MAVERICK2 USER GUIDE

Last update: May 3, 2023

Notices

- **Maverick2 will be retired from production on May 31, 2023. Please contact TACC User Services with any questions.** (05/01/2023)
- Maverick2 is TACC's dedicated Deep Learning Machine. Allocation requests must include a justification explaining your need for this resource.
- Maverick2 does not support any Visualization applications.
- Maverick2 does not mount a `/scratch` (`$SCRATCH`) file system.

Introduction

Maverick2 is an extension to TACC's services to support GPU accelerated Machine Learning and Deep Learning research workloads. The power of this system is in its multiple GPUs per node and it is mostly intended to support workloads that are better supported with a dense cluster of GPUs and little CPU compute. The system is designed to support model training via GPU powered frameworks that can take advantage of the 4 GPUs in a node. In addition to the 96 1080-TI Nvidia GPU cards, a limited number of Pascal 100 and Volta 100 cards are available to support any workloads that cannot be done in the smaller memory footprints of the primary GPU cards. The system software supports Tensorflow and Caffe and can also be augmented to run other frameworks.

System Overview

Maverick2 hosts the following GPUs: 24 nodes each with 4 NVidia GTX 1080 Ti GPUs running in a Broadwell based compute node; four nodes each with two of NVidia V100s GPUs running in a Skylake based Dell R740 based node; and three nodes each with two NVidia P100s GPUs running in a Skylake based Dell R740 node.

GTX Compute Nodes

Maverick2 hosts 24 GTX compute nodes. One GTX node is reserved for staff use, leaving 23 nodes available for general use.

Table 1. GTX Compute Node Specifications

Specification	Value
Model:	Super Micro X10DRG-Q Motherboard
Processor:	Intel(R) Xeon(R) CPU E5-2620 v4
Total processors per node:	2
Total cores per processor:	8
Total cores per node:	16
Hardware threads per core:	2

Specification	Value
Hardware threads per node:	32
Clock rate:	2.10GHz
RAM:	128 GB
L1/L2/L3 Cache:	512KiB / 2MiB / 20 MiB
Local storage:	150.0 GB (~60 GB free)
GPUs:	4 x NVidia 1080-TI GPUs

V100 Compute Nodes

Maverick2 has 4 V100 compute nodes.

Table 2. V100 Compute Node Specifications

Specification	Value
Model:	Dell PowerEdge R740
Processor:	Xeon(R) Platinum 8160 CPU @ 2.10GHz
Total processors per node:	2
Total cores per processor:	24
Total cores per node:	48
Hardware threads per core:	2
Hardware threads per node:	96
Clock rate:	2.10GHz
RAM:	192 GB
L1/L2/L3 Cache:	1536KiB / 24576KiB / 33792KiB
Local storage:	119.5 GB (~32 GB free)
GPUs:	2 NVidia V100 adapters

P100 Compute Nodes

Maverick2 has 3 P100 nodes.

Table 3. P100 Compute Node Specifications

Specification	Value
Model:	Dell PowerEdge R740
Processor:	Xeon(R) Platinum 8160 CPU @ 2.10GHz
Total processors per node:	2

Specification	Value
Total cores per processor:	24
Total cores per node:	48
Hardware threads per core:	2
Hardware threads per node:	96
Clock rate:	2.10GHz
RAM:	192 GB
L1/L2/L3 Cache:	1536KiB / 24576KiB / 33792KiB
Local storage:	119.5 GB (~32 GB free)
GPUs:	2 NVidia P100 adapters

Login Nodes

Maverick2 hosts a single login node:

- Dual Socket
- Intel Xeon CPU E5-2660 v3 (Haswell) @ 2.60GHz: 10 cores/socket (20 cores/node)
- 128 GB DDR4-2133 (8 x 16GB dual rank x4 DIMMS)
- Hyperthreading Disabled

Network

- Mellanox FDR Infiniband MT27500 Family ConnectX-3 Adapter
- up to 10/40/56Gbps bandwidth and a sub-microsecond low latency
- Fat Tree Interconnect
- Intel Ethernet Controller I350 IEEE 802.3 1Gbps Adapter

File Systems

Maverick2 mounts two shared Lustre file systems on which each user has corresponding account-specific directories `$HOME` and `$WORK`. **Unlike most TACC resources, Maverick2 does not mount a `/scratch` file system.** Both the `/home` and `/work` file systems are available from all Maverick2 nodes; the Stockyard-hosted `/work` file system is available on other TACC systems as well. A Lustre file system looks and acts like a single logical hard disk, but is actually a sophisticated integrated system involving many physical drives (dozens of physical drives for `$HOME` and thousands for `$WORK`).

