Introduction to the OMNI cluster

Jan Steiner
Zentrum für Informations- und Medientechnik

February 15/16, 2022

A word about Zoom

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

Who am I

- Jan Steiner
  - Aerospace Engineering, Uni Stuttgart (grad. 2010)
  - German Aerospace Center Braunschweig (fluid dynamics)
  - 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

Round of introductions

- What department/institute are you with?
- What is your field / research topic?
- How do you use / intend to use the cluster?
- What is your previous experience?
- Is there something specific you want to learn today?

Outline

1. Getting onto the cluster
   - Structure of a cluster
   - Getting access and help
   - Connecting to the cluster

     About 90 minutes
   - *Exercise 1: setup, login*

     About 30 minutes
2. Using the cluster
   - Workspaces
   - Environment modules
   - Jobs
   - *Exercise 2: your first job script*

     Day 1/day 2 cut roughly here
3. SLURM explained
   - Tasks, processes, cores
   - Miscellaneous SLURM stuff
   - *Exercise 3: SLURM options*

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*

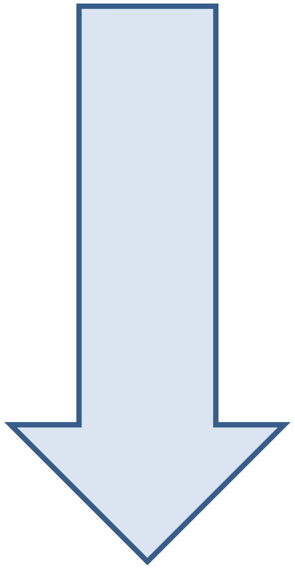
Background

- 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)

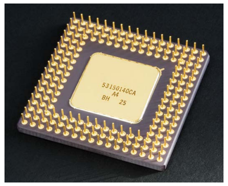
→ **Use more computers**

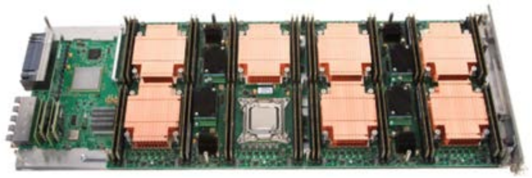
- <u>Cluster</u> of computers
  - Components similar to PC
  - But many, and interconnected

Physical structure of a cluster



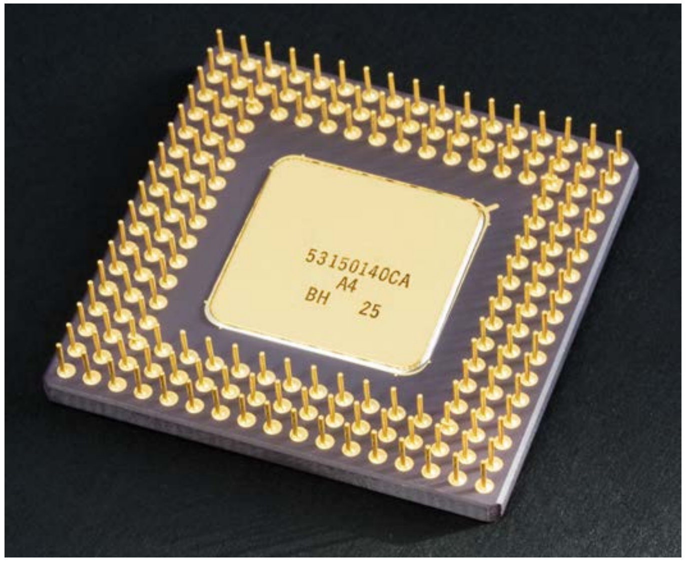
- Core (Processor)
- Node (Blade)
- Rack (Cabinet, Chassis)
- Cluster (Supercomputer)





Cores

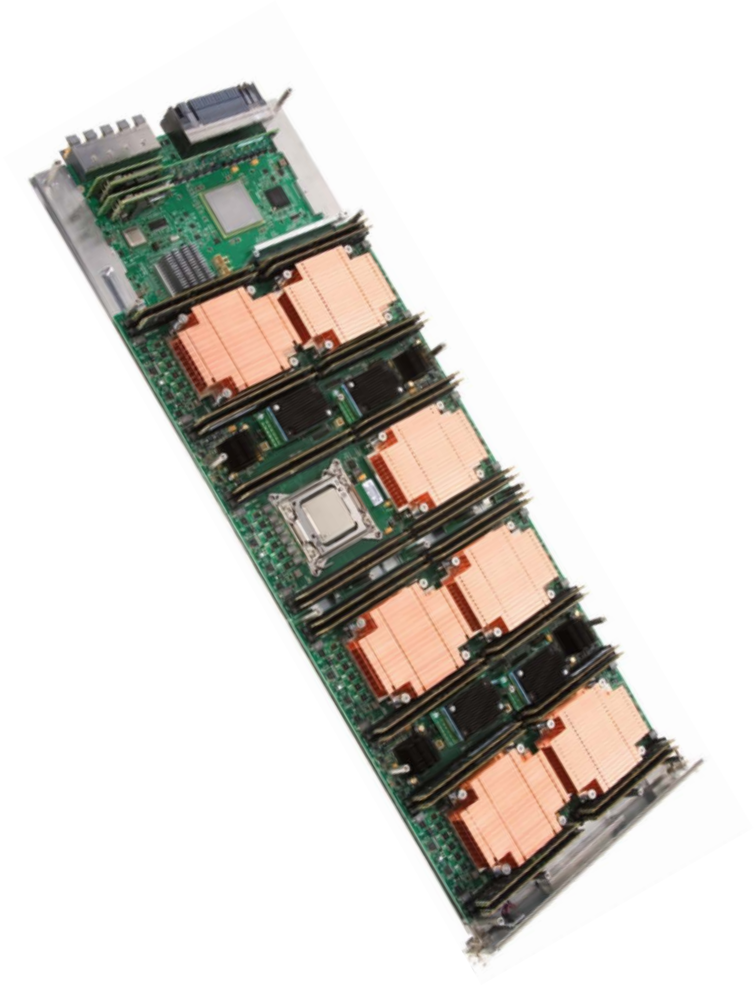
- (Almost) identical to PC processors
- General purpose
- Hyperthreading (two cores in one): disabled on clusters
- Sometimes specialized
  - E.g. graphics processors (GPU)
  - Limited operations, but faster



Source: Wikimedia Commons

Nodes

- Similar to PC motherboards
- 2-4 CPUs, each with many cores
- Usually central RAM
  - OMNI: 256 GB
- Types
  - Compute, Login, Management
  - “Fat” (more RAM), GPU
    - `smp1`: 1536 GB RAM



Cabinet

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

Cluster

- Multiple cabinets
  - OMNI: 9 cabinets, ~550 nodes, 29000 cores
- Infrastructure (e.g. fire suppression)
- Central file storage (hard disks)
  - Sometimes individual nodes have hard disks

Situation at Uni Siegen

- Current: multiple systems
  - OMNI cluster
  - HPE Moonshot (HTC nodes)
  - NEC Aurora vector computer
  - ARM cluster

- Main cluster: OMNI
  - Since early 2021
  - 3-4 times more regular CPUs than previous HoRUS cluster
  - nVIDIA GPUs

OMNI cluster hardware

- 434 regular compute nodes
  - `hpc-node001-hpc-node136`
  - 2x32 AMD EPYC Rome CPUs, 256 GB RAM each
- 2 SMP (Shared Multiprocessing) nodes
  - `smp-node001/002`
  - 64 CPUs, 1536 GB RAM each
- 10 GPU nodes with total of 24 GPUs
  - `gpu-node001-010`
  - NVIDIA Tesla V100
  - 4x1, 2x2, 4x4 GPUs

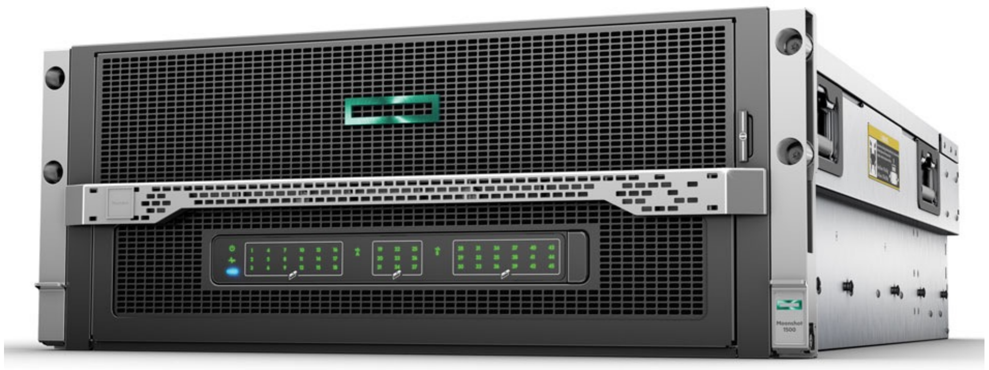
Sources: Wikimedia Commons

OMNI cluster hardware

- 4 login nodes `hpc-login01-04`
  - Identical to compute nodes except 512 GB RAM
- 2 management nodes (not accessible to users)
- Around 500 TB total hard drive space
  - Additionally 32 TB RAM SSDs ("Burst Buffer")
- Various network components etc…

Situation at Uni Siegen

- HPE Moonshot HTC System
  - 45 nodes (2x login, rest compute)
    - 8 CPUs, 64 GB RAM each
  - Designations: `htc001-htc007`
  - Shares homes with OMNI
  - High-Throughput Computing:
    - Smaller jobs, but more

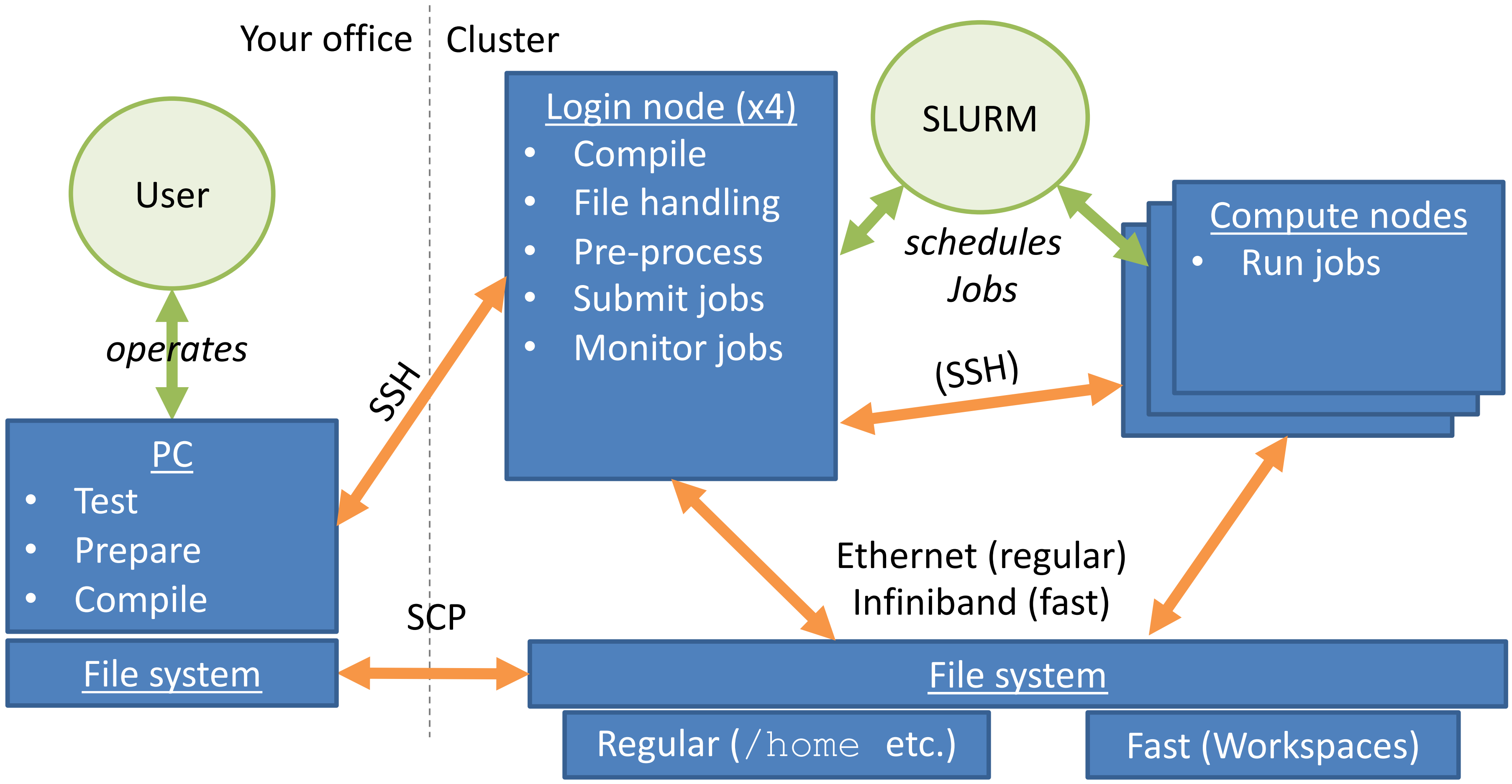

Source: hpe.com

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
  - Names: `vec01-vec02`

Source: nec.com

Logical structure of a cluster



Outline

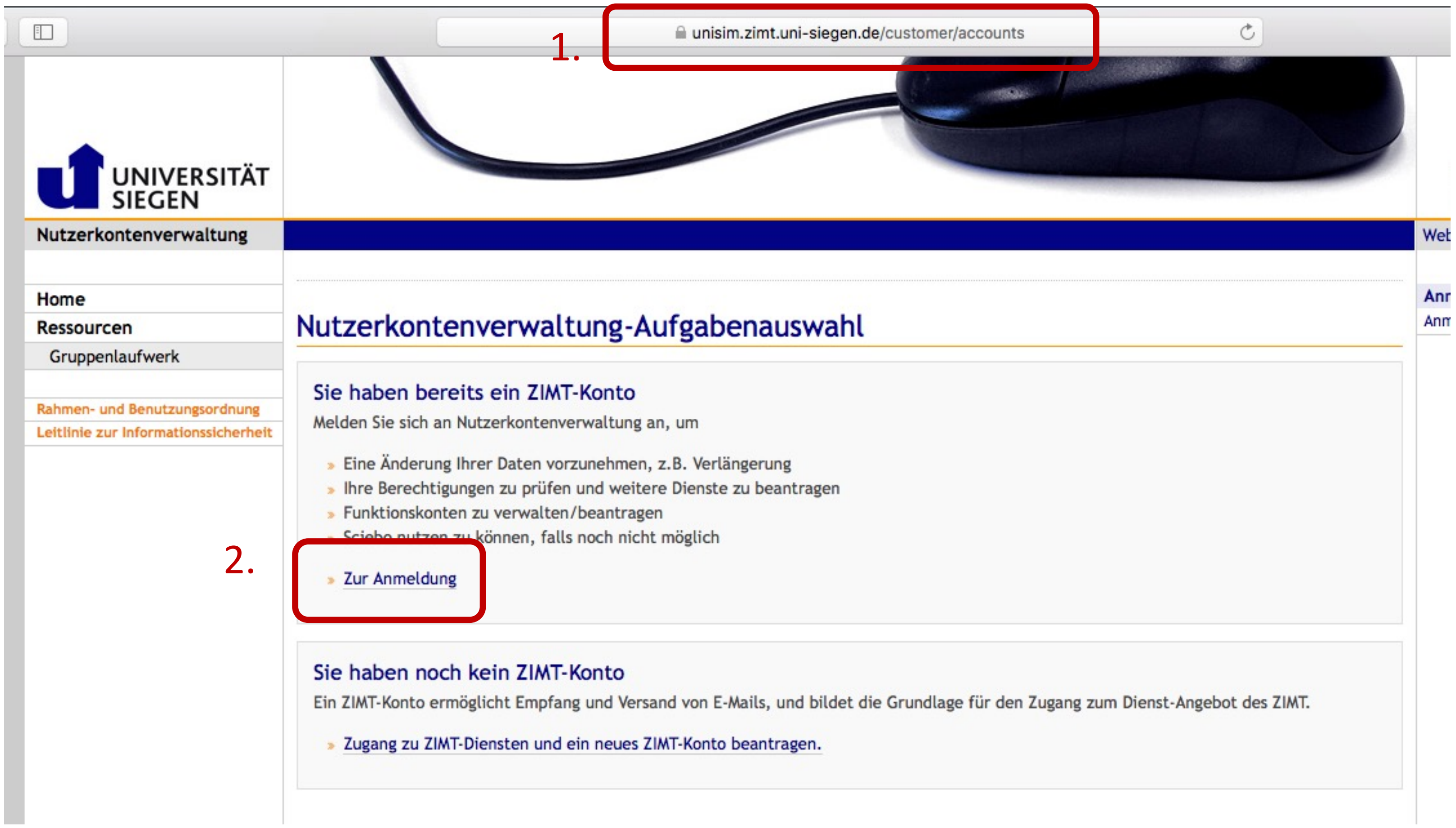
1. Getting onto the cluster
   - Structure of a cluster
   - **Getting access and help**
   - Connecting to the cluster
   - *Exercise 1: setup, login*
2. Using the cluster
   - Workspaces
   - Environment modules
   - Jobs
   - *Exercise 2: your first job script*
3. SLURM explained
   - Tasks, processes,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

Getting cluster access

- Register for cluster access
  - Employees: Nutzerkontenverwaltung
  - Students: need an employee supervisor
  - Valid for all our systems

- Set up an SSH connection
  - Explained in a moment

Registering an employee



1.

unisim.zimt.uni-siegen.de/customer/accounts

UNIVERSITÄT SIEGEN

Nutzerkontenverwaltung

- Home
- Ressourcen
  - Gruppenlaufwerk
- Rahmen- und Benutzungsordnung
- Leitlinie zur Informationssicherheit

Nutzerkontenverwaltung-Aufgabenauswahl

Sie haben bereits ein ZIMT-Konto

Melden Sie sich an Nutzerkontenverwaltung an, um

- Eine Änderung Ihrer Daten vorzunehmen, z.B. Verlängerung
- Ihre Berechtigungen zu prüfen und weitere Dienste zu beantragen
- Funktionskonten zu verwalten/beantragen
- Sciebo nutzen zu können, falls noch nicht möglich

2.

