MIT LINCOLN LABORATORY
SUPERCOMPUTING CENTER

TX-E1 REFERENCE GUIDE

Getting Help

Website: https://supercloud.mit.edu/

Email: supercloud@mit.edu

To help us resolve your issue more quickly, when emailing us, include the following information, where applicable:

- Whether you are submitting your jobs from a Jupyter Notebook running on your desktop system, or if you are using an ssh session to the login node
- The command that you used to launch your job
- The job id of your job
- The full error message you received
- Any supporting files (code, submission scripts, screenshots, etc.)

Accessing TX-E1

- Via SSH (for access to the command line):
  ssh <username>@txe1-login.mit.edu
- Via Web Portal: https://txe1-portal.mit.edu/
  - Log in to the TX-E1 Web Portal with:
    MIT Touchstone/InCommon Federation; or
    PKI Certificate/SmartCard
  - Access the ssh key portal
  - Browse the file system
  - Access the Databases system
  - Access the Jupyter Notebook system
  - Access the Online Courses
  - End your portal session

Policies

- Standard MIT Acceptable Use Policies apply to all accounts on the SuperCloud systems
- We do not back up files on the SuperCloud and strongly recommend that you regularly move files back to your local workstation
- **Regular System Maintenance**

  Routine maintenance on TX-E1 is performed on the third Thursday of the month. If the scheduled maintenance day falls on a campus holiday, the downtime will occur on the following Thursday.

  During the maintenance period, the Jupyter Portal, the Database Portal, and all of the compute nodes are unavailable.

  Email will be sent when the system has been returned to service.

Online Courses

- Online courses and tutorials (https://learn.llx.edly.io):
  - Practical HPC online course
  - Click "Explore Courses" to see other availalbe courses
- Teaching Examples (read only, copy dirs to your home dir):
  - /home/gridsan/groups/bwedx

For more information, see https://supercloud.mit.edu/online-courses

Email bwx-help@bwsix.mit.edu with issues with the course platform.

Group Shared Directories

Requesting a Group Shared Directory

Email your request to supercloud@mit.edu and provide the following:

- The name you'd like for the directory (shorter is better, and it should not contain spaces)
- The list of users who should be allowed to access the directory
- Whether this is a public dataset
- Who should give approval for new group members

Using Group Shared Directories Safely

- To upload files to the group shared directory, use "scp" or "rsync"
- In scripts and code, when referencing the shared directory, always use this path as the leading prefix to point to files when sharing with other members: /home/gridsan/groups/<shared_directory_name>
  Note that a username does not appear in this pathame

General Job Management Commands

Check System Status

$ LLfree

Monitor Your Job

$ LLstat

Stop Your Job

$ LLkill [options] <job_id_list>

job_id_list	Comma separated list of job number(s)
Options:	
-h, --help	Show this help message and exit
-t <task_list>	Specify selected job array elements to delete task_list syntax: start[-end[:step]]
-u <username>	Delete all jobs owned by username No effect if job_id_list is specified

Slurm Resources

- Native Slurm commands and options can be used for job submission and monitoring
- A two-page Command Summary can be found here:
  https://slurm.schedmd.com/pdfs/summary.pdf

Module Commands

```
module avail
module show <module>
module load | unload <module>
module list
```

Loading Modules in a Script

```
#!/bin/bash
# Load the desired module
module load <module>
…
```

File System Best Practices

File size and organization

- Store data in large files to take advantage of the file system's ability to provide fast access to files. Since the block size of files on the system is 1MB, every file will use at least 1MB of space.
- Read/write fewer large files (~100 MB) rather than many small files.
- AI data should be stored in large (>100s of MB) files. Use HDF5/ TFRecord/NetCDF/multi-page TIFF for images, etc.
- Organize data so there are less than 1,000 files/folders per directory.
- Aim for <100 files/folders for folders that you will list interactively.
- Store data and run your programs from a group shared directory.