Accessing the System

Access to all TACC systems now requires Multi-Factor Authentication (MFA). You can create an MFA pairing on the TACC User Portal. After login on the portal, go to your account profile (Home->Account Profile), then click the "Manage" button under "Multi-Factor Authentication" on the right side of the page. See Multi-Factor Authentication at TACC for further information.

Secure Shell (SSH)

The `ssh` command (SSH protocol) is the standard way to connect to Maverick2. SSH also includes support for the file transfer utilities `scp` and `sftp`. Wikipedia is a good source of information on SSH. SSH is available within Linux and from the terminal app in the Mac OS. If you are using Windows, you will need an SSH client that supports the SSH-2 protocol: e.g.

Bitvise, OpenSSH, PuTTY, or SecureCRT. Initiate a session using the `ssh` command or the equivalent; from the Linux command line the launch command looks like this:

```
localhost$ ssh username@maverick2.tacc.utexas.edu
```

Use your TUP password for direct logins to Maverick2. **Only users with an allocation on Maverick2 may log on.** You can change your TACC password through the TACC User Portal. Log into the portal, then select "Change Password" under the "HOME" tab. If you've forgotten your password, go to the TACC User Portal home page and select "Password Reset" under the Home tab.

To report a connection problem, execute the `ssh` command with the `-vvv` option and include the verbose output when submitting a help ticket.

Managing Your Files

Maverick2 mounts two Lustre file systems that are shared across all nodes: the home and work file systems. Maverick2's startup mechanisms define corresponding account-level environment variables, `$HOME` and `$WORK`, that store the paths to directories that you own on each of these file systems. Maverick2's home file system is mounted only on Maverick2, but the work file system mounted on Maverick2 is the Global Shared File System hosted on Stockyard. This is the same work file system that is currently available on Stampede2, Frontera, Lonestar6, and several other TACC resources.

Table 4. File Systems

File System	Quota	Key Features
`$HOME`	10GB, 200,000 files	**Not intended for parallel or high-intensity file operations.** Backed up regularly. Overall capacity ~1PB. NFS-mounted. Two Meta-Data Servers (MDS), four Object Storage Targets (OSTs). Defaults: 1 stripe, 1MB stripe size. Not purged.
`$WORK`	1TB, 3,000,000 files across all TACC systems, regardless of where on the file system the files reside.	**Not intended for high-intensity file operations or jobs involving very large files.** On the Global Shared File System that is mounted on most TACC systems. See Stockyard system description for more information. Defaults: 1 stripe, 1MB stripe size Not backed up. Not purged.
`$SCRATCH`	**N/A**	**Maverick2 does not mount a scratch file system.**

The `$STOCKYARD` environment variable points to the highest-level directory that you own on the Global Shared File System. The definition of the `$STOCKYARD` environment variable is of course account-specific, but you will see the same value on all TACC systems that provide access to the Global Shared File System (see Figure 3). This directory is an excellent place to store files you want to access regularly from multiple TACC resources.

Navigating the Shared File Systems

Your account-specific `$WORK` environment variable varies from system to system and (except for the decommissioned Stampede1 system) is a sub-directory of `$STOCKYARD` (Figure 3). The sub-directory name corresponds to the associated TACC resource. The `$WORK` environment variable on Maverick2 points to the `$STOCKYARD/maverick2` subdirectory, a convenient location for files you use and jobs you run on Maverick2. Remember, however, that all subdirectories contained in your `$STOCKYARD` directory are available to you from any system that mounts the file system. If you have accounts on both Maverick2 and Stampede2, for example, the `$STOCKYARD/maverick2` directory is available from your Stampede2 account, and `$STOCKYARD/stampede2` is available from your Maverick2 account. Your quota and reported usage on the Global Shared File System reflects all files that you own on Stockyard, regardless of their actual location on the file system.

