Copy of Delta XSEDE Documentation

This page was moved to https://wiki.ncsa.illinois.edu/display/DSC/Delta+User+Guide
Click in the link above if you are not automatically redirected in 10 seconds.

Delta User Guide

Last update: March 16, 2021

Status Updates and Notices

Delta is tentatively scheduled to enter production during the allocation period starting July 1, 2021.

Introduction

Delta is a dedicated, eXtreme Science and Engineering Science Discovery Environment (XSEDE) allocated resource designed by HPE and NCSA, delivering a highly capable GPU-focused compute environment for GPU and CPU workloads. Besides offering a mix of standard and reduced precision GPU resources, *Delta* also offers GPU-dense nodes with both NVIDIA and AMD GPUs. *Delta* provides high performance node-local SSD scratch filesystems, as well as both standard lustre and relaxed-POSIX parallel filesystems spanning the entire resource.

Delta's standard CPU nodes are each powered by two 64-core AMD EPYC 7763 ("Milan") processors, with 256 GB of DDR4 memory. The *Delta* GPU resource has four node types: one with 4 NVIDIA A100 GPUs (40 GB HBM2 RAM each) connected via NVLINK and 1 64-core AMD EPYC 7763 ("Milan") processor, the second with 4 NVIDIA A40 GPUs (48 GB GDDR6 RAM) connected via PCIe 4.0 and 1 64-core AMD EPYC 7763 ("Milan") processor, the third with 8 NVIDIA A100 GPUs in a dual socket AMD EPYC 7763 (128-cores per node) node with 2 TB of DDR4 RAM and NVLINK, and the fourth with 8 AMD MI100 GPUs (32GB HBM2 RAM each) in a dual socket AMD EPYC 7763 (128-cores per node) node with 2 TB of DDR4 RAM and PCIe 4.0.

Delta has 124 standard CPU nodes, 100 4-way A100-based GPU nodes, 100 4-way A40-based GPU nodes, 5 8-way A100-based GPU nodes, and 1 8-way MI100-based GPU node. Every *Delta* node has high-performance node-local SSD storage (800 GB for CPU nodes, 1.6 TB for GPU nodes), and is connected to the 7 PB Lustre parallel filesystem via the high-speed interconnect. The *Delta* resource uses the SLURM workload manager for job scheduling.

Delta supports the XSEDE core software stack, including remote login, remote computation, data movement, science workflow support, and science gateway support toolkits.

? Unknown Attachment

Figure 1. Delta System

Delta is supported by the National Science Foundation under Grant No. OAC-2005572.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

***Delta* is now accepting proposals.**

Top of Page

Account Administration

- Setting up Your Account
- Allocation Information
- How to Access the System

Configuring Your Account

- default shell, changing your shell, changing your password
- environment variables
- using Modules (or other environment manager)

System Architecture

Delta is designed to help applications transition from CPU-only to GPU or hybrid CPU-GPU codes. Delta has some important architectural features to facilitate new discovery and insight:

- a single processor architecture (AMD) across all node types: CPU and GPU
- support for NVIDIA A100 MIG GPU partitioning allowing for fractional use of the A100s if your workload isn't able to exploit an entire A100 efficiently
- ray tracing hardware support from the NVIDIA A40 GPUs
- 9 large memory (2 TB) nodes
- a low latency and high bandwidth HPE/Cray Slingshot interconnect between compute nodes
- lustre for home, projects and scratch file systems
- support for relaxed and non-posix IO
- shared-node jobs and the single core and single MIG GPU slice
- Resources for persistent services in support of Gateways, Open OnDemand, Data Transport nodes...,
- Unique AMD MI-100 resource

Model Compute Nodes

The Delta compute ecosystem is composed of 5 node types: dual-socket CPU-only compute nodes, single socket 4-way NVIDIA A100 GPU compute nodes, single socket 4-way NVIDIA A40 GPU compute nodes, dual-socket 8-way NVIDIA A100 GPU compute nodes, and a single socket 8-way AMD MI100 GPU compute nodes. The CPU-only and 4-way GPU nodes have 256 GB of RAM per node while the 8-way GPU nodes have 2 TB of RAM. The CPU-only node has 0.8 TB of local storage while all GPU nodes have 1.6 TB of local storage.