- Zur Anmeldung

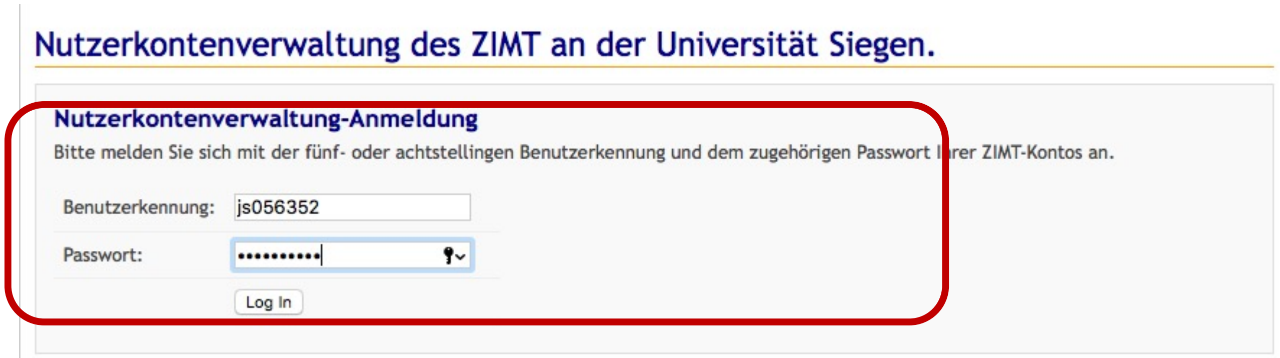
Sie haben noch kein ZIMT-Konto

Ein ZIMT-Konto ermöglicht Empfang und Versand von E-Mails, und bildet die Grundlage für den Zugang zum Dienst-Angebot des ZIMT.

- Zugang zu ZIMT-Diensten und ein neues ZIMT-Konto beantragen.

Registering an employee

3.



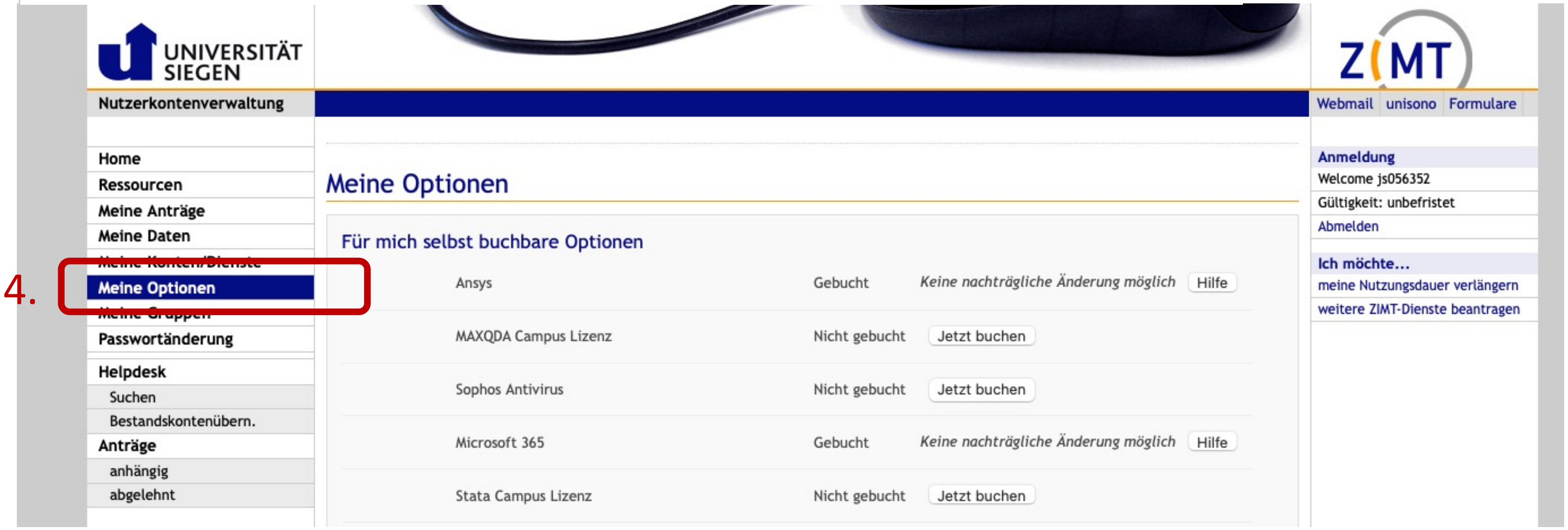
Nutzerkontenverwaltung des ZIMT an der Universität Siegen.

Nutzerkontenverwaltung-Anmeldung

Bitte melden Sie sich mit der fünf- oder achtstelligen Benutzerkennung und dem zugehörigen Passwort Ihrer ZIMT-Kontos an.

Benutzerkennung:	js056352
Passwort:	••••••••••

4.



Meine Optionen

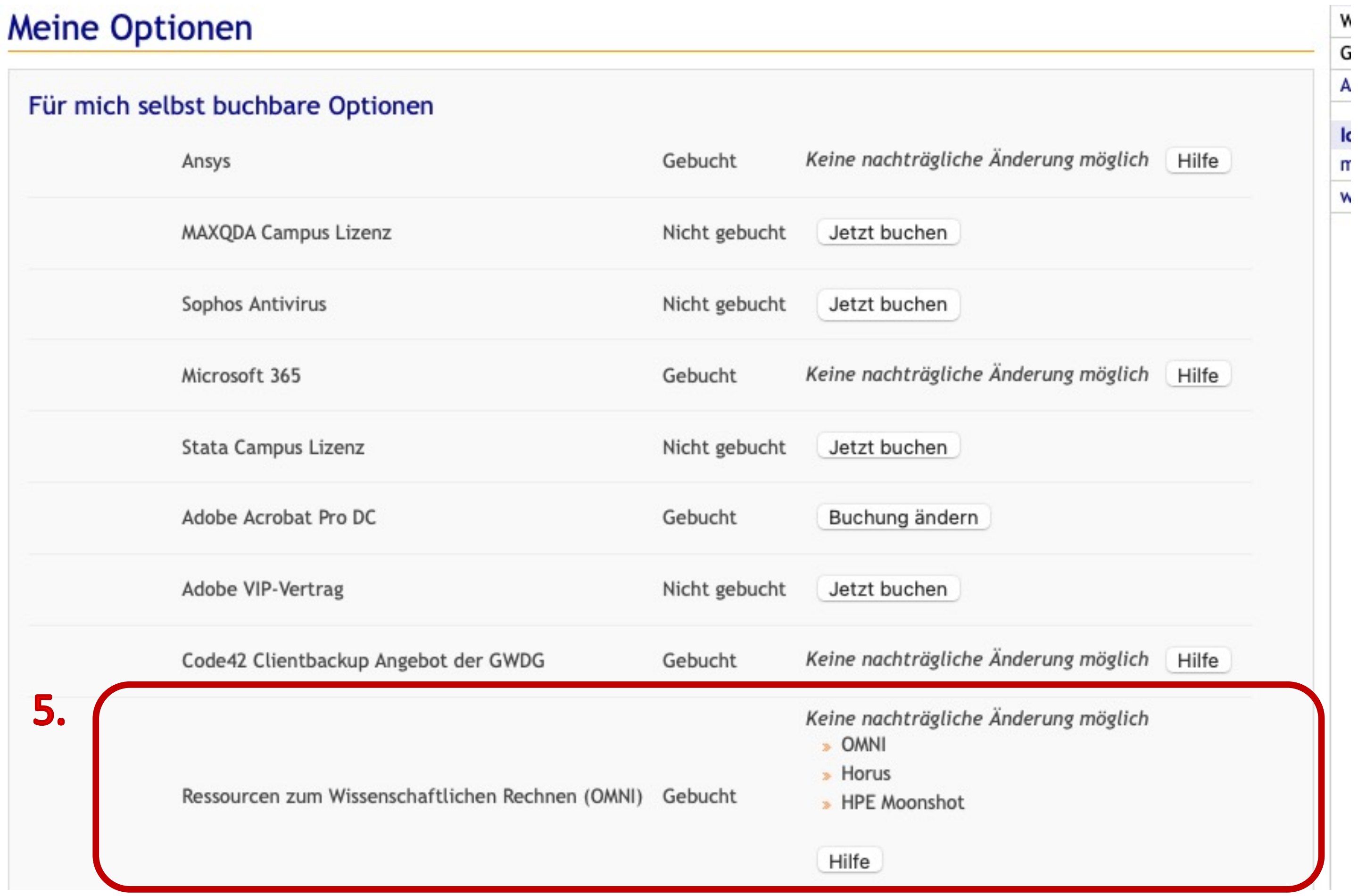
Für mich selbst buchbare Optionen

Ansys	Gebucht	*Keine nachträgliche Änderung möglich*	Hilfe
MAXQDA Campus Lizenz	Nicht gebucht	Jetzt buchen	
Sophos Antivirus	Nicht gebucht	Jetzt buchen	
Microsoft 365	Gebucht	*Keine nachträgliche Änderung möglich*	Hilfe
Stata Campus Lizenz	Nicht gebucht	Jetzt buchen	

Anmeldung
Welcome js056352
Gültigkeit: unbefristet
Abmelden

Ich möchte...
meine Nutzungsdauer verlängern
weitere ZIMT-Dienste beantragen

Registering an employee



Meine Optionen

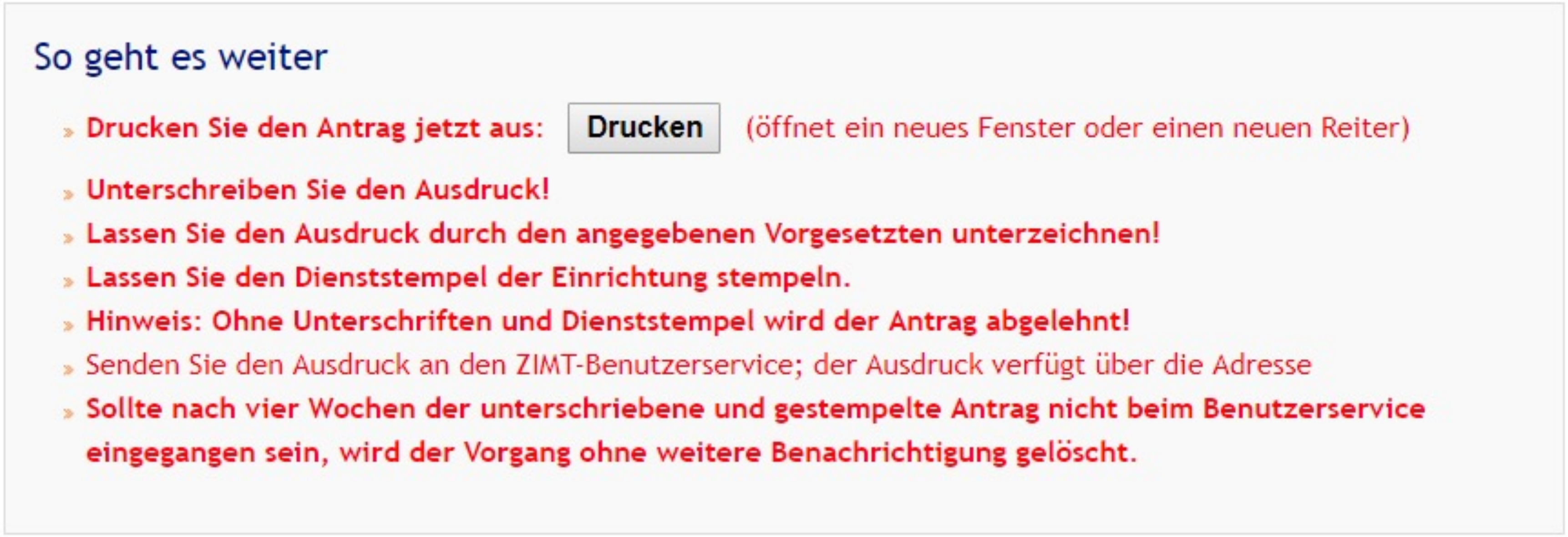
Für mich selbst buchbare Optionen

Ansys	Gebucht	*Keine nachträgliche Änderung möglich* Hilfe
MAXQDA Campus Lizenz	Nicht gebucht	Jetzt buchen
Sophos Antivirus	Nicht gebucht	Jetzt buchen
Microsoft 365	Gebucht	*Keine nachträgliche Änderung möglich* Hilfe
Stata Campus Lizenz	Nicht gebucht	Jetzt buchen
Adobe Acrobat Pro DC	Gebucht	Buchung ändern
Adobe VIP-Vertrag	Nicht gebucht	Jetzt buchen
Code42 Clientbackup Angebot der GWDG	Gebucht	*Keine nachträgliche Änderung möglich* Hilfe
Ressourcen zum Wissenschaftlichen Rechnen (OMNI)	Gebucht	*Keine nachträgliche Änderung möglich* » OMNI » Horus » HPE Moonshot Hilfe

5.

Registering an employee

6.



So geht es weiter

- Drucken Sie den Antrag jetzt aus: (öffnet ein neues Fenster oder einen neuen Reiter)
- Unterschreiben Sie den Ausdruck!
- Lassen Sie den Ausdruck durch den angegebenen Vorgesetzten unterzeichnen!
- Lassen Sie den Dienststempel der Einrichtung stempeln.
- Hinweis: Ohne Unterschriften und Dienststempel wird der Antrag abgelehnt!
- Senden Sie den Ausdruck an den ZIMT-Benutzerservice; der Ausdruck verfügt über die Adresse
- Sollte nach vier Wochen der unterschriebene und gestempelte Antrag nicht beim Benutzerservice eingegangen sein, wird der Vorgang ohne weitere Benachrichtigung gelöscht.

