Introduction to the HoRUS cluster

Jan Steiner
Zentrum für Informations- und Medientechnik

November 5, 2020

A word about Zoom

- Exercises:
  - Groups of three
  - One person shares screen
  - Solve cooperatively
  - Screen-sharer switches for next exercise
  - I will switch through

- Breakout rooms: let’s assign them now

Who am I

- Jan Steiner
  - Aerospace Engineering, Uni Stuttgart (grad. 2010)
  - PhD thesis at German Aerospace Center Braunschweig
    - Numerical simulation of aircraft icing
  - At ZIMT since July 2017
- Area (with one other colleague):
  - HPC training and support
  - Training courses (once every semester)
    - This course
    - Linux
- Additional support: performance optimization

Outline

1. Getting onto the cluster
   - Structure of a cluster
   - Getting access and help
   - Connecting to the cluster
   - *Exercise 1: setup, login*
2. Using the cluster
   - Workspaces
   - Environment modules
   - Jobs
   - *Exercise 2: your first job script*
3. SLURM explained
   - Tasks, processes, cores
   - Miscellaneous SLURM stuff
   - *Exercise 3: SLURM options*

About 90 minutes

About 30 minutes

Lunch break roughly here (60 min)

Outline

1. Getting onto the cluster
   - **Structure of a cluster**
   - Getting access and help
   - Connecting to the cluster
   - *Exercise 1: setup, login*
2. Using the cluster
   - Workspaces
   - Environment modules
   - Jobs
   - *Exercise 2: your first job script*
3. SLURM explained
   - Tasks, processes, cores
   - Miscellaneous SLURM stuff
   - *Exercise 3: SLURM options*

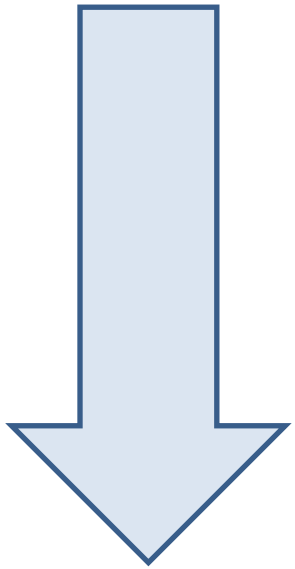
Background

- Computations can become too large for one computer
  - Too much concurrent data for RAM
  - Too much total data for hard drive
  - Execution time in months, years or more
  - Too many small problems (e.g. parameter study)

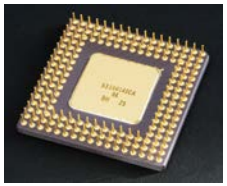
→ **Use more computers**

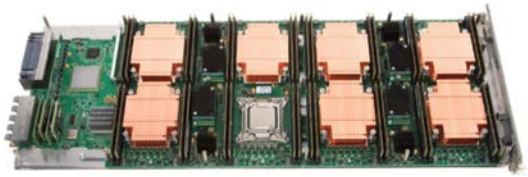
- Cluster of computers
  - Components similar to PC
  - But many, and interconnected

Physical structure of a cluster



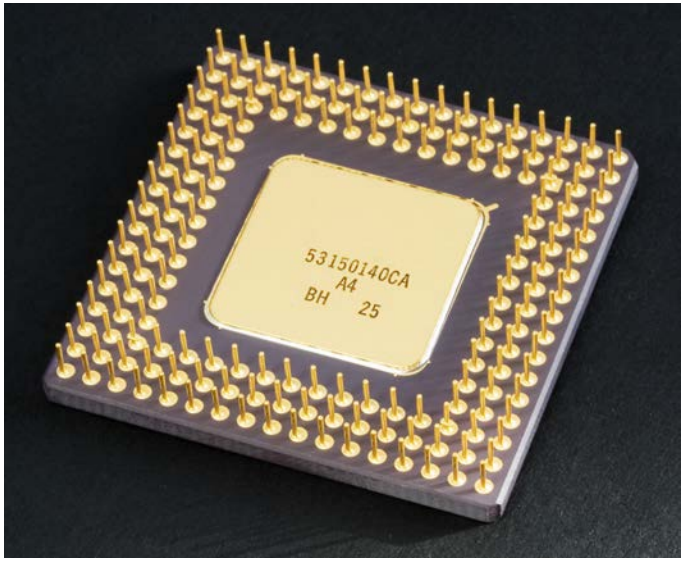
- Core (Processor)
- Node (Blade)
- Rack (Cabinet, Chassis)
- Cluster (Supercomputer)





Cores

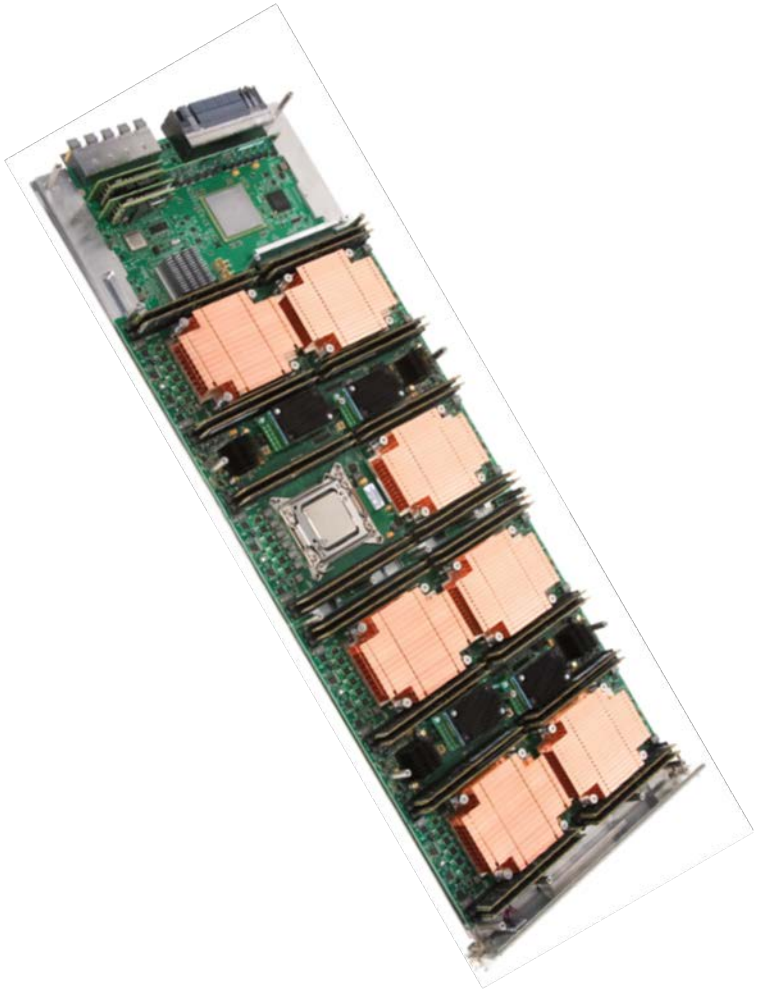
- (Almost) identical to PC processors
- General purpose
- Hyperthreading: two cores in one
- Sometimes specialized
  - E.g. graphics processors (GPU)
  - Limited operations, but faster



Source: Wikimedia Commons

Nodes

- Similar to PC motherboards
- 4-16 CPUs (+Hyperthr.)
- Usually central RAM
  - HoRUS: 48 GB
    - `medium2`: 64 GB
- Types
  - Compute, Login, Management
  - “Fat” (more RAM), GPU
    - `smp1`: 512 GB RAM



Cabinet

- Houses multiple nodes
- Cooling
- Power supply
- Interconnect (Network)
  - Faster than regular Ethernet
  - Makes cluster a cluster
  - HoRUS: Infiniband

Cluster

- Multiple cabinets
  - HoRUS: 2 cabinets, 150 nodes, 1800 cores
  - SuperMUC: dozens of cabinets, 241000 cores
- Infrastructure (e.g. fire suppression)
- Central file storage (hard disks)
  - Sometimes individual nodes have hard disks

Situation at Uni Siegen

- Current: multiple systems
  - HorUS cluster
  - HPE Moonshot (HTC nodes)
  - NEC Aurora vector computer
  - ARM cluster
- Future (very soon): OMNI cluster
  - 3-4 times more regular CPUs
  - nVIDIA GPUs
  - OpenStack and Kubernetes

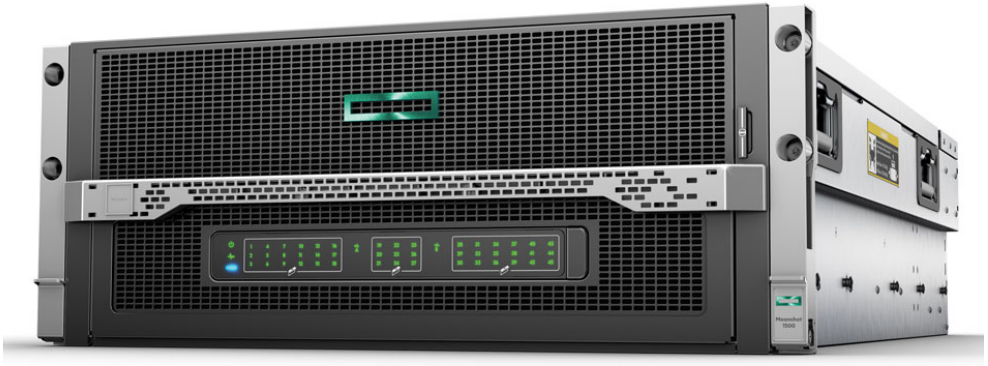
- Start date new cluster: early December

Situation at Uni Siegen

- HorUS Cluster
  - 136 regular compute nodes `cn001-cn136`
    - 2x6 Westmere CPUs, 48 GB RAM each
  - 20 newer nodes (Carolus) `cn137-cn156`
    - 2x8 CPUs, 64 GB RAM each
    - `medium2` queue
  - 1 SMP (Shared Multiprocessing) node `smp1`
    - 32 CPUs, 512 GB RAM each
    - `smp` queue
  - Around 40-60 TB total hard drive space
  - 2 login nodes, 1 management node, 1 Preprocessing node `pre1`

Situation at Uni Siegen

- HPE Moonshot HTC System
  - 48 nodes (2x login, 5x compute)
    - 8 CPUs, 64 GB RAM each
  - Designations: `htc001-htc007`
  - Shares homes with HorUS
  - Much more modern architecture
  - High-Throughput Computing:
    - Smaller jobs, but more



Source: hpe.com

Situation at Uni Siegen

- NEC SX Aurora Tsubasa System
  - 2 machines (“vector host”)
    - 2 cards (“vector engines”) each
  - Intended for testing vector architecture
    - Similar to GPUs
  - Better documentation in the near future
  - Names: `vec01-vec02`

Source: nec.com

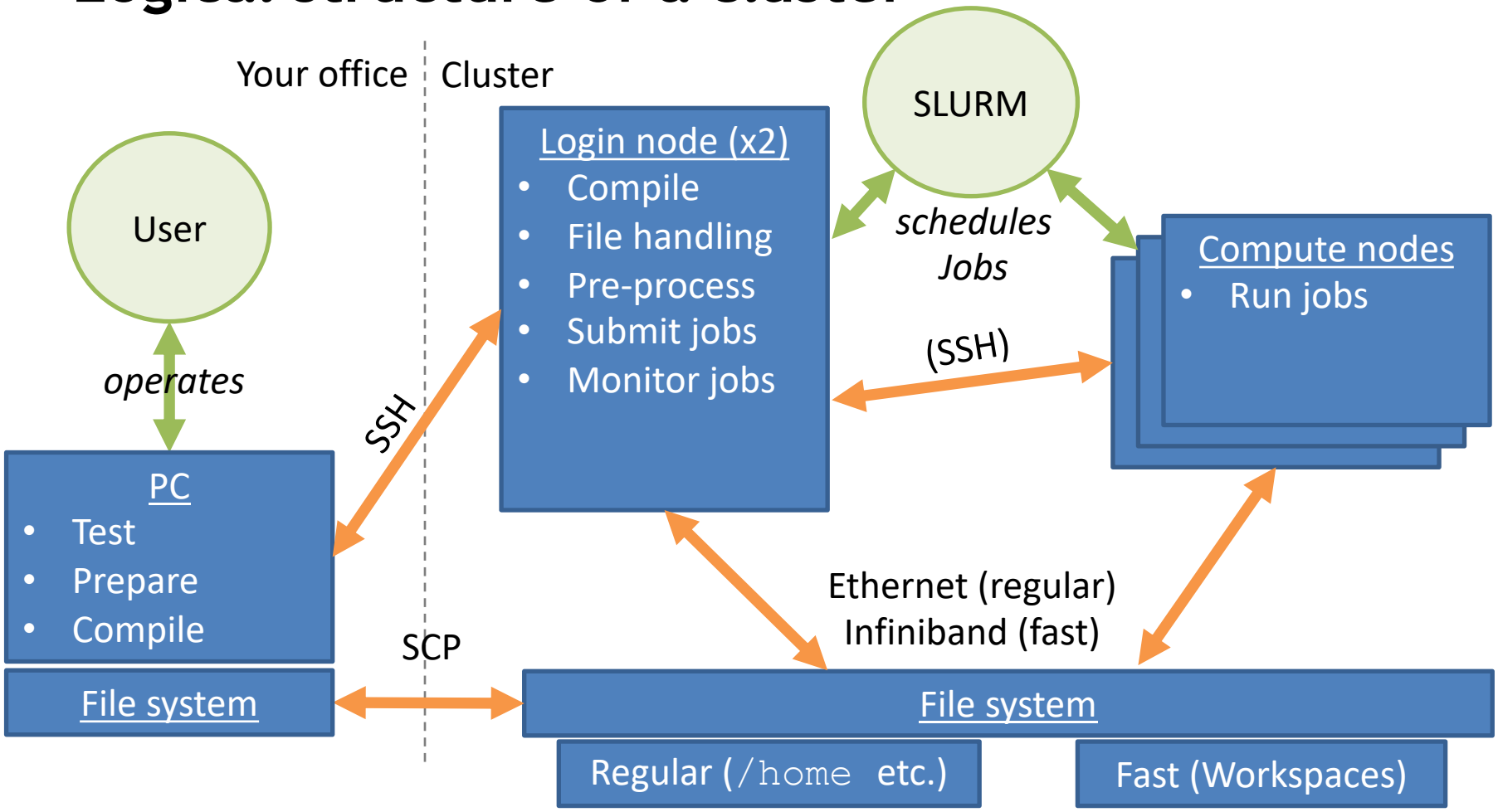
Situation at Uni Siegen

- ARM Cluster
  - ARM architecture
    - Used in mobile devices
    - Potentially more efficient
  - For testing ARM architecture
  - Better documentation in the near future
  - 4 nodes
    - Names: `arm01-04`

Situation at Uni Siegen

- OMNI cluster coming very (!) soon
  - 4 times the size of HorUS
    - 434 regular nodes, 2x AMD CPUs, 256 GB RAM each
    - OpenStack and Kubernetes partitions (5 and 8 nodes)
    - 2 SMP nodes: 4x Intel CPUs, 1.5 TB RAM each
    - 1 PB storage
  - Usable December 1 (hopefully)
  - Usage mostly similar to HorUS
    - Introduction event, “What’s new” guide when running

Logical structure of a cluster



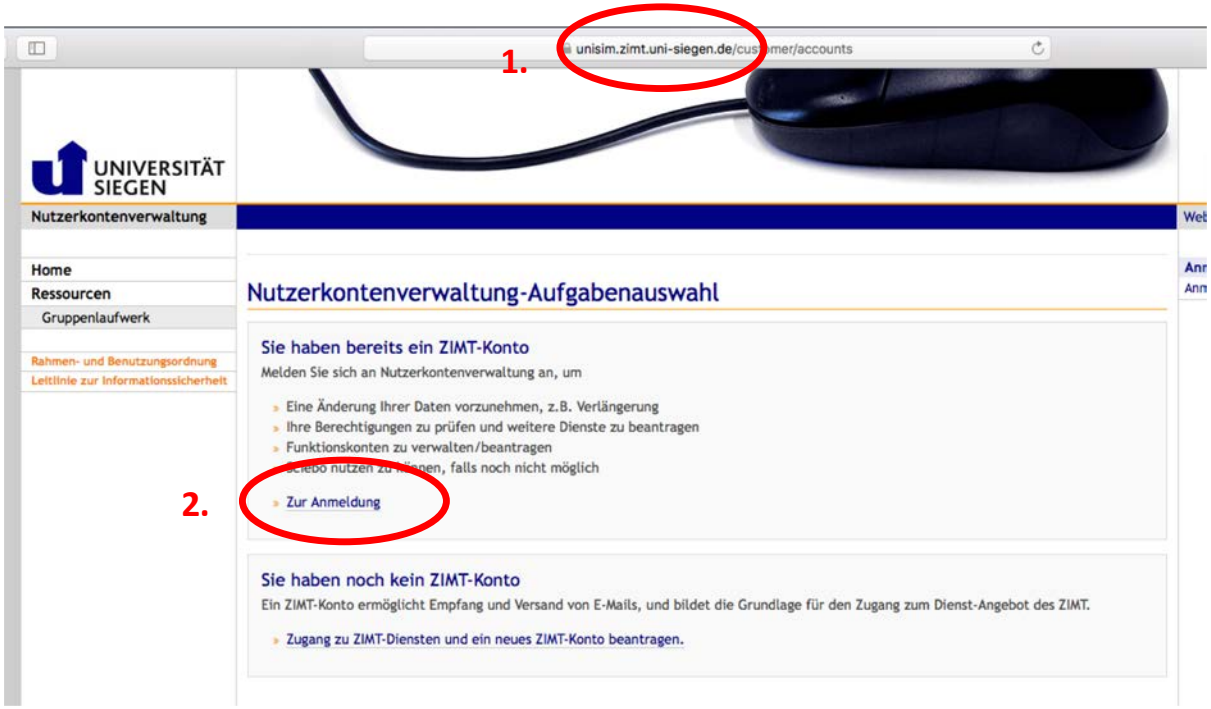
Outline

1. Getting onto the cluster
   - Structure of a cluster
   - **Getting access and help**
   - Connecting to the cluster
   - *Exercise 1: setup, login*
2. Using the cluster
   - Workspaces
   - Environment modules
   - Jobs
   - *Exercise 2: your first job script*
3. SLURM explained
   - Tasks, processes, cores
   - Miscellaneous SLURM stuff
   - *Exercise 3: SLURM options*

Getting access

- Get an account
  - Employees: Nutzerkontenverwaltung
    - Wait
  - Students: supervisor runs script
- Set up an SSH connection
  - Explained in a moment

