Introduction to SeaWulf HPC for CSE416 students

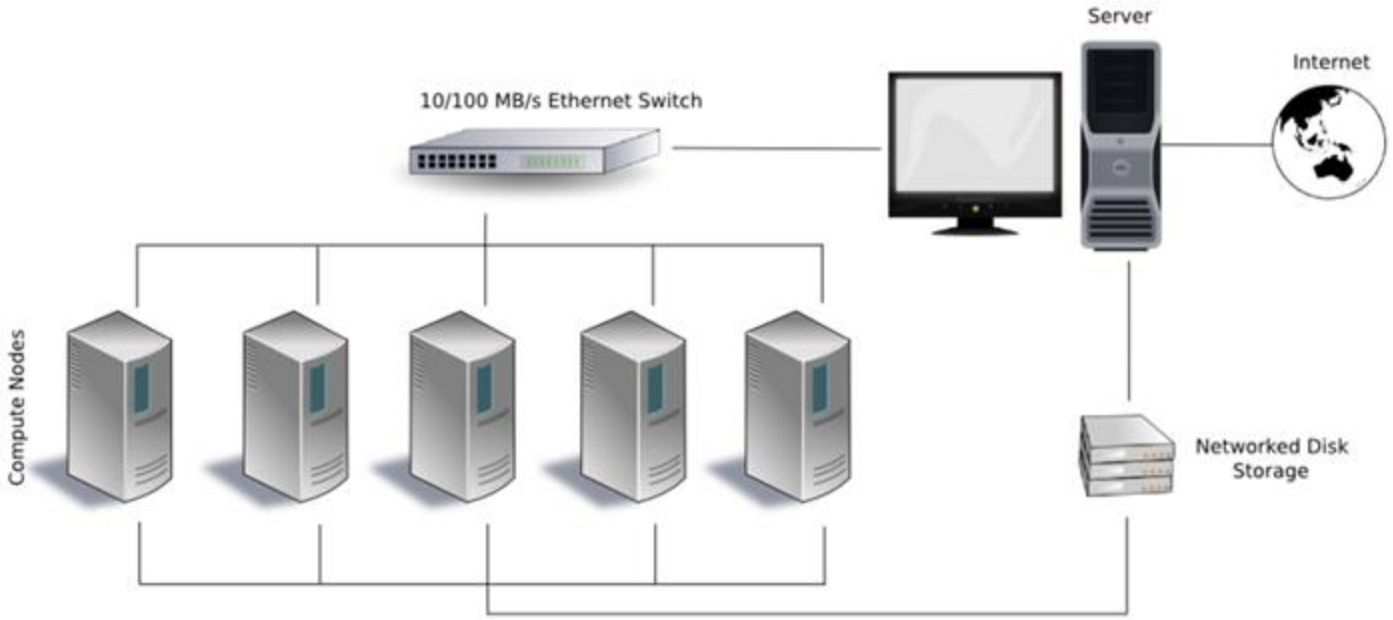
Dave Carlson

September 24nd, 2024

What's an HPC cluster, anyway?

A **High Performance Computing** (HPC) cluster contains multiple physically distinct computers ("nodes") that are connected over a network

A shared, parallel file system allows efficient access to the same data across all nodes



SeaWulf is …

- An HPC cluster dedicated to research applications for Stony Brook faculty, staff, and students

Available hardware:

- 5 **login nodes** = the entry points to the cluster

- 362 **CPU compute nodes** = where the work is done
  - 28 – 96 CPUS each
  - 128 GB – 1 TB RAM each

- 8 **GPU nodes** each with 4 Nvidia Tesla K80 GPUs
- 1 **GPU node** with 2 Tesla P100 GPUs
- 1 **GPU node** with 2 Tesla V100 GPUs
- 1 1 **GPU nodes** with 4 Tesla A100 GPUs
- Two **large memory nodes** each with 3 TB of RAM

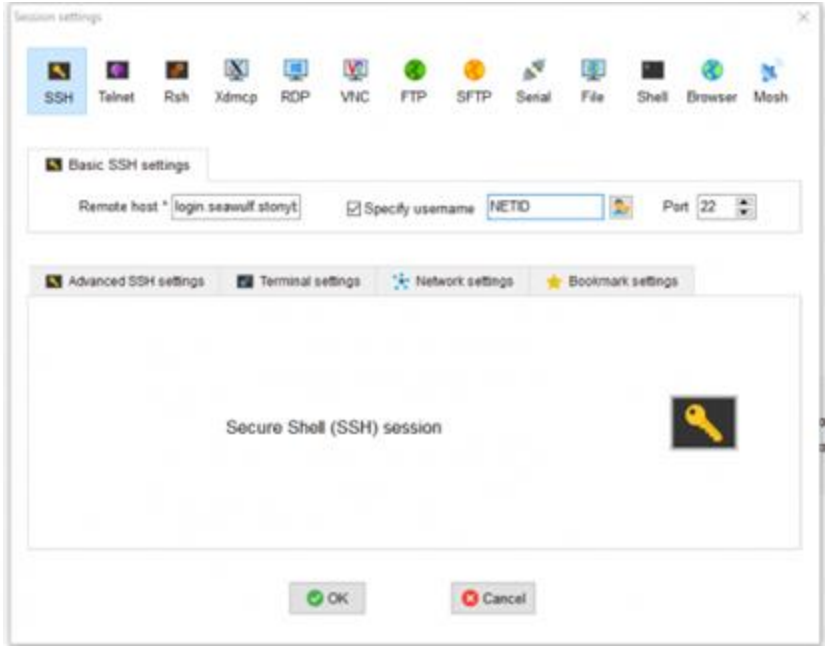
How do I connect to SeaWulf?

