

Week 1 – Introduction

02613 Python and High-Performance Computing

Today

1. Practical information
2. Python and High-Performance Computing
3. DTU's HPC system

Instructor:

Matthias Bo Stuart



Behind the scenes:

Sven Karlsson



Bernd Damman



Teaching Assistants (TAs):

Christian K. Rasmussen
Juliane B. Budde
Jeppe Klitgaard
Jacob A. Wæver
Magnus G. Pedersen
Rasmus J. Pedersen
Abdulrahman Ramadan
Michail Dikaiopoulos
Erik B. Christensen
Lucas Pedersen
Alba G. Primo

The screenshot shows a course page for "02613 Python and high-performance computing". The top navigation bar includes links for Course Home, Content, Autolab, Piazza, Video & Streaming, and a search bar. The user is identified as "Patrick Møller Jensen as Student". The main content area features a section titled "Overview and Resources" which is currently expanded, showing links to "Connecting to an HPC Terminal", "File Transfer to/from HPC", "Linux Terminal Cheat Sheet", and "Initializing the 02613". To the right, the "Overview and Resources" section is displayed under the heading "Overview and Resources". It welcomes users to the course and provides practical information, including lecture times (08:30 - 10:00 Building 116, Auditorium 81), attendance requirements (not mandatory, but strongly recommended), slide releases (every Wednesday under Week X in Contents), and streaming video availability. Exercises are listed as 10:00 - 12:00 in various rooms and foyer areas.

02613 Python and high-performan...

Course Home Content Autolab Piazza Video & Streaming

Search titles, descriptions

Overview and Resources

Connecting to an HPC Terminal

File Transfer to/from HPC

Linux Terminal Cheat Sheet

Initializing the 02613

Patrick Møller Jensen
as Student

Overview and Resources

Welcome to 02613 Python and High-Performance Computing!

On this page we will list practical information for the course as well as links to all the course material. It will be updated throughout the semester as more information becomes available.

Lectures: 08:30 - 10:00 Building 116, Auditorium 81

Attendance: not mandatory, but *strongly recommended*.

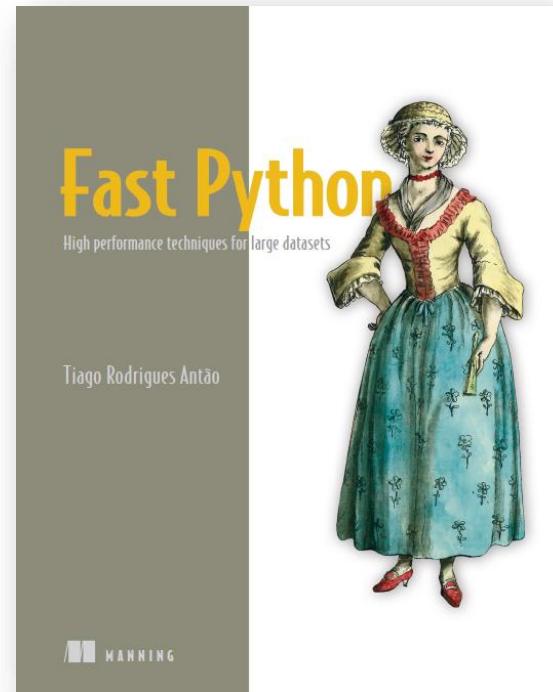
Slides: released every Wednesday under Week X in Contents.

Streaming and Video: available from the [Streaming and Video](#) tab.

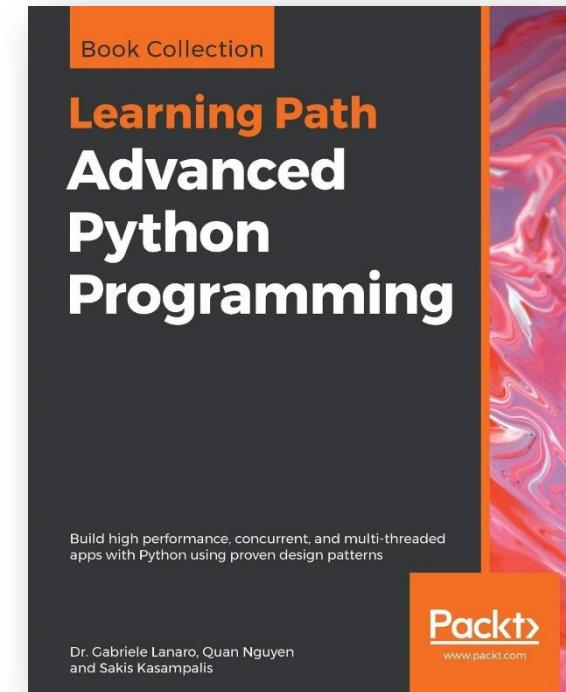
Exercises: 10:00 - 12:00 Building 116, Rooms 010, 012, 016, 018, 019 and Foyer area north, east and west.

