Rotman Research Node (RRN)

Jay / TDMDAL

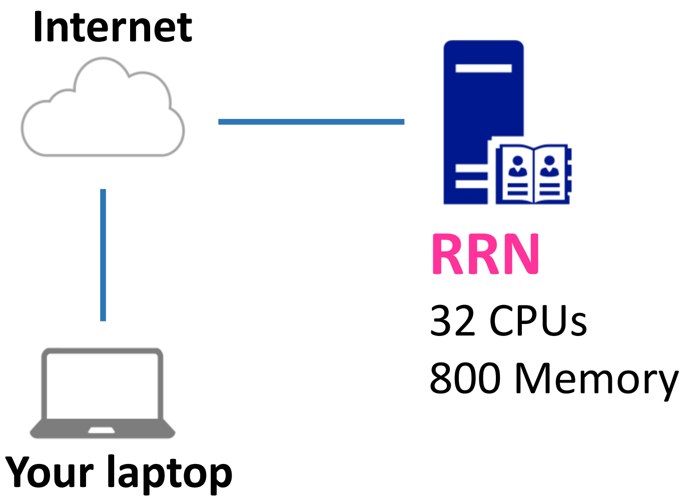
Workshop website: https://tdmdal.github.io/rrn-workshop/

Plan for Today

- Agenda
  - What is RRN?
  - When will you choose to use it?
  - How to use it depending on your use cases (demo)?

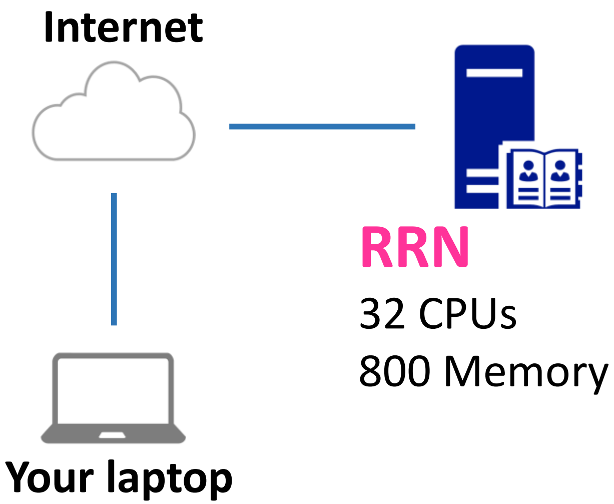
- We will focus on big pictures so
  - You get a good general understanding of the system
  - You know where to look for the details and what details
    - user manuals, Internet, TDMDAL support, etc.

What is RNN (1)

