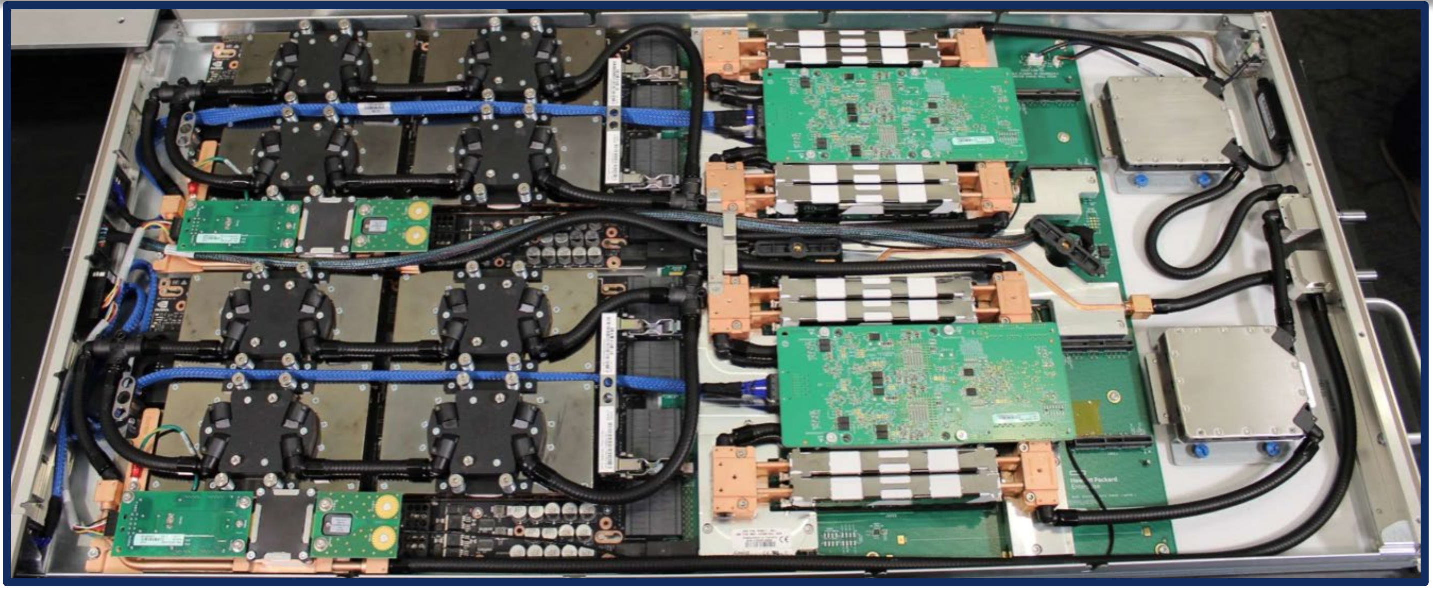


Each CPU compute blade holds 4 liquid cooled nodes, each containing

- 2 AMD EPYC Zen3 “Milan” processors, 64 cores/128 threads per socket
- 16 DDR4 DIMMS, 256GB total RAM
- 1 200 Gb/sec Cray Cassini Slingshot 11 network interface

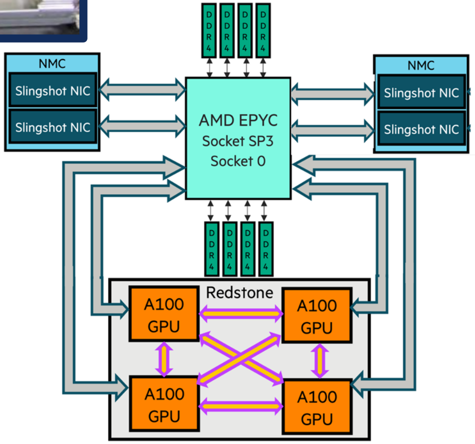


Derecho GPU Node Anatomy



Each GPU compute blade holds 2 liquid cooled nodes, each containing

- 1 AMD EPYC Zen3 “Milan” processor, 64 cores/128 threads
- 8 DDR4 DIMMS, 512GB total RAM (host)
- 4 NVIDIA A100 Ampere GPUs, each with 40GB RAM (device)
- 4 200 Gb/sec Cray Cassini Slingshot 11 network interfaces



DDR4 DDR4 DDR4 DDR4

NMC
Slingshot NIC
Slingshot NIC

AMD EPYC
Socket SP3
Socket 0

NMC
Slingshot NIC
Slingshot NIC

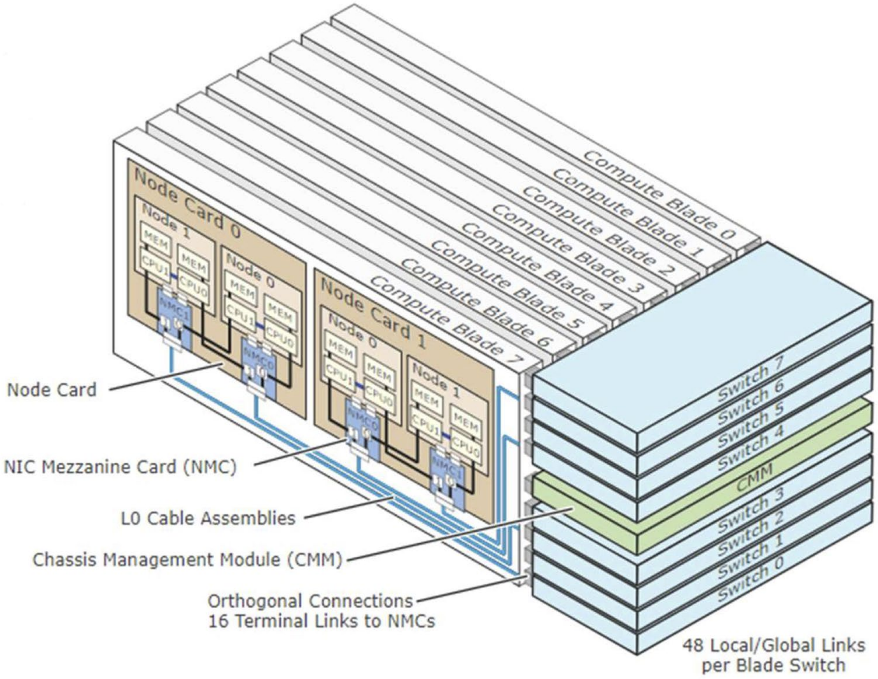
DDR4 DDR4 DDR4 DDR4

Redstone
A100 GPU
A100 GPU
A100 GPU
A100 GPU

Derecho Slingshot Network Overview

The network path begins at the chassis

- Derecho CPU chassis are composed of
  - 8 physical blades with
  - 4 nodes per blade
  - 2 Slingshot switches
- Derecho GPU rack units composed of
  - 8 physical blades with
  - 2 nodes per blade
  - 4 Slingshot switches
- Blades are mounted vertically from the front of a rack, while the switches are mounted horizontally from the back
  - For awareness, nodes in the same blade reside on different switches



Derecho Slingshot Dragonfly Topology Overview

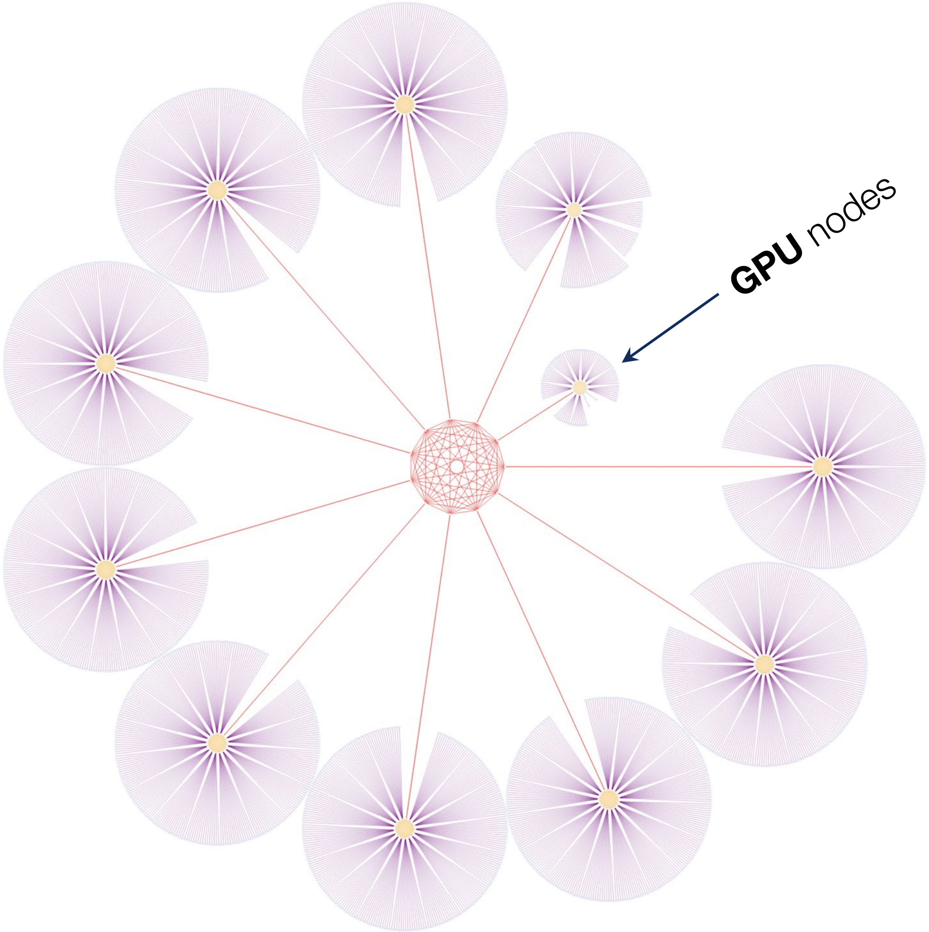
The Slingshot dragonfly network establishes 3 types of fabric connections:

1. Edge (compute nodes, external network connectivity)
2. Local - In-switch group connectivity
3. Global - Switch group interconnect

This can be thought of as small islands of compute systems with dedicated shipping lanes to their peer islands.

Derecho contains 13 switch groups comprised of:

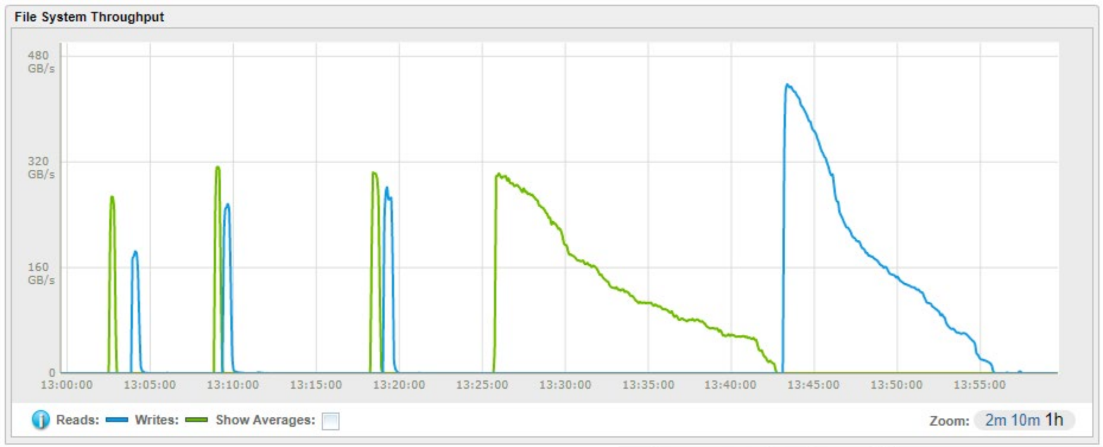
- 1x switch group for River racks (login nodes, Arista router connectivity, PBS, etc.)
- 10x switch groups for Mountain CPU racks
- 2x switch groups for the single GPU Mountain rack

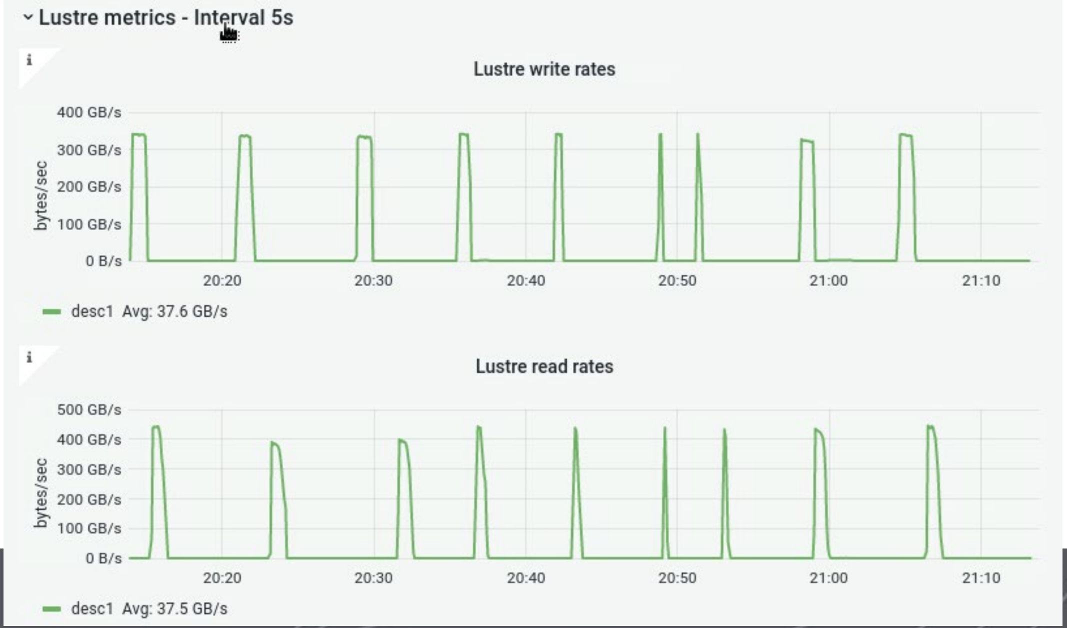


Destor: Derecho - Storage & Scratch File System

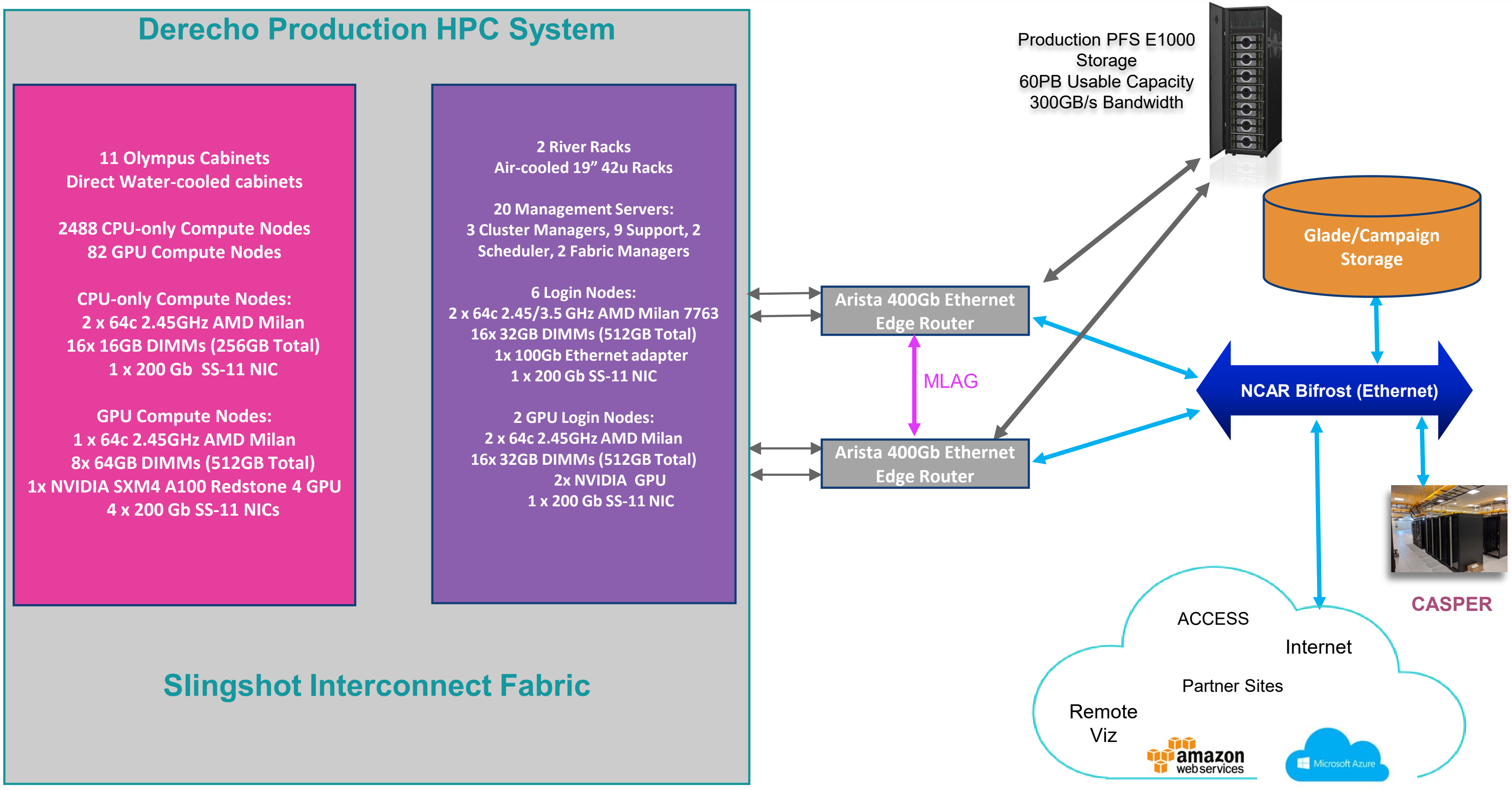
- 6 x HPE/Cray ClusterStor E1000 systems
- 60 petabytes of usable file system space
  - Can be expanded to 120 petabytes
- 300 GB/s aggregate I/O bandwidth
- 5,088 × 16-TB drives
- 40TB SSD for Lustre file system metadata
- Two metadata management units (MDU) with 4 metadata targets (MDTs)
  - One MDT exported per one MDS
  - Configured in highly available storage pairs
- Cray Lustre Parallel File System

