Caviness HPC Tutorial Series

Caviness HPC Basics

Objectives

- Overview: Caviness Community Cluster
- Part I: Get your feet wet
- Part II: Jump in

Overview: Caviness HPC Basics

Caviness Community Cluster

http://docs.hpc.udel.edu/abstract/caviness/caviness

Background

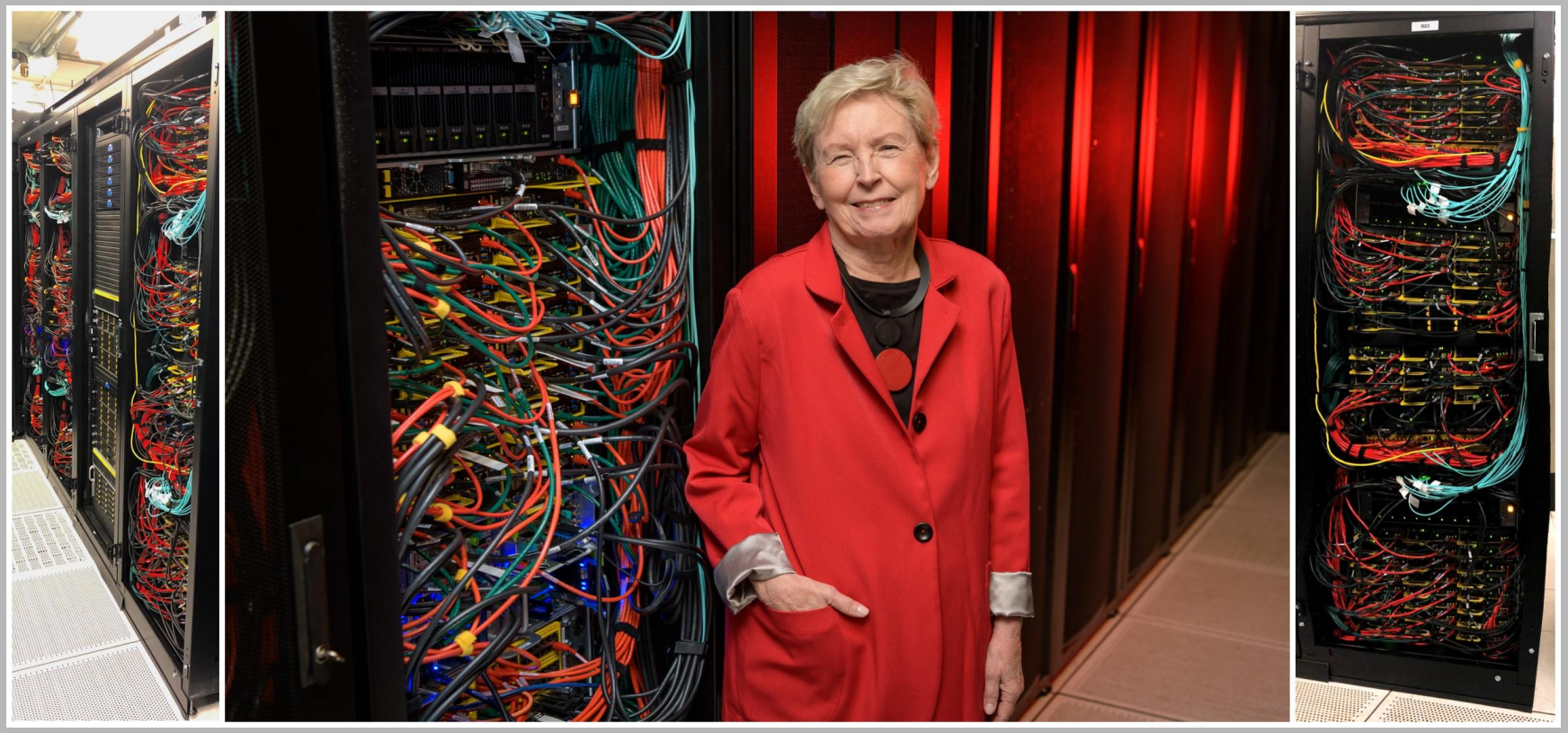
What is the Caviness cluster?

- It is the third UD community cluster
- Technical and financial partnership between UD-IT and UD faculty and researchers

Who can use it?

- UD faculty and researchers who purchased compute nodes (stakeholders)
- Researchers sponsored by a stakeholder

Caviness Cluster



https://sites.udel.edu/research-computing/caviness-cluster/

Part I: Caviness HPC Basics

Getting your feet wet

Getting your feet wet

- Accounts
- Connecting with SSH
- Bash Shell and Working Directory
- File Storage
- Groups and Workgroup(s)
- VALET
- Workgroup, Load Packages and Run Jobs
- Help

