Documentation and User Manual

For CentOS 6 nodes and Moab/Torque scheduling system only

The HPCC has migrated to a new operating system and scheduler. For more information, see "Introduction to new 2018 HPC systems".

For the new 2018 system, please refer to the documentation here.

This documentation provides an overview of the features and capabilities of the Michigan State University High Performance Computing Center. The details of the hardware and software available to users are listed here, as well as instructions on how to operate in a supercomputing environment.

- System Overview
  - Overview
  - System Configuration
  - File Systems
- System Access
- Computing Environment
  - Unix Shell
  - Environment Variables
  - Startup Scripts
  - Modules
- Transferring your Files to Machine
- Application Development
  - Programming Models
  - Compiling
  - Libraries
  - Debugging
- Running your Applications
  - Introduction
  - Job Scheduler
    - Serial Job Script
    - Shared Memory Job Script
    - Distributed Memory Job Script
  - Job Control
- Support
- Acknowledgements (Citing iCER in publications)

System Overview

Overview

The HPCC maintains seven clusters, along with additional testing nodes. We feature a single queue system where a user only has to specify job requirements. The scheduler assigns the job to an appropriate cluster.

System Configuration

Each cluster below runs the Community Enterprise Operating System (CentOS) 6.6, each with a different combination of memory and interconnect. The cluster uses TORQUE for resource management, and Moab for job management.

Cluster	Nodes	Processors (per node)	Cores (per node)	Memory (per node)	Interconnect	Local Disk	TFLOPS
intel11	2	eight 2.66 GHz Oct-core Intel Xeon E8837	64	2 TB	QDR Infiniband	292 GB	2.6
	1	four 2.66 GHz Oct-core Intel Xeon E8837	32	1 TB	QDR Infiniband	146 GB	.7
	2	four 2.66 GHz Oct-core Intel Xeon E8837	32	512 GB	QDR Infiniband	146 GB	1.3
intel14	128	Two 2.5Ghz 10-core Intel Xeon E5-2670v2	20	64 GB	FDR Infiniband	500 GB	51
	24	Two 2.5Ghz 10-core Intel Xeon E5-2670v2	20	256 GB	FDR Infiniband	500 GB	9.6
intel14-k20	40	Two 2.5Ghz 10-core Intel Xeon E5-2670v2, and two Nvidia Tesla K20 GPUs	20 (cpu) 4992 stream processors	128 GB (host) 10 GB (gpu)	FDR Infiniband	500 GB	108
intel14-phi	28	Two 2.5Ghz 10-core Intel Xeon E5-2670v2, and two Intel Xeon Phi 5110P (MIC)	20 (cpu) 120 (phi)	128 GB (host) 16 GB (phi)	FDR Infiniband	500 GB	67.2
intel14-xl	2	Four Intel Xeon CPU E7-8857 v2 @ 3.00GHz	48	1.5 TB	FDR Infiniband	854 GB	2.2
	1	Four Intel Xeon CPU E7-8857 v2 @ 3.00GHz	48	1 TB	FDR Infiniband	1.8 TB	1.1
	1	Four Intel Xeon CPU E7-8857 v2 @ 3.00GHz	48	3 TB	FDR Infiniband	1.1 TB	1.1
	1	Four Intel Xeon CPU E7-8857 v2 @ 3.00GHz	48	1.5 TB	FDR Infiniband	1.8 TB	1.1
	1	Eight Intel Xeon CPU E7-8857 v2 @ 3.00GHz	96	6 TB	FDR Infiniband	854 GB	2.2

laconia (intel16)	395	Two 2.4Ghz 14-core Intel Xeon E5-2680v4	28	128 GB (x290) 256 GB (x24) 512 GB (x6)	FDR/EDR Infiniband	240 GB SSD	395
intel16-xl	2	Four Intel Xeon 16-core E7-8867 v3 @ 2.50GHz	64	3 TB	EDR Infiniband	880 GB SSD	5.1
	1	Eight Intel Xeon 18-core E7-8867 v4 @ 2.40GHz	144	6 TB	EDR Infiniband	4 TB NVMe	5.5
intel16-k80	50	Two 2.4Ghz 14-core Intel Xeon E5-2680v4, and four Nvidia Tesla K80 GPUs	28 (cpu) 19968 stream processors	256 GB (host) 96 GB (gpu)	FDR/EDR Infiniband	240 GB SSD	375

A visual representation of the per-node CPU and memory layout of the different types of nodes available can be found in this wiki document.

File Systems

A shared disk file system provides direct disk access from login, development and compute nodes of the cluster. HPCC uses ZFS on enterprise-class disks and servers, with 670TB total of usable capacity for providing global home and research directories, and a high speed Lustre filesystem with 1.9PB for scratch storage.