Table. CPU Compute Node Specifications

Specification	Value
Number of nodes	124
CPU	AMD Milan (PCIe Gen4)
Sockets per node	2
Cores per socket	64
Cores per node	128
Hardware threads per core	2
Hardware threads per node	128
Clock rate (GHz)	~ 2.45
RAM (GB)	256
Cache (MB) L1/L2/L3	2/32/256
Local storage (TB)	0.8 TB

Table. 4-way NVIDIA A40 GPU Compute Node Specifications

Specification	Value
Number of nodes	100
GPU	NVIDIA A40 (Vendor page)
GPUs per node	4
GPU Memory (GB)	48 DDR6 with ECC
CPU	AMD Milan
CPU sockets per node	1
Cores per socket	64
Cores per node	64
Hardware threads per core	2

Hardware threads per node	128
Clock rate (GHz)	~ 2.45
RAM (GB)	256
Cache (MB) L1/L2/L3	2/32/256
Local storage (TB)	1.6 TB

Table. 4-way NVIDIA A100 GPU Compute Node Specifications

Specification	Value
Number of nodes	100
GPU	NVIDIA A100 (Vendor page)
GPUs per node	4
GPU Memory (GB)	40
CPU	AMD Milan
CPU sockets per node	2
Cores per socket	64
Cores per node	64
Hardware threads per core	1
Hardware threads per node	64
Clock rate (GHz)	~ 2.45
RAM (GB)	256
Cache (MB) L1/L2/L3	2/32/256
Local storage (TB)	1.6 TB

Table. 8-way NVIDIA A100 GPU Large Memory Compute Node Specifications

Specification	Value
Number of nodes	5
GPU	NVIDIA A100 (Vendor page)
GPUs per node	8
GPU Memory (GB)	40
CPU	AMD Milan
CPU sockets per node	2
Cores per socket	64
Cores per node	128
Hardware threads per core	2
Hardware threads per node	256
Clock rate (GHz)	~ 2.45
RAM (GB)	2,048
Cache (MB) L1/L2/L3	2/32/256
Local storage (TB)	1.6 TB

Table. 8-way AMD MI100 GPU Large Memory Compute Node Specifications

Specification	Value
Number of nodes	1
GPU	AMD MI100 (Vendor page)
GPUs per node	8
GPU Memory (GB)	32
CPU	AMD Milan
CPU sockets per node	2
Cores per socket	64
Cores per node	128
Hardware threads per core	2
Hardware threads per node	256
Clock rate (GHz)	~ 2.45
RAM (GB)	2,048
Cache (MB) L1/L2/L3	2/32/256
Local storage (TB)	1.6 TB

Login Nodes

Describe login node/s.

Specialized Nodes

Delta will support data transfer nodes or nodes in support of other services.

Network

Delta will be connected to the WAN via two 100Gbit connections.

Delta resources will be inter-connected with HPE/Cray's 100Gbit/200Gbit SlingShot

File Systems

Note: Users of Delta have access to 2 file systems at the time of system launch, a third relaxed-POSIX file system will be made available at a later date.

Delta
The Delta file system provides users with their $HOME and $scratch areas. This file system is mounted across all Delta systems at /delta and is accessible on the Delta DTN Endpoint. The aggregate performance of this subsystem is 75GB/s and it has 6PB of usable space. /delta is a Lustre file system running DDN Exascaler.

Hardware:
DDN SFA7990XE (Quantity: 3), each unit contains

- One additional SS9012 enclosure
- 168 x 16TB SAS Drives
- 7 x 1.92TB SAS SSDs

Future Hardware:
An additional pool of NVME flash from DDN will be installed in September of 2021. This flash will initially be deployed for additional metadata capability; as well as a tier for "hot" data in scratch. This subsystem will have an aggregate performance of 600GB/s and will have 3PB of usable space.