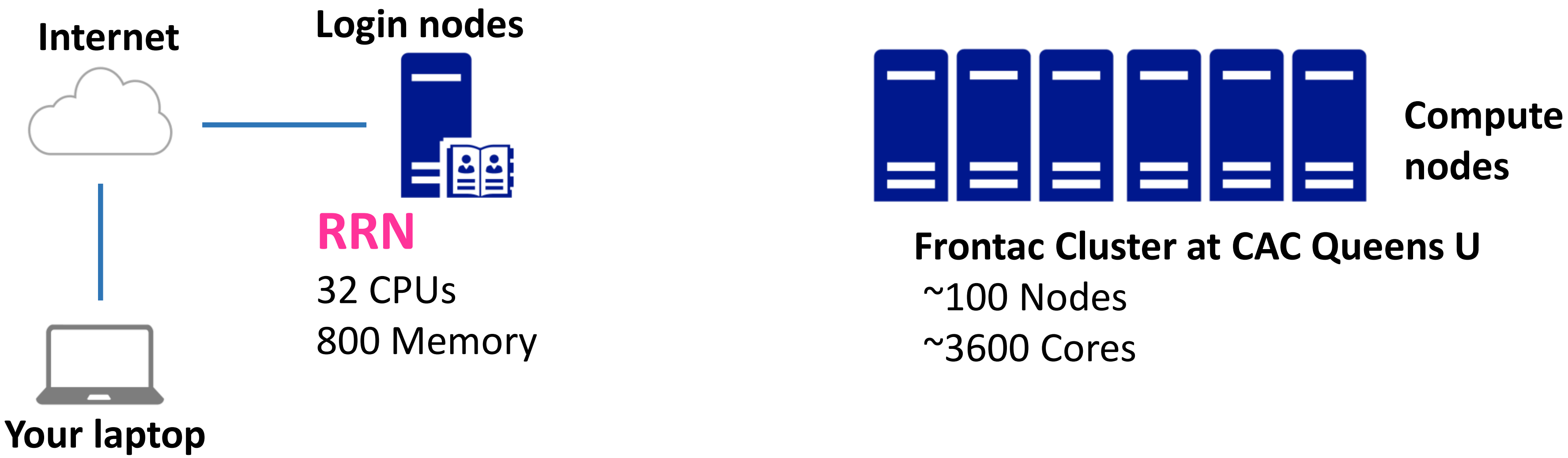


- A shared research server
  - Hardware: 32 Xeon CPUs; 800G Memory
  - Storage: Home directories; 50T shared project directory
  - Software: Linux OS; Python, R, Matlab, Stata, Julia, C, C++, Fortran, etc.
  - Dedicated to Rotman researchers
  - Hosted at Centre for Advanced Computing (CAC) at Queens U
- A gateway to a pool of computing resources
  - Zoom out (next slides)

What's RRN (Zoom Out)



What’s RRN (Zoom Out)



Internet

Login nodes

RRN

32 CPUs

800 Memory

Your laptop

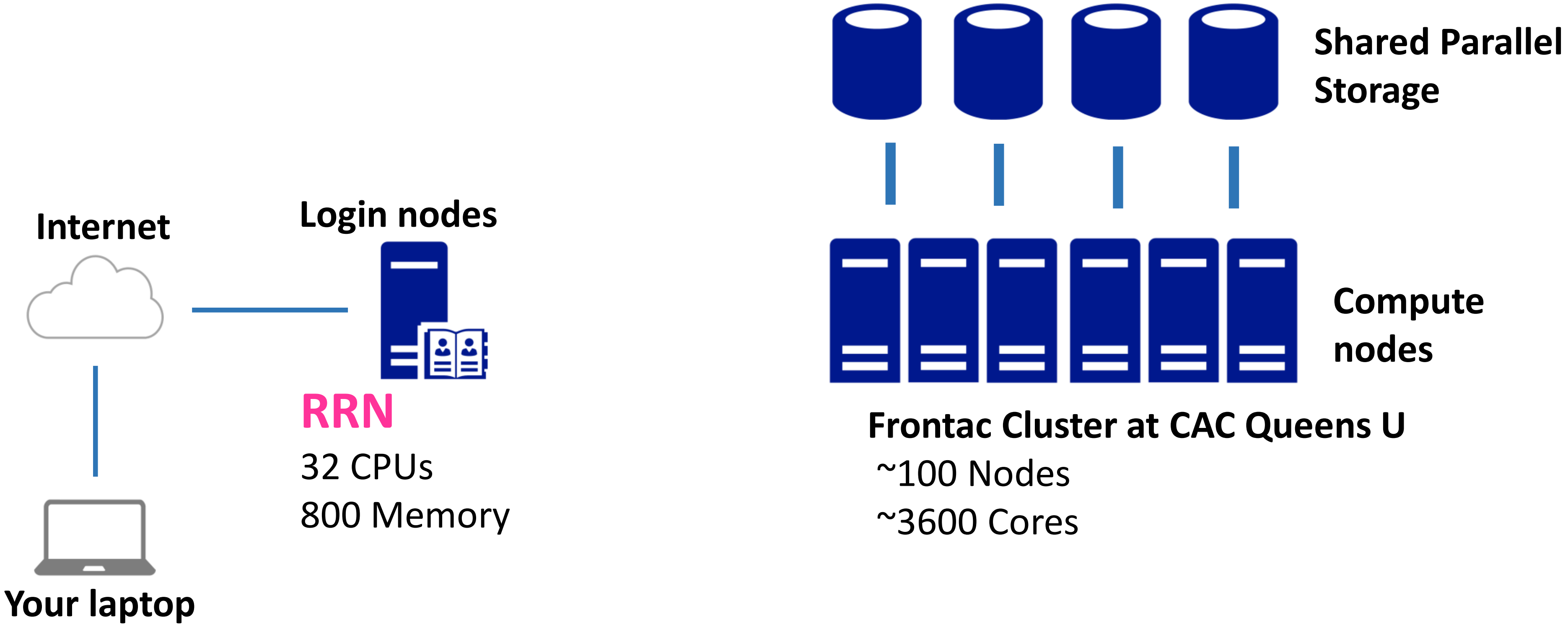
Compute nodes

Frontac Cluster at CAC Queens U

~100 Nodes

~3600 Cores

What’s RRN (Zoom Out)