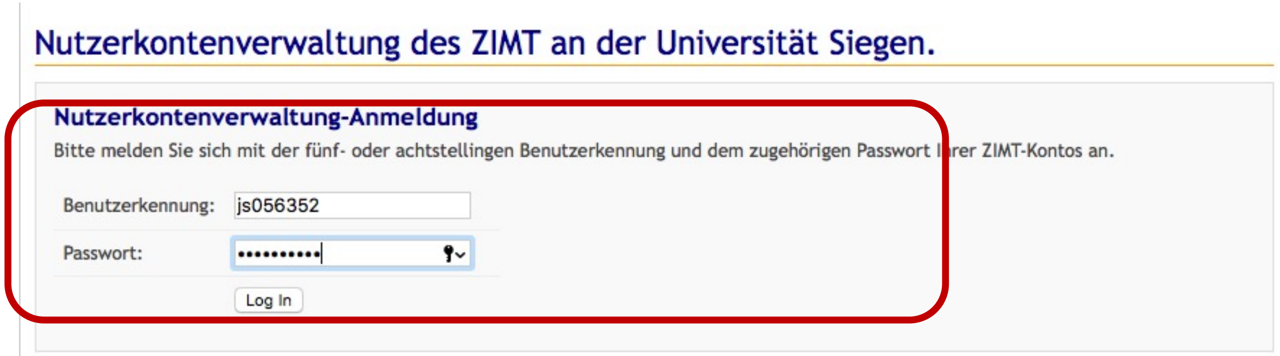
Registering a student

- Every student account has an assigned supervisor
  - Supervisor adds student in Nutzerkontenverwaltung
  - Supervisor is responsible

- Student accounts time out after 1 year
  - Can be extended by supervisor
  - Warning before time-out, data not immediately lost

- Otherwise, no restrictions for student accounts

Registering a student

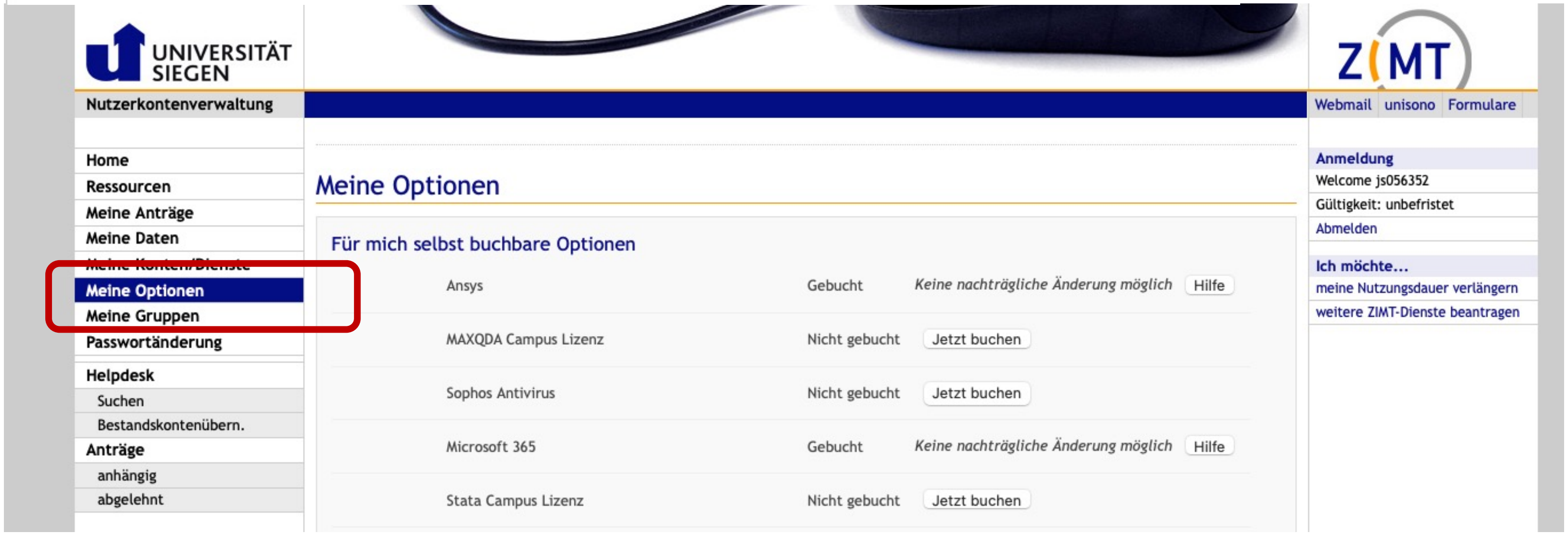


Nutzerkontenverwaltung des ZIMT an der Universität Siegen.

Nutzerkontenverwaltung-Anmeldung

Bitte melden Sie sich mit der fünf- oder achtstellingen Benutzerkennung und dem zugehörigen Passwort Ihrer ZIMT-Kontos an.

Benutzerkennung:	js056352
Passwort:	••••••••••



Meine Optionen

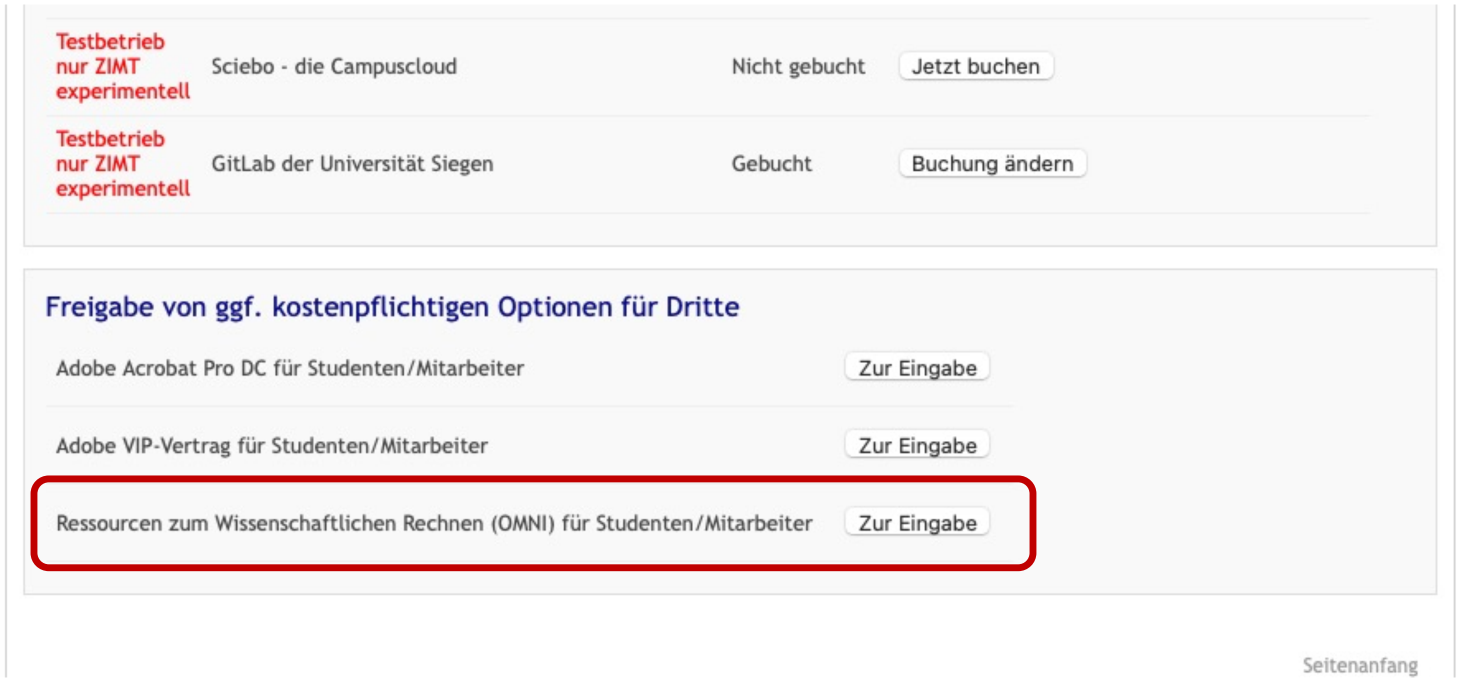
Für mich selbst buchbare Optionen

Ansys	Gebucht	*Keine nachträgliche Änderung möglich*	Hilfe
MAXQDA Campus Lizenz	Nicht gebucht	Jetzt buchen	
Sophos Antivirus	Nicht gebucht	Jetzt buchen	
Microsoft 365	Gebucht	*Keine nachträgliche Änderung möglich*	Hilfe
Stata Campus Lizenz	Nicht gebucht	Jetzt buchen	

Anmeldung
Welcome js056352
Gültigkeit: unbefristet
Abmelden

Ich möchte...
meine Nutzungsdauer verlängern
weitere ZIMT-Dienste beantragen

Registering a student



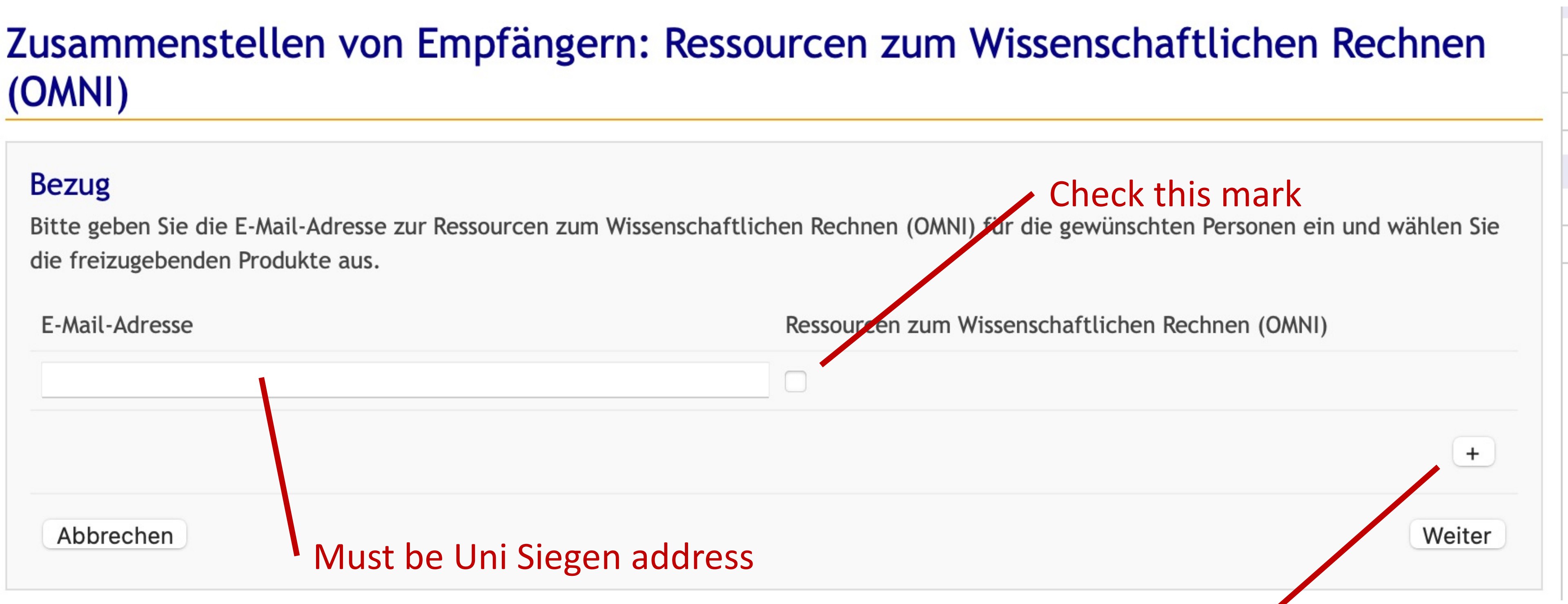
Testbetrieb nur ZIMT experimentell	Sciebo - die Campuscloud	Nicht gebucht	Jetzt buchen
Testbetrieb nur ZIMT experimentell	GitLab der Universität Siegen	Gebucht	Buchung ändern

Freigabe von ggf. kostenpflichtigen Optionen für Dritte

Adobe Acrobat Pro DC für Studenten/Mitarbeiter	Zur Eingabe
Adobe VIP-Vertrag für Studenten/Mitarbeiter	Zur Eingabe
Ressourcen zum Wissenschaftlichen Rechnen (OMNI) für Studenten/Mitarbeiter	Zur Eingabe

Seitenanfang

Registering a student



Zusammenstellen von Empfängern: Ressourcen zum Wissenschaftlichen Rechnen (OMNI)

Bezug

Bitte geben Sie die E-Mail-Adresse zur Ressourcen zum Wissenschaftlichen Rechnen (OMNI) für die gewünschten Personen ein und wählen Sie die freizugebenden Produkte aus.

E-Mail-Adresse	Ressourcen zum Wissenschaftlichen Rechnen (OMNI)
	☐

Check this mark

Must be Uni Siegen address

Add multiple people in one go if desired

After registration

- Prompt to agree to Terms of Use
  - Obstacle for people from some countries (Iran)

- Account not immediately ready
  - Usually next day
  - Contact us if still no e-mail after a week

- When ready: “Welcome to the OMNI cluster” e-mail
  - Keep this e-mail, it contains the cluster address(es)

Demo 1

Getting help

- Cluster website: https://cluster.uni-siegen.de
  - Usage information (like our courses)
  - What is installed

- Consult documentation, internet
  - Built-in help `man <command>` or `<command> -h` or `--help`

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

- Support e-mail address: hpc-support@uni-siegen.de

Demo 2

Problems

- Open a ticket
  - Email to hpc-support@uni-siegen.de
  - Centralized ZIMT ticket system
  - Tell us what error (message) is
    - For jobs: **attach job script, log file**

- 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 mostly OMNI cluster
- Using HPE Moonshot: relatively easy, similar to OMNI
- Other resources: get in contact with us

Special cases

- Jupyter portal:
  - **Not yet ready!**
  - Enter Jupyter portal address in Browser

- Adding students for a teaching event
  - Allowed in principle, **contact us**!
  - Moonshot nodes intended for this purpose
  - We may set up a reservation to avoid wait times (on a case-by-case basis)

- Absolutely not allowed: giving your password to another person

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
- Available 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 (later)
- Can specify configurations to simplify login
- Console-based, but opening windows possible
- Multiple simultaneous connections possible

Demo 3

SSH Configuration

- OpenSSH allows presets
- Can create text file `~/.ssh/config`
  - Edit if already exists
- One preset per connection (cluster etc.)
  - Specify username
  - Other options (many possibilities)
- Log in with `ssh <presetname>` instead of `ssh [options] <user>@<host>`

SSH Configuration File

Config file on laptop (not cluster)

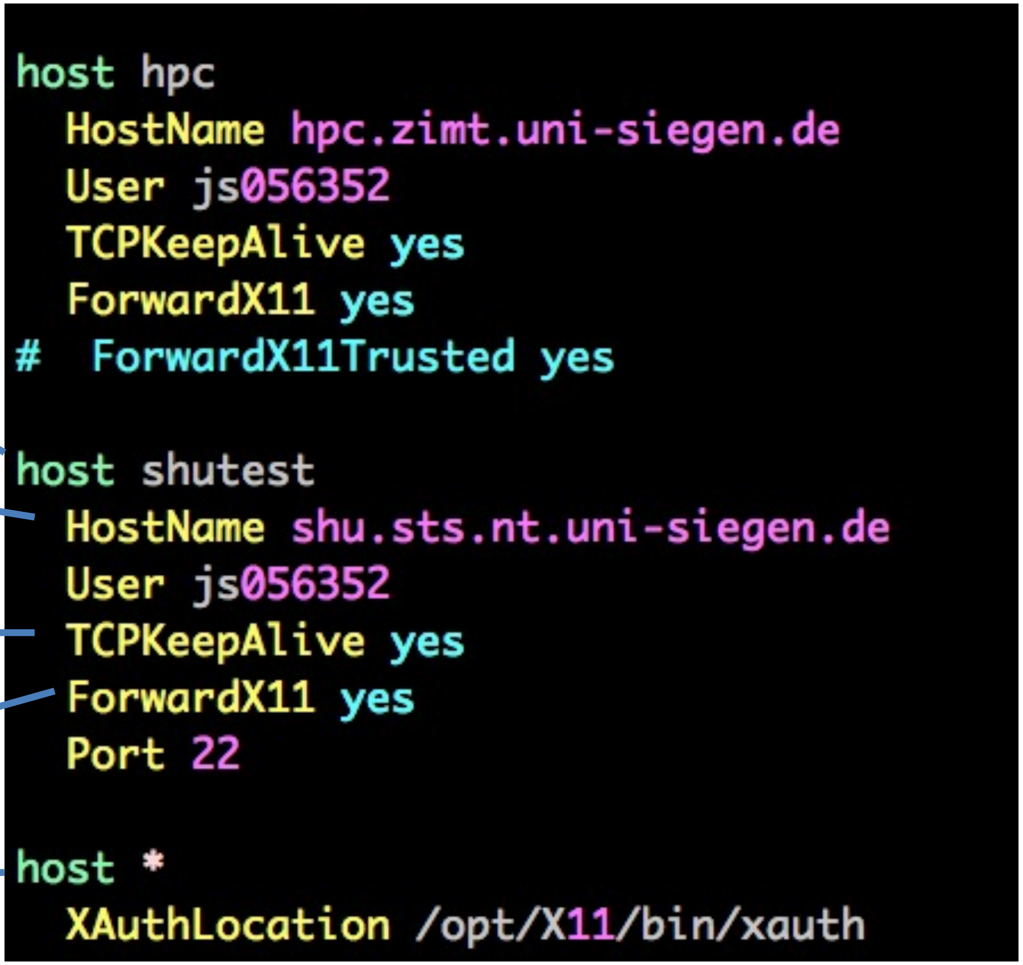
Preset name (your choice)

Target host

Various options

X window support (later)

Options for all hosts



```
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 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` directory
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

- Run SSH key generator
  1. On **local** PC, type `ssh-keygen`
  2. Enter filename for new key
     - Should be inside `~/.ssh` directory
     - **Caution:** will overwrite without asking
  3. Enter passphrase
  4. Confirm passphrase

Copy key to cluster

- On **local** PC, use `ssh-copy-id` command
  - Syntax: `ssh-copy-id -i <keyfile> <user>@<host>`
  - Not available on Windows
  - Remember you need to be inside Uni network/Uni VPN

- Alternative: copy manually
  - On **local** PC, open **public** key file with text editor
  - One line of text, three parts: algorithm, key, comment
  - On **cluster**, open `~/.ssh/authorized_keys`
  - Paste line, adjust comment as needed

Key selection and tips

- 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

Public key format

- One line per key (e.g. `authorized_keys`)

- Three elements:
  - Encryption algorithm
  - Public key
  - Comment

- Comment may be adjusted (from which device)

Public key example

Algorithm

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQC+iMPDyFgXxpIF8r0rJFFXY0S/Gy/l
1ijXuEs564y6cG/F95uxTjEC0kJEdmtix8lYfN8eIQ92xxib4/WQ0A243oh2
svD10R3gKNtzyjvc+eNMPOgf2yY0IlV1I3GfwGgLnTSKYgQH6HGcUgb/nQF3
eCcB9r3WCyoZ/tm4DrvuU8RQCD50fpq6D1p5Ol7JaXCNSor9sbXqSSodnjTR
nFyDSDf3sUGQuUUdXXlB1H35Knh2QWEBl4AjyTnehcUHDodRXmLIcrplIdlF
jDIEF1TP3h4FHeTcEY4ibRZzXgpBj60Dy6mCq3AuNnriSu6adCRIBkZ81giR
VpilSouzAbXQofzfJrzFnVcEDtPzbNUH6VCS32KMddlssraQmCwtEtFfu9nG
C9F+dXoV38ZxQs9F4cUjqLGtkv51C0viGhadWmkpn0Ir5VdV7Vu319wWZ5wL
FCrh/RR7S0UuIfr3VcnrK58F16yM1A/i8i3rQyAnZBn86/YwfnnIFRE9C5sA
KAU= jansteiner@Jans-MacBook-Pro.local
```

Public key

Single space (no linebreak)

Comment

Exercise 1a

- Reminder about exercises:
  - Groups of three
  - One person shares screen
  - Solve cooperatively
  - Screen-sharer switches for next exercise
  - I will visit each group

- You have been given Linux cheat sheets

Exercise 1a

Objectives:

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

Tasks:

- Log into the cluster with password
- Familiarize yourself with Linux console if necessary
- Set up an SSH config on your local PC
- Set up password-less login (key pair)
  - Windows users: may skip adding

Note the following page!

