WELCOME

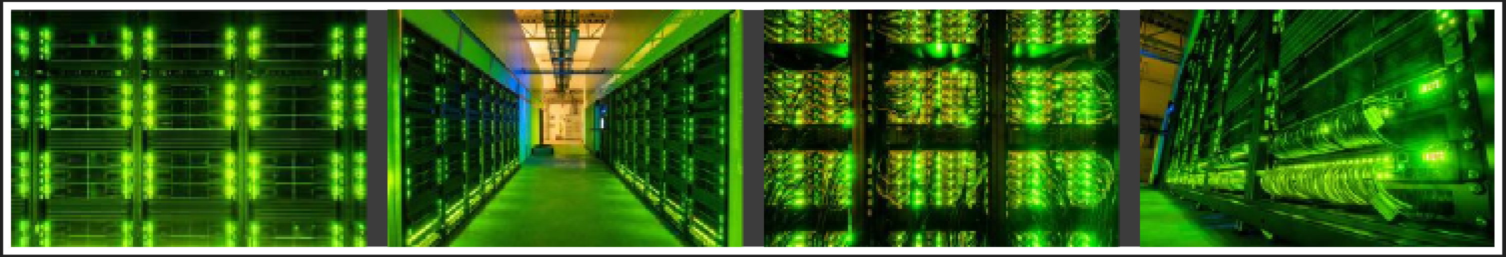
TETRALITH TRAINING SCHEDULE

Topics we will go through today:

- Tetralith introduction [Hamish]
- Accessing and setting up [Torben]
- Python environments [Torben]
- Working on Tetralith through Jupyter Notebooks [Xin]
- Building software [Peter]
- Running applications [Peter]
- Resource utilization [Peter]
- Hidden inefficiencies [Hamish]
- Getting support [Hamish]

INTRODUCING TETRALITH

TETRALITH PRESS RELEASE



TIMELINE

- Phase 1 (700 nodes)
    - 2018-07-10: NSC stability and stress testing starts
    - 2018-08-23: Tetralith and Sigma opened for users
- Phase 2 (1192 nodes)
    - 2018-12-01 (estimate): NSC stability and stress testing starts
    - 2019-01-01 (estimate): Full system (phase 1 + phase 2) open for users

TETRALITH / TRIOLITH COMPARISON

	Tetralith	Triolith
# compute nodes	1892	1600
Processors	Intel Xeon Gold 6130	Intel Xeon E5-2660
# cores per node	32	16
Memory (thin nodes)	96 GB (3 GB per core)	32 GB (2 GB per core)
# Fat nodes	60 (384 GB memory)	48 (128 GB memory)
Interconnect	Intel Omni-Path Architecture	Mellanox Infiniband FDR
Operating system	CentOS Linux 7	CentOS Linux 6
Peak performance	4 PFlops	450 TFlops
Memory bandwidth	350 TB/s	125 TB/s
Power consumption	720 kW (estimate)	500 kW
Performance efficiency	5.7 GFlops/W	0.9 GFlops/W

WHAT IS NEW WITH TETRALITH

WHAT ARE YOU LIKELY TO NOTICE?

- Hierarchical module system for developing and building software
- 32 cores per node, 3 GB memory per core
- No *huge* nodes, no GPU nodes (for now)
- `/software` file system and software installations (modules)

WHAT IS NEW WITH TETRALITH

WHAT IS NOT SO OBVIOUS

- CentOS7 operating system
- OPA interconnect
- (Easybuild)

WHAT HAS NOT CHANGED

- `/home` and `/proj` file systems
- Flat structure of the end user software modules, e.g.

```
$ module purge
$ module avail
$ module load star-ccm+/13.06.011
$ <run your star-ccm+ stuff>
```

ACCESSING NSC SYSTEMS

Accessing and using the Tetralith login nodes.

This tutorial will focus on how to access the system and getting familiar with the module system, file systems, etc.

LOGIN NODES

- Tetralith has two login nodes - tetralith1 and tetralith2
- You can, but generally shouldn't, specify what login node to connect to. There are occasions where you do want to do this, though.

Oberserve, that for ThinLinc you cannot specify what login node to connect to!

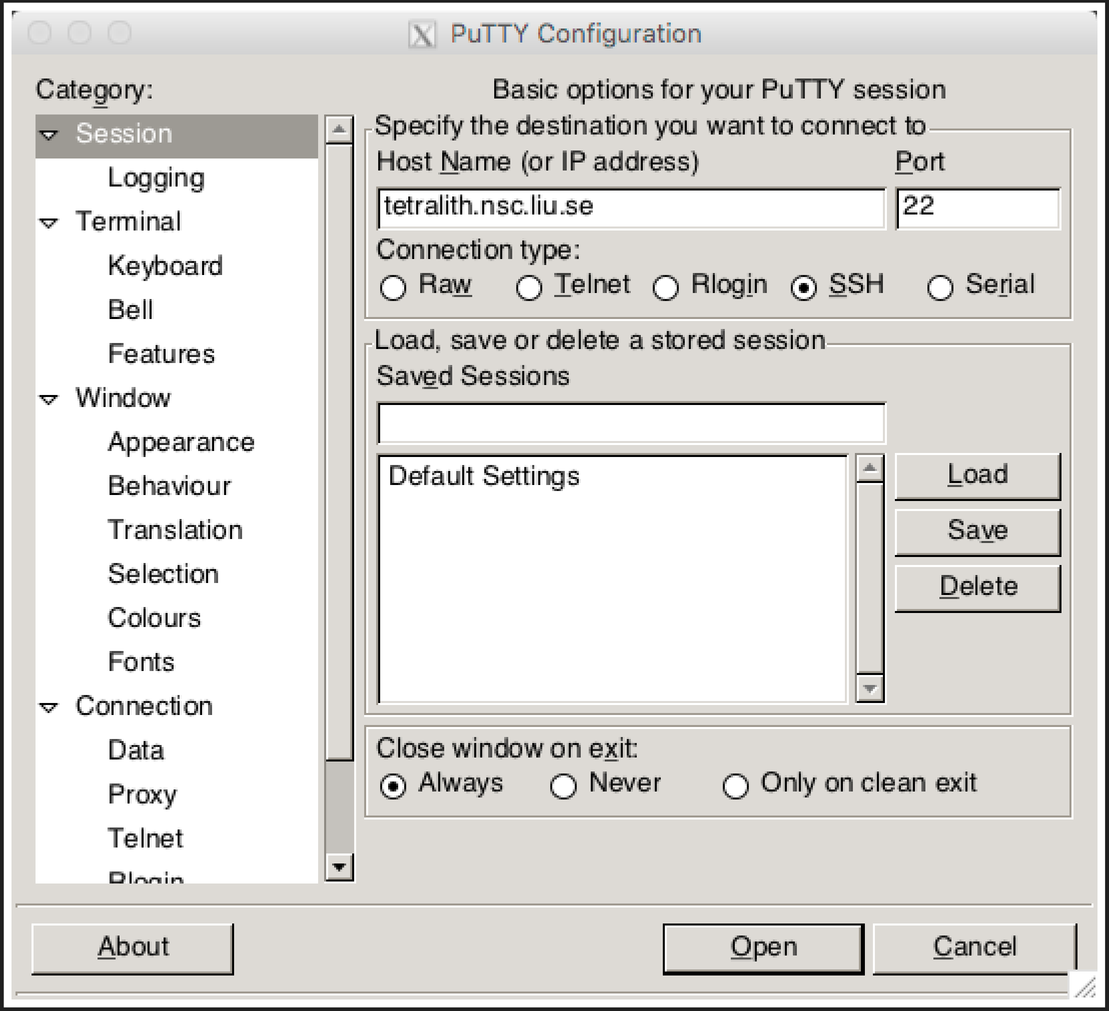
SSH - COMMAND LINE AND PUTTY

Command line ssh:

```
mylaptop$ ssh x_abcde@tetralith.nsc.liu.se
password: *************
```

Or with PuTTY:

Write the host name in the Sessions window


PuTTY Configuration

Category:
- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
  - Fonts
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): tetralith.nsc.liu.se

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions

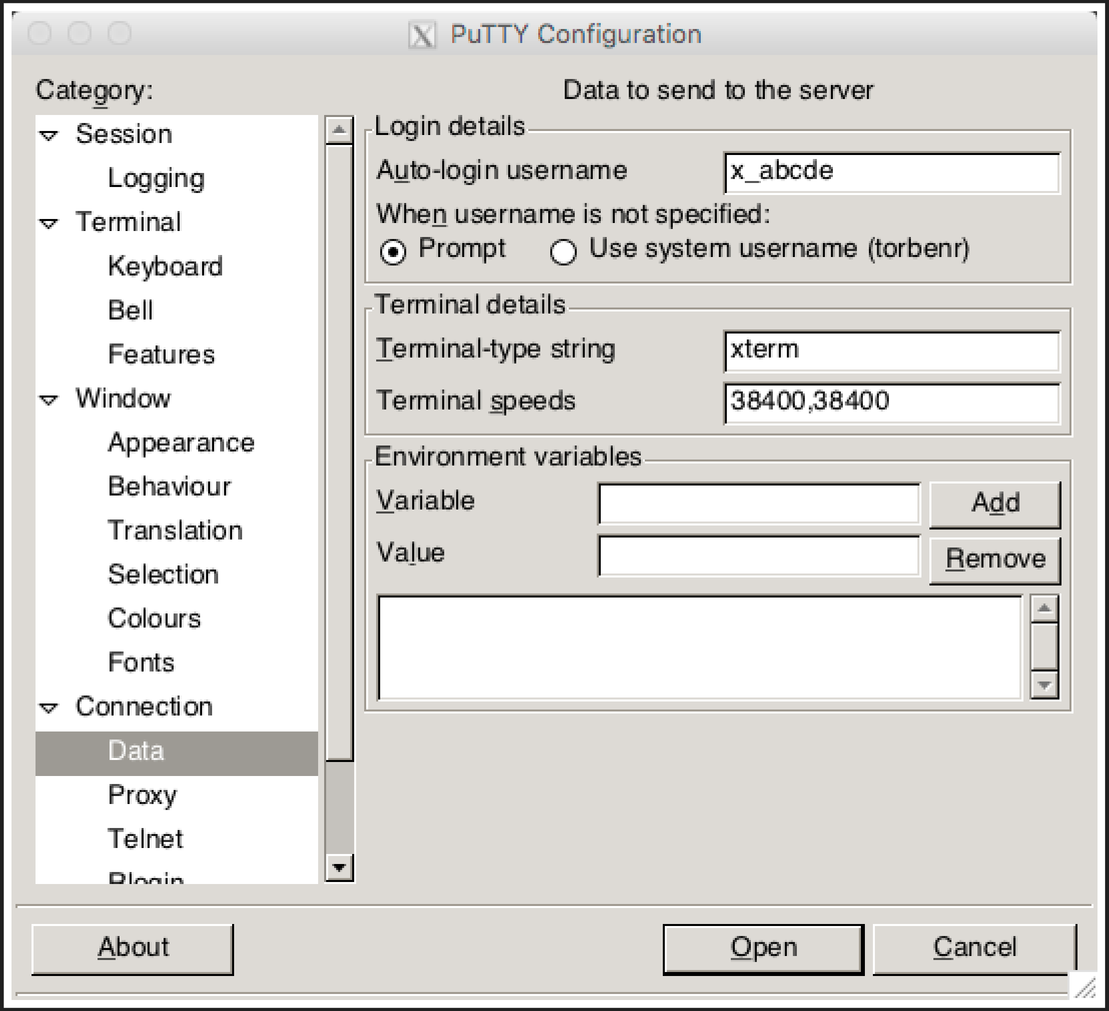
Default Settings

Load | Save | Delete

Close window on exit: [✓ Always] Never Only on clean exit

About | Open | Cancel

… and the user name in the Data window


PuTTY Configuration

Category:

- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
  - Fonts
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin

Data to send to the server

Login details

Auto-login username: x_abcde

When username is not specified:

[✓ Prompt] Use system username (torbenr)

Terminal details

Terminal-type string: xterm

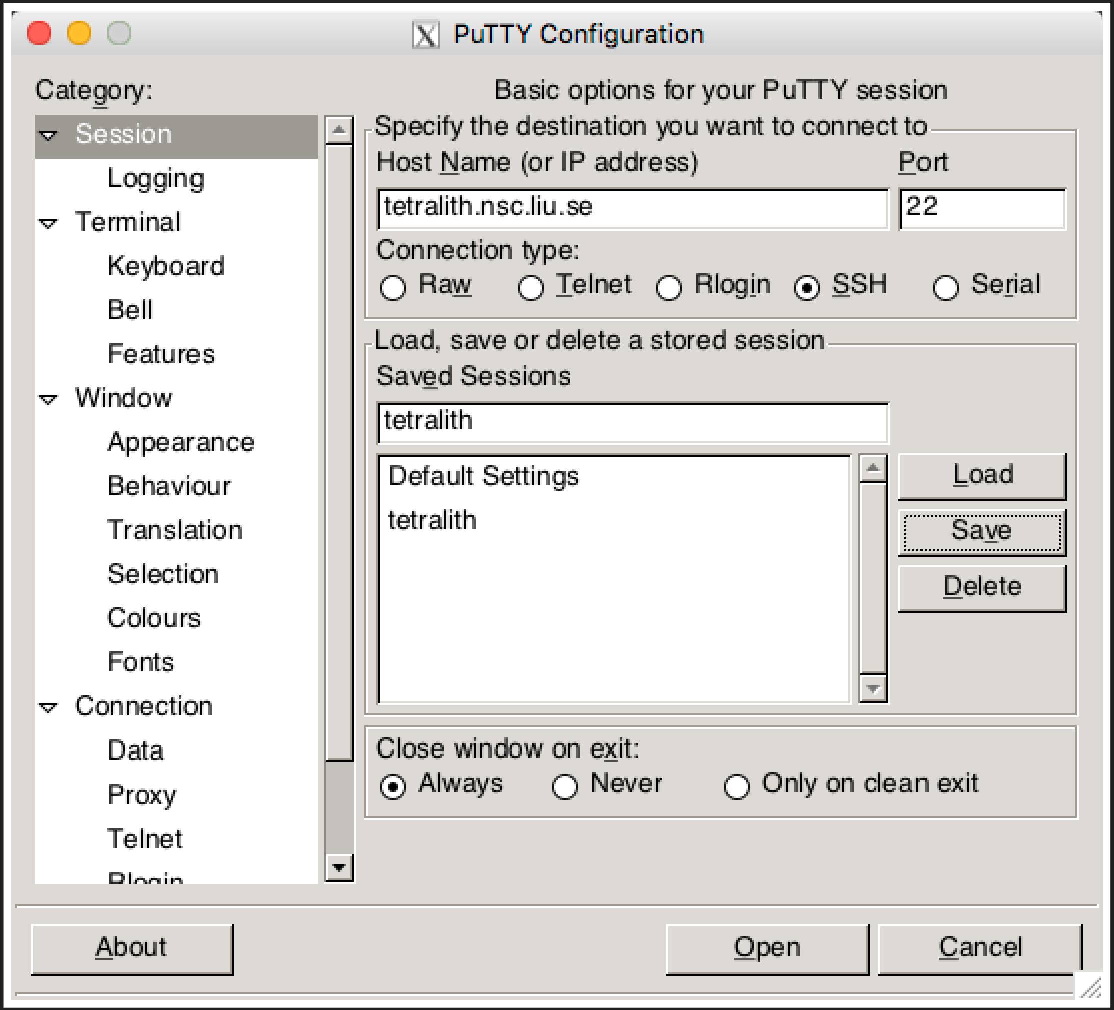
Terminal speeds: 38400,38400

Environment variables

Variable

Value

Give the session a name and save it for easy reuse:


PuTTY Configuration

Category:
- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
  - Fonts
- Connection
  - Data
  - Proxy
  - Telnet

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): tetralith.nsc.liu.se

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions: tetralith

Default Settings
tetralith

Close window on exit: [✓ Always] Never Only on clean exit

MAKING LIFE EASIER WITH PUBLIC KEYS

Logging in and copying files can be made quite convenient with ssh keys and an ssh agent.