Accounts

Caviness Accounts

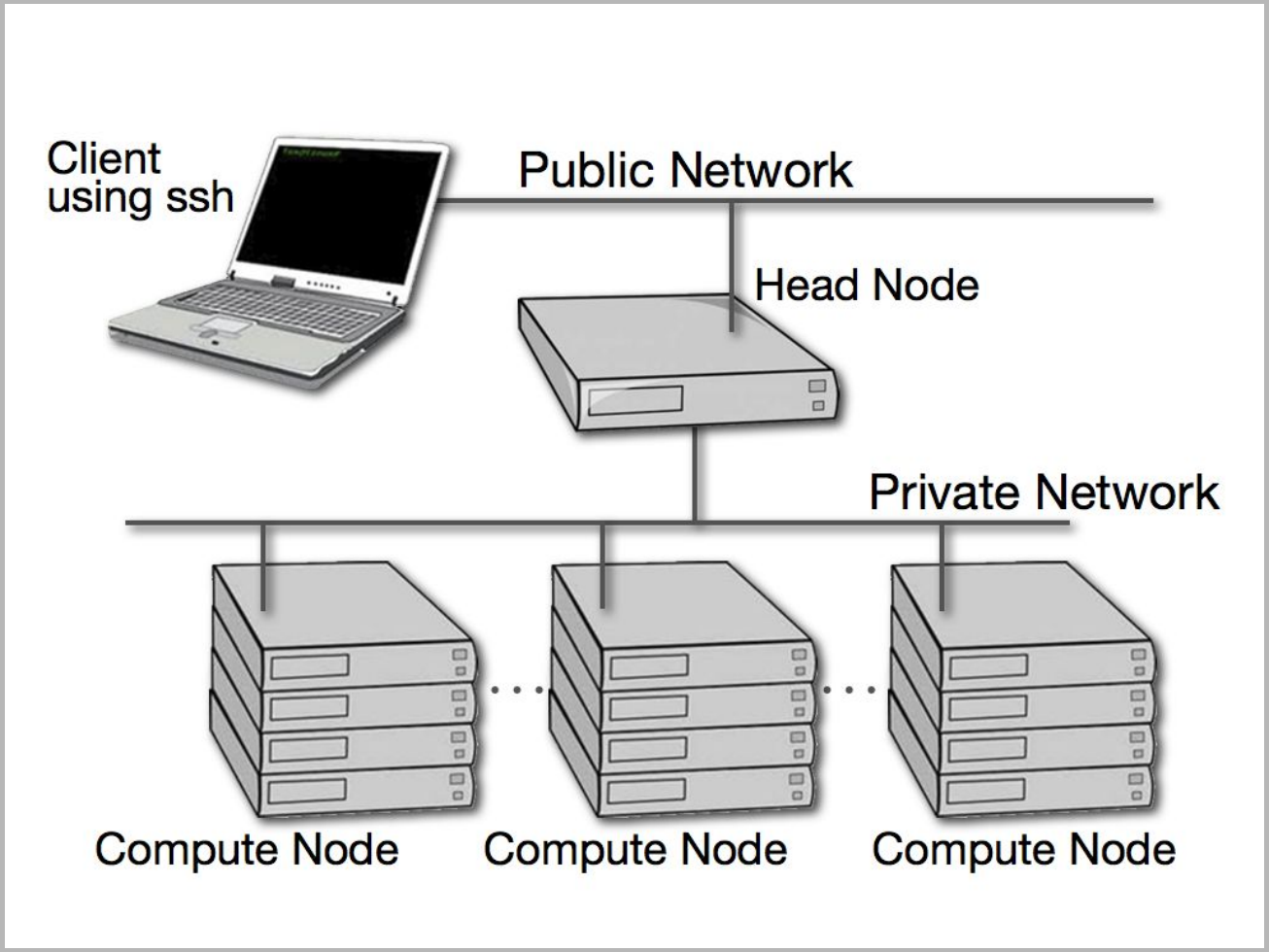
Username and Password

- UD = UDelNet ID and password; can only be changed via the on the Network page.
  www.udel.edu/network

- non-UD = `hpcguest<`***uid***`>` and password is generated by IT staff and securely sent via the UD Dropbox; please change it using:
  www.hpc.udel.edu/user?authn=login

Connecting with SSH

Overview


Client
using ssh
Public Network
Head Node
Private Network
Compute Node
Compute Node
Compute Node

SSH Client

- SSH is typically used to connect to the cluster's head (login) node.
- Standard Linux and Mac distributions provide an ssh client.
- Windows distributions require installation of an ssh client such as PuTTY.

SSH Public/Private Keys

- Eliminates entering your password for each remote connection - only need to remember a passphrase of your choice
- More convenient and efficient especially with other applications such as scp and sftp

SSH Help

- Follow documentation for Mac and Linux, or Windows configuration to get connected using X11 and SSH with public/private keys.

  http://www.udel.edu/it/research/training/config_laptop/

Connecting to Caviness

```
ssh -Y username@caviness.hpc.udel.edu
```

```
Using username "traine".

..............................................................


    Caviness cluster (caviness.hpc.udel.edu)

    This computer system is maintained by University of Delaware
    IT.  Links to documentation and other online resources can be
    found at:

    http://docs.hpc.udel.edu/abstract/caviness/

    For support, please contact consult@udel.edu


..............................................................
```

Bash Shell and Working Directory

Bash Shell

Bash prompt

- user name = referred to as `$USER`
- cluster name = head (login) node
- `~` = current working directory
- `$` = end of prompt

```
[traine@login00 ~]$
```

Working Directory

At login, you start in your home directory (~)

- `/home/<`***uid***`>`
- Referred to as `$HOME`

```
[traine@login00 ~]$ pwd
/home/1201
[traine@login00 ~]$ echo $HOME
/home/1201
```

File Storage

File Storage on Caviness

- Home directory (`/home`)

Other file storage available:

- Workgroup directory (`/work/`<***investing-entity***>)
- Lustre (`/lustre/scratch`)
- Node-local scratch (`/tmp`)

Groups and Workgroups

Workgroup(s)

Groups in the ***investing-entity category*** are used to control access to compute nodes, partitions and storage (HPC resources).

`workgroup -g` <***investing_entity***>

starts a new shell in your workgroup. You must set your workgroup to run a job on the cluster.

```
[traine@login00 ~]$ workgroup -q workgroups
 1002  it_css
[traine@login00 ~]$ workgroup -g it_css
[(it_css:traine)@login00 traine]$ echo $WORKDIR
/work/it_css
```

VALET

VALET

- UD-developed software to help configure your environment for all IT-installed software packages.
- Changes environment such as `PATH`, `LD_LIBRARY_PATH` and `MANPATH`
- Changes software development environment such as `LDFLAGS` and `CPPFLAGS`
- An alternative to the Modules software used at other HPC sites

```
man valet
```

VALET Commands

`vpkg_list`

- a list of all available software packages installed by IT

```
Available packages:
  in /opt/shared/valet/2.1/etc
    abaqus
    ambertools
    anaconda
    arpack
    arpack-ng
    atlas
    bazel
    binutils
    blacs
    boost
    cdo
    cgal
    chargemol
...
...
```

VALET Commands

`vpkg_versions` *<package_id>*

- a list of versions available for *<package_id>*
- default version marked with *

```
[(it_css:traine)@login00 ~]$ vpkg_versions intel
Available versions in package (* = default version):

[/opt/shared/valet/2.1/etc/intel.vpkg_yaml]
intel    Intel Compiler Suite
  2013   alias to intel/2013u6
  2013u6  Version 2013, SP1 Update 6 (2013_sp1.6.214)
  2015   alias to intel/2015u7
  2015u7  Version 2015, Update 7 (2015.7.235)
  2016   alias to intel/2016u5
  2016u5  Version 2016, Update 5 (2016.8.266)
  2017   alias to intel/2017u7
  2017u7  Version 2017, Update 7 (2017.7.259)
* 2018   alias to intel/2018u4
...
```

VALET Commands

`vpkg_require <package_id>`
`vpkg_devrequire <package_id>`

- set your environment or development environment for <*package_id*>

```
[(it_css:traine)@login00 ~]$ vpkg_require intel
Adding package `intel/2015.0.090` to your environment
[(it_css:traine)@login00 ~]$
```

```
[(it_css:traine)@login00 ~]$ vpkg_devrequire intel
Adding package `intel/2015.0.090` to your environment
[(it_css:traine)@login00 ~]
```

VALET Commands

`vpkg_rollback all`

- undo all changes to your environment

```
[(it_css:traine)@login00 ~]$ vpkg_rollback all
[(it_css:traine)@login00 ~]$
```

Workgroup, Load Packages, and Run Jobs

Workgroup

To use any of your HPC resources (compute nodes) you need to set your workgroup. **This will create a new shell session and change the prompt.**

`workgroup -g <investing_entity>`

```
[traine@login00~]$ workgroup -g it_css
[(it_css:traine)@login00 ~]$
```

Load Packages for Applications

- Use VALET to set up your runtime environment and/or compile-time environment.

```
vpkg_require
```

or

```
vpkg_devrequire
```

Compilers

There are three 64-bit compiler suites on Caviness:

- PGI Portland Compiler Suite
- Intel Parallel Studio XE
- GSS (GNU Compiler Collection)

We generally recommend that you use the `gcc` compilers with its rich collection of tools and libraries. If your software/application use libraries designed for Intel or PGI compilers, you might see improved performance by using the corresponding compiler. Intel Fortran, `ifort`, is a better implementation of modern Fortran standards than `gfortran`.

Run Applications

In general, applications (executables) must be run on the compute nodes, not on the login (head) node.

Use one of Slurm's job submission commands to run an application.