Exercise 1a

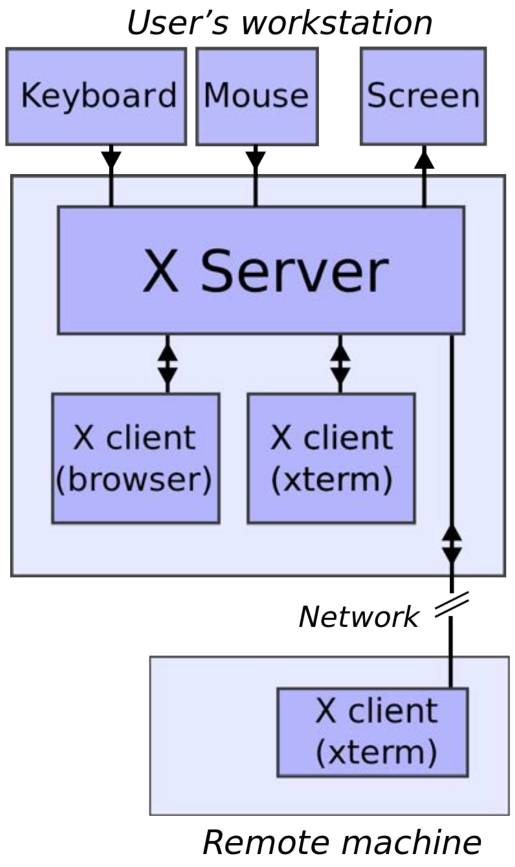
- Let's assign training users now
- You may use your own account instead
- Cluster address: `<removed>`
- If bored, get creative
  - Try launching different programs
  - Figure out how to get to the other login node
  - …

<u>Login info Feb, 15/16:</u>
User: schulungXY
PW: <removed>

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

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, graphical front-ends also exist for all OSes

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

Third-party tools: connection

- Connecting: graphical clients exist for all OSes
- Windows: particularly important because native SSH support limited
  - Two main options:
    - MobaXTerm: modern, many features
      - Integrated file transfer, X server, text editor, key generator
    - PuTTY: trusted, only SSH connections (no X server)
      - Separate X server: xming
- Mac OS: external X server necessary

Third-party tools: file transfer

- File transfer: clients exist for all OSes
  - Windows:
    - MobaXTerm
    - WinSCP
  - Mac OS
    - Forklift
    - Cyberduck (no experience)

Demo 7

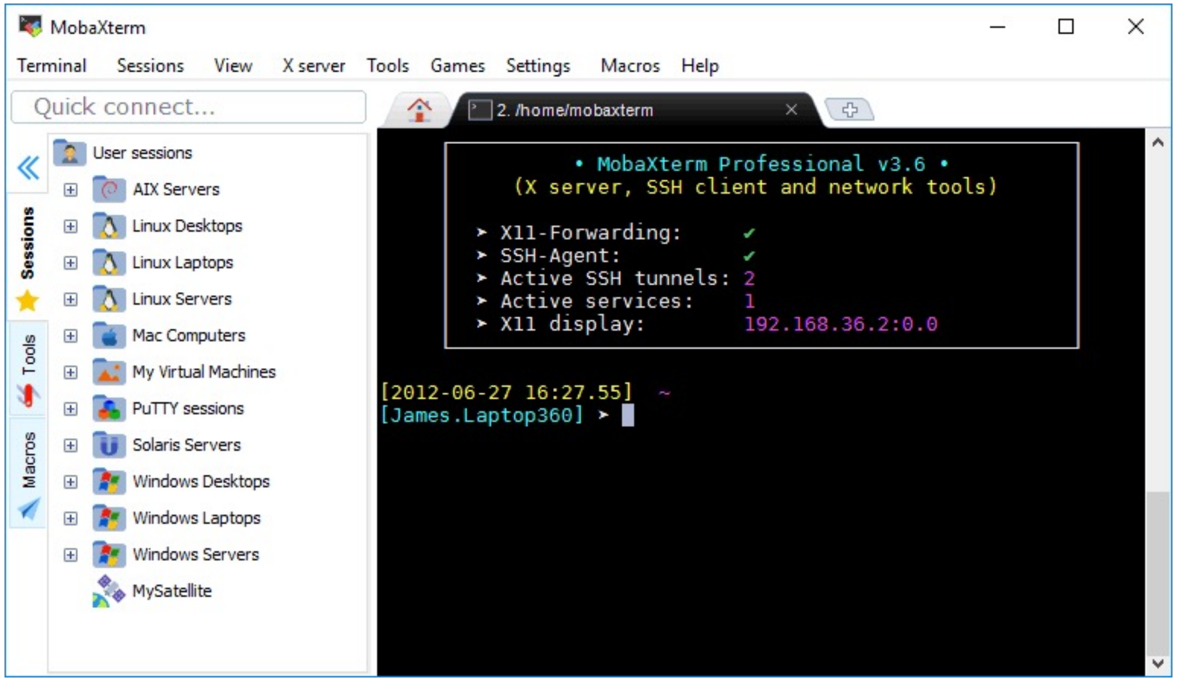
Third-party tools

- Key point:
  - All built on top of SSH and SCP
  - Same concepts still apply
  - Enter same login data

- MobaXTerm is particularly important
  - Let’s look at it more closely

Windows SSH Software

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



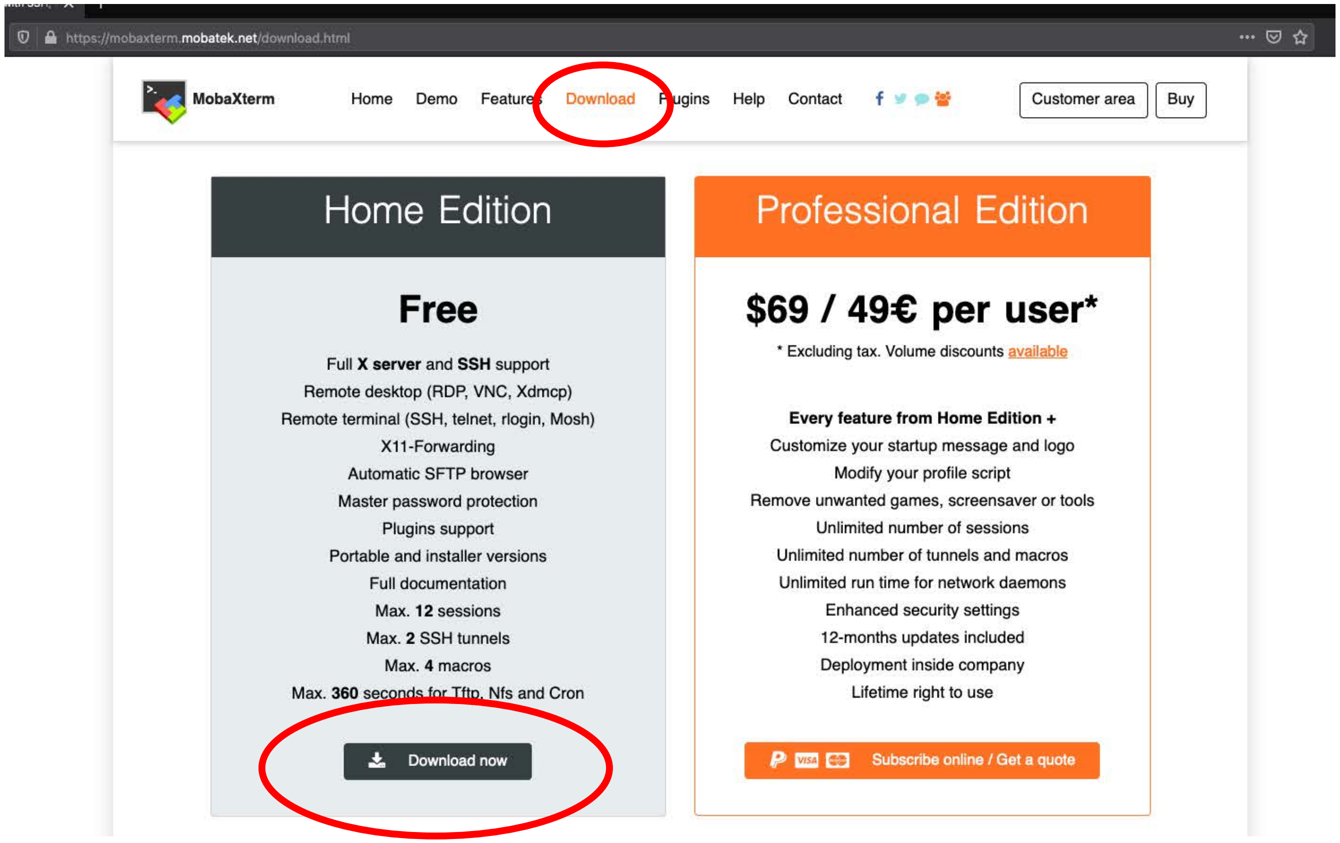
```
        • MobaXterm Professional v3.6 •
   (X server, SSH client and network tools)

 ➤ X11-Forwarding:      ✔
 ➤ SSH-Agent:           ✔
 ➤ Active SSH tunnels:  2
 ➤ Active services:     1
 ➤ X11 display:         192.168.36.2:0.0

[2012-06-27 16:27.55]  ~
[James.Laptop360] ➤
```

Source: mobatek.net

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

MobaXTerm: Download



https://mobaxterm.mobatek.net/download.html

Home Edition

Free

Full **X server** and **SSH** support
Remote desktop (RDP, VNC, Xdmcp)
Remote terminal (SSH, telnet, rlogin, Mosh)
X11-Forwarding
Automatic SFTP browser
Master password protection
Plugins support
Portable and installer versions
Full documentation
Max. **12** sessions
Max. **2** SSH tunnels
Max. **4** macros
Max. **360** seconds for Tftp, Nfs and Cron

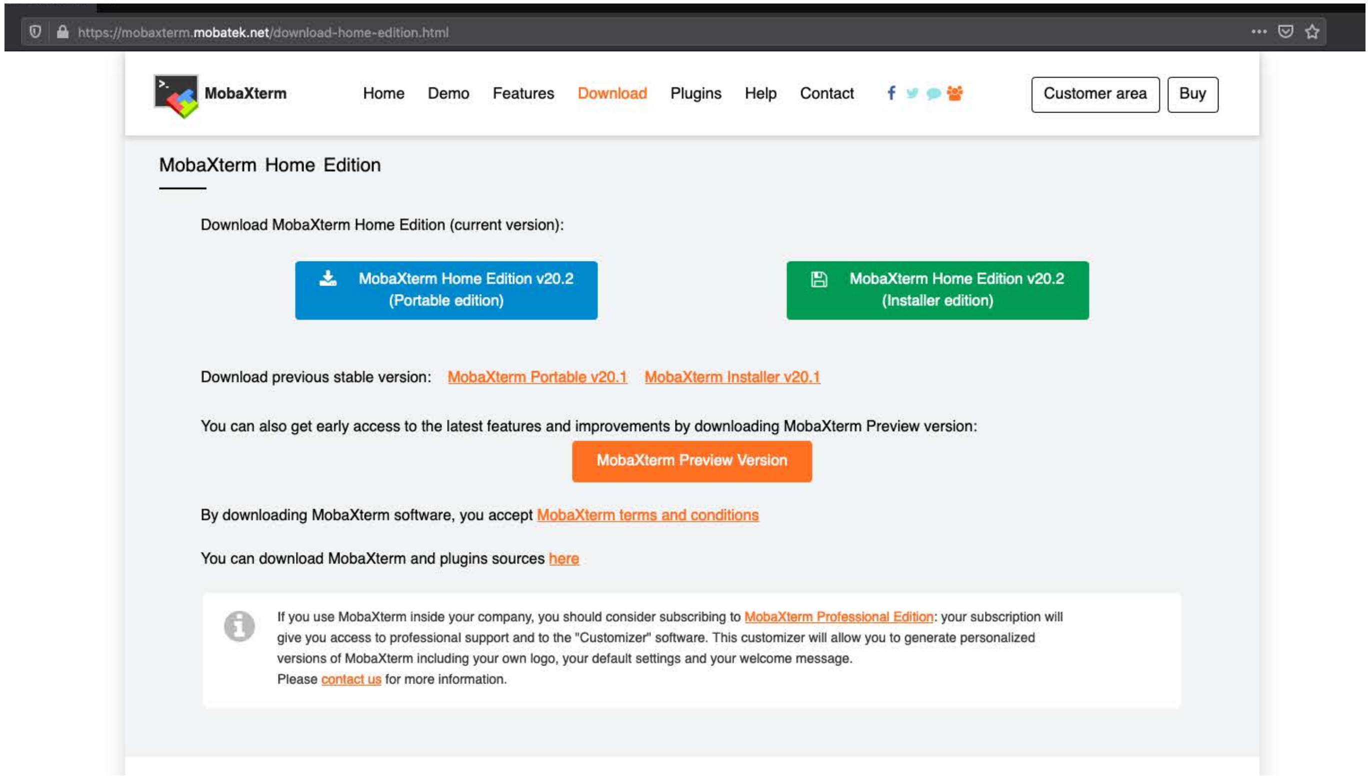
Professional Edition

$69 / 49€ per user*

* Excluding tax. Volume discounts available

Every feature from Home Edition +
Customize your startup message and logo
Modify your profile script
Remove unwanted games, screensaver or tools
Unlimited number of sessions
Unlimited number of tunnels and macros
Unlimited run time for network daemons
Enhanced security settings
12-months updates included
Deployment inside company
Lifetime right to use

Connecting: MobaXTerm


https://mobaxterm.mobatek.net/download-home-edition.html

MobaXterm
Home
Demo
Features
Download
Plugins
Help
Contact
Customer area
Buy

MobaXterm Home Edition

Download MobaXterm Home Edition (current version):

MobaXterm Home Edition v20.2 (Portable edition)
MobaXterm Home Edition v20.2 (Installer edition)

Download previous stable version: MobaXterm Portable v20.1 MobaXterm Installer v20.1

You can also get early access to the latest features and improvements by downloading MobaXterm Preview version:

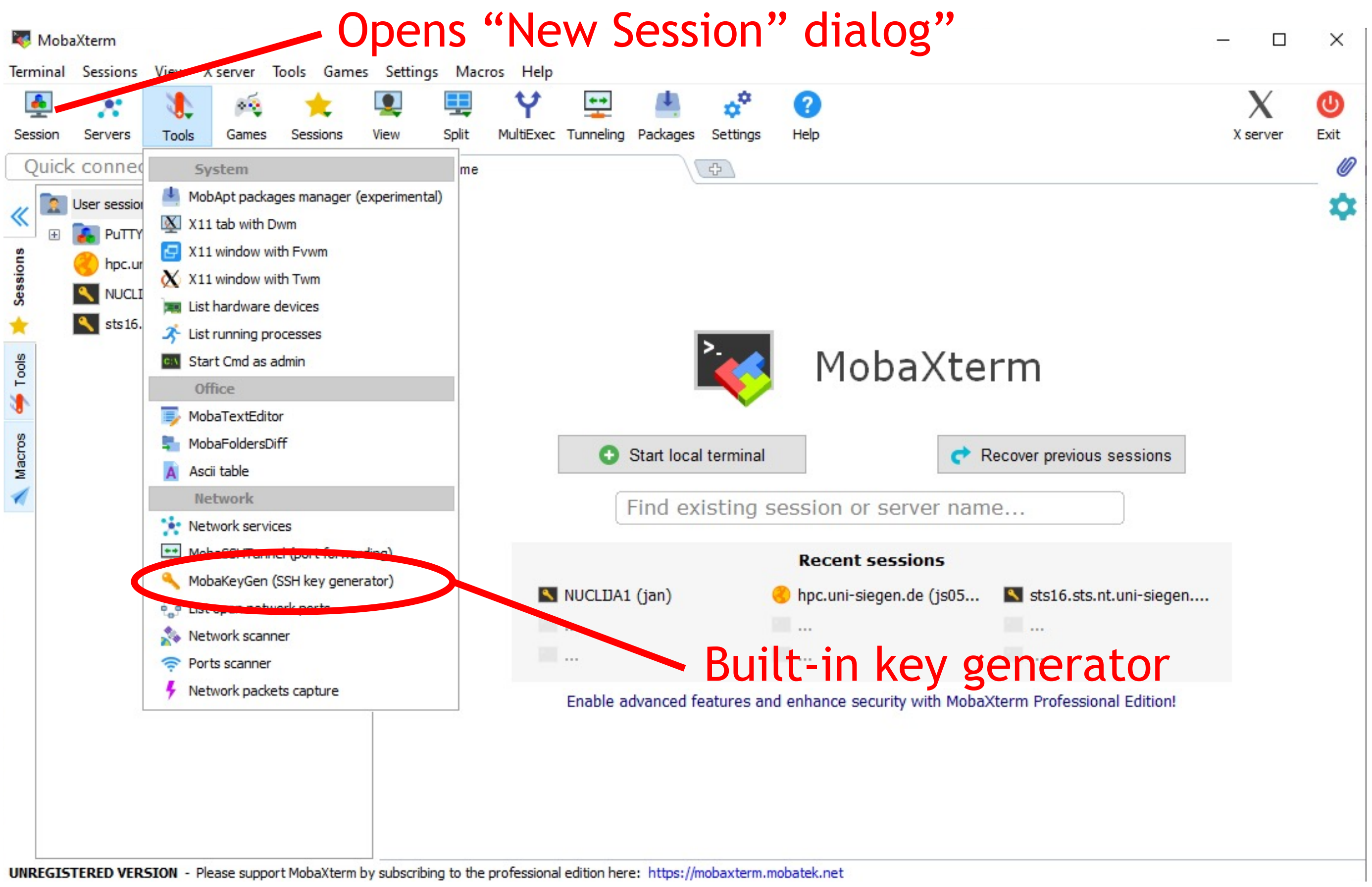
MobaXterm Preview Version

By downloading MobaXterm software, you accept MobaXterm terms and conditions

You can download MobaXterm and plugins sources here

If you use MobaXterm inside your company, you should consider subscribing to MobaXterm Professional Edition: your subscription will give you access to professional support and to the "Customizer" software. This customizer will allow you to generate personalized versions of MobaXterm including your own logo, your default settings and your welcome message.
Please contact us for more information.

Connecting: MobaXTerm



Opens “New Session” dialog”
Built-in key generator
MobaXterm
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
System
MobApt packages manager (experimental)
X11 tab with Dwm
X11 window with Fvwm
X11 window with Twm
List hardware devices
List running processes
Start Cmd as admin
Office
MobaTextEditor
MobaFoldersDiff
Ascii table
Network
Network services
MobaSSHTunnel (port forwarding)
MobaKeyGen (SSH key generator)
List open network ports
Network scanner
Ports scanner
Network packets capture
Start local terminal
Recover previous sessions
Find existing session or server name...
Recent sessions
NUCLIDA1 (jan)
hpc.uni-siegen.de (js05...
sts16.sts.nt.uni-siegen....

Enable advanced features and enhance security with MobaXterm Professional Edition!

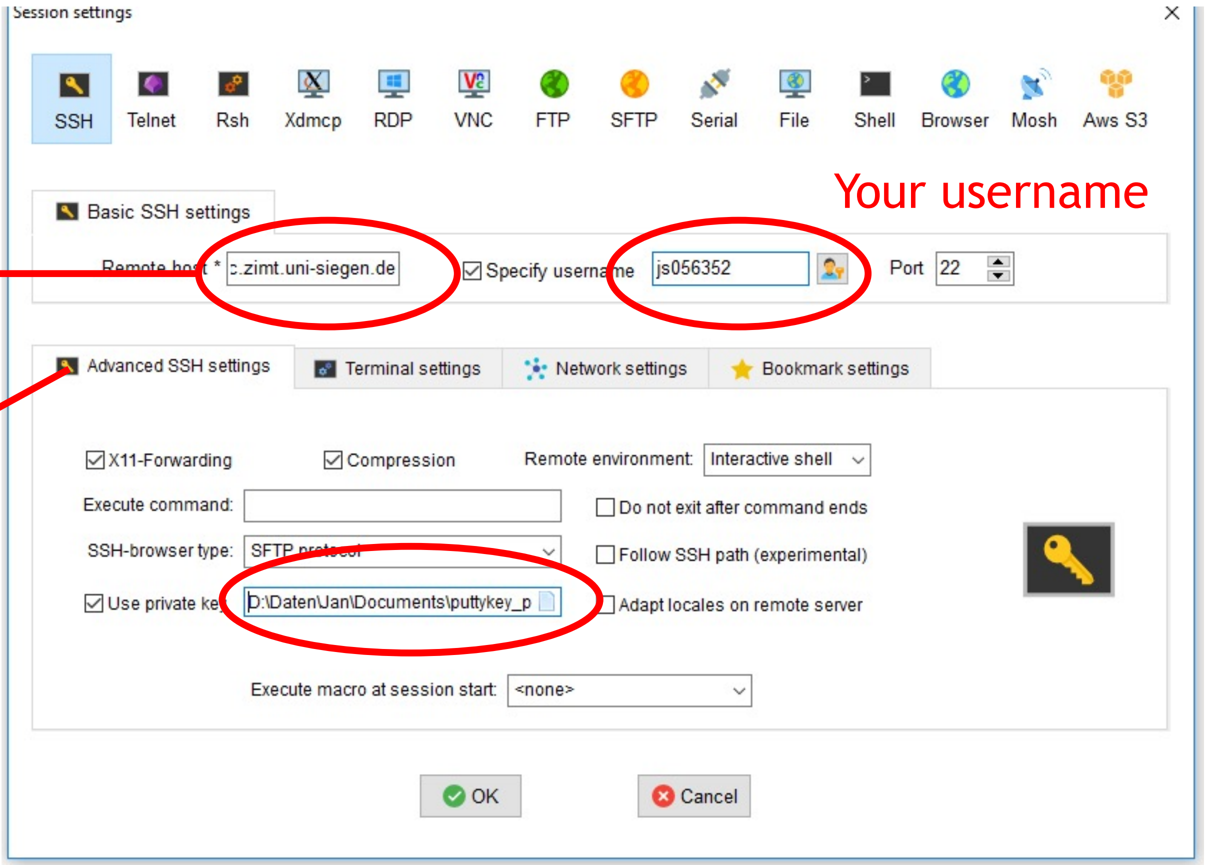
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

Connecting: MobaXTerm configuration

Cluster address

Most important tab

Same options as before



Remote host * c.zimt.uni-siegen.de ☑ Specify username js056352 Port 22

☑ X11-Forwarding ☑ Compression Remote environment: Interactive shell

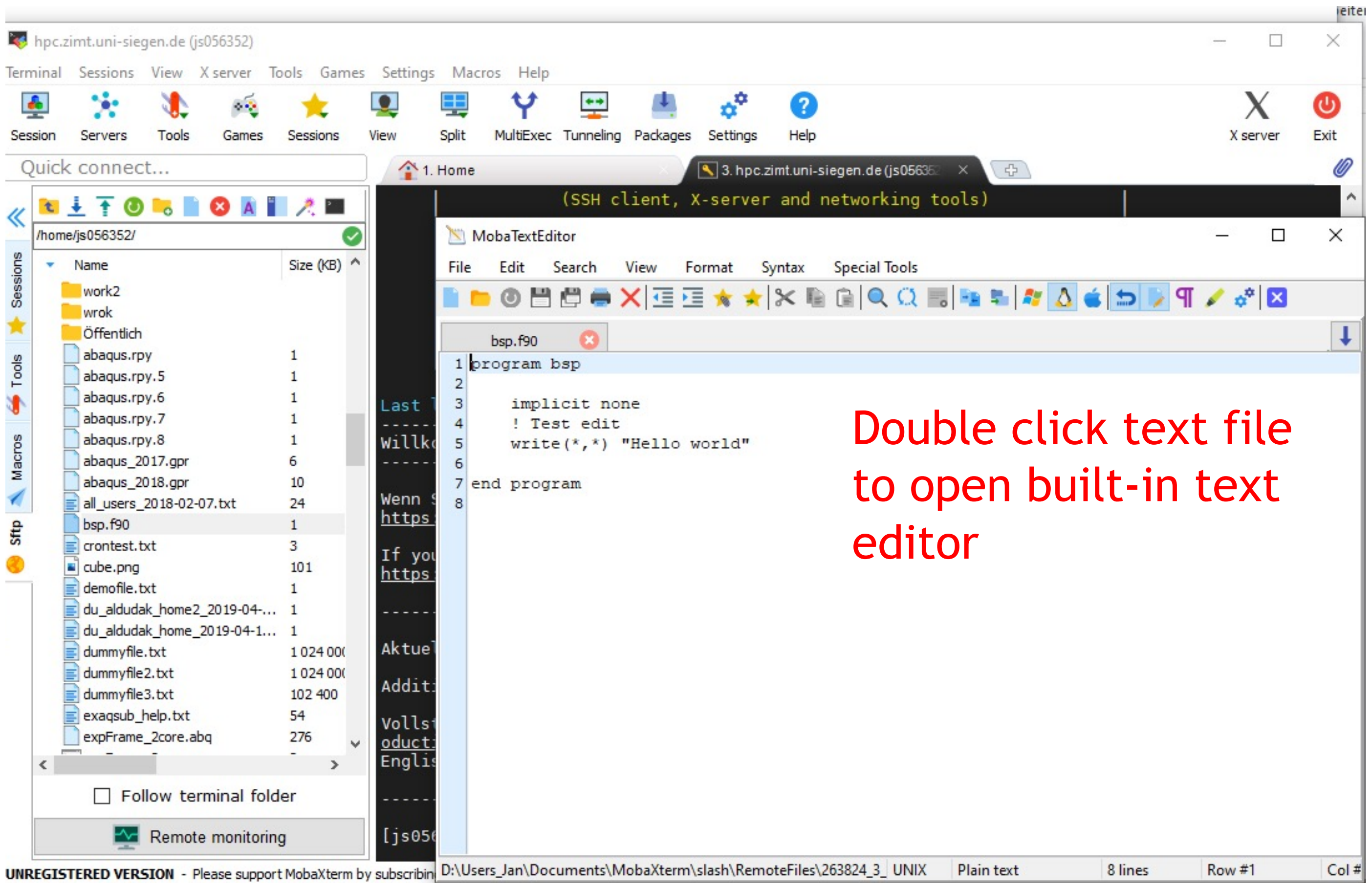
Execute command: ☐ Do not exit after command ends

SSH-browser type: SFTP protocol ☐ Follow SSH path (experimental)

☑ Use private key D:\Daten\Jan\Documents\puttykey_p ☐ Adapt locales on remote server

Execute macro at session start: <none>

Connecting: MobaXTerm features


hpc.zimt.uni-siegen.de (js056352)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
1. Home
3. hpc.zimt.uni-siegen.de (js05636
/home/js056352/
MobaTextEditor
File Edit Search View Format Syntax Special Tools
bsp.f90
Follow terminal folder
Remote monitoring
UNREGISTERED VERSION - Please support MobaXterm by subscribin
D:\Users_Jan\Documents\MobaXterm\slash\RemoteFiles\263824_3_ UNIX Plain text 8 lines Row #1 Col #

```
(SSH client, X-server and networking tools)
```

Name	Size (KB)
work2	
wrok	
Öffentlich	
abaqus.rpy	1
abaqus.rpy.5	1
abaqus.rpy.6	1
abaqus.rpy.7	1
abaqus.rpy.8	1
abaqus_2017.gpr	6
abaqus_2018.gpr	10
all_users_2018-02-07.txt	24
bsp.f90	1
crontest.txt	3
cube.png	101
demofile.txt	1
du_aldudak_home2_2019-04-...	1
du_aldudak_home_2019-04-1...	1
dummyfile.txt	1 024 000
dummyfile2.txt	1 024 000
dummyfile3.txt	102 400
exaqsub_help.txt	54
expFrame_2core.abq	276

```
program bsp

   implicit none
   ! Test edit
   write(*,*) "Hello world"

end program
```

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 1b

Objectives:

- You are familiar with X servers and file transfer
- You have a setup you are comfortable with

Tasks:

- Set up SSH connection with X server
- Practice using SCP
- If on Windows: install MobaXTerm and set it up
- If on Mac OS: install Xquartz
- Play around with GUI clients of your choice

Remember: if bored, get creative

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)
  - OMNI: `/work`
  - Burst buffer: `/fast`

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

Burst buffer

- OMNI has so-called “burst buffer”
  - Made up of SSDs
  - Considerably faster file input/output
  - 32 TB

- Only use it if you need it
  - Limited space for all users

- Works identically to normal workspaces
  - Additional option `ws_allocate -F fast <name> <days>`
  - File system: `/fast`

Workspaces

- Additional options:
  - Send e-mail before workspace expires
  - Generate calendar item : `ws_send_ical`

- 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 information
- “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 openmpi4
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

Environment setup on OMNI

- OMNI cluster is **multipurpose**
- Software from several different sources
- Tricky module setup
- Default modules always loaded: SLURM, GCC compiler, OpenMPI
  - Some modules depend on specific compiler or MPI
- Modules come in four groups

Environment setup on OMNI

- Software sources:
  - CentOS (operating system)
  - Bright (cluster management software)
  - OpenHPC (software collection)
- OpenHPC modules are only displayed if compiler and MPI is correct
  - Gnu GCC vs. Intel Compiler
  - OpenMPI vs. IntelMPI
- Some modules do not depend on any of those
  - GPU modules are even mutually exclusive

Demo 10

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 <u>**job**</u>

- 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

<u>Primary queues:</u>

- `debug`:
  - Only for testing
  - 15 minutes runtime
- `short, medium long`
- `expert`:
  - For big jobs
  - Users must contact us and obtain permission

Running computations: queues

Special queues:

- `gpu`:
  - If you want to use GPUs
  - Needs additional options in job script: `--gres=gpu:X` (where X is number of GPUs needed)

- `smp`:
  - two nodes, 1536 GB RAM, 64 cores (Intel CPUs)

- `htc`:
  - HPE Moonshot system

Demo 11

Monitoring jobs

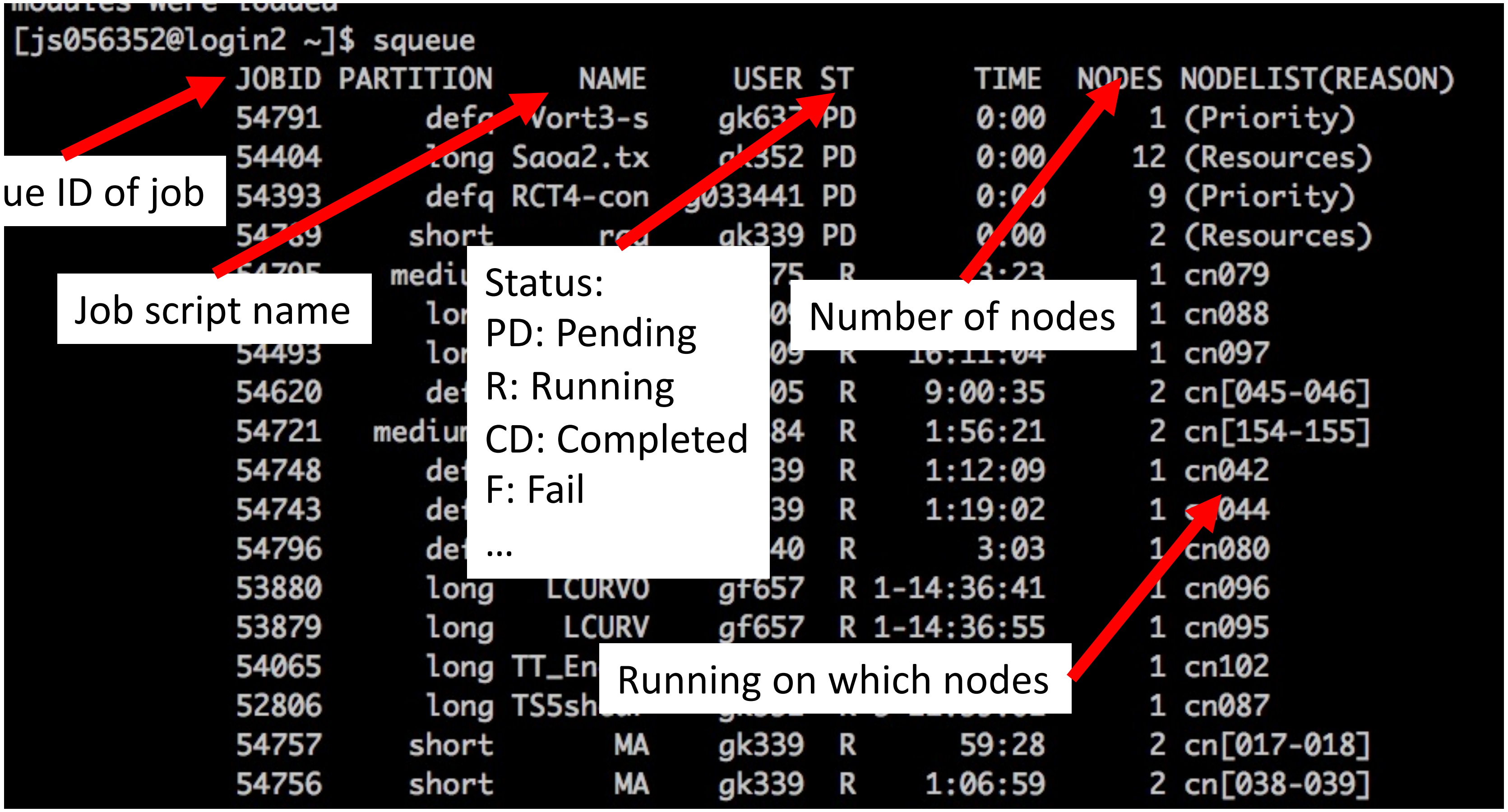
- Check regularly what your job does
  - Your first job will fail (guaranteed)
  - Might be a bug later on
  - Might be a problem with the cluster
  - Might run out of resources
  - Might not be finished when time limit is reached