Lectures: Wednesdays 8:30 – 10:00, Building 116, Auditorium 81
Streamed online using Panopto

Books: Fast Python



Advanced Python Programming
(free online)



The screenshot shows a course page on the learn.inside.dtu.dk platform. At the top, there's a header with the DTU logo, the course title '02613 Python and high-performance computing', and various navigation icons like a grid, mail, message, and notifications. A user profile for 'Patrick Møller Jensen as Student' is also visible. Below the header, a navigation bar includes links for 'Course Home', 'Content', 'Autolab', 'Piazza', 'Video & Streaming', and a search bar. The main content area features a large section titled 'Week 1: Introduction to PyHPC'. This section includes a 'Reading (optional)' note pointing to Chapter 1 of 'Fast Python', a description of the course focus on running scripts, and a link to a specific introduction article. Three other weeks are listed below: 'Week 1: Introduction to PyHPC' (highlighted with a blue border), 'Week 2: Python Bootcamp', and 'Week 3: The Memory Hierarchy'.

02613 Python and high-performance computing

Patrick Møller Jensen
as Student

Course Home Content Autolab Piazza Video & Streaming

Search titles, descriptions

Overview and Resources

Week 1: Introduction to PyHPC

Reading (optional): Ch. 1 in Fast Python.

In this course, we will focus on running Python scripts instead of interactively, e.g., in a Jupyter notebook. If you have not tried this before, there is an introduction here:

<https://greenteapress.com/thinkpython2/html/thinkpython2003.html#sec19>

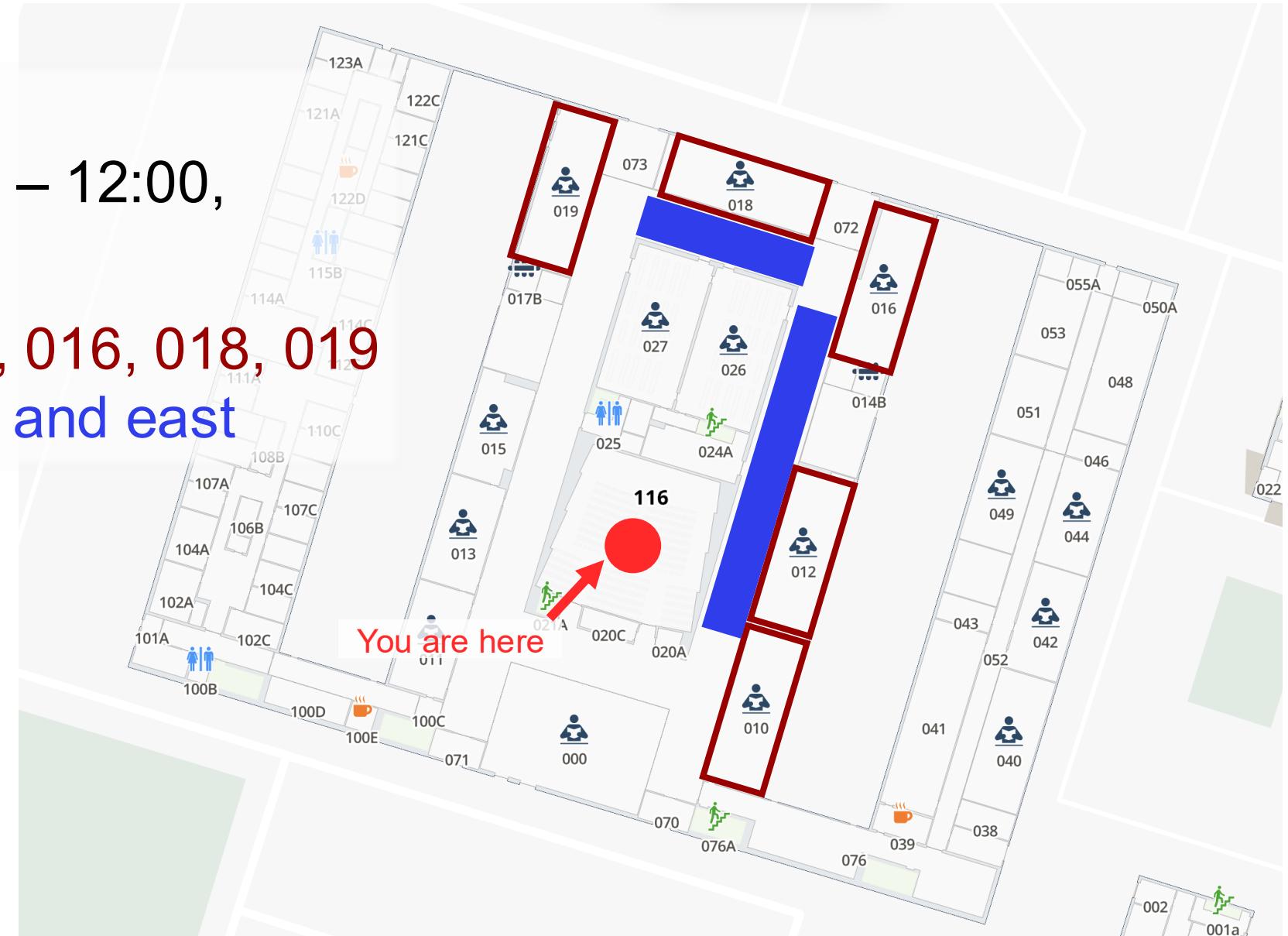
Week 2: Python Bootcamp

Week 3: The Memory Hierarchy

Exercises:

Wednesdays 10:00 – 12:00,
Building 116

Rooms 010, 012, 016, 018, 019
Foyer area north and east





Exercises: Released every Wednesday

Exercises for Week 1

Solutions: Released following Tuesday

The focus of this exercise is to learn how to connect to the HPC nodes, transfer files and run code on the HPC, preferably using the terminal.

1. **Autolab** Write a Python program that writes "Hello world" to a file (e.g., called 'content.txt'). It should also print the same text to the screen.
2. Transfer your Python file to the HPC using scp, sftp, FileZilla, or other (see guide [File Transfer to/from HPC](#) on Learn).
3. Connect to an HPC terminal (see guide [Connecting to an HPC Terminal](#) on Learn). If you are *not on a DTU network*, you may need to use a VPN or set up SSH keys - see the guide. If you used SSH, make sure you called `linuxsh` so you are *not on a login node*.

Lecture_week01

Week 1 Exercises

Starts 5 Feb

No influence on final grade, but

must complete at least 20 to attend exam and
1 from every week

Section:

Number of grace days remaining: 0

COURSE WEBSITE

GRADEBOOK

Deadline: 11th of May at 6 AM (end of semester)

Assessments

Autolab exercises not included in solutions

Hello World

Python script for exercise 1.1 in week 1

Admin Options

- [Edit assessment](#)
- [View Gradesheet](#)
- [Export assessment](#)
- [Autograder settings](#)
- [Manage extensions](#)
- [Manage submissions](#)
- [View statistics](#)
- [Bulk import grades](#)
- [Bulk export grades](#)

Danger Zone

Due: May 12th 2025, 11:59 pm CEST (UTC +02:00)

Last day to handin: May 12th 2025, 11:59 pm CEST (UTC +02:00)

Drag a file here to hand in. Click to select a file.
Files do not submit automatically.

Handin format: .py files

I affirm that I have complied with this course's academic integrity policy as defined in the syllabus.