During run-time

- Open files in read only mode if the app does not write to the file.
- Avoid wildcards when running operations on large numbers of files.
- Avoid using dir, ls, and other commands that scan the file system.
- When analyzing a data set of many files, create another file that lists the full file paths of each file in the corpus so each processor can read this file and avoids doing an ls to get the filenames.
- When reading files from many processors, use LLcopy2tmp to copy the files to the node's local disk at the beginning of your job. Use the environment variable $TMPDIR to access the files.
- Avoid external internet access (checking out code from an external repo, downloading data). Do these external accesses prior to running your job, from one of the login nodes.

Copying Files to TX-E1

There are several ways to transfer files from your computer to TX-E1

scp

scp [options] <src> <user>@txe1-login.mit.edu:<dest>

Most commonly used options:

-r	recursively copy files in all subdirectories
-v	verbose

rsync

rsync [options] <src> <user>@txe1-login.mit.edu:<dest>

Most commonly used options:

-r	Recursively copy files in all subdirectories
-l	Copy and retain symbolic links
-u	Skip files that are newer on the LLSC system
-g	Preserve group attributes (use with shared groups)
-v	Verbose
-P	Show copy progress

LLcopy2tmp

LLcopy2tmp [-gCopyGroup] <src-1> [src-2 ...]

Copy <src-#> and its contents to the local directory pointed to by the TMPDIR environment variable

- Copies once per node for the same CopyGroup, which defaults to the environment variable SLURM_JOB_ID
- Dereferences symlinks

January 2023

Using Triples Mode to Launch Your Job

The triples mode job launch is a job launch mechanism which provides more flexibility to manage memory and threads. It enables fast resource allocation and job execution by aggregating compute tasks to be executed on the same node as a single scheduling task in an array job. It can be used with LLsub, LLMapReduce and pMatlab job launches.

See https://supercloud.mit.edu/submitting-jobs#triples for more details.

To submit your job to launch with triples mode you specify the number of processes by providing three numbers:

- Nnode: number of compute nodes
- Nppn: number of processes per node
- Nttp: number of threads per process (default is 1)

LLsub job submission

[Nnode,Nppn,Nttp]

LLMapReduce

--np=[Nnode,Nppn,Nttp]

pMatlab

For pMatlab the third value in the triple is the number of OpenMP threads (default is 1)

pRUN('mfile', [Nnode Nppn OMP_NUM_THREADS], <cpuType>)

Installing Python Packages on TX-E1

See our webpage for details, best practices and performance tips:

https://supercloud.mit.edu/software-and-package-management#home-install

Installing Your package in User Space

This is the preferred method for installing the Python packages you need:

1) $ export TMPDIR=/state/partition1/user/$USER
2) $ mkdir $TMPDIR
3) $ module load anaconda/desiredAnacondaVersion
4) $ unset PYTHONPATH
5) $ pip install --user --no-cache-dir packageName

After the install is complete:

6) $ rm -rf $TMPDIR

Installing Your Package in Your Own Conda Environment

If you absolutely need to use conda to install a package, create a conda environment using our anaconda modules. If you have previously installed anaconda or miniconda in your home directory, delete it now.

1) $ export TMPDIR=/state/partition1/user/$USER
2) $ mkdir $TMPDIR
3) $ module load anaconda/desiredAnacondaVersion
4) $ unset PYTHONPATH
5) $ conda create --prefix /home/gridsan/<UserName>/.conda/envs/myenv [python=desiredPythonVersion] packageName(s) [jupyter]

Activating Your Environment

6) $ eval "$(conda shell.bash hook)"
7) $ source activate /home/gridsan/<UserName>/.conda/envs/myenv

Adding Packages to an Existing (and Activated) Conda Environment

$ conda install --prefix /home/gridsan/<UserName>/.conda/envs/myenv packageName

LLsub

Submit a job

$ LLsub <command> [N | [Nnode,Nppn,Nttp]] [options] [-- <command_args>]