Mac & Linux users via the terminal:

ssh -X netid@**login**.seawulf.stonybrook.edu

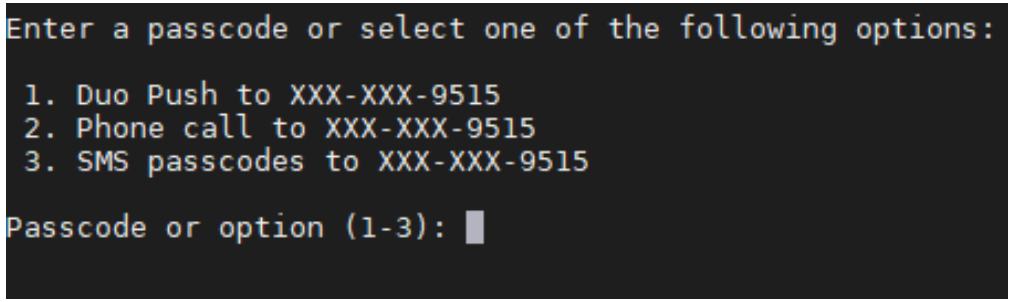
ssh -X netid@**milan**.seawulf.stonybrook.edu

Windows use



2-factor authentication with DUO

- Upon login, you will be prompted to receive and respond to a push notification, sms, or phone call:



- Multiple failures to respond can lead to temporary lockout of your account

- *DUO 2FA can be bypassed if connected to SBU's VPN (GlobalProtect)*

Which login nodes should I access?

login.seawulf.stonybrook.edu provides access to:

- ❑ 28-core nodes
- ❑ All GPU nodes except A100

milan.seawulf.stonybrook.edu & xeonmax.seawulf.stonybrook.edu provides access to:

- ❑ 40-core nodes
- ❑ 96-core nodes (AMD Milan)
- ❑ hbm-96-core nodes (Intel Sapphire Rapids)
- ❑ A100 GPU nodes

Access SeaWulf in your browser with Open OnDemand!

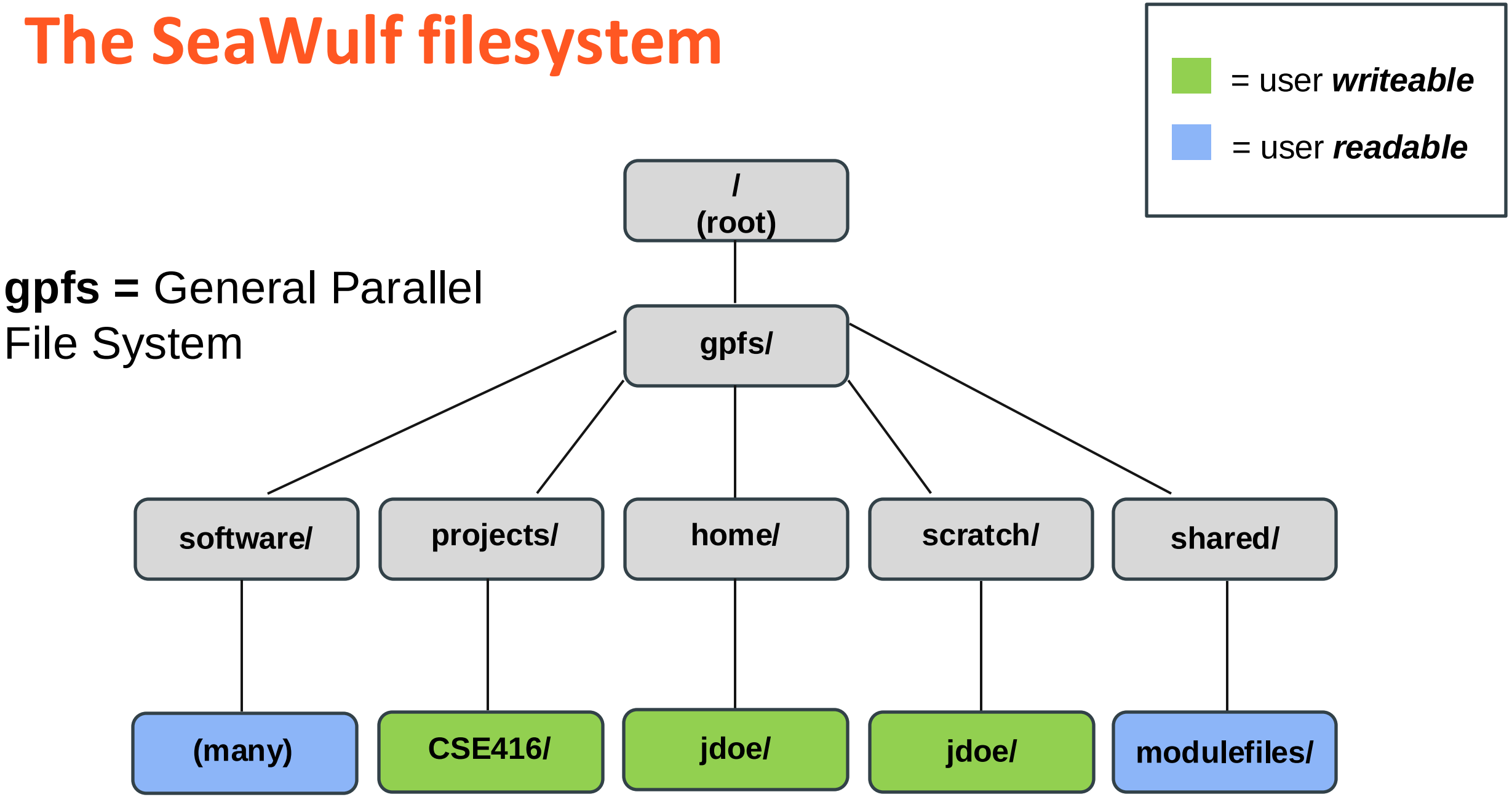
Point your browser to:

https://sn-ood.seawulf.stonybrook.edu/

See our OOD FAQ here:

https://it.stonybrook.edu/help/kb/accessing-seawulf-with-open-ondemand

The SeaWulf filesystem



Important paths to remember

- /gpfs/home/netid = **your home directory (20 GB)**

These environment variables also point to your home directory

$HOME

~

- /gpfs/scratch/netid = **your scratch directory (20 TB for housing temporary and intermediate files)**
- /gpfs/projects/CSE416 = **your project directory (5 TB shared space accessible to all class members)**

How do I transfer files onto SeaWulf?