Destor Performance - So far





Derecho Network Environment



Derecho Production HPC System

11 Olympus Cabinets
Direct Water-cooled cabinets

2488 CPU-only Compute Nodes
82 GPU Compute Nodes

CPU-only Compute Nodes:
2 x 64c 2.45GHz AMD Milan
16x 16GB DIMMs (256GB Total)
1 x 200 Gb SS-11 NIC

GPU Compute Nodes:
1 x 64c 2.45GHz AMD Milan
8x 64GB DIMMs (512GB Total)
1x NVIDIA SXM4 A100 Redstone 4 GPU
4 x 200 Gb SS-11 NICs

2 River Racks
Air-cooled 19” 42u Racks

20 Management Servers:
3 Cluster Managers, 9 Support, 2 Scheduler, 2 Fabric Managers

6 Login Nodes:
2 x 64c 2.45/3.5 GHz AMD Milan 7763
16x 32GB DIMMs (512GB Total)
1x 100Gb Ethernet adapter
1 x 200 Gb SS-11 NIC

2 GPU Login Nodes:
2 x 64c 2.45GHz AMD Milan
16x 32GB DIMMs (512GB Total)
2x NVIDIA GPU
1 x 200 Gb SS-11 NIC

Slingshot Interconnect Fabric

Production PFS E1000
Storage
60PB Usable Capacity
300GB/s Bandwidth

Glade/Campaign Storage

Arista 400Gb Ethernet Edge Router

MLAG

Arista 400Gb Ethernet Edge Router

NCAR Bifrost (Ethernet)

CASPER

ACCESS

Internet

Partner Sites

Remote
Viz

Compute Node Internet Access

- General compute node internet access is provided through a set of Network Address Translation gateways such that outbound access to internet is possible. However, direct inbound will not be allowed.
- This will allow for access to source control sites (e.g.`github.com`)
- Allows users to fetch **smaller** data sets at runtime

Login Environment / Scheduler Infrastructure

Derecho Login Nodes

- 6x CPU based login nodes that will be served out in a round-robin DNS fashion, each with a 200Gb/s Bifrost connection as an internet routable host.
- 2x GPU login nodes that is served out in the same round robin DNS fashion, each with a 200Gb/s Bifrost connection as an internet routable host and has 2x NVIDIA A100 PCIe-based GPU.
- Control Groups are used to limit the abuse of login nodes and automated emails to users and support teams will be sent out during notifications.

PBS Professional Infrastructure

- 2 systems dedicated to PBS workload management that leverages a IBM Spectrum file system as an underlying High Availability state directory for PBS Professional.
- Filesystem is made up of a RAID 10-like storage array with SSDs to maximize the IOPS of the PBS server.

Job Scheduling & Resource Allocation

Cheyenne Queue Structure

- Semi-Homogenous resources with no resource specific queues
- All queues aside from share were standalone
- Queue name denoted the priority it received when evaluated by the scheduler
- Placement sets for describing node locality requirements

premium
High priority, high cost

regular
Standard submissions

economy
Low priority, low cost

share
CPU-utilization cost

Derecho Queues: Improvements over Cheyenne

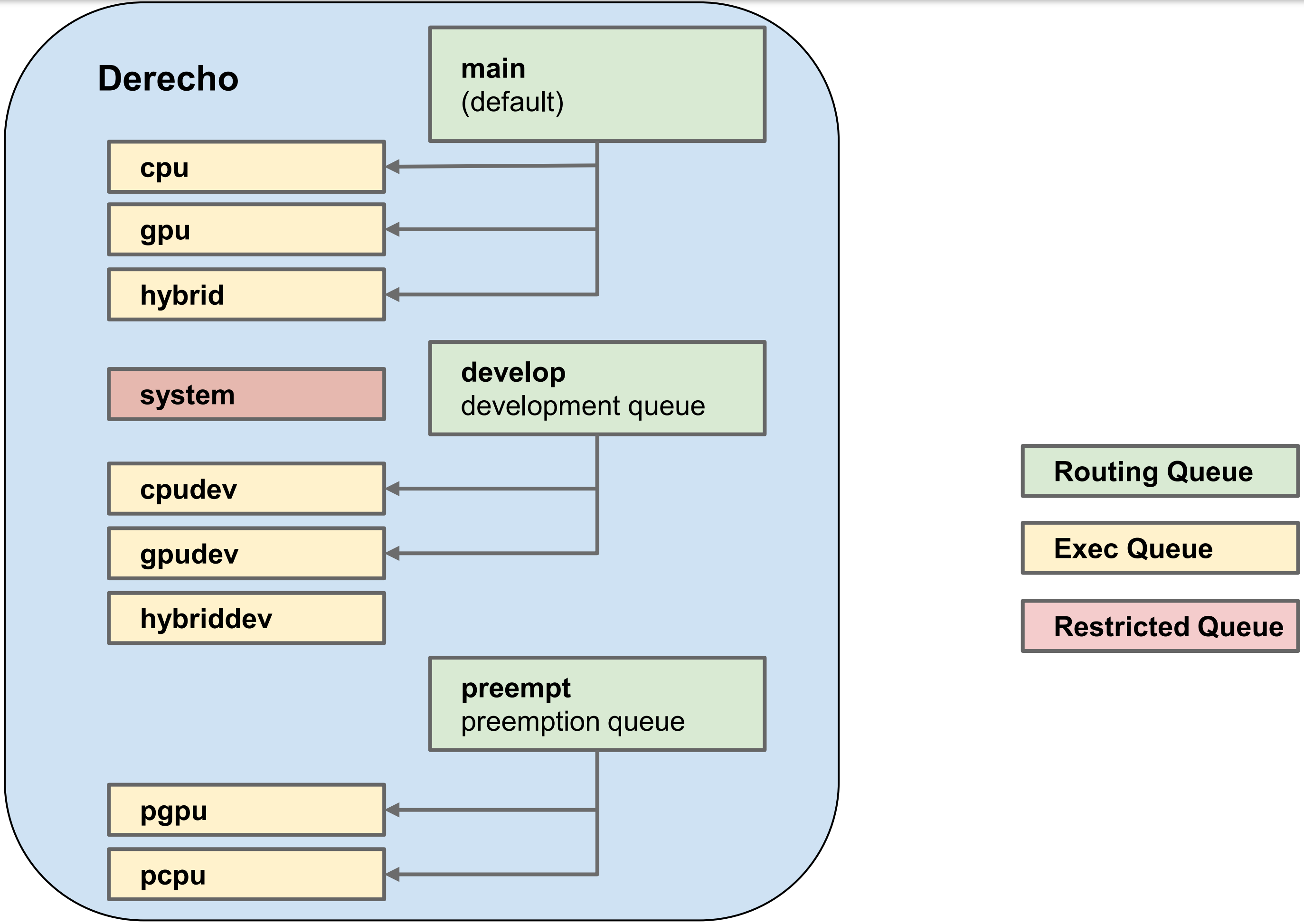
Currently…

- Routing queue based structure
- Queues for specific hardware types (CPU / GPU)
- Job Priority as a requestable resource
- Job sort formula
- cgroups for effectively sharing node resources
- Multi-Process Service (MPS) support for GPU
- Preemption queues

Eventually…

- Lustre scratch job statistics
- Power usage for exclusive resource jobs in job records
- Power profile selection for exclusive resource jobs
- Cloud bursting capability
- GPU utilization and memory statistics in job records

Derecho Queue Structure



Derecho

main
(default)

- cpu
- gpu
- hybrid

system

develop
development queue

- cpudev
- gpudev

hybriddev

preempt
preemption queue

- pgpu
- pcpu

🟩 Routing Queue

🟨 Exec Queue

🟥 Restricted Queue

Derecho Queue Structure

- Two primary submission routing queues
  - `main` for all production workloads
  - `develop` for all development workloads or testing
- CPU and GPU nodes allocated to queues
- Access to production GPU queues will require membership in a project that has been granted GPU core hours
- Preemption queues are also provided for CPU and GPU resources
  - `preempt` routing queue suitable for jobs that can be interrupted
- Job Priority is now a resource flag you can provide for a job
  - Plays heavy part in job sort formula determining order in which jobs are executed

```
qsub   - I    - q main     - A <ACCOUNT> - l select=2:ncpus=64:ngpus=4
- l job_priority=economy           - l gpu_type=a100        - l walltime=00:20:00
```

Job Priority and Job Sort Formula

- With Derecho, in the absence of specific queues for priority, we will be offering the option for users to request a job priority as part of their job request (-l job_priority=...)
- This option is taken into consideration for the job sort formula
  - economy
  - regular (default)
  - premium
- A job sort formula will be used on Derecho, similar to the scheduling environment of Casper
- The formula takes into consideration the following
  - job_priority
  - fairshare factor
  - eligible time (time spent in the queue)
  - requested CPUs
  - requested GPUs

Multi -Process Service (MPS) support for GPU

- With Derecho we will be offering support for NVIDIA’s Multi-Process Service (MPS) on the A100 equipped nodes
- MPS can be requested when requesting a job by requesting a mps resource of 1

```
qsub  - I   - q main    - A <ACCOUNT> - l walltime=00:20:00
      - l select=2:ncpus=64:mpiprocs=4:ngpus=4:          mps=1
```

- MPS server and user daemons created automatically, and then torn down once the job is finished
- MPS has also been deployed on Casper and can be used there with the V100 and A100 equipped nodes

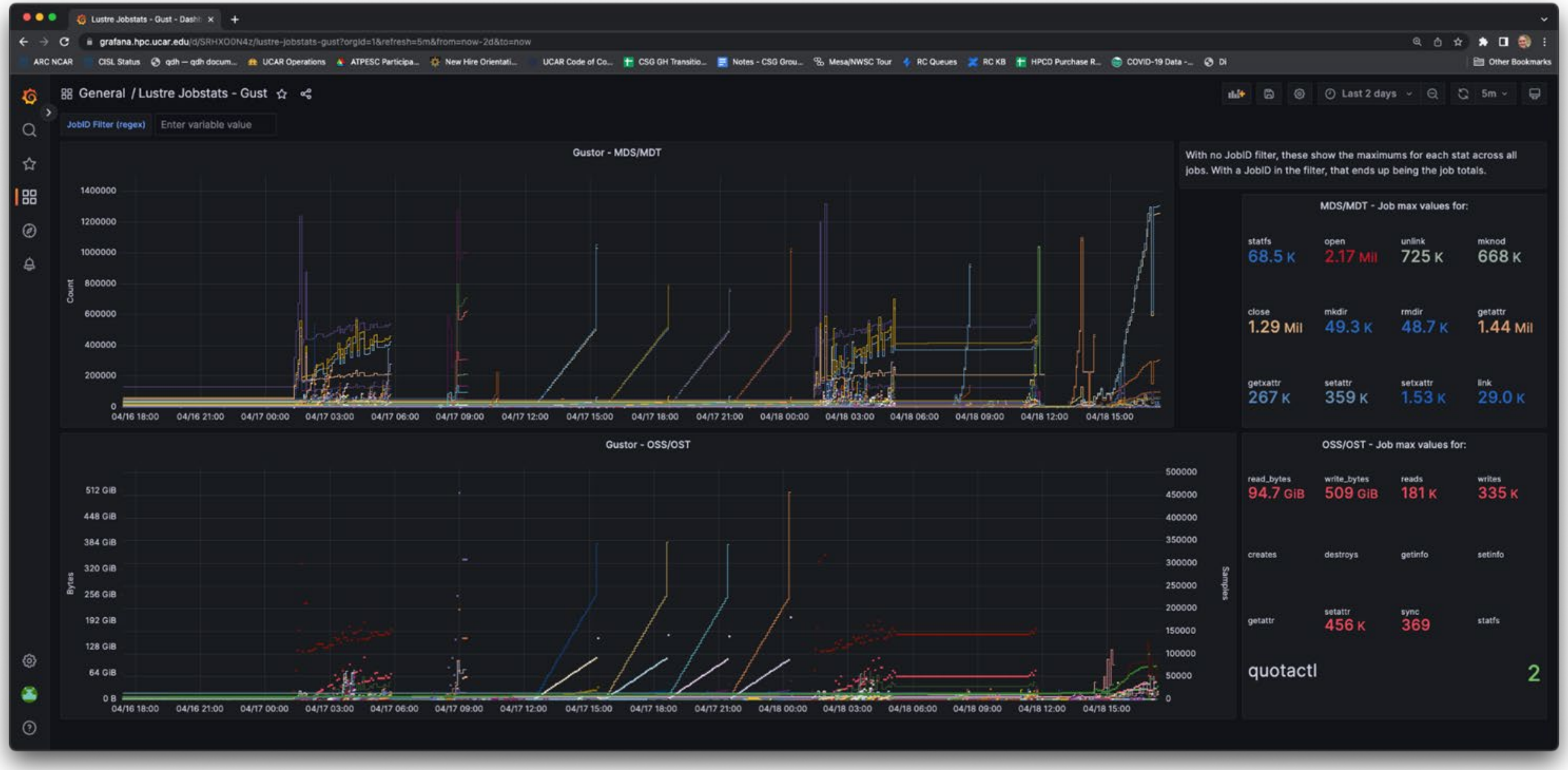
Lustre Job Statistics

- With Derecho we have the ability to poll Lustre job statistics (for the scratch filesystem) and report on certain metrics so that we can have a better understanding of any issues with performance or operation of the scratch file system
- Lustre job statistics are provided for jobs covering several key metrics:

Metadata Operations	Data Operations
• open / close • link / unlink • mkdir / rmdir • getattr / getxattr ; setattr / setxattr • statfs	• reads / writes – counts & bytes • sync

- These metrics are aggregated by`PBS_JOBID` and can be useful for uncovering application performance bottlenecks, including opaque filesystem access patterns such as excessive `${TMPDIR}` use by underlying libraries.

Lustre Job Statistics



General / Lustre Jobstats - Gust

JobID Filter (regex): Enter variable value

With no JobID filter, these show the maximums for each stat across all jobs. With a JobID in the filter, that ends up being the job totals.

