ilifu Online Training – Advanced #2 - Resource Allocation

Tinus Cloete

System Administrator & User Support, ilifu
University of Western Cape, March 2024

Inter-University Institute for Data Intensive Astronomy

ilifu: a shared resource-limited cluster

1. Supports a diverse range of projects
   - Astronomy and Bioinformatics
   - Varying resource requirements

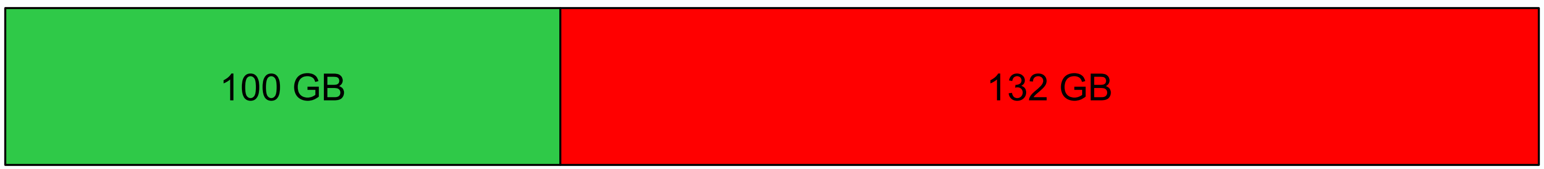
     e.g.
     CPU's, Memory,
     Running Time, GPU's etc.

2. Shared environment
3. Resource-limited

Efficient Usage of Resources

- Resource Allocation: "Picking the right amount of resources for your jobs"

  e.g.
  if a job uses 100 GB of RAM, don't want to request 232 GB



- Best practices: [Resource Allocation Guide]

Services and Partitions

Login node

Run SLURM & bash commands
cd, mkdir, ls, etc

Jupyter + JupyterDev + Devel

Development space
New code / workflows / routines
Debugging / testing software

Main partition

Stable, computationally heavy processing

HighMem/GPU

For single-high memory jobs or GPU resources

Services and Partitions: Jupyter

- Jupyter (Jupyter.ilifu.ac.za)
  - Development space for writing, testing and debugging
  - New code, software, workflows or routines
  - Highly interactive Jupyter notebook environment
  - May be primary interface for stable workflows that shouldn't use Slurm
    - ➢ short analysis routines or other highly interactive workflows

Launch Jupyter Lab

Hi tcloete. Remember to try and choose the smallest profile that fits your task. This helps us to make sure that everyone has access to the resources they need. Please visit the user documentation to learn more about Jupyter on ilifu. If you have any more questions, please send an email to ilifu support.

The following table shows the job profiles available on the ilifu cluster (as at 2024-09-17 06:40):

Job Profile	Available Jobs
GPU Session (16 cores, 1 GPU)	4
Minimum Session (1 core, dedicated)	84
Small Session (2 cores, dedicated)	42
Medium Session (4 cores, dedicated)	20
Large Session (8 cores, dedicated)	9
Half-Max Session (16 cores, dedicated)	4
Max Session (32 cores, dedicated)	1

Select a job profile:

Development Session - 2 core, 3 GB RAM, shared, 18 hrs idle timeout, max 14 days lifespan