Mac & Linux users should use scp (secure copy) to move files to and from SeaWulf

To transfer files from your computer to SeaWulf

1. Open terminal
2. scp /path/to/my/file netid@login.seawulf.stonybrook.edu:/path/to/destination/

To transfer files from SeaWulf to your computer

1. Open terminal
2. scp netid@login.seawulf.stonybrook.edu:/path/to/my/file /path/to/destination

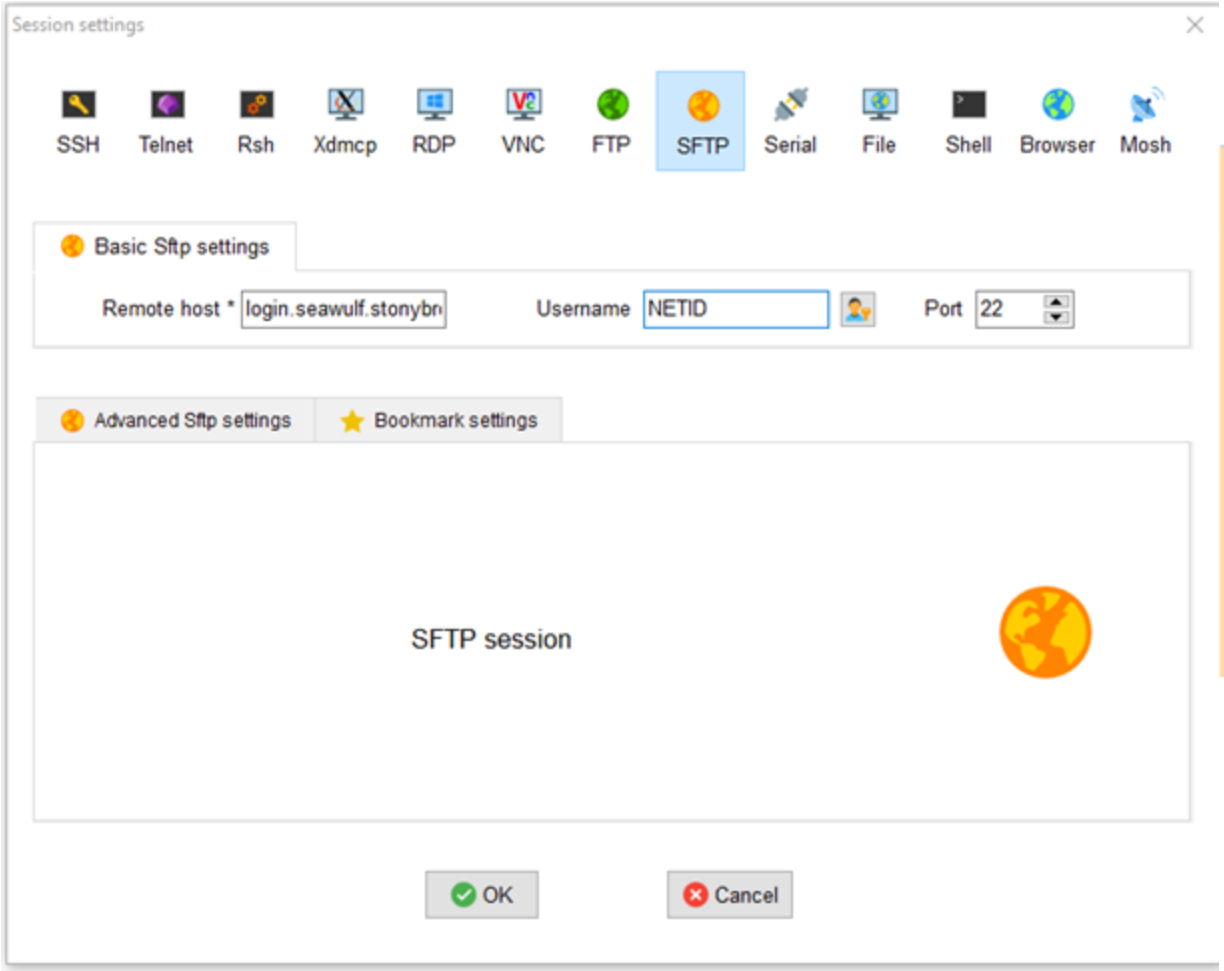
When possible, xfer archives (e.g. tarballs) or directories because:

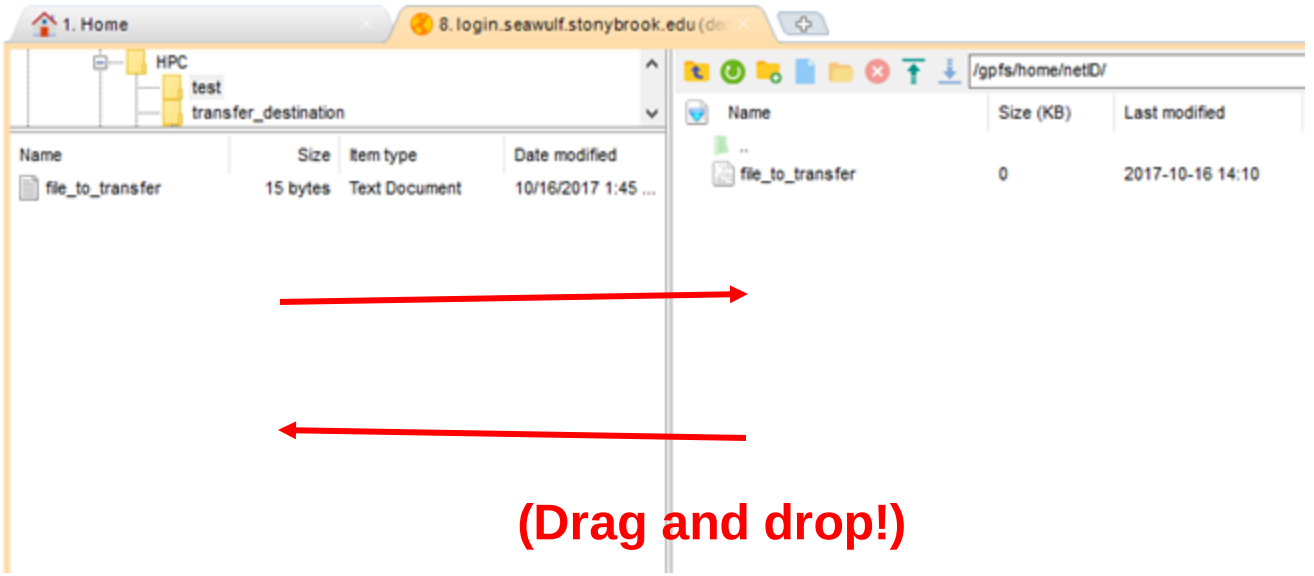
1. It's faster to transfer one large file than many small files

2. Unless connected to the SBU VPN, you may receive 1 DUO prompt for every scp command you run!

How do I transfer files onto SeaWulf?