MDS/MDT - Job max values for:

statfs	open	unlink	mknod
68.5 K	2.17 Mil	725 K	668 K
close	mkdir	rmdir	getattr
1.29 Mil	49.3 K	48.7 K	1.44 Mil
getxattr	setattr	setxattr	link
267 K	359 K	1.53 K	29.0 K

OSS/OST - Job max values for:

read_bytes	write_bytes	reads	writes
94.7 GiB	509 GiB	181 K	335 K
creates	destroys	getinfo	setinfo
getattr	setattr 456 K	sync 369	statfs

quotactl 2

In Development…

Power Reporting and Power Management

- We are working with HPE and Altair to develop a power hook that can integrate with the Cray EX hardware used for Derecho
- The hook will report power usage for resources assigned to a job requesting exclusive access to nodes
- It will also be able to set power profiles for nodes based on user requested flags for job submission
- We plan to also integrate the ability to power down nodes when not being used for jobs, and power on when they are needed
- Development is currently taking place on Gust

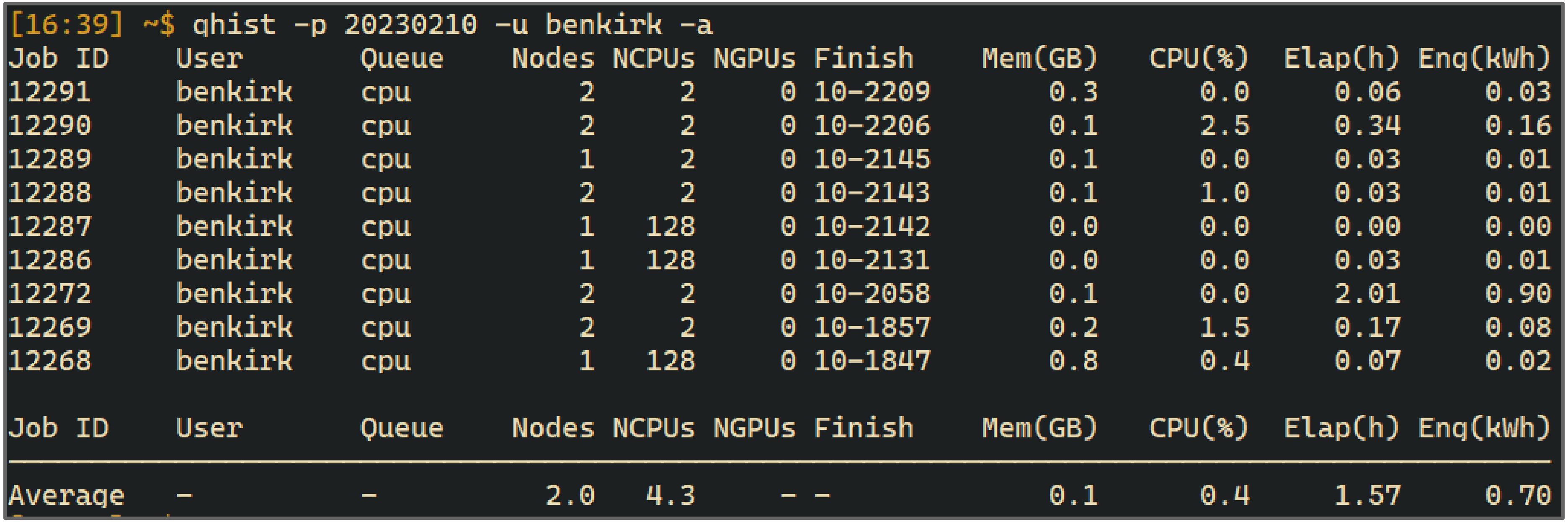
Power Reporting and Power Management

```
# PBS can report the cumulative power (kWh) per job
gust01:~ # qstat -fx 12462
Job Id: 12462.gusched01
    Job_Name = wrf
    Job_Owner = kmanning@gust02.hsn.gu.hpc.ucar.edu
    resources_used.cpupercent = 164
    resources_used.cput = 02:58:08
    resources_used.energy = 0.0169
    resources_used.mem = 28700464kb
    resources_used.ncpus = 128
```

Power Monitoring with *qhist*

qhist - which displays historical job data - will show energy use on Derecho!

- By default, node energy usage will be shown in **kWh**



```
[16:39] ~$ qhist -p 20230210 -u benkirk -a
Job ID    User       Queue     Nodes NCPUs NGPUs Finish     Mem(GB)   CPU(%)  Elap(h) Eng(kWh)
12291     benkirk    cpu           2     2     0 10-2209        0.3      0.0     0.06     0.03
12290     benkirk    cpu           2     2     0 10-2206        0.1      2.5     0.34     0.16
12289     benkirk    cpu           1     2     0 10-2145        0.1      0.0     0.03     0.01
12288     benkirk    cpu           2     2     0 10-2143        0.1      1.0     0.03     0.01
12287     benkirk    cpu           1   128     0 10-2142        0.0      0.0     0.00     0.00
12286     benkirk    cpu           1   128     0 10-2131        0.0      0.0     0.03     0.01
12272     benkirk    cpu           2     2     0 10-2058        0.1      0.0     2.01     0.90
12269     benkirk    cpu           2     2     0 10-1857        0.2      1.5     0.17     0.08
12268     benkirk    cpu           1   128     0 10-1847        0.8      0.4     0.07     0.02

Job ID    User       Queue     Nodes NCPUs NGPUs Finish     Mem(GB)   CPU(%)  Elap(h) Eng(kWh)
----------------------------------------------------------------------------------------------
Average   -          -           2.0   4.3     - -              0.1      0.4     1.57     0.70
```

** Note that averages are weighted by (walltime x nodes)…*

Derecho User Access, Software, User Environment, & Best Practices

Derecho User Access, Software, and User Environment

- Login Environment
- Spack-based software deployment
- Modules
- Filesystems & Storage Spaces

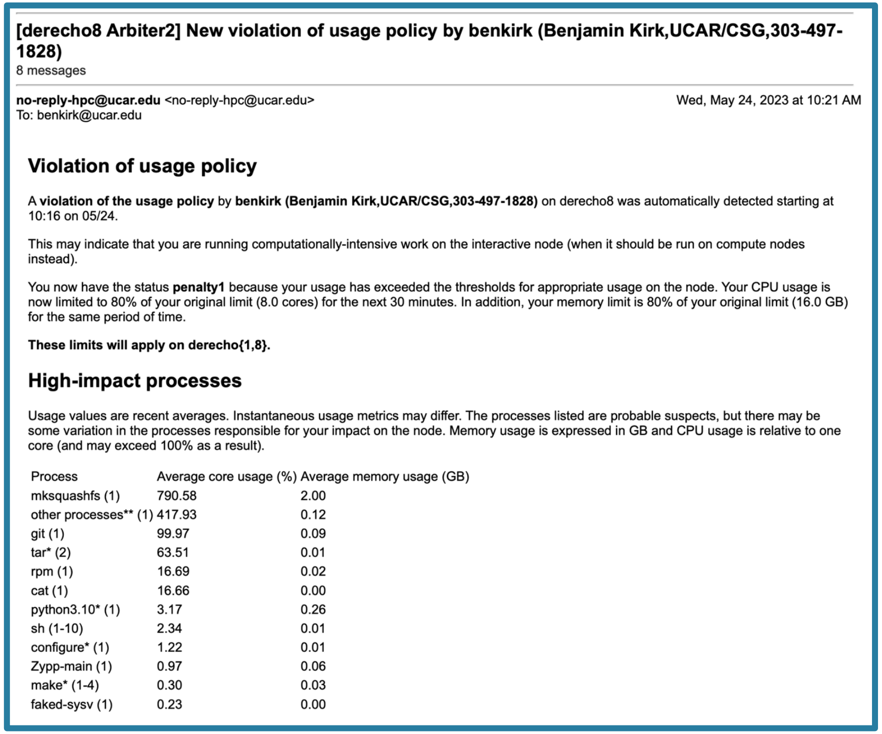
Derecho Login Environment

The system will undergo scheduled maintenance on the first Tuesday of each month for the foreseeable future.

- User login to Derecho is through **`ssh`** at **`derecho.hpc.ucar.edu`**
  - This will place you on 1 of 8 login nodes
- As typical, login node use should be limited to
  - Reading and writing text/code
  - Compiling smaller programs
  - Performing data transfers
  - Interacting with the job scheduler
- User resource utilization is monitored and throttled using the Arbiter2 utility
  - User sessions are placed in a Linux cgroup whose resources can be restricted based on usage policy
  - Intent is to prevent users from monopolizing the login resources while also not being abruptly "kicked off" the login nodes for resource exceedance

Login Environment: User Resource Restrictions

- As typical in a shared HPC environment, login nodes are a shared resource and users must be considerate of their resource utilization
  - Resource intensive workflows should be routed through the queue system
- CISL using the Arbiter2 utility to detect and restrict excessive resource consumption, with automated emails sent to users
  - This involves “squeezing” the CPUs allocated to a given users’ login session to prevent overloading the shared resource
  - If you receive such an email and need help refactoring your workflow, reach out to Consulting



[derecho8 Arbiter2] New violation of usage policy by benkirk (Benjamin Kirk,UCAR/CSG,303-497-1828)

8 messages

no-reply-hpc@ucar.edu <no-reply-hpc@ucar.edu> Wed, May 24, 2023 at 10:21 AM

To: benkirk@ucar.edu

Violation of usage policy

A **violation of the usage policy** by **benkirk (Benjamin Kirk,UCAR/CSG,303-497-1828)** on derecho8 was automatically detected starting at 10:16 on 05/24.

This may indicate that you are running computationally-intensive work on the interactive node (when it should be run on compute nodes instead).

You now have the status **penalty1** because your usage has exceeded the thresholds for appropriate usage on the node. Your CPU usage is now limited to 80% of your original limit (8.0 cores) for the next 30 minutes. In addition, your memory limit is 80% of your original limit (16.0 GB) for the same period of time.

These limits will apply on derecho{1,8}.

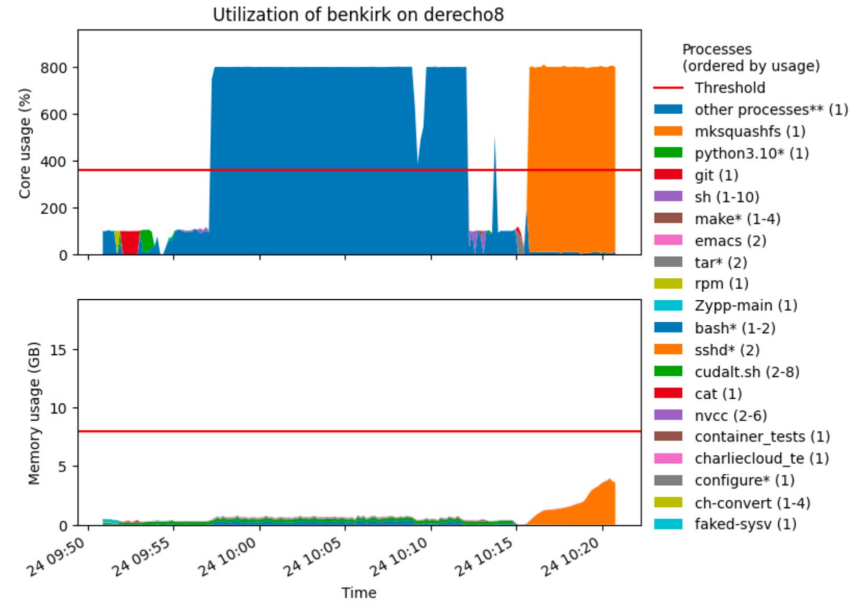
High-impact processes

Usage values are recent averages. Instantaneous usage metrics may differ. The processes listed are probable suspects, but there may be some variation in the processes responsible for your impact on the node. Memory usage is expressed in GB and CPU usage is relative to one core (and may exceed 100% as a result).

Process	Average core usage (%)	Average memory usage (GB)
mksquashfs (1)	790.58	2.00
other processes** (1)	417.93	0.12
git (1)	99.97	0.09
tar* (2)	63.51	0.01
rpm (1)	16.69	0.02
cat (1)	16.66	0.00
python3.10* (1)	3.17	0.26
sh (1-10)	2.34	0.01
configure* (1)	1.22	0.01
Zypp-main (1)	0.97	0.06
make* (1-4)	0.30	0.03
faked-sysv (1)	0.23	0.00

Login Environment: User Resource Restrictions

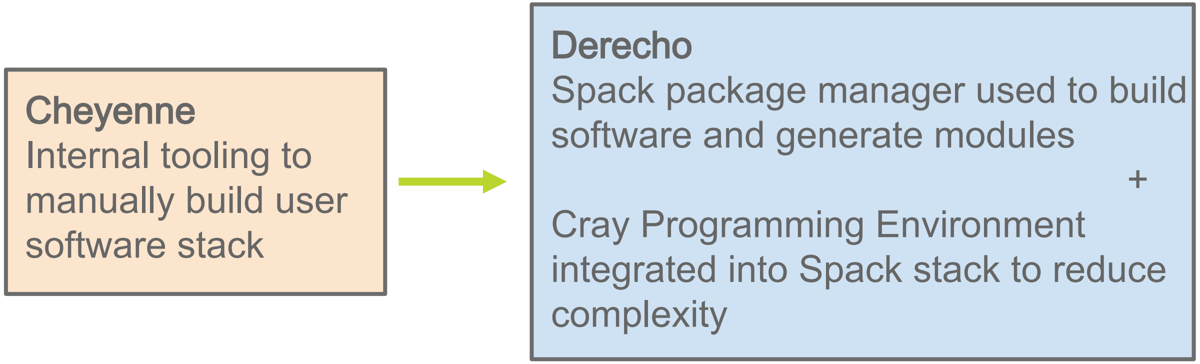
- As typical in a shared HPC environment, login nodes are a shared resource and users must be considerate of their resource utilization
  - Resource intensive workflows should be routed through the queue system
- CISL using the Arbiter2 utility to detect and restrict excessive resource consumption, with automated emails sent to users
  - This involves “squeezing” the CPUs allocated to a given users’ login session to prevent overloading the shared resource
  - If you receive such an email and need help refactoring your workflow, reach out to Consulting



**This process is generally permitted on interactive nodes and is only counted against you when considering memory usage (regardless of the process, too much memory usage is still considered bad; it cannot be throttled like CPU). The process is included in this report to show usage holistically.*

A nimble software stack provided by Spack, Cray, and Lmod

Derecho features a familiar collection of environment modules (provided via Lmod) with some notable differences from Cheyenne:



Cheyenne
Internal tooling to manually build user software stack

Derecho
Spack package manager used to build software and generate modules

+

Cray Programming Environment integrated into Spack stack to reduce complexity

Casper will use the same operating system as **Derecho**, which should enable compatible Spack stacks and binaries where possible (*Cray tools will not be on Casper*)

Spack

Spack provides 1000s of software recipes provided by a large community

Our Spack stack is:

- Easier to update
- Publicly visible via GitHub repo
- Extensible by users via Spack *upstream* support

Cray functionality is available with simplified access

If you have used Cray modules on other systems, you may expect certain modules which are missing on Derecho…

We have simplified the module structure but all Cray functionality should be available. These two lists of modules are equivalent:

Traditional Cray environment

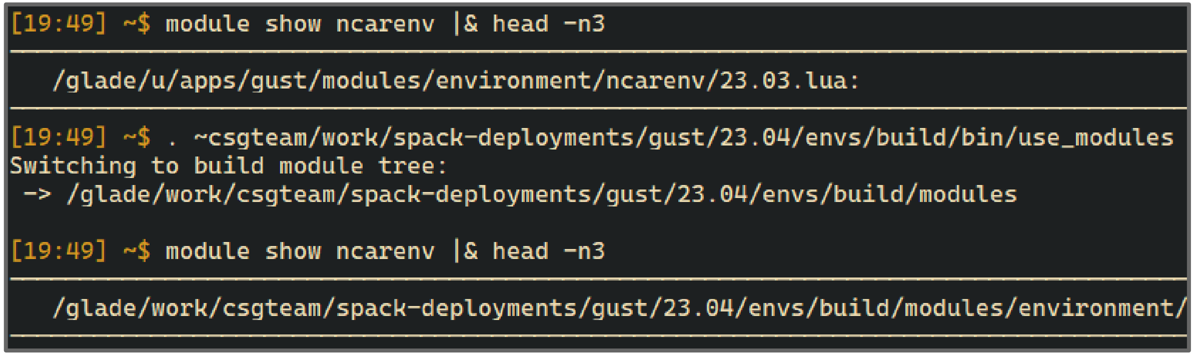
```
  1) crayenv/23.03   (S)   3) cce/15.0.1    5) cray-libsci/23.02.1.1  7) craype/2.7.20
  9) craype-x86-milan 11) craypenetwork-ofi  2) cray-pmi/6.1.10  4) PrgEnv-cray/8.3.3
  6) cray-dsmml/0.2.2  8) cray-pals/1.2.11  10) libfabric/1.15.0.0  12) cray-mpich/8.1.25
```