Internet

Your laptop

Login nodes

RRN

32 CPUs

800 Memory

Shared Parallel Storage

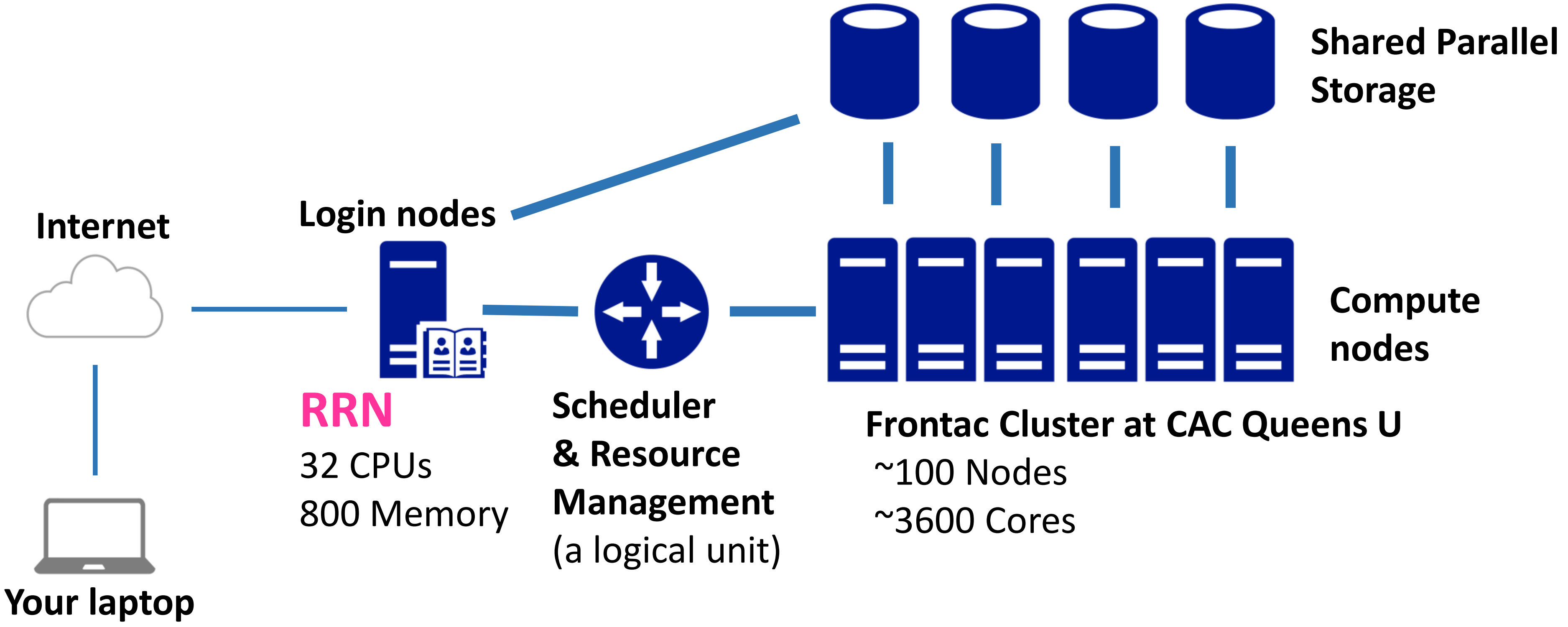
Compute nodes

Frontac Cluster at CAC Queens U

~100 Nodes

~3600 Cores

What’s RRN (Zoom Out)