`salloc` (interactive)
`sbatch` (batch)

Compile Code

C and Fortran Examples

C and Fortran program examples

- `cmatmul` and `fmatmul`

Compile scripts for each compiler to create executables (“application”)

- `compile-gcc` and `compile-intel`

Batch scripts for each compiler to run job

- `serial-gcc.qs` and `serial-intel.qs`

Copy Examples

```
cp -r ~trainf/fhpcI .
cd fhpcI
pwd
ls
```

```
[traine@login00 ~]$ cp -r ~trainf/fhpcI .
[traine@login00 ~]$ cd fhpcI/
[traine@login00 fhpcI]$ pwd
/home/1201/fhpcI
[traine@login00 fhpcI]$ ls
cmatmul  fmatmul
```

Compile Code: system cc

- Basic programs can be compiled on the login (head) node or compute nodes ("devel" partition only).
- Use VALET to set up your compile-time environment.

This example uses the system compiler (gcc) to compile a C program on the head node to create the executable `tmatmul`

```
[traine@login00 fhpcI]$ cd cmatmul
[traine@login00 cmatmul]$ ls *.c
 tmatmul.c
[traine@login00 cmatmul]$ vpkg_require gcc
 Adding package `gcc/system` to your environment
[traine@login00 cmatmul]$ make tmatmul
 cc     tmatmul.c   -o tmatmul
[traine@login00 cmatmul]$ mv tmatmul tmatmul-gcc
[traine@login00 cmatmul]$ ls -la tmatmul*
 -rw-r--r-- 1 traine everyone  818 Sep 24 17:07 tmatmul.c
 -rwxr-xr-x 1 traine everyone 8860 Sep 28 22:26 tmatmul-gcc
```

Compile Code: intel icc

- Basic programs can be compiled on the login (head) node or compute nodes (“devel” partition only).
- Use VALET to set up your compile-time environment.

This example uses a script, `compile-intel`, on the head node which has the commands to create an Intel version executable `tmatmul-intel`

```
[traine@login00 cmatmul]$ source compile-intel
 Adding package `intel/2018u4` to your environment
 icc -Wall -g -debug all  tmatmul.c   -o tmatmul
 debug executable in ./tmatmul-intel
[traine@login00 cmatmul]$ ls -la tmatmul*
 -rw-r--r-- 1 traine everyone  818 Sep 24 17:07 tmatmul.c
 -rwxr-xr-x 1 traine everyone 8860 Sep 28 22:26 tmatmul-gcc
 -rwxr-xr-x 1 traine everyone 8781 Sep 28 22:28 tmatmul-intel
```

Run Jobs

Run Jobs

- Interactively using `salloc`

  Slurm will submit an interactive job to the queuing system.

- Batch using `sbatch` <***job_script***>

  Slurm will submit a batch job <***job_script***> to the queuing system.

Note: Running Jobs is based on using the compiled code examples from the previous section, Compiling Code.

Interactive (session) job

`salloc`

Set workgroup (reminder this starts a new shell) before using `salloc`

```
[traine@login00 cmatmul]$ workgroup -g it_css
[(it_css:traine)@login00 cmatmul]$ salloc --partition=devel
salloc: Pending job allocation 7289274
salloc: job 7289274 queued and waiting for resources
salloc: job 7289274 has been allocated resources
salloc: Granted job allocation 7289274
salloc: Waiting for resource configuration
salloc: Nodes r01n46 are ready for job
```

Note: Remember we used the “devel” partition to compile code on the compute node since libraries are likely not available on the standard or workgroup partitions.

Interactive Run

Run `tmatmul-gcc` on a compute node and exit

```
[(it_css:traine)@r01n46 cmatmul]$  vpkg_require gcc
Adding package `gcc/4.8.5` to your environment
[(it_css:traine)@r01n46 cmatmul]$ . compile-gcc
gcc -g -Wall tmatmul.c   -o tmatmul
debug executable in ./tmatmul-gcc
[(it_css:traine)@r01n46 cmatmul]$ ./tmatmul-gcc
B:
        1.00000         1.00000         1.00000
        1.00000         1.50000         2.25000
        1.00000         2.00000         4.00000
        1.00000         3.00000         9.00000
C:
        1.00000         0.00000
        0.00000         1.00000
        0.50000         0.50000
B*C with loops:
        1.50000         1.50000
        2.12500         2.62500
        3.00000         4.00000
        5.50000         7.50000
[(it_css:traine)@r01n46 cmatmul]$ exit
exit
salloc: Relinquishing job allocation 7289274
```

Batch Job Example I

`sbatch <job_script>`

```
[traine@login00 cmatmul]$ workgroup -g it_css
[traine@login00 cmatmul]$ cd fhpcI/cmatmul/
[(it_css:traine)@login00 cmatmul]$ sbatch --partition=devel serial-gcc.qs
 Submitted batch job 7499728
[(it_css:traine)@login00 cmatmul]$ more slurm-7499728.out
Adding package `gcc/4.8.5` to your environment
B:
     1.00000      1.00000      1.00000
     1.00000      1.50000      2.25000
     1.00000      2.00000      4.00000
     1.00000      3.00000      9.00000
C:
     1.00000      0.00000
     0.00000      1.00000
     0.50000      0.50000
B*C with loops:
     1.50000      1.50000
     2.12500      2.62500
     3.00000      4.00000
     5.50000      7.50000
```

Note: We can also use the "devel" partition to test simple code rather than using the standard or workgroup partitions.

Batch Job Example II

- Sample <*job_script*> was copied from `/opt/templates/slurm/generic/serial.qs` and modified as `serial-intel.qs`

```
#SBATCH --job-name=tmatmul-intel_job
#SBATCH --partition=devel
...
#SBATCH --mail-user='<user_email>@udel.edu'
#SBATCH --mail-type=END,FAIL,TIME_LIMIT_90
…
vpkg_require intel
#srun date
./compile-intel
srun ./tmatmul-intel
```

Batch Job Script: Part I serial-intel.qs

```
[(it_css:traine)@login00 cmatmul]$ cat serial-intel.qs
#!/bin/bash -l
#
# Sections of this script that can/should be edited are delimited by a
# [EDIT] tag.  All Slurm job options are denoted by a line that starts
# with "#SBATCH " followed by flags that would otherwise be passed on
# the command line.  Slurm job options can easily be disabled in a
# script by inserting a space in the prefix, e.g. "# SLURM " and
# reenabled by deleting that space.
#
# This is a batch job template for a program using a single processor
# core/thread (a serial job).
#
#SBATCH --ntasks=1
#
# [EDIT] All jobs have memory limits imposed.  The default is 1 GB per
#         CPU allocated to the job.  The default can be overridden either
#         with a per-node value (--mem) or a per-CPU value (--mem-per-cpu)
#         with unitless values in MB and the suffixes K|M|G|T denoting
#         kibi, mebi, gibi, and tebibyte units.  Delete the space between
#         the "#" and the word SBATCH to enable one of them:
#
# SBATCH --mem=8G
# SBATCH --mem-per-cpu=1024M
#
```

Batch Job Script: Part II
serial-intel.qs

```
# [EDIT] Each node in the cluster has local scratch disk of some sort
#         that is always mounted as /tmp.  Per-job and per-step temporary
#         directories are automatically created and destroyed by the
#         auto_tmpdir plugin in the /tmp filesystem.  To ensure a minimum
#         amount of free space on /tmp when your job is scheduled, the
#         --tmp option can be used; it has the same behavior unit-wise as
#         --mem and --mem-per-cpu.  Delete the space between the "#" and the
#         word SBATCH to enable:
#
# SBATCH --tmp=24G
#
# [EDIT] It can be helpful to provide a descriptive (terse) name for
#         the job (be sure to use quotes if there's whitespace in the
#         name):
#
#SBATCH --job-name=tmatmul-intel_job
#
# [EDIT] The partition determines which nodes can be used and with what
#         maximum runtime limits, etc.  Partition limits can be displayed
#         with the "sinfo --summarize" command.
#
# SBATCH --partition=standard
#SBATCH --partition=devel
#
#         To run with priority-access to resources owned by your workgroup,
#         use the "_workgroup_" partition:
#
# SBATCH --partition=_workgroup_
```