Simplified Cray environment on Derecho

```
  1) ncarenv/23.04  (S)    2) craype/2.7.20   3) cce/15.0.1   4) cray-mpich/8.1.25
  5) cray-libsci/23.02.1.1
```

Prototyping and Version Control Allows for User Testing and Co -design

Easy switching to testing module tree

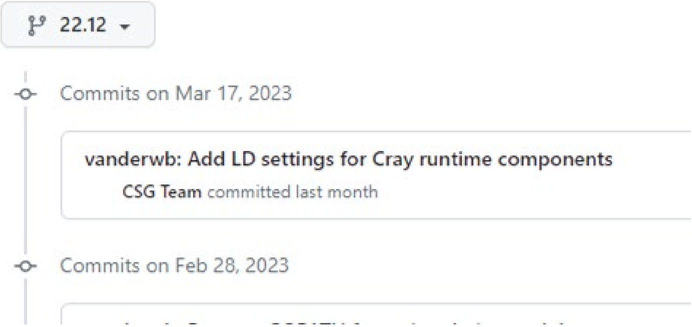


```
[19:49] ~$ module show ncarenv |& head -n3
-----------------------------------------------------------------------------------
   /glade/u/apps/gust/modules/environment/ncarenv/23.03.lua:
-----------------------------------------------------------------------------------
[19:49] ~$ . ~csgteam/work/spack-deployments/gust/23.04/envs/build/bin/use_modules
Switching to build module tree:
 -> /glade/work/csgteam/spack-deployments/gust/23.04/envs/build/modules

[19:49] ~$ module show ncarenv |& head -n3
-----------------------------------------------------------------------------------
   /glade/work/csgteam/spack-deployments/gust/23.04/envs/build/modules/environment/
-----------------------------------------------------------------------------------
```

use_modules allows us to involve users in testing modules before we make them public

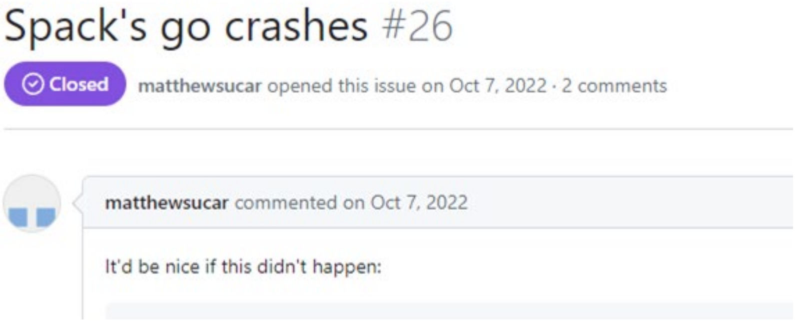
Software changes are publicly tracked and debugging is collaborative



Commits on Mar 17, 2023

vanderwb: Add LD settings for Cray runtime components
CSG Team committed last month

Commits on Feb 28, 2023



Spack's go crashes #26

matthewsucar opened this issue on Oct 7, 2022 · 2 comments

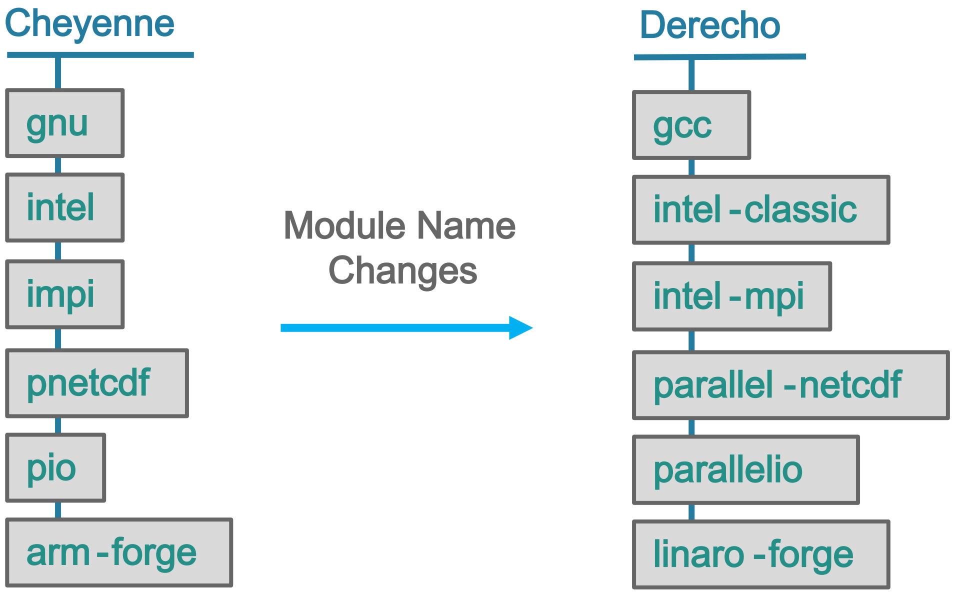
matthewsucar commented on Oct 7, 2022

It'd be nice if this didn't happen:

User Software Environment: Known Issues and Limitations

- MPI
  - Cray-MPICH is the only reliable MPI implementation currently available on Slingshot 11
    - Cray-MPICH is CUDA-Aware, and works with CUDA Managed Memory, but is not particularly performant in this case (automated host-buffer copies occur behind the scenes).
      - Optimal CUDA/MPI performance should avoid sending managed memory buffers, *if possible*
    - We strive to ensure **`MPICH_GPU_SUPPORT_ENABLED`** and **`MPICH_GPU_MANAGED_MEMORY_SUPPORT_ENABLED`** are automatically set on GPU nodes. ***DO NOT UNSET THESE ENVIRONMENT VARIABLES AS NODES WILL HARD -CRASH!!***
  - MVAPICH2 support for Slingshot 11 is in beta and will be fully evaluated once officially released,
  - Intel MPI is currently installed for CPU-only nodes,
  - OpenMPI is currently not available on Derecho.
    - We are pursuing OpenMPI with high priority, but currently have no target date for availability
- Compiler Issues
  - Several, tracked separately
- Cray's LibSci, Intel's MKL, and other numerical libraries
  - LibSci works best with Cray-provided compilers (CCE and GCC). We have seen problems when mixing libraries across toolchains
  - MKL is available and works with most compilers to provide a performant BLAS/LAPACK
  - NVHPC provides BLAS and LAPACK libraries - we recommend you use those with its compiler

Cheyenne → Derecho Modules Cheatsheet



Cheyenne

gnu

intel

impi

pnetcdf

pio

arm-forge

Module Name Changes

Derecho

gcc

intel-classic

intel-mpi

parallel-netcdf

parallelio

linaro-forge

While we aim to provide a familiar experience, some notable differences from Cheyenne modules do exist:

- **ncarenv** is now the top-level module, providing access to all other software (*versions indicate entirely different software stacks*)
- Some former modules (e.g., git) are now in your environment by default
- Some packages will autoload dependencies (e.g., **netcdf** will autoload **hdf5**)

GLADE File Spaces

Users can access several distinct “file spaces” under NCAR’s *GL*obally *A*ccessible *D*ata *E*nvironment (*GLADE*):

File Space	Quota	Backups	Technology	Uses
Home `/glade/u/home/${USER}`	50 GB	Yes	IBM Spectrum Scale	Users’ settings, source code, scripts
Work `/glade/work/${USER}`	1 TB*	No	IBM Spectrum Scale	Compiled codes, models
Scratch (Derecho) `/glade/derecho/scratch/${USER}`	30 TB	NO!! *Purged!*	Cray ClusterStor Lustre	Run Directories, Temporary outputs **Purged at 180 days**
Scratch (Cheyenne) `/glade/cheyenne/scratch/${USER}`	10 TB	NO!! *Purged!*	IBM Spectrum Scale	Run Directories, Temporary outputs Purged at 120 days
Campaign Storage `/glade/campaign/<PROJECTS>`	*Allocated, Project Specific*	No	IBM Spectrum Scale	Project Spaces, Long-term Curated Data Sets. Automated file compression.

GLADE will undergo a significant transformation throughout the calendar year to accommodate Cheyenne/Derecho overlap and eventual Cheyenne retirement.

GLADE File Spaces

GLADE filespaces visible from Derecho compute nodes:

```
/glade/u/home/${USER}                # ${HOME} common across systems
/glade/u/apps/
/glade/work/${USER}                  # ${WORK} common across systems
/glade/cheyenne/scratch/${USER}  # ${CHEYENNE_SCRATCH}
/glade/derecho/scratch/${USER}   # ${SCRATCH} & ${DERECHO_SCRATCH}
/glade/campaign/
```

We provide several environment variables through the **ncarenv** module to facilitate data access across systems.

Key points:

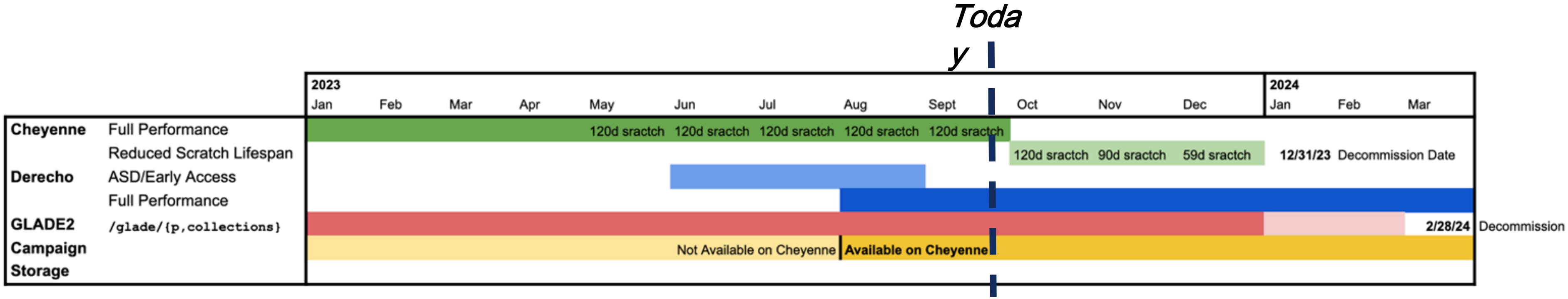
- Users' *Cheyenne* scratch data are accessible on *Derecho* and can be read directly during system overlap
- If you need assistance **moving** large quantities of data, reach out to Consulting or consider sample migration script in backup .

GLADE / Derecho / Cheyenne Overlap & Decommission Plan

GLADE will undergo a significant transformation throughout the calendar year to accommodate Cheyenne/Derecho overlap and eventual Cheyenne retirement.

Key Points :

- *Cheyenne* will run its last job on 12/31/23, with its scratch file system available only for an additional ~60 days (through 2/28/2024),
- `/glade/p` and `/glade/collections` live on this same hardware that will be retired 2/28/2024,
- `/glade/campaign` has been mounted more broadly, can take the place of `p` and `collections`



**GLADE2 is the storage hardware associated with the lifetime of Cheyenne - if you hear us speak of GLADE2 it really means a particular hardware subcomponent of the logical concept that is GLADE*

Workflow Migration

Overview

- Compiling Software
- Job Monitoring and Execution
- Job Submission
  - MPI & hybrid MPI/Threads jobs
  - GPU jobs
  - NUMA Domains & Binding
- Scratch Lustre File System and MPI-IO

Compiling Code on Derecho

Compiling Code on Derecho

Derecho users have access to :

- *Intel (Classic/OneAPI)*
- *Cray Compiling Environment (CCE)*
- *GNU Compiler Collection (GCC)*
- *NVIDIA HPC Software Development Kit (SDK).*

Wrapper scripts are loaded by default (**`ncarcompilers`** module) to streamline the compiling and linking process by adding include header and library path flags for you. Unlike on Cheyenne, the wrapper will not explicitly add library references (**`-lnetcdf`** for example).

Example :

- Building with netCDF using wrappers:

```
ifort model.f90 -lnetcdff -o model
```

- Building with netCDF without the wrappers:

```
setenv NETCDF /path/to/netcdf
ifort -I${NETCDF}/include model.f90 -L${NETCDF}/lib -lnetcdff -o model
```

Compilers Available on Derecho

Compiler	Language	Commands for serial programs	Commands for MPI programs (with ncarcompilers)	Flags to enable OpenMP (for serial and MPI)
Intel (Classic/OneAPI)	Fortran	`ifort/ifx foo.f90`	`mpif90 foo.f90`	`-qopenmp`
	C	`icc/icx foo.c`	`mpicc foo.c`	
	C++	`icpc/icpx foo.C`	`mpicxx foo.C`	
Cray Compiler (CCE)	Fortran	`ftn foo.f90`	`mpif90 foo.f90`	`-fopenmp`
	C	`cc foo.c`	`mpicc foo.c`	
	C++	`CC foo.C`	`mpicxx foo.C`	
GNU (GCC)	Fortran	`gfortran foo.f90`	`mpif90 foo.f90`	`-fopenmp`
	C	`gcc foo.c`	`mpicc foo.c`	
	C++	`g++ foo.C`	`mpicxx foo.C`	
NVIDIA HPC SDK	Fortran	`nvfortran foo.f90`	`mpif90 foo.f90`	`-mp`
	C	`nvc foo.c`	`mpicc foo.c`	
	C++	`nvc++ foo.C`	`mpicxx foo.C`	

Intel Compilers on Derecho

- Similar to the previous NCAR systems, the Intel compiler suite is available via the **`intel`** compiler module. It includes compilers for C, C++, and Fortran codes.

Compiler	Language	**Commands for serial programs**	**Commands for MPI programs (with ncarcompilers)**	**Flags to enable OpenMP (for serial and MPI)**
Intel (Classic/OneAPI)	Fortran	`ifort/ifx foo.f90`	`mpif90 foo.f90`	`-qopenmp`
	C	`icc/icx foo.c`	`mpicc foo.c`	
	C++	`icpc/icpx foo.C`	`mpicxx foo.C`	

- Derecho supports both Intel OneAPI and Intel Classic Compilers. Intel is planning to retire the Intel Classic compilers and is moving toward Intel OneAPI. Intel Classic Compiler commands (**`ifort`**, **`icc`**, and **`icpc`**) will be replaced by the Intel OneAPI compilers (**`ifx`**, **`icx`**, and **`icpx`**) in future. Permutations available through the modules:
  - `intel*`
  - `intel-classic`
  - `intel-oneapi`

Intel Compilers on Derecho

- Intel compilers provide several different optimization and vectorization options. Please refer to compiler manual page to explore available optimization options. (e.g. **`man ifort`**) or use help menu (**`ifort --help`**).

Be aware that compiling CPU code with the Intel compiler on Derecho is subtly different from using the Intel compiler on the Cheyenne system due to different architecture:

Flags that are commonly used on Cheyenne might cause Derecho jobs to fail or run much more slowly than otherwise possible.

- On Derecho, ***Do Use***: **`-march=core-avx2`**
- On Derecho, **Do NOT** Use: **`-xHost, -axHost, -xCORE-AVX2, -axCORE-AVX2`**

Cray Compiling Environment (CCE)

- Derecho users can access the Cray compilers using the**cce** module.
- The Cray compiler collection provides Cray Fortran and Cray C/C++ compilers using **cc/CC** and **ftn** commands.

Compiler	Language	Commands for serial programs	Commands for MPI programs (with ncarcompilers)	Flags to enable OpenMP (for serial and MPI)
Cray Compiler (CCE)	Fortran	`ftn foo.f90`	`mpif90 foo.f90`	`-fopenmp`
	C	`cc foo.c`	`mpicc foo.c`	
	C++	`CC foo.C`	`mpicxx foo.C`	

- Unlike other MPI libraries, **Cray MPICH** does not provide MPI wrapper commands like **mpicc**, **mpicxx**, and **mpif90**. Rather, use the same **cc**, **CC**, and **ftn** commands you use to compile a serial code.
- But the **ncarcompilers** module will translate a call to "**mpicc**" to "**cc**" (and likewise for the other languages) as a convenience.
- Cray compilers enables offloading of computation from CPUs to GPUs via OpenMP and OpenACC.