Windows users:





(Drag and drop!)

Using the module system to access software

Useful module commands:

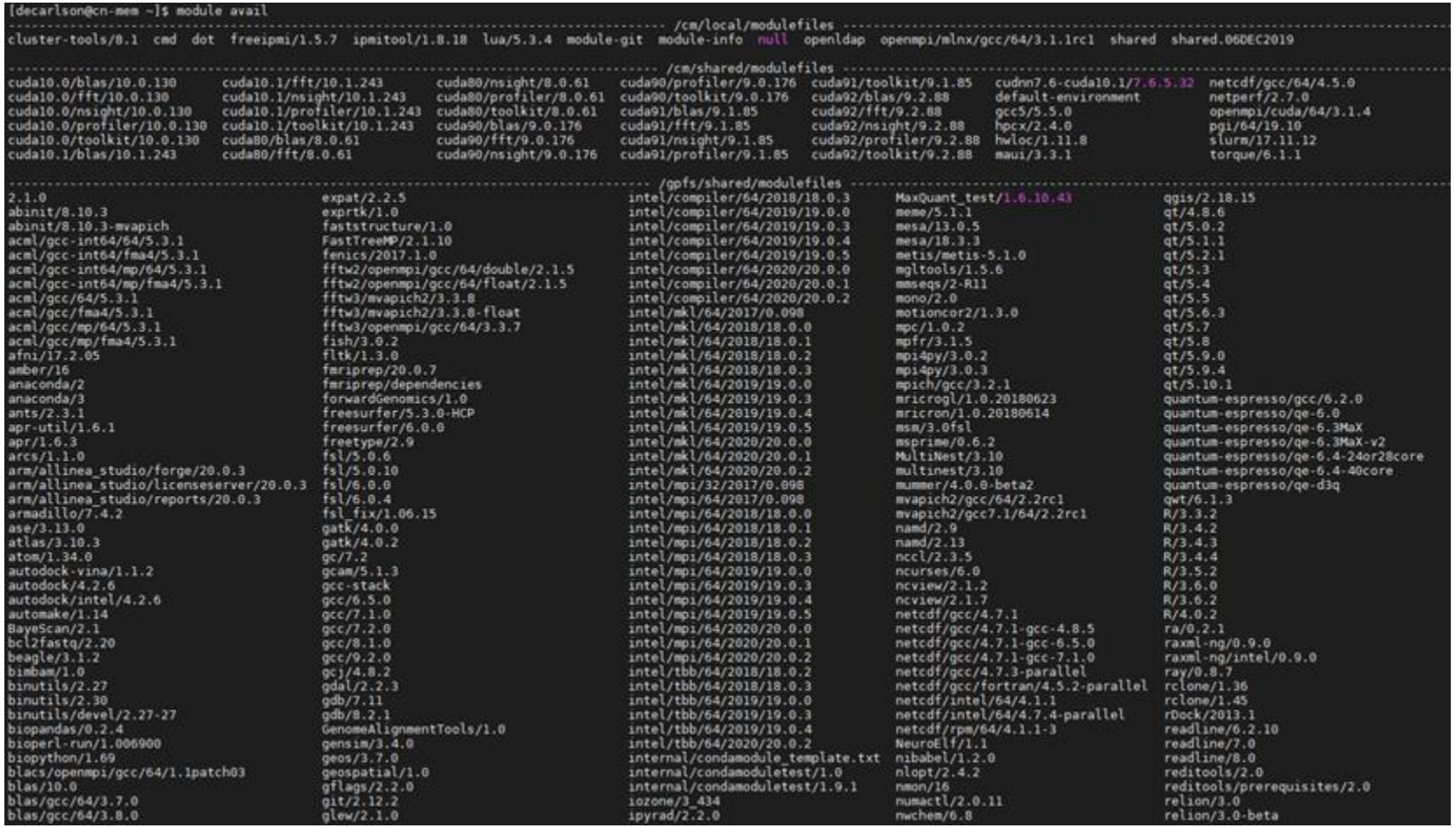
module avail

module load

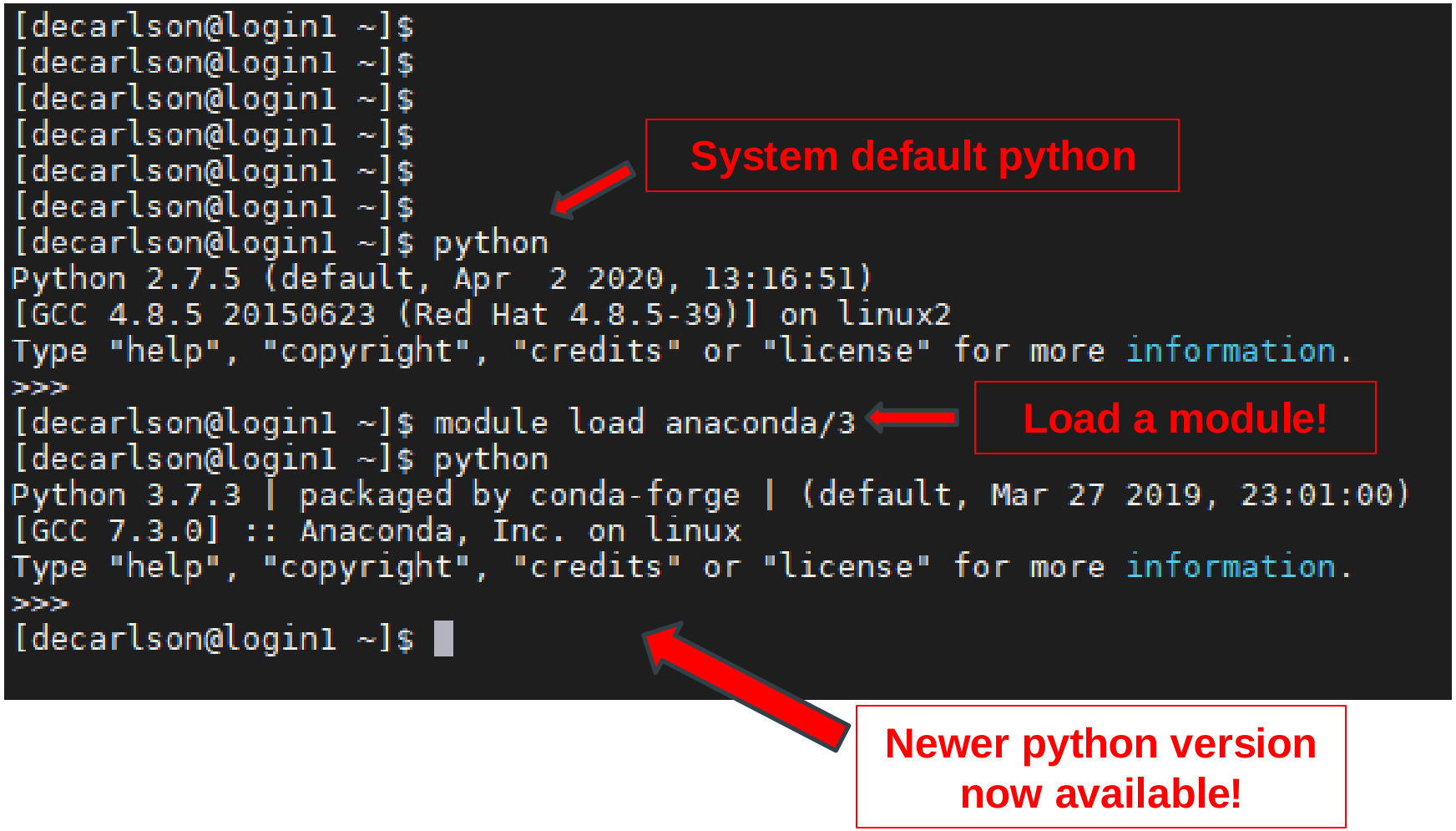
module list

module unload

module purge



Using the module system to access software



How do I do work on SeaWulf?

Computationally intensive jobs should *not* be done on the login node!

- To run a job, you must submit a batch script to the Slurm Workload Manager
- A batch script is a collection of bash commands issued to the scheduler, which which distributes your job across one or more compute nodes
- The user requests specific resources (nodes, cpus, job time, etc.), while the scheduler places the job into the queue until the resources are available

Example Slurm Job Script

All jobs submitted through a job scheduling system using scripts

```
#!/usr/bin/env bash

#SBATCH --nodes=1
#SBATCH --time=05:00
#SBATCH --partition=short-40core
#SBATCH --job-name=parallel_job
#SBATCH --output=squared_numbers.txt

# load required modules

module load anaconda/3
module load gnu-parallel/6.0

# parallelize the calculation across each input, running 40 processes at a time

parallel --jobs 40 python number_square.py {} ::: {1..100}
```

SBATCH Flags
- Specify # nodes, CPUs, running time, queue, and email options

Load modules – add programs to your path and set important environment variables

Execute your script or command

How do I execute my Slurm script?

Jobs are submitted via the Slurm Workload Manager using the "sbatch" command

```
[decarlson@login1 iqtree]$
[decarlson@login1 iqtree]$
[decarlson@login1 iqtree]$
[decarlson@login1 iqtree]$
[decarlson@login1 iqtree]$
[decarlson@login1 iqtree]$ module load slurm/17.11.12