- Main command to check what your job is doing:

```
squeue
```

- If possible, use checkpointing (write intermediate results)

Monitoring: squeue example



```
modules were loaded
[js056352@login2 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             54791      defq  Vort3-s    gk637 PD       0:00      1 (Priority)
             54404      long Saoa2.tx    gk352 PD       0:00     12 (Resources)
             54393      defq RCT4-con g033441 PD       0:00      9 (Priority)
             54789     short      rcq    gk339 PD       0:00      2 (Resources)
             54795    mediu                  75  R       3:23      1 cn079
                        lor                  0                    1 cn088
             54493      lor                  09  R   16:11:04      1 cn097
             54620      def                  05  R    9:00:35      2 cn[045-046]
             54721    mediu                  84  R    1:56:21      2 cn[154-155]
             54748      def                  39  R    1:12:09      1 cn042
             54743      def                  39  R    1:19:02      1 cn044
             54796      def                  40  R       3:03      1 cn080
             53880      long   LCURVO    gf657  R 1-14:36:41      1 cn096
             53879      long    LCURV    gf657  R 1-14:36:55      1 cn095
             54065      long TT_En                                1 cn102
             52806      long TS5sh                                1 cn087
             54757     short       MA    gk339  R      59:28      2 cn[017-018]
             54756     short       MA    gk339  R    1:06:59      2 cn[038-039]
```

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 12

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
```

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

SLURM will print job ID

Demo 13

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 simple job script
2. You can interpret the output of `squeue` and `sinfo`

Tasks:

- 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
```

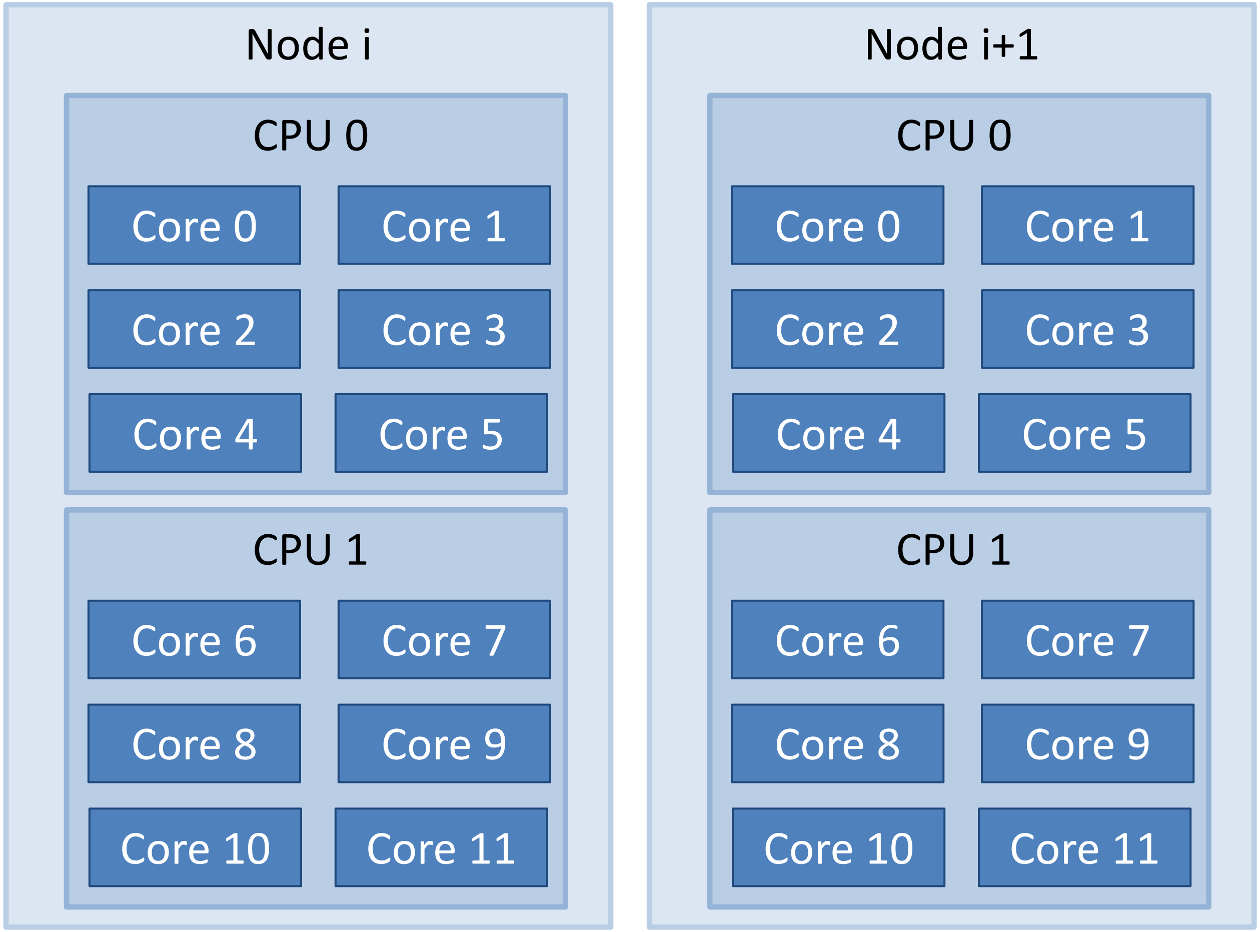
Outline

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

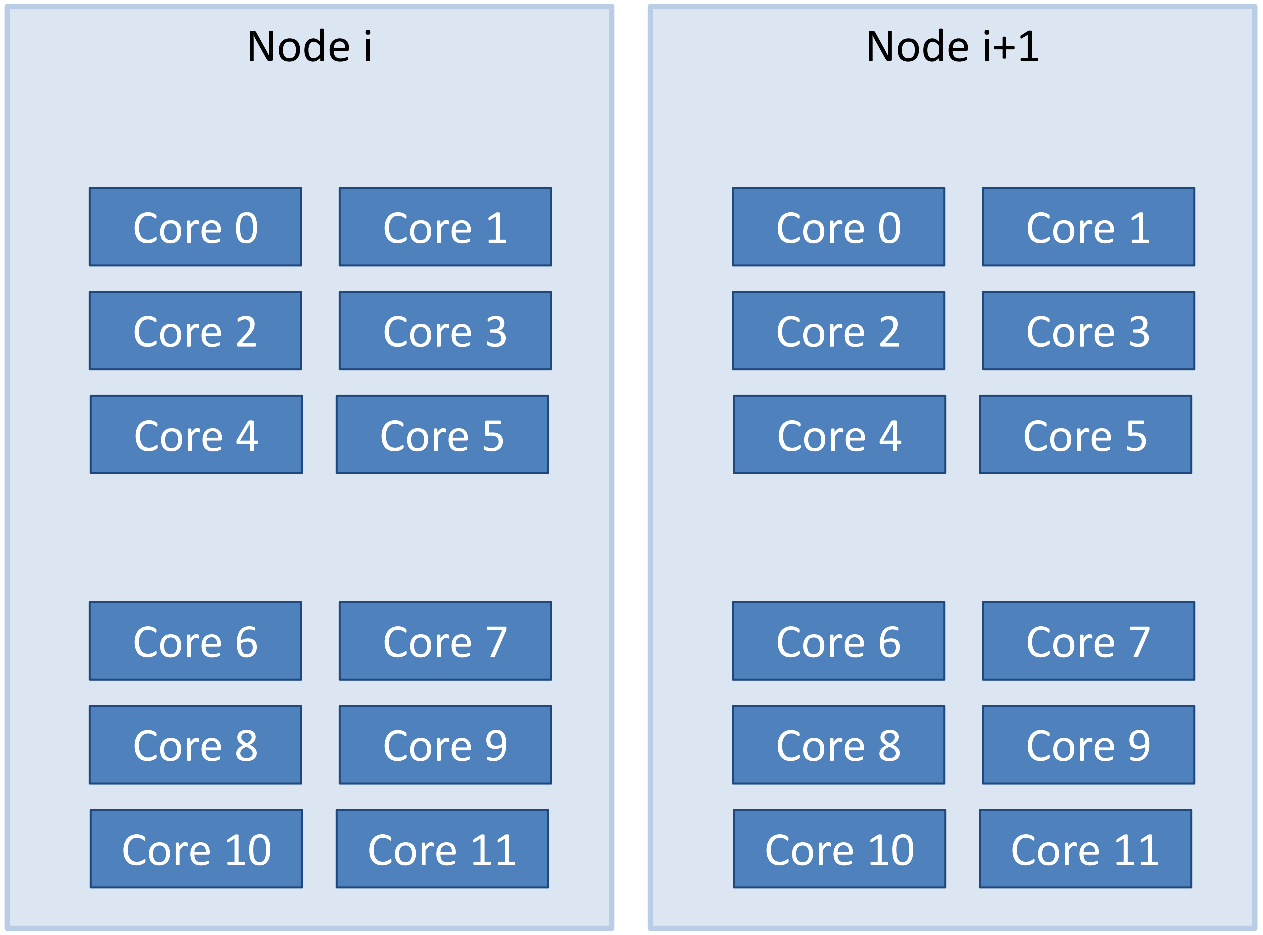


Hardware visualized

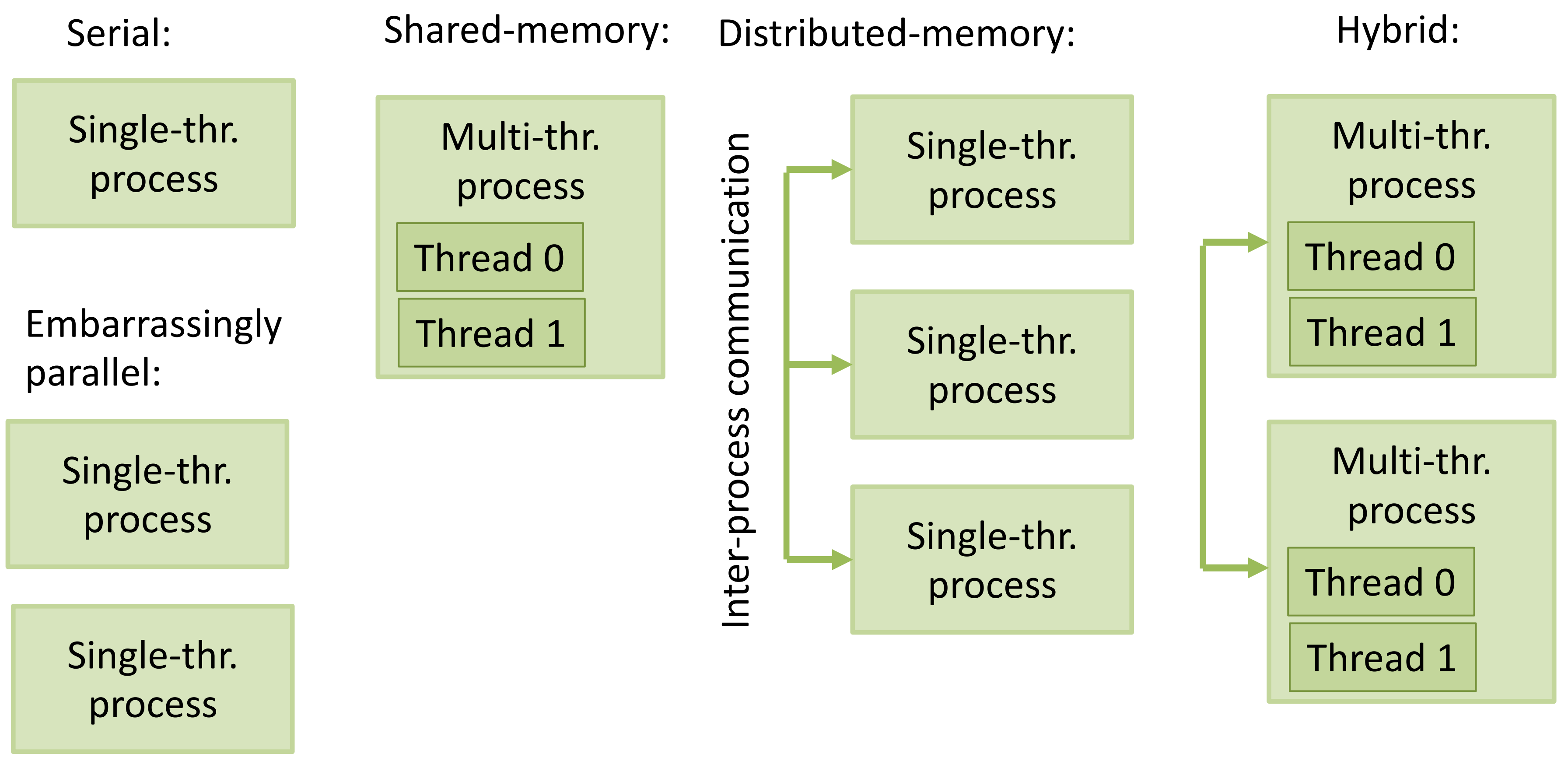
Simplification:

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

→ **Ignored for now**



Workloads common in HPC

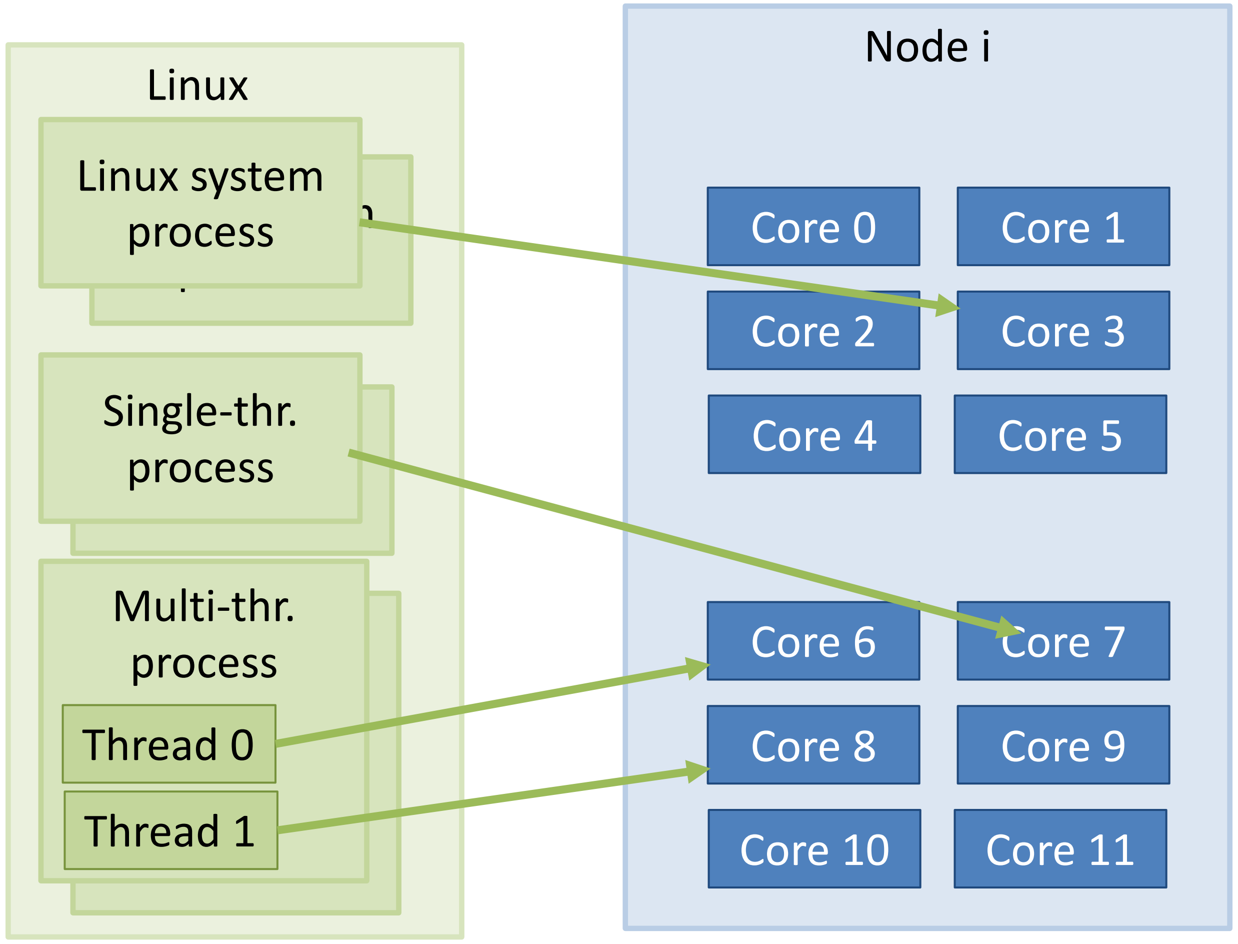


Software visualized

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)