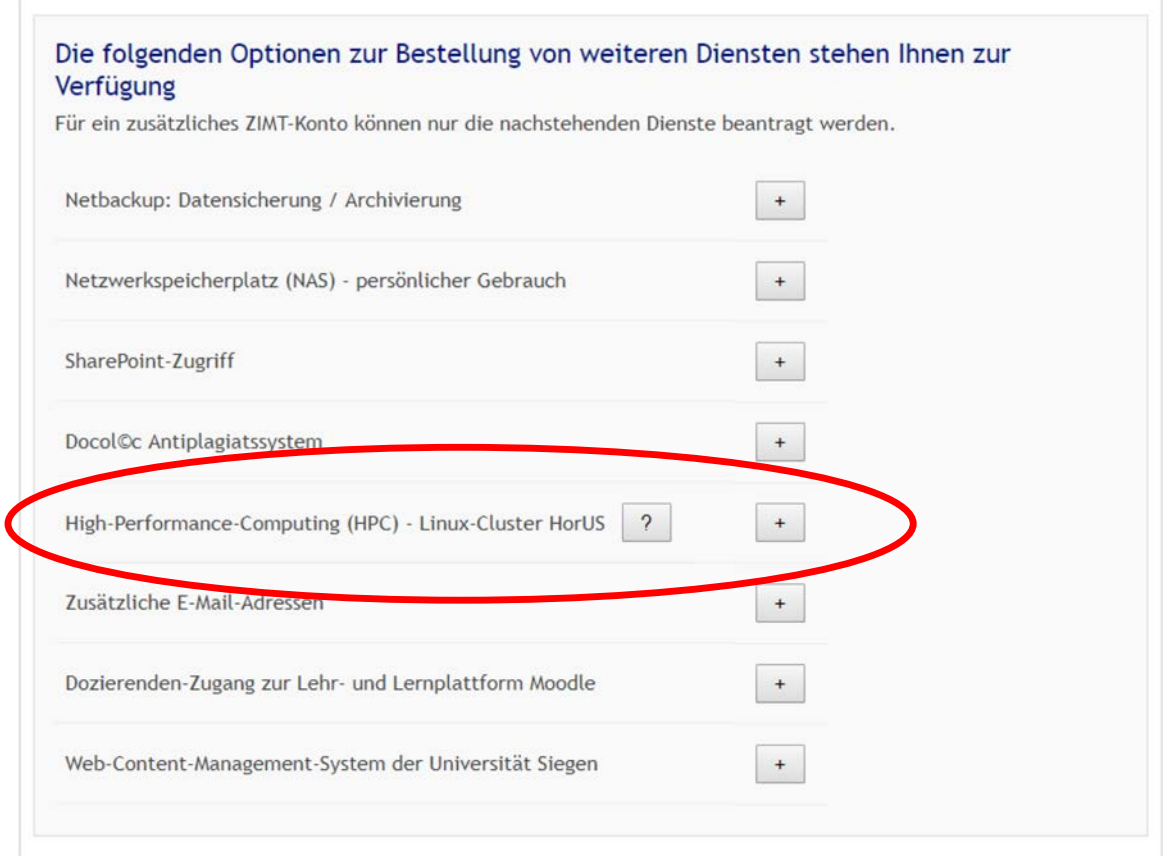
Registering an employee



Registering an employee

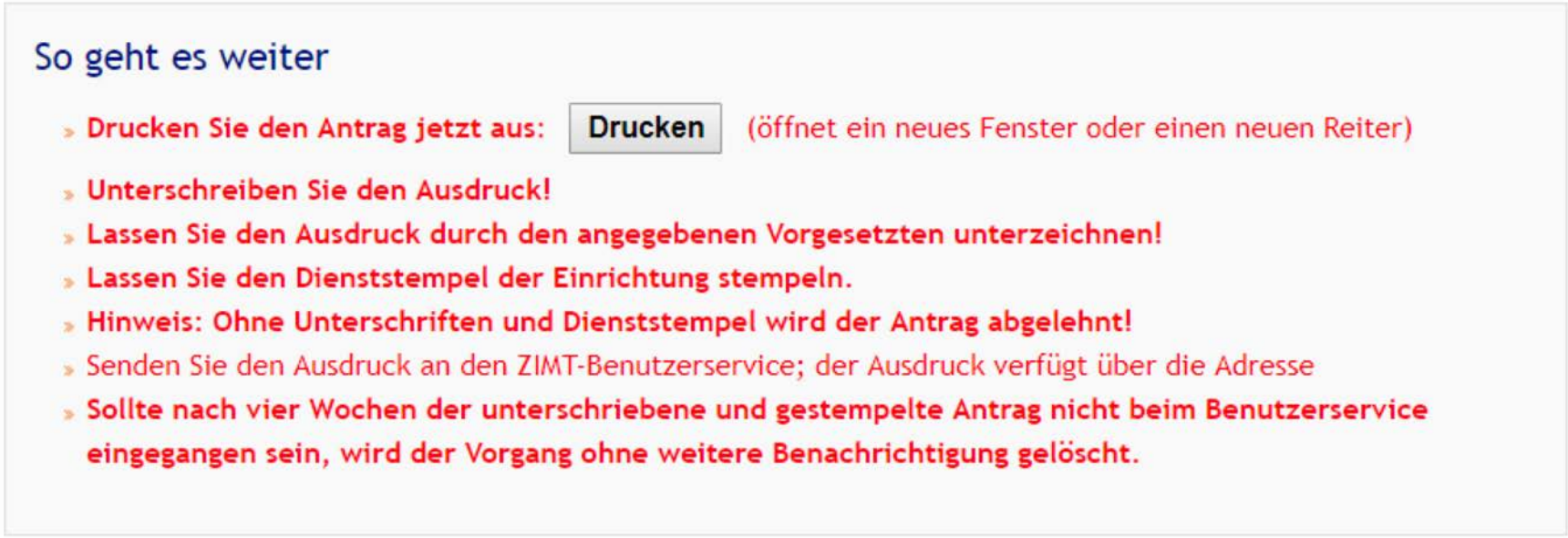
Registering an employee

5.



Registering an employee

5.



Registering a student

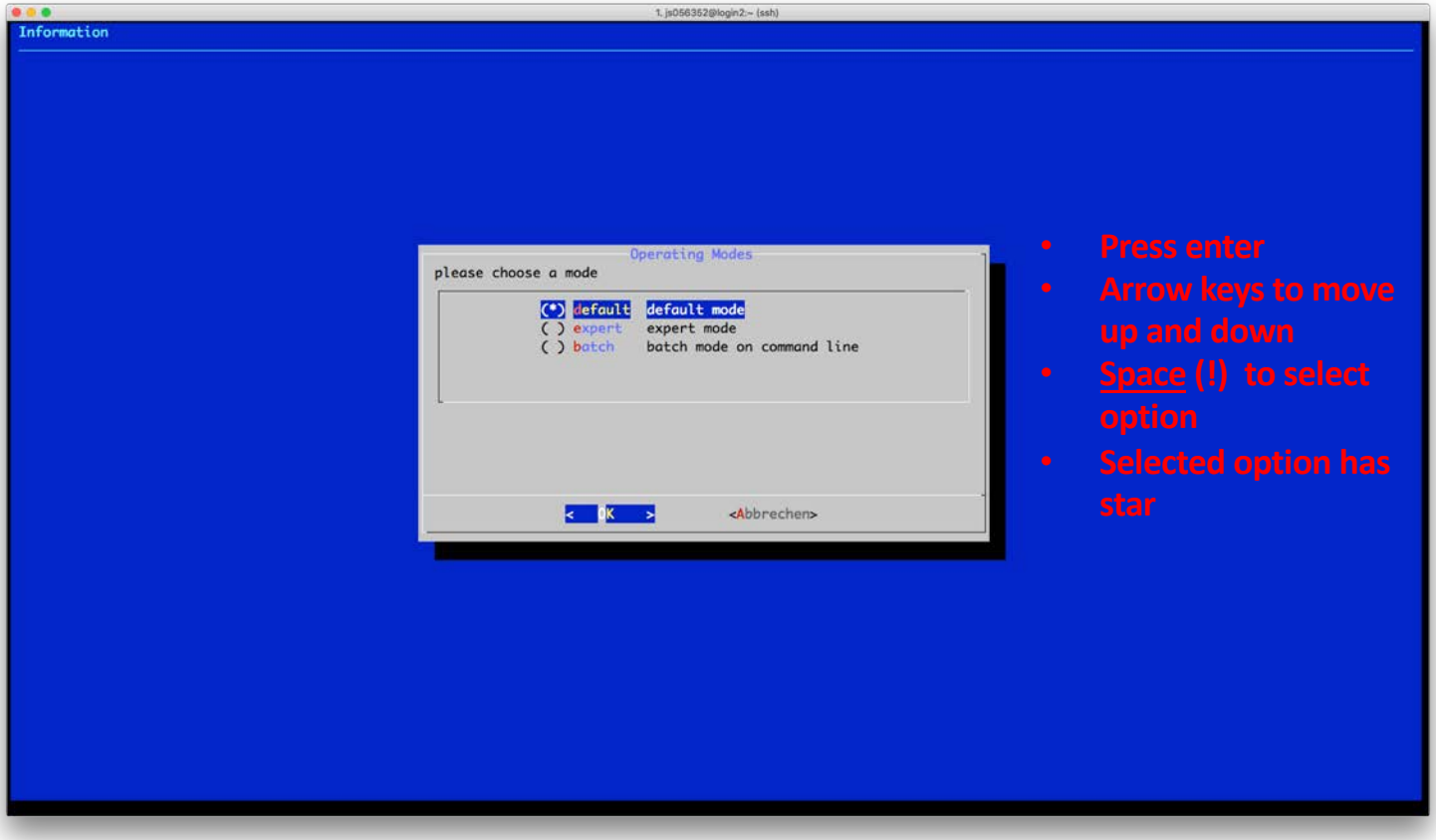
- Every student account has an assigned supervisor
- Student accounts time out eventually
  - Prolong manually
  - Warning before time-out
  - Data not immediately lost
- Software used will be different on OMNI

Registering a student

- Supervisor:
  - Type `kinit`
  - Enter your password
  - Type `module load studentadmin`
  - `hpc_user_add.sh` (or `_list.sh`, `_prolong.sh`)
- Follow dialogue
- Student usually goes in **hpc-student** group

Registering a student

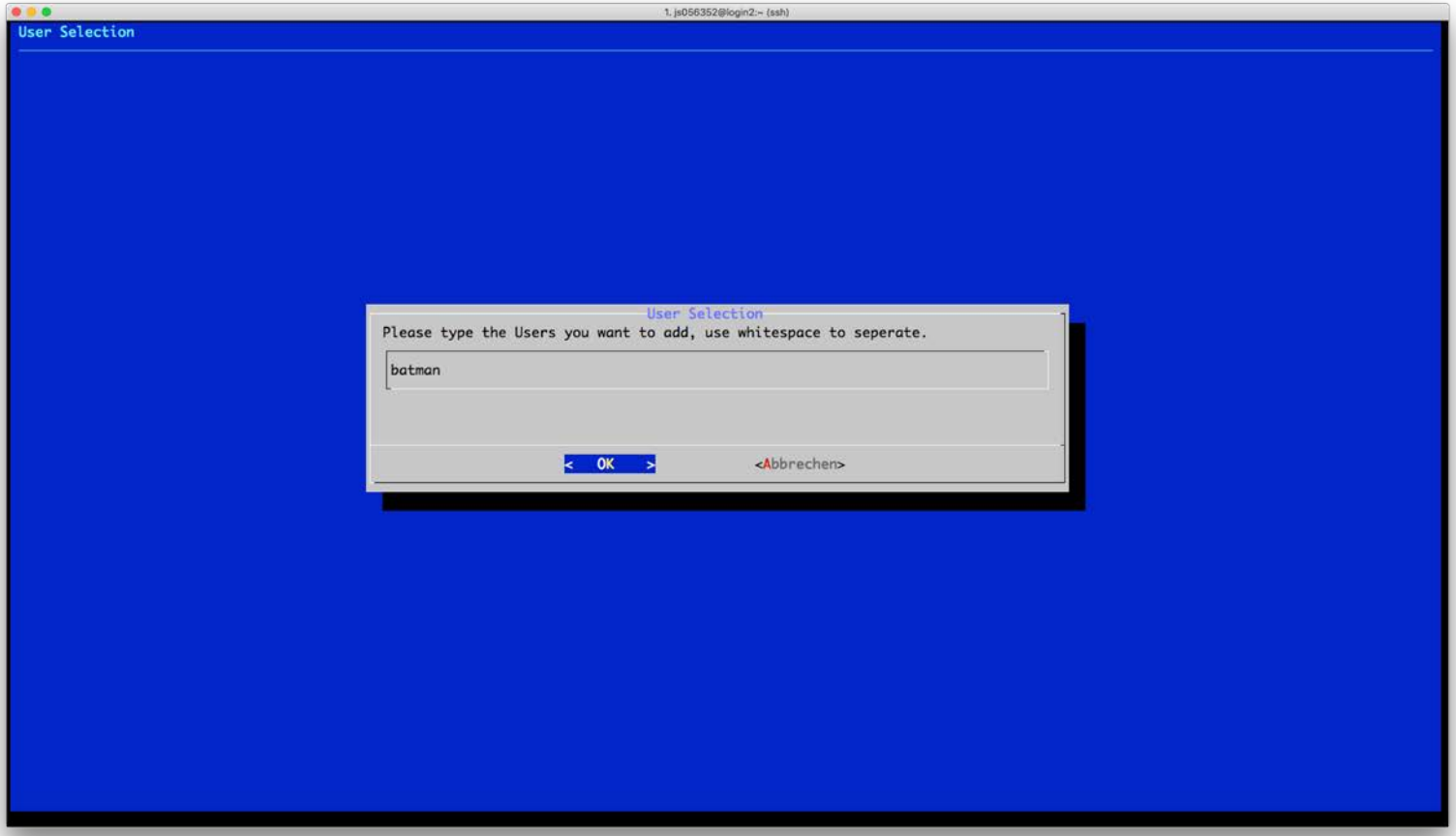
1.



- Press enter
- Arrow keys to move up and down
- Space (!) to select option
- Selected option has star

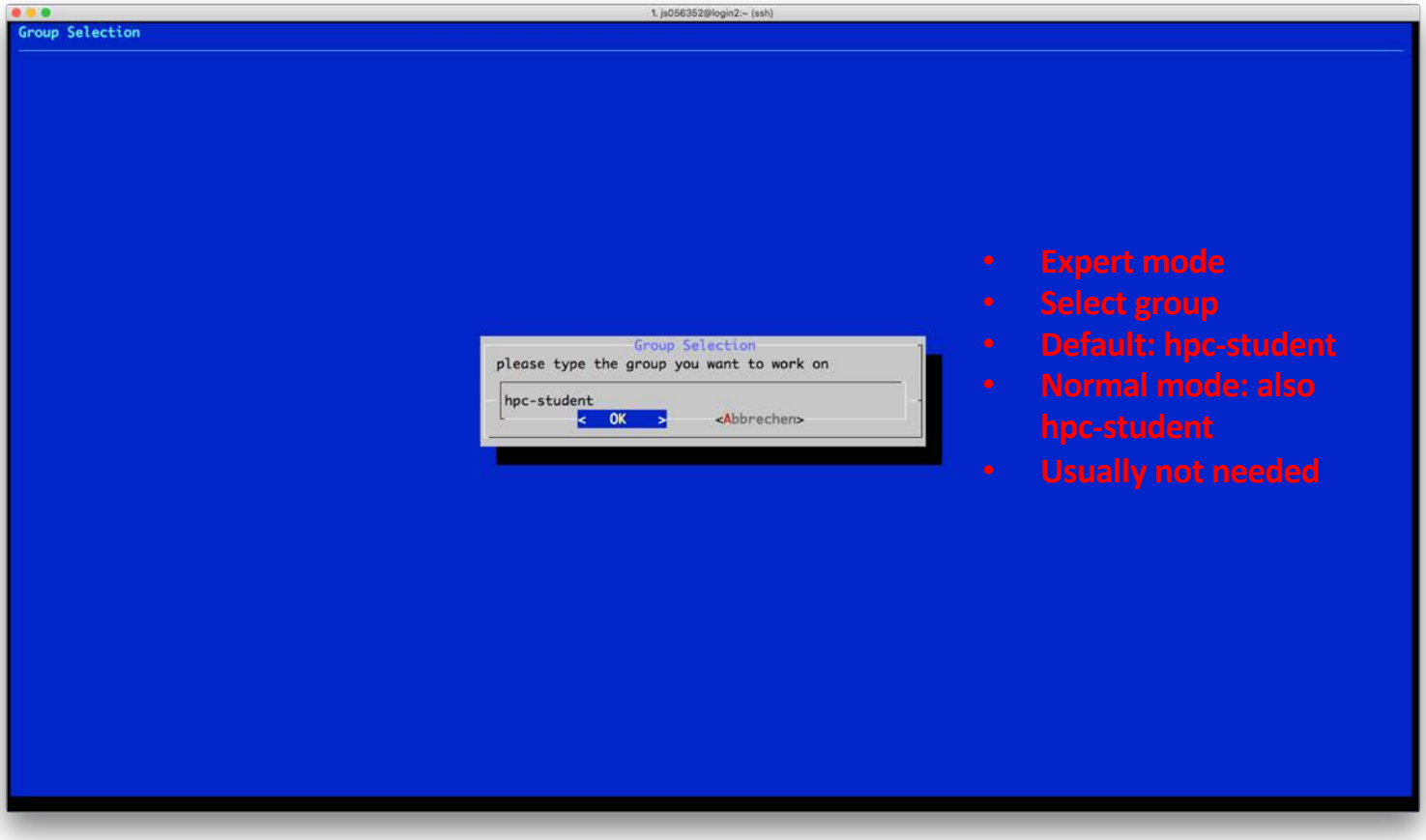
Registering a student

2.



Registering a student

2.