Let's generate a key-pair.

MAC AND LINUX VERSION

On Mac and Linux, you open a terminal and use the OpenSSH command line tools:

Obs. you do this on your own local computer!

```
mylaptop$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/torbenr/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/torbenr/.ssh/id_rsa.
Your public key has been saved in /Users/torbenr/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:8ekuAHDIjyX4P7uehUkMahKuXFKnvOFCleCHI/pBT58 torbenr@gaia.nsc.liu.se
The key's randomart image is:
+---[RSA 2048]----+
| + .             |
|o B +            |
|o=o%..   .       |
|++O==. . o .     |
|+=.*+.E S o      |
|*.+o=o.   .      |
|.o.ooo..   .     |
|  . .o   ..      |
|   .+.    ..     |
+----[SHA256]-----+
```

NEVER USE AN EMPTY PASSPHRASE!

> *Use some system for handling your passwords!*

- Private paper notebook
- Encrypted file(s)
- OS build-in tool (*e.g.* Keychain Access on Mac)
- Password manager applications (*e.g.* KeePass)
- Cloud services (*e.g.* LastPass)

Copy the **public** key to Tetralith. On some systems you can use `ssh-copy-id`. However, you can also just copy the `id_rsa.pub` file to Tetralith and manually add it to your `authorized_keys` file in your `.ssh/` directory:

```
mylaptop$ scp /Users/torbenr/.ssh/id_rsa.pub x_abcde@tetralith.nsc.liu.se:.ssh/
password: *************
id_rsa.pub                                    100%   405     0.4KB/s   00:00
mylaptop$ ssh x_abcde@tetralith.nsc.liu.se
password: *************
$ cd .ssh/
$ cat id_rsa.pub >> authorized_keys
$ rm id_rsa.pub
$ ls -l
total 129
-rw------- 1 x_abcde x_abcde   811 Nov 23 14:44 authorized_keys
-rw------- 1 x_abcde x_abcde 20480 Sep 21 18:10 known_hosts
```

- Observe the file permissions for `authorized_keys`.
- Should be readable and writable only for user (*i.e.* not group and other)!
- Change the permissions with `chmod go= authorized_keys` if needed.

On your local Mac or Linux machine, you can now add your private key to the ssh agend with `ssh-add`.

Many systems (*e.g.* Mac and Linux Ubuntu) will automatically load SSH keys when you login, so you will not have to do anything except enter your passphrase once the first time you try to use the key.

WINDOWS (PUTTY) VERSION

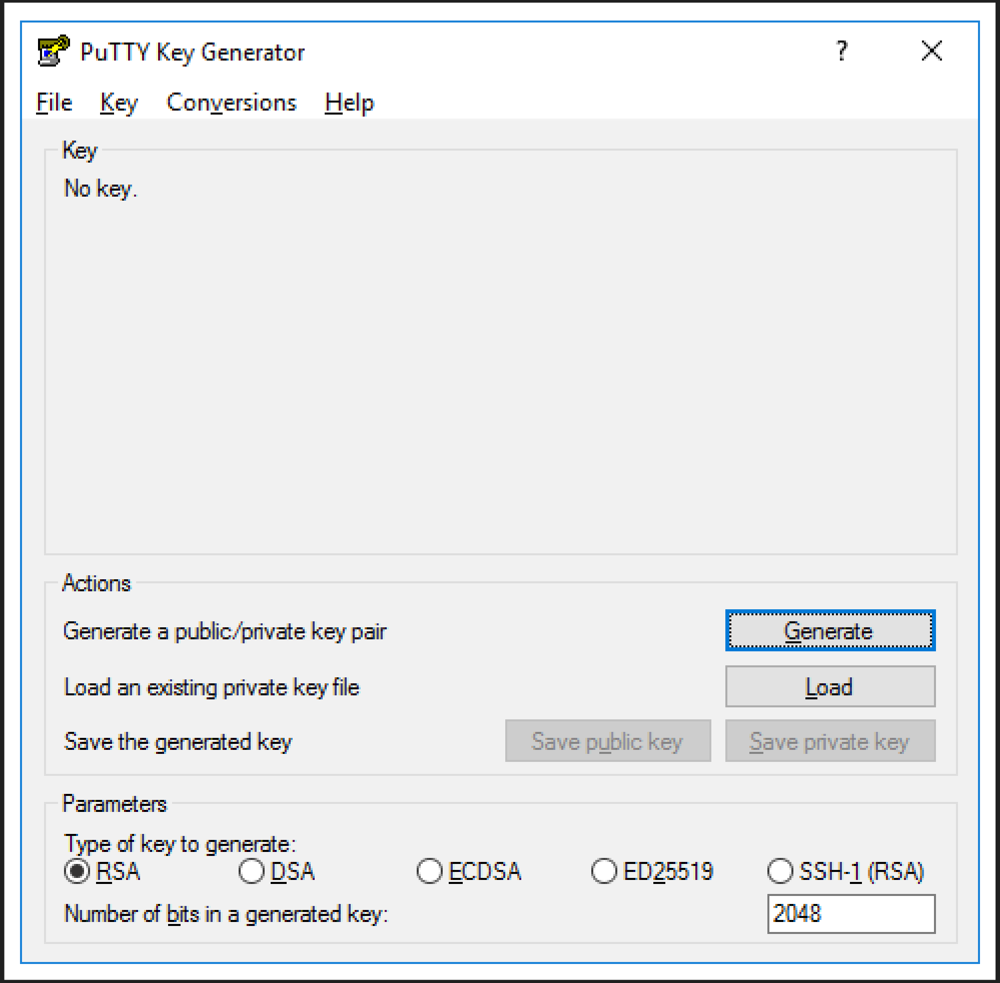
There are also ways of getting OpenSSH in Windows and later updates of Windows 10 apparently include this natively. However, we will outline the well established PuTTY version:

1. Create an ssh key-pair with PuTTYgen
2. Copy the **public** key to .ssh/authorized_keys on Tetralith
3. Set up Pageant, the ssh agent, to run at Windows startup

Once running, Pageant should work with PuTTY, WinSCP, and FileZilla!

I've added some potentially useful links regarding PuTTY and Pageant under Further information

Generating a key-pair with PuTTYgen:


PuTTY Key Generator

Key

No key.

Actions

Generate a public/private key pair

Load an existing private key file

Save the generated key

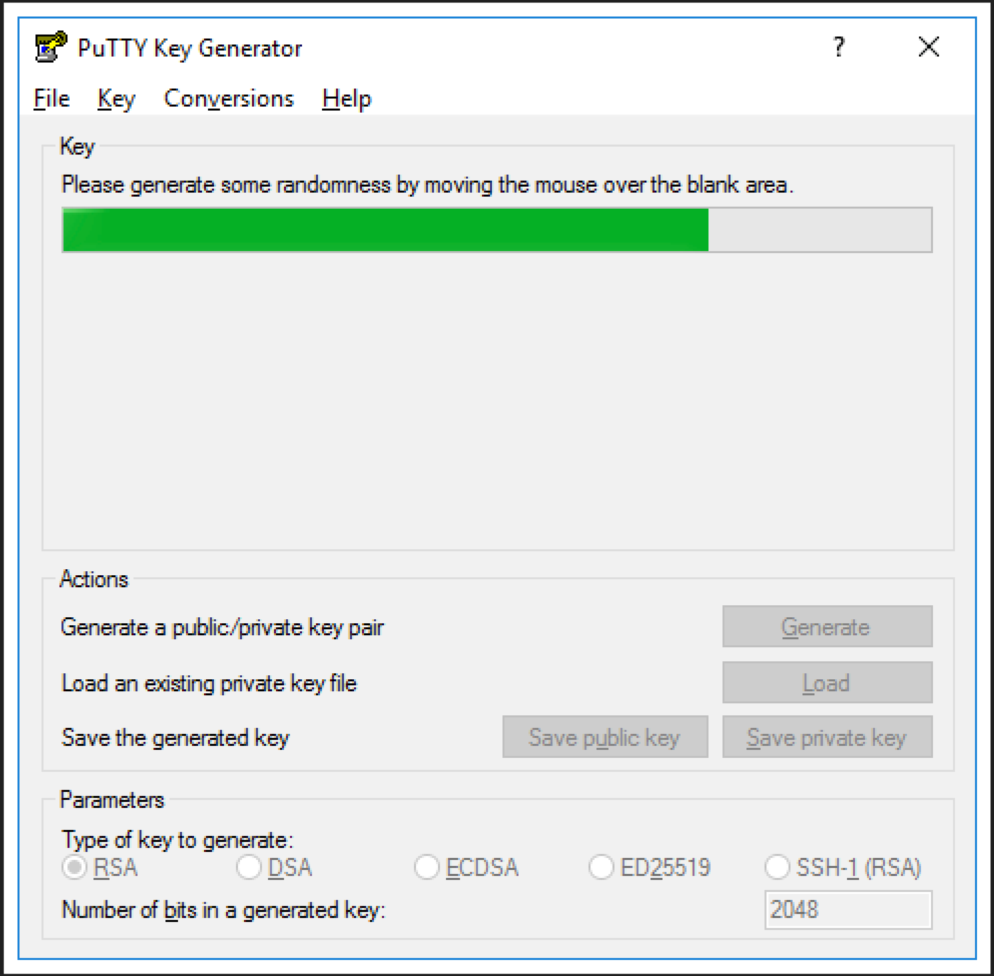
Parameters

Type of key to generate:

[✓ RSA] DSA ECDSA ED25519 SSH-1 (RSA)

Number of bits in a generated key: 2048

Move the mouse around:



Key

Please generate some randomness by moving the mouse over the blank area.

Actions

Generate a public/private key pair — Generate

Load an existing private key file — Load

Save the generated key — Save public key — Save private key

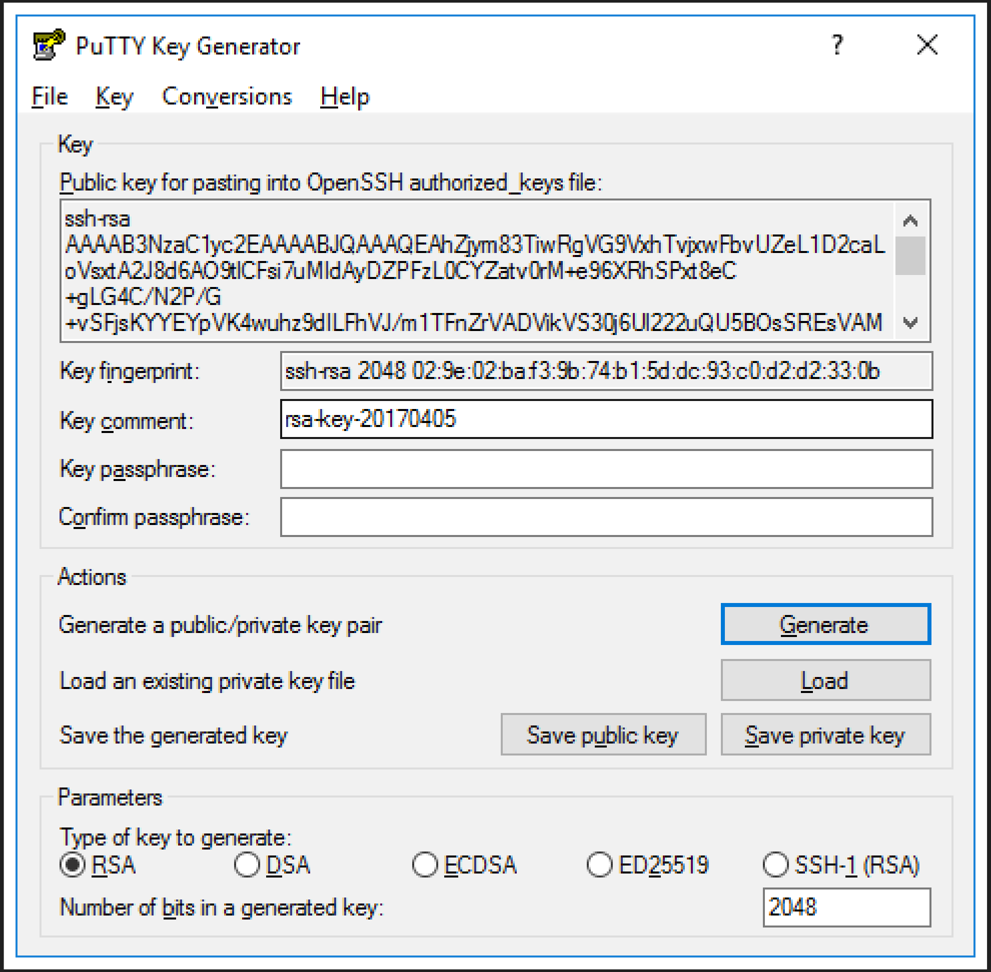
Parameters

Type of key to generate:

[✓ RSA] DSA ECDSA ED25519 SSH-1 (RSA)

Number of bits in a generated key: 2048

Enter a passphrase and save the keys:



Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEAhZjym83TiwRgVG9VxhTvjxwFbvUZeL1D2caL
oVsxtA2J8d6AO9tlCFsi7uMIdAyDZPFzL0CYZatv0rM+e96XRhSPxt8eC
+gLG4C/N2P/G
+vSFjsKYYEYpVK4wuhz9dILFhVJ/m1TFnZrVADVikVS30j6UI222uQU5BOsSREsVAM
```

Key fingerprint: ssh-rsa 2048 02:9e:02:ba:f3:9b:74:b1:5d:dc:93:c0:d2:d2:33:0b

Key comment: rsa-key-20170405

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair

Load an existing private key file

Save the generated key

Parameters

Type of key to generate:

[✓ RSA] DSA ECDSA ED25519 SSH-1 (RSA)

Number of bits in a generated key: 2048

MAKING LIFE *EVEN* EASIER WITH A CONFIG FILE

One of the nice things with PuTTY is that you can save settings of a session for easy reuse.

We can achieve something very similar for the OpenSSH command line tools using an ssh config file:

```
mylaptop$ cat .ssh/config
# Configuration for Tetralith at NSC
Host tetralith
  HostName tetralith.nsc.liu.se
  User x_abcde
```

With this config file, the command line is as simple as:

```
mylaptop$ ssh tetralith
Last login: Sat Nov 24 16:18:33 2018 from 2001:6b0:17:fc08:890f:8e26:35d4:4cdd
Welcome to NSC and Tetralith!
$
```

… and copying a file is as easy as:

```
mylaptop$ scp example_file tetralith:
example_file                                       100%    0     0.0KB/s   00:00
```

If you're using macOS Sierra 10.12.* or later, you might need something like this:

```
myMaclaptop$ cat .ssh/config
# Default for all
Host *
  AddKeysToAgent yes
  UseKeychain yes
# Configuration for Tetralith at NSC
Host tetralith
  HostName tetralith.nsc.liu.se
  User x_abcde
```

… to automatically load keys into the ssh-agent and store passphrases in your keychain.

Example of a config entry for local port forwarding that might come in handy later when we will work with Jupyter notebooks:

```
mylaptop$ cat .ssh/config
# Configuration for Tetralith at NSC
Host tetralith
  HostName tetralith.nsc.liu.se
  User x_abcde
# Running Jupyter notebook on Tetralith login node
Host tetralith_jupyter
  HostName tetralith.nsc.liu.se
  User x_abcde
  LocalForward 9988 localhost:9988
```

Observe that everyone must use a unique port, so you cannot use 9988, that's mine ;-)

For example, use 9989, 9990, 9991, etc.

SOME OTHER CONVENIENCE TOOLS

When working on a remote server, like a Tetralith login node, it can be very convenient to not have to "finish" what you are doing when you have to close your laptop and run for the bus.

Tools that allow you to maintain "persistent" sessions on a login node that you can detach from and then later re-attach to:

1. screen (command line tool for multiple interactive shells)
2. tmux (command line tool for multiple interactive shells)
3. ThinLinc (remote desktop with support for server side hardware accelerated OpenGL graphics)

Obs. the server side hardware accelerated OpenGL graphics is actually provided by VirtualGL, but this works really well with ThinLinc.

SCREEN COMMANDS SHORT LIST

command	explanation
`screen`	start a session
`screen -S <name>`	start a named session
`screen -ls`	list running sessions
`screen -x`	attach to a running session
`screen -r <name>`	attach to session with `<name>`

The screen commands inside a screen session are prefixed by an escape key, by default `C-a` (that's Ctrl + a)

screen command	explanation
`C-a d`	detach from session
`exit` (`C-d`)	kill the window/session
`C-a c`	create new window
`C-a C-a`	change to last-visited window
`C-a <number>`	change to window by number
`C-a "`	see window list (and select)

TMUX COMMANDS SHORT LIST

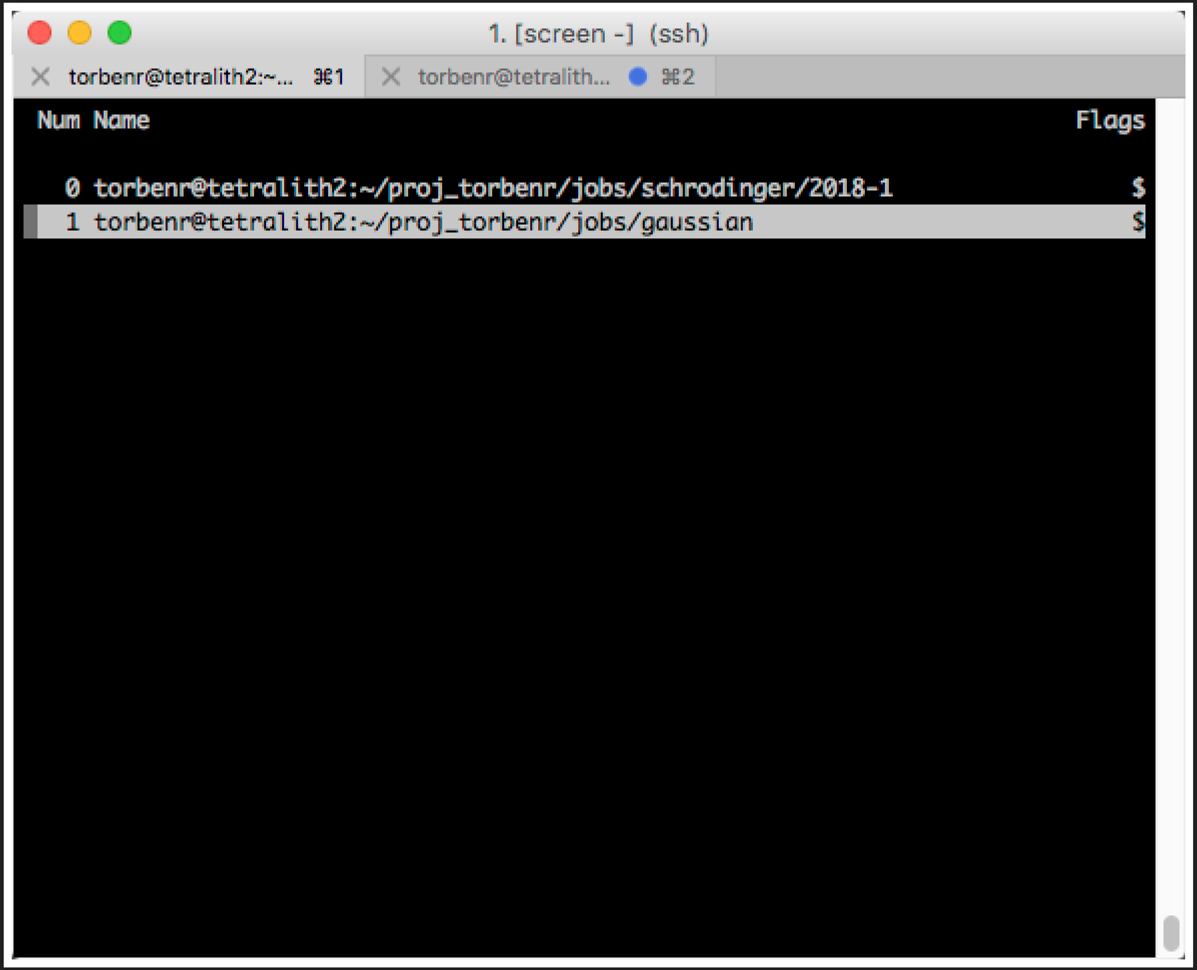
command	explanation
`tmux`	start a session
`tmux new -s <name>`	start a named session
`tmux ls`	list running sessions
`tmux a`	attach to a running session
`tmux a -t <name>`	attach to session with `<name>`

The tmux commands inside a tmux session are prefixed by an escape key, by default `C-b` (that's Ctrl + b)

tmux command	explanation
`C-b d`	detach from session
`exit` (`C-d`)	kill the window/session
`C-b c`	create new window
`C-b l`	change to last-visited window
`C-b <number>`	change to window by number
`C-b w`	see window list (and select)

SCREEN EXAMPLE

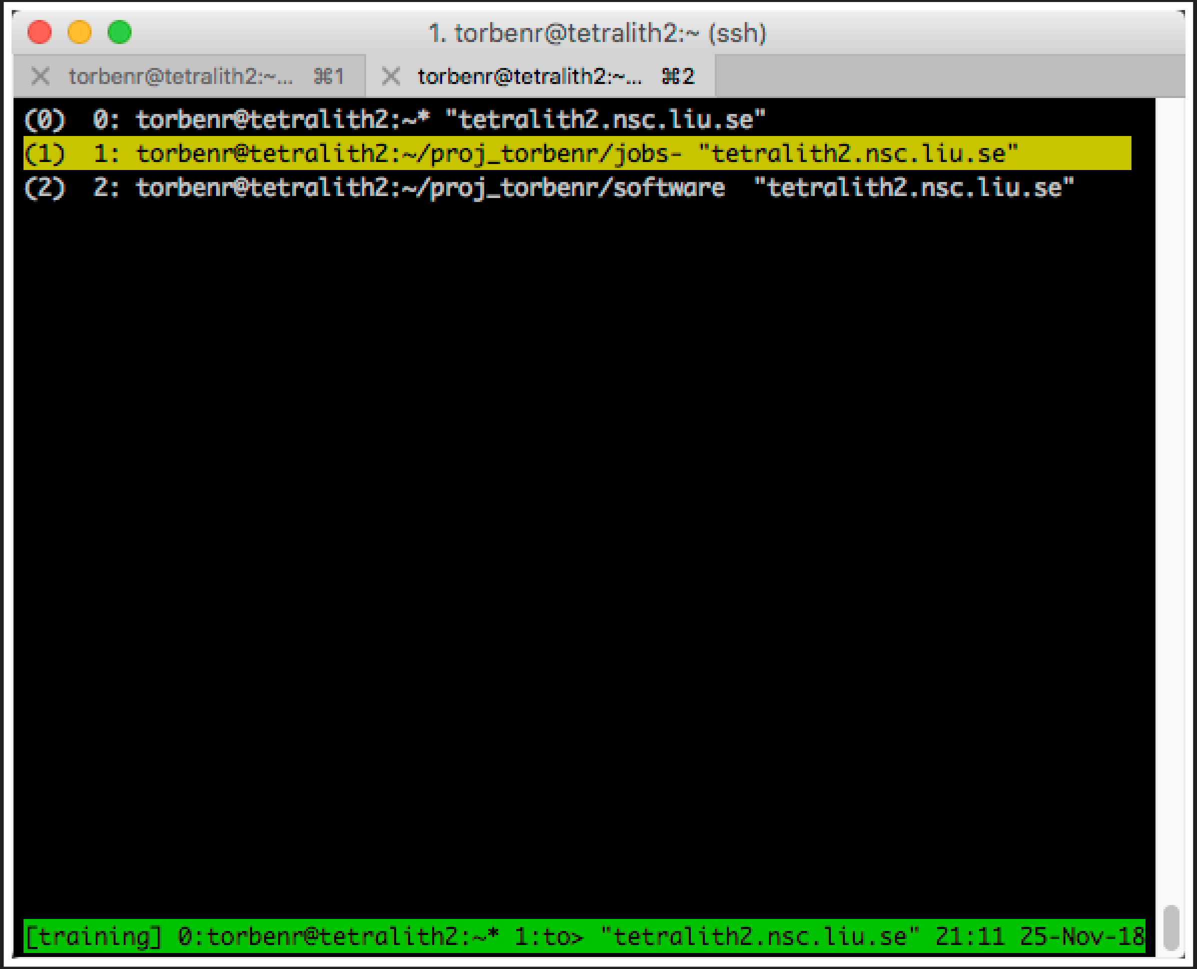
```
mylaptop$ ssh tetralith
Welcome to NSC and Tetralith!

tetralith1$ screen -S training
tetralith1$ cd <some proj dir>
tetralith1$ C-a c
tetralith1$ cd <some other proj dir>
tetralith1$ C-a "
```



```
Num Name                                                          Flags

  0 torbenr@tetralith2:~/proj_torbenr/jobs/schrodinger/2018-1         $
  1 torbenr@tetralith2:~/proj_torbenr/jobs/gaussian                   $
```

TMUX EXAMPLE

```
mylaptop$ ssh tetralith
Welcome to NSC and Tetralith!

tetralith2$ tmux new -s training
tetralith2$ C-b c
tetralith2$ cd <some proj dir>
tetralith2$ C-b c
tetralith2$ cd <some other proj dir>
tetralith2$ C-b w
```


1. torbenr@tetralith2:~ (ssh)

```
(0)  0: torbenr@tetralith2:~* "tetralith2.nsc.liu.se"
(1)  1: torbenr@tetralith2:~/proj_torbenr/jobs- "tetralith2.nsc.liu.se"
(2)  2: torbenr@tetralith2:~/proj_torbenr/software  "tetralith2.nsc.liu.se"

[training] 0:torbenr@tetralith2:~* 1:to> "tetralith2.nsc.liu.se" 21:11 25-Nov-18
```

RE-ATTACHING TO A SCREEN OR TMUX SESSION

To re-attach to an existing screen or tmux session, I (obviously) need to login to the login node where the session is running.

Hence …

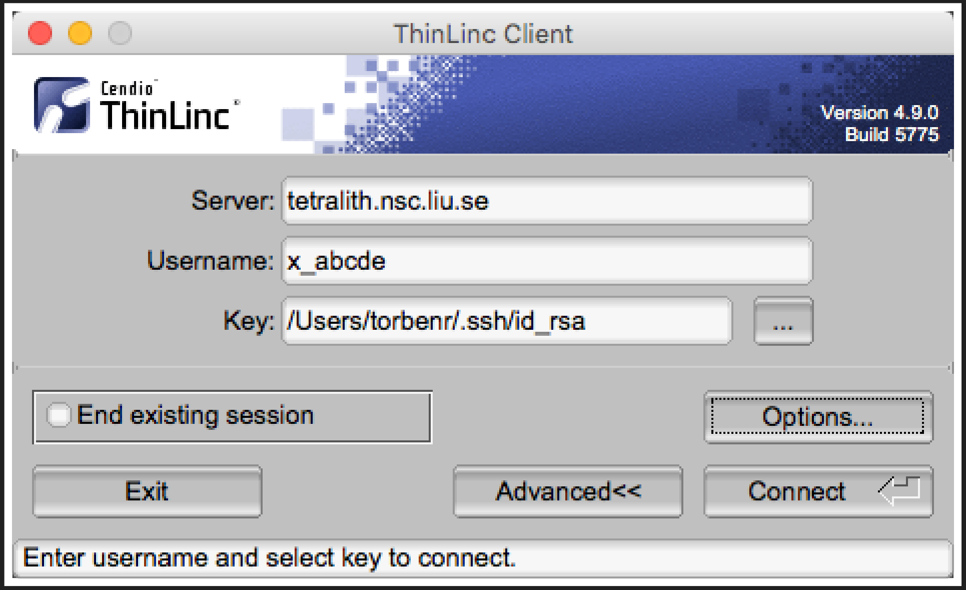
Re-attaching to my screen session