Class	Path	Quota	Backup Schedule	Purge Policy
Home Directories	/mnt/home	50 GB - 1TB (or more)	daily	none
Research Directories	/mnt/research	50 GB - 1TB (or more)	daily	none
Scratch Directory	/mnt/scratch	1 million files	none	after 45 days
Small-IO Scratch Directory	/mnt/ffs17	100 GB	none	none
Local Disk	/mnt/local	as available on node	none	after 8 days

Personal data should be stored in /mnt/home/[NetID]. Here, each user has their own 50 GB of space (more available upon request), which is backed up daily; a zfs snapshot is also taken hourly. A 1 TB block of space to be shared among members of a research group may be obtained if the group can provide adequate justification. Additional space may also be purchased. Files on this system, located at /mnt/research/[groupname], are also backed up daily. Both /mnt/home and /mnt/research have compression enabled. This allows better performance and enables significant space savings. The /mnt /scratch file system consists of 1.9 PB of high-speed, distributed space, shared among all users. A user should store files here when a job requires those files to be accessible from all nodes during a computation. This space is not intended for long-term storage, and is thus not backed up. Additionally, any files in /mnt/scratch that are older than 45 days are flagged for automatic deletion.

⚠ if you use the rsync protocol to copy files to scratch, original file creation times are often preserved; such files are also subject to the purge policy

If a user has files needed by multiple jobs running on the same node, those files should be placed in the /mnt/local file system.

research spaces are only mounted on demand. For example,

```
cd /mnt/research/test
```

will mount the research space test before changing your current working directory to /mnt/research/test. The command

```
ls /mnt/research
```

will only show mounted file systems.

System Access

To obtain (free) HPCC account(s), a MSU faculty member must request accounts by filling out the form at http://www.hpcc.msu.edu/request. Information required to complete the form includes

- a list of names and NetIDs of research group members requiring accounts
- a statement on whether or not export controlled software or data will be used
- a project abstract (several paragraphs).

By applying for accounts, the Principal Investigator is agreeing that all group members will abide by the Statement of Acceptable Use.

The cluster is primarily accessed by means of the Secure Shell (SSH) network protocol. An SSH connection can be opened to hpcc.msu.edu from a terminal prompt (Mac/Linux) or a program like PuTTY (Windows). Please review the walkthrough or a video tutorial for instructions on connecting to HPCC. Optional instructions for establishing SSH key-based authentication and directions for configuring an X-server on Windows (for graphical applications) are also available.

Note: accounts created must be done so with an active MSU NetID and they must have a msu email associated with their NetID. Please ensure the requested NetID is searchable to expedite this process. Please also keep in mind that the users who have their information hidden may not be discoverable, and may wish to allow this information to be searched to allow a faster setup. You may confirm visibility of the requested NetIDs at the following link:

https://search.msu.edu/people/index.php

External partners should contact their home institution's support person for their organization's process to request accounts on MSU's HPCC.

Computing Environment

Unix Shell

A Unix/Linux shell is a command-line interpreter which provides a user interface for the Unix/Linux operating system. Users control the operation of a computer by submitting single commands or by submitting one or more commands via a shell script. Several common shell choices are available on HPCC:

bash	a Bourne-shell (sh) compatible shell with many newer advanced features as well
tcsh	an advanced variant on csh with all the features of modern shells
zsh	an advanced shell which incorprates all the functionality of bash, tcsh, and ksh combined
csh	the original C-style shell

The default shell provided to HPCC users is the bash shell. To change your shell, please contact HPCC support staff. To discover your current shell:

```
echo $SHELL
```

Environment Variables

Environment variables are a set of dynamically named values which can control the way running processes will behave on a computer. Many of the UNIX commands and tools require certain environment variables to be set. Many of these are set automatically for the users when they log in or load applications via the module command. To view your current set of environment variables do the following command:

```
env
```

To assign a new value to an environment variable in either bash or zsh:

```
export <name>=<value>
```

To assign a new value to an environment variable in either tcsh or csh:

```
setenv <name> <value>
```

To print the value of a variable:

```
echo $<name>
```

Startup Scripts

A startup script is a shell script which the login process executes. It provides an opportunity to alter your environment. You are free to setup your own startup scripts but be careful to make sure they are set up correctly for both interactive and batch access or it may negatively affect your ability to log in or run batch jobs on the system:

bash	~/.bashrc
tcsh	~/.chsrc
zsh	~/.zshrc
csh	~/.cshrc

Modules

A module manages environment variables needed to load a particular piece of software. The login process loads a few modules by default for HPCC users. The modules include the GNU compilers, the OpenMPI communication library, MATLAB and R.