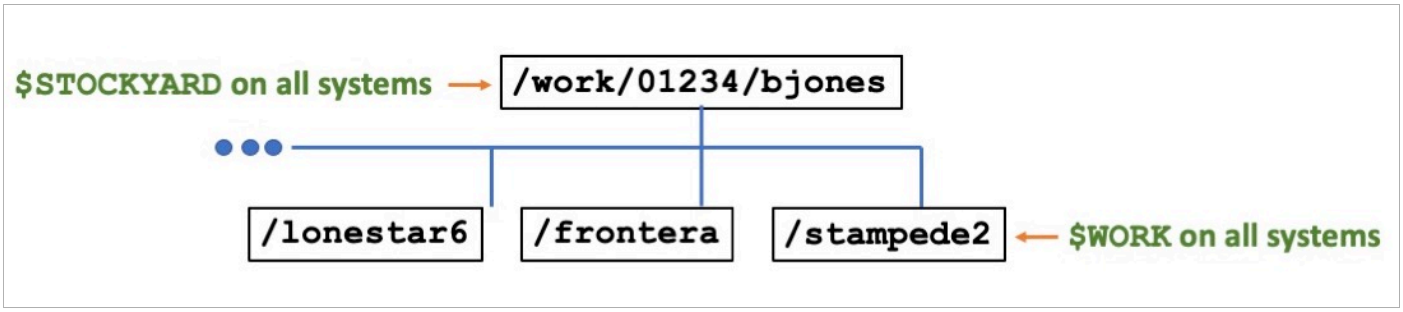

Figure 3. Account-level directories on the work file system (Global Shared File System hosted on Stockyard). Example for fictitious user `bjones`. All directories usable from all systems. Sub-directories (e.g. `frontera`, `maverick2`) exist only when you have allocations on the associated system.

Note that resource-specific sub-directories of `$STOCKYARD` are nothing more than convenient ways to manage your resource-specific files. You have access to any such sub-directory from any TACC resources. If you are logged into Maverick2, for example, executing the alias `cdw` (equivalent to `cd $WORK`) will take you to the resource-specific sub-directory `$STOCKYARD/maverick2`. But you can access this directory from other TACC systems as well by executing `cd $STOCKYARD/maverick2`. These commands allow you to share files across TACC systems. In fact, several convenient account-level aliases make it even easier to navigate across the directories you own in the shared file systems:

Table 5. Built-in Account Level Aliases

Alias	Command
`cd` or `cdh`	`cd $HOME`
`cdw`	`cd $WORK`
`cdy` or `cdg`	`cd $STOCKYARD`

Transferring Files Using `scp` and `rsync`

You can transfer files between Maverick2 and Linux-based systems using either `scp` or `rsync`. Both `scp` and `rsync` are available in the Mac Terminal app. Windows ssh clients typically include `scp`-based file transfer capabilities.

The Linux `scp` (secure copy) utility is a component of the OpenSSH suite. Assuming your Maverick2 username is `bjones`, a simple `scp` transfer that pushes a file named `myfile` from your local Linux system to Maverick2 `$HOME` would look like this:

```
localhost$ scp ./myfile bjones@maverick2.tacc.utexas.edu:  # note colon after net address
```

You can use wildcards, but you need to be careful about when and where you want wildcard expansion to occur. For example, to push all files ending in `.txt` from the current directory on your local machine to `/work/01234/bjones/scripts` on Maverick2:

```
localhost$ scp *.txt bjones@maverick2.tacc.utexas.edu:/work/01234/bjones/maverick2
```

To delay wildcard expansion until reaching Maverick2, use a backslash (`\`) as an escape character before the wildcard. For example, to pull all files ending in `.txt` from `/work/01234/bjones/scripts` on Maverick2 to the current directory on your local system:

```
localhost$ scp bjones@maverick2.tacc.utexas.edu:/work/01234/bjones/maverick2/\*.txt .
```

You can of course use shell or environment variables in your calls to `scp`. For example:

```
localhost$ destdir="/work/01234/bjones/maverick2/data"
localhost$ scp ./myfile bjones@maverick2.tacc.utexas.edu:$destdir
```

You can also issue `scp` commands on your local client that use Maverick2 environment variables like `$HOME` and `$WORK`. To do so, use a backslash (`\`) as an escape character before the `$`; this ensures that expansion occurs after establishing the connection to Maverick2:

```
localhost$ scp ./myfile bjones@maverick2.tacc.utexas.edu:\$WORK/data   # Note backslash
```