```
mylaptop$ ssh tetralith1
Welcome to NSC and Tetralith!

tetralith1$ screen -r training
```

and re-attaching to my tmux session

```
mylaptop$ ssh tetralith2
Welcome to NSC and Tetralith!

tetralith2$ tmux a -t training
```

THINLINC



Server: tetralith.nsc.liu.se

Username: x_abcde

Key: /Users/torbenr/.ssh/id_rsa

Enter username and select key to connect.

Remote desktop with support for server side hardware accelerated OpenGL graphics!

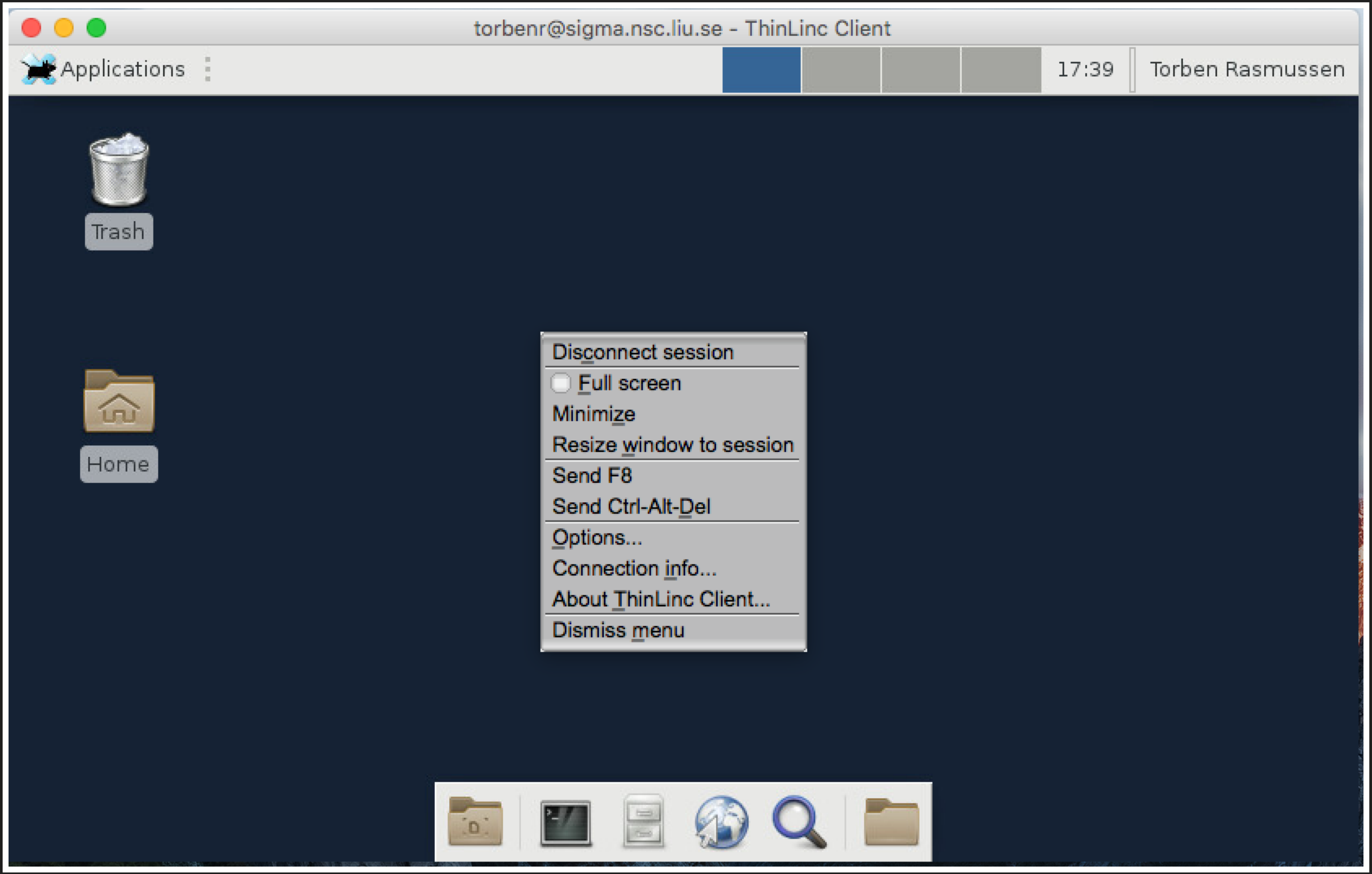
When should I use ThinLinc?

Whenever you want to run a graphical user interface (GUI).

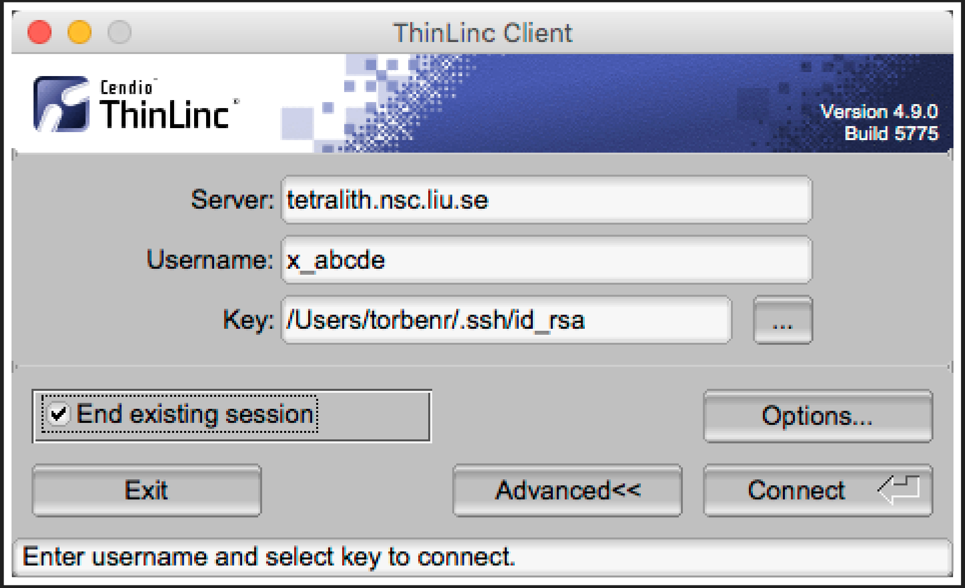
When should I not use ThinLinc?

When you are on a slow (wireless) internet connection.

Some settings can be changed in an active session.



ThinLinc automatically re-attaches to your existing session (if you have one) unless you tell it to end the existing session (tick box) and create a new one.



ThinLinc Client

Version 4.9.0
Build 5775

Server: tetralith.nsc.liu.se

Username: x_abcde

Key: /Users/torbenr/.ssh/id_rsa

[✓ End existing session]

Enter username and select key to connect.

We "garbage"-collect inactive ThinLinc sessions, but please make it a habit to logout if you don't need to re-attact to your session.

Attention Mac users!

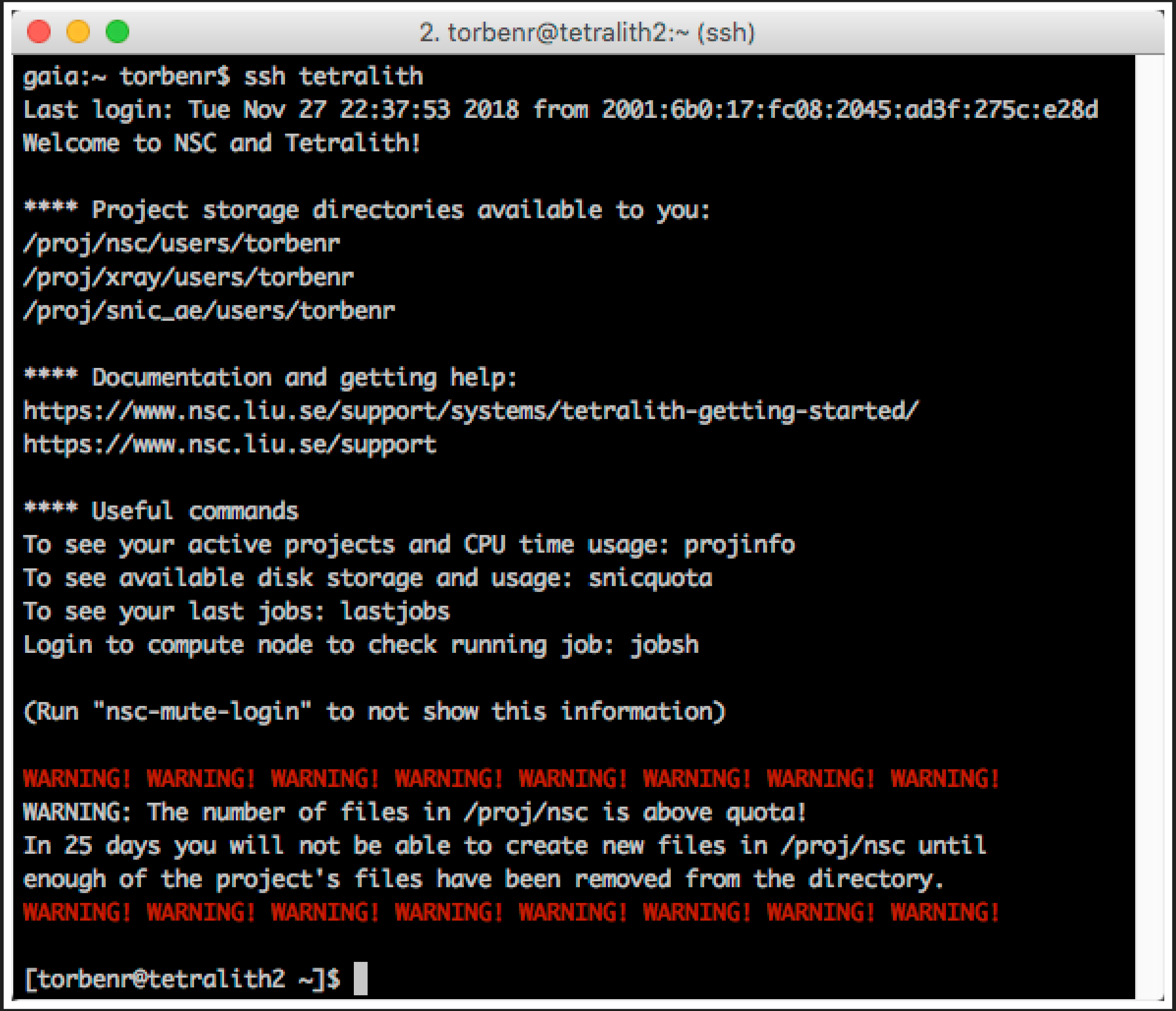
To setup a ThinLinc session to both Sigma and Tetralith at the same time, you need to start *two* client windows before starting the sessions:

1. Start ThinLinc client
2. Press cmd-N (File>)
3. Login to Tetralith in one window and to Sigma in the other

Several of you are using ThinLinc already. Is there anything you wonder about regarding ThinLinc settings and using ThinLinc?

WORKING WITH FILES AND DIRECTORIES

First, we always read the login message:



2. torbenr@tetralith2:~ (ssh)

```
gaia:~ torbenr$ ssh tetralith
Last login: Tue Nov 27 22:37:53 2018 from 2001:6b0:17:fc08:2045:ad3f:275c:e28d
Welcome to NSC and Tetralith!

**** Project storage directories available to you:
/proj/nsc/users/torbenr
/proj/xray/users/torbenr
/proj/snic_ae/users/torbenr

**** Documentation and getting help:
https://www.nsc.liu.se/support/systems/tetralith-getting-started/
https://www.nsc.liu.se/support

**** Useful commands
To see your active projects and CPU time usage: projinfo
To see available disk storage and usage: snicquota
To see your last jobs: lastjobs
Login to compute node to check running job: jobsh

(Run "nsc-mute-login" to not show this information)

WARNING! WARNING! WARNING! WARNING! WARNING! WARNING! WARNING! WARNING!
WARNING: The number of files in /proj/nsc is above quota!
In 25 days you will not be able to create new files in /proj/nsc until
enough of the project's files have been removed from the directory.
WARNING! WARNING! WARNING! WARNING! WARNING! WARNING! WARNING! WARNING!

[torbenr@tetralith2 ~]$
```

RECOMMENDATIONS

- Setup symbolic links to your directories under `/proj` in your home directory

```
$ cd $HOME
$ ln -s /proj/<project name>/users/x_abcde my_proj_dir
```

- Run your jobs and store your data under `/proj`
- Run snicquota regularly
- Setup folders under `/proj/<project name>/` for sharing data within the project

