Great Lakes

ARC-TS will replace Flux and update other existing clusters. The new research cluster, called Great Lakes will eventually replace Flux. As part of bringing Great Lakes online, the other clusters operated by ARC-TS will be updated to an OS, software, and scheduling environment that matches Great Lakes. In the case of the Flux Operating Environment, this will result in the migration of nodes to Lighthouse, the FOE replacement.

More information on the cluster: https://arc-ts.umich.edu/greatlakes/
User guide: https://arc-ts.umich.edu/greatlakes/user-guide/

Login

To login ssh to “greatlakes.arc-ts.umich.edu”.

```
[user@laptop ~]$ ssh -XY user@greatlakes.arc-ts.umich.edu
*********************************************************************
* By your use of these resources, you agree to abide by Proper Use of  *
* Information Resources, Information Technology, and Networks at the   *
* University of Michigan (SPG 601.07), in addition to all relevant     *
* state and federal laws. http://spg.umich.edu/policy/601.07           *
*********************************************************************

Password:
Duo two-factor login for user

Enter a passcode or select one of the following options:

 1. Duo Push to XXX-XXX-1234
 2. Phone call to XXX-XXX-1234

Passcode or option (1-4): 2
Success. Logging you in...
Last login: Mon Aug 19 11:20:16 2019 from 141.211.22.204
 Welcome to Great Lakes
 Please send feedback to hpc-support@umich.edu
    * Early User documentation:https://arc-ts.umich.edu/greatlakes/early-user/
    * User documentation:https://arc-ts.umich.edu/greatlakes/user-guide/
    * Known issues:https://arc-ts.umich.edu/greatlakes/known-issues/

[user@gl-login1 ~]
```

Software Modules

The software modules on Great Lakes are organized differently from Flux. The software is grouped by compilers and MPI families.

For example, only core software shows up initially:

```
[user@gl-login1 ~]$ module available
------------------------ Core applications including compilers -----------------------
   ANTs/2.1.0              gcc/4.8.5              (D)    mkl/11.1.2
   ANTsR/2.1.0             gcc/8.2.0                     mkl/2018.3.222       (D)
   R/3.5.0                 gsl/2.1                       mricron/30apr2016
   Rtidyverse/3.5.0        hello/1.2                     mrtrix/3.0_RC3
   afni/18.0.27            image-libraries/160610        ncl/6.4.0
   boost/1.67.0            image-libraries/170303 (D)    python-anaconda3/5.2
   cmake/3.5.2             intel/14.0.2                  python-dev/3.6.5
   ffmpeg/3.0.2     (D)    intel/18.0.3           (D)    settarg
   ffmpeg/3.2.4            launcher/3.1.1                yasm/1.3.0
   freesurfer/6.0.0        lmod
   fsl/5.0.11              matlab/R2017b

------------------------------- Software collections --------------------------------
   Bioinformatics    OnCampusAccessOnly    RestrictedLicense

  Where:
   D:  Default Module

Use "module spider" to find all possible modules.
Use "module keyword key1 key2 ..." to search for all possible modules matching any of
the "keys".
```

The correct way to search for software is to use “module spider”:

```
[user@gl-login1 ~]$ module spider openfoam
----------------------------------------------------------------------------
```

```
  openfoam:
------------------------------------------------------------------------------
    Description:
      OpenFOAM is a free, open source CFD software.
     Versions:
        openfoam/extend-4.0
        openfoam/v6-20190620
        openfoam/4.1
------------------------------------------------------------------------------
  For detailed information about a specific "openfoam" module (including how to load the modules) use the
module's full name.
  For example:
     $ module spider openfoam/v6-20190620
------------------------------------------------------------------------------

[user@gl-login1 ~]$ module spider openfoam/v6-20190620
------------------------------------------------------------------------------
  openfoam: openfoam/v6-20190620
------------------------------------------------------------------------------
    Description:
      OpenFOAM is a free, open source CFD software.

    You will need to load all module(s) on any one of the lines below before the "openfoam/v6-20190620"
module is available to load.
      gcc/8.2.0  openmpi/3.1.4

    Help:
      OpenFOAM CFD Toolbox
```