Start

IDIA

UNIVERSITY of the WESTERN CAPE

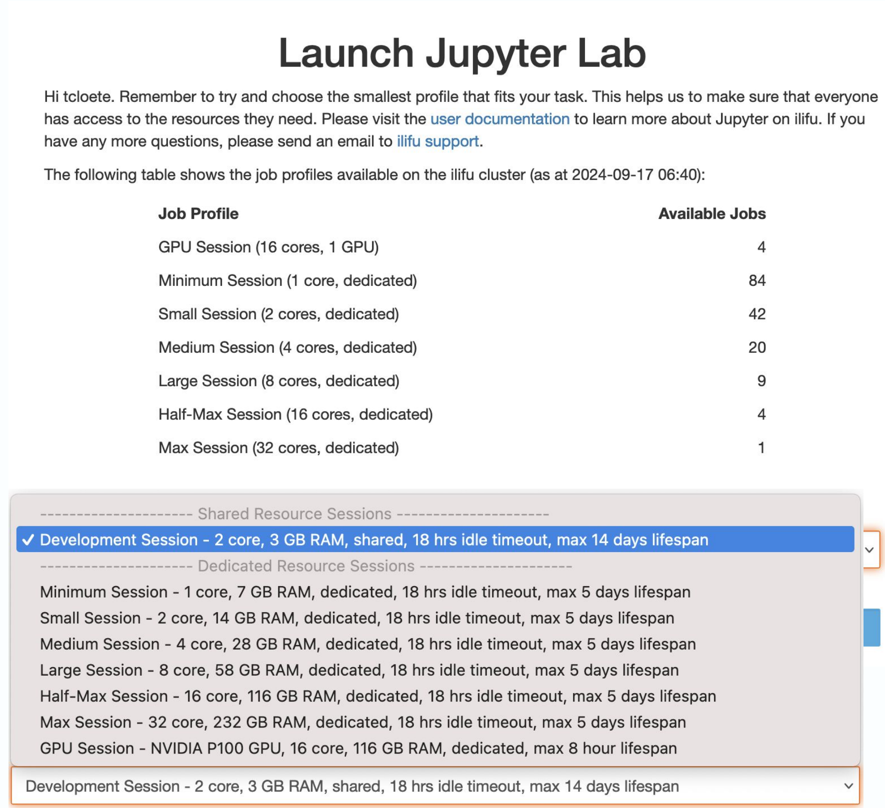
ilifu

Jupyter: Shared Resource Allocation

- Two types of Jupyter Sessions:
  - Shared Resource (1 node)
  - Dedicated Resource (12 nodes)

- Default: Development Session
  - On JupyterDev Node (jupyter-001)
  - 2 CPU cores + 3 GB RAM
  - Shares memory and CPU between users on node

- Aimed at lighter testing and development workflows that don't need dedicated resourc es (i.e. CPU and RAM).



Launch Jupyter Lab

Hi tcloete. Remember to try and choose the smallest profile that fits your task. This helps us to make sure that everyone has access to the resources they need. Please visit the user documentation to learn more about Jupyter on ilifu. If you have any more questions, please send an email to ilifu support.

The following table shows the job profiles available on the ilifu cluster (as at 2024-09-17 06:40):

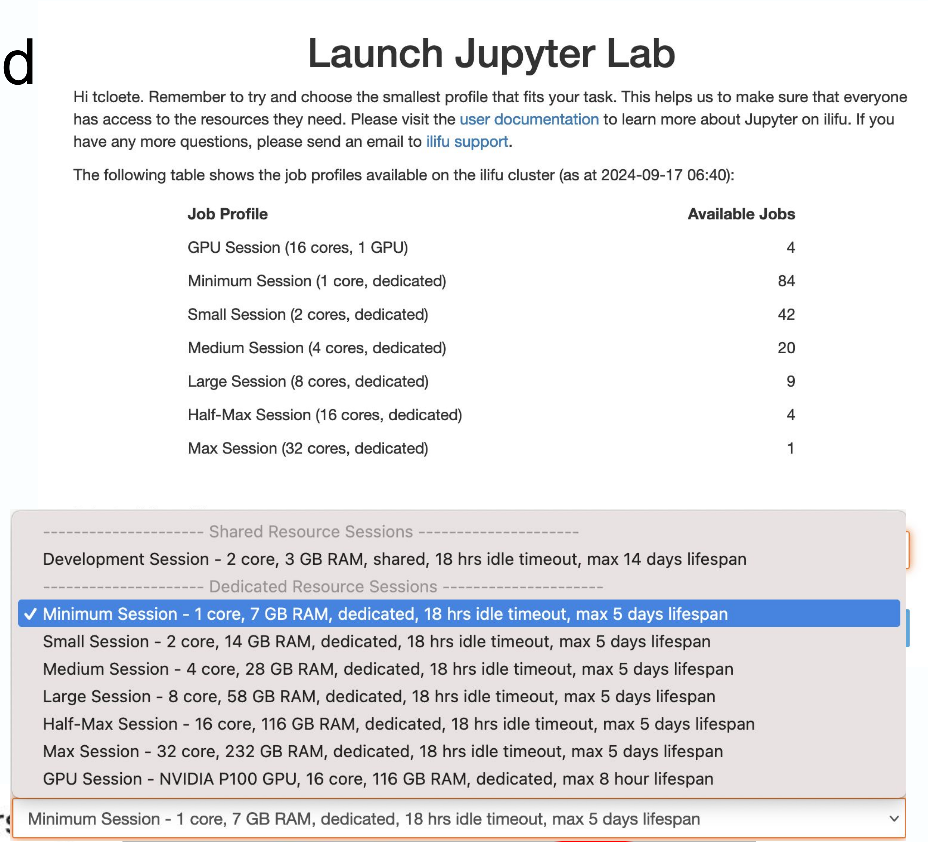
Job Profile	Available Jobs
GPU Session (16 cores, 1 GPU)	4
Minimum Session (1 core, dedicated)	84
Small Session (2 cores, dedicated)	42
Medium Session (4 cores, dedicated)	20
Large Session (8 cores, dedicated)	9
Half-Max Session (16 cores, dedicated)	4
Max Session (32 cores, dedicated)	1

