SUBMITTING JOBS AND RUNNING PROGRAMS

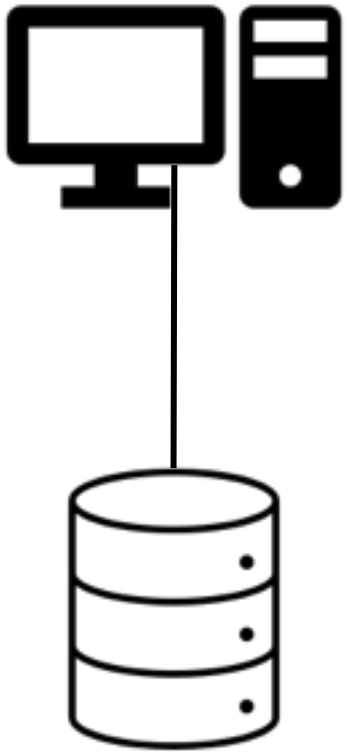
Module 1.3

LEARNING OBJECTIVES

In the present module we will cover:

- Writing, modifying, and transferring programs from a local computer to a remote computer and vice versa.
- Connecting a local computer to the super computers, submitting, and running programs.

WRITING CODE IN YOUR LOCAL COMPUTER



Local computer

Local storage (hard drive)

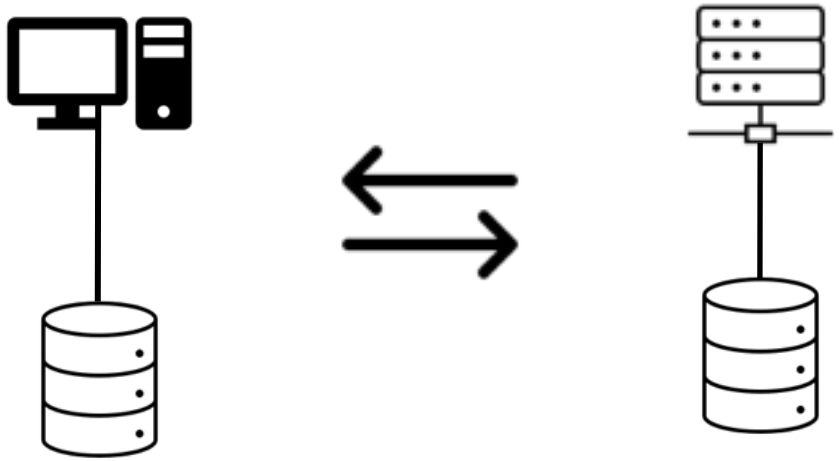
Using a development environment you should be able to write your source code in your computer. The source code can be initially stored in the local hard drive.

You may want to also store your source code using a revision control system (e.g., SVN, S, Git).

TRANSFERING CODE FROM YOUR LOCAL COMPUTER TO THE SUPERCOMPUTER

Local computer

Supercomputer



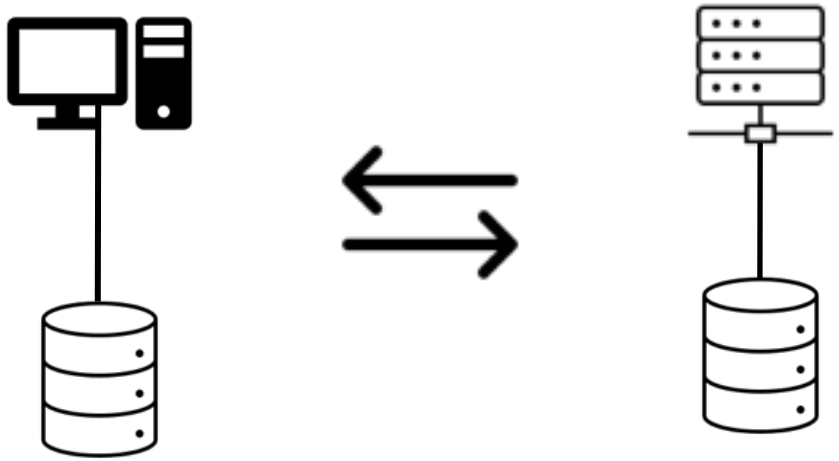
You will need to transfer the source code from your local computer to the login-node of the supercomputer. There are different methods to do so:

1. Cloning a revision control repository.
2. Copying the files using ‘scp’.
3. Using rsync to transfer the files.
4. Using globus online to transfer the source code.

TRANSFERING CODE TO YOUR LOCAL COMPUTER FROM THE SUPERCOMPUTER

Local computer

Supercomputer



You may do changes to your code, so it performs well on the supercomputer. Therefore, you may want to transfer your new source code from the supercomputer to your local machine. The methods are:

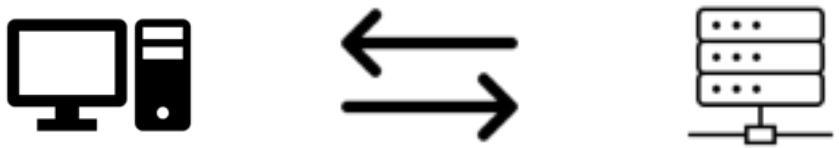
1. **Committing to a subversion control repository.**
2. Copying the files using ‘scp’.
3. Using ‘rsync’ to transfer the files.
4. Using ‘globusonline’ to transfer the source code.

In general it is a good practice to use a revision control system as you can create a branch for each new supercomputer that you develop for and then merge the changes to the main code repository.