- Expert mode
- Select group
- Default: hpc-student
- Normal mode: also hpc-student
- Usually not needed

Other scripts

`hpc_user_list.sh`

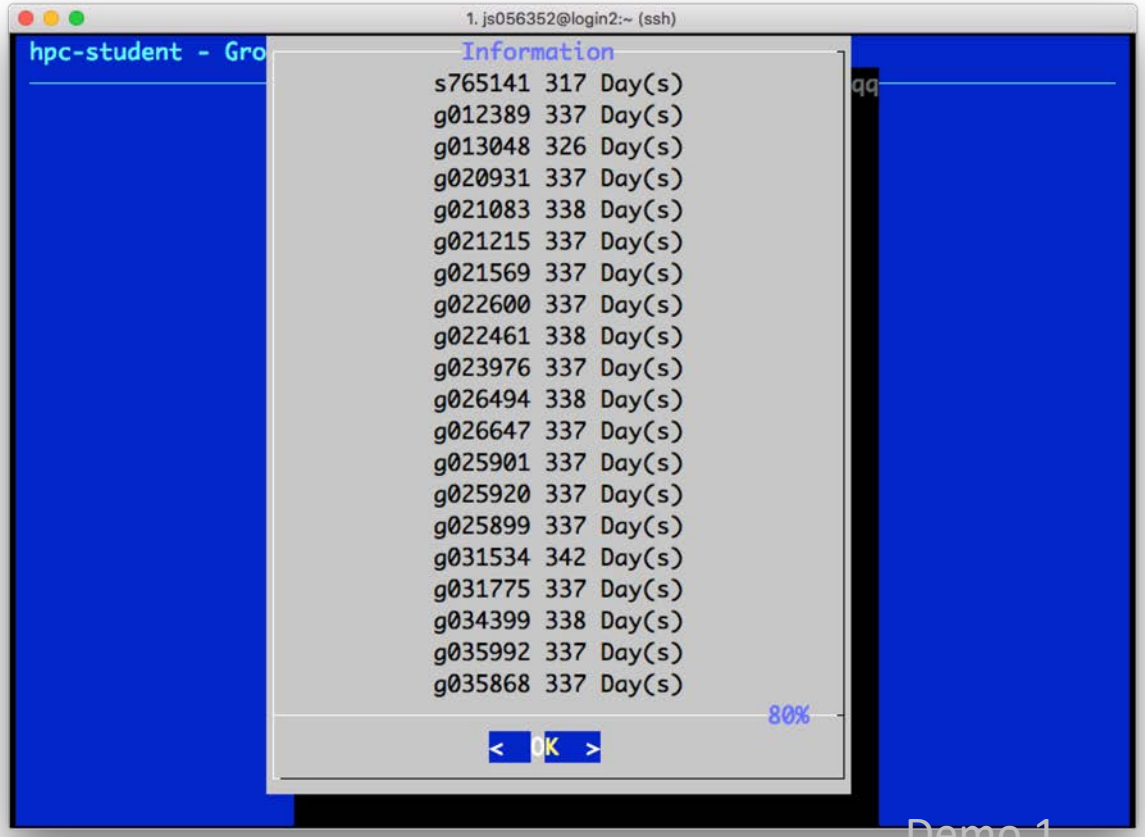
- Type group name (usually hpc-student)
- Page-up/page-down to scroll list
- Enter to leave

`hpc_user_prolong.sh`

- Extends by 365 days

`hpc_user_remove.sh`

- For admins only



Getting help

- Cluster website
  https://cluster.uni-siegen.de
  - Basic usage information
  - What is installed

- Consultation hour
  - Every Tuesday 2 PM – 3 PM
  - Online (link on cluster website → Events page)

- hpc-support@uni-siegen.de

- Google, Stack Overflow, etc.

- Built-in help `man <command>` or `<command> -h` or `--help`

Demo 2

Problems

- Problems: open a ticket
  - Email to hpc-support@uni-siegen.de
  - Centralized ZIMT ticket system

- Entire HPC team sees it
  - Admin stuff: Mr. Pokorra
  - Usage questions and consulting: me

- Please don’t email us directly
  - Person might be on vacation etc.
  - Entire team has an overview what’s wrong
  - Also not good: hpc-team@uni-siegen.de

How to use other resources

- This course covers HorUS only
- OMNI cluster: similar, will give details at go-live
- Using HPC Moonshot: relatively easy, similar to HorUS
- Other resources: get in contact with us

Outline

1. Getting onto the cluster
   - Structure of a cluster
   - Getting access and help
   - **Connecting to the cluster**
   - *Exercise 1: setup, login*
2. Using the cluster
   - Workspaces
   - Environment modules
   - Jobs
   - *Exercise 2: your first job script*
3. SLURM explained
   - Tasks, processes, cores
   - Miscellaneous SLURM stuff
   - *Exercise 3: SLURM options*

Connecting to the cluster

- You can connect from any system via console
  - Linux: Easiest
  - Mac OS: Relatively easy
  - Windows: now also built in

- Outside university network:
  - Needs VPN for user/password access
  - VPN not necessary for key-based access

SSH Software

- Clusters typically accessed via Secure Shell (SSH) protocol
- Most commonly OpenSSH software
- For all operating systems
  - Linux: original
  - Mac OS: basically identical
  - Windows 10 (since 2019): integrated in cmd/Powershell
- Additional tools, especially on Windows: Putty, MobaXTerm

SSH Basic Use

- Connect with `ssh` command

```
ssh [options] <username>@<hostname>
```

- You will be asked for password
- Alternative: set up public/private key pair
- Can specify configurations to simplify login
- Console-based, but opening windows possible
- Multiple simultaneous connections possible

Demo 3

SSH Configuration

- OpenSSH allows setting presets
- Directory `~/.ssh` contains config file
  - Simply named `config`
  - Editable text file
- One preset per cluster
  - Specify username
  - Other options (many possibilities)

```
host shu
  HostName shu.sts.nt.uni-siegen.de
  User js056352
  TCPKeepAlive yes
  ForwardX11 yes

host horus
  HostName horus.zimt.uni-siegen.de
  User js056352
  TCPKeepAlive yes
  ForwardX11 yes

host hpc
  HostName hpc.zimt.uni-siegen.de
  User js056352
  TCPKeepAlive yes
  ForwardX11 yes
#  ForwardX11Trusted yes

host shutest
  HostName shu.sts.nt.uni-siegen.de
  User js056352
  TCPKeepAlive yes
  ForwardX11 yes
  Port 22

host *
  XAuthLocation /opt/X11/bin/xauth
```

- `ssh <presetname>` instead of `ssh <options> <user>@host>`

SSH Key-based authentication

- Login with public/private key pair instead of password
- Convenient
  - Good for automated connections
- Potentially more secure
- Only as secure as your PC
  - **Treat private key file like a physical key**

Key pair workflow

1. You generate key pair
   - On your PC
   - Tool `ssh-keygen` (comes with OpenSSH)
   - Keys are text files in ~/.ssh
2. You copy public key to cluster
   - `ssh-copy-id` (comes with OpenSSH)
   - Windows: manually copy and paste key
3. When logging in, OpenSSH will select key

Key generation

- SSH key generator
  - On **local** PC, type `ssh-keygen`
  - Enter filename for new key
    - Should be inside `~/.ssh` directory
    - **Caution**: will overwrite without asking
  - Enter passphrase
    - Can be left empty, but not recommended
  - Confirm passphrase

Key generation

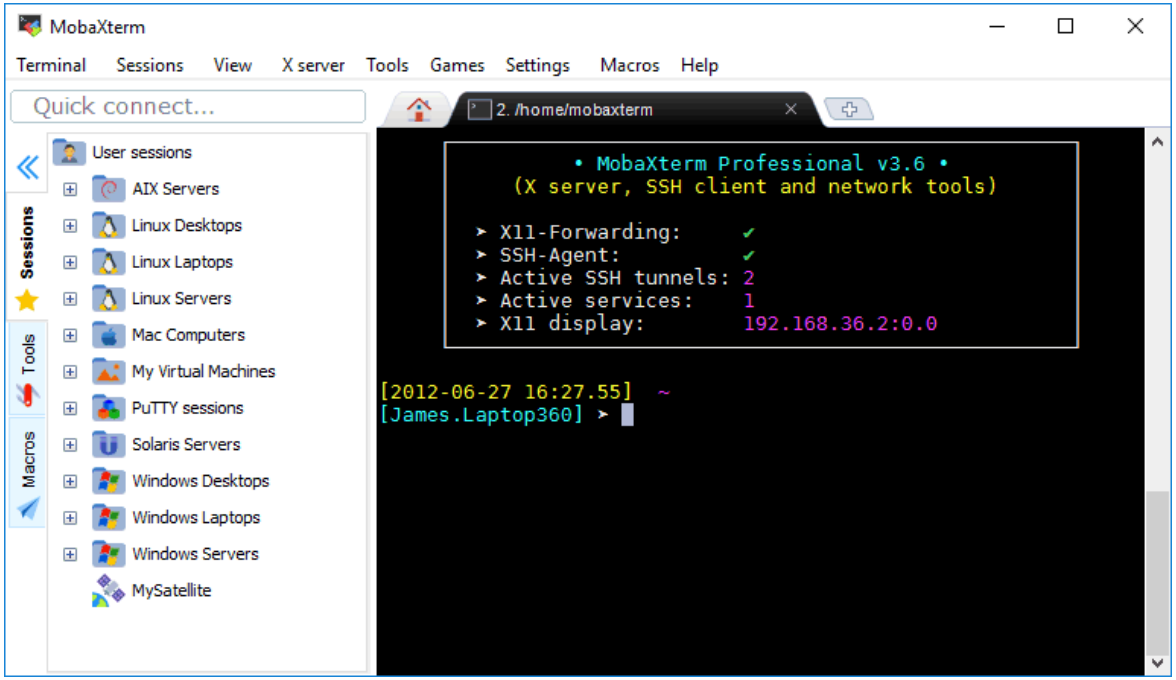
- When logging in, key will be used automatically
  - May specify key file manually if needed (option `-i`)
  - If you get asked for password, key not recognized

- Tips:
  - Use one key per PC (in case of theft/compromise)
  - Not recommended to leave passphrase empty
    - But only needs to be entered once

Demo 4

Windows SSH Software

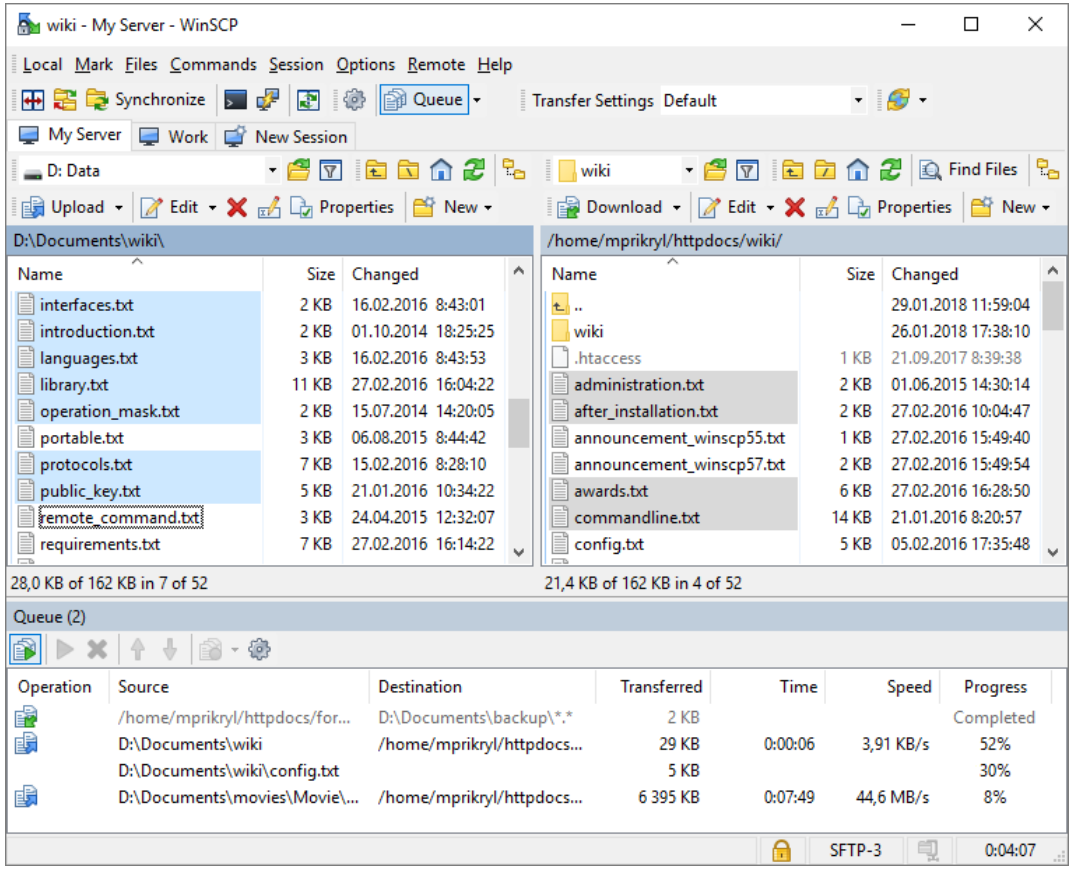
- Windows system: MobaXTerm
  - Free software (mobatek.net)
  - All-in-one client
  - Does not need to be installed
  - Specify host and user
  - Good for newbies



Source: mobatek.net

Windows SSH Software

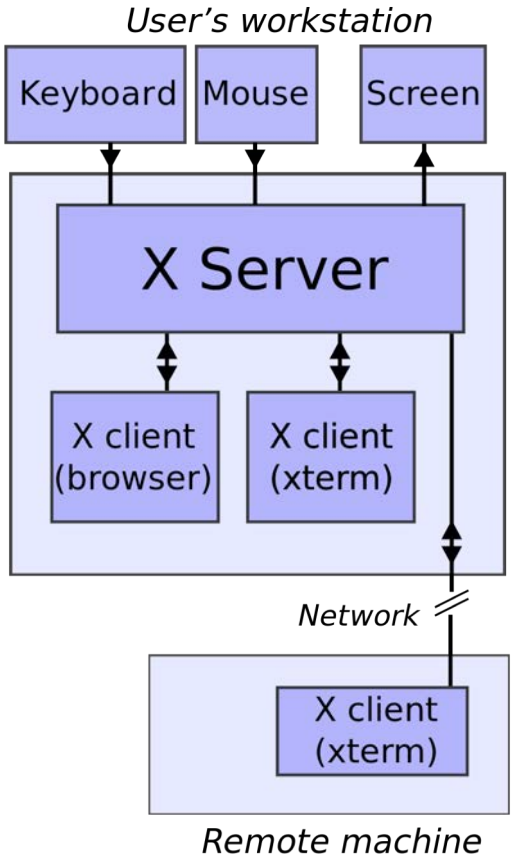
- Windows system: Putty+WinSCP
  - SSH client
  - Separate SCP client
  - Specify host and user



Source: winscp.net

Linux Graphical User Interface

- X window system
- Basis of all Linux displays
- Can display windows from other computers
- X server needs to run on PC
- X client is software that window belongs to
- X windows can be transmitted by SSH connections



Graphics via SSH

- Requirements
  - X server installed on PC
  - SSH connection with X support
  - (Cluster supports X windows)

- Linux: X server built in
- Mac OS: Xquartz
- Windows: xming, MobaXTerm

Connecting with X support

- Enable X support in SSH

  `ssh -X <user>@<host>`

  - Must be upper case X
  - Sometimes -Y used
    - "Trusted" connection
    - Less safe, sometimes necessary for things to work

- In config file: `ForwardX11 yes` or `ForwardX11Trusted yes`

Demo 5

File Transfer

- Copying files between PC and cluster:
  - Use `scp` command (secure copy)
- Syntax similar to Linux `cp` command
- Uses SSH, can use same settings/presets
- Console-based, has many front-ends for all OSes
  - WinSCP, SSHFS

File Transfer

- Syntax:

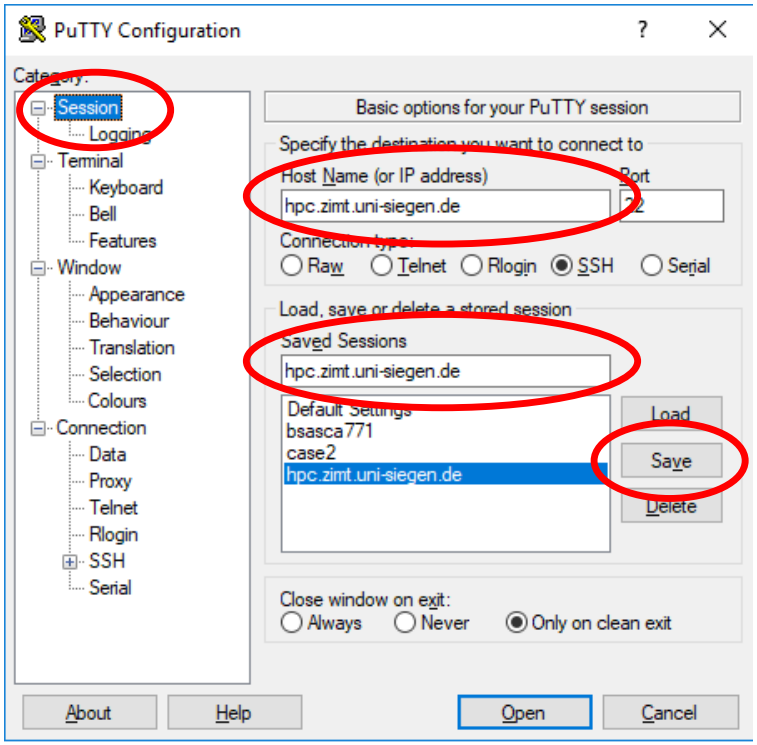
```
scp [options] sourcehost:sourcefile targethost:targetfile
```

  - Host may be left out if local
  - Host may be SSH preset
  - Source or target or both can be remote
- Same rule as `cp` about `-r` when copying entire directories
- Unlike `cp`: will print status of file transfer to screen
- Not only possibility (`rsync`)

