ilifu Online Training – Advanced

Dr Jordan Collier

ilifu Support Astronomer, IDIA, Department of Astronomy, University of Cape Town
Adjunct Fellow, Western Sydney University

UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

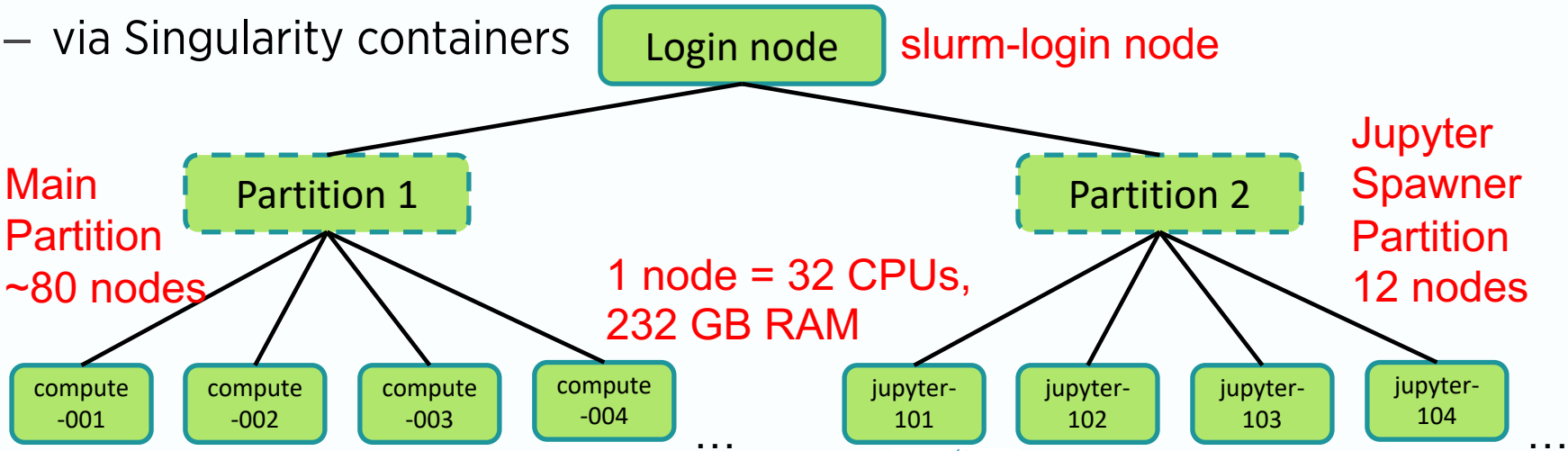
ilifu

IDIA Inter-University Institute for Data Intensive Astronomy

WESTERN SYDNEY UNIVERSITY

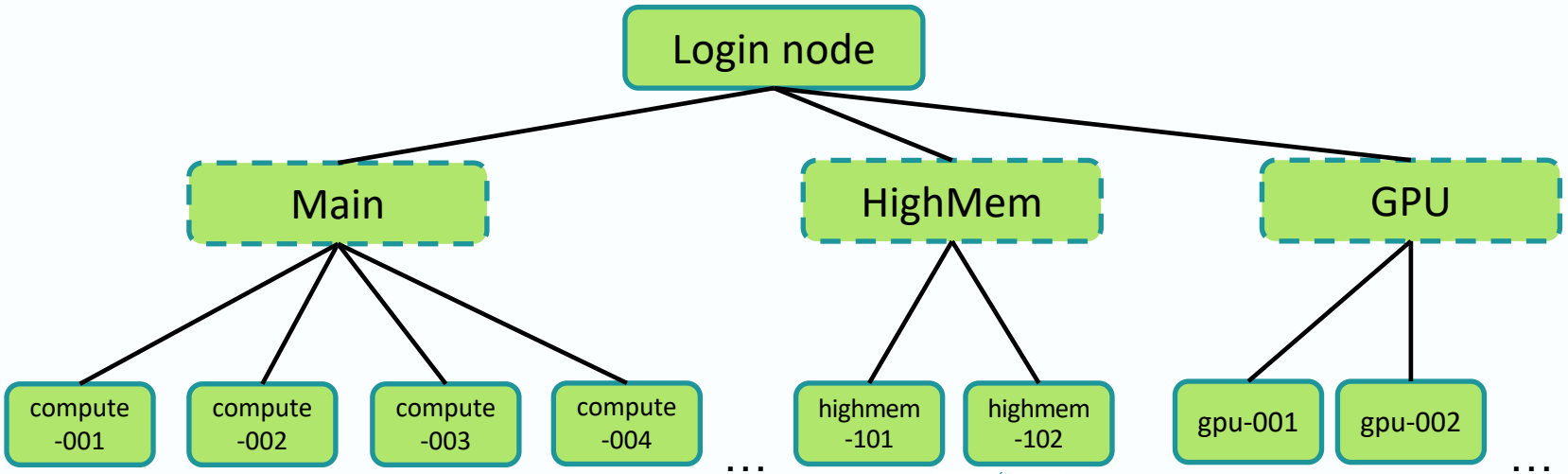
SLURM

- http://docs.ilifu.ac.za/#/getting_started/submit_job_slurm
- Login node (job submission & management)
  - where you land when you log in (also known as “head node”)
  - **run SLURM commands/submit jobs, but not software/heavy processes**
- Compute nodes
  - Where your processes run (also known as “worker nodes”)
  - via Singularity containers



SLURM

- http://docs.ilifu.ac.za/#/tech_docs/running_jobs?id=_4-specifying-resources-when-running-jobs-on-slurm
- Partitions (other than Jupyter) – see with ‘sinfo’:
  - Main: 70 nodes (currently), each w/ 32 CPUs, 232 GB (usable) RAM
  - HighMem: 2 nodes, each w/ 32 CPUs, 480 GB (usable) RAM
  - GPU: 4 nodes, each w/ 2 GPUs, 32 CPUs, 232 GB (usable) RAM



Compute Workflow

- Login node
  - Run SLURM commands and basic bash commands (cd, mkdir, ls, etc)
- Jupyter / development node
  - Development space – new code / workflows / routines
  - Debugging or testing software
- Main partition
  - Generally for stable, computationally-heavy processing
- HighMem partition
  - For single high-memory jobs that can't be split into multiple jobs by MPI
- Transfer node
  - Internal or external copying (see later slide)

SLURM – advanced user commands