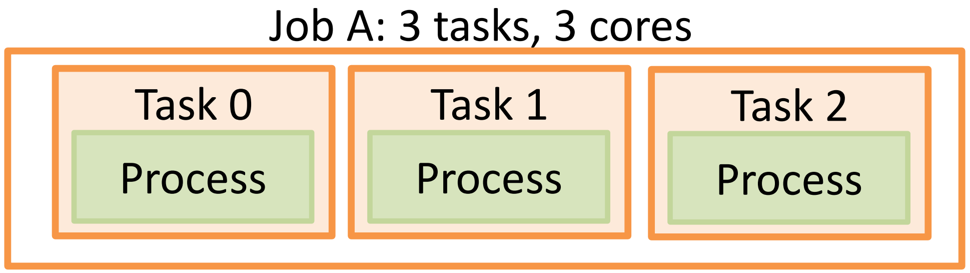
Example SLURM jobs

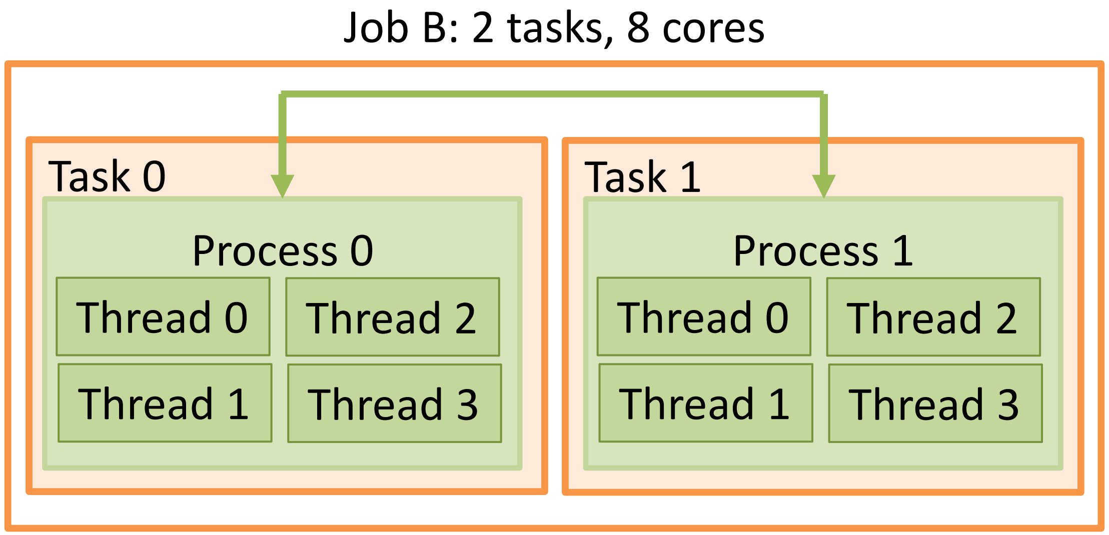
- User specifies which and how many processes and threads to run in job

→ **Tasks**

What SLURM does:

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





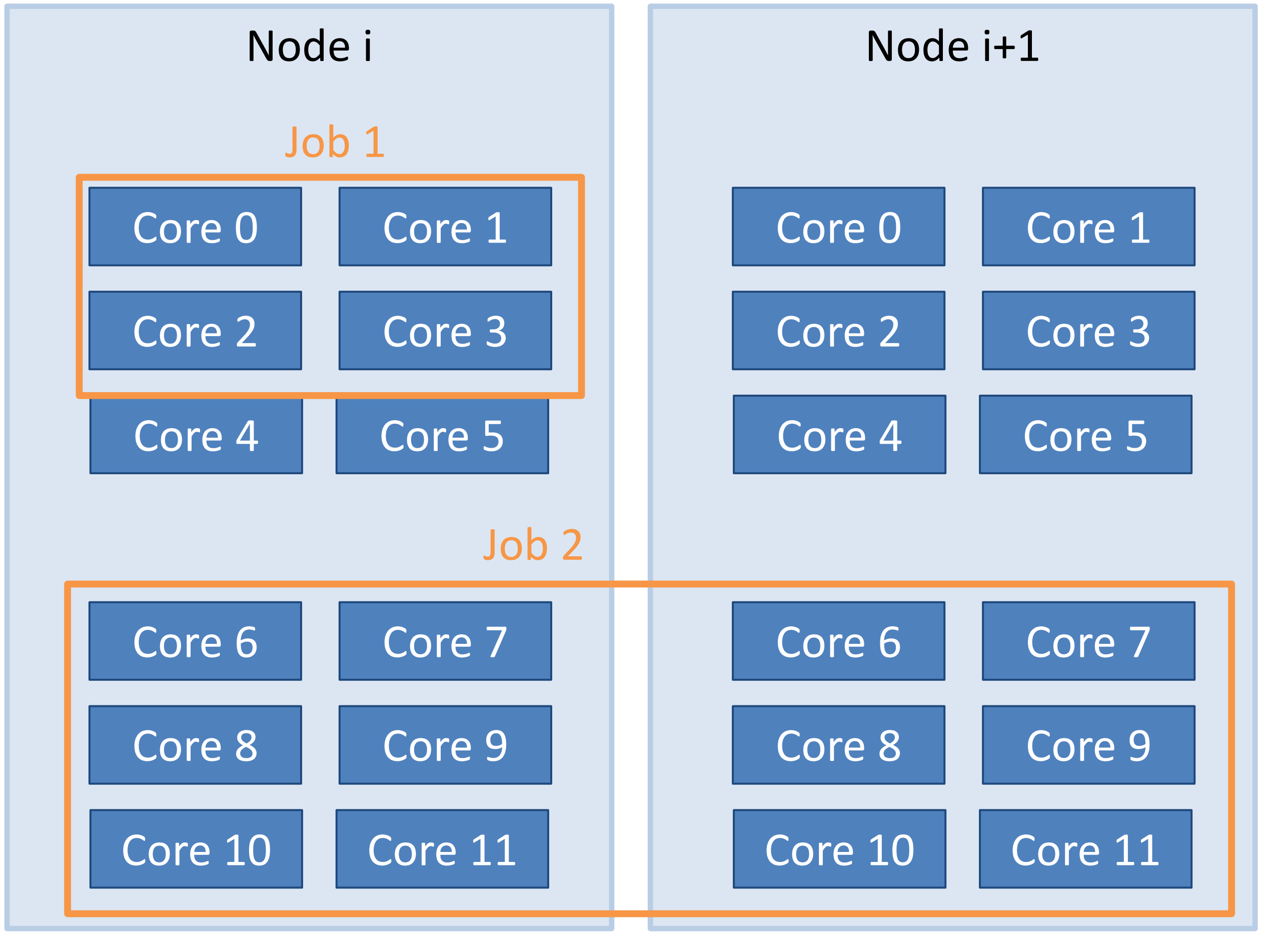
Example SLURM jobs

- 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

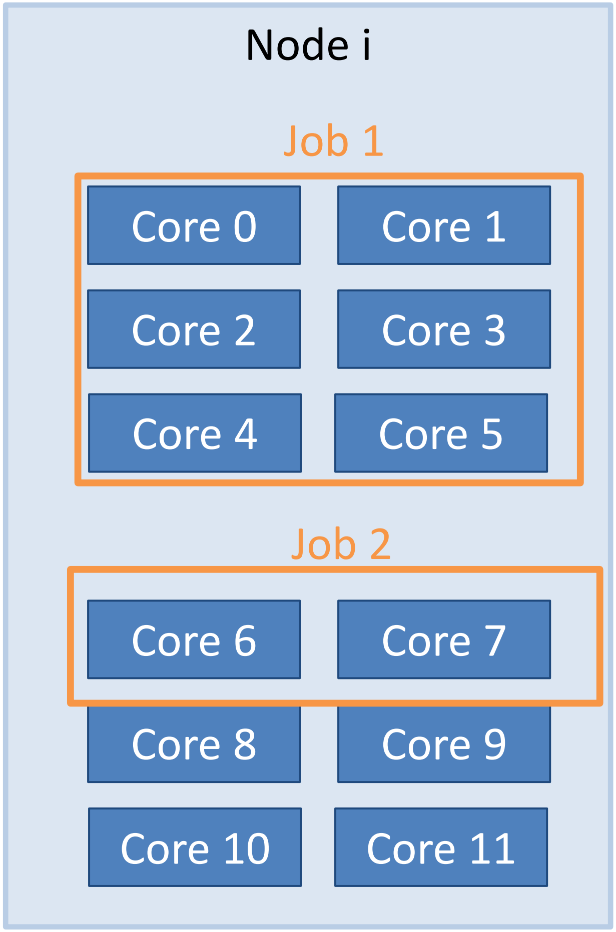


Example SLURM jobs

Complications:

- Longer queues allow 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=128
```

```
#SBATCH -n 128
```

- Or alternatively:

```
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=64
```

```
#SBATCH -N 2
```

- In both cases possible:

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

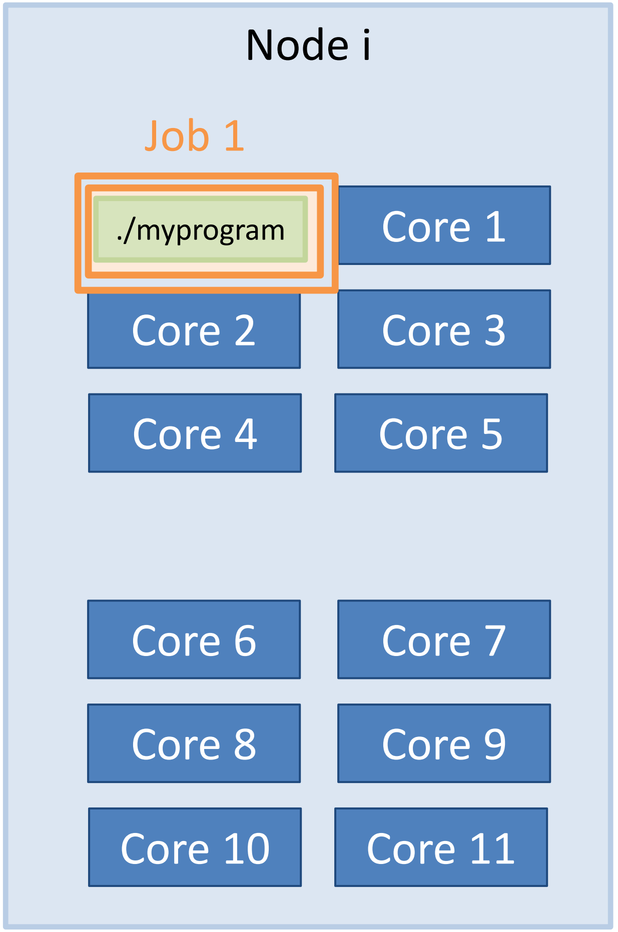
Caution: sometimes starts with n, sometimes doesn't

- Consult `sbatch` documentation

Example job scripts

Simple serial program

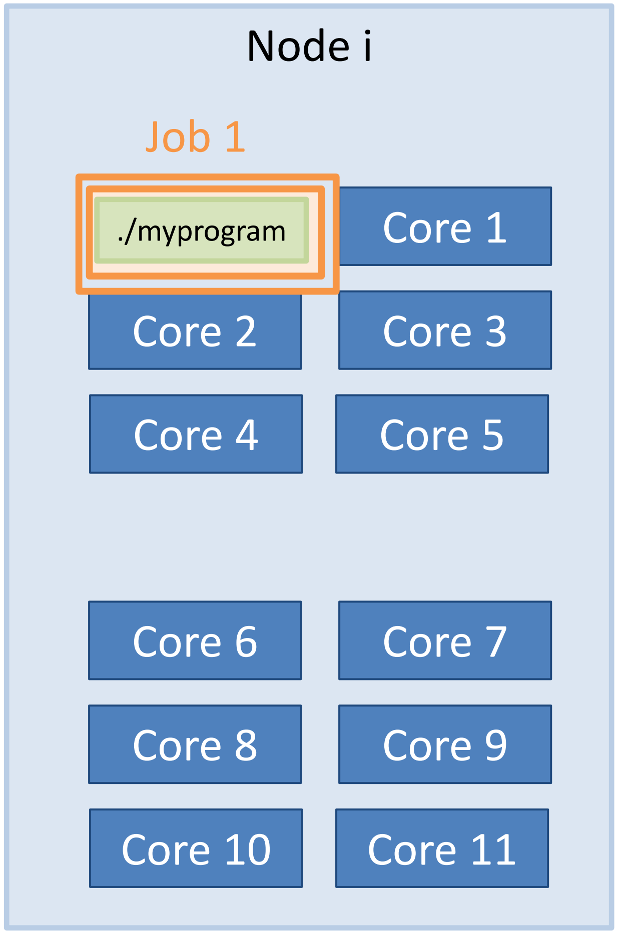
```
#!/bin/bash
#SBATCH --time=0:20:00
#SBATCH --ntasks=1
#SBATCH --partition=short

./myprogram
```



Example job scripts

Simple serial program

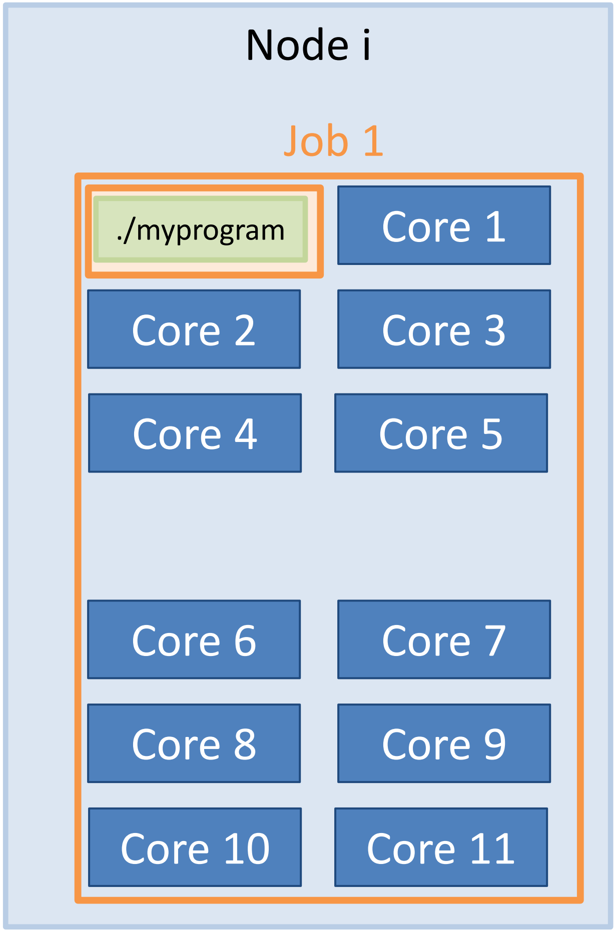
- Simplest possible case
- Number of tasks = number of processes
- Default: only part of RAM used
  - RAM allocation covered later



Example job scripts

Wrong: serial program

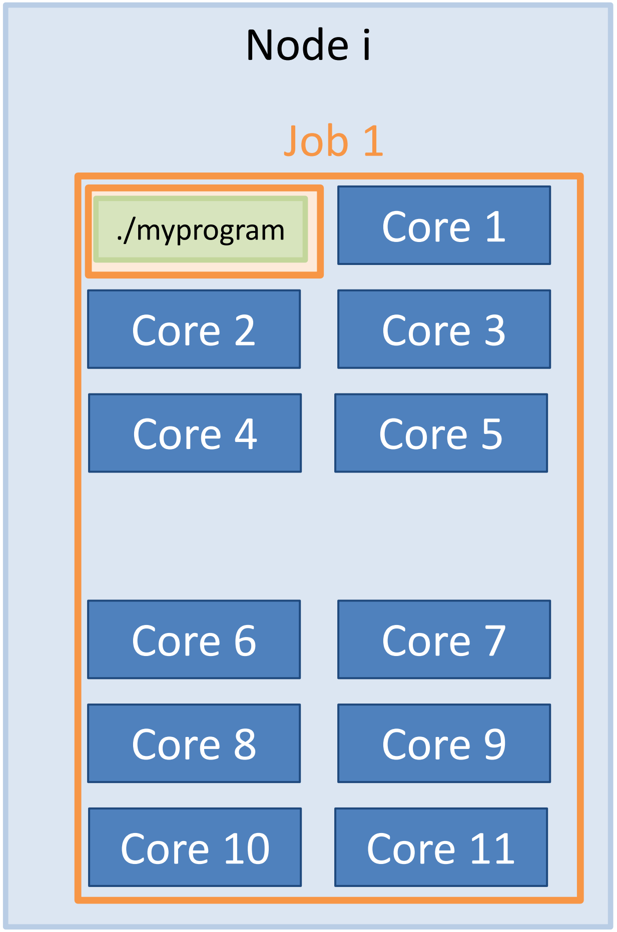
```
#!/bin/bash
#SBATCH --time=0:20:00
#SBATCH --nodes=1
#SBATCH --partition=short