Taiga
Taiga is NCSA's global file system which provides users with their $WORK area. This file system is mounted across all Delta systems at /taiga and is accessible on both the *Delta* and *Taiga* DTN endpoints. For Illinois researchers, *Taiga* is also mounted on *HAL* and *Radiant*. This storage subsystem has an aggregate performance of 140GB/s and 1PB of its capacity allocated to users of the *Delta* system. /taiga is a Lustre file system running DDN Exascaler.

Hardware:
DDN SFA400NVXE (Quantity: 2), each unit contains

- 4 x SS9012 enclosures
- NVME for metadata and small files

DDN SFA18XE (Quantity: 1), each unit contains

- 10 x SS9012 enclosures

File System	Quota	Snapshots	Purged	Key Features
$HOME	**25GB.** 400,000 files per user.	No /TBA	No	Area for software, scripts, job files, etc. **NOT** intended as a source/destination for I/O during jobs
$WORK	**500 GB**. Up to 1-25 TB by allocation request	No /TBA	No	Area for shared data for a project, common data sets, software, results, etc.
$SCRATCH	**1000 GB**. Up to 1-100 TB by allocation request.	No	Yes; files older than 30-days (access time)	Area for computation, largest allocations, where I/O from jobs should occur

Top of Page

Accessing the System

Describe access to the system

- NCSA Duo enabled multi-factor authentication
- available via SSO hub

List and detail methods (e.g., ssh, Globus, gsissh), providing command-line examples.

XSEDE Single Sign-On Hub

XSEDE users can also access Delta via the XSEDE Single Sign-On Hub.

When reporting a problem to the help desk, please execute the gsissh command with the "-vvv" option and include the verbose output in your problem description.

Citizenship

You share Delta with thousands of other users, and what you do on the system affects others. Exercise good citizenship to ensure that your activity does not adversely impact the system and the research community with whom you share it. Here are some rules of thumb.

List any Best Practices or conversely, a list of don't's. Some examples:

- Don't run jobs on the login nodes
- Don't stress filesystem with known-harmful access patterns (many thousands of small files in a single directory)
- submit an informative help-desk ticket

Managing and Transferring Files

File Systems

- Tips on navigating any shared file systems

- Detail any pertinent environment variables, e.g., $HOME, $WORK, and any built-in aliases.
- Tips on backups/storage

Transferring your Files

Discuss methods of transferring files and provide command-line examples

- scp
- rsync
- Globus

Sharing Files with Collaborators

Building Software

GCC, AOCC, PGI

OpenMPI ...

OpenMP

OpenACC

Describe how to build software:

Serial

To build (compile and link) a serial program in Fortran, C, and C++:

GCC	AOCC	PGI
gfortran *myprog*.f gcc *myprog*.c g++ *myprog*.cc	flang *myprog*.f clang *myprog*.c clang *myprog*.cc	pgfortran *myprog*.f pgcc *myprog*.c pgc++ *myprog*.cc

MPI

To build (compile and link) a MPI program in Fortran, C, and C++:

MPI Implementation	modulefile for MPI/Compiler	Build Commands
OpenMPI (Home Page / Documentation)	TBD	Fortran 77: mpif77 *myprog*.f Fortran 90: mpif90 *myprog*.f90 C: mpicc *myprog*.c C++: mpicxx *myprog*.cc
TBD	TBD	

OpenMP

To build an OpenMP program, use the -fopenmp / -mp option:

GCC	AOCC	PGI
gfortran -fopenmp *myprog*.f gcc -fopenmp *myprog*.c g++ -fopenmp *myprog*.cc	flang -fopenmp *myprog*.f clang -fopenmp *myprog*.c clang -fopenmp *myprog*.cc	pgfortran -mp *myprog*.f pgcc -mp *myprog*.c pgc++ -mp *myprog*.cc

Hybrid MPI/OpenMP

To build an MPI/OpenMP hybrid program, use the -fopenmp / -mp option with the MPI compiling commands:

GCC		PGI

mpif77 -fopenmp *myprog*.f mpif90 -fopenmp *myprog*.f90 mpicc -fopenmp *myprog*.c mpicxx -fopenmp *myprog*.cc		mpif77 -mp *myprog*.f mpif90 -mp *myprog*.f90 mpicc -mp *myprog*.c mpicxx -mp *myprog*.cc