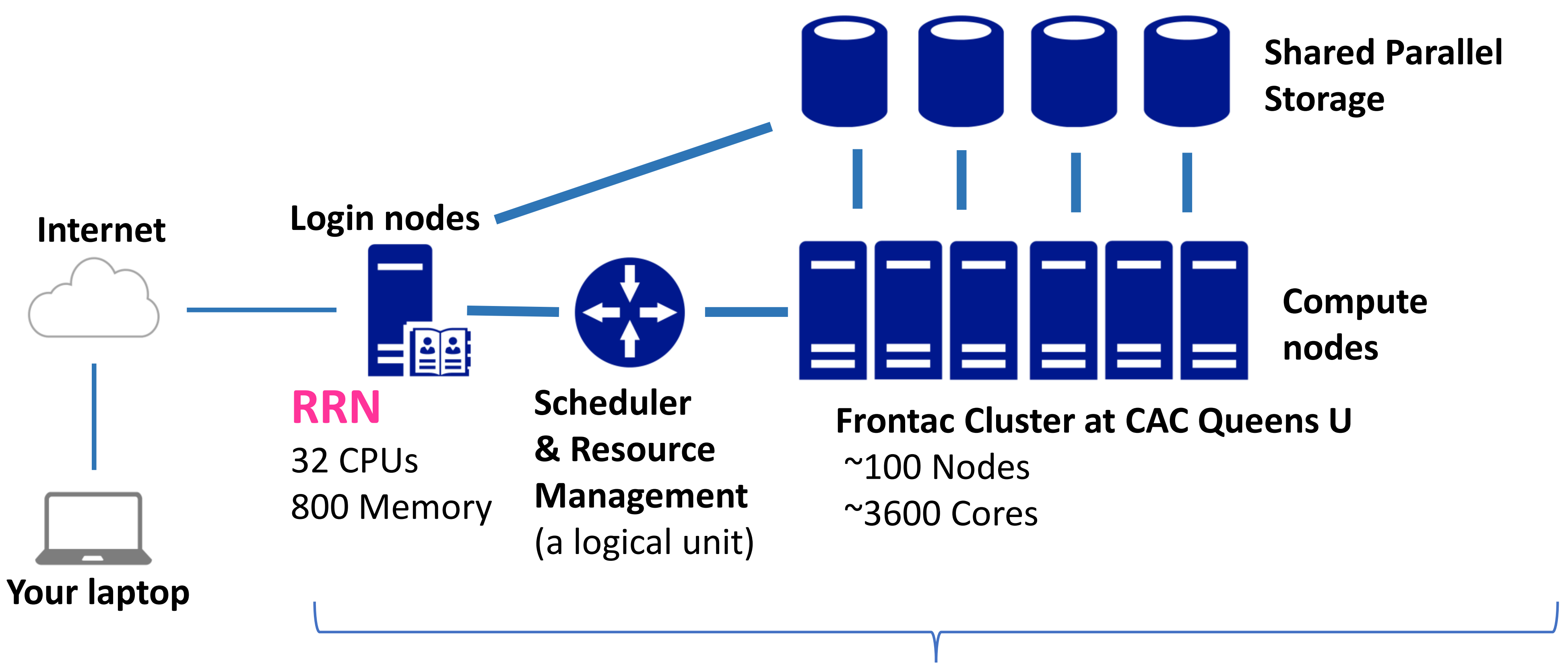


RRN
32 CPUs
800 Memory

Scheduler
& Resource
Management
(a logical unit)

Frontac Cluster at CAC Queens U
~100 Nodes
~3600 Cores

What’s RRN (Zoom Out)



High Performance Computing System (HPC)

Why using RRN

- My code takes too long to run
  - it need more CPUs (i.e. computing intensive work)
  - it need more memory (i.e. memory intensive work)

- I want to use XYZ (ex. Matlab), but the license cost is too high

- I want a stable environment for a long running code
  - ex. collecting data through web scraping

Questions to ask before considering RRN

- One big misunderstanding
  - my (unmodified) code will run faster on the server (**NO** in most cases)

- I need more CPUs
  - have I optimized my code (vectorization; better algorithm)?
  - have I tried parallel computing on my desktop?
    - modern desktop has 2-4 CPUs (4-8 with hyperthreading)
    - no license cost if you use open source language (R, Python, Julia, etc.)