CONNECTING A LOCAL COMPUTER TO A SUPERCOMPUTER

Local computer

Supercomputer



In order to compile and run your source code you will need to gain access to the supercomputer. Typically a set of nodes called *login-nodes* serve the users interactively. The typical connection method is using secure shell (ssh). Once in the login node you will be presented with a terminal (e.g., bash, tcsh, etc)

To connect to the supercomputer you will need a username and password. You will also be assigned a one-time password (OTP). The OTP is a sequence of numbers and characters that change in time which prevents access to your account if someone steals your password. The OTP can be delivered either using a physical key fob, or a virtual app in your smart phone.

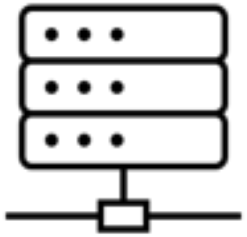
RSA OTP key fob

It is advisable to multiplex your connection so you can edit, compile and test without having to create multiple ssh instances. Terminal multiplexing can be accomplished using GNU Screen or Tmux; the latter two must be running on the supercomputer.

SUBMITTING PROGRAMS

A supercomputer is a shared resource that is used by several users concurrently. To optimize access to the resource the minimal unit of execution of a program is defined as a job. Jobs are scheduled to run in a group of nodes for a given amount of time as requested by the user. The queue manager, so-called scheduler, ensures that jobs are queued using some fair use policy (e.g., TORQUE).

Supercomputer



The action of sending a job to the queueing systems is called job submission. Therefore, in contrast to running programs on a local computer, programs in a supercomputer are first queued and subsequentely managed by a workload manager (e.g., Moab, Slurm).

my-job.pbs:

```
#!/bin/bash
#PBS -N my-job
#PBS -l mppwidth=64
#PBS -l mppnppn=32
#PBS -l walltime=00:10:00

# set PATH to include the ThreadSpotter™ bin directory
PATH=$PATH:installation_directory/bin

# change directory where the job was submitted from
cd $PBS_O_WORKDIR

aprun -b -n 64 -N 32 sample -g 1 -o my-job-samplefiles/process-%r.smp \
      -r ./my-job arg1 arg2
```

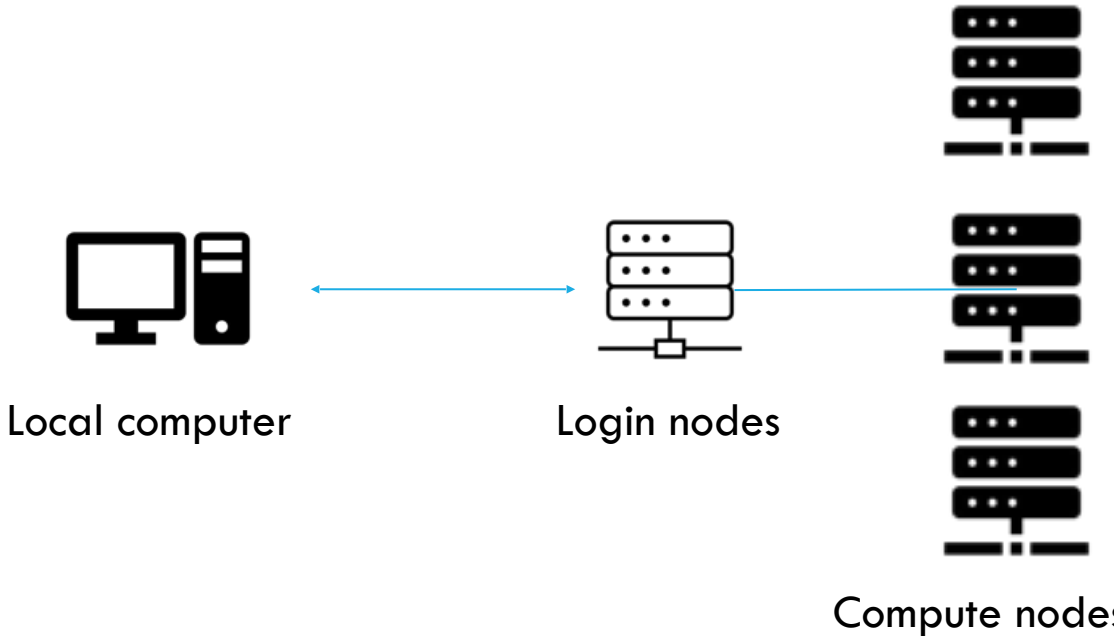
and invoke this script using:

```
$ qsub my-job.pbs
```

To submit the job you must write a submission script.

RUNNING PROGRAMS

After submission, programs are executed in so called-compute nodes. Program execution is controlled by the launcher (e.g., ALPS) and the details of the program execution are determined by the user before submission (e.g., aprun parameters).



my-job.pbs:

```
#!/bin/bash
#PBS -N my-job
#PBS -l mppwidth=64
#PBS -l mppnppn=32
#PBS -l walltime=00:10:00

# set PATH to include the ThreadSpotter™ bin directory
PATH=$PATH:installation_directory/bin

# change directory where the job was submitted from
cd $PBS_O_WORKDIR

aprun -b -n 64 -N 32 sample -g 1 -o my-job-samplefiles/process-%r.smp \
      -r ./my-job arg1 arg2
```

and invoke this script using:

```
$ qsub my-job.pbs
```

REFERENCES

- man scp: https://man7.org/linux/man-pages/man1/scp.1.html
- man rsync: https://man7.org/linux/man-pages/man1/rsync.1.html
- Globus online: https://www.globus.org/
- Revision control system: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control
- Man screen: https://www.gnu.org/software/screen/manual/screen.html
- Man tmux: https://man7.org/linux/man-pages/man1/tmux.1.html
- Using aprun: https://bluewaters.ncsa.illinois.edu/using-aprun
- Queueing policies: https://bluewaters.ncsa.illinois.edu/queues-and-scheduling-policies
- Accessing compute nodes: https://bluewaters.ncsa.illinois.edu/accessing