-------------------- Shared Resource Sessions --------------------
✓ Development Session - 2 core, 3 GB RAM, shared, 18 hrs idle timeout, max 14 days lifespan
-------------------- Dedicated Resource Sessions --------------------
Minimum Session - 1 core, 7 GB RAM, dedicated, 18 hrs idle timeout, max 5 days lifespan
Small Session - 2 core, 14 GB RAM, dedicated, 18 hrs idle timeout, max 5 days lifespan
Medium Session - 4 core, 28 GB RAM, dedicated, 18 hrs idle timeout, max 5 days lifespan
Large Session - 8 core, 58 GB RAM, dedicated, 18 hrs idle timeout, max 5 days lifespan
Half-Max Session - 16 core, 116 GB RAM, dedicated, 18 hrs idle timeout, max 5 days lifespan
Max Session - 32 core, 232 GB RAM, dedicated, 18 hrs idle timeout, max 5 days lifespan
GPU Session - NVIDIA P100 GPU, 16 core, 116 GB RAM, dedicated, max 8 hour lifespan

Development Session - 2 core, 3 GB RAM, shared, 18 hrs idle timeout, max 14 days lifespan

Jupyter: Dedicated Resource Allocation

- Dedicated resources: CPU and Memory is allocated to you throughout the job duration
- Memory often most important
- Jupyter shows current **memory usage** at the bottom
- Emailed about usage stats e.g. low memory usage
- Shut down your session



Launch Jupyter Lab

Hi tcloete. Remember to try and choose the smallest profile that fits your task. This helps us to make sure that everyone has access to the resources they need. Please visit the user documentation to learn more about Jupyter on ilifu. If you have any more questions, please send an email to ilifu support.

The following table shows the job profiles available on the ilifu cluster (as at 2024-09-17 06:40):

Job Profile	Available Jobs
GPU Session (16 cores, 1 GPU)	4
Minimum Session (1 core, dedicated)	84
Small Session (2 cores, dedicated)	42
Medium Session (4 cores, dedicated)	20
Large Session (8 cores, dedicated)	9
Half-Max Session (16 cores, dedicated)	4
Max Session (32 cores, dedicated)	1