Batch Job Script: Part III
serial-intel.qs

```
#
# [EDIT] The maximum runtime for the job; a single integer is interpreted
#         as a number of minutes, otherwise use the format
#
#         d-hh:mm:ss
#
#         Jobs default to the default runtime limit of the chosen partition
#         if this option is omitted.
#
#SBATCH --time=0-02:00:00
#
#         You can also provide a minimum acceptable runtime so the scheduler
#         may be able to run your job sooner.  If you do not provide a
#         value, it will be set to match the maximum runtime limit (discussed
#         above).
#
# SBATCH --time-min=0-01:00:00
# [EDIT] By default SLURM sends the job's stdout to the file "slurm-<jobid>.out"
#         and the job's stderr to the file "slurm-<jobid>.err" in the working
#         directory.  Override by deleting the space between the "#" and the
#         word SBATCH on the following lines; see the man page for sbatch for
#         special tokens that can be used in the filenames:
#
# SBATCH --output=%x-%j.out
# SBATCH --error=%x-%j.out
#
```

Batch Job Script: Part IV
serial-intel.qs

```
# [EDIT] Slurm can send emails to you when a job transitions through various
#         states: NONE, BEGIN, END, FAIL, REQUEUE, ALL, TIME_LIMIT,
#         TIME_LIMIT_50, TIME_LIMIT_80, TIME_LIMIT_90, ARRAY_TASKS.  One or more
#         of these flags (separated by commas) are permissible for the
#         --mail-type flag.  You MUST set your mail address using --mail-user
#         for messages to get off the cluster.
#
#SBATCH --mail-user='traine@udel.edu'
#SBATCH --mail-type=END,FAIL,TIME_LIMIT_90
#
# [EDIT] By default we DO NOT want to send the job submission environment
#         to the compute node when the job runs.
#
#SBATCH --export=NONE
#


#
# [EDIT] Define a Bash function and set this variable to its
#         name if you want to have the function called when the
#         job terminates (time limit reached or job preempted).
#
#         PLEASE NOTE:  when using a signal-handling Bash
#         function, any long-running commands should be prefixed
#         with UD_EXEC, e.g.
#
#              UD_EXEC mpirun vasp
#
```

Batch Job Script: Part V
serial-intel.qs

```
#
#         If you do not use UD_EXEC, then the signals will not
#         get handled by the job shell!
#
#job_exit_handler() {
#  # Copy all our output files back to the original job directory:
#  cp * "$SLURM_SUBMIT_DIR"
#
#  # Don't call again on EXIT signal, please:
#  trap - EXIT
#  exit 0
#}
#export UD_JOB_EXIT_FN=job_exit_handler

#
# [EDIT] By default, the function defined above is registered
#         to respond to the SIGTERM signal that Slurm sends
#         when jobs reach their runtime limit or are
#         preempted.  You can override with your own list of
#         signals using this variable -- as in this example,
#         which registers for both SIGTERM and the EXIT
#         pseudo-signal that Bash sends when the script ends.
#         In effect, no matter whether the job is terminated
#         or completes, the UD_JOB_EXIT_FN will be called.
#
#export UD_JOB_EXIT_FN_SIGNALS="SIGTERM EXIT"
```

Batch Job Script: Part VI
serial-intel.qs

```
#
# If you have VALET packages to load into the job environment,
# uncomment and edit the following line:
#
#vpkg_require intel/2019
#
# Do general job environment setup:
#
. /opt/shared/slurm/templates/libexec/common.sh
#
# [EDIT] Add your script statements hereafter, or execute a script or program
#         using the srun command.
#
#srun date
. compile-intel
srun ./tmatmul-intel
```

Example Batch Job template running serial script

`sbatch <script_name>.qs`

```
[traine@login01 ~]$ workgroup -g it_css
[(it_css:traine)@login00 ~]$
[(it_css:traine)@login00 cmatmul]$ sbatch serial-intel.qs
Submitted batch job 7319222
```

The `serial-intel.qs` job script specifies to run this job on the “devel” partition and send an email notification to `traine@udel.edu` when the job is complete.

Note: There is a possibility of a slight delay between when the job completes and the generation of the slurm output file.

Batch Run Output

Output in `slurm-<job_id>.out`

```
[(it_css:traine)@login00 cmatmul]$ more slurm-7319222.out
Adding package `intel/2018u4` to your environment
icc -Wall -g -debug all   tmatmul.c   -o tmatmul
debug executable in ./tmatmul-intel
B:
     1.00000      1.00000      1.00000
     1.00000      1.50000      2.25000
     1.00000      2.00000      4.00000
     1.00000      3.00000      9.00000
C:
     1.00000      0.00000
     0.00000      1.00000
     0.50000      0.50000
B*C with loops:
     1.50000      1.50000
     2.12500      2.62500
     3.00000      4.00000
     5.50000      7.50000
[(it_css:traine)@login01 cmatmul]$
```

Exercise

Exercise

- Pick a compiler: gcc or intel
- Compile and batch run the Fortran example in `fmatmul` using

  `compile-<`***compiler***`>` to compile
  `serial-<`***compiler***`>.qs` to batch run

This example is using a simple Fortran program to create the executable `tmatmul-gcc` using the system gfortran (gcc) compiler

Set Workgroup and Change Directory

- Set your workgroup if not already done.

```
[traine@login00 cmatmul]$ workgroup -g it_css
[(it_css:traine)@login00 cmatmul]$
```

- Change into the fmatmul directory.

```
[(it_css:traine)@login00 cmatmul]$ cd ~/fhpcI/fmatmul
[(it_css:traine)@login00 fmatmul]$ pwd
/home/1201/fhpc/fmatmul
[(it_css:traine)@login00 fmatmul]$
```

Compile Code

- Create `tmatmul-gcc` fortran executable by sourcing the compile script `compile-gcc`

```
source compile-gcc
```

```
[(it_css:traine)@login00 fmatmul]$ cat compile-gcc
vpkg_devrequire gcc
export FC=gfortran
export FFLAGS='-ffree-form -Wall -g'
make tmatmul &&  mv tmatmul tmatmul-gcc
test -x tmatmul-gcc && echo "debug version in ./tmatmul-gcc"
[(it_css:traine)@login00 fmatmul]$ source compile-gcc
Adding package `gcc/4.8.5` to your environment
gfortran -ffree-form -Wall -g   tmatmul.f   -o tmatmul
debug version in ./tmatmul-gcc
[(it_css:traine)@login00 fmatmul]$
```

Batch Job Run

- Submit a batch job to run the fortran executable `tmatmul-gcc`

```
[(it_css:traine)@login00 fmatmul]$ cat serial-gcc.qs
#!/bin/bash -l
...
vpkg_require gcc
...
# [EDIT] Add your script statements hereafter, or execute a script or
program
#        using the srun command.
#
#srun date
srun ./tmatmul-gcc

[(it_css:traine)@login01 fmatmul]$
```

`sbatch serial-gcc.qs`

```
[(it_css:traine)@login00 fmatmul]$ sbatch serial-gcc.qs
 Submitted batch job 7321828
```

Batch Job Output

```
[(it_css:traine)@login00 fmatmul]$ more slurm-7321828.out
Adding package `gcc/4.8.5` to your environment
 B:
    1.0000       1.0000       1.0000
    1.0000       1.5000       2.2500
    1.0000       2.0000       4.0000
    1.0000       3.0000       9.0000
 C:
    1.0000       0.0000
    0.0000       1.0000
    0.50000      0.50000
 B*C with intrinsic matmul
    1.5000       1.5000
    2.1250       2.6250
    3.0000       4.0000
    5.5000       7.5000
 B*C  with loops
    1.5000       1.5000
    2.1250       2.6250
    3.0000       4.0000
    5.5000       7.5000
```

Need Help?