Danger

Avoid using `scp` for recursive (`-r`) transfers of directories that contain nested directories of many small files:

```
localhost$ scp -r  ./mydata     bjones@maverick2.tacc.utexas.edu:\$WORK  # DON'T DO THIS
```

Hint

Instead, use `tar` to create an archive of the directory, then transfer the directory as a single file:

```
localhost$ tar cvf ./mydata.tar mydata                                  # create archive
localhost$ scp     ./mydata.tar bjones@maverick2.tacc.utexas.edu:\$WORK  # transfer archive
```

The `rsync` (remote synchronization) utility is a great way to synchronize files that you maintain on more than one system: when you transfer files using `rsync`, the utility copies only the changed portions of individual files. As a result, `rsync` is especially efficient when you only need to update a small fraction of a large dataset. The basic syntax is similar to `scp`:

```
localhost$ rsync       mybigfile bjones@maverick2.tacc.utexas.edu:\$WORK/data
localhost$ rsync -avtr mybigdir  bjones@maverick2.tacc.utexas.edu:\$WORK/data
```

The options on the second transfer are typical and appropriate when synching a directory: this is a recursive update (`-r`) with verbose (`-v`) feedback; the synchronization preserves time stamps (`-t`) as well as symbolic links and other meta-data (`-a`). Because `rsync` only transfers changes, recursive updates with `rsync` may be less demanding than an equivalent recursive transfer with `scp`.

See the Good Conduct document for additional important advice about striping the receiving directory when transferring large files; watching your quota on `$HOME` and `$WORK`; and limiting the number of simultaneous transfers. Remember also that `$STOCKYARD` (and your `$WORK` directory on each TACC resource) is available from several other TACC systems: there's no need for `scp` when both the source and destination involve sub-directories of `$STOCKYARD`. See Managing Your Files for more information about transfers on `$STOCKYARD`.

Sharing Files with Collaborators

If you wish to share files and data with collaborators in your project, see Sharing Project Files on TACC Systems for step-by-step instructions. Project managers or delegates can use Unix group permissions and commands to create read-only or read-write shared workspaces that function as data repositories and provide a common work area to all project members.

Notes on Small Files Under Lustre

The Stockyard/`$WORK` file system is a Lustre file system which is optimized for large scale reads and writes. As some workloads, such as image classification, leverage using multiple small files, we advise users not work directly on `$WORK` with these workloads. Users should have their jobs copy these files to `/tmp` on the compute node, compute against the `/tmp` data, store their results on the `$WORK` file system, and clean up `/tmp`. We are currently working on solutions to expand the 60 GB `/tmp` capacity.

Striping Large Files

Lustre can **stripe** (distribute) large files over several physical disks, making it possible to deliver the high performance needed to service input/output (I/O) requests from hundreds of users across thousands of nodes. Object Storage Targets (OSTs) manage the file system's spinning disks: a file with 20 stripes, for example, is distributed across 20 OSTs. One designated Meta-Data Server (MDS) tracks the OSTs assigned to a file, as well as the file's descriptive data.

Before transferring large files to Maverick2, or creating new large files, be sure to set an appropriate default stripe count on the receiving directory. To avoid exceeding your fair share of any given OST, a good rule of thumb is to allow at least one stripe for each 100GB in the file. For example, to set the default stripe count on the current directory to 30 (a plausible stripe count for a directory receiving a file approaching 3TB in size), execute:

```
$ lfs setstripe -c 30 $PWD
```

Note that an `lfs setstripe` command always sets both stripe count and stripe size, even if you explicitly specify only one or the other. Since the example above does not explicitly specify stripe size, the command will set the stripe size on the directory to Maverick2's system default (1MB). In general there's no need to customize stripe size when creating or transferring files.

Remember that it's not possible to change the striping on a file that already exists. Moreover, the `mv` command has no effect on a file's striping if the source and destination directories are on the same file system. You can, of course, use the `cp` command to create a second copy with different striping; to do so, copy the file to a directory with the intended stripe parameters.