Loading GCC makes software built with GCC available:

```
[user@gl-login1 ~]$ module load gcc
[user@gl-login1 ~]$ module available
--------------------- Applications compiled with GCC 8.2.0 ---------------------
```

```
   boost/1.50.0                    mkl/11.1.2
   boost/1.70.0             (D)    mkl/2018.0.4          (D)
   fftw/3.3.8                      netcdf-c/4.6.2
   gsl/2.5                         netcdf-fortran/4.4.5
   hdf5/1.8.21                     openblas/0.3.5
   image-libraries/190711.1        openmpi/1.10.7
   impi/4.1.3                      openmpi/3.1.4         (D)
   impi/2018.4.274          (D)    szip/2.1.1
   lapack/3.8.0

-------------------- Core applications including compilers ---------------------
   ANTs/2.3.1                      image-libraries/170303
   R/3.5.2                         image-libraries/190711   (D)
   R/3.6.1                (D)      intel/14.0.2
   Rmpi/3.5.2                      intel/18.0.5             (D)
   Rmpi/3.6.1             (D)      jags/4.3
   Ropenblas/3.5.2                 julia/1.1.1
   Ropenblas/3.6.1        (D)      lmod
   Rtidyverse/3.5.2                mathematica/12.0.0
   Rtidyverse/3.6.1       (D)      matlab/R2018b

lines 1-25
```

Then, loading OpenMPI makes software built with OpenMPI available:

```
[user@gl-login1 ~]$ module load openmpi/3.1.4
[user@gl-login1 ~]$ module available
---------- /sw/arcts/centos7/modulefiles/MPI/openmpi-3_1_4/gcc-8_2_0 -----------
   openfoam/v6-20190620

-------------------- Applications compiled with GCC 8.2.0 ---------------------
   boost/1.50.0                    mkl/11.1.2
   boost/1.70.0             (D)    mkl/2018.0.4          (D)
   fftw/3.3.8                      netcdf-c/4.6.2
```

```
   gsl/2.5                          netcdf-fortran/4.4.5
   hdf5/1.8.21                      openblas/0.3.5
   image-libraries/190711.1         openmpi/1.10.7
   impi/4.1.3                       openmpi/3.1.4        (L,D)
   impi/2018.4.274          (D)     szip/2.1.1
   lapack/3.8.0

------------------- Core applications including compilers ---------------------
   ANTs/2.3.1                       image-libraries/170303
   R/3.5.2                          image-libraries/190711   (D)
   R/3.6.1                (D)       intel/14.0.2
   Rmpi/3.5.2                       intel/18.0.5             (D)
   Rmpi/3.6.1             (D)       jags/4.3
   Ropenblas/3.5.2                  julia/1.1.1
   Ropenblas/3.6.1        (D)       lmod
   Rtidyverse/3.5.2                 mathematica/12.0.0
   Rtidyverse/3.6.1       (D)       matlab/R2018b
lines 1-25
```

Software Collections

Specialized software is listed under “Software Collections”. Software available for use on-campus only is under the “OnCampusAccessOnly” module, and likewise package with special license restrictions is under “RestrictedLicense”. More details about software collections and conditions for use will be provided at a later date.

Cluster Resources and Job Scheduling

Slurm is an open source cluster management and job scheduling system for Linux clusters. Great Lakes has Slurm setup with enough resources to allow all users to run small test jobs.