- I need more memory
  - do I really need to load all those data
  - have I optimized my code (delete big variables/objects after use; better algorithm)

How to Use RRN – User Manuals

- Our RRN User Manual
  - Focus on RRN
  - Good for getting started

- CAC User manual
  - Including HPC usage (Compute Nodes)
  - For users with highly compute-intensive jobs

How to Use RRN – Account & Logon

- Email tdmdal@rotman.utoronto.ca for an account

- SSH (Secure Shell) client
  - Windows (ex. Mobaxterm)
  - Mac (terminal + Xquartz)

```
ssh -X yourUserName@rrlogin.cac.queensu.ca
```

https://tdmdal.github.io/computing-research/getting-started.html#logging-on-to-rrn

Folder Structure & Disk Quota

Folder	Path	Quota	Usage
Home	/global/home/yourUserName/	3T	Main storage
Project	/global/project/rotman_research/yourUserName/	50T shared	Additional storage
Scratch	/global/scratch/yourUserName/	5T	Temporary storage

Note: 1) Only you have access to those 3 folders
2) Your project folder shares the 50T quota with other Rotman project folders
3) project and scratch folders can also be accessed via two shortcuts in the home folder: **rotman_research** and **scratch**

How to Use RRN – Transfer Files

- a SFTP client (transfer files between local PC and RRN)
  - Windows (ex. WinSCP, Cyberduck, FileZilla)
  - Mac (Cyberduck, FileZilla)

- Other methods available too
  - scp, rsync, etc.
  - globus
  - …

https://tdmdal.github.io/computing-research/getting-started.html#transferring-files
https://cac.queensu.ca/wiki/index.php/UploadingFiles:Frontenac

Using Software – Module System

- A software Environment Module system

```
module avail

module load <module_name>

module list
```

https://tdmdal.github.io/computing-research/getting-started.html#loading-software

When to Use What

	Interactive Mode	Batch Mode
RRN (1 node)		
Compute Nodes (Many nodes)		

When to Use What

	Interactive Mode	Batch Mode
RRN (1 node) • CPU: 32 cores • Memory: 800G; quite large • **Shared** (among Rotman researchers) • Easy to use: Code runs as soon as you ask it to run		
Compute Nodes (Many nodes)		

When to Use What

	Interactive Mode	Batch Mode
RRN (1 node) • CPU: 32 cores • Memory: 800G; quite large • **Shared** (among Rotman researchers) • Easy to use: Code runs as soon as you ask it to run		
Compute Nodes (Many nodes) • CPU: 24- up to 128-core node • Memory: mostly 256G nodes; a few large ones (512G, 1T & 2T) • **Exclusive** use once allocated to you • Slightly harder to use • Need to write a **script** to talk to the schedular/resource manager • Need to wait in a **queue** to compete for resources		

When to Use What

	Interactive Mode	Batch Mode
RRN (1 node) - CPU: 32 cores - Memory: 800G; quite large - **Shared** (among Rotman researchers) - Easy to use: Code runs as soon as you ask it to run	Debug code Run small jobs - < 6 cores - < 0.5 hrs.	
Compute Nodes (Many nodes) - CPU: 24- up to 128-core node - Memory: mostly 256G nodes; a few large ones (512G, 1T & 2T) - **Exclusive** use once allocated to you - Slightly harder to use - Need to write a **script** to talk to the schedular/resource manager - Need to wait in a **queue** to compete for resources		

Ex.1 Matlab - RNN Interactive Mode

- Load matlab module

```
module load matlab/R2018b
```

- Run matlab with GUI

```
matlab
```

- Run matlab without GUI

```
matlab -nodesktop -nosplash -nodisplay
```

When to Use What

	Interactive Mode	Batch Mode
RRN (1 node) • CPU: 32 cores • Memory: 800G; quite large • **Shared** (among Rotman researchers) • Easy to use: Code runs as soon as you ask it to run	Debug code Run small jobs • < 6 cores • < 0.5 hrs.	Run intermediate compute & memory intensive jobs • ~12 cores, 1-2 hrs. or … • ~1-2 cores, longer hrs.
Compute Nodes (Many nodes) • CPU: 24- up to 128-core node • Memory: mostly 256G nodes; a few large ones (512G, 1T & 2T) • **Exclusive** use once allocated to you • Slightly harder to use • Need to write a **script** to talk to the schedular/resource manager • Need to wait in a **queue** to compete for resources		