[decarlson@login1 iqtree]$ sbatch gnu_parallel_example.slurm
Submitted batch job 356633
[decarlson@login1 iqtree]$
```

This is your job ID.

Useful Slurm commands

sbatch <script> = submit a job

scancel <job id> = cancel a job

squeue = get job status

sinfo = get info on node/queue status and utilization

(see the following for a full list of Slurm commands)

https://slurm.schedmd.com/archive/slurm-21.08.8/man_index.html

What queue should I submit to?

How many nodes do you need?

How many cores per node?

How much time do you need?

- ❑ **Only jobs using MPI should request more than 1 node!**
- ❑ **Use a "shared" queue if you don't need all the resources on a node**
- ❑ **There is often a tradeoff between resource usage and wait time!**
- ❑ **Don't wait until the last minute to submit jobs when you have a deadline!**

Queues accessed from `milan1` and `milan2`:

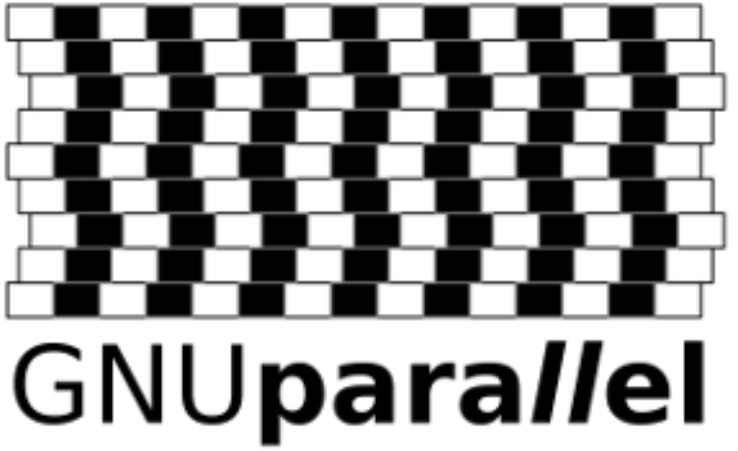
Queue	CPU Architecture	Vector/Matrix Extension	CPU Cores per Node	GPUs per Node	Node Memory[1]	Default Runtime	Max Runtime	Max Nodes	Min Nodes	Max Simultaneous Jobs per User	Multiple Users per Node
debug-40core	Intel Skylake	AVX512	40	0	192 GB	1 hour	1 hour	8	n/a	n/a	No
short-40core	Intel Skylake	AVX512	40	0	192 GB	1 hour	4 hours	8	n/a	4	No
short-40core-shared	Intel Skylake	AVX512	40	0	192 GB	1 hour	4 hours	4	n/a	n/a	Yes
medium-40core	Intel Skylake	AVX512	40	0	192 GB	4 hours	12 hours	16	6	1	No
long-40core	Intel Skylake	AVX512	40	0	192 GB	8 hours	48 hours	6	n/a	3	No
long-40core-shared	Intel Skylake	AVX512	40	0	192 GB	8 hours	24 hours	3	n/a	n/a	Yes
extended-40core	Intel Skylake	AVX512	40	0	192 GB	8 hours	7 days	2	n/a	3	No
extended-40core-shared	Intel Skylake	AVX512	40	0	192 GB	8 hours	3.5 days	1	n/a	n/a	Yes