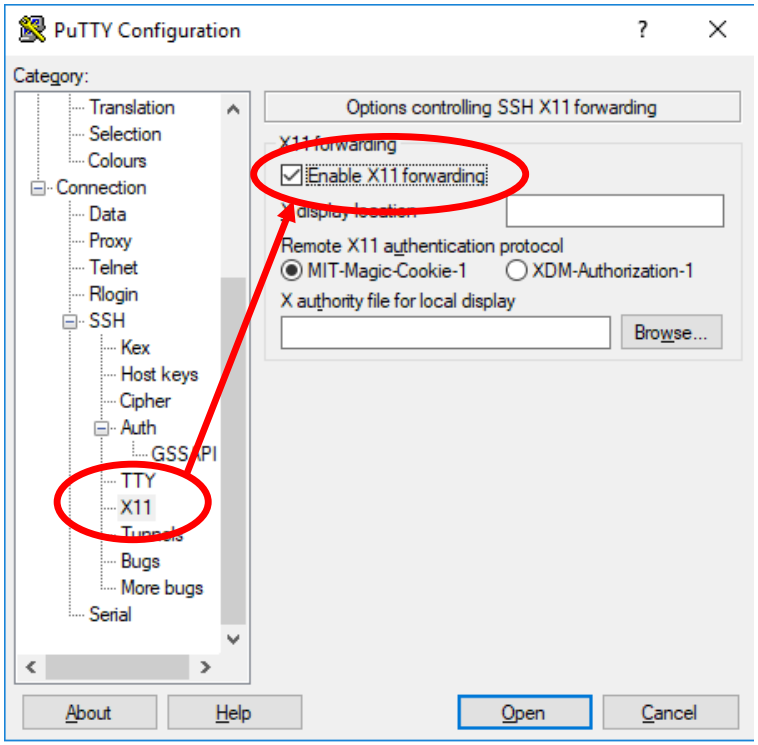
Demo 6

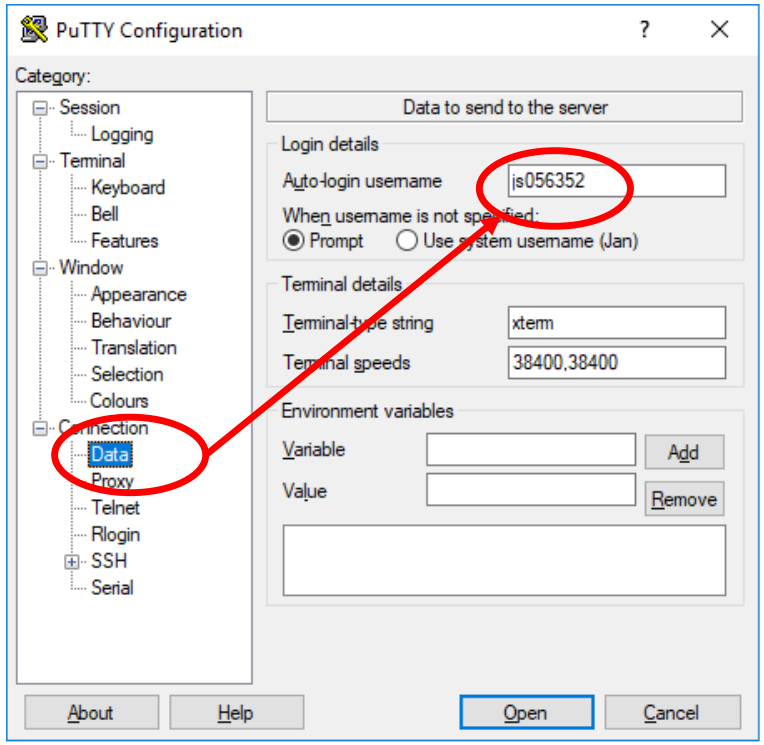
Connecting: Putty configuration

- Extensive options tree
- All discussed options are present
- Create new configuration:
  - Type session name and target
  - Hit Save
  - Configure all other options
  - Go back to this page at the end and hit Save after all changes (easy to forget)



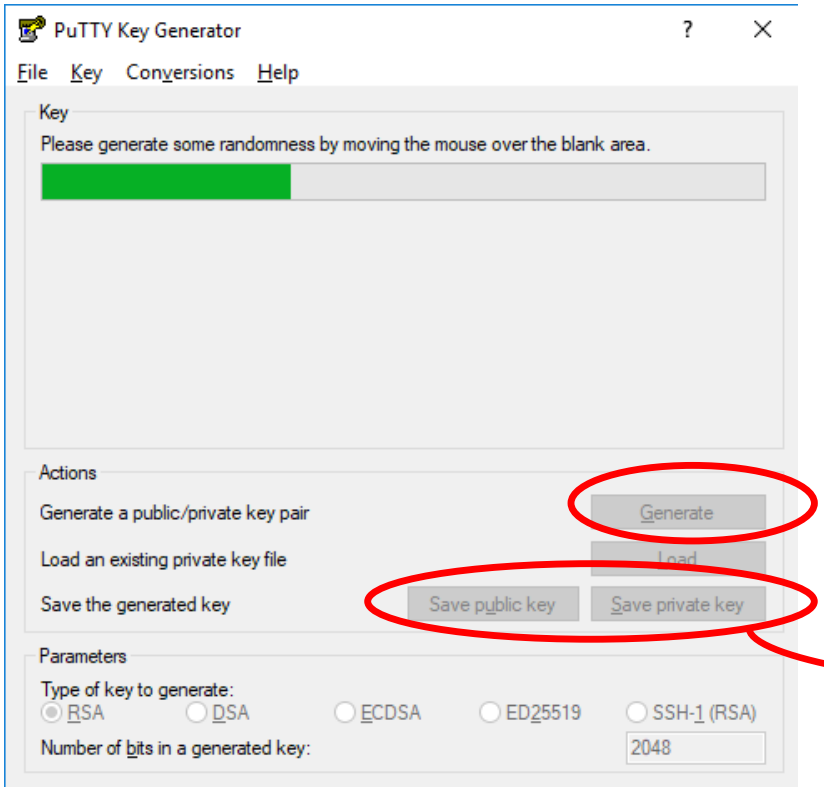
Connecting: Putty configuration

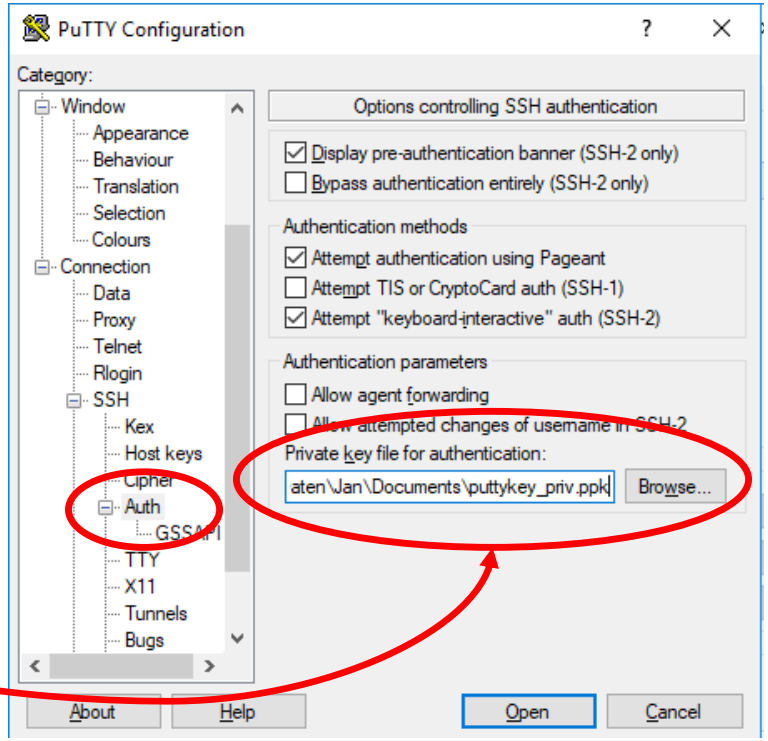




xming needs to be installed and must be started before Putty

Putty Key Generator (also MobaXTerm)

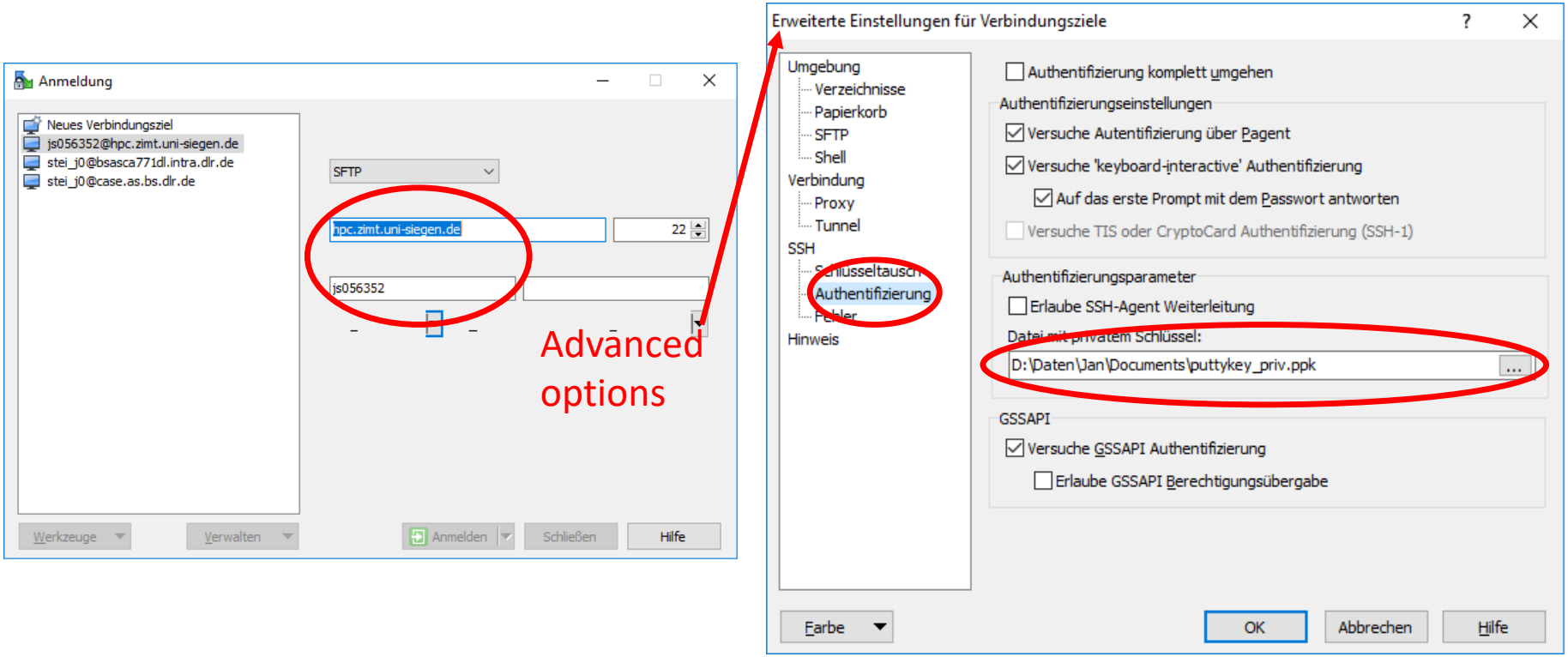




Separate program: Puttygen (installed with Putty)

- Don't forget to paste public key into `authorized_keys` on cluster
- If key fails, enter password

Connecting: WinSCP configuration

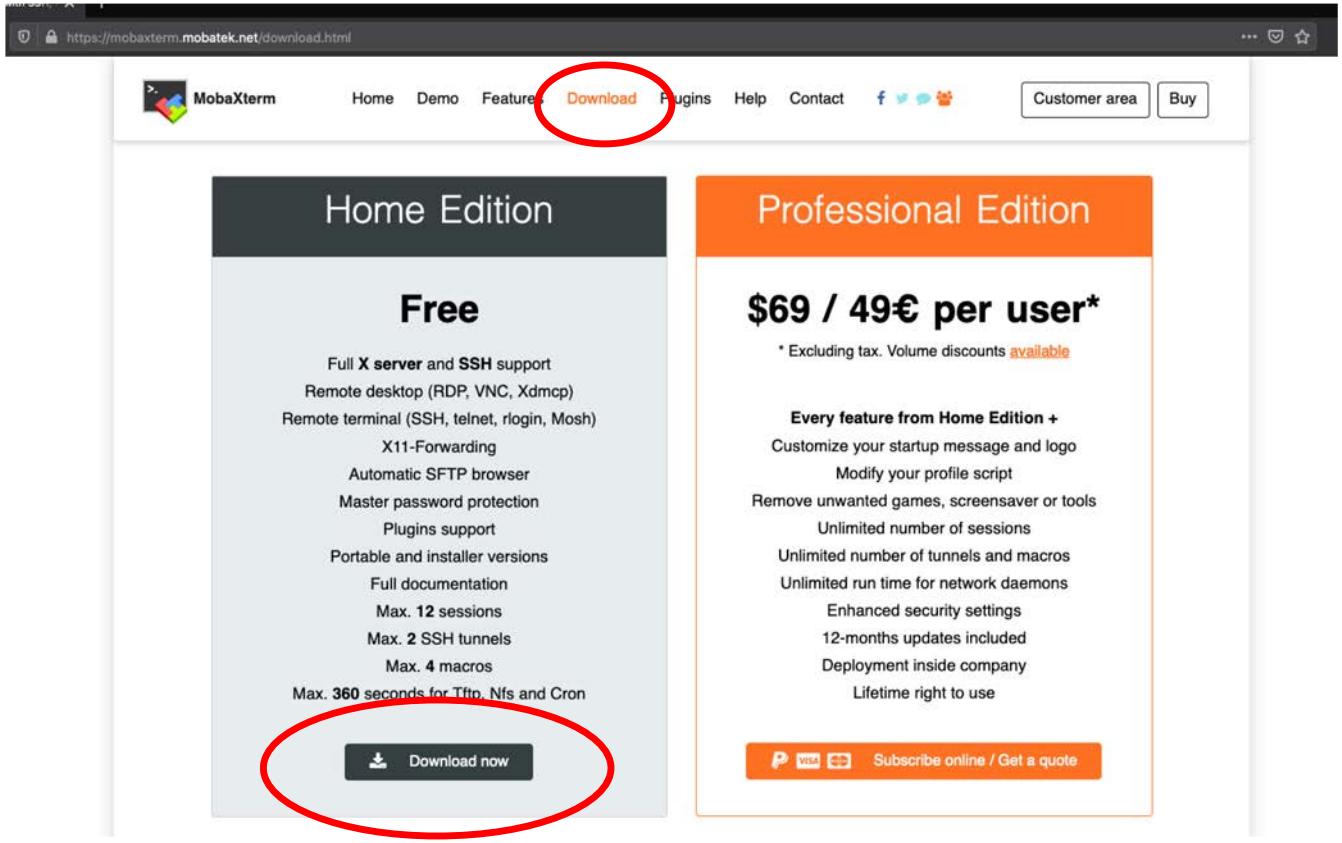


WinSCP basically uses similar options (even the private key can be reused)

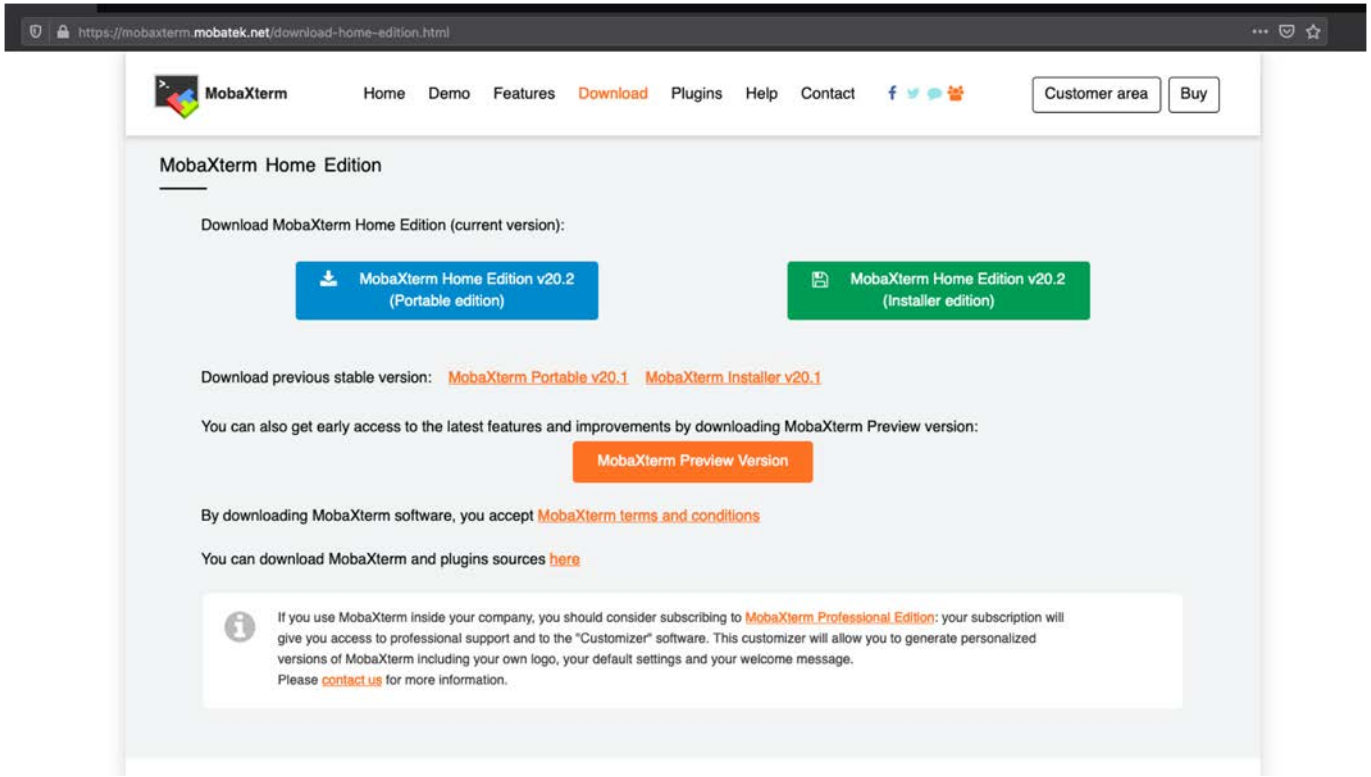
Connecting: MobaXTerm download

- Download MobaXTerm from https://mobaxterm.mobatek.net/
- Free
- Comes in “Installer” and “Portable” versions
  - CIP Pools: download portable version, unzip, run `.exe`
  - Cancel Windows firewall warning, it works anyway
- Windows users will do this in the first exercise