- **Submit** a Research Computing Help Request form
- **Phone**: (302) 831-6000

Please select *High Performance Computing* as the Problem Type, and specify Caviness in the details of your problem in the Description when submitting your request or calling IT Support Center.

Part II: Caviness HPC Basics

Jumping in

Jumping in

- File Storage and recovery options
- Bash startup files
- Setting Environment, Running and Monitoring Jobs
- Local (non-standard) Commands

File Storage

File Storage on Caviness

- Home directory 72 TB of usable space
  - Personal directory: 20GB (`/home/<`***uid***`>`)
    `df -h $HOME`
  - Workgroup directory: 1 TB+
    (`/work/`***<investing-entity>***)
    `df -h $WORKDIR`
- Lustre ~191 TB of usable space
  - Public scratch directory (`/lustre/scratch`)
  - IT staff will run regular cleanup procedures to purge aged files or directories
- Node-local scratch (`/tmp`)

Recovery Options

Recovering files /home

/home filesystem is a larger permanent storage with snapshots.

- Use read-only $HOME/.zfs/snapshot to recover files

```
[traine@login00 cmatmul]$ ls -al tmatmul.c
-rw-r--r-- 1 traine everyone 818 Sep 24 17:07 tmatmul.c
[traine@login00 cmatmul]$ rm tmatmul.c
[traine@login00 cmatmul]$ ls -al tmatmul.c && echo 'oops !!'
ls: cannot access tmatmul.c: No such file or directory
oops !!
[traine@login00 cmatmul]$ pwd
/home/1201/fhpcI/cmatmul
[(it_css:traine)@login00 fmatmul]$ ls ~/.zfs/snapshot
20200314-1315  20200316-0115  20200317-0115  20200318-0115  20200319-0115
20200315-1315  20200316-1315  20200317-1315  20200318-1315  20200319-1315
[traine@login00 cmatmul]$ cp -p
~/.zfs/snapshot/20200319-1315/fhpcI/cmatmul/tmatmul.c .
[traine@login00 cmatmul]$ ls -al tmatmul.c
-rw-r--r-- 1 traine everyone 818 Sep 24 17:07 tmatmul.c
```

Recovering workgroup files

Workgroup files have their own read-only snapshots, which span a week in

`$WORKDIR/.zfs/snapshot`

```
[traine@login00 cmatmul]$ workgroup -g it_css
[(it_css:traine)@login00 cmatmul]$ ls $WORKDIR/.zfs/snapshot/
20200304-1815  20200305-1815  20200306-1815  20200307-1815
20200308-1815  20200309-1815  20200310-1815  20200311-1815
20200312-1815  20200313-1815  20200314-1815  20200315-1815
20200316-0615  20200316-1815  20200317-0615  20200317-1815
20200318-0615  20200318-1815  20200319-0615
```

Bash Startup Files

Keep startup files clean

- Make sure you understand what your startup files are doing.
- Environments are different for the login (head) node versus the compute nodes.
- If you make changes, test by starting a new login, don't logout.
- You can always restore your startup files to the system versions, if you make a mistake. Which is why it is important to not log out until you have tested with a new login.

Startup files

- `.bash_profile`
- `.bashrc`
- `.bash_udit`
- `.bash_logout`

Note: A "`.`" as part of the file or directory name causes the file to be hidden. To see these files or directories, use the command "`ls -a`" instead of "`ls`" with no options.

.bash_profile

- Executed once at login
- `.bashrc` in your home directory is sourced
- Add lines to set your environment and start programs after the comment line in red

```
[traine@login00 ~]$ more .bash_profile
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin

export PATH
```

.bashrc

- Executed by each new shell, including your login shell via `.bash_profile`
- Add lines to create aliases and bash functions after the comment line in red

```
[(it_css:traine)@login00 ~]$ more .bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and functions
```

.bash_udit: Part I

- Executed by each new shell
- Opt into IT suggested environment changes

```
[(it_css:traine)@login00 ~]$ more .bash_udit
##
## Change from "no" to "yes" to enable IT's suggested environment changes.
## The behaviors enabled by the remainder of this file are contingent on
## enabling IT_WANT_ENV_EXTENSIONS:
##
IT_WANT_ENV_EXTENSIONS="no"

##
## If you have multiple workgroups available to you, change this to the one
## you want to be the default; otherwise, the first one listed by
## "workgroup -q workgroups" will be your default:
##
IT_DEFAULT_WORKGROUP=""

##
## If you want the "workgroup" command to by default change your working
## directory to the $WORKDIR, change from "no" to "yes".
##
IT_WORKGROUP_CHDIR="no"
```

.bash_udit: Part II

```
##
## By default when you login to the cluster head node you are in the
## "everyone" group and need to issue a "workgroup" command to prepare
## for submitting jobs, etc.
##
## Change this flag from "no" to "yes" if you want your login shell to
## automatically issue the command to change into your default
## workgroup.  Your default workgroup will come from IT_DEFAULT_WORKGROUP
## if set above, or it will be the first group in the list produced by
## the command
##
##   /opt/bin/workgroup --query workgroups
##
IT_SET_WORKGROUP_ON_LOGIN="no"

[(it_css:traine)@login00 ~]$
```

.bash_logout

- Executed only when you log out from the head (login) node, but not when you exit from a compute when you use `salloc` or `sbatch`.

```
[(it_css:traine)@login00 ~]$ more .bash_logout
# ~/.bash_logout
```

Restore your startup files

To restore all your startup files (`.bashrc`, `.bash_profile`, `.bash_udit`, and `.bash_logout`) to the system default, type the command

```
cp /etc/skel/.bash* $HOME
```

Exercise (.bash_udit)

Exercise (.bash_udit)

To see what aliases are defined use

`alias`

```
[(it_css:traine)@login00 ~]$ alias
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mc='. /usr/libexec/mc/mc-wrapper.sh'
alias vi='vim'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot
--show-tilde'

[(it_css:traine)@login00 ~]$
```

Exercise (.bash_udit): Part I

Customize our startup file `.bash_udit` to opt into IT suggested environment changes by setting a default workgroup so we only need to type

```
workgroup
```

instead of

```
workgroup -g <investing_entity>
```

Exercise (.bash_udit): Part I

Edit (nano) `.bash_udit`

```
[(it_css:traine)@login00 ~]$ vim .bash_udit
##
## Change from "no" to "yes" to enable IT's suggested environment changes.
## The behaviors enabled by the remainder of this file are contingent on
## enabling IT_WANT_ENV_EXTENSIONS:
##
IT_WANT_ENV_EXTENSIONS="yes"

##
## If you have multiple workgroups available to you, change this to the one
## you want to be the default; otherwise, the first one listed by
## "workgroup -q workgroups" will be your default:
##
IT_DEFAULT_WORKGROUP="it_css"

##
## If you want the "workgroup" command to by default change your working
## directory to the $WORKDIR, change from "no" to "yes".
##
IT_WORKGROUP_CHDIR="no"
```

Exercise (.bash_udit): Part I

Try out our new `.bash_udit`

- **Do not logout!** Start a new login session
- Now you only need to type `workgroup`

```
[traine@login00 ~]$ alias
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mc='. /usr/libexec/mc/mc-wrapper.sh'
alias vi='vim'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot
alias workgroup='/usr/local/bin/workgroup  -g it_css'
[traine@login00 ~]$ workgroup
[(it_css:traine)@login00 ~]$
```

Exercise (.bash_udit): Part II

Customize our startup file `.bash_udit` to opt into IT suggested environment changes so the login shell is automatically set to your workgroup and also changes into your default workgroup directory `$WORKDIR`.