-------------------- Shared Resource Sessions --------------------
Development Session - 2 core, 3 GB RAM, shared, 18 hrs idle timeout, max 14 days lifespan
-------------------- Dedicated Resource Sessions --------------------
✓ Minimum Session - 1 core, 7 GB RAM, dedicated, 18 hrs idle timeout, max 5 days lifespan
Small Session - 2 core, 14 GB RAM, dedicated, 18 hrs idle timeout, max 5 days lifespan
Medium Session - 4 core, 28 GB RAM, dedicated, 18 hrs idle timeout, max 5 days lifespan
Large Session - 8 core, 58 GB RAM, dedicated, 18 hrs idle timeout, max 5 days lifespan
Half-Max Session - 16 core, 116 GB RAM, dedicated, 18 hrs idle timeout, max 5 days lifespan
Max Session - 32 core, 232 GB RAM, dedicated, 18 hrs idle timeout, max 5 days lifespan
GPU Session - NVIDIA P100 GPU, 16 core, 116 GB RAM, dedicated, max 8 hour lifespan

Minimum Session - 1 core, 7 GB RAM, dedicated, 18 hrs idle timeout, max 5 days lifespan

Services and partitions: Devel

- Devel (--partition=Devel)
  - Development of routines within shared resource environment
    - ➢ Submit jobs instantly / quickly
    - ➢ Resources shared, not solely allocated to your jobs
  - Interactivity via a shell
  - Generally for testing higher level workflows and pipelines
  - Access simply using the sinteractive command

```
tcloete@slurm-login:~$ sinteractive
Starting interactive Slurm session.
srun: job 9387238 queued and waiting for resources
srun: job 9387238 has been allocated resources
tcloete@compute-001:~$ 
```

Services and partitions: Main

- Main partition
  - Default Slurm partition
  - Generally for stable, computationally-heavy workflows and pipelines
  - Can be used for:
    - Many small jobs OR
    - A few large jobs allocated many resources
  - For large workflows, better to first test on Devel or Jupyter (e.g. subset of data / fewer iterations)

```
tcloete@slurm-login:~$ sinfo
PARTITION  AVAIL  TIMELIMIT  NODES  STATE NODELIST
Main*         up 14-00:00:0      3   plnd compute-[004,215,239]
Main*         up 14-00:00:0     35    mix compute-[009-010,015,101-105,201-202,204-206,210-211,213,216,218,222,230,235-237,240-242,246,248-251,253,257-259]
Main*         up 14-00:00:0     44  alloc compute-[002-003,005-008,011-014,016-018,203,207-209,212,214,217,219-221,223-229,231-234,238,243-245,247,252,254-25
6,260]
Jupyter       up   infinite      5    mix jupyter-[002-005,012]
Jupyter       up   infinite      6  alloc jupyter-[006-011]
Jupyter       up   infinite      1   idle jupyter-013
JupyterGPU    up 14-00:00:0      2   idle gpu-[003-004]
JupyterDev    up   infinite      1  alloc jupyter-001
HighMem       up 14-00:00:0      3  alloc highmem-[001-003]
GPU           up 14-00:00:0      7   idle gpu-[001-007]
Devel         up 5-00:00:00      1    mix compute-001
```

Services and partitions: GPU and HighMem

- HighMem partition
  – Single high-memory jobs that can't be split into multiple jobs using MPI
- GPU partition
  – Jobs making use of GPUs
  – Not for jobs that only require CPUs (rather use Main)

```
tcloete@slurm-login:~$ sinfo
PARTITION  AVAIL  TIMELIMIT  NODES  STATE NODELIST
Main*         up 14-00:00:0      3   plnd compute-[004,215,239]
Main*         up 14-00:00:0     35    mix compute-[009-010,015,101-105,201-202,204-206,210-211,213,216,218,222,230,235-237,240-242,246,248-251,253,257-259]
Main*         up 14-00:00:0     44  alloc compute-[002-003,005-008,011-014,016-018,203,207-209,212,214,217,219-221,223-229,231-234,238,243-245,247,252,254-25
6,260]
Jupyter       up   infinite      5    mix jupyter-[002-005,012]
Jupyter       up   infinite      6  alloc jupyter-[006-011]
Jupyter       up   infinite      1   idle jupyter-013
JupyterGPU    up 14-00:00:0      2   idle gpu-[003-004]
JupyterDev    up   infinite      1  alloc jupyter-001
HighMem       up 14-00:00:0      3  alloc highmem-[001-003]
GPU           up 14-00:00:0      7   idle gpu-[001-007]
Devel         up 5-00:00:00      1    mix compute-001
```