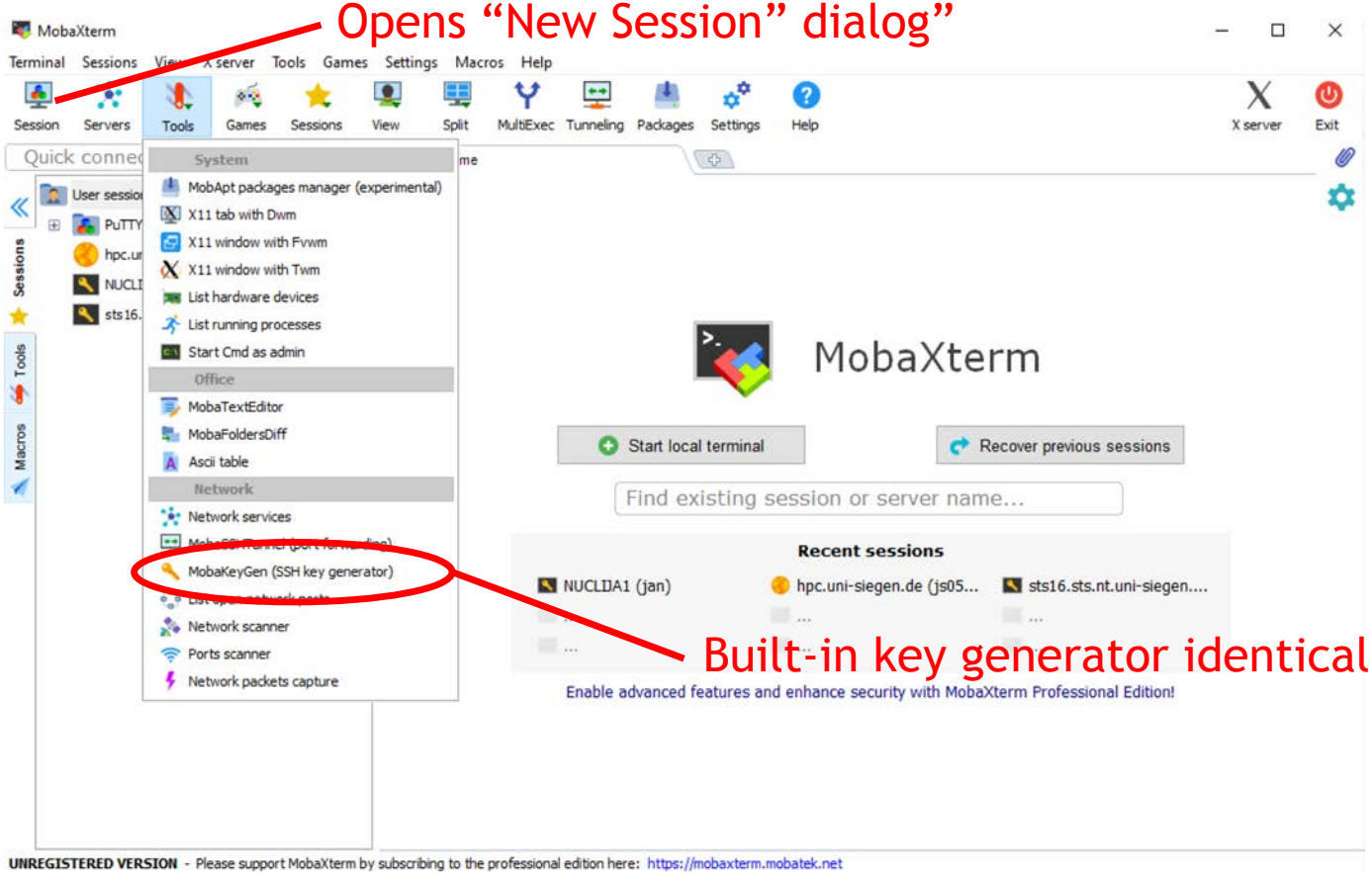
Connecting: MobaXTerm



Connecting: MobaXTerm



Connecting: MobaXTerm

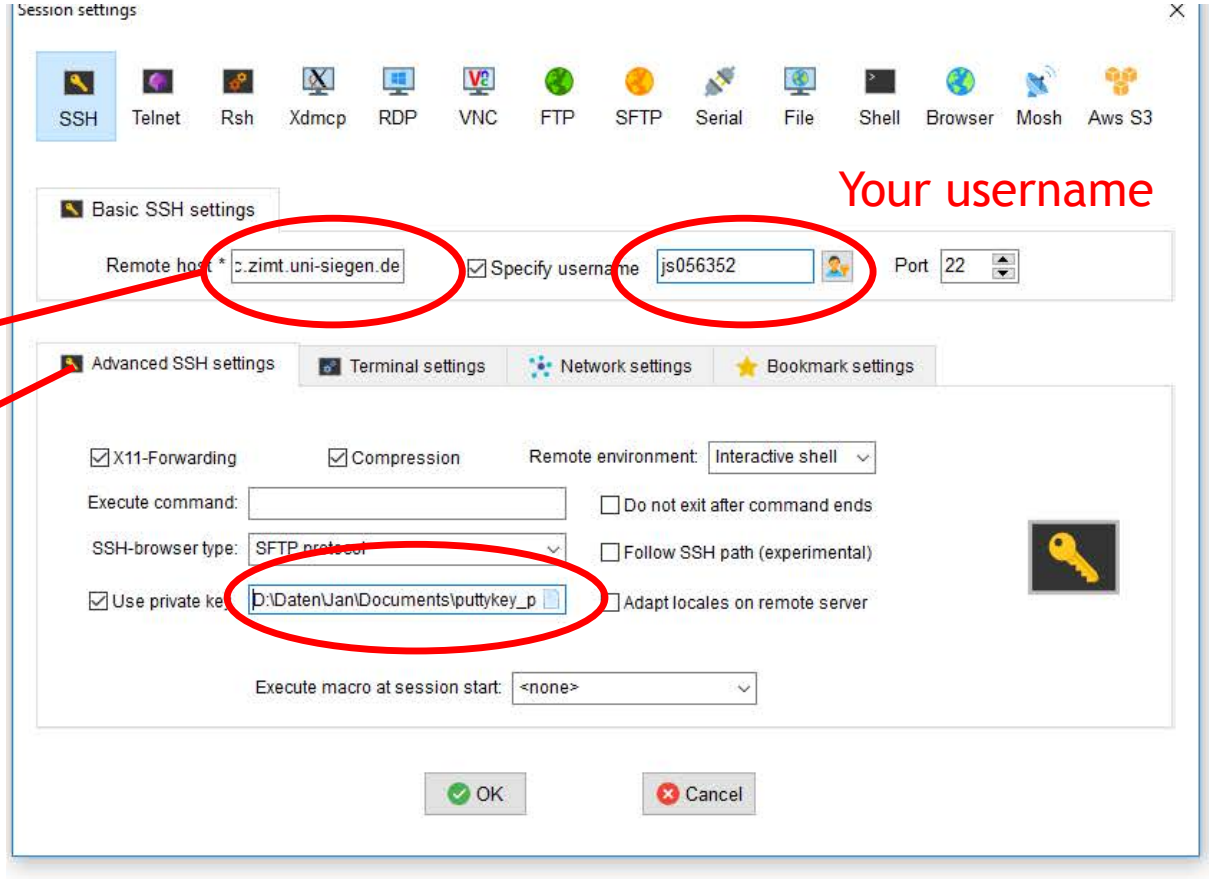


Connecting: MobaXTerm configuration

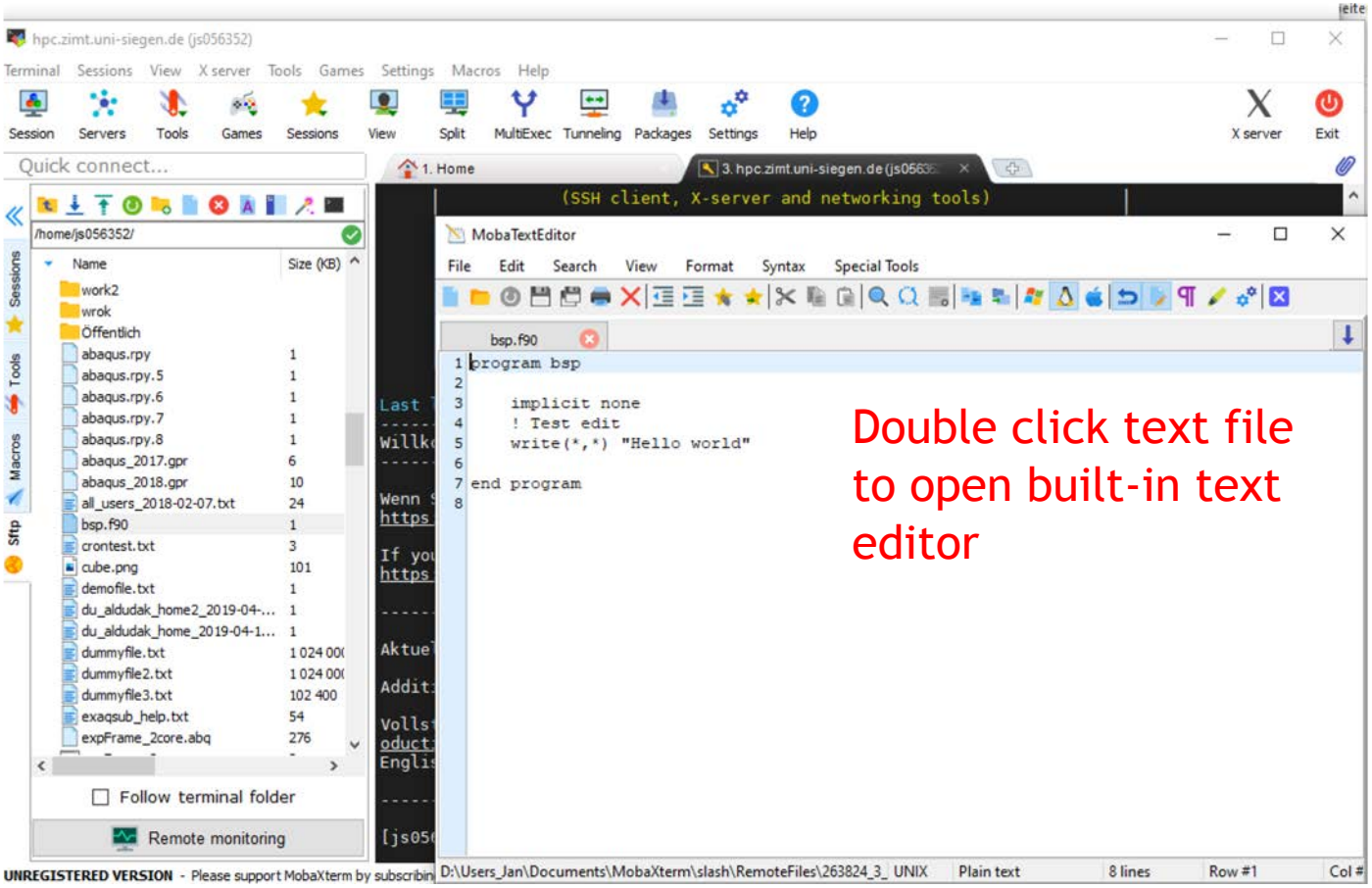
`hpc.zimt.uni-siegen.de`

Most important tab

Same options as before



Connecting: MobaXTerm features



Drag and drop files etc.

Double click text file to open built-in text editor

Outline

1. Getting onto the cluster
   - Structure of a cluster
   - Getting access and help
   - Connecting to the cluster
   - ***Exercise 1: setup, login***
2. Using the cluster
   - Workspaces
   - Environment modules
   - Jobs
   - *Exercise 2: your first job script*
3. SLURM explained
   - Tasks, processes, cores
   - Miscellaneous SLURM stuff
   - *Exercise 3: SLURM options*

Exercise 1

Objectives:

- You understand the basics of SSH
- You have a working cluster login configuration

Login info WiSe 20/21:
User: schulungXY
PW: Cluster##511

(where XY is a number between 01-12, will be assigned during course)

Tasks:

- Set up the SSH/SCP client of your choice
  - X11 forwarding
  - SSH configurations
- Set up password-less login by generating a private/public key pair

Note the following page!

Exercise 1

- If bored, get creative
  - Try copying files back and forth
  - Try launching different applications
  - Find out how to open multiple windows from one terminal
  - Figure out how to get to the other login node
  - Compare `rsync` and `scp`
  - ...

  - You have cheat sheets at the end of your handout

Outline

1. Getting onto the cluster
   - Structure of a cluster
   - Getting access and help
   - Connecting to the cluster
   - *Exercise 1: setup, login*
2. Using the cluster
   - **Workspaces**
   - Environment modules
   - Jobs
   - *Exercise 2: your first job script*
3. SLURM explained
   - Tasks, processes, cores
   - Miscellaneous SLURM stuff
   - *Exercise 3: SLURM options*

Using the cluster

- Key differences to regular PC
  - Home vs. Workspaces
  - Environment modules
  - Parallel programs/libraries
  - Jobs

Workspaces

- `/home` usually limited in size (100 GB in our cluster)

- Workspaces for CFD data
  - Higher bandwidth
  - Unlimited storage (but limited in time)
  - HoRUS: `/work`

- Workspace mechanism: allocate for X days
  - `ws_allocate <name> <days>`
  - `ws_list`
  - `ws_release <name>`

Workspaces

- Additional options:
  - Send e-mail before workspace expires: `ws_send_ical`
  - Generate calendar item
- Maximum duration: 30 days
  - `ws_extend <ws-name> <days>`
  - Can be extended up to 3 times
  - Extensions and remaining time with `ws_list`
- After that, **data is GONE**!
  - Can be rescued by admins for 10 days after that
  - Do not rely on this

Demo 8

Workspaces

- Common problems:
  - Forgetting duration in `ws_allocate <name>` (will result in 1 day duration)
  - Forgetting to renew WS

- Tip: set up your e-mail address
  - Put a file named `.ws_user.conf` in your home directory
  - Inside file: `mail: <Your e-mail address>`
    - Note space after colon (YAML syntax)
  - When creating workspace: `ws_allocate -r <days>`
  - You will get an e-mail <days> before expiration

Outline

1. Getting onto the cluster
   - Structure of a cluster
   - Getting access and help
   - Connecting to the cluster
   - *Exercise 1: setup, login*
2. Using the cluster
   - Workspaces
   - **Environment modules**
   - Jobs
   - *Exercise 2: your first job script*
3. SLURM explained
   - Tasks, processes, cores
   - Miscellaneous SLURM stuff
   - *Exercise 3: SLURM options*

Environment module system

Excursion: what happens in Linux if you type a command?

- Linux looks for program with that name
- Directories where Linux looks: defined by PATH environment variable
  - Directories set by Linux
  - Directories added by installed software (so it gets found)
  - You can add your own
- Goes through in order listed in PATH
  - First hit gets executed

Environment module system

- PATH is called an environment variable
- Other variables set by Linux, e.g.: HOME, USER
- Set by programs to find libraries etc.
- Used by SLURM
  - Special variables inside job
  - Used to provide job inormation
- “Environment” because process sees it, provides it to subprocesses

Environment module system

- Many users with different needs
  - Different versions of same software/library
  - Different software with same commands
- Reconfigure environment for every user?
- Better: modular environment
  - Users load module that they need
- Example:

```
module load PrgEnv/intel-openmpi
module avail
```

Environment module system

- Modules may be loaded as dependency
- Some modules are loaded on login for each cluster user
- `module list` shows loaded modules
- `module purge` unloads everything (e.g. debugging)
- Possible to define own modules (see website)

Demo 9

Outline

1. Getting onto the cluster
   - Structure of a cluster
   - Getting access and help
   - Connecting to the cluster
   - *Exercise 1: setup, login*
2. Using the cluster
   - Workspaces
   - Environment modules
   - **Jobs**
   - *Exercise 2: your first job script*
3. SLURM explained
   - Tasks, processes, cores
   - Miscellaneous SLURM stuff
   - *Exercise 3: SLURM options*

Running computations: jobs

- A single HPC computation is called a **job**

- Job Scheduler SLURM
  - Manages when to run jobs
  - Efficient usage of resources
  - Several commands (each with `-h` for options)

- One job = one command/script
  - Start job: `srun (--pty) <options> <linux-command>`
    `sbatch <options> <scriptname>`
  - Monitor jobs: `squeue`, show partitions: `sinfo`
  - Delete job: `scancel <job-id>`

Running computations: queues

- Jobs are put into queues (called partitions in SLURM)
  - Different runtime
  - Different size
  - Different type of node (e.g. GPU)
- Each queue has default values
- You pick queue, runtime, number of nodes

→ **As many resources as necessary, as few as possible (with safety margin)**

Running computations: queues

There are a few special queues:

- `medium2`:
  - 20 nodes with more RAM
  - Financed by Prof. Carolus (Fluid Mechanics)
  - Available to everyone, but lower priority

- `smp`:
  - single node, 512 GB RAM, 32 cores

- `htc`:
  - HPE Moonshot system

Monitoring jobs

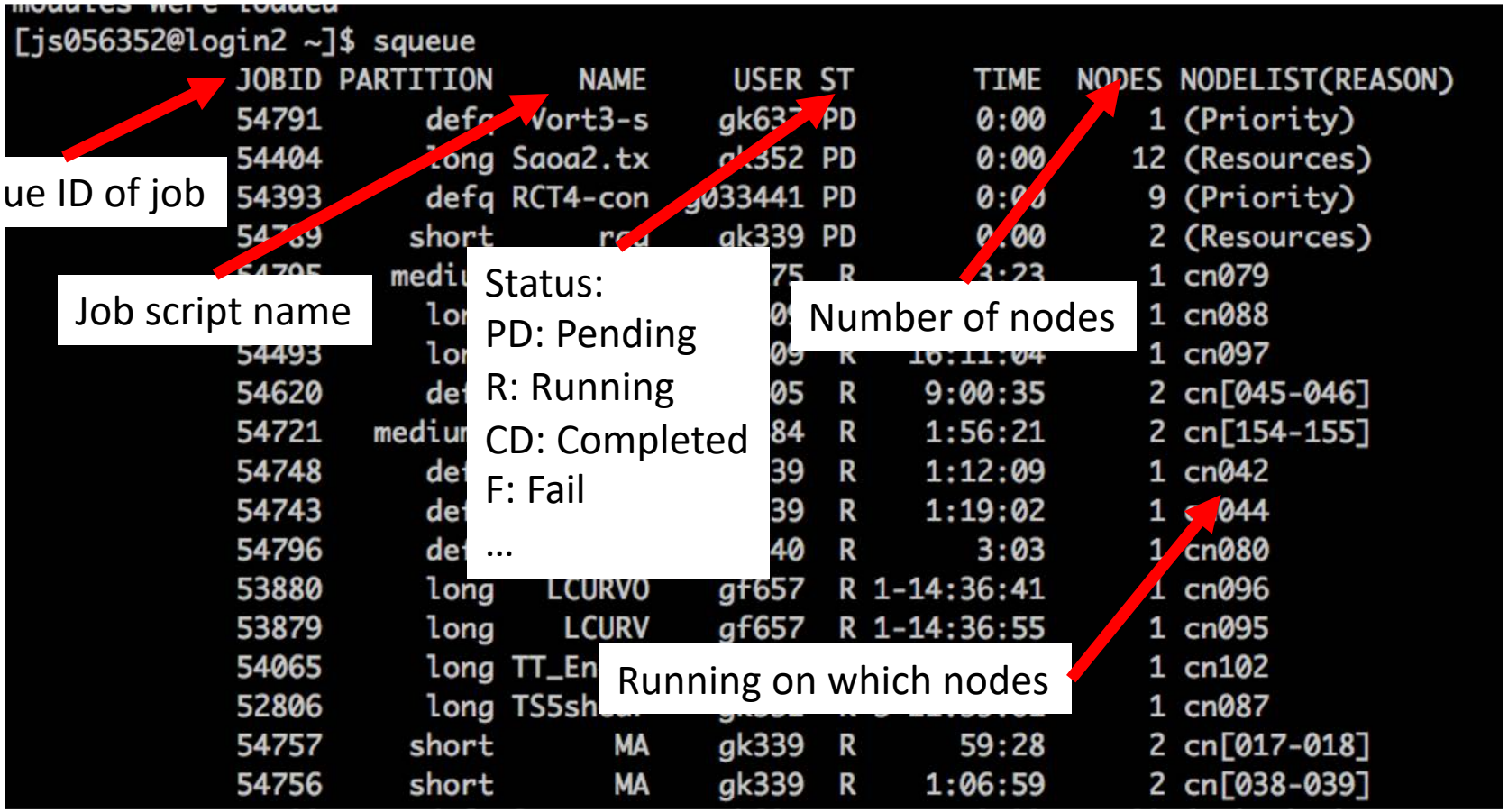
- Check regularly what your job does
  - Your first job will fail (guaranteed)
  - Might be a bug later on
  - Might be a problem with the cluster
  - Might run out of resources
  - Might not be finished when walltime is reached