- https://slurm.schedmd.com/ (shows long and short form)
- See basic training for basic SLURM user commands
- Before running jobs:

```
$ sinfo -O "partition,available,cpus,nodes,memory,
statecompact" #list partitions and their specs

$ sacctmgr show user $USER -s #list your SLURM
format=account%30,cluster%15 accounting groups
```

- Syntax for srun and sbatch (after #SBATCH) params:

```
$ srun --partition=Main --account=b05-pipelines-ag
script.sh #submit to Main under specific account

$ srun --partition=GPU script.sh
#submit job to GPU partition
```

SLURM – advanced user commands

- https://slurm.schedmd.com/
- After / during running jobs
  - jobID given from sbatch output (to screen) or from squeue

```
$ scontrol show jobID 129916 #shows extensive info about job running (including working directory)

$ sacct -o JobID%-15,JobName%-15,Partition,Account,Elapsed,NNodes%6,NTasks%6,NCPUS%5,MaxDiskRead,MaxDiskWrite,NodeList%20,MaxRSS,CPUTime,State,ExitCode #shows useful information for multi-CPU jobs

$ sacct -S 2021-04-10-09:00 -E 2021-04-12-12:00 -X -o JobID,JobName,Start,End,State #shows jobs started and completed between these dates

$ sacct -j 882242 --cluster=ilifu-slurm #shows job on old cluster
```

SLURM – advanced user commands

- https://slurm.schedmd.com/
- http://docs.ilifu.ac.za/#/tech_docs/running_jobs?id=_4-specifying-resources-when-running-jobs-on-slurm
- Email parameters:

```
$ srun --mail-user=<address> --mail-type=<event_types>
```

  - May want (comma-separated) events: BEGIN,END,FAIL,TIME_LIMIT

- Exclude nodes
  - e.g. problematic nodes (please report to ilifu support):

```
$ srun --exclude=compute-[101,102-105]
```