See our FAQ page on SeaWulf's queues

Parallel processing on the cluster

- **Parallelization *within a single compute node***
  - Lots of ways of doing this
  - Some tasks easily parallelized with scripting (e.g, "Embarrassingly Parallel" tasks)
  - Language-specific options (e.g., Python's Multiprocessing library)
  - OpenMP for multithreaded C/C++/Fortran tasks
- **Parallelization across *multiple nodes***
  - No communication – Slurm Array
  - Communication between processes needed – Requires the use of MPI

Parallel processing on a single node with GNU Parallel



- Perfect for "embarrassingly parallel" situations
- Available as a module: gnu-parallel/6.0
- Can easily take in a series of inputs (e.g., files or values) and run a command on each input simultaneously
- Lots of tutorials and resources available on the web!

https://www.gnu.org/software/parallel/parallel_tutorial.html (thorough!!)
https://www.msi.umn.edu/support/faq/how-can-i-use-gnu-parallel-run-lot-commands-parallel (many practical examples)

Parallel processing with GNU Parallel

```
#!/usr/bin/env bash

#SBATCH --nodes=1
#SBATCH --time=05:00
#SBATCH --partition=short-40core
#SBATCH --job-name=parallel_job
#SBATCH --output=squared_numbers.txt

# load required modules

module load anaconda/3
module load gnu-parallel/6.0

# parallelize the calculation across each input, running 40 processes at a time

parallel --jobs 40 python number_square.py {} ::: {1..100}
```

Optional flags to control behavior

This command will be run on each input

The input values to parallelize over

{} = placeholder for each input

Parallel processing on a single node with Python's multiprocessing library


```
#!/usr/bin/env python

import numpy as np
import multiprocessing as mp

# define a function for the calculation
def square_me(num):
        result = int(np.square(num))
        return(result)

# spawn a pool of parallel workers

p = mp.Pool(processes=40)

# using a pool of 40 parallel workers, loop over the values of 1 through 100 and apply the math function to each
for num in range(1,101):
        results = p.apply_async(square_me, [num])
        print(f'The square of {num} is {results.get()}')


# close the pool of workers
p.close()
```

Brief introduction to parallelization options for python:

https://www.anyscale.com/blog/parallelizing-python-code

Multithreading on a single node with OpenMP

OpenMP

- Framework for parallelization (multithreading) in a shared-memory (single node) context for C, C++, and Fortran
- Typically involves creating multiple threads within a single process instead of spawning multiple processes
- Useful when communication between parallel tasks is required
- Implemented in most modern compilers (GCC, Intel, LLVM, etc.)
- Resource usage controlled at runtime by environment

Check out this comprehensive tutorial!

Multithreading on a single node with OpenMP

Estimating pi with C and OpenMP

```c
#include <stdio.h>
#include <time.h>
#include <omp.h>
#include <stdint.h>

#define NPTS 1000000000

void main() {
   uint64_t i;
   double a,b,c,pi,dt,mflops;
   struct timespec tstart,tend;
   clock_gettime(CLOCK_REALTIME,&tstart);
   a = 0.5;
   b = 0.75;
   c = 0.25;
   pi = 0;
   #pragma omp parallel for reduction(+:pi)
   for (i = 1; i <= NPTS; ++i)
      pi += a/((i-b)*(i-c));
   clock_gettime(CLOCK_REALTIME,&tend);
   dt = (tend.tv_sec+tend.tv_nsec/1e9)-(tstart.tv_sec+tstart.tv_nsec/1e9);
   mflops = NPTS*5.0/(dt*1e6);
   printf("NPTS = %ld, pi = %f, threads = %d\n",NPTS,pi,omp_get_max_threads());
   printf("time = %f, estimated MFlops = %f\n",dt,mflops);
   }
```

```
#!/usr/bin/env bash

#SBATCH --job-name=openmp_pi
#SBATCH --output=openmp_pi.log
#SBATCH --nodes=1
#SBATCH --time=05:00
#SBATCH -p short-40core

# load a gcc module
module load gcc/12.1.0

# Environment variable to set how many threads we'll be using

export OMP_NUM_THREADS=40

# compile the code
gcc /gpfs/projects/samples/pi/openmp_pi.c -o openmp_pi -fopenmp

# execute the code
./openmp_pi
```

Load a compiler module

Specify # of threads

Compile w/ –fopenmp

Can I use multiple nodes for a single job?

MVAPICH

Yes! (...well...maybe)

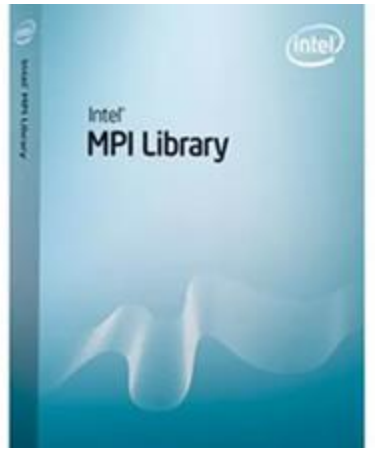
Message Passing Interface (MPI) facilitates communication between processes within or among nodes

Multiple "flavors" of MPI are available on SeaWulf

- **Mvapich*, Intel* mpich, OpenMPI**

*=officially supported

Open MPI



Can I use multiple nodes for a single job?