Primary Resources

- http://docs.ilifu.ac.za/#/tech_docs/resource_allocation
- Primary resources
  1. CPU
  2. Memory
  3. Wall-time
- Notes
  - Nodes have 2 CPUs (sockets), each with 16 cores, all of which Slurm calls “CPUs”

Allocating Resources

- How to allocate resources
  - Accurately determine your resource requirements
  - Use what you require
- Effect
  - Avoid wasting resources (allocated but not used)
  - Increase resource availability
  - Allow other users' jobs to run
  - Improves efficiency of Slurm scheduler
  - Decreased job wait times
  - Better fairshare priority for future job submissions.

Determining resource requirements

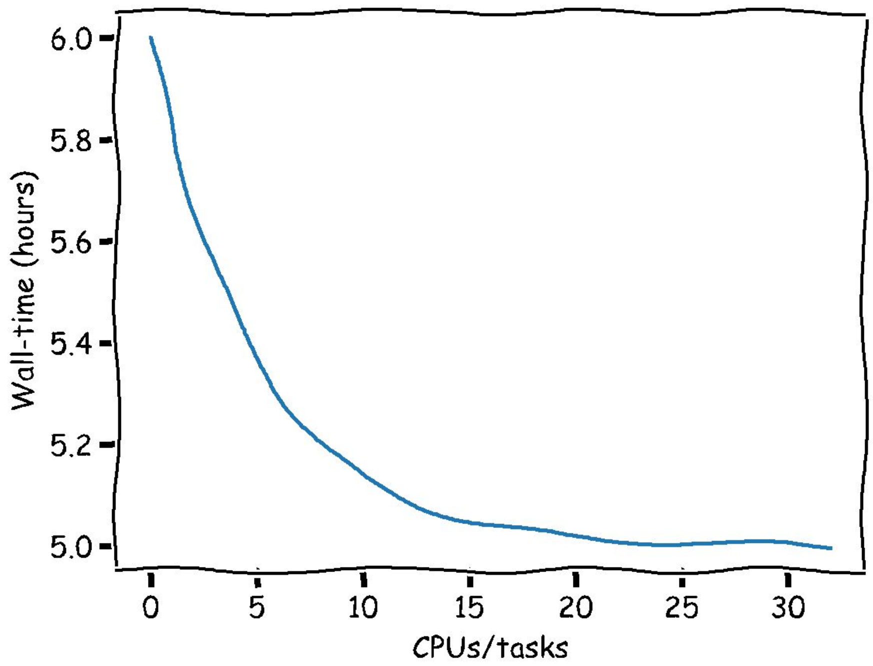
1. Determine parallelism of software
2. Profiling previous similar jobs
3. Scaling up test jobs

Determining resource requirements

- Determining parallelism of software
  - Many software packages only use 1 CPU
  - CPU-level parallelism: Max 1 Node of CPUs
  - Task-level parallelism: >= 1 Node

Determining resource requirements

- Determining parallelism of software
  - Most parallel processing software doesn't scale linearly
  - Maximum performance often least efficient
    - i.e. shortest wall-time but large allocation necessary
  - Need to find middle ground
  - MPI jobs may perform worse for larger allocations (scatter/gather)
  - Most efficient generally to break into many small independent jobs
    - High-throughput approach



Profiling previous similar jobs

- Find job ID
  - Job id is shown when you submitted your job
  - Can search for historical jobs
  - Display jobs named ‘my-job’ submitted during particular time range:
  - `sacct -X --name=my-job --starttime=YYYY-MM-DD --endtime=YYYY-MM-DD`
  - Omit job name (or end time) to show all jobs

- Once you have job ID, you can search for specific information about resource usage