Ex.2 Matlab - RNN Batch Mode

- Load matlab module

```
module load matlab/R2018b
```

- Run in foreground
  - prompt taken; need to wait for result; **not recommend**

```
matlab -nodesktop -nosplash -nodisplay <matlab_test.m &>matlab_test.log
```

- Run in background and no hang up after logout (**recommend**)

```
nohup matlab -nodesktop -nosplash -nodisplay <matlab_test.m &>matlab_test.log &
```

When to Use What

	Interactive Mode	Batch Mode
RRN (1 node) • CPU: 32 cores • Memory: 800G; quite large • **Shared** (among Rotman researchers) • Easy to use: Code runs as soon as you ask it to run	Debug code Run small jobs • < 6 cores • < 0.5 hrs.	Run intermediate compute & memory intensive jobs • ~12 cores, 1-2 hrs. or … • ~1-2 cores, longer hrs.
Compute Nodes (Many nodes) • CPU: 24- up to 128-core node • Memory: mostly 256G nodes; a few large ones (512G, 1T & 2T) • **Exclusive** use once allocated to you • Slightly harder to use • Need to write a **script** to talk to the schedular/resource manager • Need to wait in a **queue** to compete for resources	Get **a interactive node** to: Debug code Run small jobs • CPUs: as many as you request on a node • < 1 hrs.	

Ex.3 Matlab – Interactive Compute Node

- Allocate an interactive node

```
salloc -c 4 --mem=8g
```

- Load matlab module

```
module load matlab/R2018b
```

- Run matlab in interactive mode with or without GUI
- Run matlab in batch mode. **However, don’t exit the interactive node!**

When to Use What

	Interactive Mode	**Batch Mode**
RRN (1 node) • CPU: 32 cores • Memory: 800G; quite large • **Shared** (among Rotman researchers) • Easy to use: Code runs as soon as you ask it to run	Debug code Run small jobs • < 6 cores • < 0.5 hrs.	Run intermediate compute & memory intensive jobs • ~12 cores, 1-2 hrs. or … • ~1-2 cores, longer hrs.
Compute Nodes (Many nodes) • CPU: 24- up to 128-core node • Memory: mostly 256G nodes; a few large ones (512G, 1T & 2T) • **Exclusive** use once allocated to you • Slightly harder to use • Need to write a **script** to talk to the schedular/resource manager • Need to wait in a **queue** to compete for resources	Get **a interactive node** to: Debug code Run small jobs • CPUs: as many as you request on a node • < 1 hrs.	Run highly compute & memory intensive jobs

Ex.4 Matlab – Batch Compute Node (1)

- Write a job submission script (an example, **`job.sh`**)

use bash shell to execute this script

```
#!/bin/bash
```

request resources (Slurm command)

```
#SBATCH --job-name=MATLAB_test
#SBATCH --partition=standard
#SBATCH --mail-type=ALL
#SBATCH --mail-user=jay.cao@rotman.utoronto.ca
#SBATCH --output=STD.out
#SBATCH --error=STD.err
#SBATCH -c 4
#SBATCH --time=30:00
#SBATCH --mem=5000
```

load matlab & run my code

```
module load matlab/R2018b
matlab -nodesktop -nosplash -nodisplay <matlab_test.m
```

Ex.4 Matlab – Batch Compute Node (2)

```
#!/bin/bash

#SBATCH --job-name=MATLAB_test   # set job name
#SBATCH --partition=standard     # set job partition (group of nodes)
#SBATCH --mail-type=ALL          # email me when job start, stop, etc.
#SBATCH --mail-user=jay.cao@rotman.utoronto.ca
#SBATCH --output=STD.out         # save standard output to STD.out
#SBATCH --error=STD.err          # save std. error output to STD.out
#SBATCH -c 4                     # ask for 4 CPUs
#SBATCH --time=30:00             # set wall time to be 30mins
#SBATCH --mem=5000               # ask for 5G memory

module load matlab/R2018b
matlab -nodesktop -nosplash -nodisplay <matlab_test.m
```