- Main command to check what your job is doing:

```
squeue
```

- If possible, use checkpointing (write intermediate results)

Monitoring: squeue example



Other SLURM commands

- `srun` can also be used within job
  - Runs command once in every task
  - Warning: scripts need to be executable
- `squeue -u <Your Username>` will list all your jobs
- `sinfo` will list available partitions, `spartition` lists defaults
- `scancel <Job ID>` will kill a job
  - `scancel -u <Your Username>` kills all your jobs
- `scontrol` allows more in-depth information
  - Example: `scontrol show job <Job ID>`

Other key concepts of SLURM

- SLURM allows you to choose how many and which resources to use
  - Nodes
  - RAM
  - Running time

- For now: one task = one program, using one CPU core

Demo 10

Workflow: queuing a job script

1. You write the job script
   - Calls your software
   - Provides job settings
   - Loads environment
   - Any other necessary tasks

1. You prepare your software and files, workspace etc.

3. You queue your script with `sbatch`

4. You wait for job to complete, check intermediate results

Example job script

```
#!/bin/bash
#SBATCH --time=0:20:00
#SBATCH --nodes=1
#SBATCH --tasks-per-node=6
#SBATCH --mem 48000
#SBATCH --partition=short

module load abaqus/2017

echo "Number of tasks: "
echo $SLURM_NTASKS

abq2017hf9 job=Test.inp mp_mode=mpi interactive cpus=$SLURM_NTASKS
```

Example job script

```
#!/bin/bash
#SBATCH --time=0:20:00
#SBATCH --nodes=1
#SBATCH --tasks-per-node=6
#SBATCH --mem 48000
#SBATCH --partition=short

module load abaqus/2017

echo "Number of tasks: "
echo $SLURM_NTASKS

abq2017hf9 job=Test.inp mp_mode=mpi interactive cpus=$SLURM_NTASKS
```

Which shell to use (Linux command)

- At least two different families (csh,bash)
  - Different syntax
- Default on cluster: bash
- Does not have to be shell

Example job script

```
#!/bin/bash
#SBATCH --time=0:20:00
#SBATCH --nodes=1
#SBATCH --tasks-per-node=6
#SBATCH --mem 48000
#SBATCH --partition=short

module load abaqus/2017

echo "Number of tasks: "
echo $SLURM_NTASKS

abq2017hf9 job=Test.inp mp_mode=mpi interactive cpus=$SLURM_NTASKS
```

SLURM settings

- Most important:
  - How many tasks(processes)/nodes
  - Which queue (partition)
  - For how long
  - Different combinations
- Additional settings
  - Defaults exist for most

Example job script

```
#!/bin/bash
#SBATCH --time=0:20:00
#SBATCH --nodes=1
#SBATCH --tasks-per-node=6
#SBATCH --mem 48000
#SBATCH --partition=short

module load abaqus/2017

echo "Number of tasks: "
echo $SLURM_NTASKS

abq2017hf9 job=Test.inp mp_mode=mpi interactive cpus=$SLURM_NTASKS
```


Load environments

- Environment variables are handed over
- But not modules
- Not always necessary

Example job script

```
#!/bin/bash
#SBATCH --time=0:20:00
#SBATCH --nodes=1
#SBATCH --tasks-per-node=6
#SBATCH --mem 48000
#SBATCH --partition=short

module load abaqus/2017

echo "Number of tasks: "
echo $SLURM_NTASKS

abq2017hf9 job=Test.inp mp_mode=mpi interactive cpus=$SLURM_NTASKS
```

Additional tasks

- e.g. `cd <YourWorkDir>`
- Set variables
- Here: print number of tasks to logfile

Example job script

```
#!/bin/bash
#SBATCH --time=0:20:00
#SBATCH --nodes=1
#SBATCH --tasks-per-node=6
#SBATCH --mem 48000
#SBATCH --partition=short

module load abaqus/2017

echo "Number of tasks: "
echo $SLURM_NTASKS

abq2017hf9 job=Test.inp mp_mode=mpi interactive cpus=$SLURM_NTASKS
```

Call your program

- Program settings, parameter files, etc.
- Might be in loop
- Here: called with SLURM-set variable

Using sbatch to queue your job script

Call sbatch command
`sbatch --help` for details

Options override script/default options

```
$ sbatch -p "medium" jobscript.sh
Submitted batch job 54428
```

SLURM will print job ID

Your job script

- Does not need to be executable
- But needs to have `#!/executable` at the top
  - E.g. `#!/bin/bash`

Demo 11

Outline

1. Getting onto the cluster
   - Structure of a cluster
   - Getting access and help
   - Connecting to the cluster
   - *Exercise 1: setup, login*
2. Using the cluster
   - Workspaces
   - Environment modules
   - Jobs
   - ***Exercise 2: your first job script***
3. SLURM explained
   - Tasks, processes, cores
   - Miscellaneous SLURM stuff
   - *Exercise 3: SLURM options*

Exercise 2

Objectives:

1. You know how to create a workspace
2. You know how to create a simple job script
3. You can interpret the output of `squeue` and `sinfo`

Tasks:

- Create a workspace and change to it

- Write a job script that prints its working directory, sleeps for 30 seconds, then exits
  - Remember the cheat sheets
  - You are allowed to google basic Linux commands

Note the following page!

Exercise 2

- If bored, get creative:
  - Use `sinfo` to find out how much of the cluster is currently busy
  - Load and unload modules, use `which` command to see which program is called with a command
  - Try finding out file transfer speeds between your PC, your home directory and your workspace
  - Try `sbatch`-ing a script in a different language
  - …

Solution: job script

```
#!/bin/bash
#SBATCH --time=0:05:00
#SBATCH --tasks=1
#SBATCH --partition=short

# Print directory.
pwd

# Sleep.
sleep 30s
```

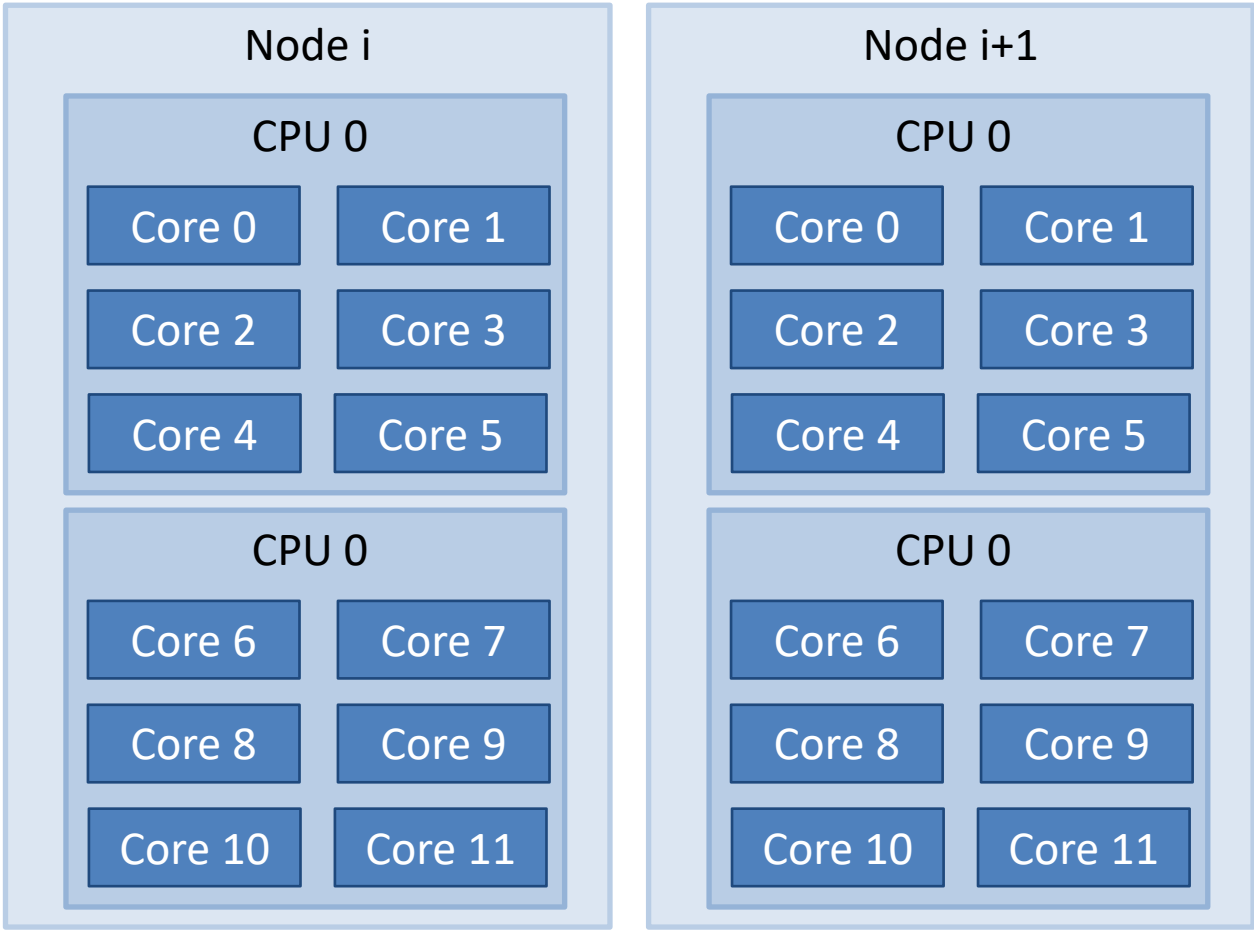
Outline

1. Getting onto the cluster
   - Structure of a cluster
   - Getting access and help
   - Connecting to the cluster
   - *Exercise 1: setup, login*
2. Using the cluster
   - Workspaces
   - Environment modules
   - Jobs
   - *Exercise 2: your first job script*
3. **SLURM explained**
   - **Tasks, processes, cores**
   - Miscellaneous SLURM stuff
   - *Exercise 3: SLURM options*

Hardware visualized

Hardware:

- Cluster has nodes
- Nodes may have multiple CPUs (each on its socket), often 2
  - Not always important which CPU
- CPU has multiple cores

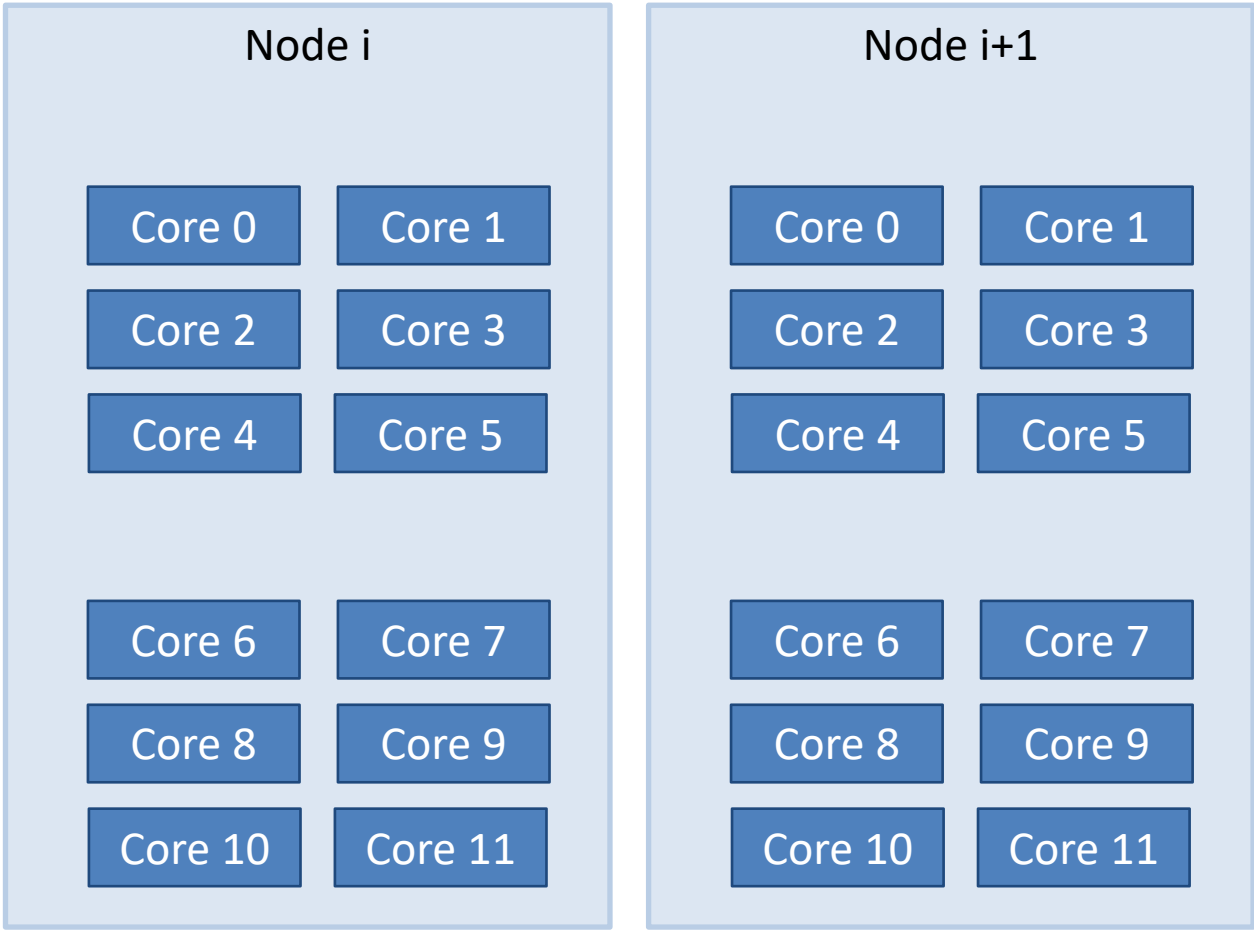


Hardware visualized

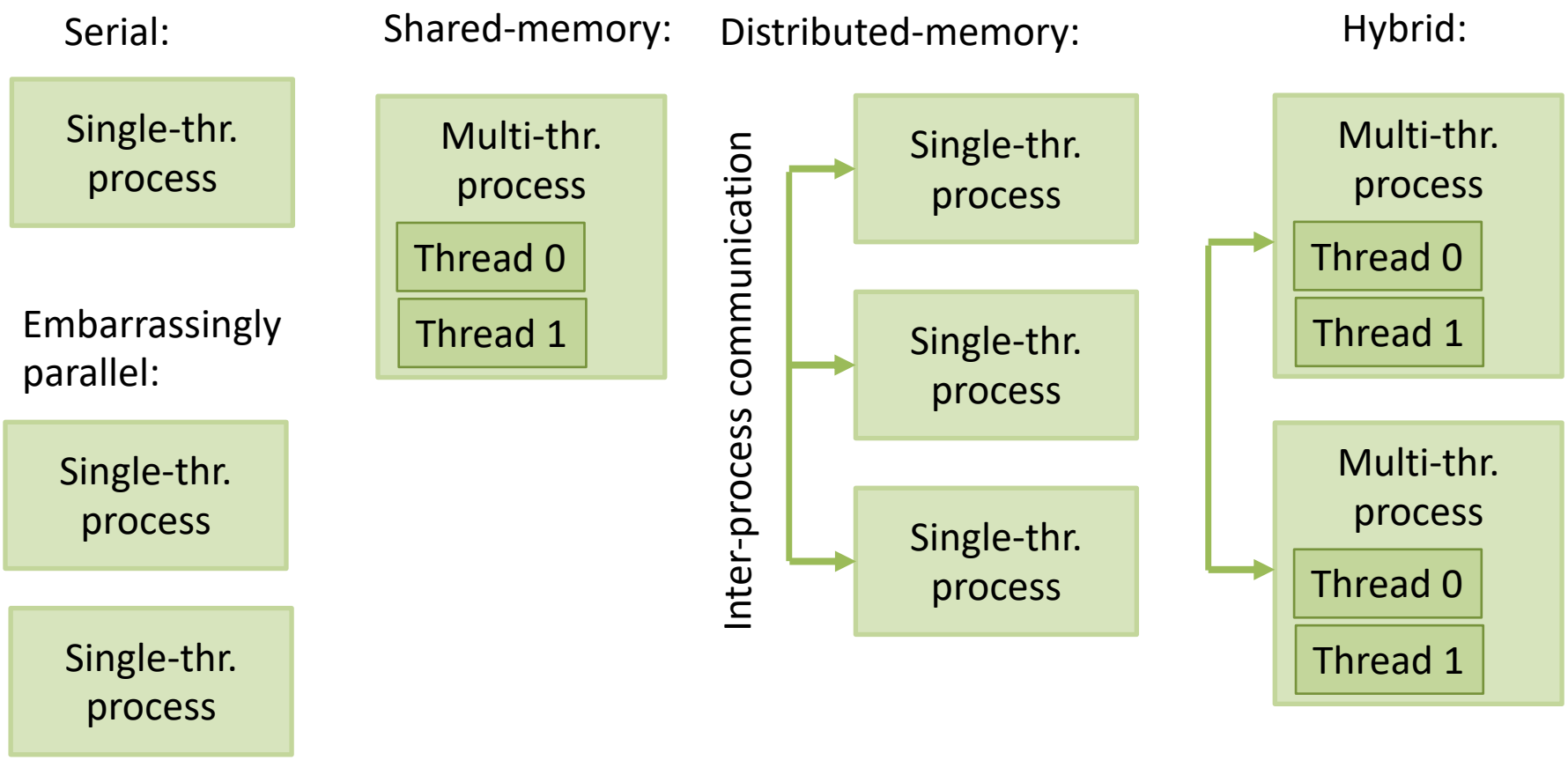
Simplification:

- Difference between CPUs mostly matters for high-performance applications
- Communication between sockets is longer
- Separate caches

→ **Ignored for now**



Workloads common in HPC

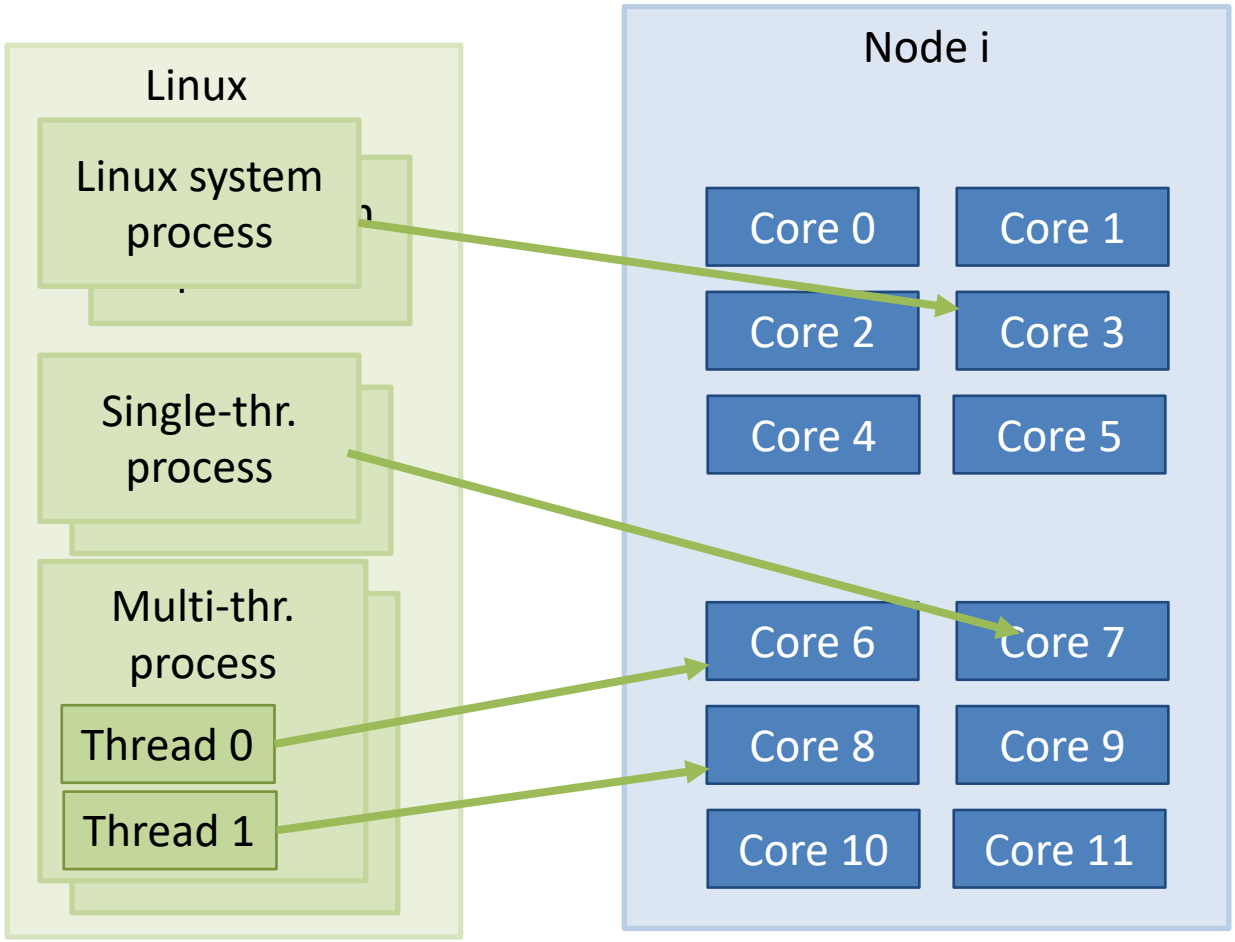


Software visualized

Operating system:

- Each node is a separate computer
- OS runs processes
- Processes may have one or multiple threads

OS decides which process runs on which core(s)



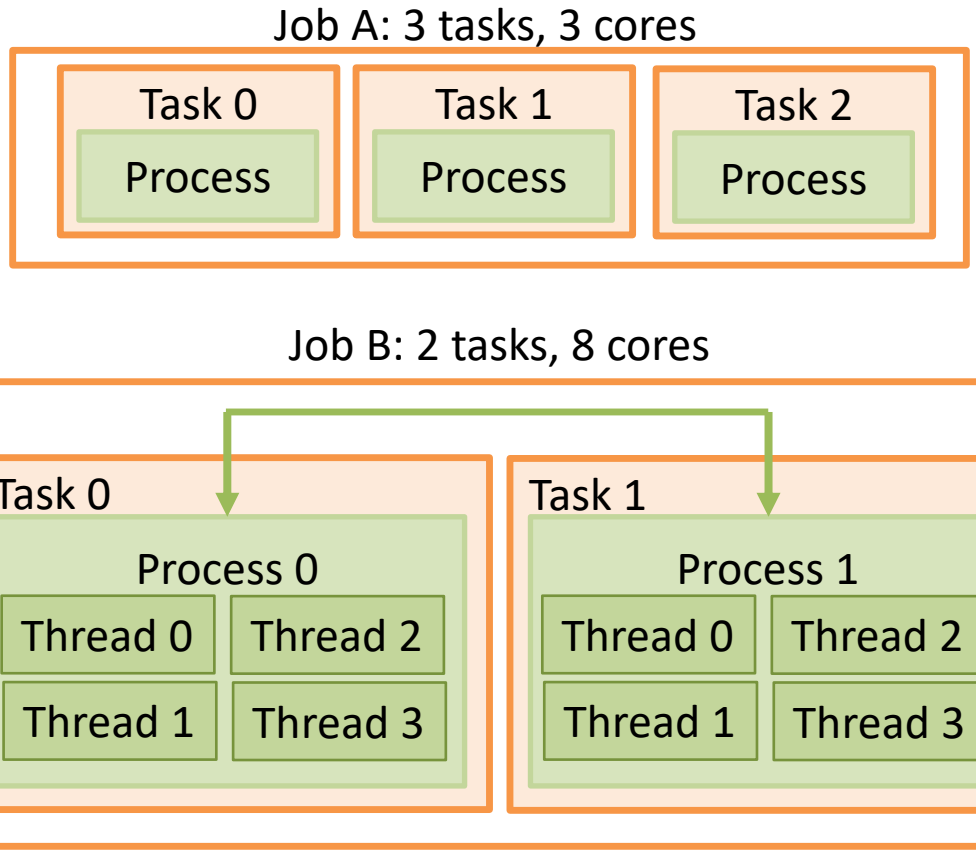
Example SLURM jobs

- User specifies which and how many processes and threads to run in job

→ **Tasks**

What SLURM does:

- Decides on which nodes to run job
- Decides which nodes and processes a job gets
- Distributes tasks



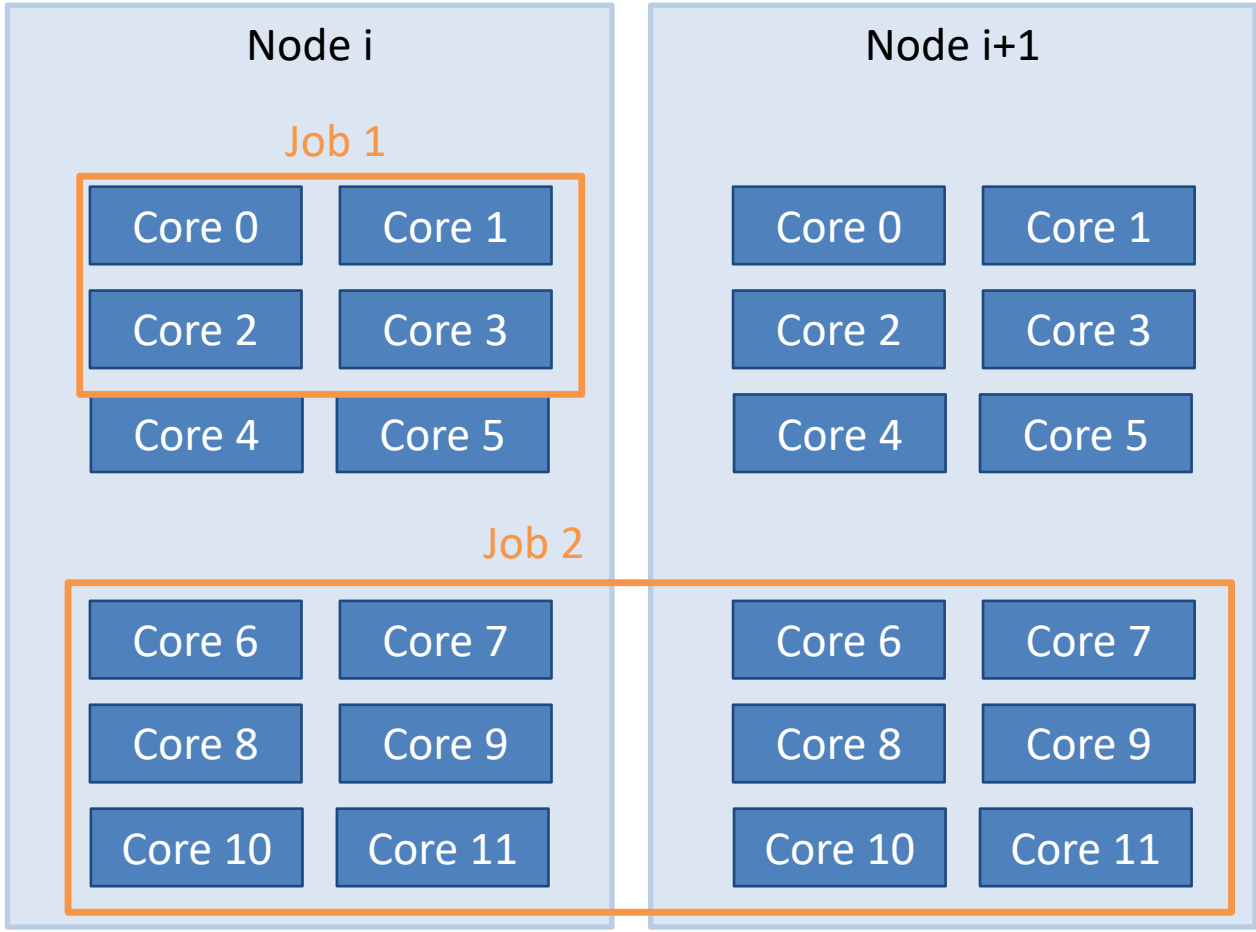
Example SLURM jobs

- User specifies which and how many processes and threads to run in job

→ **Tasks**

What SLURM does:

- Decides on which nodes, cores to run job
- Allocates (reserves) resources
- Launches tasks

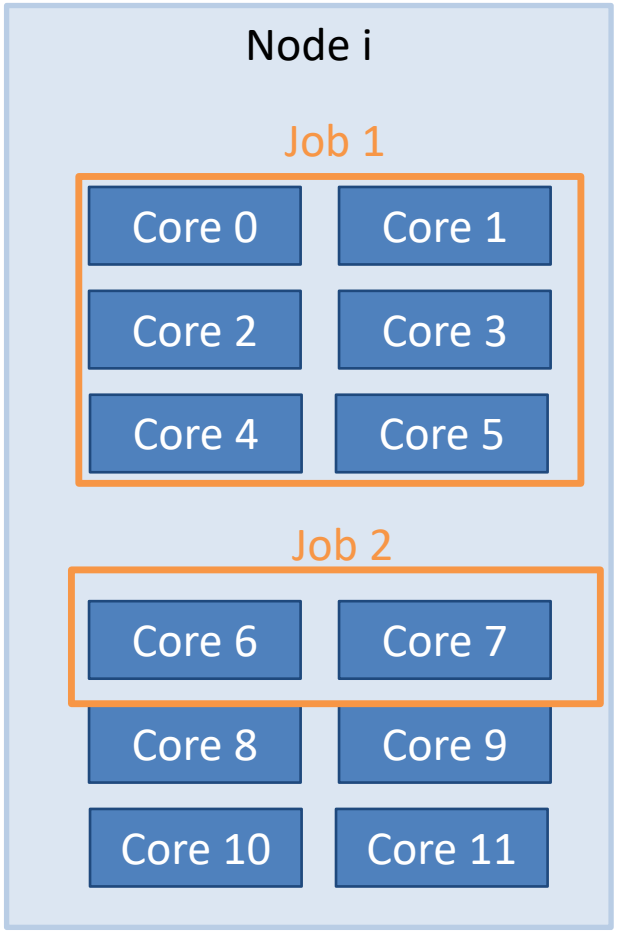


Example SLURM jobs

Complications:

- HoRUS allows two jobs to share a node
  - Unusual for HPC clusters

- Annoying: SLURM refers to “cores” as “CPUs”
  - Again: more important for high performance applications
  - Don’t get confused
  - Term “sockets” used when this is important



What does this mean for you?

- Mostly, you need to decide how many tasks, how many CPUs per task
- Job options:

```
#SBATCH --ntasks=24
```

```
#SBATCH -n 24
```

- Or alternatively:

```
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=12
```

```
#SBATCH -N 24
```

- In both cases possible:

```
#SBATCH --cpus-per-task=4
```

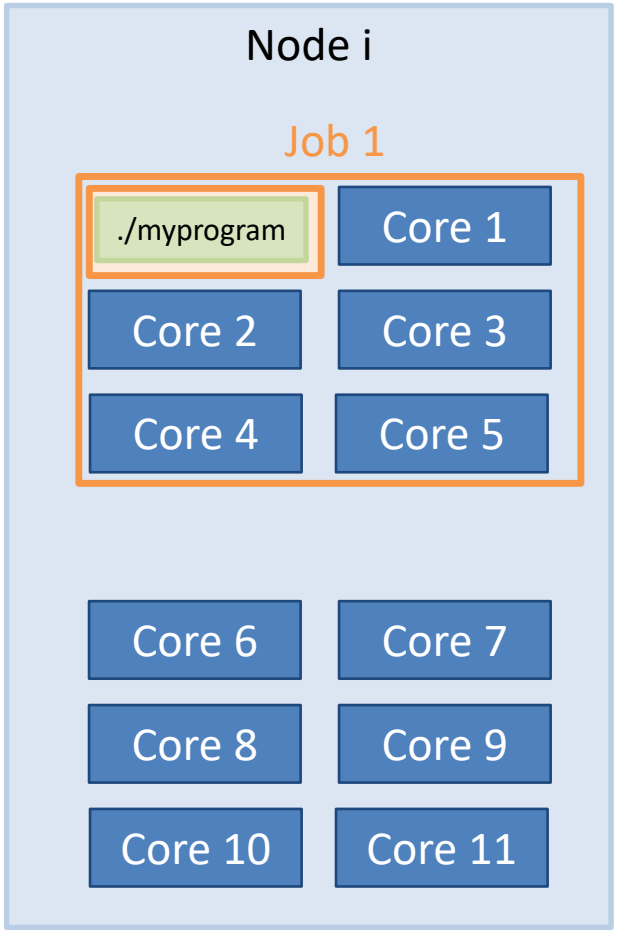
Caution: sometimes starts with n, sometimes doesn't

- Consult `sbatch` documentation

Example job scripts

Simple serial program

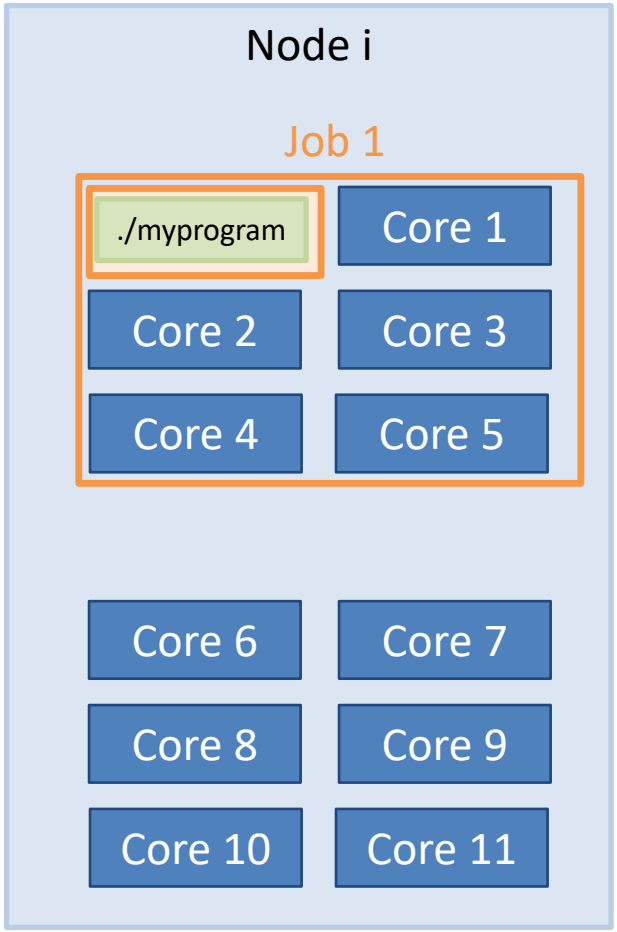
```
#!/bin/bash
#SBATCH --time=0:20:00
#SBATCH --nodes=1
#SBATCH --tasks-per-node=6
#SBATCH --partition=short

./myprogram
```