Running Jobs

Job Accounting

Like all TACC systems, Maverick2's accounting system is based on node-hours: one unadjusted Service Unit (SU) represents a single compute node used for one hour (a node-hour). For any given job, the total cost in SUs is the use of one compute node for one hour of wall clock time plus any charges or discounts for the use of specialized queues, e.g. Frontera's `flex` queue, Stampede2's `development` queue, and Longhorn's `v100` queue. The queue charge rates are determined by the supply and demand for that particular queue or type of node used and are subject to change.

Maverick2 SUs billed = (# nodes) x (job duration in wall clock hours) x (charge rate per node-hour)

The Slurm scheduler tracks and charges for usage to a granularity of a few seconds of wall clock time. **The system charges only for the resources you actually use, not those you request.** If your job finishes early and exits properly, Slurm will release the nodes back into the pool of available nodes. Your job will only be charged for as long as you are using the nodes.

Note

TACC does not implement node-sharing on any compute resource. Each Maverick2 node can be assigned to only one user at a time; hence a complete node is dedicated to a user's job and accrues wall-clock time for all the node's cores whether or not all cores are used.

Tip

Your queue wait times will be less if you request only the time you need: the scheduler will have a much easier time finding a slot for the 2 hours you really need than say, for the 12 hours requested in your job script.

Principal Investigators can monitor allocation usage via the TACC User Portal under "Allocations->Projects and Allocations". Be aware that the figures shown on the portal may lag behind the most recent usage. Projects and allocation balances are also displayed upon command-line login.

Tip

To display a summary of your TACC project balances and disk quotas at any time, execute:

```
login1$ /usr/local/etc/taccinfo       # Generally more current than balances displayed on the portals.
```

Slurm Job Scheduler

Maverick2 employs the Slurm Workload Manager job scheduler. Slurm commands enable you to submit, manage, monitor, and control your jobs.

The Stampede2 User Guide discusses Slurm extensively. See the following sections for detailed information:

- Submitting Jobs with `sbatch`
- Common `sbatch` options
- Launching Applications

Slurm Partitions (Queues)

Queues and limits are subject to change without notice.

Execute `qlimits` on Maverick2 for real-time information regarding limits on available queues.

See Stampede2's Monitoring Jobs and Queues section for additional information.

Table 6. Maverick2 Production Queues

Queue Name (available nodes)	Max Nodes per Job (assoc'd cores)	Max Duration	Max Jobs in Queue	Charge Rate (per node-hour)
`gtx` (24 nodes)	4 nodes (64 cores)	24 hours	4	1 SU
`v100` (4 nodes)	4 nodes (192 cores)	24 hours	4	1 SU
`p100` (3 nodes)	3 nodes (144 cores)	24 hours	4	1 SU

Job Scripts

This section provides sample Slurm job scripts for commond programming models: serial applications, MPI, OpenMP and hybrid (MPI + OpenMP) programming models.

Click on a tab for a job-script. Copy and customize for your own applications.

Serial Job in Normal Queue | MPI Job in Normal Queue | OpenMP Job in Normal Queue | Hybrid Job in Normal Queue

Your job will run in the environment it inherits at submission time; this environment includes the modules you have loaded and the current working directory. In most cases you should **run your applications(s) after loading the same modules that you used to build them**. You can of course use your job submission script to modify this environment by defining new environment variables; changing the values of existing environment variables; loading or unloading modules; changing

directory; or specifying relative or absolute paths to files. **Do not use the Slurm `--export` option to manage your job's environment**: doing so can interfere with the way the system propagates the inherited environment.

Remote Desktop Access

Remote desktop access to Maverick2 is formed through a VNC connection to one or more visualization nodes. Users must first connect to a Maverick2 login node (see System Access) and submit a special interactive batch job that:

- allocates a set of Maverick2 visualization nodes
- starts a vncserver process on the first allocated node
- sets up a tunnel through the login node to the vncserver access port

Once the vncserver process is running on the visualization node and a tunnel through the login node is created, an output message identifies the access port for connecting a VNC viewer. A VNC viewer application is run on the user's remote system and presents the desktop to the user.

Tip

If this is your first time connecting to Maverick2, you must run `vncpasswd` to create a password for your VNC servers. This should NOT be your login password! This mechanism only deters unauthorized connections; it is not fully secure, as only the first eight characters of the password are saved. All VNC connections are tunneled through SSH for extra security, as described below.