https://cac.queensu.ca/wiki/index.php/SLURM

Ex.4 Matlab – Batch Compute Node (2)

```
#!/bin/bash

#SBATCH --job-name=MATLAB_test   # set job name
#SBATCH --partition=standard     # set job partition (group of nodes)
#SBATCH --mail-type=ALL          # email me when job start, stop, etc.
#SBATCH --mail-user=jay.cao@rotman.utoronto.ca # set my email address
#SBATCH --output=STD.out         # save standard output to STD.out
#SBATCH --error=STD.err          # save std. error output to STD.out
#SBATCH -c 4                     # ask for 4 CPUs
#SBATCH --time=30:00             # set wall time to be 30mins
#SBATCH --mem=5000               # ask for 5G memory

module load matlab/R2018b
matlab -nodesktop -nosplash -nodisplay <matlab_test.m
```

https://cac.queensu.ca/wiki/index.php/SLURM

Ex.4 Matlab – Batch Compute Node (2)

```
#!/bin/bash

#SBATCH --job-name=MATLAB_test   # set job name
#SBATCH --partition=standard     # set job partition (group of nodes)
#SBATCH --mail-type=ALL          # email me when job start, stop, etc.
#SBATCH --mail-user=jay.cao@rotman.utoronto.ca # set my email address
#SBATCH --output=STD.out         # save standard output to STD.out
#SBATCH --error=STD.err          # save std. error output to STD.out
#SBATCH -c 4                     # ask for 4 CPUs
#SBATCH --time=30:00             # set wall time to be 30mins
#SBATCH --mem=5000               # ask for 5G memory

module load matlab/R2018b
matlab -nodesktop -nosplash -nodisplay <matlab_test.m
```

https://cac.queensu.ca/wiki/index.php/SLURM

Ex.4 Matlab – Batch Compute Node (2)

```
#!/bin/bash

#SBATCH --job-name=MATLAB_test   # set job name
#SBATCH --partition=standard     # set job partition (group of nodes)
#SBATCH --mail-type=ALL          # email me when job start, stop, etc.
#SBATCH --mail-user=jay.cao@rotman.utoronto.ca # set my email address
#SBATCH --output=STD.out         # save standard output to STD.out
#SBATCH --error=STD.err          # save std. error output to STD.out
#SBATCH -c 4                     # ask for 4 CPUs
#SBATCH --time=30:00             # set wall time to be 30mins
#SBATCH --mem=5000               # ask for 5G memory

module load matlab/R2018b
matlab -nodesktop -nosplash -nodisplay <matlab_test.m
```

https://cac.queensu.ca/wiki/index.php/SLURM

Ex.4 Matlab – Batch Compute Node (3)

- run job script to submit your code to a compute node
  - currently CAC doesn’t enable inter-node jobs

```
sbatch job.sh
```

- Show status of jobs

```
squeue --job <job_id>
```

- Cancel jobs

```
scancel <job_id>
```

https://cac.queensu.ca/wiki/index.php/SLURM

Extra Stuff

Persist Your Sessions: tmux & x2go

- What does it mean
  - after you log out the system, you can still log in back to where you left off

- Why is it useful
  - long running code on RRN in foreground

- Tools to achieve it
  - persist a **non-GUI** session: **tmux** (recommended; demo) or screen
  - persist a **GUI** session: **x2go** (recommended; demo), VNC or xpra

tmux: minimum to get started

launch: **`tmux`**

split current pane vertically: **`ctrl-b %`**

split current pane horizontally: **`ctrl-b “`**

moving between panes: **`ctrl-b ↑, ↓, →, ←`**

close a pane (close the last pane to exit tmux): **`exit`**

detach from a session: **`ctrl-b d`**

re-attach to a session (assuming you only have 1 session): **`tmux attach`**

more on getting started with tmux: https://www.hamvocke.com/blog/a-quick-and-easy-guide-to-tmux/

Processes: ps, top, htop, kill, pkill

- check all the process you are running

```
ps -u yourUserName
```

- display system info (CPU & memory usage, process, etc)

`top or htop` (type q to exit)

- terminate a process

```
kill processID
```

- terminate all processes you have (this will log you out too)

```
pkill –u yourUserName
```

more on htop: https://codeahoy.com/2017/01/20/hhtop-explained-visually/