ARC-TS Slurm User Guide: http://arc-ts.umich.edu/slurm-user-guide/
From Torque to Slurm: http://arc-ts.umich.edu/migrating-from-torque-to-slurm/

Identifying Resources

To list the resource accounts you have access to do the following:

```
[user@gl-login1 ~]$ my_accounts

{
  Account         : "earlyuser"
  MaxWall         : "2-00:00:00"
  QOS             : "interactive,normal"
}

{
  Account         : "engin"
  QOS             : "interactive,normal"
}
```

The account “earlyuser” has limits:

- 2 days wall clock time

Users without assigned resources will result in no output:

```
[user@gl-login1 ~]$ my_accounts none
```

This is an example for another user:

```
[user@gl-login1 ~]$ my_accounts mmiranda

{
  Account         : "support"
  QOS             : "interactive,normal"
}

{
  Account         : "engin"
  QOS             : "interactive,normal"
}
```

To show your usage in CPU-minutes:

```
[user@gl-login1 ~]$ my_usage
--------------------------------------------------------------------------------
Cluster/Account/User Utilization 2001-01-01T00:00:00 - 2018-10-31T23:59:59 (562719600 secs)
Usage reported in CPU Minutes
--------------------------------------------------------------------------------
  Cluster         Account     Login     Proper Name     Used   Energy
--------- --------------- --------- --------------- -------- --------
greatlak+            test      user Some Linux User      402        0
```

The command may not be useful now, but once ACR-TS starts charging for Great Lakes, it will make more sense.

The cluster is organized into sets of nodes called partition. A node may belong to more than one partition. To lookup partitions:

```
[user@gl-login1 ~]$ sinfo
PARTITION   AVAIL  TIMELIMIT  NODES  STATE NODELIST
largemem       up 14-00:00:0      1  alloc gl0000
largemem       up 14-00:00:0      2   idle gl[0001-0002]
gpu            up 14-00:00:0      8    mix gl[1000-1007]
gpu            up 14-00:00:0     12   idle gl[1008-1019]
viz            up 14-00:00:0      4   idle gl[2000-2003]
```

```
standard*      up 14-00:00:0      6  down* gl[3124,3216,3234,3292,3320,3350]
standard*      up 14-00:00:0     30   drng gl[3165-3194]
standard*      up 14-00:00:0     35   resv gl[3011-3041,3160-3163]
standard*      up 14-00:00:0      7    mix gl[3132-3134,3233,3242,3304,3307]
standard*      up 14-00:00:0    133  alloc
gl[3009-3010,3125-3131,3195-3215,3217-3232,3293-3303,3305-3306,3308-3319,3321-3349,3351-3383]
standard*      up 14-00:00:0    163   idle gl[3042-3123,3135-3159,3235-3241,3243-3291]
standard-oc    up 14-00:00:0      9   idle gl[3000-3008]
```

For data on specific nodes:

```
[user@gl-login1 ~]$ scontrol show node gl3042
NodeName=gl3042 Arch=x86_64 CoresPerSocket=9
   CPUAlloc=0 CPUTot=36 CPULoad=0.01
   AvailableFeatures=(null)
   ActiveFeatures=(null)
   Gres=(null)
   NodeAddr=gl3042 NodeHostName=gl3042 Version=18.08
   OS=Linux 3.10.0-957.10.1.el7.x86_64 #1 SMP Mon Mar 18 15:06:45 UTC 2019
   RealMemory=184320 AllocMem=0 FreeMem=183193 Sockets=4 Boards=1
   State=IDLE ThreadsPerCore=1 TmpDisk=0 Weight=1 Owner=N/A MCS_label=N/A
   Partitions=standard
   BootTime=2019-08-07T16:01:17 SlurmdStartTime=2019-08-20T09:24:39
   CfgTRES=cpu=36,mem=180G,billing=15869
   AllocTRES=
   CapWatts=n/a
   CurrentWatts=0 LowestJoules=0 ConsumedJoules=0
   ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s
   Reason=New Build or Unknown
```

Node “gl3042” currently idle and has 36 cores with 180GB of usable memory.