First, let's get back to the login shell

```
[(it_css:traine)@login00 ~]$ exit
[traine@login00 ~]$
```

Exercise (.bash_udit): Part II

Edit (vim) `.bash_udit`

```
[traine@login00 ~]$ vim .bash_udit
##
## Change from "no" to "yes" to enable IT's suggested environment changes.
## The behaviors enabled by the remainder of this file are contingent on
## enabling IT_WANT_ENV_EXTENSIONS:
##
IT_WANT_ENV_EXTENSIONS="yes"

##
## If you have multiple workgroups available to you, change this to the one
## you want to be the default; otherwise, the first one listed by
## "workgroup -q workgroups" will be your default:
##
IT_DEFAULT_WORKGROUP="it_css"

##
## If you want the "workgroup" command to by default change your working
## directory to the $WORKDIR, change from "no" to "yes".
##
IT_WORKGROUP_CHDIR="yes"
```

Exercise (.bash_udit): Part II

Edit (vim) `.bash_udit`

```
##
## By default when you login to the cluster head node you are in the
## "everyone" group and need to issue a "workgroup" command to prepare
## for submitting jobs, etc.
##
## Change this flag from "no" to "yes" if you want your login shell to
## automatically issue the command to change into your default
## workgroup.  Your default workgroup will come from IT_DEFAULT_WORKGROUP
## if set above, or it will be the first group in the list produced by
## the command
##
##   /opt/bin/workgroup --query workgroups
##
IT_SET_WORKGROUP_ON_LOGIN=" yes"
```

Exercise (.bash_udit): Part II

Try out our new `.bash_udit`

- **Do not logout!** Start a new login session

```
.................................................................

     Caviness cluster (caviness.hpc.udel.edu)

     This computer system is maintained by University of Delaware
     IT.  Links to documentation and other online resources can be
     found at:

     http://docs.hpc.udel.edu/abstract/caviness/

     For support, please contact consult@udel.edu
.................................................................

Last login: Tue Mar 24 09:40:31 2020 from 71.56.238.196
WARNING:  Your working directory has been changed to /work/it_css

[(it_css:traine)@login00 it_css]$ pwd
/work/it_css
[(it_css:traine)@login00 it_css]$
```

Exercise (.bashrc)

Exercise (.bashrc)

Customize our startup file `.bashrc` to create aliases for whatis, workgroups and other file storage directories

- Create a new alias `whatis`
- Create a new alias for each <***investing_entity***> to define a workgroup
- Create a new alias for each file storage personal work directory and change to it

Exercise (whatis)

Create an alias to determine what is the type of a command.

Example line shown in red

```
[(it_css:traine)@login00 ~]$ vim .bashrc
  1 # .bashrc
  2
  3 # Source global definitions
  4 if [ -f /etc/bashrc ]; then
  5         . /etc/bashrc
  6 fi
  7
  8 # User specific aliases and functions
  9
 10 alias whatis='builtin type -t'
```

Exercise (whatis)

Try out our new .bashrc

- Do not logout! Start a new login session.

```
[traine@login00 ~]$ alias
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mc='. /usr/libexec/mc/mc-wrapper.sh'
alias vi='vim'
alias whatis='builtin type -t'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot
--show-tilde'
alias workgroup='/usr/local/bin/workgroup  -g it_css'
[traine@login00 ~]$ which vpkg_require
/usr/bin/which: no vpkg_require in
(/opt/bin:/opt/shared/valet/2.0/bin/bash:/opt/shared/valet/2.0/bin:/opt/shared/un
iva/current/bin/lx-amd64:/usr/lib64/qt-3.3/bin:/usr/local/bin:/bin:/usr/bin:/usr/
local/sbin:/usr/sbin:/sbin:/opt/ibutils/bin:/home/1201/bin)
[traine@login00 ~]$ whatis vpkg_require
function
[traine@login00 ~]$
```

Exercise (workgroup)

Create an alias for each <*investing_entity*> to set the workgroup

Example lines shown in red for `it_css`

```
[(it_css:traine)@login00 ~]$ vim .bashrc
  1 # .bashrc
  2
  3 # Source global definitions
  4 if [ -f /etc/bashrc ]; then
  5         . /etc/bashrc
  6 fi
  7
  8 # User specific aliases and functions
  9
 10 alias whatis='builtin type -t'
 11 alias it_css='workgroup -g it_css'
```

Exercise (workgroup)

Try out our new .bashrc

- Do not logout! Start a new login session
- Now `it_css` and `workgroup` work the same.

```
[traine@login00 ~]$ alias
alias it_css='\workgroup -g it_css'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mc='. /usr/libexec/mc/mc-wrapper.sh'
alias vi='vim'
alias whatis='builtin type -t'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot
--show-tilde'
alias workgroup='/usr/local/bin/workgroup  -g it_css'
[traine@login00 ~]$ it_css
[(it_css:traine)@login00 ~]$ exit
exit
[traine@login00 ~]$ workgroup
[(it_css:traine)@login00 ~]$
```

Exercise (file storage)

Make sure you have a your own personal directory created for each file storage area. This may vary for each <*investing_entity*> research group (eg. `users` or `projects` subdirectory may exist).

These exercises assume your username will be in the base work directories and you have set your workgroup before you start.

- `/work/<`***investing_entity***`>/`
- `/lustre/scratch/`

Exercise (/work)

Check for your username in
`/work/<`***investing_entity***`> or $WORKDIR`

Example shows creating a personal directory for `traine` in `/work/it_css`

```
[traine@login00 ~]$ workgroup -g it_css
[(it_css:traine)@login00 ~]$ cd $WORKDIR
[(it_css:traine)@login00 it_css]$ ls -lad traine
ls: cannot access traine: No such file or directory
[(it_css:traine)@login00 it_css]$ mkdir traine
[(it_css:traine)@login00 it_css]$ ls -lad traine
drwxr-sr-x 2 traine it_css 2 Sep 29 00:10 traine
```

Exercise (/lustre/scratch)

Check for your username in `/lustre/scratch/`

Example shows a personal directory exists for `traine` in `/lustre/scratch`

```
[(it_css:traine)@login00 ~]$ cd /lustre/scratch
[(it_css:traine)@login00 scratch]$ ls -lad traine
drwxr-sr-x 2 traine it_css 4096 Sep 29 00:13 traine
[(it_css:traine)@login00 scratch]$
```

Exercise (file storage)