```
/proj/<project name>/users/...              - personal areas
/proj/<project name>/datasets/YYYY/MM/DD    - shared datasets
/proj/<project name>/scripts                - useful scripts
/proj/<project name>/pkg/someapp-X.Y        - applications
[...]
```

- Write job-temporary files to `/scratch/local`
- Read through the NSC pages on storage
- If you mess-up, then have a look in `/proj/.snapshots` (or `/home/.snapshots`)

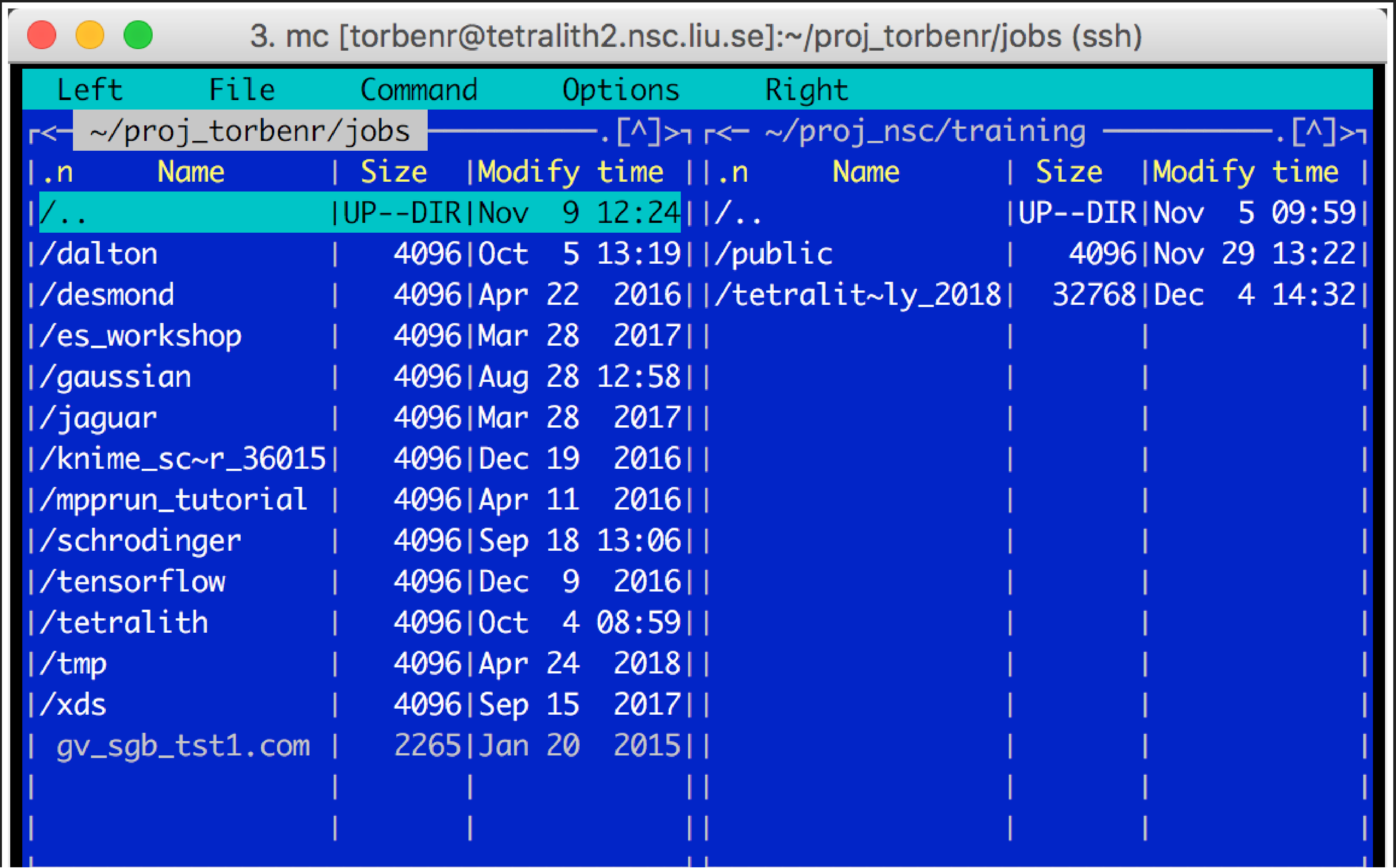
COMMAND (VERY) SHORT LIST

command	explanation
`cd <dir name>`	change to directory
`mkdir <dir name>`	make directory
`chmod MODE <file or dir name>`	change permissions for file or directory
`ls`	list directory contents
`mv <file or dir name> <new file or dir name>`	change file or directory name
`rm <file or dir name>`	remove file or directory

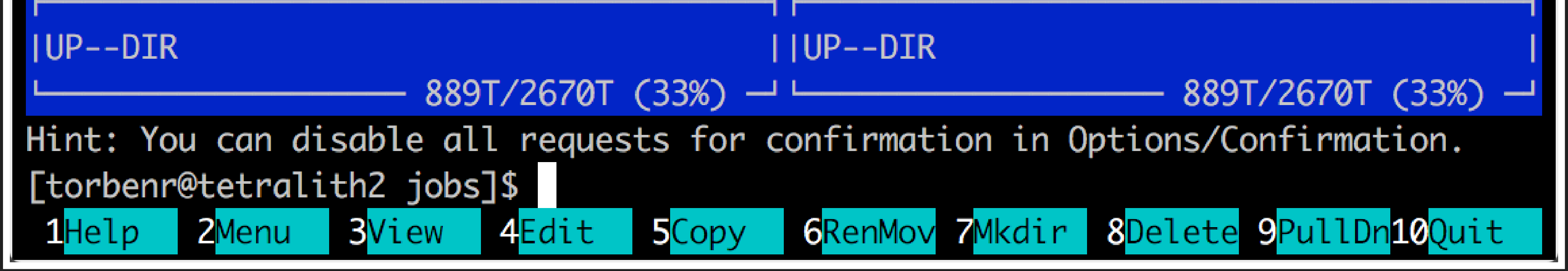
Use tab-completion!

You may also like Midnight Commander:

```
$ module load mc/4.8.21
$ mc
```

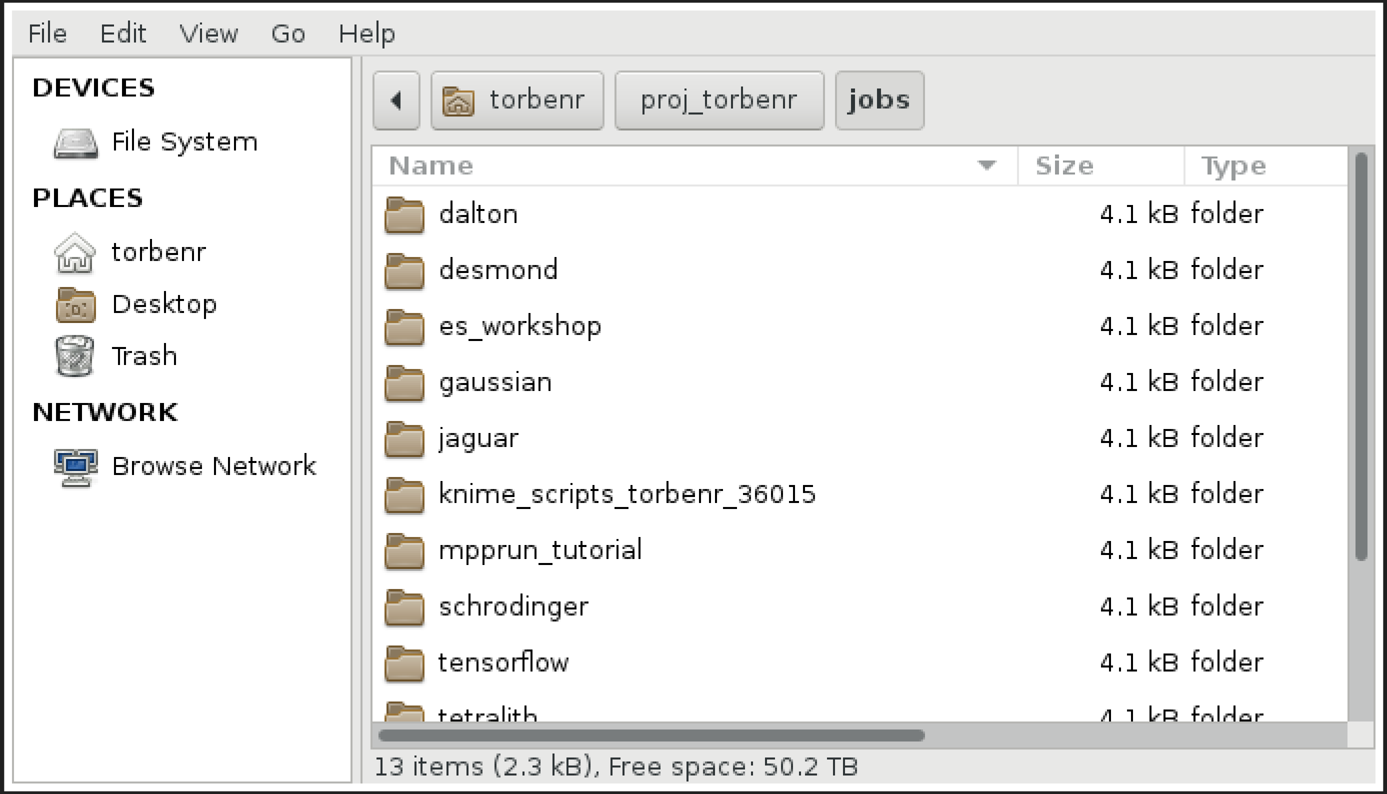

3. mc [torbenr@tetralith2.nsc.liu.se]:~/proj_torbenr/jobs (ssh)
Left
File
Command
Options
Right

```
┌<─ ~/proj_torbenr/jobs ───────────.[^]>┐┌<─ ~/proj_nsc/training ──────────.[^]>┐
|.n      Name        | Size  |Modify time ||.n      Name        | Size  |Modify time |
|/..                 |UP--DIR|Nov  9 12:24||/..                 |UP--DIR|Nov  5 09:59|
|/dalton             |   4096|Oct  5 13:19||/public             |   4096|Nov 29 13:22|
|/desmond            |   4096|Apr 22  2016||/tetralit~ly_2018   |  32768|Dec  4 14:32|
|/es_workshop        |   4096|Mar 28  2017||                    |       |            |
|/gaussian           |   4096|Aug 28 12:58||                    |       |            |
|/jaguar             |   4096|Mar 28  2017||                    |       |            |
|/knime_sc~r_36015   |   4096|Dec 19  2016||                    |       |            |
|/mpprun_tutorial    |   4096|Apr 11  2016||                    |       |            |
|/schrodinger        |   4096|Sep 18 13:06||                    |       |            |
|/tensorflow         |   4096|Dec  9  2016||                    |       |            |
|/tetralith          |   4096|Oct  4 08:59||                    |       |            |
|/tmp                |   4096|Apr 24  2018||                    |       |            |
|/xds                |   4096|Sep 15  2017||                    |       |            |
| gv_sgb_tst1.com    |   2265|Jan 20  2015||                    |       |            |
|                    |       |            ||                    |       |            |
|                    |       |            ||                    |       |            |
```



```
|UP--DIR                                          ||UP--DIR                                          |
└────────────────────────── 889T/2670T (33%) ─┘└────────────────────────── 889T/2670T (33%) ─┘
Hint: You can disable all requests for confirmation in Options/Confirmation.
[torbenr@tetralith2 jobs]$ 
 1Help   2Menu   3View   4Edit   5Copy   6RenMov 7Mkdir  8Delete 9PullDn10Quit
```

In ThinLinc, you can obviously also use the desktop provided file manager!



Name	Size	Type
dalton	4.1 kB	folder
desmond	4.1 kB	folder
es_workshop	4.1 kB	folder
gaussian	4.1 kB	folder
jaguar	4.1 kB	folder
knime_scripts_torbenr_36015	4.1 kB	folder
mpprun_tutorial	4.1 kB	folder
schrodinger	4.1 kB	folder
tensorflow	4.1 kB	folder
tetralith	4.1 kB	folder

13 items (2.3 kB), Free space: 50.2 TB

SOME COMMAND LINE SHORTCUTS

command	explanation
`C-c`	Interrupt (kill) the current foreground process
`C-d`	Close the shell
`C-l`	Clear the screen in the terminal
`C-a`	Go to the beginning of the line
`C-e`	Go to the end of the line
`Left Arrow`	Go left one character
`Right Arrow`	Go right one character
`C-k`	Cut the part of the line after the cursor
`C-u`	Cut the part of the line before the cursor
`C-_`	Undo your last key press
`Up Arrow`	Go to the previous command in the command history
`Down Arrow`	Go to the next command in the command history
`C-r`	Search command history matching the characters you provide
`C-j`	Stop search and allow to edit found command

If you are a `vi` person, you can tell bash to use `vi`-style key bindings

```
$ set -o vi
```

Mostly makes a difference when you are editing a command line

HOW TO MANAGE YOUR ENVIRONMENT (MODULE SYSTEM)

- Find installed applications

```
$ module avail

---------------------------- /software/sse/modules ----------------------
ABINIT/recommendation                                (D)
ABINIT/8.8.2-nsc1-intel-2018a-eb
allinea-DDT/recommendation                           (D)
allinea-DDT/18.2.1
[...]
```

- Search for a specific application

```
$ module avail gaussian

---------------------------- /software/sse/modules ----------------------
Gaussian/recommendation          (D)    Gaussian/16.B.01-avx2-nsc1-bdist
Gaussian/09.E.01-avx-nsc1-bdist

Where:
D:  Default Module
```

- List loaded modules

```
$ module list
Currently Loaded Modules:
1) Gaussian/16.B.01-avx2-nsc1-bdist   2) mpprun/4.0   3) nsc/.1.0 (H,S)   4) m

Where:
 S:  Module is Sticky, requires --force to unload or purge
 H:             Hidden Module
```

- Setup default modules

```
$ module load Gaussian/16.B.01-avx2-nsc1-bdist
$ module save
Saved current collection of modules to: "default"
```

FURTHER INFORMATION

Some (potentially) useful links:

- NSC page on security
- PuTTY User Manual
- YouTube video: Using PuTTYgen and Pageant on Windows 7
- Some blog post: Starting Pageant on Windows 10
- A tutorial: Pageant and PuTTY
- Another (older) tutorial: Pageant and PuTTY
- screen quick reference
- tmux cheat sheet

PYTHON @ NSC

Python environments on Tetralith

At login you immediately have access to the standard CentOS 7 provided `python` command.

PYTHON INSTALLATIONS FROM THE MODULE SYSTEM

If you need more Python packages or want to use Python 3, then you should have a look at the Python installations that we provide as modules:

```
$ module avail python

----------------------------------- /software/sse/modules -----------------------------------
OpenBabel/2.4.1-Python-2.7.14-nsc1-intel-2018a-eb        Python/2.7.14-nsc1-intel-2018a-eb
Python/recommendation                              (D)    Python/3.6.3-anaconda-5.0.1-nsc1
Python/2.7.14-anaconda-5.0.1-nsc1                         Python/3.6.4-nsc1-intel-2018a-eb
Python/2.7.14-nsc1-gcc-2018a-eb                           Python/3.6.4-nsc2-intel-2018a-eb
```

We have both Anaconda installations and Python installations that we build!

CHECK AVAILABLE PACKAGES IN A MODULE

If you are looking for a specific Python package, then first load a Python module.

Some guidelines to help you choose a module:

1. There are generally more packages included in the Anaconda installations
2. If there are modules that only differ in the -nsc build/installation tag, then choose the one with the highest integer (*e.g.* nsc2 rather than nsc1)

ANACONDA MODULES

To list the installed packages in an Anaconda Python installation, load the module and run `conda list`.

If you are looking for a specific package, then pipe the output from `conda list` to `grep`:

```
$ module load Python/3.6.3-anaconda-5.0.1-nsc1
$ conda list | grep -i scipy
scipy                     0.19.1           py36h9976243_3
```

More info: conda, conda list

NSC BUILD MODULES

To list the installed packages in an NSC build Python installation, load the module and run `pip list`.

If you are looking for a specific package, then pipe the output from `pip list` to `grep`:

```
$ module load Python/3.6.4-nsc2-intel-2018a-eb
$ pip list --format=legacy | grep -i scipy
scipy (1.0.0)
```

More info: pip, pip list

PERSONAL PYTHON ENVIRONMENTS

If none of the available Python modules directly provide exactly what you are looking for, then have a look at setting up personal Python environments based on either one of the Anaconda Python modules or one of the NSC build Python modules.

CONDA ENVIRONMENTS

You create a conda environment with:

```
$ module load Python/3.6.3-anaconda-5.0.1-nsc1
$ conda create --name myenv
```

More info: conda, conda create

WHEN TO USE CONDA ENVIRONMENTS

- You need a python package that is available in Anaconda, but isn't already part of our installation
- You need a different version of an already installed package
- You can (probably) also use conda environments for packages that only require a simple `pip install`
- And you can potentially even use conda environments for packages that are installed with `python setup.py install`

EXAMPLE

Jupyter with Slurm magics and ipyparallel:

```
$ cd /some/path/
$ wget https://github.com/NERSC/slurm-magic/archive/master.zip
$ unzip master.zip
$ module load Python/3.6.3-anaconda-5.0.1-nsc1
$ conda create -n jupyter python=3.6
$ source activate jupyter
(jupyter) $ conda install jupyter numpy matplotlib pandas ipyparallel
(jupyter) $ cd /some/path/slurm-magic-master
(jupyter) $ python setup.py install
```

WHEN *NOT* TO USE CONDA ENVIRONMENTS

- If you need to install a python package that requires *compiling*, then you shouldn't use a conda environment!
- In this case, you can try to use a virtual environment based on one of the NSC build Python modules and if that fails, then contact support for help.

RUNNING JUPYTER NOTEBOOKS ON TETRALITH

You can both run Jupyter notebooks on the login node and on the compute nodes.

USING THINLINC

Starting Jupyter under ThinLinc is done in a few simple steps!

RUNNING A NOTEBOOK ON A LOGIN NODE

```
tetralith$ cd dir/with/notebook
tetralith$ module load Python/3.6.3-anaconda-5.0.1-nsc1
tetralith$ jupyter-notebook
```

RUNNING A NOTEBOOK ON A COMPUTE NODE

```
tetralith$ interactive -N 1 -t 120 -A snic2018-x-yy
n1$ module load Python/3.6.3-anaconda-5.0.1-nsc1
n1$ jupyter-notebook --no-browser --ip=n1
tetralith$ firefox
```

Copy the URL from the interactive terminal on n1 and paste it into firefox (not starting Firefox first, but just pressing Ctrl and clicking on the URL may also work)

Observe: The node name `n1` is just an example name. You should use the name of the node you get allocated for the `--ip` option.

USING AN SSH TUNNEL

If you would rather use a browser on your local laptop/desktop, you can instead use an ssh tunnel.

RUNNING A NOTEBOOK ON A LOGIN NODE

```
mylaptop$ ssh -L 9988:localhost:9988 x_abcde@tetralith.nsc.liu.se
tetralith$ module load Python/3.6.3-anaconda-5.0.1-nsc1
tetralith$ jupyter-notebook --port=9988 --no-browser
```

Then copy-and-paste the URL displayed in the terminal into a browser running on "mylaptop".

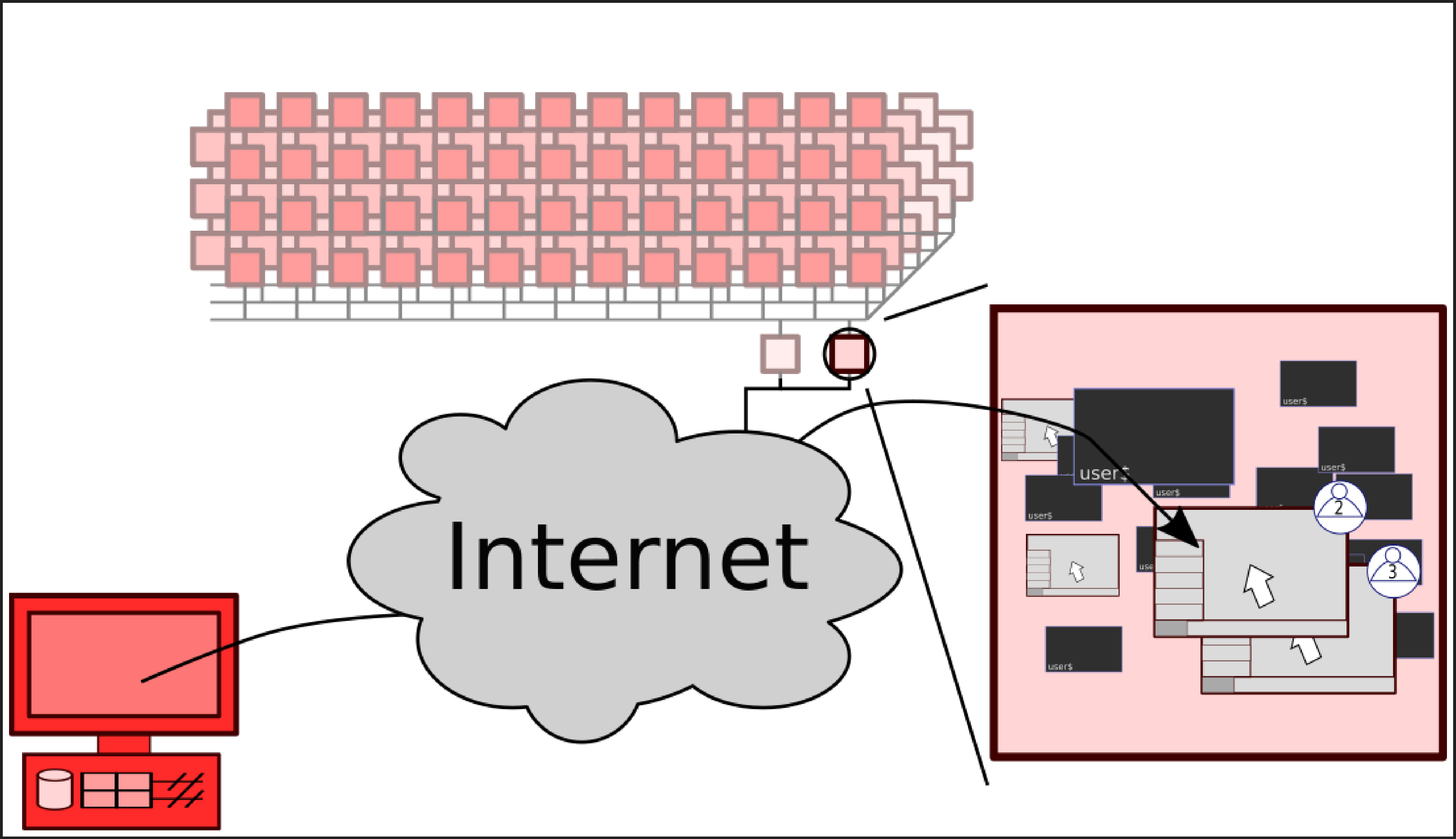
Observe: The port 9988 is somewhat arbitrary but less risk to collide with other users than the default 8888. If 9988 is already in use, then just try 9989, etc.

BUILDING SOFTWARE

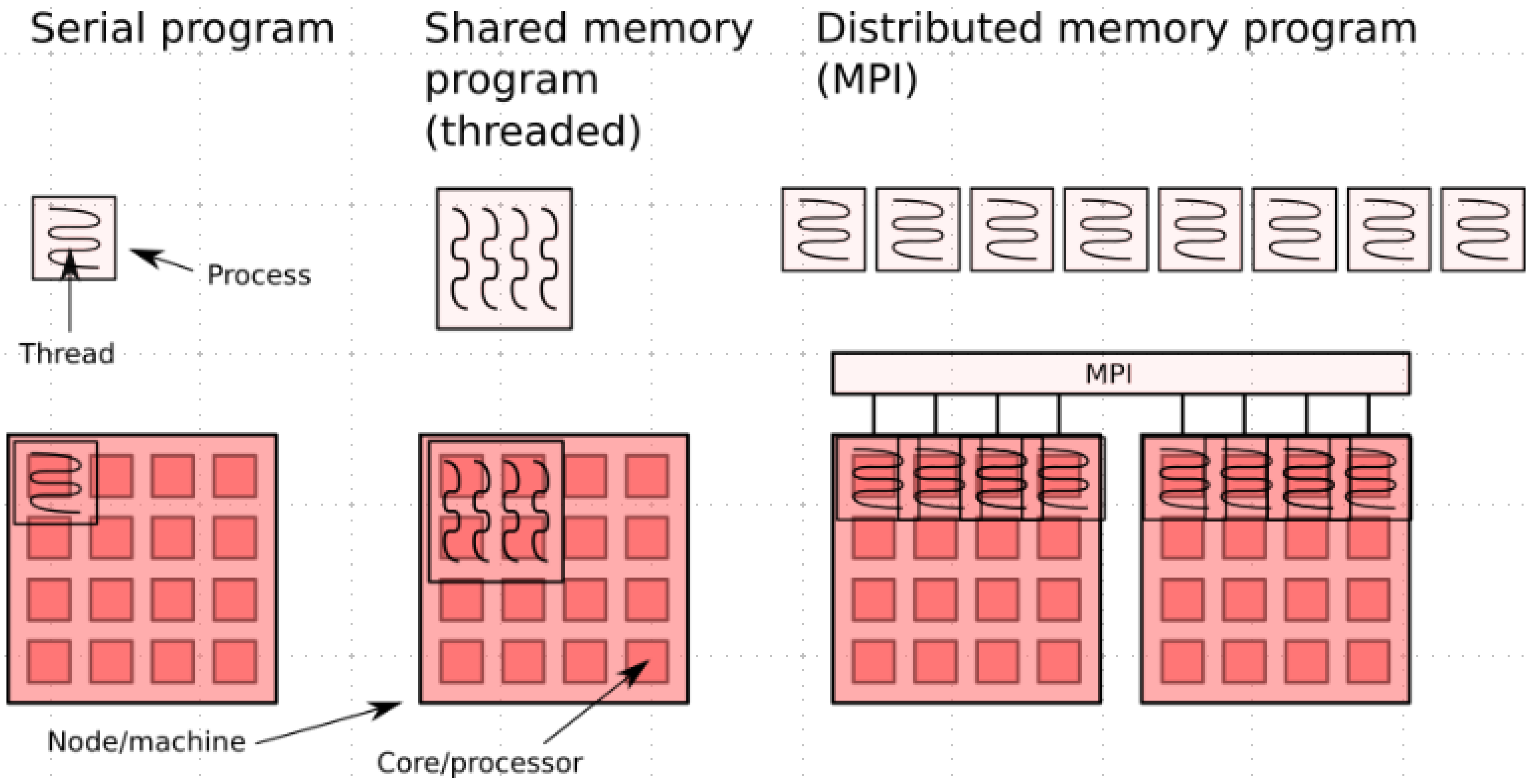
The basics of building (and installing) scientific software on your own.

Focus: Fortran/C/C++ using MPI and/or OpenMP

WHAT AND WHERE



Software: Forms of computational programs


Serial program
Shared memory program (threaded)
Distributed memory program (MPI)
Process
Thread
MPI
Node/machine
Core/processor

ISN'T THIS REALLY HARD?

It can be anything from **really simple** to seemingly impossible (or even actually **impossible...**)

FACTORS INCLUDE

- Buildsystem
- How well does it match in time and spirit?
- External dependencies
- Size and age (of the software in question)

BASIC DEPENDENCIES (COMPILER, MPI)

Provided by a special type of module, a buildenv

```
$ module avail buildenv

------------------------- /software/sse/modules --------------------------
   buildenv-gcc/recommendation                  (D)
   ...
   buildenv-intel/2018a-eb
   buildenv-intel/2018b-eb
   buildenv-intel/2018.u1-bare
```

```
$ module load buildenv-intel/2018.u1-bare
***************************************************
You have loaded an intel buildenv module
***************************************************
The buldenv-intel module makes available:
 - Compilers: icc, ifort, etc.
 - Mpi library with mpi-wrapped compilers: intel mpi with mpiicc,
mpiifort, etc.
 - Numerical libraries: intel MKL
...
$
```

EXTERNAL DEPENDENCIES

Build and install them too

Use modules revealed by loading that buildenv

MPI

MPI is a library (that implies header and library files)

MPI compiler wrappers (mpicc, mpifort, mpicxx)

Intel variants (mpiicc, mpiifort, mpiicpc)

OPENMP

OpenMP is a language extension, part of the compiler.

Enabled with compiler flag, often "-fopenmp"

BUILD SYSTEMS - FROM SIMPLE TO COMPLEX

A SINGLE SOURCE FILE