To see a list of modules that are currently loaded:

```
module list
```

To see a list of modules that are available to be loaded:

```
module avail
```

To see what environment variables would be set/changed if you load a specific module:

```
module show <module_name>
```

To load a module:

```
module load <module_name>
```

To unload a module:

```
module unload <module_name>
```

To swap a module:

```
module swap GNU Intel
```

To load a modulefile in a non-standard directory:

```
module use <path_to_module>
module load <modulename>
```

To find out if a module is available:

```
module spider <software_name>
```

Click here for a complete list of available modules and module names.

Transferring your Files to Machine

Copying files to and from one's HPCC file space can be done in a multitude of ways. The wiki page on Transferring Files to the HPCC describes the use of the FileZilla GUI, Dropbox, and several Unix commands. The page on Documentation and User Manual explains how a user can mount his/her HPCC home directory on a personal computer, with instructions for Windows, Mac OS X, and Ubuntu operating systems.

Documentation and User Manual

Application Development

Programming Models

A programming model is an abstraction of a computer system. For example, the "von Neumann model" is a model used in traditional sequential computers. For parallel computing, there are models typically reflecting different ways processors can be interconnected. The most common are based on shared memory, distributed memory, and a hybrid of the two. Some terminology:

- A serial program is a single process which executes as a sequential stream of instructions on one computer.
- A shared-memory program is a single process that takes advantage of a multi-core processor and its shared memory to achieve a form of parallel computing called multithreading. Open Multi-Processing (OpenMP) is a specific implementation of the shared-memory model and is a collection of parallelization directives, library routines, and environment variables. It distributes the work of a process over several cores of a multi-core processor.
- A message-passing program is a set of processes (often multiple copies of a single process) that take advantage of distributed-memory systems by communicating with each other via the sending and receiving of messages. The Message-Passing Interface (MPI) is a specific implementation of the message-passing model and is a collection of library functions. MPICH2 and OpenMPI are two implementations of the MPI-2 standard.
- A CUDA program is a single process that uses the CUDA language to take advantage of graphics accelerators .
- A hybrid program combines shared-memory and/or message-passing and/or CUDA attributes to take advantage of compute clusters with multi-core or Gpu empowered compute nodes.

All five types of programming models are available at HPCC depending on the type of code and required performance. Any use of OpenMP (threaded) only code types would need to be restricted to a single node to work correctly (e.g. nodes=1:ppn=4).

Compiling

Available Compilers are Fortran 77, Fortran 90, Fortran 95, C, C++ and CUDA. The compilers can produce general-purpose and architecture-specific optimizations to improve performance. These include loop-level optimizations, inter-procedural analysis and cache optimizations. The compilers support automatic and user-directed parallelization of Fortran, C, C++ and CUDA applications for multiprocessing execution. The compilers available include the GNU (default), Intel, and PGI compiler sets.

To use the Intel compilers, type

```
module unload GNU OpenMPI
module load Intel OpenMPI
```

Libraries

The Intel Math Kernel Library (MKL) is available on HPCC, and contains ScaLAPACK, LAPACK, Sparse Solver, BLAS, Sparse BLAS, CBLAS, GMP, FFTs, DFTs, VSL, VML, and Interval Arithmetic routines. MKL is loaded by default. Some useful libraries that can be linked against include:

library	linking flags
libmkl_blacs_openmpi_lp64.a	-lmkl_blacs_openmpi_lp64
libmkl_gnu_thread.a	-lmkl_gnu_thread
libmkl_scalapack_lp64.a	-lmkl_scalapack_lp64
libmkl_lapack95_lp64	-lmkl_lapack95_lp64
libmkl_solve_lp64.a	-lmkl_lp64_solver

Debugging

There are various tools available for debugging a program on HPCC. We recommend using TotalView, an advanced debugger for all parallel (and serial) programs. To use, compile your code with a "-g" flag (or "-g -G" flags if it is a CUDA based program), and then type

```
totalview ./programname -a <arguments>
```

Alternatively, just load totalview

```
totalview
```

and use the user interface to load programs you wish to debug and setup parameters.