Compiling GPU codes on Derecho

- GPU applications should be built with eithe**the Cray compilers** or the **NVIDIA HPC SDK** compilers and libraries.

NVIDIA HPC SDK (Software Development Kit)

NVIDIA HPC SDK is a comprehensive suite of tools, compilers, and libraries designed to help developers build and optimize HPC applications for NVIDIA GPUs, as well as multicore CPUs. It includes:

  - NVIDIA Compilers including nvfortran, nvc, and nvc++.
  - CUDA / OpenACC / OpenMP support for GPU nodes
  - NSight & more for performance analysis

- Compilation flags for GPU code will depend in large part on the GPU-programming paradigm used (e.g., OpenACC, OpenMP, CUDA).

- Read the relevant man page for the chosen compiler for customizations and optimizations options.

Compiling GPU codes on Derecho

OpenACC

- Compile with OpenACC directives using nvc, nvc++, or nvfortran and adding **-acc** flag:

  ```
  nvfortran -o acc_bin -acc acc_code.f90
  ```

- Gain insights into GPU acceleration decisions with the **-Minfo=accel** flag.
- Target specific GPU architectures (V100 or A100) with the -gpu=cc70,cc80 flag:

  ```
  nvfortran -o acc_bin -acc -gpu=cc70,cc80 acc_code.f90
  ```

OpenMP

- compile with GPU offloading using the **-mp=gpu** flag:

  ```
  nvfortran -o omp_gpu -mp=gpu omp.f90
  ```

- Diagnostic and target flags from OpenACC examples also apply to OpenMP offloading.

Compiling GPU codes on Derecho

CUDA

- Fortran example:
  - Use **nvfortran** as it supports CUDA directly.
  - Enable CUDA automatically with .cuf file extension or use the **-Mcuda** flag:

    ```
    nvfortran -Mcuda -o cf_bin cf_code.f90
    ```

- C++ example (two-stage process using **nvcc** and **g++**):
  - Use **nvcc**, the nvidia CUDA compiler, to compile the CUDA code:

    ```
    nvcc -c -arch=sm_80 cuda_code.cu
    ```

  - Load the appropriate cuda environment module with a non-NVIDIA C++ compiler.
  - Link CUDA objects with C++ main:

    ```
    g++ -o cuda_bin -lcuda -lcudart main.cpp cuda_code.o
    ```

Cray Programming Environment (CrayPE module)

- **`CrayPE`** module loaded by default contains drivers, **`cc`**, **`CC`**, and **`ftn`** to compile for the CCE, GNU, NVHPC, and Intel Programming Environments.
- **`CrayPE`** module is needed for building MPI applications with Cray MPICH MPI.

```
# Intel Compilers (default)
module reset
ftn model.f90 -o model # Fortran
cc model.c -o model     # C
CC model.C -o model     # C++
```

```
# Cray Compilers
module swap intel cce/15.0.1
ftn model.f90 -o model # Fortran
cc model.c -o model     # C
CC model.C -o model     # C++
```

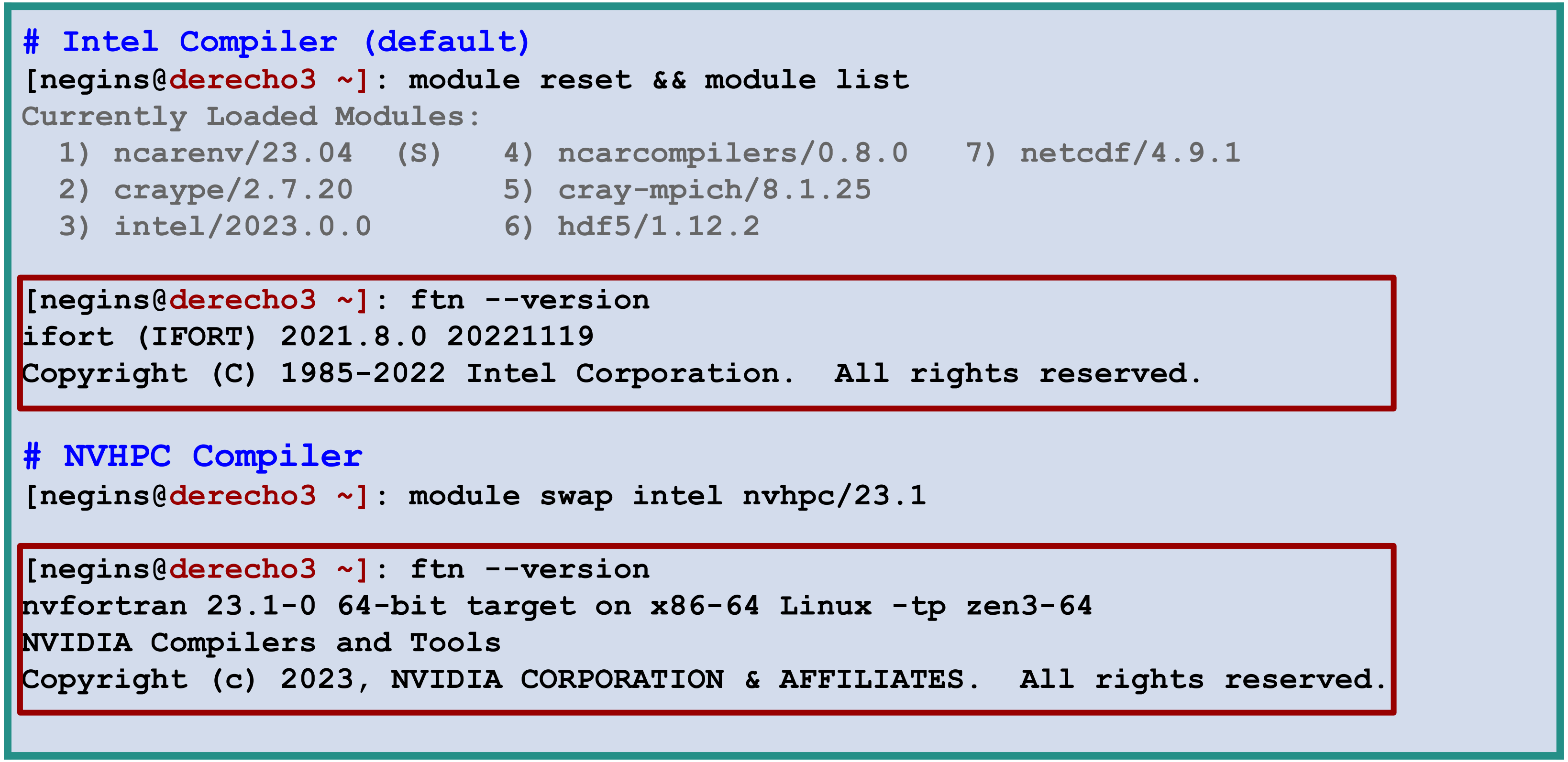
```
# GNU Compiler
module swap intel gcc/12.2.0
ftn model.f90 -o model # Fortran
cc model.c -o model     # C
CC model.C -o model     # C++
```

```
# NVHPC compiler
module swap intel nvhpc/23.1
ftn model.f90 -o model # Fortran
cc model.c -o model     # C
CC model.C -o model     # C++
```

Please note that the compiler wrappers - **`cc/CC`** and **`ftn`** - are not Cray compilers themselves. Instead, they call Intel, GNU, or Cray compilers based on the programming environment module that is loaded.

Cray Programming Environment (CrayPE module)

⚠ Please note that the compiler wrappers - ftn, cc, and CC - are not Cray compilers themselves. Instead, they call Intel, GNU, or Cray compilers based on the programming environment module that is loaded. Users can use -V or --version to see which base compiler the wrapper is pointing to.



```
# Intel Compiler (default)
[negins@derecho3 ~]: module reset && module list
Currently Loaded Modules:
  1) ncarenv/23.04   (S)    4) ncarcompilers/0.8.0    7) netcdf/4.9.1
  2) craype/2.7.20          5) cray-mpich/8.1.25
  3) intel/2023.0.0         6) hdf5/1.12.2

[negins@derecho3 ~]: ftn --version
ifort (IFORT) 2021.8.0 20221119
Copyright (C) 1985-2022 Intel Corporation.  All rights reserved.

# NVHPC Compiler
[negins@derecho3 ~]: module swap intel nvhpc/23.1

[negins@derecho3 ~]: ftn --version
nvfortran 23.1-0 64-bit target on x86-64 Linux -tp zen3-64
NVIDIA Compilers and Tools
Copyright (c) 2023, NVIDIA CORPORATION & AFFILIATES.  All rights reserved.
```

Clang 16+ compilers and older C code

Recent C compilers based on Clang- including **intel/2023** and **cce/16 (Cray)** have turned C-standard compliance checks from *warnings* to *errors*.

For example, if you compile non-standard C code, you may see errors like this:

```
error: ...; ISO C99 and later do not support implicit
function declarations [-Wimplicit-function-declaration]
```

The best solution to these errors is to update the code to be compliant to the specified standard (contact us if you need help)!

However, a corresponding fix may force these errors back to warnings. For example:

```
icx -Wno-error=implicit-function-declaration ...
```

Cross-Compiler & Cross-System Development Strategies

With the new Spack user software environment we have introduced several environment variables developers may find useful when switching between compilers, MPIs and even systems:

```
benkirk@derecho1(26)$ module list && env | grep NCAR_BUILD

Currently Loaded Modules:
  1) ncarenv/23.04 (S)    4) ncarcompilers/0.8.0   7) cray-mpich/8.1.25
  2) craype/2.7.20        5) hdf5/1.12.2
  3) gcc/12.2.0           6) netcdf/4.9.1


...

NCAR_BUILD_ENV_COMPILER=derecho-gcc-12.2.0
NCAR_BUILD_ENV_MPI=derecho-gcc-12.2.0-cray-mpich-8.1.25
NCAR_BUILD_ENV=derecho-gcc-12.2.0-cray-mpich-8.1.25
```

*Will be deployed on the new **Casper OS** too*

Cross -Compiler & Cross -System Development Strategies

With the new Spack user software environment we have introduced several environment variables developers may find useful when switching between compilers, MPIs and even systems:

```
benkirk@derecho1(27)$ module load nvhpc && \
                      module list && env | grep NCAR_BUILD
Currently Loaded Modules:
  1) ncarenv/23.04 (S)   4) ncarcompilers/0.8.0   7) cray-mpich/8.1.25
  2) craype/2.7.20       5) hdf5/1.12.2
  3) nvhpc/23.1          6) netcdf/4.9.1

...

NCAR_BUILD_ENV_COMPILER=derecho-nvhpc-23.1
NCAR_BUILD_ENV_MPI=derecho-nvhpc-23.1-cray-mpich-8.1.25
NCAR_BUILD_ENV=derecho-nvhpc-23.1-cray-mpich-8.1.25
```

*Will be deployed on the new **Casper OS** too*

Running Jobs on Derecho

Job Monitoring and Execution

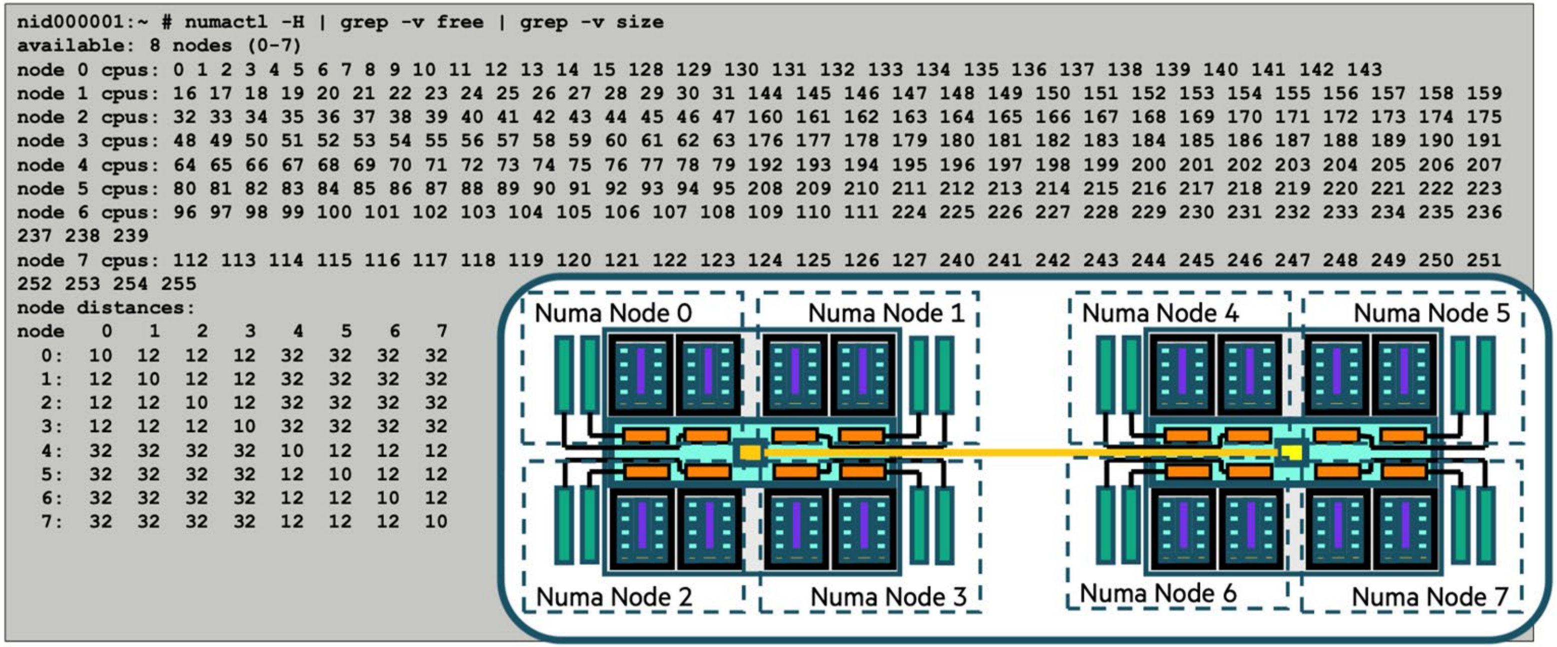
- Derecho job monitoring and execution is nearly identical to the user experience on Cheyenne and Casper
- PBS commands
  - **`qsub`**
  - **`qdel`**
  - **`qstat`**
  - **`qhist`** (NCAR-specific query tool)
- Complete documentation is available online, what follows are Derecho-specific details important for users familiar with PBS batch submission in general

Running Jobs on Derecho

- Use the PBS **qsub** command to submit batch jobs to the "**main**" queue
- Resource requests will determine if the job is routed to the cpu or gpu queues for execution
- Binding MPI ranks and OpenMP threads is important for performance
- GPU to MPI rank assignment can be controlled via the **CUDA_VISIBLE_DEVICES** environment variable, or programmatically in source code
- Placement of MPI ranks for GPU code can affect performance due to multiple NICs on the GPU nodes

Running Jobs on Derecho: NUMA Domains

- Nonuniform Memory Access (NUMA) has important performance implications, especially for Derecho’s multi-socket **CPU** compute nodes
- Derecho’s **GPU** Nodes have only a single socket, but GPU↔ Network Interface Card mapping is also important



```
nid000001:~ # numactl -H | grep -v free | grep -v size
available: 8 nodes (0-7)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
node 1 cpus: 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159
node 2 cpus: 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175
node 3 cpus: 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191
node 4 cpus: 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207
node 5 cpus: 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223
node 6 cpus: 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 224 225 226 227 228 229 230 231 232 233 234 235 236
237 238 239
node 7 cpus: 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 240 241 242 243 244 245 246 247 248 249 250 251
252 253 254 255
node distances:
node   0   1   2   3   4   5   6   7
  0:  10  12  12  12  32  32  32  32
  1:  12  10  12  12  32  32  32  32
  2:  12  12  10  12  32  32  32  32
  3:  12  12  12  10  32  32  32  32
  4:  32  32  32  32  10  12  12  12
  5:  32  32  32  32  12  10  12  12
  6:  32  32  32  32  12  12  10  12
  7:  32  32  32  32  12  12  12  10
```

Running Jobs on Derecho

Binding MPI and Hybrid MPI + OpenMP Jobs

- Binding is important for performance of MPI and hybrid jobs

The mpiexec binding options do not work well when oversubscribing hardware resources. We recommend a maximum of one rank or thread per physical CPU core.