Create an alias for each file storage to change to that work directory

Example lines shown in red for `traine` and `it_css`

```
[(it_css:traine)@login00 ~]$ vim .bashrc
  1 # .bashrc
  2
  3 # Source global definitions
  4 if [ -f /etc/bashrc ]; then
  5         . /etc/bashrc
  6 fi
  7
  8 # User specific aliases and functions
  9
 10 alias whatis='builtin type -t'
 11 alias it_css='workgroup -g it_css'
 12 alias cdwork='cd /work/it_css/traine'
 13 alias cdscratch='cd /lustre/scratch/traine'
```

Exercise (file storage)

Try out our new .bashrc

- Do not logout! Start a new login session

```
alias cdscratch='cd /lustre/scratch/traine'
alias cdwork='cd /work/it_css/traine'
alias it_css='workgroup -g it_css'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mc='. /usr/libexec/mc/mc-wrapper.sh'
alias vi='vim'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot
--show-tilde'
alias workgroup='/usr/local/bin/workgroup  -g it_css'
[(it_css:traine)@login00 traine]$ cdwork
[(it_css:traine)@login00 traine]$ pwd
/work/it_css/traine
[(it_css:traine)@login00 traine]$ cdscratch
[(it_css:traine)@login00 traine]$ pwd
/lustre/scratch/traine
```

Set Environment, Run and Monitor Jobs

Python Example

Python program example using a Python 3.6.5 script.

- `pylib`

Python Script & Data File

- `unitConvert.py` and `dataFile.csv`

Batch job scripts

- `serial-python3.qs`

VALET

- Use VALET to set our environment, first see what Python versions are available using the VALET command

```
vpkg_versions python
```

```
[(it_css:traine)@login00 ~]$ vpkg_versions python

Available versions in package (* = default version):

[/opt/shared/valet/2.1/etc/python.vpkg_yaml]
python   Python Programming Language
  2      alias to python/2.7.15
  2.7.5    System OS Python (/usr/bin/python)
* 2.7.15   Python 2 with 200+ common add-on modules
  3      alias to python/3.6.5
  3.6.5    Python 3 with 200+ common add-on modules
  3.7.4    Python 3 with 200+ common add-on modules
```

Exercise

Set Environment and Copy Examples

Use your new aliases, `it_css` and `cdwork`, to set your workgroup, and change to your workgroup directory. Next copy the examples and change into the examples directory.

```
it_css
cdwork
cp -r ~trainf/fhpcIII ./
cd fhpcIII/pylib
```

```
[traine@login00 ~]$ it_css
[(it_css:traine)@login00 traine]$ cdwork
[(it_css:traine)@login00 traine]$ pwd
/home/work/it_css/traine
[(it_css:traine)@login00 traine]$ cp -r ~trainf/fhpcIII ./
[(it_css:traine)@login00 traine]$ cd fhpcIII/pylib
[(it_css:traine)@login00 clib]$ pwd
/home/work/it_css/traine/fhpcIII/pylib
[(it_css:traine)@login00 pylib]$ ls
dataFile.csv  serial-python3.qs  unitConvert.py
```

Batch job script: Part I

```
[(it_css:traine)@login00 clib]$ cat serial-python3.qs
#!/bin/bash -l
#
# Sections of this script that can/should be edited are delimited by a
# [EDIT] tag.  All Slurm job options are denoted by a line that starts
# with "#SBATCH " followed by flags that would otherwise be passed on
# the command line.  Slurm job options can easily be disabled in a
# script by inserting a space in the prefix, e.g. "# SLURM " and
# reenabled by deleting that space.
#
# This is a batch job template for a program using a single processor
# core/thread (a serial job).
#
#SBATCH --ntasks=1
#
# [EDIT] All jobs have memory limits imposed.  The default is 1 GB per
#        CPU allocated to the job.  The default can be overridden either
#        with a per-node value (--mem) or a per-CPU value (--mem-per-cpu)
#        with unitless values in MB and the suffixes K|M|G|T denoting
#        kibi, mebi, gibi, and tebibyte units.  Delete the space between
#        the "#" and the word SBATCH to enable one of them:
#
# SBATCH --mem=8G
# SBATCH --mem-per-cpu=1024M
```

Batch job script: Part II

```
# [EDIT] Each node in the cluster has local scratch disk of some sort
#        that is always mounted as /tmp.  Per-job and per-step temporary
#        directories are automatically created and destroyed by the
#        auto_tmpdir plugin in the /tmp filesystem.  To ensure a minimum
#        amount of free space on /tmp when your job is scheduled, the
#        --tmp option can be used; it has the same behavior unit-wise as
#        --mem and --mem-per-cpu.  Delete the space between the "#" and
the
#        word SBATCH to enable:
#
# SBATCH --tmp=24G
#
# [EDIT] It can be helpful to provide a descriptive (terse) name for
#        the job (be sure to use quotes if there's whitespace in the
#        name):
#
#SBATCH --job-name=serial_python3_job
#
# [EDIT] The partition determines which nodes can be used and with what
#        maximum runtime limits, etc.  Partition limits can be displayed
#        with the "sinfo --summarize" command.
#
#SBATCH --partition=devel
#
#        To run with priority-access to resources owned by your workgroup,
#        use the "_workgroup_" partition:
# SBATCH --partition= workgroup
```

Batch job script: Part III

```
# [EDIT] The maximum runtime for the job; a single integer is interpreted
#        as a number of minutes, otherwise use the format
#
#        d-hh:mm:ss
#
#        Jobs default to the default runtime limit of the chosen partition
#        if this option is omitted.
#
#SBATCH --time=0-02:00:00
#
#        You can also provide a minimum acceptable runtime so the scheduler
#        may be able to run your job sooner.  If you do not provide a
#        value, it will be set to match the maximum runtime limit (discussed
above).
#
# SBATCH --time-min=0-01:00:00
#
# [EDIT] By default SLURM sends the job's stdout to the file "slurm-<jobid>.out"
#        and the job's stderr to the file "slurm-<jobid>.err" in the working
#        directory.  Override by deleting the space between the "#" and the
#        word SBATCH on the following lines; see the man page for sbatch for
#        special tokens that can be used in the filenames:
#
# SBATCH --output=%x-%j.out
# SBATCH --error=%x-%j.out
```

Batch job script: Part IV

```
# [EDIT] Slurm can send emails to you when a job transitions through
various
#       states: NONE, BEGIN, END, FAIL, REQUEUE, ALL, TIME_LIMIT,
#       TIME_LIMIT_50, TIME_LIMIT_80, TIME_LIMIT_90, ARRAY_TASKS.  One or
more
#       of these flags (separated by commas) are permissible for the
#       --mail-type flag.  You MUST set your mail address using
--mail-user
#       for messages to get off the cluster.
#
# SBATCH --mail-user='my_address@udel.edu'
# SBATCH --mail-type=END,FAIL,TIME_LIMIT_90
#
# [EDIT] By default we DO NOT want to send the job submission environment
#       to the compute node when the job runs.
#
#SBATCH --export=NONE
#
```

Batch job script: Part V

```
#
# [EDIT] Define a Bash function and set this variable to its
#        name if you want to have the function called when the
#        job terminates (time limit reached or job preempted).
#
#        PLEASE NOTE:  when using a signal-handling Bash
#        function, any long-running commands should be prefixed
#        with UD_EXEC, e.g.
#
#                UD_EXEC mpirun vasp
#
#        If you do not use UD_EXEC, then the signals will not
#        get handled by the job shell!
#
#job_exit_handler() {
#  # Copy all our output files back to the original job directory:
#  cp * "$SLURM_SUBMIT_DIR"
#
#  # Don't call again on EXIT signal, please:
#  trap - EXIT
#  exit 0
#}
#export UD_JOB_EXIT_FN=job_exit_handler
```