Running your Applications

Introduction

HPCC has several development nodes that are available for users to compile their code, and do short (< 2 hour) runs to estimate run-time and memory usage. These development nodes can be accessed once the user is logged into hpcc.msu.edu

name	Processors	Cores	Memory	accelerators
dev-intel14	Two 2.5Ghz 10-core Intel Xeon E5-2670v2	20	256GB	none
dev-intel14-k20	Two 2.5Ghz 10-core Intel Xeon E5-2670v2	20	128GB	2x Nvidia Tesla K20 GPUs

dev-intel14-phi	Two 2.5Ghz 10-core Intel Xeon E5-2670v2	20	128GB	2x Xeon Phi 5110P accelerators
dev-intel16	Two 2.4Ghz 14-core Intel Xeon E5-2680v4	28	128GB	none
dev-intel16-k80	Two 2.4Ghz 14-core Intel Xeon E5-2680v4, and four Nvidia Tesla K80 GPUs	28	256GB	4x Nvidia Tesla K80 GPUs

Once the code is compiled, tested, and resource usage estimated, users MUST submit a job to the queue, specifying various constraints such as
job duration, memory usage, number of CPUs, software license reservations. HPCC uses Torque which offers job control such as submitting, monitoring, and deleting a job.

Additionally, HPC also has evaluation nodes that can be used for development jobs:

name	Processors	Cores	Memory	accelerators
eval-k10	Two oct-core Intel E5-2620	16	64 GB	single Nvidia K10 accelerator

Job Scheduler

At HPCC, jobs run in a queue based on their priority settings as long as the resources are available to run the job. If the resources are not available, the next job in the queue that can be run and not negatively impact the expected run time of the jobs before them will attempt to run (i.e., backfill mode enabled). We encourage users to properly estimate the amount of resources required (specifically walltime and memory) to optimize the start time of their jobs.

To submit a job to the cluster, we suggest that users create a job script which contains commands to be run. There are three main types of jobs that can be run on the cluster. Serial jobs (where the program being run does not have any built-in parallelism), shared memory jobs (where the program is able to leverage multiple cores on a single machine), and distributed memory jobs (where the program is able to leverage multiple machines). If you don't know which
type of job you will be running, please schedule an appointment to talk with one of the staff.

Serial Job Script

A sample job script name myjob.qsub might include the following commands for a serial job.

```
#!/bin/bash -login
#PBS -l walltime=00:01:00,nodes=1:ppn=1
#PBS -j oe
#PBS -N name_of_job

# load necessary modules, e.g.
# module load HDF5
module load HDF5


# change to the working directory where your code is located
cd <path_to_code>

# call your executable
<exectuable> <calling parameters>
```

Shared Memory Job Script

A sample job script name myjob.qsub might include the following commands for a shared memory job.

```
#!/bin/bash -login
#PBS -l walltime=00:01:00,nodes=1:ppn=4
#PBS -j oe
#PBS -N name_of_job

# load necessary modules, e.g.
# module load HDF5
module load HDF5


# change to the working directory where your code is located
cd <path_to_code>

# set the number of OpenMP threads
export OMP_NUM_THREADS=4

# call your executable
<exectuable> <calling parameters>
```

Distributed Memory Job Script

A sample job script name myjob.qsub might include the following commands for a distributed memory job.

```
#!/bin/bash -login
#PBS -l walltime=00:01:00,nodes=5:ppn=1
#PBS -j oe
#PBS -N name_of_job

# load necessary modules, e.g.
# module load HDF5
module load HDF5


# change to the working directory where your code is located
cd <path_to_code>

# call your executable
mpirun -np 5 <exectuable>
```

Please click on the link for more information about submitting jobs.
Once the job script has been created, the job can be submitted using the command

```
qsub myjob.qsub
```

Job Control

To show submitted user jobs, type

```
qstat -u <username>
```

For more detailed information once the submitted job has been processed by the resource manager, type

```
showq -u <username>
```

To delete a job, type

```
qdel job_id
```

To delete all submitted job, type

```
qdel $(qselect -u username)
```

⚠ the maximum number of jobs that can be queued at any time is 1000 jobs.

Support

Please take some time to read through the tutorials and FAQs.

- An extensive amount of documentation is available through our wiki.
- User requests for software or support can be submitted via our tracking system. This system is also monitored off hours in case of an emergency.
- You can also schedule an appointment or drop into the HPCC/iCER office, located in 1440 BPS, Monday through Friday from 9am until 5pm.
- An increase to your home directory quota space can be requested using the form at http://www.hpcc.msu.edu/quota.

A wide variety of performance statistics concerning cluster performance are available on the wiki here. If you would like to instant message iCER staff during 9-5pm, please visit http://www.hipchat.com/gn7JbZLSS

Acknowledgements (Citing iCER in publications)

We encourage HPCC users to acknowledge iCER / MSU in publications arising from simulations performed on our resources. Let us know that we have been referenced (http://contact.icer.msu.edu), and we will link to your publication on our site, https://icer.msu.edu/research/publications, which will further increase the visibility of your work by leveraging the coin citation protocol which is used by search engines to find scholarly publications (Such as Google Scholar). A sample statement below can be tailored

"This work was supported in part by Michigan State University through computational resources provided by the Institute for Cyber-Enabled Research."