SLURM – array jobs

- Array jobs allow quick submission many similar jobs, each with same resources, without need to manually launch each
  - Ideally passes array task ID into script, which changes the behaviour of the job each time – i.e. different input selection
  - Can also be used to run several related steps in a serial process
- Example job array, running 20 jobs, with 5 run concurrently

```
#!/bin/bash

#SBATCH --array=1-20%5

#SBATCH --output=logs/%x-%A_%a.out

#SBATCH --error=logs/%x-%A_%a.err

#SBATCH --mail-user=jordan@idia.ac.za

#SBATCH --mail-type=FAIL,ARRAY_TASKS

singularity exec python myscript.py --input $SLURM_ARRAY_TASK_ID
```

SLURM substitutions & environment variables

- https://slurm.schedmd.com/sbatch.html

Parameter	Substitution / filename pattern	Environment Variable
jobID of the running job	%j	SLURM_JOB_ID
Job name	%x	SLURM_JOB_NAME
Job array's master job allocation number	%A	SLURM_ARRAY_JOB_ID
Job array task ID (index) number	%a	SLURM_ARRAY_TASK_ID
CPUs per task		SLURM_CPUS_PER_TASK

- Useful for multi-CPU jobs:
  `export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK`

SLURM – running an interactive job

- http://docs.ilifu.ac.za/#/tech_docs/running_jobs?id=_3-interactive-sessions
- Specifying lower wall-time (default 3 hours) and less memory (default ~7 GB) increases chance of job launching immediately
- In steps:

```
$ srun --pty --time=10 --mem=1GB bash
$ singularity shell /idia/software/containers/casa-stable.img
$ casa --nologger --log2term
```

- In single call:

```
$ srun --pty --time=00:10:00 --mem=1GB singularity exec
/idia/software/containers/casa-stable.img casa --nologger --log2term
```

- Must manually process after this

SLURM – running a job

- In previous srun example, job run in your bash session, output printed on screen, and process is volatile (srun / screen / tmux / mosh), subject to restarting SLURM login node
- Best to write sbatch job, which runs in the background and writes to log files. An sbatch file of the previous example:

```
#!/bin/bash

#SBATCH --job-name=casa_job

#SBATCH --time=01:00:00

#SBATCH --mem=1GB

#SBATCH --output=logs/%x-%j.out

#SBATCH --error=logs/%x-%j.err

singularity exec /idia/software/containers/casa-stable.img casa --nologger --nogui
--logfile logs/${SLURM_JOB_NAME}-${SLURM_JOB_ID}.casa -c myscript.py

$ sbatch casa_job.sh
```

SLURM – dependencies

- Allows jobs to be scheduled for running, based on the status of a previous job
  - e.g. only begin a particular job once previous one successfully completes

```
$ sbatch -d afterok:882242 --kill-on-invalid-dep=yes another_job.sh #submit another_job.sh to SLURM queue, to begin after jobID 882242 successfully completes (exit code 0), or cancel the job if jobID 882242 fails

$ sbatch -d afterany:882242:882243 another_job.sh #submit another_job.sh to SLURM queue, to begin after jobIDs 882242 & 882243 complete (any exit code)
```

Parallelism

- Oxford definition for parallel processing
  - *a mode of operation in which a process is split into parts, which are executed simultaneously on different processors attached to the same computer [or different computers attached to the same cluster].*
  - A cluster includes many connected nodes, each with its own RAM & CPUs
  - A node = single computer / server / VM / machine / box
- The work is partitioned into smaller jobs, sometimes with a partition of the dataset

Parallelism

- Can be achieved on a single machine / node
  - Distributes work over many CPUs
  - Typically implemented using OpenMP

- Or over multiple machines / nodes
  - Distributes work over many tasks, over 1+ nodes
  - Each given amount of memory to use
  - Generally requires a cluster
  - Typically implemented using OpenMPI
  - Requires a message passing interface (MPI) wrapper
    - mpirun, aprun, srun (SLURM), mpicasa (CASA)

- Managed on ilifu by SLURM

Parallelism

- Implementing a normal job in SLURM
  - Will only use 1 CPU, 1 task, and 1 node
  - Default for many processes
- Implementing an OpenMP job in SLURM
  - Need to use >1 CPU, while nodes & tasks must be 1 (unless also using MPI)
    - `cpus-per-task`
    - May need to export `OMP_NUM_THREADS`
- Implementing an MPI job in SLURM
  - Need to use >1 task, while nodes and CPUs can be 1
    - `nodes, ntasks-per-node, cpus-per-task`
    - Need to wrap singularity in MPI call
- Cannot exceed 32 CPUs (or tasks) per node