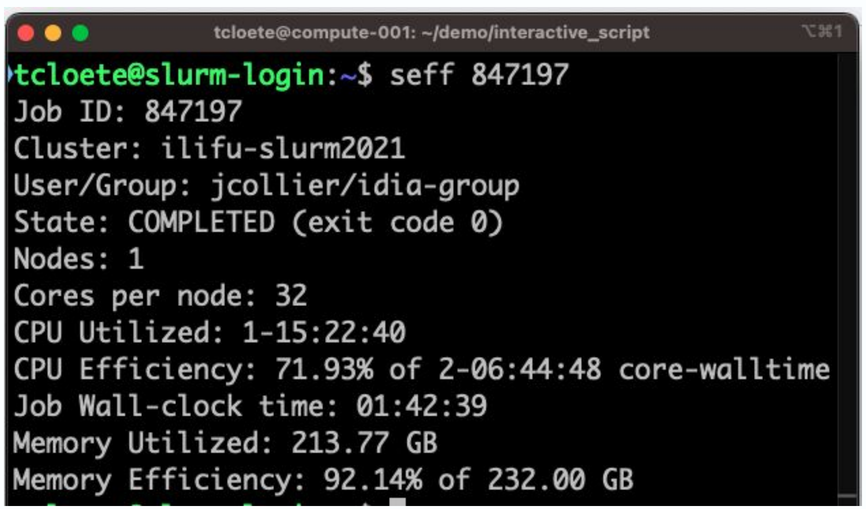
Profiling previous similar jobs

- Memory usage
  - Find MaxRSS statistic
    - Maximum memory usage of a job (sampled every 20 seconds)
    - Display MaxRSS for job ID 123456 compared to requested memory
    - `sacct -j 123456 --unit=G -o JobID,JobName,MaxRSS,ReqMem`
    - *Notes: 232 Gn = 232 GB per node; 7.25 Gc = 7.25 GB per CPU*
  - Once memory requirement determined
    - Schedule future jobs with **~10-20% buffer**
      - Avoids out-of-memory (OOM) error
    - Avoid excessive usage of memory

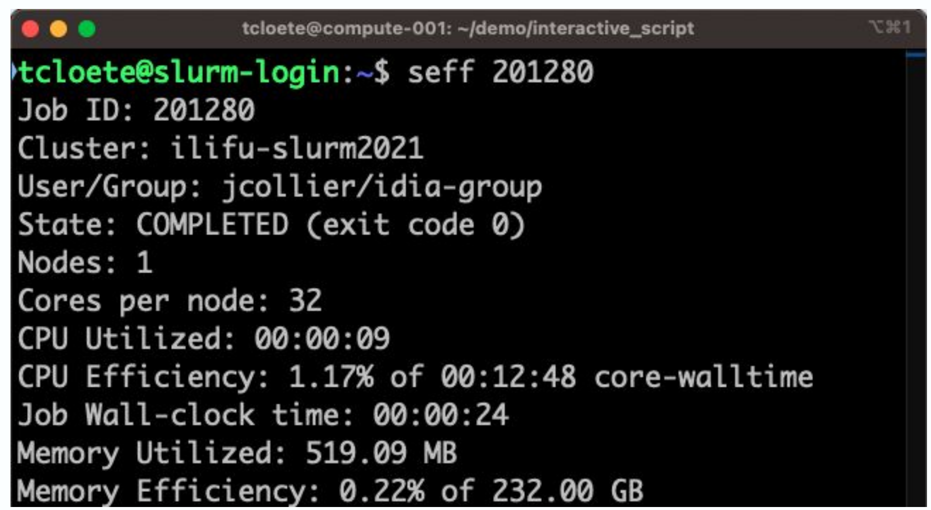
```
tcloete@slurm-login:~$ sacct -j 847197 --unit=G
-o JobID,JobName,MaxRSS,ReqMem
JobID           JobName     MaxRSS     ReqMem
------------ ---------- ---------- ----------
847197       selfcal_p+                  232G
847197.batch      batch    213.77G
847197.exte+     extern          0
```