OpenACC

To build an OpenACC program, use the -acc option and the -mp option for multi-threaded:

NON-MULTITHREADED		**MULTITHREADED**
pgfortran -acc *myprog*.f pgcc -acc *myprog*.c pgc++ -acc *myprog*.cc		pgfortran -acc -mp *myprog*.f pgcc -acc -mp *myprog*.c pgc++ -acc -mp *myprog*.cc

- list compilers and recommendations
- any architecture-specific flags
- how to build 3rd party software in your account

Software

- lmod
- spack/EasyBuild
- NVIDIA NGC containers
- OpenCL
- CUDA

- How to search for/discover locally installed software
- Include job scripts for commonly run software packages
- describe procedures for any licenses

Launching Applications (TBD)

- Launching One Serial Application
- Launching One Multi-Threaded Application
- Launching One MPI Application
- Launching One Hybrid (MPI+Threads) Application
- More Than One Serial Application in the Same Job
- MPI Applications One at a Time
- More than One MPI Application Running Concurrently
- More than One OpenMP Application Running Concurrently

Running Jobs

Job Accounting

The charge unit for *Delta* is the Service Unit (SU). This corresponds to the equivalent use of one compute core utilizing less than or equal to 2G of memory for one hour, or 1 GPU or fractional GPU using less than the corresponding amount of memory or cores for 1 hour (see table below). *Keep in mind that your charges are based on the resources that are reserved for your job and don't necessarily reflect how the resources are used.* Charges are based on either the number of cores or the fraction of the memory requested, whichever is larger. The minimum charge for any job is 1 SU.

Node Type		**Service Unit Equivalence**		
		Cores	GPU Fraction	Host Memory
CPU Node		1	N/A	2 GB
GPU Node	Quad A100	2	1/7 A100	8 GB
	Quad A40	16	1 A40	64 GB
	8-way A100	2	1/7 A100	32 GB
	8-way MI100	16	1 MI100	256 GB

Please note that a weighting factor will discount the charge for the reduced-precision A40 nodes, as well as the novel AMD MI100 based node - this will be documented through the XSEDE SU converter.

Job Accounting Considerations

- A node-exclusive job that runs on a compute node for one hour will be charged 128 SUs (128 cores x 1 hour)
- A node-exclusive job that runs on a 4-way GPU node for one hour will be charge 4 SUs (4 GPU x 1 hour)
- A node-exclusive job that runs on a 8-way GPU node for one hour will be charge 8 SUs (8 GPU x 1 hour)
- A shared job that runs on an A100 node will be charged for the fractional usage of the A100 (eg, using 1/7 of an A100 for one hour will be 1/7 GPU x 1 hour, or 1/7 SU per hour, except the first hour will be 1 SU (minimum job charge).

Accessing the Compute Nodes

Describe how to run jobs

- batch job
- interactive sessions
- ssh from a login node directly to a compute node

Job Scheduler

Describe the job scheduler & scheduling algorithms

Most, if not all, XSEDE resources are running Slurm and this documentation already exists in some form.

Partitions (Queues)

Describe current partitions.

Table. Delta Production Queues

Queue Name	**Node Type**	**Max Nodes per Job**	**Max Duration**	**Max Jobs in Queue***	**SU Charge Rate** **(per node-hour)**
TBD	TBD	TBD	TDB	TDB	TBD

Node Policies

Node-sharing will supported

GPU NVIDIA MIG (GPU slicing) for the A100 will be supported.

Interactive Sessions

Describe any tools for running interactive jobs on the compute nodes.

- built-in tools for running interactive jobs, e.g. PSC’s interact, TACC’s idev

Sample Job Scripts (TBD)

Sample job scripts are the most requested documentation.

Provide sample job scripts for common job type scenarios.

- Serial jobs
- MPI
- OpenMP
- Hybrid (MPI + OpenMP)
- Parametric / Array / HTC jobs

Job Management

Batch jobs are submitted through a *job script* using the sbatch command. Job scripts generally start with a series of SLURM *directives* that describe requirements of the job such as number of nodes, wall time required, etc… to the batch system/scheduler (SLURM directives can also be specified as options on the sbatch command line; command line options take precedence over those in the script). The rest of the batch script consists of user commands.