SUBMIT

(unlimited submissions left)

Submission Summary

Ver	File	Submission Date	Correctness (100.0)	Days Late (Grace Days Used)	Total Score (100.0)	Tweak	Instructor Actions
1	View Source Download Submission	2025-01-07 16:35:03 +0100	100.0	0 days (0 days)	100.0	-	(Regrade) (Destroy) (Edit)

Autograding Result for Hello World - Correctness (patmjen@dtu.dk)

Feedback

Completed

Autograder [Tue Jan 14 12:10:13 2025]: Received job 02613-F25_Hello-World_1_patmjen@dtu.dk:1
Autograder [Tue Jan 14 12:10:15 2025]: Success: Autodriver returned normally
Autograder [Tue Jan 14 12:10:15 2025]: Here is the output from the autograder:

Autodriver: Job exited with status 0

Running submission file...

Checking result...

- * Ran successfully: True
- * Printed 'Hello world': True
- * Printed nothing on stderr: True
- * Made new file: True
- * File content is same as printed: True

{"scores": {"Correctness": 100.0}}

Results

Autograder Scores

Correctness: 100.0

Autograding Total: 100.0

Paste in code

Paste your function (and only your function) here:

```
def listsum(arr):  
    # Your code
```

I affirm that I have complied with this course's academic integrity policy as defined in the syllabus.

SUBMIT QUERY

Quizzes

What does HPC stand for?

- Holistic Program Computations
- High Performance Compute
- Half Price Cost

What is the Python language named after?

- The snake!
- Monty Python's Flying Circus
- Nobody knows...

I affirm that I have complied with this course's academic integrity policy as defined in the syllabus.

SUBMIT QUERY

piazza.com (link on Learn)

The screenshot shows the Piazza interface for the class DTU_F 25_02613. The top navigation bar includes links for LIVE Q&A, Drafts, week01 through week11, and more. A search bar and a 'New Post' button are also present. The main content area displays a 'Class at a Glance' section with a lock icon, updated 25 seconds ago, and a 'Reload' button. It shows 'no unread posts' and indicates an 'active instructor license'. A large red-bordered box contains two main messages: one about inappropriate questions and another about sensitive information.

Question not appropriate for Piazza?
02613@compute.dtu.dk (me + TAs)

Question with “sensitive” info?
Write me (mbst@dtu.dk)

We are pleased to introduce an optional AI-powered summarization feature to help your students get more out of discussions, especially in larger classes. This feature is disabled by default and per your Institution's policies, you can enable and disable this feature any time in the Manage Class tab under 'Q&A Settings'.

Followup Summarization: If enabled, any post with 5 or more followup comments will display a 'Summarize' button.

Mini-Project

Goal: work with course topics in a more “real” way

Work in groups of 3 or 4

Will be released in week 4

Hand-in: Small report on

Sunday 3rd of May, 2026 at 23:59 (week 12)

Must be **completed and approved** to attend exam

Exam:

4 hours written (digital) exam

All aids and materials permitted - without internet access.

Laptop, tablet, and similar devices allowed.

Generative AI is not allowed.

Previous exams + solutions available on Learn.

Full details on learn

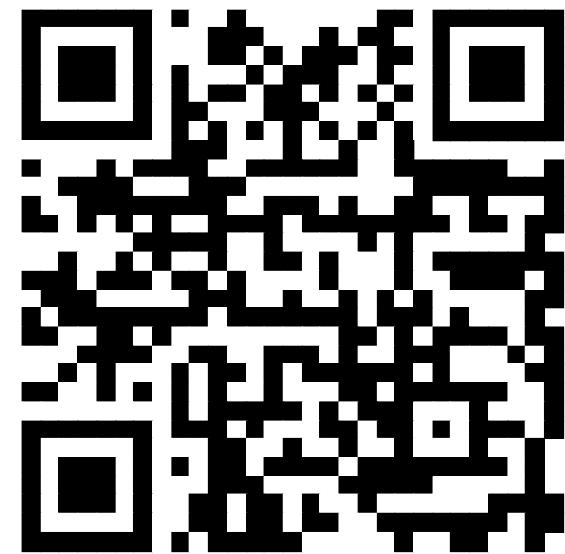
Questions?

Join the Vevox session

Go to **vevox.app**

Enter the session ID: **113-073-658**

Or scan the QR code





#/#/#

Join at: vevox.app

ID: XXX-XXX-XXX

Question slide



What do you work with?

