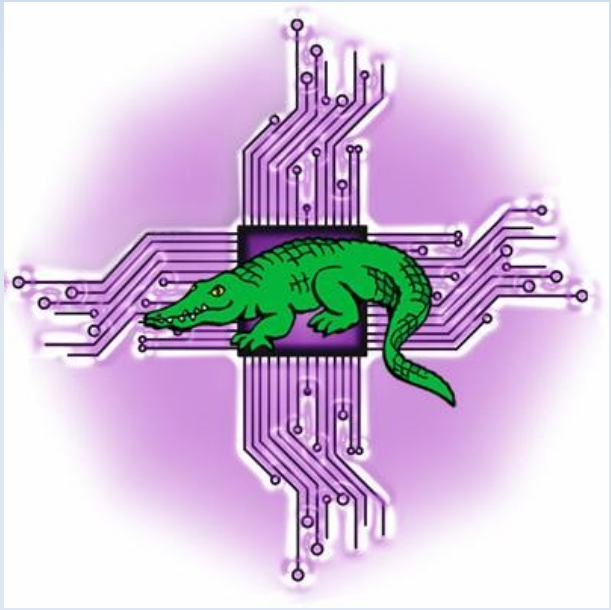
HPC Job Submission and Programming

COMP90024 Cluster and Cloud Computing



University of Melbourne, March 28, 2023

lev.lafayette@unimelb.edu.au

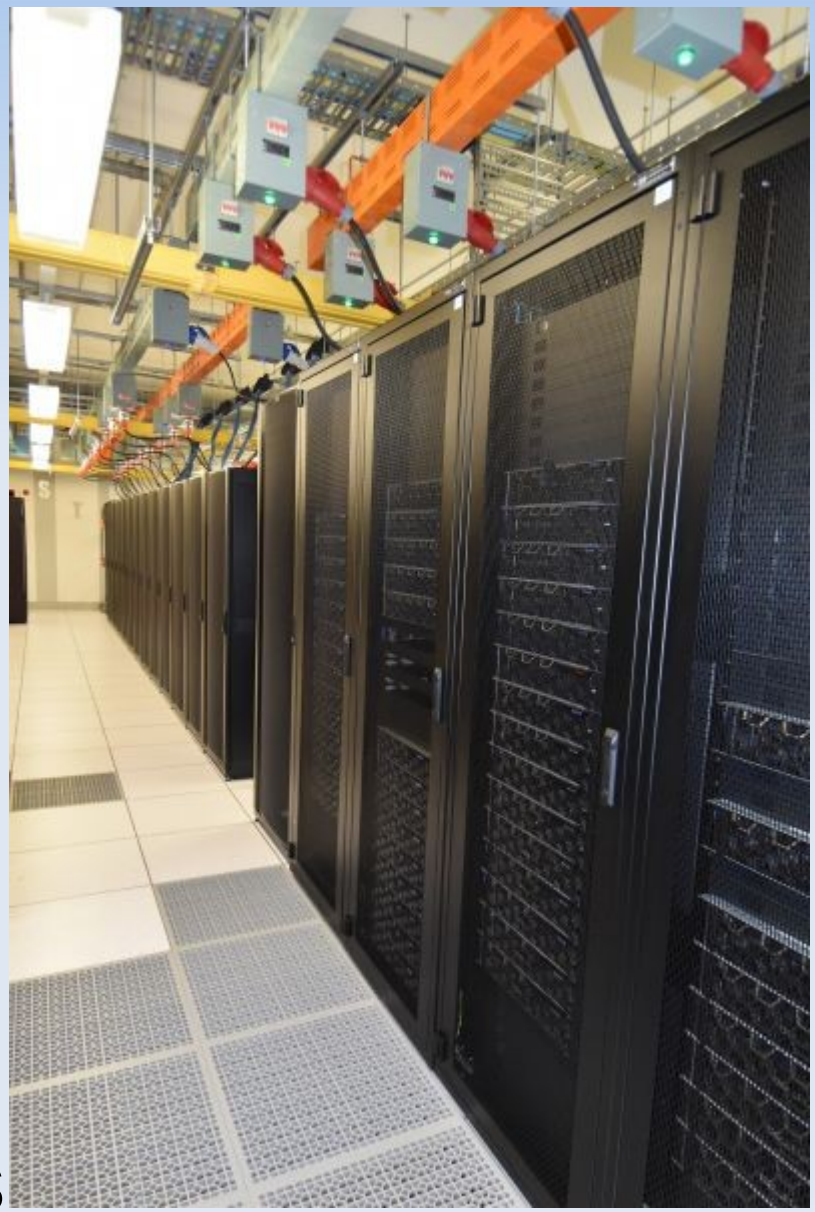
Outline of Workshop

Review of Architecture, SSH, and Core Slurm Directives

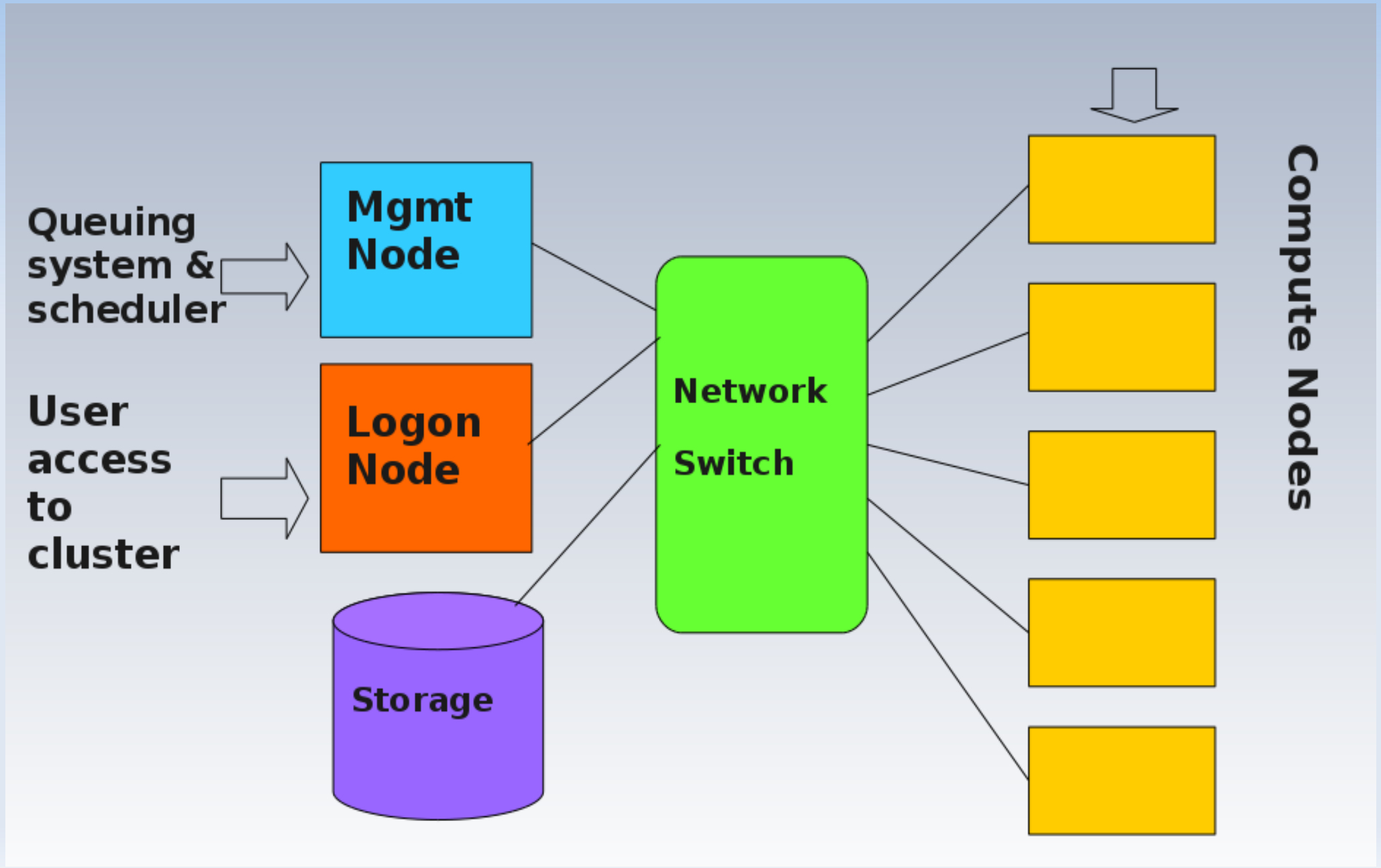
Two Models of Parallel Computation

Distributed Memory Programming with OpenMPI

Examples, Questions & Answers



Architecture Review: HPC Design



Accounts and Secure Shell (SSH)

To login, a user must (a) have an account (b) belong to a project and (c) have an SSH client (standard on MacOS and Linux, on MS-Windows use bash shell extensions, Putty, or MobaXterm).

Standard method

```
ssh your-username@spartan.hpc.unimelb.edu.au
```

Better Methods

1. Use public key access.

On a Mac or Linux local system create the key pair with:

```
ssh-keygen -t rsa
```

Put the public key in ~/.ssh/authorized_keys on Spartan. Keep your private key private!

2. Use an SSH config

3. Use SSH agent forwarding

See: https://dashboard.hpc.unimelb.edu.au/ssh/

Modules and Extensions

Do not use sudo, or apt. Applications have been compiled by the system engineers from sourcecode.