```
$ icc program.c
$ ifort -fopenmp program.c
```

A STATIC MAKEFILE

```
$ # edit makefile
$ make
```

AUTOCONF + MAKE

```
$ ./configure --help
$ ./configure --with-some-stuff --prefix=/where/to/install
$ make
```

CMAKE + MAKE

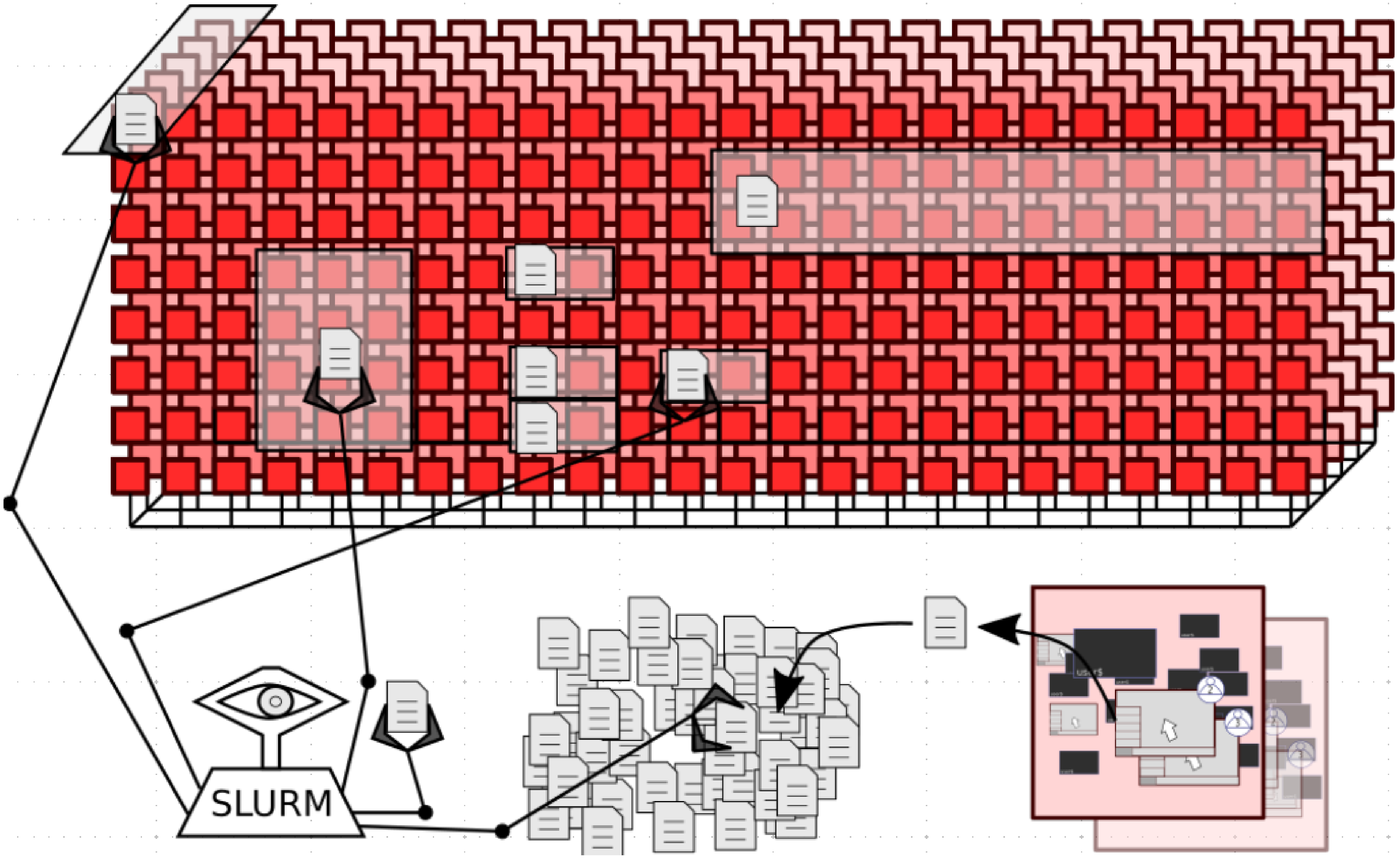
```
$ mkdir build ; cd build
$ cmake -DXXXXX -DYYYY ../
$ make
```

THE END

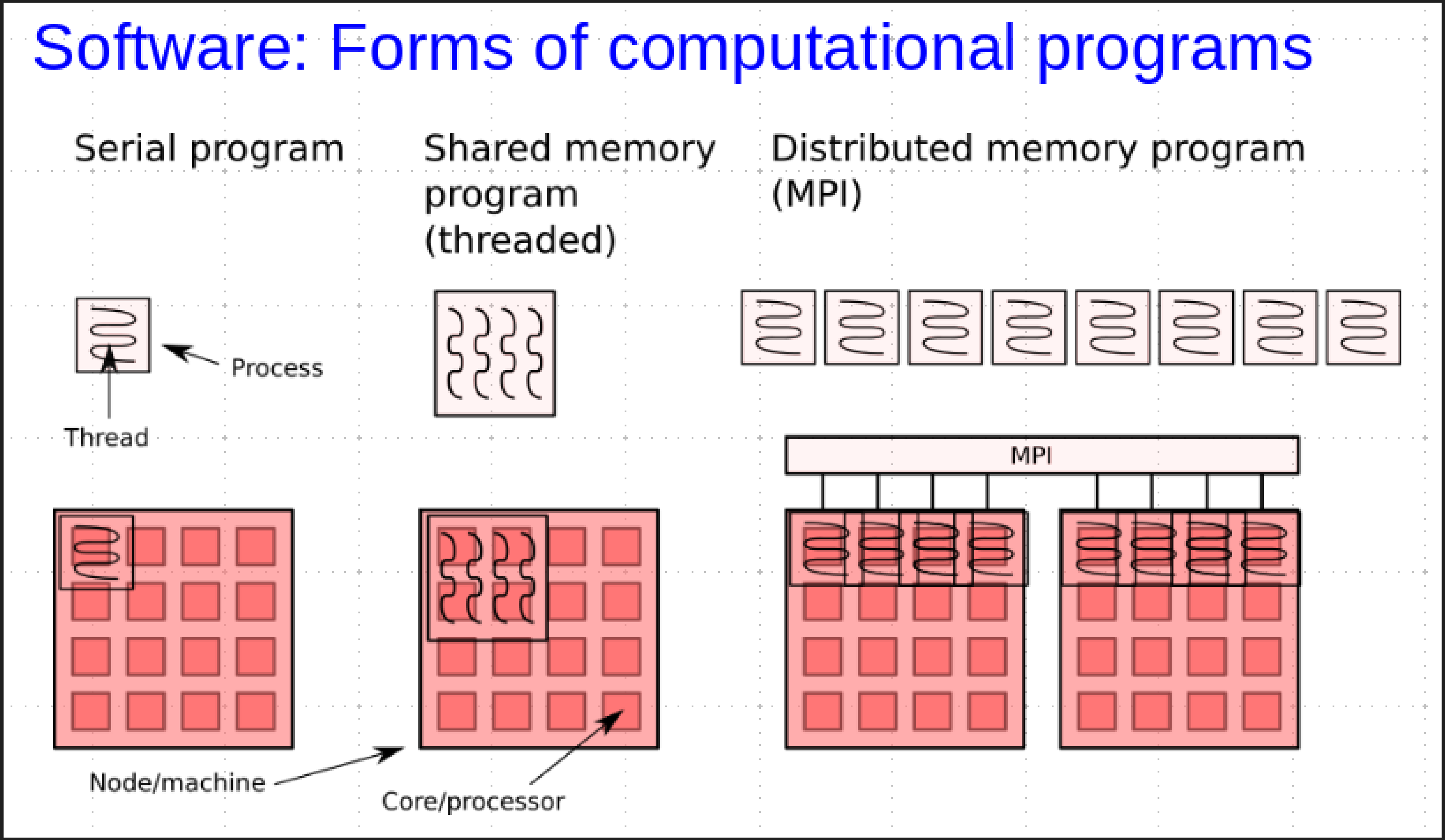
RUNNING APPLICATIONS, USING THE COMPUTE NODES

How not to run on the tiny crowded login node(s)

Queue system: Slurm



LET'S REVISIT A SLIDE...



Software: Forms of computational programs

Serial program
Shared memory program (threaded)
Distributed memory program (MPI)
Process
Thread
MPI
Node/machine
Core/processor

BASICS

1. Create jobs (sbatch, interactive). Each job will be assigned a unique jobid.
2. View existing queued or running jobs (squeue)
3. Jobs run until they hit their specified wall time limit, are cancelled by the user, crash or end normally. That is, they will *not move* nor *pause*.

Completed or failed jobs cannot be seen by squeue (use lastjobs etc.)

ACCOUNTS (PROJECT) AND RESERVATIONS

Account is the Slurm term for project. Users with membership in multiple projects will have to specify this.

Reservations are named sets of nodes with special rules. To use a reservation you have to specify this.

THREE WAYS OF SPECIFYING (IN INCREASING PRIORITY)

OPTIONS SPECIFIED IN JOB SCRIPTS

```
#!/bin/bash
#SBATCH --time=00:13:00

sleep 1h
```

OPTIONS SET IN THE ENVIRONMET

```
$ export SBATCH_TIMELIMIT=00:14:00
```

OPTIONS SPECIFIED AS COMMAND OPTIONS

```
$ sbatch --time=00:14:00
```

WHAT NEEDS TO BE SPECIFIED?

1. What (how many tasks, nodes, threads, ...)
2. For how long (time limit)

SUBMITTING A JOB THE TRADITIONAL WAY, SBATCH

1. make script
2. submit script
3. job runs
4. monitory results (output files)

```
$ export SBATCH_RESERVATION=introday1
$ sbatch --time=00:14:00 -n 128 job.sh
```

RUNNING INTERACTIVELY

Great for testing and prototyping.

Allocates the resources requested until the interactive shell is terminated.

Often used with the devel reservation.

```
$ interactive -n1 -t30 --reservation=devel
```

MAKING JOB DIRECTORIES AND KEEPING ORDER

Put job directories in the right place. Consider naming and leaving information to the future you.

Naming jobs and output files is a great tool

Add a few extra pieces of information to your run script

HOW TO SPECIFY AND START SERIAL JOBS

```
$ sbatch -n1 ... job.sh

# script
./program
```

HOW TO SPECIFY AND START OPENMP JOBS

```
$ sbatch -n1 -c8 ... job.sh

# script
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
./openmp-program
```

HOW TO SPECIFY AND START MPI JOB

```
$ sbatch -n128 ... job.sh

# script
mpprun ./mpi-program
```

Note that there are no buildenv modules in the scripts. It may even be a good idea to put a "module purge" in there.

MONITORING RUNNING JOBS

The squeue command

```
$ squeue --user=x_petkj

$ squeue --state=running

$ export SQUEUE_USERS=x_petkj
$ squeue --long
```

The jobsh command

```
$ jobsh -j 12345

$ jobsh n123
```

The jobload command

```
$ jobload 12345
n123: cpu% 1205 of 3200   memG 10 of 90
n124: cpu% 1206 of 3200   memG 10 of 90
```

NSC BOOST TOOLS

```
$ nsc-boost-timelimit --help
Usage: nsc-boost-timelimit [options] <job id>

This command will increase the time limit of one running job to the
specified amount.
```

```
$ nsc-boost-priority --help
Usage: nsc-boost-priority [options] <job id>

This command will boost the priority of one queued job.
```

THE END

MONITORING, ANALYZING, UNDERSTANDING

IS THIS RUNNING OK?

A QUICK LOOK

Use jobload and compare what it reports against what you expected

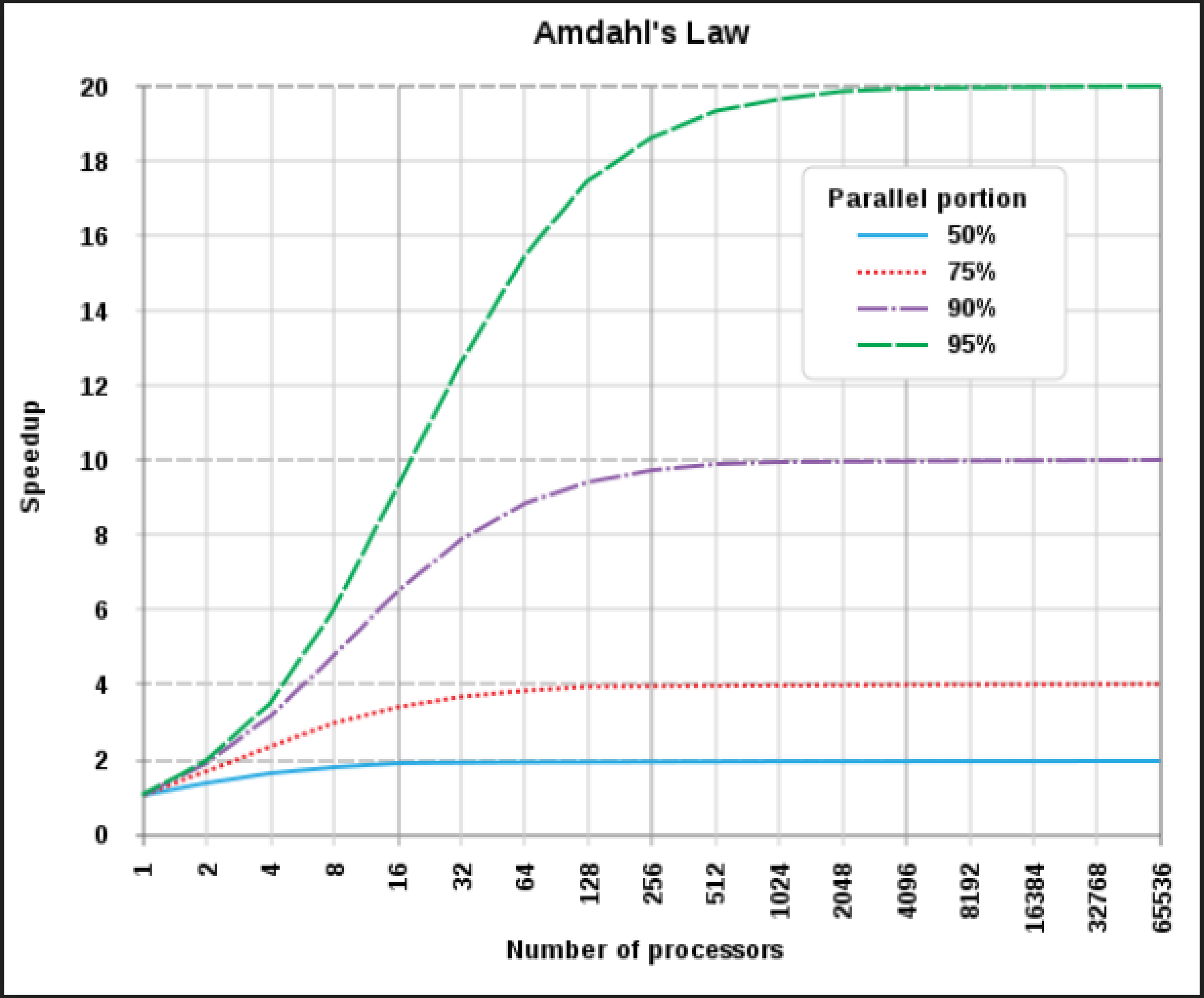
```
$ sbatch -n 64 -t 10 job.sh
Submitted batch job 12345

$ jobload 12345
n123: cpu% 1205 of 3200   memG 10 of 90
n124: cpu% 1206 of 3200   memG 10 of 90
```

A CLOSER LOOK THE MANUAL WAY

Combining jobsh with things such as:

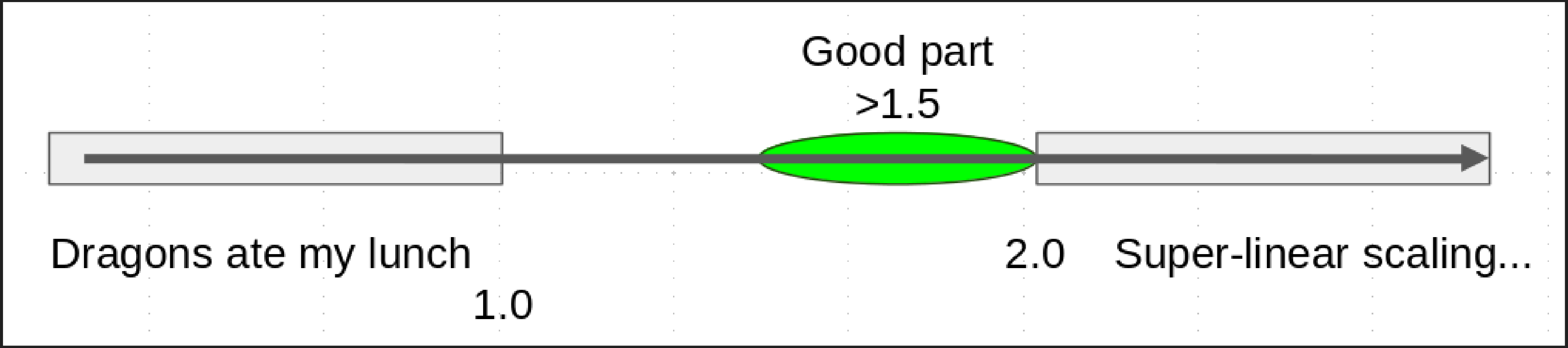
1. htop
2. hwloc-ps
3. collectl

HOW DO PARALLEL PROGRAMS BEHAVE?



PERFORMING A SIMPLE SCALING ANALYSIS

1. Figure out how to time or otherwise measure the speed of your application
2. Run it at the size you think will be good
3. Run it again using half the resources (nodes / cores)
4. Compute the score as the ratio of the two measurements



GENERATING A PERFORMANCE REPORT (AVAILABLE IN NEXT MPPRUN RELEASE)

Using Allinea (now Arm) performance report to gain insight into an MPI type job.