./myprogram
```



Example job scripts

Wrong: serial program

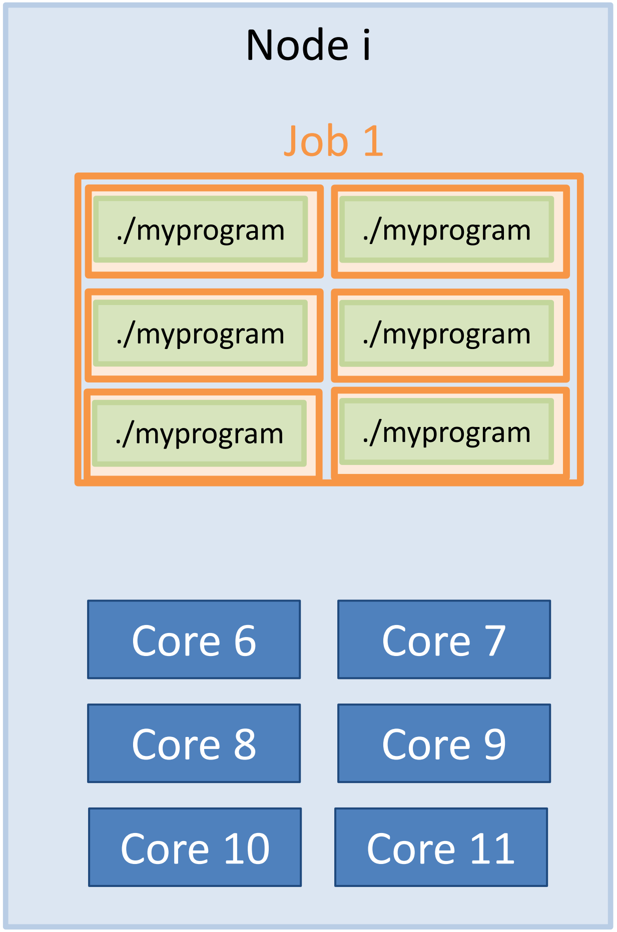
- Waste of resources: full node allocated, only 1 task launched
- Might be legitimate reasons not to use all cores
  - RAM allocation covered later
- But: allocation should reflect reality



Example job scripts

Many instances of serial program

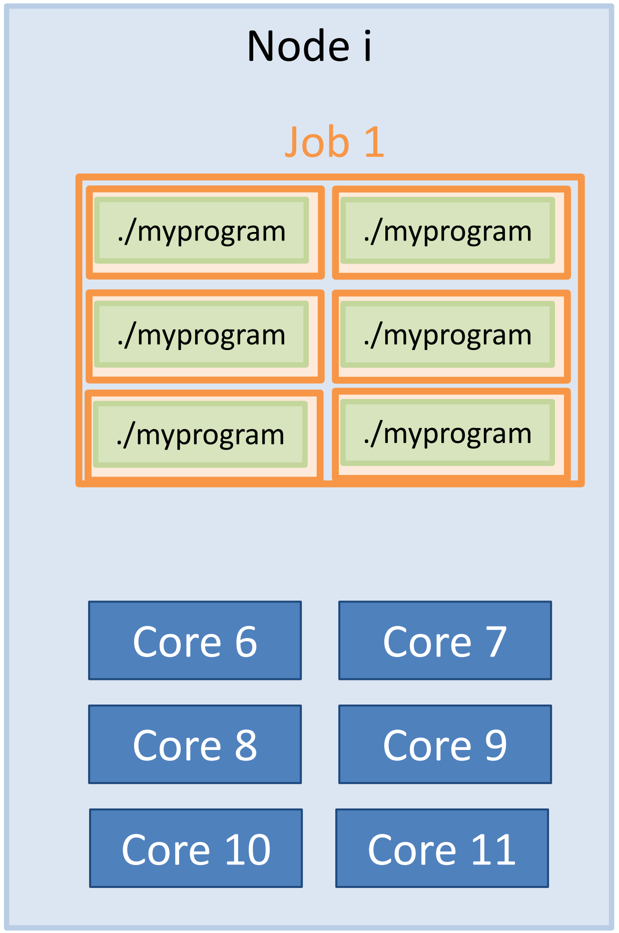
```
#!/bin/bash
#SBATCH --time=0:20:00
#SBATCH --ntasks=1
#SBATCH --partition=short

srun ./myprogram
```



Example job scripts

<u>Many instances of serial program</u>

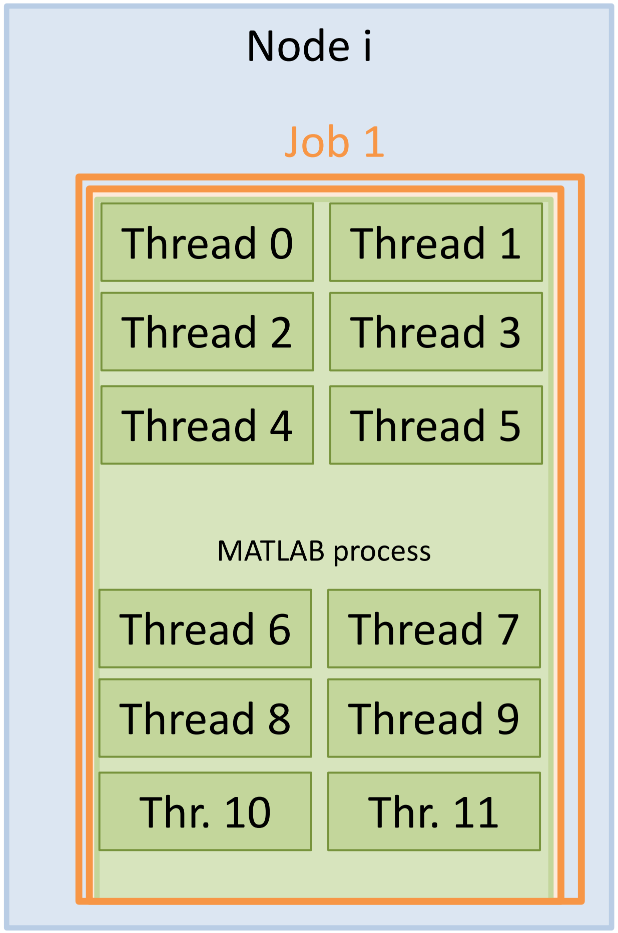
- SLURM simply launches program multiple times
- Cannot talk to each other
- Be careful that they write to different files



Example job scripts

Launching MATLAB

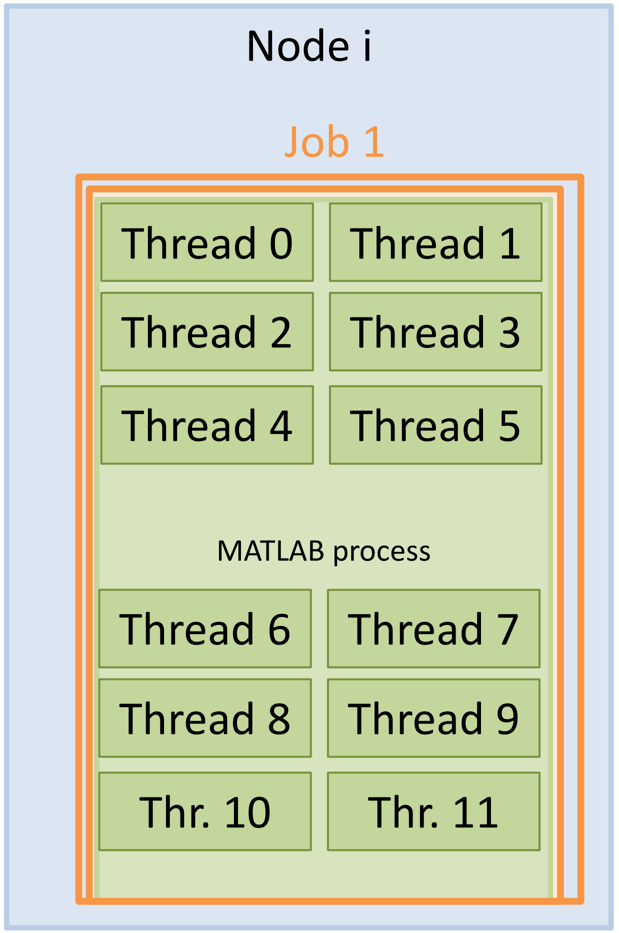
```
#!/bin/bash
#SBATCH --time=0:20:00
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=12
#SBATCH --partition=short

matlab -nodisplay -r myscript
```



Example job scripts

Launching MATLAB

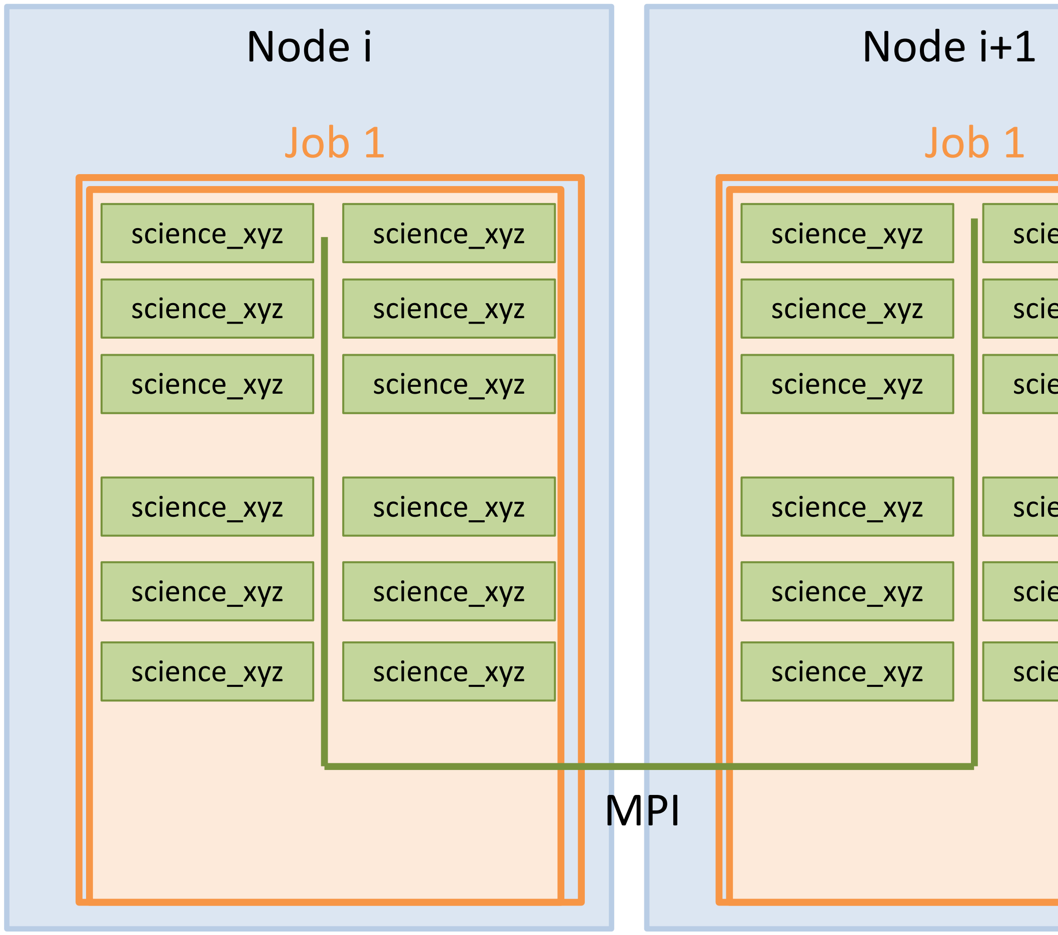
- Commercial applications often multi-threaded
- MATLAB: you do not even need to program differently
- Often good at auto-detecting which resources they have been given
- Does not use MATLAB advanced parallel features (pools), not covered



Example job scripts

Launching scientific application

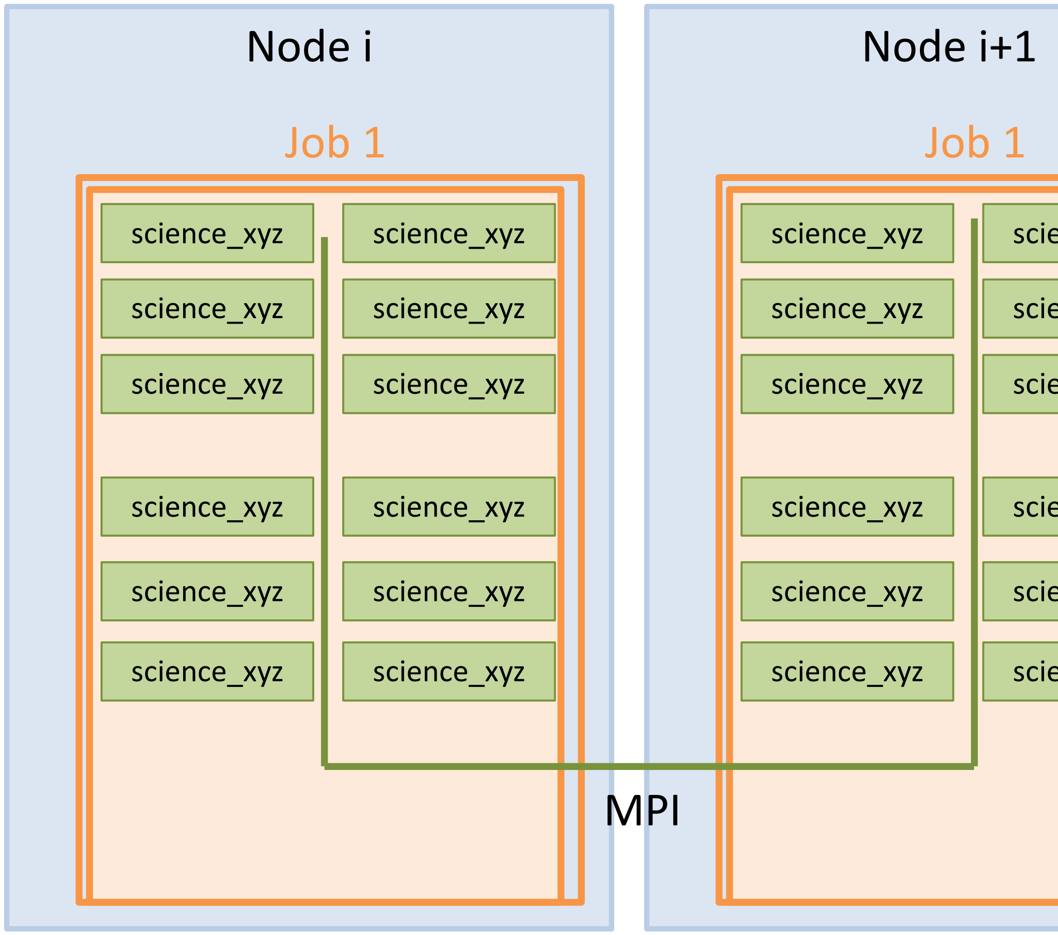
```
#!/bin/bash
#SBATCH --time=4:00:00
#SBATCH --nodes=20
#SBATCH --ntasks-per-node=12

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



Example job scripts

<u>Launching scientific application</u>

- 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`"
- 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>`
  - Only while running
- SLURM config readable to everyone: `scontrol show conf`

Demo 14

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: 64 cores, 256 GB RAM total (240 for apps)
  - Option `--mem 0` 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 15

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 16

Miscellaneous SLURM info

- Job arrays: multiple identical jobs
  - Grouped, don’t pollute queue
  - Max 200 jobs per user on OMNI

- If you see “accounts” mentioned, not used on OMNI

- 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. 100 CPUs on one node)
  - …

Thank you for your attention

Questions?