In the example below node “gl1008” is similar except it also has 2 GPUs.

```
[user@gl-login1 ~]$ scontrol show node gl1008
```

```
NodeName=gl1008 Arch=x86_64 CoresPerSocket=10
   CPUAlloc=0 CPUTot=40 CPULoad=0.05
   AvailableFeatures=(null)
   ActiveFeatures=(null)
   Gres=gpu:tesla:2
   NodeAddr=gl1008 NodeHostName=gl1008 Version=18.08
   OS=Linux 3.10.0-957.10.1.el7.x86_64 #1 SMP Mon Mar 18 15:06:45 UTC 2019
   RealMemory=184320 AllocMem=0 FreeMem=184242 Sockets=4 Boards=1
   State=IDLE ThreadsPerCore=1 TmpDisk=0 Weight=1 Owner=N/A MCS_label=N/A
   Partitions=gpu
   BootTime=2019-08-08T00:24:59 SlurmdStartTime=2019-08-20T09:24:38
   CfgTRES=cpu=40,mem=180G,billing=10100,gres/gpu=2
   AllocTRES=
   CapWatts=n/a
   CurrentWatts=0 LowestJoules=0 ConsumedJoules=0
   ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s
```

Job Submission

Slurm has more options than PBS, but job submission scripts can be very simple.

Slurm scripts use the directive “#SBATCH” for options. Options are in the form “--option=value”. Note there should be no spaces right before the “=” or any place after it.

A simple example Slurm job:

```
[user@gl-login1 test]$ cp /sw/coe/centos7/examples/intro-gl/simple.sbat ./
[user@gl-login1 test]$ cat simple.sbat
#!/bin/bash
#“#SBATCH” directives that convey submission options:
##### The name of the job
#SBATCH --job-name=JOBNAME
##### When to send e-mail: pick from NONE, BEGIN, END, FAIL, REQUEUE, ALL
#SBATCH --mail-type=END,FAIL
```

```
##### Resources for your job
# number of physical nodes
#SBATCH --nodes=1
# number of task per a node (number of CPU-cores per a node)
#SBATCH --ntasks-per-node=1
# memory per a CPU-core
#SBATCH --mem-per-cpu=1000m
##### Maximum amount of time the job will be allowed to run
#####    Recommended formats:  MM:SS, HH:MM:SS, DD-HH:MM
#SBATCH --time=5:00
##### The resource account; who pays
#SBATCH --account=test
########## End of preamble! ########################################
# No need to “cd”. Slurm starts the job in the submission directory.
####################################################################
# The application(s) to execute along with its input arguments and options:
/bin/hostname
sleep 120
echo “done!”
```

Edit the account to be earlyuser or training. Anything listed for you by `my_accounts` should also work.

To submit a job use “sbatch”:

```
[user@gl-login1 test]$ sbatch simple.sbat
Submitted batch job 337
```

The default output file for stdout/stderr is “slurm-<jobnumer>.out”:

```
[user@gl-login1 test]$ ls
simple.sbat  slurm-337.out
[user@gl-login1 test]$ cat slurm-337.out
gl3101.arc-ts.umich.edu
[user@gl-login1 test]$ sleep 120; cat slurm-337.out
gl3101.arc-ts.umich.edu
“done!”
```

Changing Default Job Submission Options

Some of the “#SBATCH” for options are defaults. You do not need to set them, since they are set for you, but you may want to change them in some cases.

Output File stdout/stderr

The name of the stdout/stderr output file is set by default. For example the default output file for job 337 is “slurm-337.out”. If it was to be “JOBNAME-337.log”, the submission script would include this line:

```
#SBATCH --output=%x-%j.log
```

More details for using this option are in the sbatch man page, “filename pattern” section. Make sure that the output directory exists **before** you submit the job! Do not use any directory like /tmp, which is not the same filesystem across all the nodes.

Partition

The default partition is “standard”, except on Lighthouse, and most of the time you do not need to change it. The partitions are used to divide the nodes into groups for special uses. For example, the “gpu” partition has nodes with GPUs, and the “debug” is setup to allow one job per a user to start up faster. This would be added to your script for using the GPU partition:

```
#SBATCH --partition=gpu
#SBATCH --gres=gpu:1
```

The gres is used to request the GPU hardware, which is one GPU in this case.