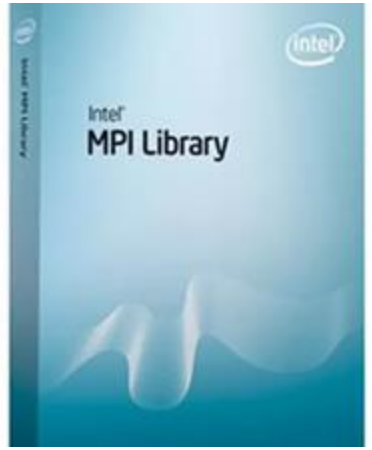
MVAPICH

To use MPI:

- Write code with MPI functions
- Compile with MPI wrapper

Language	GCC command	Mvapich wrapper
C	gcc	mpicc
C++	g++	mpicxx
Fortran	gfortran	mpif90

Open MPI



- Specify resource requirements in your Slurm script
- Execute code with mpiexec or mpirun

Example MPI Job Submission Script

```
#!/usr/bin/env bash

#SBATCH --job-name=mpi_pi
#SBATCH --output=mpi_pi.log
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=28
#SBATCH --time=05:00
#SBATCH -p short-28core

# load a gcc module
module load mvapich2/gcc12.1/2.3.7

# set env variables which may help performance

export MV2_HOMOGENEOUS_CLUSTER=1
export MV2_ENABLE_AFFINITY=0

#export HWLOC_COMPONENTS=-gl

# compile the code with the mpi compiler wrapper
mpicc /gpfs/projects/samples/pi/mpi_pi.c -o mpi_pi

# execute the code with MPI
mpirun ./mpi_pi
```

Specify resource usage (nodes and MPI tasks)

Load an MPI module

Compile with MPI

Execute with MPI

Estimating pi with C and MPI

Parallelization FAQS:

Part 1: embarrassingly parallel tasks

Part 2: OpenMP & MPI

MPI Hello World

```
#!/bin/bash

#SBATCH --job-name=gcc_mpi_hello
#SBATCH --output=gcc_mpi_hello.log
#SBATCH --ntasks-per-node=4
#SBATCH --nodes=2
#SBATCH --time=05:00
#SBATCH -p short-40core


module load mvapich2/gcc12.1/2.3.7

mpirun ./gcc_mpi_hello
```

mpirun

“Hello from process 1 on node 1”
“Hello from process 2 on node 1”
“Hello from process 3 on node 1”
“Hello from process 4 on node 1”

“Hello from process 1 on node 2”
“Hello from process 2 on node 2”
“Hello from process 3 on node 2”
“Hello from process 4 on node 2”

Need to troubleshoot? Use an interactive job!

Example:

```
[decarlson@dg-mem ~]$ srun -N 1 -p short-40core --pty bash
srun: job 293816 queued and waiting for resources
srun: job 293816 has been allocated resources
[decarlson@dn029 ~]$
```

”srun”: allocate a compute node in the short-40core queue

“--pty bash” run the bash shell on the compute node

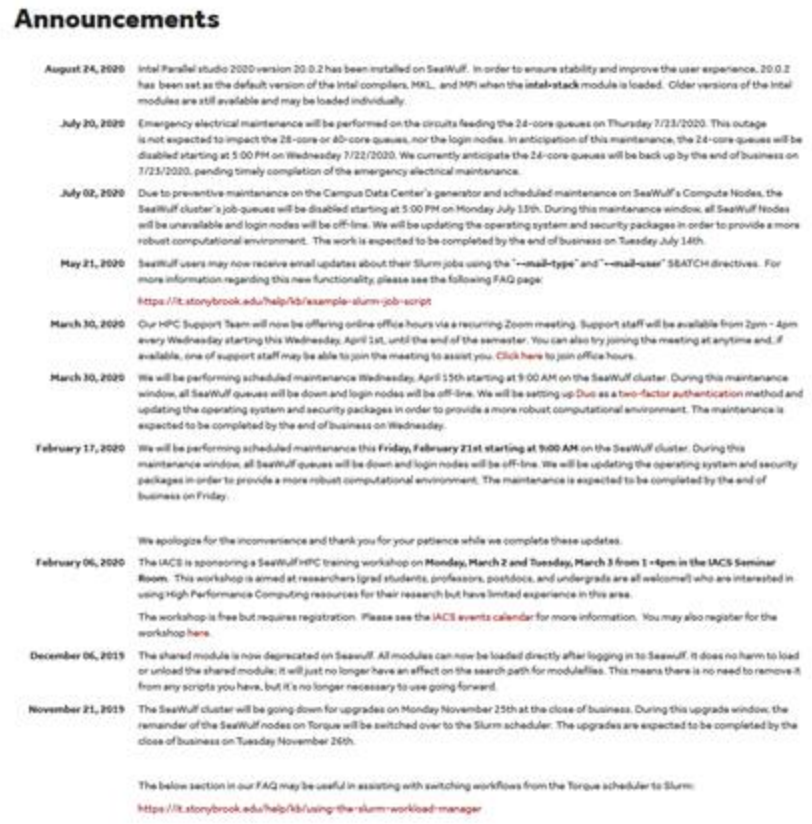
Once a node is available, you can issue commands on the command line

Good for troubleshooting,

Inefficient once your code is working

Need more help or information?

Check out our FAQ: https://it.stonybrook.edu/services/high-performance-computing



Announcements

August 24, 2020 Intel Parallel studio 2020 version 20.0.2 has been installed on SeaWulf. In order to ensure stability and improve the user experience, 20.0.2 has been set as the default version of the Intel compilers, MKL, and MPI when the **intel+stack** module is loaded. Older versions of the Intel modules are still available and may be loaded individually.

July 20, 2020 Emergency electrical maintenance will be performed on the circuits feeding the 24-core queues on Thursday 7/23/2020. This outage is not expected to impact the 28-core or 40-core queues, nor the login nodes. In anticipation of this maintenance, the 24-core queues will be disabled starting at 5:00 PM on Wednesday 7/22/2020. We currently anticipate the 24-core queues will be back up by the end of business on 7/23/2020, pending timely completion of the emergency electrical maintenance.

July 02, 2020 Due to preventive maintenance on the Campus Data Center's generator and scheduled maintenance on SeaWulf's Compute Nodes, the SeaWulf cluster's job queues will be disabled starting at 5:00 PM on Monday July 13th. During this maintenance window, all SeaWulf Nodes will be unavailable and login nodes will be off-line. We will be updating the operating system and security packages in order to provide a more robust computational environment. The work is expected to be completed by the end of business on Tuesday July 14th.