Machine Learning and Deep Learning



##.##%

Statistics



##.##%

Modelling and Simulation



##.##%

Optimization



##.##%

Robotics



##.##%

Other



##.##%



#/#/#

Join at: vevox.app

ID: XXX-XXX-XXX

Results slide



What do you work with?

Machine Learning and Deep Learning



##.##%

Statistics



##.##%

Modelling and Simulation



##.##%

Optimization



##.##%

Robotics



##.##%

Other



##.##%

RESULTS SLIDE



#/#/#

Join at: vevox.app

ID: XXX-XXX-XXX

Question slide



What is your experience with Python

Never used it



##.##%

Used it once



##.##%

Used it a little, e.g., for one course



##.##%

Used it a lot

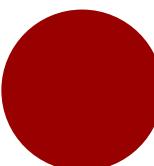


##.##%

"I'm literally a snake!"



##.##%





#/#/#

Join at: vevox.app

ID: XXX-XXX-XXX

Results slide



What is your experience with Python

Never used it



##.##%

Used it once



##.##%

Used it a little, e.g., for one course



##.##%

Used it a lot



##.##%

"I'm literally a snake!"



##.##%

RESULTS SLIDE



#/#/#

Join at: vevox.app

ID: XXX-XXX-XXX

Question slide



What is your experience with the Linux terminal?

Never used it



##.##%

Used it once



##.##%

Used it a little, e.g., for one course



##.##%

Used it a lot

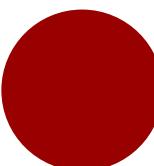


##.##%

"I stream Netflix as ASCII art!"



##.##%





#/#/#

Join at: vevox.app

ID: XXX-XXX-XXX

Results slide



What is your experience with the Linux terminal?

Never used it



##.##%

Used it once



##.##%

Used it a little, e.g., for one course



##.##%

Used it a lot



##.##%

"I stream Netflix as ASCII art!"



##.##%

RESULTS SLIDE



#/#/#

Join at: vevox.app

ID: XXX-XXX-XXX

Question slide



What is your experience with DTU's HPC systems?

Never used it



##.##%

Used ThinLinc once or twice



##.##%

Used it in interactive mode



##.##%

Used the batch job system

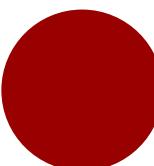


##.##%

"I bsub my breakfast!"



##.##%





#/#/#

Join at: vevox.app

ID: XXX-XXX-XXX

Results slide



What is your experience with DTU's HPC systems?

Never used it



##.##%

Used ThinLinc once or twice



##.##%

Used it in interactive mode



##.##%

Used the batch job system



##.##%

"I bsub my breakfast!"



##.##%

RESULTS SLIDE

Who we expect:



Experience:
Expected



Experience:
Nice but not needed

Today's lecture

1. Practical information

2. Python and High-Performance Computing

3. DTU's HPC system

What is high-performance computing?



##/##

Join at: vevox.app

ID: XXX-XXX-XXX

Question slide



What does High Performance Computing mean to you?