Batch job script: Part VI

```
# [EDIT] By default, the function defined above is registered
#        to respond to the SIGTERM signal that Slurm sends
#        when jobs reach their runtime limit or are
#        preempted.  You can override with your own list of
#        signals using this variable -- as in this example,
#        which registers for both SIGTERM and the EXIT
#        pseudo-signal that Bash sends when the script ends.
#        In effect, no matter whether the job is terminated
#        or completes, the UD_JOB_EXIT_FN will be called.
#
#export UD_JOB_EXIT_FN_SIGNALS="SIGTERM EXIT"

#
# If you have VALET packages to load into the job environment,
# uncomment and edit the following line:
#
#vpkg require intel/2019
vpkg_require python/3.6.5
```

Batch job script: Part VII

```
#
# Do general job environment setup:
#
. /opt/shared/slurm/templates/libexec/common.sh

#
# [EDIT] Add your script statements hereafter, or execute a script or
program
#        using the srun command.
#
#srun date

echo ""
echo "---- Run Test ----"
python3 ./unitConvert.py
#Adding 20 second pause, so job can be seen in monitoring step.
sleep 20s
echo "---- End of Test ----"
```

Job Submission

`sbatch` : Submitting a batch job.

```
[(it_css:traine)@login00 pylib]$ sbatch serial-python3.qs
Submitted batch job 7585812
[(it_css:traine)@login00 pylib]$
```

Job Monitoring

Check the status of queued jobs.

`squeue -j <job_id>`

- Pending

```
[(it_css:traine)@login00 pylib]$ squeue -j 7585812
             JOBID PARTITION     NAME      USER  ST       TIME  NODES NODELIST(REASON)
             7585812    devel    serial_j    traine PD       0:00      1 (None)
```

- Running

```
[(it_css:traine)@login00 pylib]$ squeue -j 7585812
             JOBID PARTITION     NAME      USER  ST       TIME  NODES NODELIST(REASON)
             7585812    devel    serial_j    traine R       0:01      1 r00n56
```

More monitoring

`sstat -a <job_id>`

- To check the status information of a running job/step.

`scontrol show job <job_id>`

- For monitoring and modifying queued jobs, as well as holding and releasing jobs

More monitoring

`sacct -j <job_id>`

- To check the information about a job from history (i.e. a job that has already completed).

`scancel <job_id>`

- Removes pending and running jobs from the queue

Batch Run output

Look at batch run output

```
[(it_css:traine)@login00 pylib]$  tail slurm-7585812.out
||Temperature (degC): 22.6 || Wind Speed (m/s): 21.2 || Pressure (Pa): 102290.0||
||Temperature (degC): 22.5 || Wind Speed (m/s): 21.3 || Pressure (Pa): 102280.0||
||Temperature (degC): 22.5 || Wind Speed (m/s): 21.5 || Pressure (Pa): 102270.0||
=================================================================
Total Amount of Conversions: 26151
Total Amount of Rows: 8717 seconds
=================================================================
Total Run Time: 0.0328888893127
End of the python script!
--- End of Test ----
```

Local (non-standard) Commands

Local Commands

UD's IT status commands can be found at:

`/opt/shared/slurm/add-ons/bin`

`/usr/local/bin`

These are "non-standard" commands that are specific to Caviness; UD community clusters.

Local Commands

`hpc-user-info -a` ***username***

`hpc-user-info -h`

Display information about ***username***

```
[(it_css:traine)@login00 ~]$ hpc-user-info -a traine
full-name = Student Training
last-name = Student Training
home-directory = /home/1201
email-address = traine@udel.edu
clusters = Farber, Caviness
[(it_css:traine)@login01 ~]$
```

Local Commands

`sjobs -u` ***username***

`sjobs -h` ***or*** `--help`

Displays jobs status in more compact form.

```
[(it_css:traine)@login00 ~]$ sjobs -u traine
  JOBID USER    STATE     JOBNAME    GROUP NCPUS NNODES NTASKS
------- ----- ------- ---------- ------ ----- ------ ------
7546246 traine RUNNING serial_job it_css   1      1      1
```

Local Commands

`sworkgroup --workgroup <investing_entity>`

`sworkgroup -h` ***or*** `--help`

Displays the partitions and resources that a workgroup have access to on the Caviness cluster.

```
[(it_css:traine)@login00 ~]$ sworkgroup --workgroup it_css
Partition
---------
devel
it_css
reserved
standard
vnc
```

Local Commands

`qhost -u <user_name>`

`qhost -h` ***or*** `--help`

A wrapper written for Caviness to consolidate the information collected from Slurm commands and joined together to display the host/node information.

```
[(it_css:traine)@login00 ~]$ qhost -u traine
HOSTNAME   ARCH       NCPU NSOC NCOR NTHR NLOAD  MEMTOT  MEMUSE  SWAPTO  SWAPUS
-------------------------------------------------------------------------------
r00g00  E5-2695v4     72     2   72   72  0.00    124.0G   0.0M   0.0     0.0
r00g01  E5-2695v4     36     2   36   36  1.00    124.0G   8.0G   0.0     0.0
r00g02  E5-2695v4     36     2   36   36  0.06    250.0G   4.0G   0.0     0.0
r00g03  E5-2695v4     36     2   36   36  0.83    502.0G  30.0G   0.0     0.0
r00g04  E5-2695v4     36     2   36   36  0.06    250.0G   4.0G   0.0     0.0
...
...
```

Local Commands

`squota -g <investing_entity>`

`squota -h` ***or*** `--help`

Displays the current utilization of guaranteed (purchased) resources for a workgroup.

```
[(it_css:traine)@login00 ~]$ squota
resource     used limit  pct
~~~~~~~~~~~~~ ~~~~ ~~~~~ ~~~~
cpu             0    72 0.0%
gres/gpu:p100   0    1 0.0%
[(it_css:traine)@login00 ~]$ squota -g it_css
resource     used limit  pct
~~~~~~~~~~~~~ ~~~~ ~~~~~ ~~~~
cpu             0    72 0.0%
gres/gpu:p100   0    1 0.0%
```

Local Commands

```
qstatgrp
qstatgrp -g <investing_entity>
qstatgrp -h or --help
```

Displays the current utilization of resources within a workgroup.

```
[(it_css:traine)@login00 ~]$ qstatgrp -g it_css
PARTITION                        NODES        CPUS     MAX MEM  MAX CPUS
-------------------------------------------------------------------------
vnc                                  1        4
- TOTAL                              1        4
```

Local Commands

`spreempted`

`spreempted --jobid <job-id>{,<job-id>..},`
`-j <job-id>{,<job-id>..}`

Due to a certain level of inconsistency in Slurm's error logging, preempted jobs are notified as FAILED which leads to jobs to immediately exit rather than waiting for a grace period.

```
[(it_css:traine)@login00 ~]$ spreempted -j 410289
preempted, did not reach grace period limit
```

Need Help?

- **Submit** a Research Computing Help Request form
- **Phone**: (302) 831-6000

Please select *High Performance Computing* as the Problem Type, and specify Caviness in the details of your problem in the Description when submitting your request or calling IT Support Center.