- MPI Example: 2-nodes, 128 ranks/node

```
mpiexec -n 256 -ppn 128 ./executable_name
```

by default this will bind to a "thread"

various other ways to bind: "core", "numa", "list"

```
mpiexec -n 256 -ppn 128 --cpu-bind core ./executable_name
```

- Hybrid Example: 2-nodes, 32 ranks/node, 4 threads per rank

this should be bound as:

```
mpiexec --cpu-bind depth -n 64 -ppn 32 -d 4 ./executable_name
```

Binding of threads can be additionally controlled via OpenMP env vars

```
export OMP_PLACES=threads
export OMP_PROC_BIND=close
```

```
mpiexec --cpu-bind depth -n 64 -ppn 32 -d 4 ./executable_name
```

Running Jobs on Derecho

Pure MPI Job example

```
#!/bin/bash
#PBS -A project_code
#PBS -N mpi_job
#PBS -q main
#PBS -l walltime=01:00:00
#PBS -l select=2:ncpus=128:mpiprocs=128

# load necessary module environment
module purge
module load ncarenv craype cce cray-mpich

# Run application using cray-mpich MPI
mpiexec -n 256 -ppn 128 ./executable_name
```

Running Jobs on Derecho: CPU Binding

Hybrid MPI + OpenMP Job example

```
#!/bin/bash
#PBS -A project_code
#PBS -N hybrid_job
#PBS -q main
#PBS -l walltime=01:00:00
#PBS -l select=2:ncpus=128:mpiprocs=32:ompthreads=4

# load necessary module environment
module purge
module load ncarenv craype cce cray-mpich

# Run application using cray-mpich MPI
export OMP_PLACES=threads
export OMP_PROC_BIND=close
mpiexec --cpu-bind depth -n 64 -ppn 32 -d 4 ./executable_name
```

Running Jobs on Derecho

Mapping MPI Ranks to GPUs and NICs

- Setting **`MPICH_OFI_NIC_POLICY=GPU`** will assign MPI ranks to the NIC closest to its associated GPU
- The mapping between MPI ranks and GPU device IDs can be done programmatically, or can leverage the **`CUDA_VISIBLE_DEVICES`** environment variable.
- As a convenience a script, ***`set_gpu_rank`***, is provided as part of the ncarenv module that will set **`CUDA_VISIBLE_DEVICES`** individually for each MPI rank, to ensure balanced use of the 4 available GPUs on a node.
- The ***`set_gpu_rank`*** script is called on the mpiexec command line before the executable to be launched

```
export MPICH_OFI_NIC_POLICY=GPU
mpiexec -n 16 -ppn 4 set_gpu_rank ./executable
```

Running Jobs on Derecho

MPI enabled GPU example (NVHPC and Cray MPICH)

```
#!/bin/bash
#PBS -A project_code
#PBS -N gpu_job
#PBS -q main
#PBS -l walltime=01:00:00
#PBS -l select=2:ncpus=64:mpiprocs=4:ngpus=4

# load necessary module environment
module purge
module load ncarenv nvhpc cuda cray-mpich

# Run application with efficient mapping to GPUs and NICs
export MPICH_OFI_NIC_POLICY=GPU
mpiexec -n 8 -ppn 4 set_gpu_rank ./executable_name
```

Running Jobs on Derecho

MPI enabled GPU example with MPS (NVHPC and Cray MPICH)

```
#!/bin/bash
#PBS -A project_code
#PBS -N gpu_job
#PBS -q main
#PBS -l walltime=01:00:00
#PBS -l select=2:ncpus=64:mpiprocs=16:ngpus=4:mps=1

# load necessary module environment
module purge
module load ncarenv nvhpc cuda cray-mpich

# Run application with efficient mapping to GPUs and NICs
export MPICH_OFI_NIC_POLICY=GPU
mpiexec -n 32 -ppn 16 set_gpu_rank ./executable_name
```

Running Jobs on Derecho

Single Node / Multi-GPU PyTorch Example

```
#!/bin/bash -l
#PBS -l select=1:ncpus=64:ngpus=4
#PBS -l walltime=1:00:00
#PBS -N ptbench
#PBS -j oe
#PBS -o ptbench.out
#PBS -A SCSG0001
#PBS -q main
module --force purge
module load ncarenv/23.04 cuda/11.7.1 cudnn/8.5.0.96-11.7 conda/latest
conda activate pytorch_bench
# run pytorch model on 4 GPUs
python3 benchmark_models.py -g 4
```

ML frameworks like PyTorch and TensorFlow handle GPU assignment internally

Additional MPI Environment Variables

- There are many environment settings that can affect MPI launching, performance, rank mappings, error and output handling, binding and so on. Very useful to read the man pages (**man mpi / man mpiexec**)
- A few settings that may be helpful in debugging MPI and Slingshot issues

```
MPICH_OFI_VERBOSE=1
MPICH_OFI_NIC_VERBOSE=1,2
MPICH_OFI_CXI_COUNTER_REPORT=1,2,3,4,5
MPICH_OFI_CXI_COUNTER_VERBOSE=1
MPICH_MEMORY_REPORT=1,2,3
```

Scratch Lustre File System and MPI -IO

Lustre, Cray -MPICH, and MPI -IO

- Derecho's **`/glade/derecho/scratch`** Lustre file system is the preferred location for staging large model data

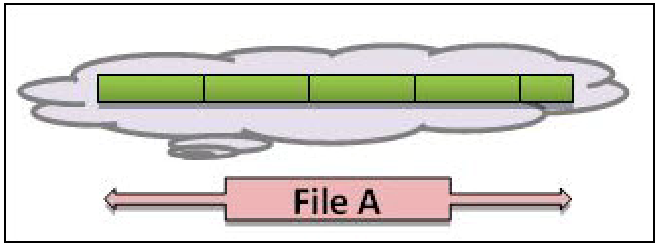
- Lustre achieves high performance through "striping" files over many storage servers
  - Sensible defaults are applied system-wide, however users may want to alter the striping parameters for a specific workflow
  - **`lfs <getstripe|setstripe>`** can be used to view/set striping parameters

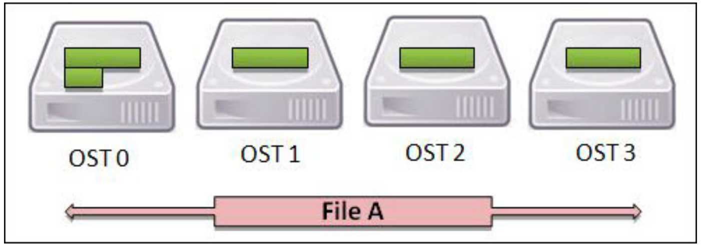
- Additionally, Lustre provides a number of user-facing tools specifically designed to ease the pain of working with large parallel file systems
  - **`lfs find`** is an efficient find replacement; **`lfs df -h`** can indicate capacity and health

- Cray-MPICH supports MPI-IO through ROMIO and has additional tuning parameters and diagnostics specifically relevant to Lustre

Lustre File Striping

- Lustre achieves high performance through file striping:





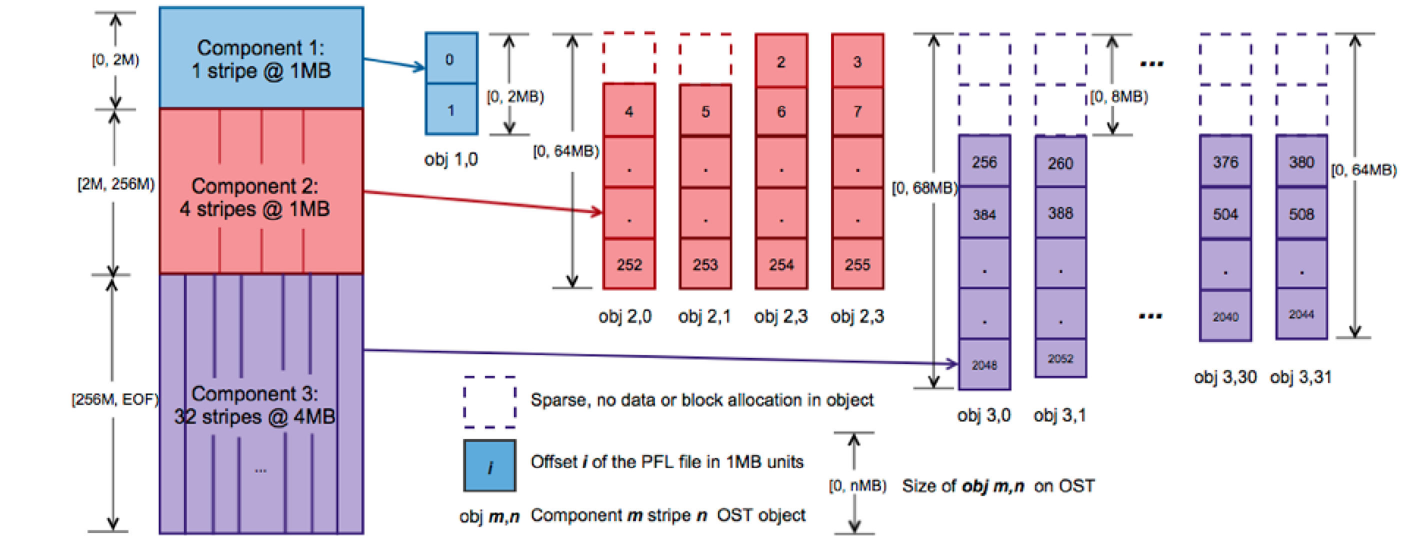
Here File A is broken into segments of a given **stripe width**. These segments are then striped across one or more storage devices according to the **stripe count**.

- The historical challenge of finding a “one-size-fits-all” striping pattern for a general purpose file system is largely mitigated by the use of **Progressive File Layouts** in modern versions of Lustre.

See our ARC Portal Documentation for more information

Lustre Progressive File Layouts

- **PFL**s allow both the stripe size and count to change as the file data extent grows.
- Most frequently used to stripe small files only over 1 or few OSTs, adding additional OSTs as the file size increases.



Mapping from 2055MB PFL file data blocks to OST objects of three components

See our ARC Portal Documentation for more information

Lustre File Striping

Sample **`lfs setstripe`** command for Derecho's default configuration:

```
lfs setstripe -E 16M -c 1  -S  1M \
              -E 16G -c 4  -S 16M \
              -E 64G -c 12 -S 16M \
              -E  -1 -c 24 -S 16M
```

File Segment	Stripe Count	Stripe Size
0 - 16MB	1	1MB
16MB - 16GB	4	16MB
16GB - 64GB	12	16MB
64GB +	24	16MB

General Striping Considerations & Tradeoffs :

- For large, aggregated files (MPI-IO or NetCDF Parallel files) the default striping should be adequate. In general you will want large stripe counts to increase read/write bandwidth.
- For applications that perform per-rank file I/O (many, many files of modest size) you might want to decrease the stripe count to as low as 1-2.

  This is because the separate files themselves will naturally spread across storage devices through round-robin allocation, and additional striping simply increases the number of remote procedure calls (RPCs) the filesystem performs, potentially negatively impacting performance.
- File striping is inherited from a parent directory, or can be set directly prior to file creation.
- **`lfs getstripe <file|dir>`** reports stripe configuration

Cray-MPICH and MPI-IO

- Cray-MPICH uses the ROMIO MPI-IO implementation and can be controlled through a large number tuning environment variables.
  See **`man MPI`** for a full listing.

Key variables for performance tuning and experimentation :

```
MPICH_MPIIO_HINTS="*:striping_factor=<STRIPE_COUNT>:striping_unit=<STRIPE_WIDTH (bytes)>"
MPICH_MPIIO_HINTS_DISPLAY=1
MPICH_MPIIO_TIMERS=1
MPICH_MPIIO_STATS=1
```

- **`MPICH_MPIIO_HINTS`** can be used to set many parameters, difficult at this point to provide general guidance.
  - Experiment and let us know what you find!
  - Can control stripe size & count, ***but only effective when creating a new file*** (not when overwriting an existing one) so remove intermediate files if you are experimenting with different values.
- **`MPICH_MPIIO_TIMERS`** & **`MPICH_MPIIO_STATS`** are key for performance profiling diagnostic output.

Cray-MPICH and MPI -IO

MPICH_MPIIO_TIMERS & **MPICH_MPIIO_STATS** diagnostic output
(# ranks=7680, # nodes=240/ppn=32, 502GB file, 96 stripes, 32MB stripe width):

```
| MPIIO write by phases, writers only, for rico.rst
|                                          min         max         ave
|                                     ----------  ----------  ----------
|   file write        time         =        3.01        3.57        3.23
|
|   time scale: 1 = 2**5     clock ticks    min         max         ave
|                                     ----------  ----------  ---------- ---
|   total                          =                          570943385
|
|   imbalance                      =       51743       71793       62964  0%
|   local compute                  =     6882480     7395141     7083814  1%
|   wait for coll                  =    11317159    18393655    15796415  2%
|   collective                     =     1272223     1353043     1313908  0%
|   exchange/write                 =      892269     1257552     1015762  0%
|   data send                      =   139823769   174452729   158803482 27%
|   sieve read                     =        2601        4964        4465  0%
|   file write                     =   276359086   328204673   296420695 51%
|   other                          =    66542347   109511397    89235916 15%
|
|   data send BW (MiB/s)           =                           3716.457
|   raw write BW (MiB/s)           =                         159118.185
|   net write BW (MiB/s)           =                          82610.508
+--------------------------------------------------------------------------+
```

Additional Resources

Resource	Cheyenne (docs)	Derecho (docs)
Login Nodes	`cheyenne.ucar.edu` ; 6 CPU	`derecho.hpc.ucar.edu` ; 6 CPU + 2 GPU
Compute Nodes	● **145,152 total CPU cores** ● 4,032 CPU nodes: 36 cores dual socket Intel Xeon E5-2697V4 (Broadwell) processors, single 25 Gb/s Mellanox EDR Infiniband port per node ○ 3,168 64 GB/node “smallmem” nodes ○ 864 128 GB/node “largemem” nodes	● **323,712 total CPU cores** ● 2,488 CPU nodes: 128 cores (dual socket) 3rd Gen AMD EPYC™ 7763 Milan processors with 256GB DDR4 RAM, single 200Gb/s Slingshot injection port per node ● 82 GPU nodes: 64 core 3rd Gen AMD EPYC™ 7763 Milan processors with 512GB DDR4 RAM, (4x) NVidia A100 40GB GPU, (4x) 200Gb/s Slingshot injection port per node
Interconnect	Mellanox EDR InfiniBand high-speed interconnect. Partial 9D Enhanced Hypercube single-plane interconnect topology with 25Gb/s bidirectional bandwidth per link.	HPE Slingshot v11 high-speed interconnect. Dragonfly topology with 200Gb/s bidirectional bandwidth per link.