SLURM – multi-CPU and multi-task jobs

- Example sbatch job using many CPUs within 1 node

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=8
#SBATCH --mem-per-cpu=1GB
singularity exec /path/to/container.simg python myscript.py
```

- Example sbatch job using many tasks within 1 node

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=16
#SBATCH --cpus-per-task=1
/path/to/mpirun singularity exec /path/to/container.simg python myscript.py
```

SLURM – multi-task jobs

- Example sbatch job using many tasks over multiple nodes

```
#!/bin/bash

#SBATCH --nodes=2

#SBATCH --ntasks-per-node=8

#SBATCH --cpus-per-task=1

#SBATCH --mem=10GB #memory per node, so 20GB allocated overall (usable by some software)

/path/to/mpirun singularity exec /path/to/container.simg python myscript.py
```

- Example sbatch job using many tasks and CPUs within 1 node

```
#!/bin/bash

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=4

#SBATCH --cpus-per-task=8

/idia/software/pipelines/casa-prerelease-5.3.0-115.el7/bin/mpicasa singularity exec
/idia/software/containers/casa-stable-5.6.2-2.simg casa –c tclean_script.py
```

IDIA MeerKAT Pipeline – A Good Framework

- Parallelised package for HPC processing (SLURM + cluster)
  - Uses multi-measurement sets (MMS) to parallelise across a cluster
- Outputs calibrated data (push of button!)
- HPC-friendly – dynamically uses resources & submits to queue
- Each job/script is a logical step that does / doesn't use MPI, and optionally uses a different container
- User can insert your scripts at start, middle or end
- https://idia-pipelines.github.io/docs/processMeerKAT
- Demo time!

Data Transfers

- http://docs.ilifu.ac.za/#/data/data_transfer
- Do not use login node!
- Don’t write large files to /users, only scripts/config files
- transfer.ilifu.ac.za
  - For cp, scp and rsync (internal ilifu transfers, or external transfers)
    - e.g. `$ scp /path/to/file/<filename> <username>@transfer.ilifu.ac.za:/idia/users/<username>/scripts/`
- Globus
  - Faster than scp and rsync
  - Uses dedicated data transfer node (DTN)

Data Transfers: Globus

- GridFTP transfer service setup between SARAO and IDIA
  - Used by collaborators in UK, Netherlands, Italy, India, etc
- User-friendly globus connect built on top of gridFTP
  - GUI/web app or CLI
- Can connect any arbitrary end points
  - Server (DTN), desktop, etc for Mac OS X, Windows & Linux
  - Offers user-friendly but computationally efficient transfer service

CARTA

- Cube Analysis and Rendering Tool for Astronomy
  - IDIA (South Africa) – NRAO (US) – ASIAA (Taiwan)
  - Cloud-based Visual analytic of remote large image cubes
  - Supports many image formats: FITS, CASA, Miriad, and HDF5
- CARTA on ilifu
  - http://docs.ilifu.ac.za/#/astronomy/astronomy_software?id=carta
  - https://carta.idia.ac.za – login with same credentials as Jupyter
  - The only visible files are in /carta_share/
  - User should write IDIA HDF5 file to this filesystem
  - Even for continuum (single channel) image
    - `srun /carta_share/hdf_convert/run_hdf_converter -o /carta_share/users/$USER/image.hdf5 image.fits`
- Demo time!

Best practices

- Don't run software / heavy processes / scp on the login node
  - Only submit jobs and run SLURM commands (sbatch, srun, squeue, etc)
  - Use transfer.ilifu.ac.za to transfer data (external/internal), not login node
- Before running a large job, identify the available resources
  - Use sinfo. Don't hog the cluster. Reduce your allocation if possible
  - Increase likelihood of jobs running with less memory and less walltime
- Use sbatch (srun / screen / tmux / mosh are volatile)
- Cleanup files that aren't needed
  - Old raw data, temporary products, /scratch data, etc
- Don't place large files in your home directory (/users)
- Use Singularity (you cannot install software on the nodes)

THANK YOU

Dr Jordan Collier

ilifu Support Astronomer, IDIA
Department of Astronomy,
University of Cape Town

Adjunct Fellow, School of Science,
Western Sydney University

Jordan@idia.ac.za

IDIA Inter-University Institute for Data Intensive Astronomy

UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

WESTERN SYDNEY UNIVERSITY