Options for job submission only:

command	Specify the program file to execute
N	Specify the number of tasks/processes
[Nnode,Nppn,Nttp]	Triple mode launch: numNodes, numProcPerNode, numThreadsPerProc

Request an interactive session

$ LLsub -i [full | -s <slots_per_task> | -N <num_nodes>] [n <ntasks>] [--x11] [--resv-ports <count>] [options]

Options for interactive session only:

full	Request exclusive node access
-N <num_nodes>	Number of nodes to allocate
-n <ntasks>	Number of tasks to execute for interactive MPI jobs
--x11	Enable X11 forwarding for interactive session

Most commonly used options for job submission or interactive session:

-h	Show this help message and exit
-c xeon-p8 | xeon-g6	Request compute nodes with specified CPU type
-g volta:<count>	Number of GPU units to be used (1 or 2)
-s <slots_per_task>	Number of slots (cores) per task (default: 1)
-a mpi	Indicates the command is an MPI application mpirun command will be selected from your path

Jupyter Notebooks

URL: https://txe1-portal.mit.edu/jupyter/

On this page you can launch a session with default settings, or use the Advanced Launch Options to change your Jupyter Job allocation and settings. In most cases, the default settings are sufficient for a Jupyter instance.

Advanced Options

- Partition : xeon-p8 | xeon-g6-volta
- CPU Type : Intel Xeon-P8 | Intel Xeon-G6
- CPU Count : Number of cpus/cores
- GPU Resource flag : Volta V100
- GPU Resource Count : 1–2
- Exclusive: allocates a full node
- Time Limit: Length of notebook session
- Anaconda/Python version
- Application : Jupyter Notebook | Jupyter Lab

Your Jupyter instance will open in your home directory. By clicking on the "New" button in the top right, you can:

- Start a new notebook (Julia, Python, Matlab, Octave, R)
- Open a terminal window
- Create a new text file with a simple text editor

LLMapReduce

LLMapReduce --mapper <script> --input <input_dir> --output <output_dir> [options]

Most commonly used options (see webiste for full list of options); default values are shown in **bold**:

-h, --help
  Show this help message and exit
--apptype=siso | mimo
  Specify the application type
--cpuType= xeon-p8 | xeon-g6
  Request compute nodes with a specific CPU type
--exclusive=true | **false**
  Enable exclusive mode
--gpuNameCount=volta:<count>
  Number of GPU units to be used for each task (1 or 2)
--input=<input_dir>
  Path to input files or file containing list of input files
--keep=true | **false**
  Keep temporary MAPRED.PID directory
--mapper=<mapper>
  Specify the mapper program to execute
--ndata=<count>
  # input data files to be processed per task (default: 1)
--np=<nprocs>
  # processes to run concurrently (default: # input files)
  Specified as N (total num proc) or [Nnode,Nppn,Nttp] (numNodes, numProcPerNode, numThreadsPerProc)
--output=<output_dir>
  Path for the output files
--reducer=<reducer>
  Specify the reducer program to execute
--slotsPerTask=<num_slots>
  Number of slots (cores) per task (default: 1)

Launching MATLAB®/Octave Jobs

Use these commands at the MATLAB®/Octave prompt or in a MATLAB®/Octave function or script.

pMatlab Job

- Run the leader process (process 0) on your desktop and remaining processes on LLSC systems:

  eval(pRUN('myScriptName', <nproc | [Nnode,Nppn,Nttp]>, 'grid'))

- Launch your job to run entirely on LLSC systems:

  eval(pRUN('myScriptName', <nproc | [Nnode,Nppn,Nttp]>, 'grid&'))

Single MATLAB®/Octave Job

- Launch your serial functions on LLSC systems:

  launch_status = LaunchFunctionOnGrid(m_file)
  launch_status = LaunchFunctionOnGrid(m_file, <variables>)

- Launch your MATLAB® functions that call the parfor() function on LLSC systems:

  launch_status = LaunchParforOnGrid(m_file)
  launch_status = LaunchParforOnGrid(m_file, <variables>)