Follow the steps below to start an interactive session.

1. Start a Remote Desktop

   TACC has provided a VNC job script (`/share/doc/slurm/job.vnc`) that requests one node in the `development` queue for two hours, creating a VNC session.

   ```
   login1$ sbatch /share/doc/slurm/job.vnc
   ```

   You may modify or overwrite script defaults with `sbatch` command-line options:

   - `-t hours:minutes:seconds` modify the job runtime
   - `-A projectnumber` specify the project/allocation to be charged
   - `-N nodes` specify number of nodes needed
   - `-p partition` specify an alternate queue.

   See more `sbatch` options in the Common `sbatch` Options section of the Stampede2 resource guide.

   All arguments after the job script name are sent to the vncserver command. For example, to set the desktop resolution to 1440x900, use:

   ```
   login1$ sbatch /share/doc/slurm/job.vnc -geometry 1440x900
   ```

   The `vnc.job` script starts a vncserver process and writes to the output file, `vncserver.out` in the job submission directory, with the connect port for the vncviewer. Watch for the "To connect via VNC client" message at the end of the output file, or watch the output stream in a separate window with the commands:

   ```
   login1$ touch vncserver.out ; tail -f vncserver.out
   ```

   The lightweight window manager, `xfce`, is the default VNC desktop and is recommended for remote performance. Gnome is available; to use gnome, open the `~/.vnc/xstartup` file (created after your first VNC session) and replace `startxfce4` with `gnome-session`. Note that gnome may lag over slow internet connections.

2. Create an SSH Tunnel to Maverick2

   TACC requires users to create an SSH tunnel from the local system to the Maverick2 login node to assure that the connection is secure. The tunnels created for the VNC job operate only on the `localhost` interface, so you must use `localhost` in the port forward argument, not the Maverick2 hostname. On a Unix or Linux system, execute the following command once the port has been opened on the Maverick2 login node:

   ```
   localhost$ ssh -f -N -L xxxx:localhost:yyyy username@maverick2.tacc.utexas.edu
   ```

   where:

   - *yyyy* is the port number given by the vncserver batch job
   - *xxxx* is a port on the remote system. Generally, the port number specified on the Maverick2 login node, *yyyy*, is a good choice to use on your local system as well
   - `-f` instructs SSH to only forward ports, not to execute a remote command
   - `-N` puts the `ssh` command into the background after connecting
   - `-L` forwards the port

   On Windows systems find the menu in the Windows SSH client where tunnels can be specified, and enter the local and remote ports as required, then `ssh` to Maverick2.

3. Connecting vncviewer

   Once the SSH tunnel has been established, use a VNC client to connect to the local port you created, which will then be tunneled to your VNC server on Maverick2. Connect to `localhost:xxxx`, where *xxxx* is the local port you used for your tunnel. In the examples above, we would connect the VNC client to `localhost::xxxx`. (Some VNC clients accept `localhost:xxxx`).

   We recommend the TigerVNC VNC Client, a platform independent client/server application.

   Once the desktop has been established, two initial xterm windows are presented (which may be overlapping). One, which is white-on-black, manages the lifetime of the VNC server process. Killing this window (typically by typing `exit` or `ctrl-D` at the prompt) will cause the vncserver to terminate and the original batch job to end. Because of this, we recommend that this window not be used for other purposes; it is just too easy to accidentally kill it and terminate the session.

   The other xterm window is black-on-white, and can be used to start both serial programs running on the node hosting the vncserver process, or parallel jobs running across the set of cores associated with the original batch job. Additional xterm windows can be created using the window-manager left-button menu.