Use the environment modules system in your scripts. Recommended that toolchain is loaded first to avoid conflicts

module avail (e.g., module avail python/)
module load (e.g., module load python/3.7.4)
module unload
module list & etc.

If additional extensions are required (Python/Perl “modules”, R libraries) there are specific instructions in /usr/local/common. Some are available as environment modules. e.g., module av numpy, module av mpi4py)

For Python in particular make use of virtual environments if the extension is installed in the environment module.

See: /usr/local/common/Python/virtualenv.md

Slurm Parallel Directives

Slurm is the job submission system. It uses a short script that (1) invokes a shell (2) Includes directives to the scheduler and (3) shell script commands (e.g., loading modules, running commands, invoking other scripts).

1) Nodes are system units. Cores are CPUs within nodes.
2) Increasing the number of cores (or nodes!) requested does not make the application parallel.
3) The directive cpus-per-task is for shared memory, multithreaded applications (e.g., OpenMP, multiproc).
4) The directive ntasks (or ntasks-per-node) is for distributed memory, message passing applications (e.g., OpenMPI, MPI4Py).

Sample Slurm Jobs

Single core jobs available in /usr/local/common/. See examples in the IntroLinux, R, MATLAB, and Python subdirectories.

Multicore jobs can be either built on multithreading (1 task, cpus-per-task) or message passing (many tasks). Multithreading examples in Admixture, OpenMP, MATLAB, and Python subdirectories. Message Passing examples in IntroLinux, OpenMPI, and Python subdirectories. Note difference between nodes and tasks!

Multicore via capacity can also be achieved by job arrays. Job array examples in Array, Octave, R, Python subdirectories.

All jobs are submitted with `sbatch scriptname`

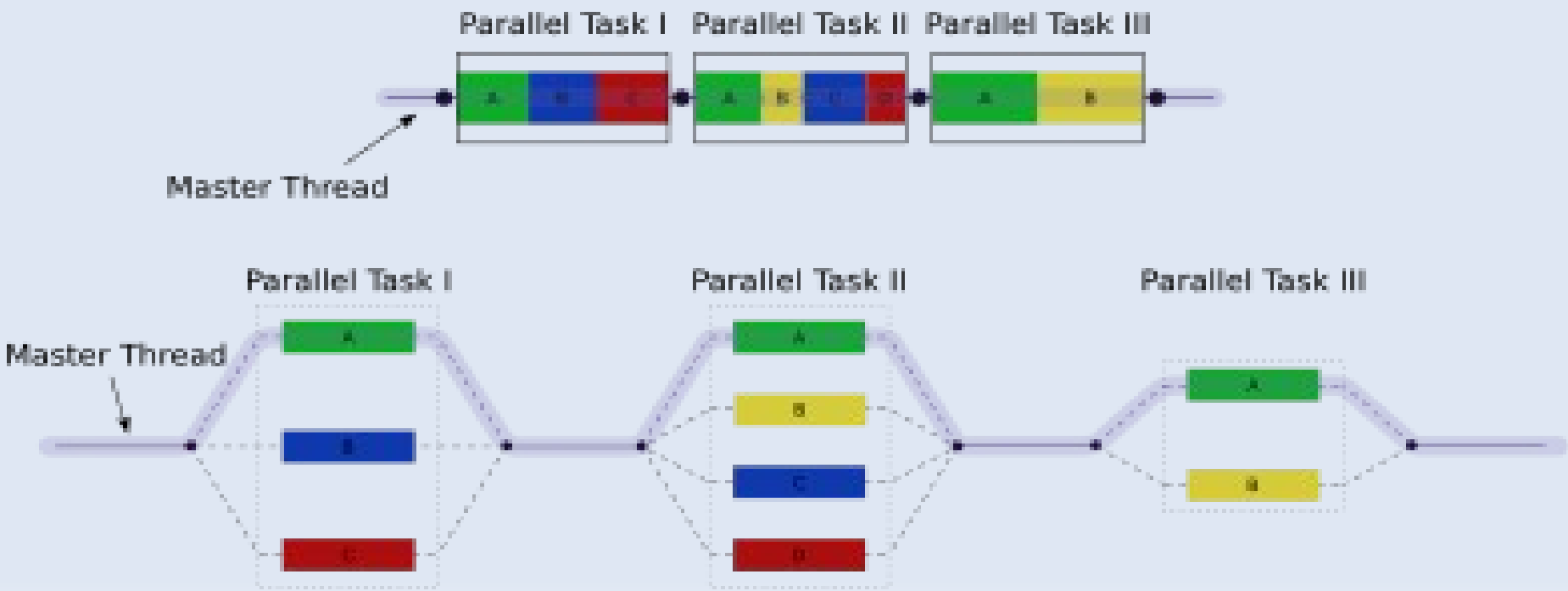
The Two Parallel Methods

One form of parallel programming is multithreading, where a master thread forks a number of sub-threads and divides tasks between them. The threads will then run concurrently and are then joined at a subsequent point to resume normal serial application. One implementation of multithreading is OpenMP (Open Multi-Processing). It *must* run on a single node.