PBS queues & sample `select`	**regular**, premium, economy
	`#PBS -l select=NNodes:ncpus=36:mpiprocs=18:ompthreads=2`
		main, preempt, develop
		`#PBS -l select=NNodes:ncpus=128:mpiprocs=32:ompthreads=4`
		`#PBS -l select=NNodes:ncpus=64:mpiprocs=4:ngpus=4`

User Software & Deployment	Lmod + Homegrown Installation Scripts	Lmod + Spack deployment with `spack-downstreams.sh` support for user extensions
Default Compiler	Intel 19.1.1.217 20200306	Intel 2021.8.0 20221119, use `-march=core-avx2`
Default MPI	SGI/HPE MPT v2.25, `mpiexec_mpt -n … <exe>`	cray-MPICH v8.1.25, `mpiexec -n … -ppn … <exe>`
Process Binding	`omplace` / `dmplace`	`mpiexec --cpu-bind` (see `man MPI`), `set_gpu_rank` for GPU jobs
Scratch Filesystem	`/glade/scratch/${USER}` GPFS, 10TB default quota, 120 day purge	`/glade/derecho/scratch/${USER}` Lustre, 30TB default quota, 180 day purge
Outbound Internet Access	Login nodes only	Login and Compute nodes

Getting Help & Reporting Issues

- Advanced Research Computing Documentation:
  - https://arc.ucar.edu/knowledge_base_documentation
- CISL Help Desk:
  - https://rchelp.ucar.edu
  - Submit a ticket to request help with a particular issue.
- Virtual Consulting byAppointment
- Monthly Users Meetings
- HPC Tutorials:
  - https://www2.cisl.ucar.edu/what-we-do/training-library/hpc-tutorials
  - In-depth tutorials on numerous topics, including additional details on many of the items covered here today.
    - Introduction to NCAR HPC Systems
    - Job Scheduling with PBS Pro
    - JupyterHub at NCAR
    - NCAR Storage Spaces
    - Optimizing Resource Use in Scheduled Jobs
    - Remote desktop services on Casper
    - Starting Casper Jobs with PBS Pro
    - Using Globus at NCAR

Best Practices for Support Tickets

When submitting a support ticket please include as much detail as possible to enable quicker resolution:

- Resource name (Derecho, Casper, JupyterHub,...),
- **Exact** error messages and/or paths to error output,
- Batch script location,
- PBS JobID(s) of failed effort,
- Run & source directory paths (ideally UNIX-readable by ‘others’),
- Any other pertinent information:
  - Last time this **exact** workflow was successful, if any (or changes since last success),
  - Troubleshooting steps already attempted, etc. …

And please remember to let us know when your issue is resolved!

https://rchelp.ucar.edu

Derecho -Specific Support Resources

- In addition to our Jira-based ticketing system at https://rchelp.ucar.edu , Consulting Services has implemented additional support formats suited for the fast-paced environment of Derecho deployment

- Please consider joining the *#derecho -users* channel on the NCAR HPC Users Slack workspace
  - Monitored by admins, consultants, and other users
  - Share experiences, ask questions, get rapid updates

- Issues relating to the software environment can be reported and tracked on GitHub at https://github.com/NCAR/spack -derecho

Slack

http://ncarhpcusergroup.slack.com

Check out
#derecho-users
#cheyenne-users
#casper-users

Office Hours / Advanced Topics

Advanced Topics

- Preemption
- Containers
- Extending the User Software Environment

Preemption

- Derecho provides **`apreempt`** queue which can be used to run jobs that might be low priority or suitable for interruption.

  Some possible examples:
  - An archive process suitable for incremental progress,
  - An analysis code with a robust checkpoint / restart mechanism.
- When a job is preempted by another job from a higher priority queue:
  - It can be rescheduled automatically (default behavior)
    - Controlled by the **`#PBS -r <y|n>`** rerunnable attribute (*yes* by default)
  - It is first sent a Unix **`SIGTERM`**; if properly configured can perform a user-defined action. It is killed 10 minutes later.

```
#!/bin/bash
#PBS -A project_code
#PBS -N preemptable_job
#PBS -q preempt
#PBS -r n
#PBS -l walltime=01:00:00
#PBS -l select=2:ncpus=64:mpiprocs=4:ngpus=4
...
```

PBS Job Preemption and Signal Handling

- Signal handling allows an application to “know” preemption has been requested and that it will be terminated imminently:
  - PBS will send a job **SIGTERM**, wait 10 minutes, and then kill the application with **SIGKILL**
  - An application can ‘catch’ **SIGTERM** and invoke checkpoint or cleanup functions if desired
- Signal Handling General Process:
  1. Provide a ***signal handler function*** which will receive the termination request (listing in backup),
  2. ***Registering*** this signal handler function with the operating system (listing in backup),
  3. ***Application-specific checkpointing & termination***, triggered by the signal handler.

For a complete demonstration - including integration with MPI - see

https://github.com/NCAR/hpc-demos/blob/main/PBS/preempt/minimal_mpi.cpp

Containers

- Derecho (and Gust) provide several container runtimes with a mixed set of functionality:
  - Singularity (via the Apptainer project)
  - Charliecloud
  - Podman
- For CPU applications, Singularity and Charliecloud have both been tested with cray mpich and found generally performant
  - Requires some gymnastics to bind-mount the host MPI into the container.
    - Practical implication is that you will want to use MPICH as the base MPI inside your container.
- Additionally, Singularity has been used to run GPU applications with cray-mpich (notably, the open-source version of FastEddy®)

Details: 2023-04 State of Containers

Cray-MPICH & Containers

Running performant MPI inside a container requires judiciously bind-mounting the host MPI "on top of" the container MPI. The two MPIs need to be ABI compatible, which is readily accomplished by using MPICH 3.4.x inside the container.

```
mpiexec --np 2 --ppn 1 --no-transfer \
   set_gpu_rank \
   singularity exec \
      --bind /run \
      --bind /usr/lib64:/host/lib64 \
      --bind /opt/cray \
      --env LD_LIBRARY_PATH=${CRAY_MPICH_DIR}/lib-abi-mpich:/opt/cray/pe/lib64:${LD_LIBRARY_PATH}:/host/lib64 \
      --env MPICH_SMP_SINGLE_COPY_MODE=NONE \           # necessary for successful execution
      ${container_image} ${exe}
```

Details: 2023-04 State of Containers

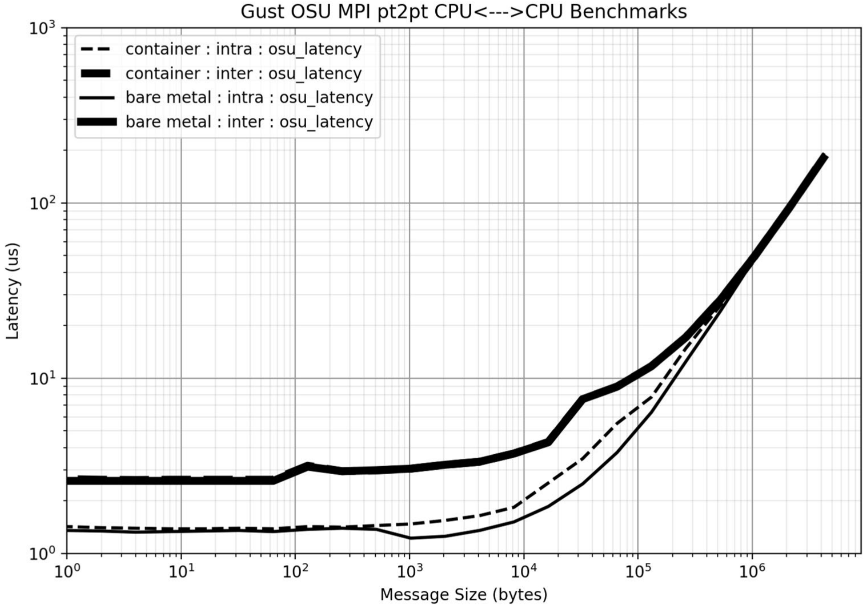
CUDA-Aware Cray -MPICH & Containers

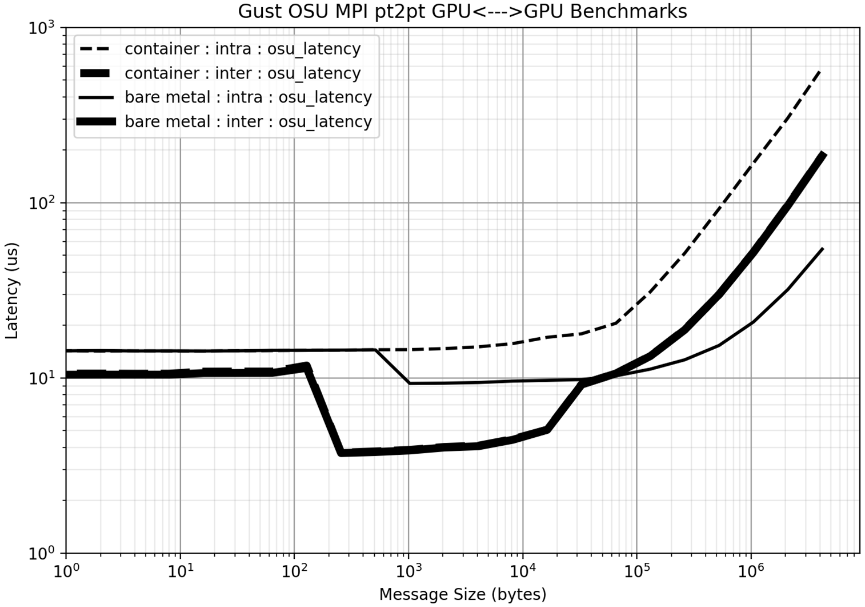
Running CUDA-Aware MPI inside a container can be integrated with Cray-MPICH in much the same way. The final trick is to inject the missing GTL library with **`LD_PRELOAD`**.

```
mpiexec --np 2 --ppn 1 --no-transfer \
   set_gpu_rank \
   singularity exec \
      --bind /run \
      --bind /usr/lib64:/host/lib64 \
      --bind /opt/cray \
      --env LD_LIBRARY_PATH=${CRAY_MPICH_DIR}/lib-abi-mpich:/opt/cray/pe/lib64:${LD_LIBRARY_PATH}:/host/lib64 \
      --env MPICH_SMP_SINGLE_COPY_MODE=NONE \            # necessary for successful execution
      --env MPICH_GPU_SUPPORT_ENABLED=1 \                # typical MPI/GPU env vars
      --env MPICH_GPU_MANAGED_MEMORY_SUPPORT_ENABLED=1 \ # typical MPI/GPU env vars
      --env LD_PRELOAD=/opt/cray/pe/mpich/8.1.21/gtl/lib/libmpi_gtl_cuda.so.0 \
      ${container_image} ${exe}
```

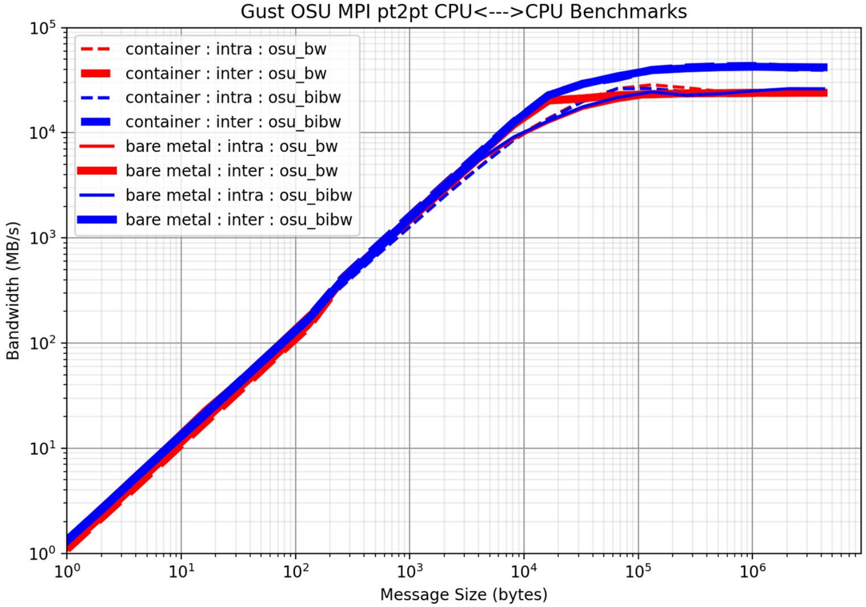
Details: 2023-04 State of Containers

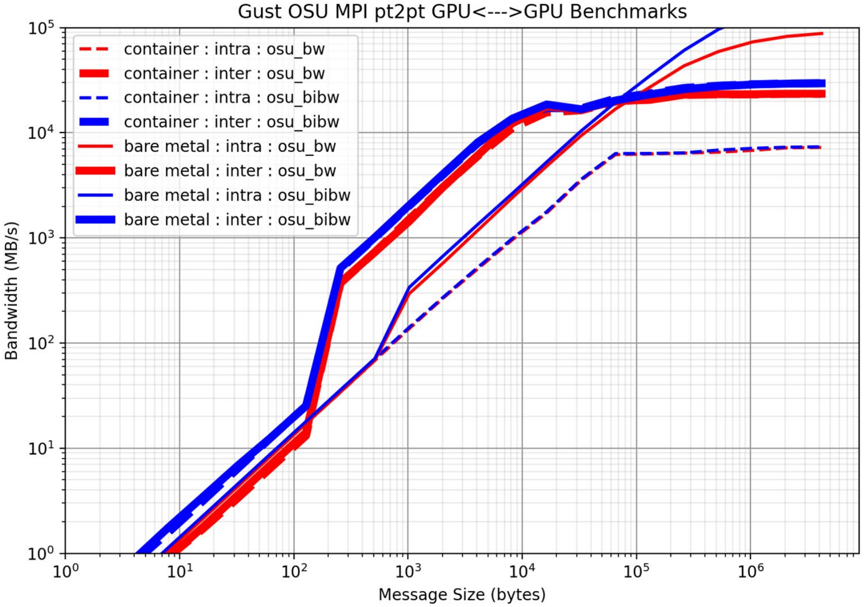
Results: pt2pt `osu_latency`





Results: pt2pt `osu_bw / osu_bibw`





Extending the User Software Environment

- The Spack package manager allows for the extension of the system environment with a connected user Spack installation
- This user Spack installation can be used to install software to the user’s work directory
  - Can utilize the system Spack packages as dependencies for any application a user installs.
  - Easily install versions of applications that may not be available on Derecho
- CISL managed installation and linking script **`spack-downstreams.sh`**

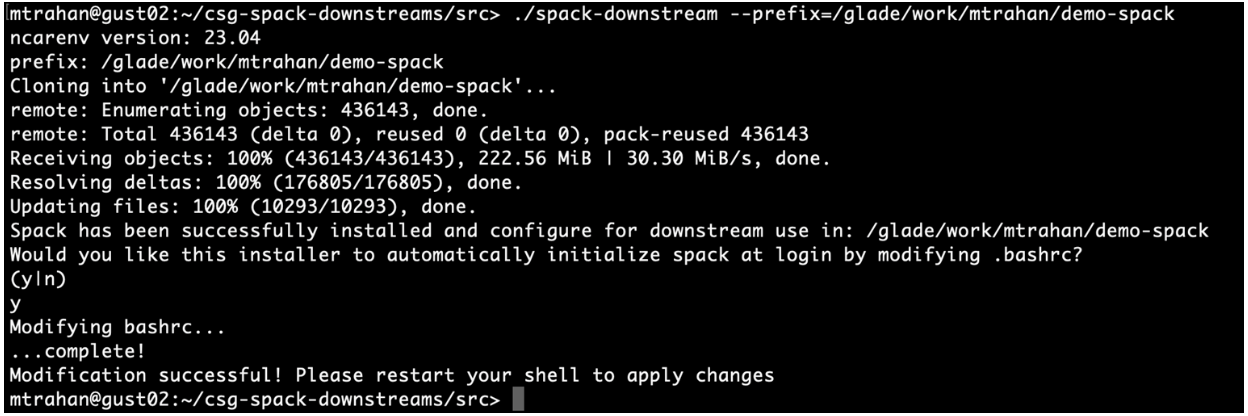
Spack Downstream Setup

- Basic Usage: ./ **spack - downstreams.sh**

```
-v|--verbose                 : print out each installation step to the terminal
   --prefix=<install-path>   : specify spack installation location. Default:
                               /glade/work/<user>/spack_version
   --modify-rc=<True|False>. : modify .bashrc to load Spack at startup.
-h|--help                    : print this message
```

- Application pulls from spack from github and installs the application in a default space on your work directory
- Clones over configuration from the system installation
- Adds a setup command in your bashrc
  - Can opt out if you'd like to initialize it yourself!