Software

As of January 17, 2023, the following software modules are currently installed on Maverick2. You can discover already installed software using TACC's Software Search tool or via `module` commands e.g., `module spider`, `module avail` to retrieve the most up-to-date listing.

```
login1$ module avail

-------------------- /opt/apps/intel18/impi18_0/modulefiles --------------------
   boost/1.66                    phdf5/1.10.4   (D)
   fftw3/3.3.6                   pnetcdf/1.8.1
   parallel-netcdf/4.3.3.1       python2/2.7.16 (L,D)
   parallel-netcdf/4.6.2   (D)   python3/3.7.0  (D)
   phdf5/1.8.16

------------------------ /opt/apps/intel18/modulefiles ------------------------
```

```
   hdf5/1.8.16        mkl-dnn/0.18.1    netcdf/4.3.3.1        python3/3.7.0
   hdf5/1.10.4 (D)    nco/4.6.9         netcdf/4.6.2   (D)    udunits/2.2.25
   impi/18.0.2 (L)    ncview/2.1.7      python2/2.7.16

---------------------------- /opt/apps/modulefiles -----------------------------
   TACC          (L)      git/2.24.1       (L)      mcr/9.6
   autotools/1.2 (L)      hwloc/1.11.2              mcr/9.9               (D)
   cmake/3.8.2            idev/1.5.5                ncl_ncarg/6.3.0
   cmake/3.10.2           intel/16.0.3              nvhpc/21.3.0
   cmake/3.16.1  (L,D)    intel/17.0.4              ooops/1.3
   cuda/10.0     (g)      intel/18.0.2     (L,D)    sanitytool/2.0
   cuda/10.1     (g)      launcher_gpu/1.0          settarg
   cuda/11.0     (g)      lmod                      swr/18.3.3
   cuda/11.3     (g,D)    mathematica/12.0          tacc-singularity/3.7.2
   gcc/5.4.0              matlab/2018b              tacc_tips/0.5
   gcc/6.3.0              matlab/2019a              xalt/2.9.6            (L)
   gcc/7.1.0              matlab/2020b     (D)
   gcc/7.3.0     (D)      mcr/9.5

  Where:
   D:  Default Module
   L:  Module is loaded
   g:  built for GPU
```

At this time, with the limited size of the local disks on Maverick2, we are keeping the number of packages supported to a reduced size to accommodate the work done on this system that is not possible or practical on other TACC systems.

Users must provide their own license for commercial packages. TACC will work on a best effort level with any commercial vendors to support that software on the system, but make no guarantee that licences can migrate to our systems or can be supported within the support framework at TACC.

You are welcome to install packages in your own `$HOME` or `$WORK` directories. No super-user privileges are needed, simply use the `--prefix` option when configuring then making the package.

Deep Learning Packages

See: Tensorflow at TACC

See the Remote Desktop Access at TACC tutorial to set up a VNC or DCV connection.

Building Software

Like Stampede2, Maverick2's default programming environment is based on the Intel compiler and Intel MPI library. For compiling MPI codes, the familiar commands `mpicc`, `mpicxx`, `mpif90` and `mpif77` are available. Also, the compilers `icc`, `icpc`, and `ifort` are directly accessible. To access the most recent versions of GCC, load the `gcc` module.

You're welcome to download third-party research software and install it in your own account. Consult the Stampede2 User Guide for detailed information on building software.

Help Desk

TACC Consulting operates from 8am to 5pm CST, Monday through Friday, except for holidays. You can submit a help desk ticket at any time via the TACC User Portal with "Maverick2" in the Resource field. Help the consulting staff help you by following these best practices when submitting tickets.

- **Do your homework** before submitting a help desk ticket. What does the user guide and other documentation say? Search the internet for key phrases in your error logs; that's probably what the consultants answering your ticket are going to do. What have you changed since the last time your job succeeded?

- **Describe your issue as precisely and completely as you can:** what you did, what happened, verbatim error messages, other meaningful output. When appropriate, include the information a consultant would need to find your artifacts and understand your workflow: e.g. the directory containing your build and/or job script; the modules you were using; relevant job numbers; and recent changes in your workflow that could affect or explain the behavior you're observing.

- **Subscribe to Maverick2 User News.** This is the best way to keep abreast of maintenance schedules, system outages, and other general interest items.

- **Have realistic expectations.** Consultants can address system issues and answer questions about Maverick2. But they can't teach parallel programming in a ticket, and may know nothing about the package you downloaded. They may offer general advice that will help you build, debug, optimize, or modify your code, but you shouldn't expect them to do these things for you.

- **Be patient.** It may take a business day for a consultant to get back to you, especially if your issue is complex. It might take an exchange or two before you and the consultant are on the same page. If the admins disable your account, it's not punitive. When the file system is in danger of crashing, or a login node hangs, they don't have time to notify you before taking action.

References

- `idev` documentation
- Tensorflow at TACC
- TACC Analysis Portal
- Multi-Factor Authentication at TACC