Lighthouse does not have a default partition, so it must be specified. The partition must match the account on Lighthouse, for example:

```
#SBATCH --account=support
#SBATCH --partition=support
```

Mail User

If email is requested, the default is to send it to the user, that is uniqname@umich.edu. It may be changed by adding this option and specifying the desired email address:

```
#SBATCH --mail-user=username@umich.edu
```

Job Status and Cancellation

Jobs status is shown with “squeue” or “scontrol show job”, and a job maybe canceled with “scancle”.
This is a simple example:

```
[user@gl-login1 test]$ sbatch --test-only simple.sbat
sbatch: Job 337 to start at 2018-10-24T13:40:49 using 1 processors on nodes gl3101 in partition standard
```

Job 337 did not actually run, but the batch script was validated. The returned message is an estimate of when a job would be scheduled to run given the current state of the cluster.

```
[user@gl-login1 test]$ sbatch simple.sbat
Submitted batch job 338
[user@gl-login1 test]$ sbatch simple.sbat
Submitted batch job 339
[user@gl-login1 test]$ sbatch simple.sbat
Submitted batch job 340
[user@gl-login1 test]$ squeue -u $USER
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
               340  standard  JOBNAME     user PD       0:00      1 (AssocMaxJobsLimit)
               339  standard  JOBNAME     user R        0:04      1 gl3101
               338  standard  JOBNAME     user R        0:08      1 gl3101
```

ST stands for status or state. The following are common states:

- CA CANCELLED Job was explicitly cancelled by the user or system administrator.
- CD COMPLETED Job has terminated all processes on all nodes with an exit code of zero.
- PD PENDING Job is awaiting resource allocation.
- R RUNNING Job currently has an allocation.
- S SUSPENDED Job has an allocation, but execution has been suspended and CPUs have been released for other jobs.

```
[user@gl-login1 test]$ scancel 340
[user@gl-login1 test]$ squeue -u $USER
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
               339  standard  JOBNAME     user R        0:48      1 gl3101
               338  standard  JOBNAME     user R        0:52      1 gl3101
```

The following is an example of detailed job status:

```
[user@gl-login1 test]$ sbatch simple.sbat
```

```
Submitted batch job 374
[user@gl-login1 test]$ scontrol show job 374
JobId=374 JobName=JOBNAME
   UserId=user(9876543210) GroupId=users(22) MCS_label=N/A
   Priority=102687 Nice=0 Account=test QOS=normal WCKey=*
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:09 TimeLimit=00:05:00 TimeMin=N/A
   SubmitTime=2018-10-25T13:27:51 EligibleTime=2018-10-25T13:27:52
   AccrueTime=2018-10-25T13:27:52
   StartTime=2018-10-25T13:27:52 EndTime=2018-10-25T13:32:52 Deadline=N/A
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   LastSchedEval=2018-10-25T13:27:52
   Partition=standard AllocNode:Sid=gl-login1:42919
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=gl3101
   BatchHost=bgl3101
   NumNodes=1 NumCPUs=1 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=1,mem=1000M,node=1,billing=1001
   Socks/Node=* NtasksPerN:B:S:C=1:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryCPU=1000M MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/home/user/test/simple.sbat
   WorkDir=/home/user/test
   StdErr=/home/user/test/slurm-374.out
   StdIn=/dev/null
   StdOut=/home/user/test/slurm-374.out
   Power=
```

If you run it after the job is finished the result maybe an error message:

```
[user@gl-login1 test]$ scontrol show job 374
slurm_load_jobs error: Invalid job id specified
```

Interactive Job:

If you run it after the job is finished the result maybe an error message:

```
[user@gl-login1 ~]$ srun --pty --nodes=1 --cpus-per-task=4 --time=30:00 --account=training /bin/bash
[mmiranda@gl3377 mpi-example]$ exit
exit
[mmiranda@gl-login2 mpi-example]$
```

MPI example:

First copy the example source code:

```
[user@gl-login1 ~]$ cp -rv /sw/coe/centos7/examples/intro-gl/mpi-example/ ./
‘/sw/coe/centos7/examples/intro-gl/mpi-example/mpi-hello.c’ -> ‘./mpi-example/mpi-hello.c’
‘/sw/coe/centos7/examples/intro-gl/mpi-example/mpi-hello.f90’ -> ‘./mpi-example/mpi-hello.f90’
‘/sw/coe/centos7/examples/intro-gl/mpi-example/Makefile’ -> ‘./mpi-example/Makefile’
‘/sw/coe/centos7/examples/intro-gl/mpi-example/hello.f90’ -> ‘./mpi-example/hello.f90’
‘/sw/coe/centos7/examples/intro-gl/mpi-example/hello.c’ -> ‘./mpi-example/hello.c’
‘/sw/coe/centos7/examples/intro-gl/mpi-example/hello.h’ -> ‘./mpi-example/hello.h’
[user@gl-login1 ~]$ cd ./mpi-example/
[user@gl-login1 mpi-example]$ ls
hello.c  hello.f90  hello.h  Makefile  mpi-hello.c  mpi-hello.f90
```

Second, load the Intel compiler and OpenMPI modules:

```
[user@gl-login1 mpi-example]$ module load intel openmpi
```

Next, build the executable:

```
[user@gl-login1 mpi-example]$ make  mpi_c_hello
mpicc -O2 -xHost -o mpi-hello mpi-hello.c
[user@gl-login1 mpi-example]$ ls
hello.c  hello.f90  hello.h  Makefile  mpi-hello  mpi-hello.c  mpi-hello.f90
```

Then, test the executable before submitting a job:

```
[user@gl-login1 mpi-example]$ srun -n 4 mpi-hello
0: We have 4 processors
0: Processor 1. Are you there?
```

```
1: Yes'm.  Processor 1 reporting for duty
0: Processor 2. Are you there?
2: Yes'm.  Processor 2 reporting for duty
0: Processor 3. Are you there?
3: Yes'm.  Processor 3 reporting for duty
0: Everybody's here. Let's get this show on the road.
```

Now, copy the submission script:

```
[user@gl-login1 mpi-example]$ cp /sw/coe/centos7/examples/intro-gl/mpi.sbat ./
[user@gl-login1 mpi-example]$ cat mpi.sbat
#!/bin/bash
# The interpreter used to execute the script:
#“#SBATCH” directives that convey submission options:
##### The name of the job
#SBATCH --job-name=mpi-hello
##### When to send e-mail: pick from NONE, BEGIN, END, FAIL, REQUEUE, ALL
#SBATCH --mail-type=END,FAIL
##### Resources for your job
# number of physical nodes
#SBATCH --nodes=2
# number of task per a node (number of CPU-cores per a node)
#SBATCH --ntasks-per-node=2
# memory per a CPU-core
#SBATCH --mem-per-cpu=1000m
##### Maximum amount of time the job will be allowed to run
#####    Recommended formats:  MM:SS, HH:MM:SS, DD-HH:MM
#SBATCH --time=5:00
##### The resource account; who pays
#SBATCH --account=test
########## End of preamble! ##########################################
# No need to “cd”. Slurm starts the job in the submission directory.
######################################################################
# The application(s) to execute along with its input arguments and options:
```

```
echo "Hostname:"
/bin/hostname
if [[ $SLURM_JOB_NODELIST ]] ; then
    echo "SLURM_JOB_NODELIST=$SLURM_JOB_NODELIST"
    echo "Nodes:"
    scontrol show hostnames $SLURM_JOB_NODELIST
    echo "Node for each core:"
    srun hostname
fi
echo
echo "Actual work we need to do:"
echo "mpirun or mpiexec :"
mpirun mpi-hello
echo
echo "srun is the Slurm way to run MPI:"
srun mpi-hello
```

Finally submit the job:

```
[user@gl-login1 mpi-example]$ sbatch mpi.sbat
```