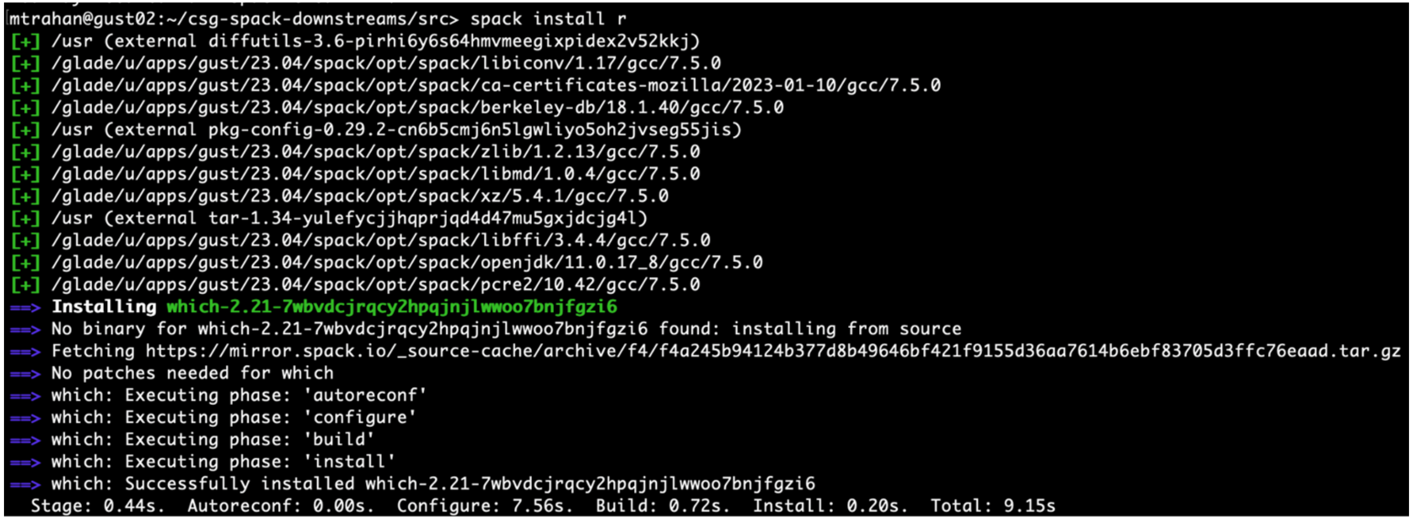
Spack Downstreams



```
mtrahan@gust02:~/csg-spack-downstreams/src> ./spack-downstream --prefix=/glade/work/mtrahan/demo-spack
ncarenv version: 23.04
prefix: /glade/work/mtrahan/demo-spack
Cloning into '/glade/work/mtrahan/demo-spack'...
remote: Enumerating objects: 436143, done.
remote: Total 436143 (delta 0), reused 0 (delta 0), pack-reused 436143
Receiving objects: 100% (436143/436143), 222.56 MiB | 30.30 MiB/s, done.
Resolving deltas: 100% (176805/176805), done.
Updating files: 100% (10293/10293), done.
Spack has been successfully installed and configure for downstream use in: /glade/work/mtrahan/demo-spack
Would you like this installer to automatically initialize spack at login by modifying .bashrc?
(y|n)
y
Modifying bashrc...
...complete!
Modification successful! Please restart your shell to apply changes
mtrahan@gust02:~/csg-spack-downstreams/src>
```

Installation can be done in any directory you need.

Applications you install will detect any dependencies in the upstream spack instance.



```
mtrahan@gust02:~/csg-spack-downstreams/src> spack install r
[+] /usr (external diffutils-3.6-pirhi6y6s64hmvmeegixpidex2v52kkj)
[+] /glade/u/apps/gust/23.04/spack/opt/spack/libiconv/1.17/gcc/7.5.0
[+] /glade/u/apps/gust/23.04/spack/opt/spack/ca-certificates-mozilla/2023-01-10/gcc/7.5.0
[+] /glade/u/apps/gust/23.04/spack/opt/spack/berkeley-db/18.1.40/gcc/7.5.0
[+] /usr (external pkg-config-0.29.2-cn6b5cmj6n5lgwliyo5oh2jvseg55jis)
[+] /glade/u/apps/gust/23.04/spack/opt/spack/zlib/1.2.13/gcc/7.5.0
[+] /glade/u/apps/gust/23.04/spack/opt/spack/libmd/1.0.4/gcc/7.5.0
[+] /glade/u/apps/gust/23.04/spack/opt/spack/xz/5.4.1/gcc/7.5.0
[+] /usr (external tar-1.34-yulefycjjhqprjqd4d47mu5gxjdcjg4l)
[+] /glade/u/apps/gust/23.04/spack/opt/spack/libffi/3.4.4/gcc/7.5.0
[+] /glade/u/apps/gust/23.04/spack/opt/spack/openjdk/11.0.17_8/gcc/7.5.0
[+] /glade/u/apps/gust/23.04/spack/opt/spack/pcre2/10.42/gcc/7.5.0
==> Installing which-2.21-7wbvdcjrqcy2hpqjnjlwwoo7bnjfgzi6
==> No binary for which-2.21-7wbvdcjrqcy2hpqjnjlwwoo7bnjfgzi6 found: installing from source
==> Fetching https://mirror.spack.io/_source-cache/archive/f4/f4a245b94124b377d8b49646bf421f9155d36aa7614b6ebf83705d3ffc76eaad.tar.gz
==> No patches needed for which
==> which: Executing phase: 'autoreconf'
==> which: Executing phase: 'configure'
==> which: Executing phase: 'build'
==> which: Executing phase: 'install'
==> which: Successfully installed which-2.21-7wbvdcjrqcy2hpqjnjlwwoo7bnjfgzi6
  Stage: 0.44s.  Autoreconf: 0.00s.  Configure: 7.56s.  Build: 0.72s.  Install: 0.20s.  Total: 9.15s
```

Spack Downstreams Planned features

- Support for **`tcsh`** and **`csh`**
- User spack module integration with upstream spack
- Additional chaining for multiple spack installations
- Better cleanup utilities for easy removal

To get started with spack upstreams, contact CISL help for the application

Backup

PBS Job Preemption and Signal Handling

```c
#include <stdio.h>
#include <unistd.h>
#include <time.h>
#include <signal.h>

static int checkpoint_requested = 0;

/* a signal handler can be any function that takes an int and returns void */
void my_sig_handler (int signum)
{
 time_t now;
 time(&now);

 switch (signum)
   {
   case SIGINT:
   case SIGTERM:
   case SIGUSR1:
     checkpoint_requested = 1;
     printf("...caught signal %d at %s", signum, ctime(&now));
     break;

   default:
     printf("...caught other unknown signal: %d at %s", signum, ctime(&now));
     printf("   see \"man 7 signal\" for a list of known signals\n");
     break;
   }
}
```

PBS Job Preemption and Signal Handling

```c
#include <signal.h>

int main (int argc, char **argv)
{
  /* register our user-defined signal handler, can be used to catch multiple signals */
  signal(SIGINT,  my_sig_handler);
  signal(SIGTERM, my_sig_handler);
  signal(SIGUSR1, my_sig_handler);

  return 0;
}
```

Cross -Compiler & Cross -System Development Strategies

With the new Spack user software environment we have introduced several environment variables developers may find useful when switching between compilers, MPIs and even systems:

```
benkirk@casper16(2)$ module list && env | grep BUILD

Currently Loaded Modules:
  1) ncarenv/23.04          (S)   4) cuda/11.7.1      7) hdf5/1.12.2
  2) intel/2023.0.0               5) ucx/1.13.1       8) netcdf/4.9.1
  3) ncarcompilers/0.8.0          6) openmpi/4.1.5

...

NCAR_BUILD_ENV_COMPILER=casper-oneapi-2023.0.0
NCAR_BUILD_ENV_MPI=casper-oneapi-2023.0.0-openmpi-4.1.5
NCAR_BUILD_ENV=casper-oneapi-2023.0.0-openmpi-4.1.5
```

Cross -Compiler & Cross -System Development Strategies

- The ${NCAR_BUILD_ENV*} and previous GLADE environment variables can be used to simplify builds across systems, compiler stacks, and facilitate portable shell scripts.
- This is especially true when using properly configured Autotools or CMake packages that support distinct source, build, and installation directories:

```
# check mytool builds with (6 compiler suites) X (3 MPI families)

$ cd ${WORK}/codes && git clone <mytool> && cd ./mytool && SRC_DIR=$(pwd)

$ for COMPILER in nvhpc gcc cce intel intel-oneapi intel-classic; do
   for MPI in cray-mpich mvapich2 openmpi; do
    module load ${COMPILER} ${MPI} || continue
    cd ${SRC_DIR} && mkdir -p ${NCAR_BUILD_ENV} && cd ${NCAR_BUILD_ENV}
    ../configure --prefix=$(pwd)/install && make && make install
   done
  done

$ PATH=${WORK}/mytool/${NCAR_BUILD_ENV}/install/bin:${PATH}
```

** Today we only have cray-mpich on Derecho, but are pursuing others.*

Compiling code with Intel Compilers

```
[negins@derecho3 ~]:
└──> module list

Currently Loaded Modules:
  1) ncarenv/23.04   (S)    4) ncarcompilers/0.8.0    7) netcdf/4.9.1
  2) craype/2.7.20          5) cray-mpich/8.1.25
  3) intel/2023.0.0         6) hdf5/1.12.2
```

Compiling with Intel Classic Compiler:

```
[negins@derecho3 ~]:
└──> ifort model.f90 -o model -qopenmp
```

Compiling with Intel OneAPI :

```
[negins@derecho3 ~]:
└──> ifx model.f90 -o model -qopenmp
```

Compiling code with Cray Compiling Environment (CCE)

```
[negins@derecho3 ~]:
└──> module swap intel/2023.0.0 cce/15.0.1
```

Compiling with CCE

```
[negins@derecho3 ~]:
└──> ftn model.f90 -o model -fopenmp
```

The **`ncarcompilers`** module will translate a call to “**`mpifort`**” to “**`ftn`**” :

```
[negins@derecho3 ~]:
└──> mpifort model.f90 -o model -fopenmp
```

Synchronizing *Cheyenne* Scratch → *Derecho* Scratch

- *Derecho's* scratch filesystem:
  - **`/glade/derecho/scratch/${USER}`**, also **`${SCRATCH}`** and **`${DERECHO_SCRATCH}`**
- *Cheyenne*'s scratch filesystem is also available on *Derecho*:
  - **`/glade/cheyenne/scratch/${USER}`**, also **`${CHEYENNE_SCRATCH}`**
- Small files or directories can be relocated using **`mv`** or **`cp`**
- Large directory trees can be synchronized with **`rsync`** or using Globus
- For users that want to move from *Cheyenne* scratch to *Derecho* entirely, we have developed a utility PBS script.
  - Replicates *Cheyenne* scratch contents into **`/glade/derecho/scratch/${USER}/FROM_CHEYENNE/`**
  - Once files are synchronized (expensive), they can be **`mv`**'ed within the same filesystem (quick)

```
# PBS Script to synchronize contents
# From: /glade/cheyenne/scratch/${USER}/
# To:   /glade/derecho/scratch/${USER}/FROM_CHEYENNE/

$ qsub -A <ACCOUNT> \
   /glade/u/home/benkirk/repos/csg-utils/filesystem/scratch_migration/sync_scratch.sh
```

Independent Tasks & PBS Job Arrays

- *Derecho* nodes are uniquely assigned to user jobs, therefore efficient user workflows should make full use of these dedicated resources.
- *Casper* is the ideal for very small jobs, requiring only a handful of CPU cores.
- For large parametric sweeps of small jobs (e.g. 1000s), however, it is possible to "pack" many each job onto Derecho nodes.
  - On *Cheyenne*, this could be accomplished using MPT to launch a series of independent processes .
  - In the following, we demonstrate how to launch many similar, serial tasks with different inputs on *Derecho* compute nodes using PBS Job Arrays.

```
# The -J min-max syntax specifies a Job Array.
# the script job.pbs will be executed repeatedly, each with a unique
# PBS_ARRAY_INDEX in [min,max] (inclusive)
# job.pbs can therefore be considered a template that is applied repeatedly

$ qsub -J 0-7 job.pbs
```

Independent Tasks & PBS Job Arrays

```
$ git clone --branch derecho /glade/work/benkirk/consulting/ASD/job_arrays

$ cd job_arrays
$ cat README
test.py: demo python script that sleeps a few random seconds and prints whatever command
         line arguments it was called with. Demonstration surrogate for user application.

inputs.txt:   command-line arguments for each step.
              '#' in the first column denotes a comment.
              each non-comment line indicates a 'step' that will be run.

getline.sh: a simple bash script to return the requested (non-comment) line.
            usage: ./getline.sh ./inputs.txt <linenumber>

job.pbs:   A PBS script that uses PBS "job arrays" to execute each step in inputs.txt
           Each PBS submission will request a full node, and use it to execute the number
           of steps equal to to the number of processors on the node.
           (the last node will be undersubscribed in general.)

Makefile: glues it all together.
          change the project code on the first line.
          ppn is set at 128, which is appropriate for Cheyenne or Casper.
          determines the number of steps from inputs.txt
          determines the number of PBS 'job array steps'
          submits to derecho. make run to launch.
```

Replace with user - provided elements

Independent Tasks & PBS Job Arrays, PBS Script (1/2)

```
#!/bin/bash
#PBS -N array_example
#PBS -j oe
#PBS -l walltime=00:10:00
## PBS_ARRAY_INDEX range, inclusive:  (can be overridden by qsub command line arguments)
#PBS -J 0-3

### Set temp to scratch
export TMPDIR=${SCRATCH}/tmp && mkdir -p ${TMPDIR}

## determine the number of nodes, and processors per node we were assigned
## (inferred based on select statement)
nodeslist=( $(cat ${PBS_NODEFILE} | sort | uniq | cut -d'.' -f1) )
nnodes=$(cat ${PBS_NODEFILE} | sort | uniq | wc -l)
nranks=$(cat ${PBS_NODEFILE} | sort | wc -l)
nranks_per_node=$((${nranks} / ${nnodes}))

[ ${nnodes} -eq 1 ] || { echo "ERROR: this example is for 1 node, but with perhaps many array steps"; exit 1; }

echo "${nranks} ${nnodes}x${nranks_per_node}"

nsteps=$( cat inputs.txt | grep -v '#' | wc -l )

# this PBS_ARRAY_INDEX will compute multiple "steps" from inputs.txt, up to ppn
start_idx=$(( ${PBS_ARRAY_INDEX} * ${nranks_per_node} ))
stop_idx=$(( ${start_idx} + ${nranks_per_node} - 1 ))

echo "nsteps: ${nsteps}, array index: ${PBS_ARRAY_INDEX}"
echo "start_idx=${start_idx} stop_idx=${stop_idx}"
...
```

Independent Tasks & PBS Job Arrays, PBS Script (2/2)

```
...
# create a 'logs/ directory to hold stdout from each process
mkdir -p ./logs/

# loop over each 'step' for which we are responsible.
# launch our ./test.py process, in the background
for step in $(seq ${start_idx} ${stop_idx}); do

    # the last PBS_ARRAY_INDEX could go past nsteps if the number of inputs.txt
    # is not evenly divisible by ppn - don't let it
    [ ${step} -ge ${nsteps} ] && break

    # get the command line arguments from inputs.txt for this step
    # (note that the step counter is 0 based, so add 1)
    cmdargs=$( ./getline.sh ./inputs.txt $(( ${step} + 1)) )
    echo "   PBS_ARRAY_INDEX=${PBS_ARRAY_INDEX} launching step ${step} / args=${cmdargs}"

    # finally, launch our desired application with the requested arguments.
    # Redirect stdout/stderr to the ./logs/ directory.
    ./test.py ${cmdargs} > ./logs/stdout-$(printf '%04d' $((${step}+1))).log 2>&1 &
done

# wait for all the background processes to complete.
# (otherwise, when this script exits, PBS thinks it is done and will kill any remaining processes...)
wait

echo "Done: PBS_ARRAY_INDEX=${PBS_ARRAY_INDEX} finished on $(date)"
```

⧆ This example works best when each “task” has a ~similar duration. If a small number of tasks take much longer than average, this approach will lead to imbalance. Reach out to Consulting for other approaches to address this scenario.

Intel Modules available on Derecho

Module	Fortran (ftn)	C (cc)	C++ (CC)
`intel/2023.0.0 (D)`	`ifort`	`icx`	`icpx`
`intel-oneapi/2023.0.0`	`ifx`	`icx`	`icpx`
`intel-classic/2023.0.0`	`ifort`	`icc`	`icpc`

Intel C/C++ Compiler Classic (icc/icpc) is deprecated and will be removed in a oneAPI release in the second half of 2023.

Intel recommends that customers transition now to using the LLVM-based Intel® oneAPI DPC++/C++ Compiler (icx/icpx) for continued Windows* and Linux* support, new language support, new language features, and optimizations.

Intel Fortran Compiler Classic (ifort) is going to be deprecated soon and replaced by ifx.