May 21, 2020 SeaWulf users may now receive email updates about their Slurm jobs using the "**--mail-type**" and "**--mail-user**" SBATCH directives. For more information regarding this new functionality, please see the following FAQ page:

https://it.stonybrook.edu/help/kb/example-slurm-job-script

March 30, 2020 Our HPC Support Team will now be offering online office hours via a recurring Zoom meeting. Support staff will be available from 2pm - 4pm every Wednesday starting this Wednesday, April 1st, until the end of the semester. You can also try joining the meeting at anytime and, if available, one of support staff may be able to join the meeting to assist you. Click here to join office hours.

March 30, 2020 We will be performing scheduled maintenance Wednesday, April 15th starting at 9:00 AM on the SeaWulf cluster. During this maintenance window, all SeaWulf queues will be down and login nodes will be off-line. We will be setting up Duo as a two-factor authentication method and updating the operating system and security packages in order to provide a more robust computational environment. The maintenance is expected to be completed by the end of business on Wednesday.

February 17, 2020 We will be performing scheduled maintenance this **Friday, February 21st starting at 9:00 AM** on the SeaWulf cluster. During this maintenance window, all SeaWulf queues will be down and login nodes will be off-line. We will be updating the operating system and security packages in order to provide a more robust computational environment. The maintenance is expected to be completed by the end of business on Friday.

We apologize for the inconvenience and thank you for your patience while we complete these updates.

February 06, 2020 The IACS is sponsoring a SeaWulf HPC training workshop on **Monday, March 2 and Tuesday, March 3 from 1 -4pm in the IACS Seminar Room**. This workshop is aimed at researchers (grad students, professors, postdocs, and undergrads are all welcome!) who are interested in using High Performance Computing resources for their research but have limited experience in this area.

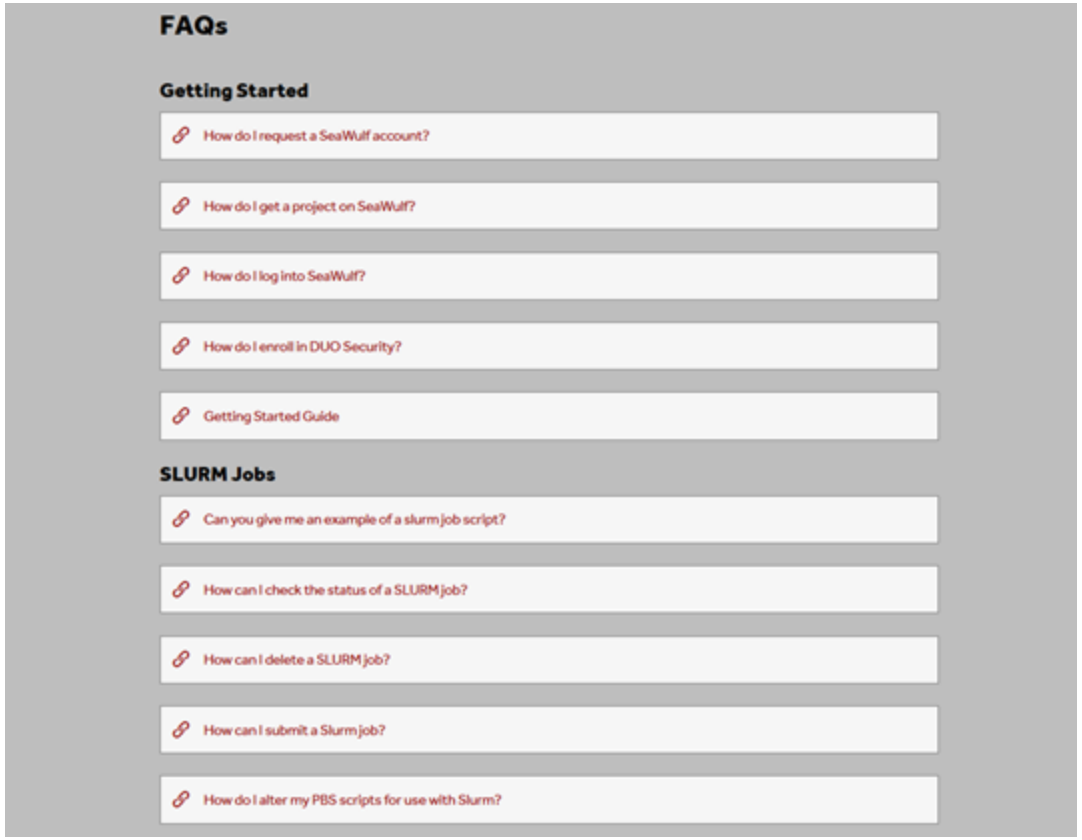
The workshop is free but requires registration. Please see the IACS events calendar for more information. You may also register for the workshop here.

December 06, 2019 The shared module is now deprecated on Seawulf. All modules can now be loaded directly after logging in to Seawulf. It does no harm to load or unload the shared module; it will just no longer have an effect on the search path for modulefiles. This means there is no need to remove it from any scripts you have, but it's no longer necessary to use going forward.

November 21, 2019 The SeaWulf cluster will be going down for upgrades on Monday November 25th at the close of business. During this upgrade window, the remainder of the SeaWulf nodes on Torque will be switched over to the Slurm scheduler. The upgrades are expected to be completed by the close of business on Tuesday November 26th.

The below section in our FAQ may be useful in assisting with switching workflows from the Torque scheduler to Slurm:

https://it.stonybrook.edu/help/kb/using-the-slurm-workload-manager



FAQs

Getting Started

- How do I request a SeaWulf account?
- How do I get a project on SeaWulf?
- How do I log into SeaWulf?
- How do I enroll in DUO Security?
- Getting Started Guide

SLURM Jobs

- Can you give me an example of a slurm job script?
- How can I check the status of a SLURM job?
- How can I delete a SLURM job?
- How can I submit a Slurm job?
- How do I alter my PBS scripts for use with Slurm?

Submit a ticket: https://iacs.supportsystem.com


QUESTIONS?