Profiling previous similar jobs

- CPU (and memory) usage
  - Determine used vs. allocated/requested
  - Show Slurm resource efficiency (seff) for job ID 123456
  - Shows % used vs. allocated (for memory, uses MaxRSS stat)
  - `seff 123456`
  - Can run this from Jupyter terminal (to determine resource selection)



```
tcloete@slurm-login:~$ seff 847197
Job ID: 847197
Cluster: ilifu-slurm2021
User/Group: jcollier/idia-group
State: COMPLETED (exit code 0)
Nodes: 1
Cores per node: 32
CPU Utilized: 1-15:22:40
CPU Efficiency: 71.93% of 2-06:44:48 core-walltime
Job Wall-clock time: 01:42:39
Memory Utilized: 213.77 GB
Memory Efficiency: 92.14% of 232.00 GB
```



```
tcloete@slurm-login:~$ seff 201280
Job ID: 201280
Cluster: ilifu-slurm2021
User/Group: jcollier/idia-group
State: COMPLETED (exit code 0)
Nodes: 1
Cores per node: 32
CPU Utilized: 00:00:09
CPU Efficiency: 1.17% of 00:12:48 core-walltime
Job Wall-clock time: 00:00:24
Memory Utilized: 519.09 MB
Memory Efficiency: 0.22% of 232.00 GB
```

Profiling previous similar jobs

- Wall-time usage
  - Accurate estimation improves Slurm scheduler efficiency and may reduce your job wait time
  - Show used vs. requested wall-time for job ID 123456
  - `sacct -o jobID,jobName,Elapsed,TimeLimit`
  - Once wall-time requirement determined
    - Schedule future jobs with **~20-30% buffer** (avoids job timing out)
    - Avoid excessive wall-time
    - Contact support@ilifu.ac.za to see if we may increase your time limit

```
tcloete@compute-001: ~/demo/interactive_script
tcloete@slurm-login:~$ sacct -j 838338 -o jobID,jobName,Elapsed,TimeLimit
JobID           JobName    Elapsed  Timelimit
------------ ---------- ---------- ----------
838338       quick_tcl+   00:21:12   01:00:00
838338.batch      batch   00:21:12
838338.exte+     extern   00:21:12
```

Scaling tests

- Accurately estimating wall-time difficult to do
- Profile previous similar jobs, or
- Run test / scaling jobs
  - Start small test job (e.g. small subset of data)
  - Test the wall-time
    - Reasonable to over-allocate when running scaling test
    - Or if under-estimate, and test small enough, doesn't matter if crashes
  - Repeat process to see how resource usage scales
    - as a function of input (e.g. data volume)
    - as a function of CPUs / tasks (if doing parallel processing)
  - By the end, should have good idea of scaling and efficient choice
    - Allow for buffer for future jobs

Scaling tests on running jobs

- Get MaxRSS for running job
  - `sstat -j 123456 -o MaxRSS`
  - Given in kB units. Divide by 1024^2 for GB

- Display real time stats on dashboard (`top` / `htop`)
  - <u>For sbatch</u>:
    First ssh into the login node using authentication forwarding.
    ```
    ssh -A <username>@slurm.ilifu.ac.za
    ```
    It's required to have a job running on a worker node.
    You can then ssh into that worker node (e.g. node 102)
    ```
    ssh compute-102
    ```
  - <u>For Jupyter</u>: can simply open a new terminal.
  - Now Run: `htop -u $USER`

- Can monitor real-time usage

Maximum Resources

- If using **all** CPUs or memory, node becomes fully allocated
  - Any remaining CPUs / memory unavailable to other jobs (incl. your own)

E.g.
Typical worker node: 32 CPU and 232 GB RAM

Job Requesting: 2 CPU and 232 GB RAM == Full Node
30 CPU **not accessible to other jobs**

If possible to split into two smaller jobs, if they ran on different nodes then:

1 CPU and 116 GB	1 CPU and 116 GB
31 CPUs accessible	31 CPUs accessible

Account allocation

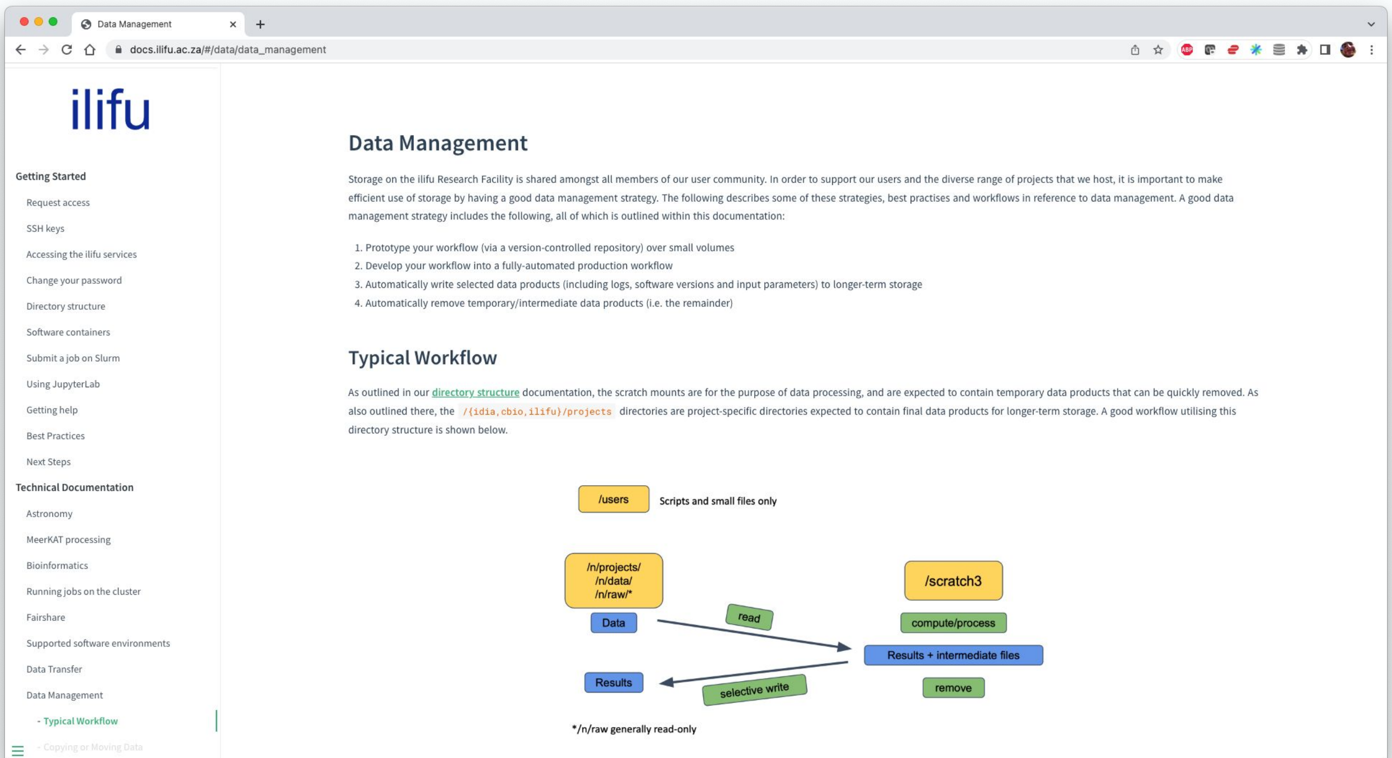
- Each ilifu project has a Slurm account
- Resource usage charged against account (affects fairshare)
- View your accounts:
  - `shelp`
- View your default account
  - `sacctmgr show user $USER`
- Change default
  - `sacctmgr modify user name=${USER} set DefaultAccount=<account>`
- Set account (after #SBATCH for sbatch jobs)
  - `--account=b05-pipelines-ag`

Resource Allocation Guide

DEMO TIME!

Data Management Guidelines

- Hot off the press!
- https://docs.ilifu.ac.za/#/data/data_management



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!

Thanks to Jordan Collier for letting me use his Slides

Remember our support channels!

support@ilifu.ac.za
https://docs.ilifu.ac.za