```
#!/bin/bash
#SBATCH -n64
#SBATCH -t 2:00:00

module purge
mpprun --perf-report mpiapp.x
```

Then view the resulting .txt or .html file.

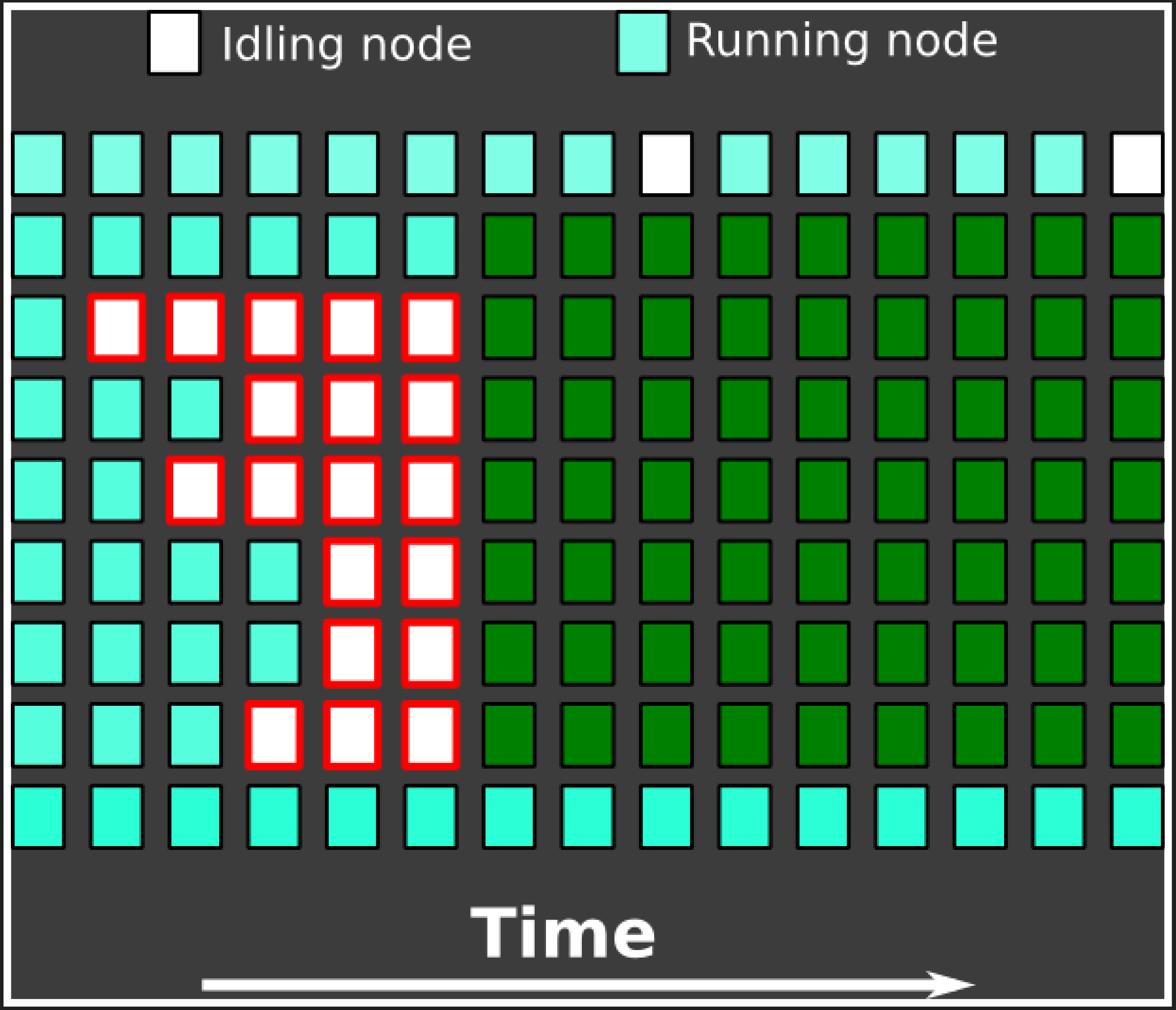
THE END

HIDDEN INEFFICIENCIES

SYSTEM OVERHEAD

1. Job overhead
   - Scheduling cycle
   - node checks (start and stop)
   - Job accounting
2. Scheduling overhead

SCHEDULING OVERHEAD



SYSTEM OVERHEAD

GENERAL RECOMMENDATION: THE SYSTEM IS TUNED TO WORK BEST FOR BATCH JOBS WHICH ARE LONGER THAN CA. 1 HOUR

There are alternatives for development and testing, e.g. the 'devel' reservation and the `interactive` command.

HOW TO HANDLE RUNNING MANY SHORT JOBS:

- bash loops + `srun` within your run script
  - quickly becomes complicated
  - requires bash coding experience

HOW TO HANDLE RUNNING MANY SIMILAR JOBS

RUN MANY SLURM ALLOCATIONS USING E.G.:

- SLURM job arrays
  - e.g `sbatch --array=0-99%4` will run 100 jobs with no more than 4 running at any one time

HOW TO HANDLE RUNNING MANY SIMILAR JOBS

RUN MANY INSTANCES WITHIN A SINGLE SLURM ALLOCATION USING E.G.:

- GNU parallel
- make, snakemake

RUNNING MANY SMALL JOBS

SOME GUIDELINES:

- Test your setup before full production jobs
- Do not have more that a few hundred jobs queued + running at any one time (resource usage in the job scheduler is limited to 40000)
- Consider how to store your data.

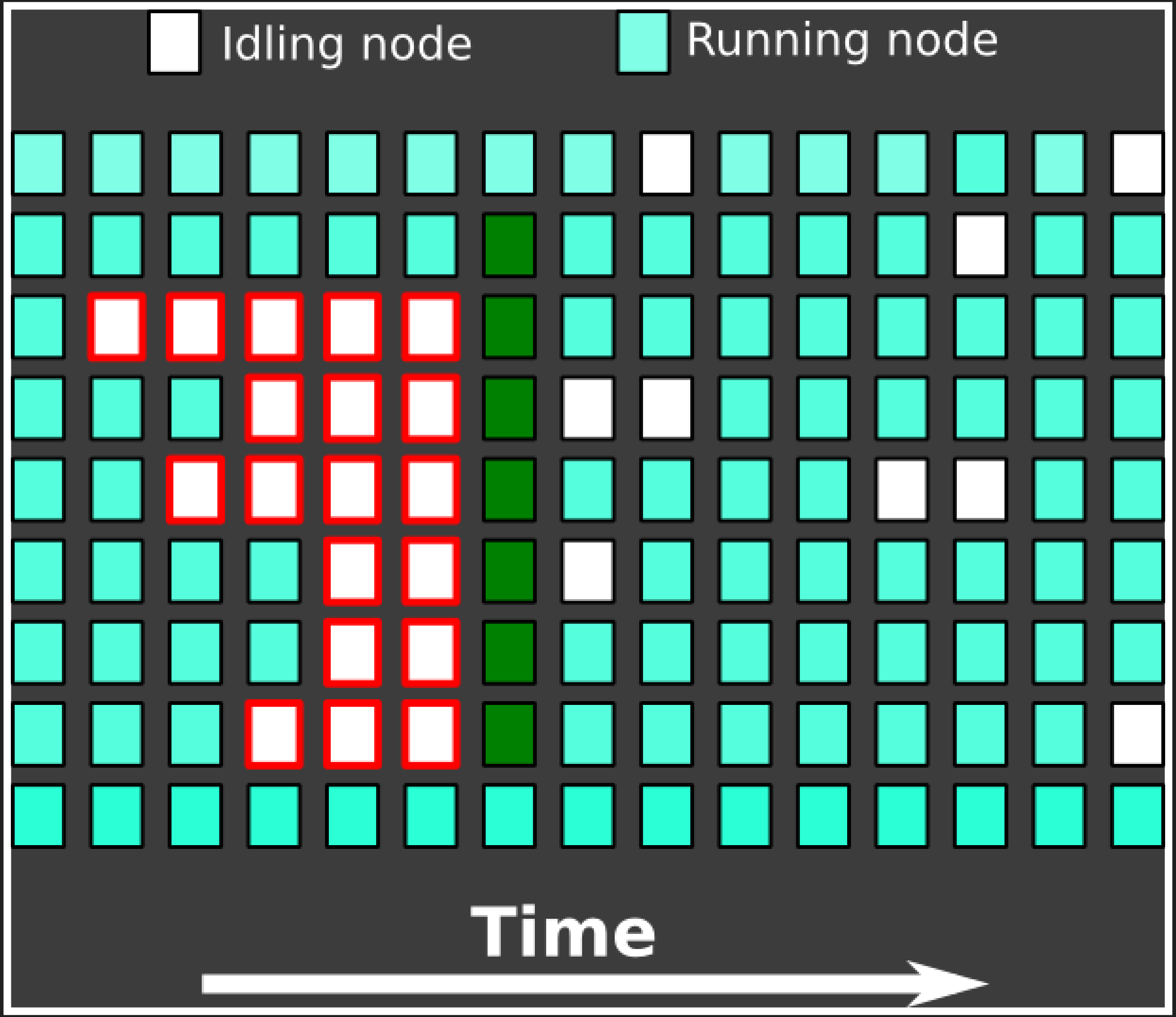
If you are unsure or require help please contact us (support@nsc.liu.se)

BASIC DATA MANAGEMENT

RECOMMENDATIONS:

- Use separate working directories for your submitted jobs
- Use sensible names for working directories and files
- Try at least some basic data management:
  - Remove redundant files and folders
  - Compress your data where appropriate (gzip)
  - *Keep down the number of files (e.g.* `tar` *or* `cat`*)*
  - Keep track of how much space you use (`snicquota`)

FLAT JOBS



FLAT JOBS

In some cases flat jobs are necessary and will be permitted (e.g for application scaling tests).

Please contact NSC Support before queueing any such jobs as there are ways to minimize the impact to other users from your jobs (e.g node reservations).

HOW TO GET SUPPORT

1. email: support@nsc.liu.se
2. SUPR (menu: Support)

WHAT CAN YOU EXPECT HELP WITH:

- Administration, resource allocations and security
- Using SUPR
- Running jobs
- Using installed software
- Software installation requests
- Help with building (compiling) software
- Using storage
- ...

Please dont suffer in silence. If you have an issue/problem/question don't hesitate to ask.

EXTENDED SUPPORT

Advanced support can be provided in the form on short or long-term projects decided on a case-by-case basis. See some examples here.

Example topics:

- performance analysis, tuning and optimizing code
- parallelizing code

We are also happy to take suggestions for:

- creating tutorials and manuals
- organizing workshops

HOW TO WORK EFFICIENTLY WITH SUPPORT

(WRITING GOOD SUPPORT QUESTIONS)

How efficiently your support request is processed is strongly correlated with how well you describe the issue/problem.

- Basics:
  - Include a descriptive subject to your email
  - State which resource you are using (e.g. Tetralith)
  - Give your UserID
  - What were the exact steps you took (e.g. modules loaded, etc.)
  - What error messages do you get? What do you expect, what do you actually see?

LOGGING IN, ACCESSING THE SYSTEM

- What OS does your computer run?
- What client software are you using? (OpenSSH, putty, Thinlinc, ...)?
- What time did you last attempt a login?
- **Never send your password**

USING INSTALLED SOFTWARE

- Software name, version
- what modules are loaded, other environment changes?
- Provide full paths to input and output files

RUNNING JOBS

- List of job IDs (use `lastjobs` to help)
- Description of software (e.g. installed software or built locally?)
- Save the input and output files and include the full paths in your support email.

SOFTWARE INSTALLATION REQUESTS

- Software name, version
- Where can the software be downloaded?
- Type of license

NSC DOCUMENTATION

General Tetralith information including:

- Tetralith migration guide
- Tetralith getting started guide
- Tetralith software list
- The module system
- Storage

General NSC software environment

NSC user support

Tetralith and Sigma software list

SOFTWARE DOCUMENTATION ON TETRALITH

SOME INSTALLED SOFTWARE INCLUDES DOCUMENTATION. E.G. GROMACS:

```
$ module purge
$ man gmx
$ module load GROMACS/2018.1-nsc2-gcc-2018a-eb
$ man gmx
$ man gmx-gangle
```

SOFTWARE DOCUMENTATION EXAMPLE: POV-RAY

```
$ module purge
$ man povray
$ module load POV-Ray/3.7.0.0-nsc1
$ man povray
```

Connect to Tetralith via Thinlinc and in a browser, e.g.:

- /software/sse/manual/POV-Ray/3.7.0.0-nsc1-intel-2018a-eb/share/doc/povray-3.8/html/index.html
- /software/sse/manual/git/2.19.1/gcc485/share/doc/git/git.html

EXTERNAL DOCUMENTATION

Most softwares have external web pages, some with user guides, support/forums ... e.g.

```
$ module help namd/2.12-nsc1-intel-2018a-eb
$
$ ------------------------------- Module Specific Help for "namd/2.12-nsc1-intel-2018a-eb" ----------------------------
$ NAMD molecular dynamics software
$ http://www.ks.uiuc.edu/Research/namd/
$
```

Copy and paste link into your browser.

SUMMARY / KEY POINTS

- Access:
  - ssh keys
  - modify your ssh config file
- Convenience tools (persistent sessions):
  - If you do not use ThinLinc, consider `screen` or `tmux`
- Keep up-to-date with project status:
  - `projinfo, snicquota, lastjobs` or SUPR
- Jupyter Notebooks
  - Can be run on both login and compute nodes

SUMMARY / KEY POINTS

- Building software
- Running applications
  - `man sbatch`
  - `man squeue`
- Resource utilization
  - `jobload`
  - Try a simple scaling test
- Support
  - Help us help you (by writing well structured support questions)