The syntax for sbatch is:

sbatch [list of sbatch options] script_name

The main sbatch options are listed below. Refer to the sbatch man page for options.

- The common resource_names are:
  --time=*time*

  time=maximum wall clock time (d-hh:mm:ss) *[default: maximum limit of the queue(partition) summitted to]*

  --nodes=*n*

  --ntasks=*p* Total number of cores for the batch job

  --ntasks-per-node=*p* Number of cores per node

  n=number of N-core nodes *[default: 1 node]*
  p=how many cores(ntasks) per job or per node(ntasks-per-node) to use (1 through 128) *[default: 1 core]*

  Examples:
  --time=00:30:00
  --nodes=2
  --ntasks=256

  or

  --time=00:30:00
  --nodes=2
  --ntasks-per-node=128

  Memory: The compute nodes have at lest 256GB.

  Example:
  --time=00:30:00
  --nodes=2
  --ntask=256
  --mem=118000

  or

  --time=00:30:00
  --nodes=2
  --ntasks-per-node=64
  --mem-per-cpu=7375

squeue/scontrol/sinfo

Commands that display batch job and partition information .

SLURM EXAMPLE COMMAND	DESCRIPTION
squeue -a	List the status of all jobs on the system.
squeue -u $USER	List the status of all your jobs in the batch system.
squeue -j JobID	List nodes allocated to a running job in addition to basic information..
scontrol show job JobID	List detailed information on a particular job.
sinfo -a	List summary information on all the partition.

See the manual (man) pages for other available options.

Useful Batch Job Environment Variables

DESCRIPTION	SLURM ENVIRONMENT VARIABLE	DETAIL DESCRIPTION
JobID	$SLURM_JOB_ID	Job identifier assigned to the job
Job Submission Directory	$SLURM_SUBMIT_DIR	By default, jobs start in the directory that the job was submitted from. So the "cd $SLURM_SUBMIT_DIR" command is not needed.
Machine(node) list	$SLURM_NODELIST	variable name that contains the list of nodes assigned to the batch job
Array JobID	$SLURM_ARRAY_JOB_ID $SLURM_ARRAY_TASK_ID	each member of a job array is assigned a unique identifier

See the sbatch man page for additional environment variables available.

srun

The srun command initiates an interactive job on the compute nodes.

For example, the following command:

```
srun --time=00:30:00 --nodes=1 --ntasks-per-node=64 --pty /bin/bash
```

will run an interactive job in the default queue with a wall clock limit of 30 minutes, using one node and 16 cores per node. You can also use other sbatch options such as those documented above.

After you enter the command, you will have to wait for SLURM to start the job. As with any job, your interactive job will wait in the queue until the specified number of nodes is available. If you specify a small number of nodes for smaller amounts of time, the wait should be shorter because your job will backfill among larger jobs. You will see something like this:

```
srun: job 123456 queued and waiting for resources
```

Once the job starts, you will see:

```
srun: job 123456 has been allocated resources
```

and will be presented with an interactive shell prompt on the launch node. At this point, you can use the appropriate command to start your program.

When you are done with your runs, you can use the exit command to end the job.

scancel

The scancel command deletes a queued job or kills a running job.

- scancel JobID deletes/kills a job.

Visualization

Delta A40 nodes support NVIDIA raytracing hardware.

- describe visualization capabilities & software.
- how to establish VNC/DVC/remote desktop

Containers

Delta will support container use with Singularity.

NVIDIA NGC containers will be made available.

Protected Data (N/A)

IF APPLICABLE

- Describe the system’s capabilities for handling protected data.
- Data Retention Policies
- How to run jobs with protected data.
- Describe any mandated workflows.

Help

Describe how to get help.

Acknowledge

To acknowledge the NCSA Delta system in particular, please include the following

This research is part of the Delta research computing project, which is supported by the National Science Foundation (award OCI 2005572), and the State of Illinois. Delta is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications.

To include acknowledgement of XSEDE contributions to a publication or presentation please see https://portal.xsede.org/acknowledge and https://www.xsede.org/for-users/acknowledgement.

References

List any supporting documentation resources