Another form of parallel programming is message passing, where a root process coordinates message passing between processors in a communications world. One implementation of multithreading is OpenMPI (Open Multi-Passing Interface). It *may* run on multiple nodes.

Multithreading Structure

Multithreading requires a master thread that forks a number of sub-threads and divides tasks between them. In OpenMP this is determined by pragma (C) or sentinels (Fortran) statements. With multiproc (Python) it is determined by functions.

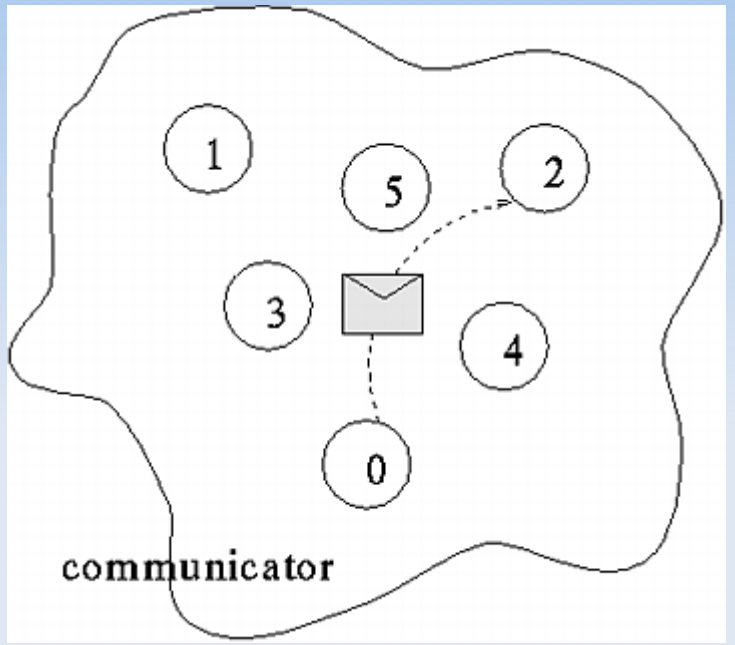


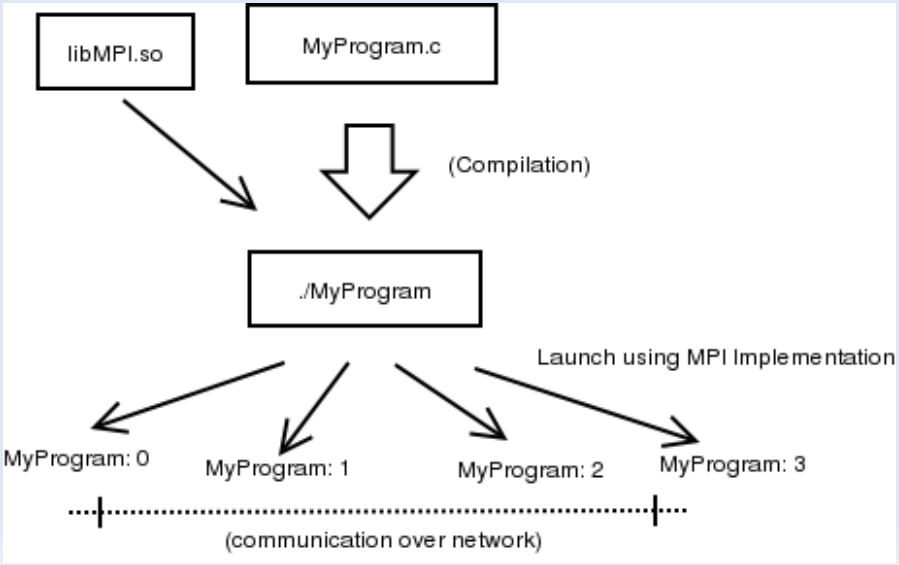
MPI Structure (OpenMPI, MPI4PY)

Message passing creates a communications world where processors can conduct their own activities, but pass relevant messages to one another as required. Similar to a postage system.

Message passing options include collective communications, reduction operations, even separate communication groups.

Additional programming effort is required for these routines.





OpenMP and MPI Examples

Example OpenMP and MPI code in /usr/local/common/COMP90024

Example multi and MPI4Py code

/usr/local/common/Python

Copy code to /home/$user or /data/projects/$projectID directories, review, compile, and submit.

Note the OpenMP code is easier, pragma-based directives. The OpenMPI code is based on MPI routines.

Use the mpi-pingpong example to test the performance between compute nodes and within a compute node.


THANKS FOR WATCHING
& LISTENING PATIENTLY
makeameme.org