Example job scripts

Simple serial program

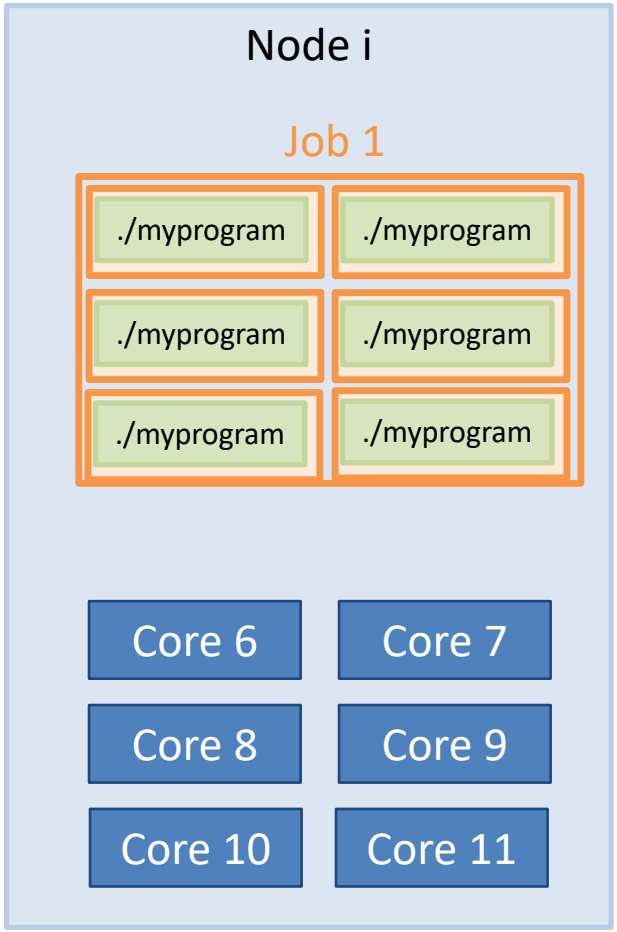
- Simplest possible case
- Typically waste of resources: 6 cores allocated, only 1 task launched
- Might be legitimate use for not using all cores: maybe all the RAM of the node is needed
  - But: always number of tasks = number of processes
  - RAM allocation covered later



Example job scripts

Many instances of serial program

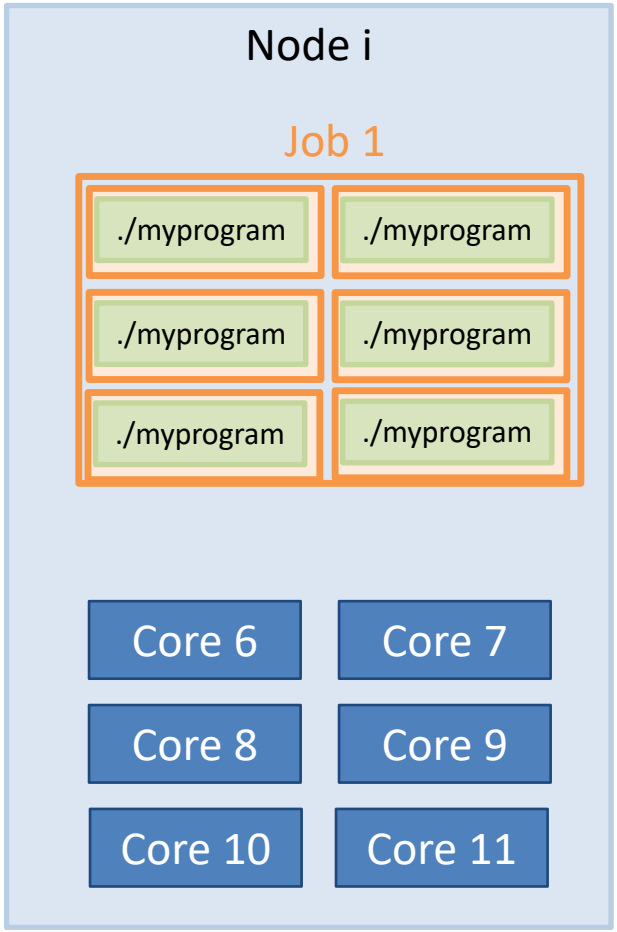
```
#!/bin/bash
#SBATCH --time=0:20:00
#SBATCH --nodes=1
#SBATCH --tasks-per-node=6
#SBATCH --partition=short

srun ./myprogram
```



Example job scripts

Many instances of serial program

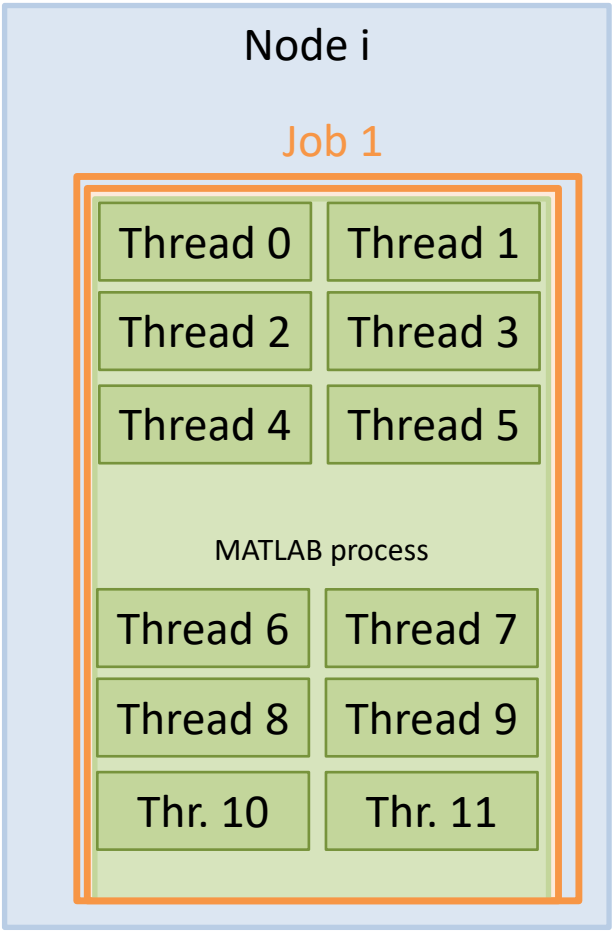
- SLURM simply launches program multiple times
- Cannot talk to each other
- Be careful that they write to different files



Example job scripts

Launching MATLAB

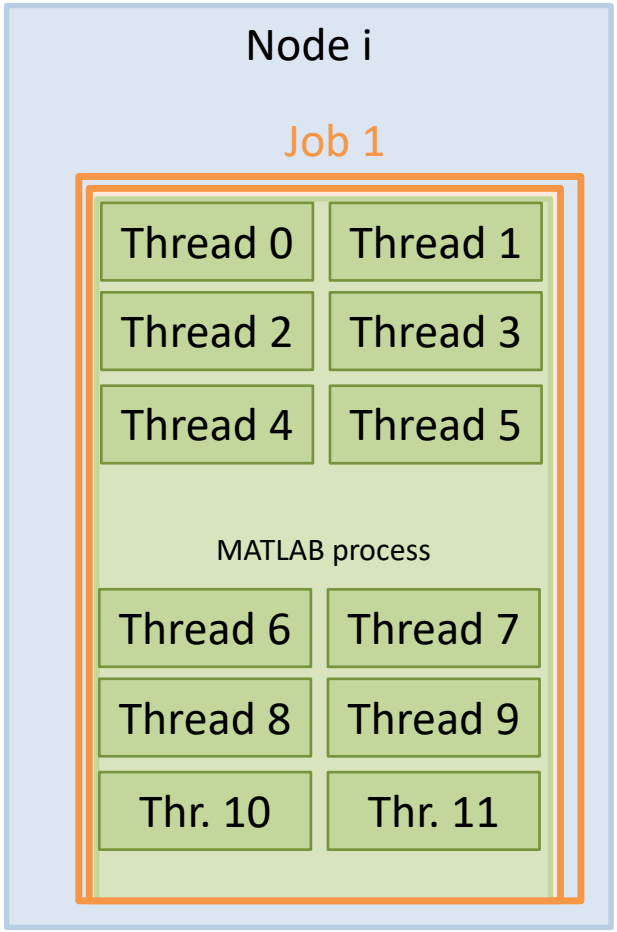
```
#!/bin/bash
#SBATCH --time=0:20:00
#SBATCH --tasks=1
#SBATCH --cpus-per-task=12
#SBATCH --partition=short

matlab -nodisplay -r myscript
```



Example job scripts

Launching MATLAB

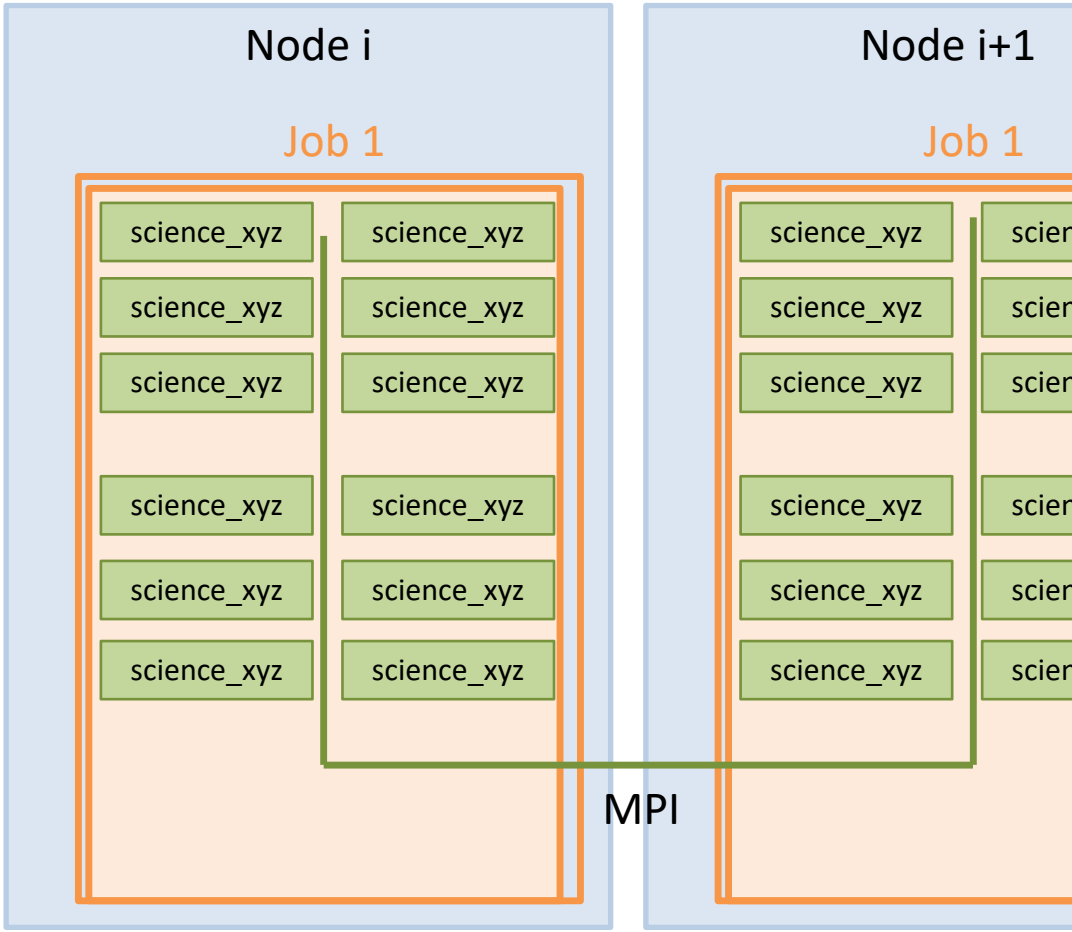
- Commercial applications often multi-threaded
- MATLAB: you do not even need to program differently
- Often good at auto-detecting which resources they have been given
- Does not use MATLAB advanced parallel features (pools), not covered



Example job scripts

Launching scientific application

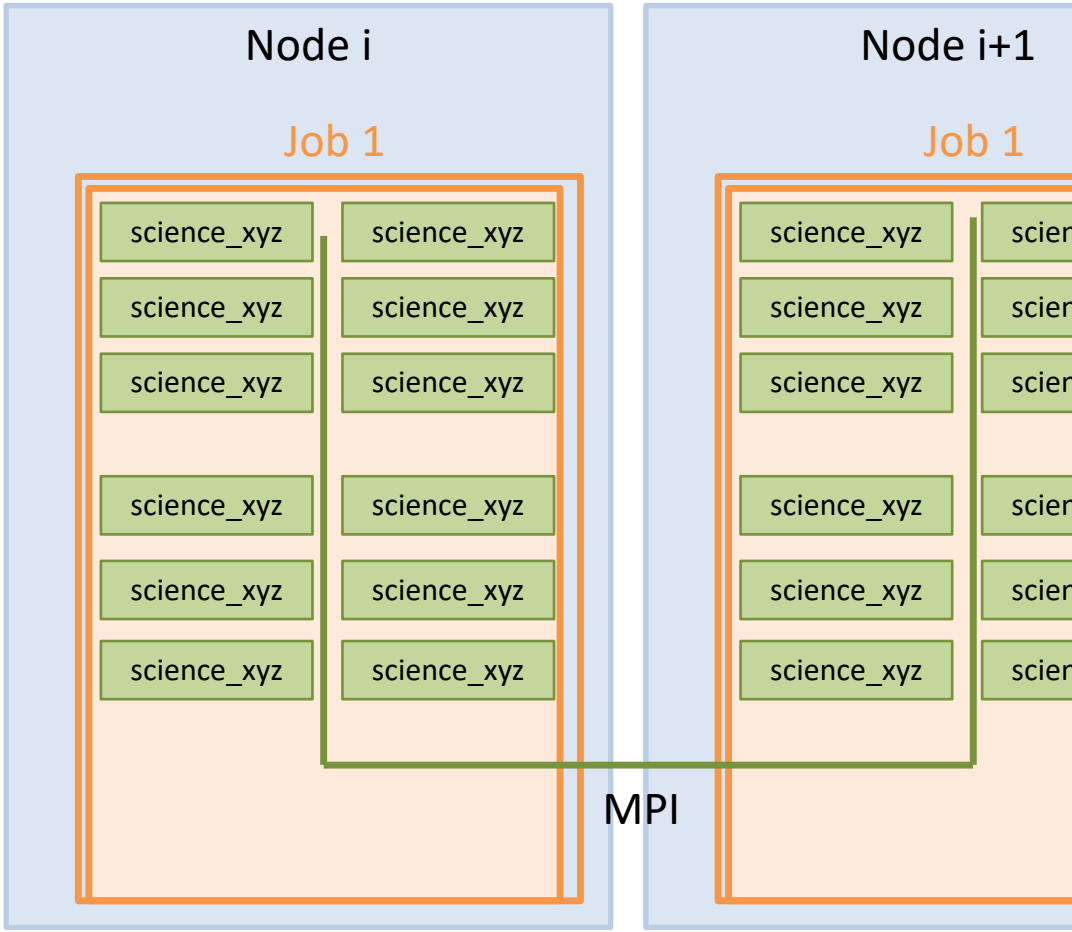
```
#!/bin/bash
#SBATCH --time=4:00:00
#SBATCH --nodes=20
#SBATCH -tasks-per-node=12

mpirun -np 240 science_xyz
parafile >log.txt
```



Example job scripts

Launching scientific application

- Often lots of nodes
- Often distrib.-mem.
  - Especially MPI
- Launch with `mpirun -np [N]`
- SLURM and MPI can generally talk to each other
- Documentation: “can also use `srun` instead of `mpirun`”
  - Does not work on HorUS
- Commercial: sometimes use internal MPI



Outline

1. Getting onto the cluster
   - Structure of a cluster
   - Getting access and help
   - Connecting to the cluster
   - *Exercise 1: setup, login*
2. Using the cluster
   - Workspaces
   - Environment modules
   - Jobs
   - *Exercise 2: your first job script*
3. SLURM explained
   - Tasks, processes, cores
   - **Miscellaneous SLURM stuff**
   - *Exercise 3: SLURM options*

SLURM defaults

- SLURM has defaults for most options
  - Default queue (partition)
  - Default runtime: `spartition` command to show
  - Default task setup (caution)
    - 1 task, 1 CPU per task

- Find out what your job actually ran with:
  - `scontrol show job <Job ID>`
  - Does not work for crashed jobs on HorUS

- SLURM config readable to everyone: `scontrol show conf`

Demo 12

SLURM: resources other than CPU cores

- So far we have only talked about CPUs
- SLURM does the same kind of management for RAM
- Default: 4 GB RAM per task
  - Reason: 12 cores, 48 GB RAM total
  - Option `--mem 48000` to allocate all RAM on node
  - Alternative: `--mem-per-cpu <amount in MB>`
- Again, `scontrol show job` is your friend here

More about the srun command

- `srun` command has multiple uses

- Inside a job:
  - launch a process for every SLURM task

- Outside of job:
  - launch a job running one Linux command (as opposed to a script)
  - launch interactive job

The srun command and interactive jobs

Scenario:

- You want to use a CPU-intensive application yourself
  - e.g. visualization, post-processing
- Will slow down/block entire login node
  - We reserve the right to kill processes
- You cannot start a batch job because application needs your input

Solution:

- Interactive job
  - Resources allocated like any other job
  - But only a console is opened and you can work within it

The srun command and interactive jobs

Interactive job:

1. Use `srun`, not `sbatch`
2. Use `--pty` option
3. Use other SLURM options as needed
4. Specify which command (typically `/bin/bash`)
5. Wait for job to start (console stuck, then it opens)

```
srun --pty -t 5:00:00 /bin/bash
```

Demo 13

SLURM: job priorities

- How does SLURM decide when your job runs?
  - Setup such that people do not have to wait too long
  - `sprio` command
- Priority for each job
  - How long has it been waiting? (16%)
  - How many core-hours has the user recently used? (80%)
  - Bigger job are slightly preferred (4%)
- Additionally: “backfill” mechanism
  - Plays “Tetris”, fits small jobs onto free nodes

Demo 14

Miscellaneous SLURM info

- Job arrays: multiple identical jobs
  - Grouped, don't pollute queue
  - Max 200 jobs per user on HorUS
- If you see "accounts" mentioned, not used on HorUS
- Remember environment variables inside job
- When in doubt: SLURM documentation is quite extensive
  - Many more options: run on specific nodes etc.

Outline

1. Getting onto the cluster
   - Structure of a cluster
   - Getting access and help
   - Connecting to the cluster
   - *Exercise 1: setup, login*
2. Using the cluster
   - Workspaces
   - Environment modules
   - Jobs
   - *Exercise 2: your first job script*
3. SLURM explained
   - Tasks, processes, cores
   - Miscellaneous SLURM stuff
   - ***Exercise 3: SLURM options***

Exercise 3

Objectives:

1. You understand the differences between SLURM task options
2. You can interpret `scontrol show job` output

Tasks:

- Take your job script from earlier, try different combinations of parameters
  - Also leave out parameters
  - Remember the cheat sheets + Google + SLURM documentation
- Check what your job actually did with `squeue` and `scontrol show job`

Note the following page!

Exercise 3

- If bored, get creative:
  - Find out how to queue a job with `srun` instead of `sbatch`
  - Look into SLURM conf file with `scontrol show conf`. What do you recognize/not recognize?
  - Google SLURM job parameters that you do not recognize
  - Find out what happens if you try impossible parameters (e.g. 50 CPUs on one node)
  - …

Thank you for your attention

Questions?