##/##

Join at: vevox.app

ID: XXX-XXX-XXX

Results slide



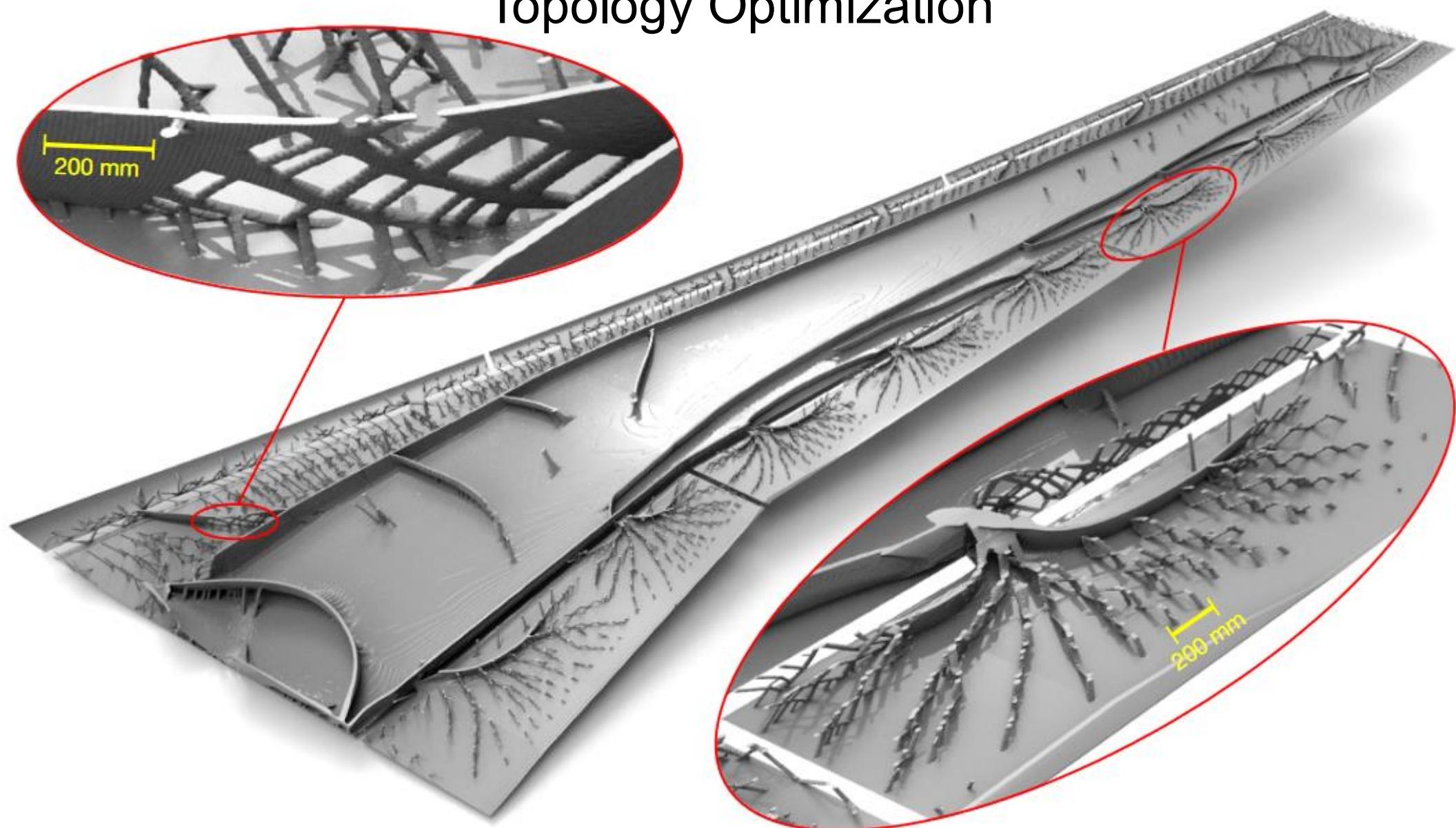
What does High Performance Computing mean to you?

RESULTS SLIDE



Why high-performance computing?

Topology Optimization



Wind Turbine Design





ChatGPT



GitHub
Copilot

Why me?





Course goals

- Identify slow code and improve its performance
- Understand how hardware affects performance
- Efficiently utilize single core, multi core and GPU resources
- Use a modern HPC system (not just your own laptop/PC)

Official list on course base page

Week by week

Week 1	Intro meeting, diagnostic test, HPC intro	CPU
Week 2	Python primer and running Python on HPC	
Week 3	Memory hierarchy: caches, disks, etc.	
Week 4	Profiling and High-Performance NumPy	(mini-project released)
Week 5 + 6	Parallelism: Scalability and Amdahl's law	
Week 7	High-Performance Pandas and Apache Arrow	
Week 8	Storing Big Data	
Week 9	Numba and GPU Computing	GPU
Week 10	CuPy and GPU profiling	
Week 11	HPC Workflows	HPC Workflows
Week 12	Project Work (hand-in Sunday)	
Week 13	Common HPC Pitfalls and Q&A	

Today's lecture

1. Practical information

2. Python and High-Performance Computing

3. DTU's HPC system



What is an HPC cluster?



How do I access it!?



Your computer



Your computer

SSH or ThinLinc

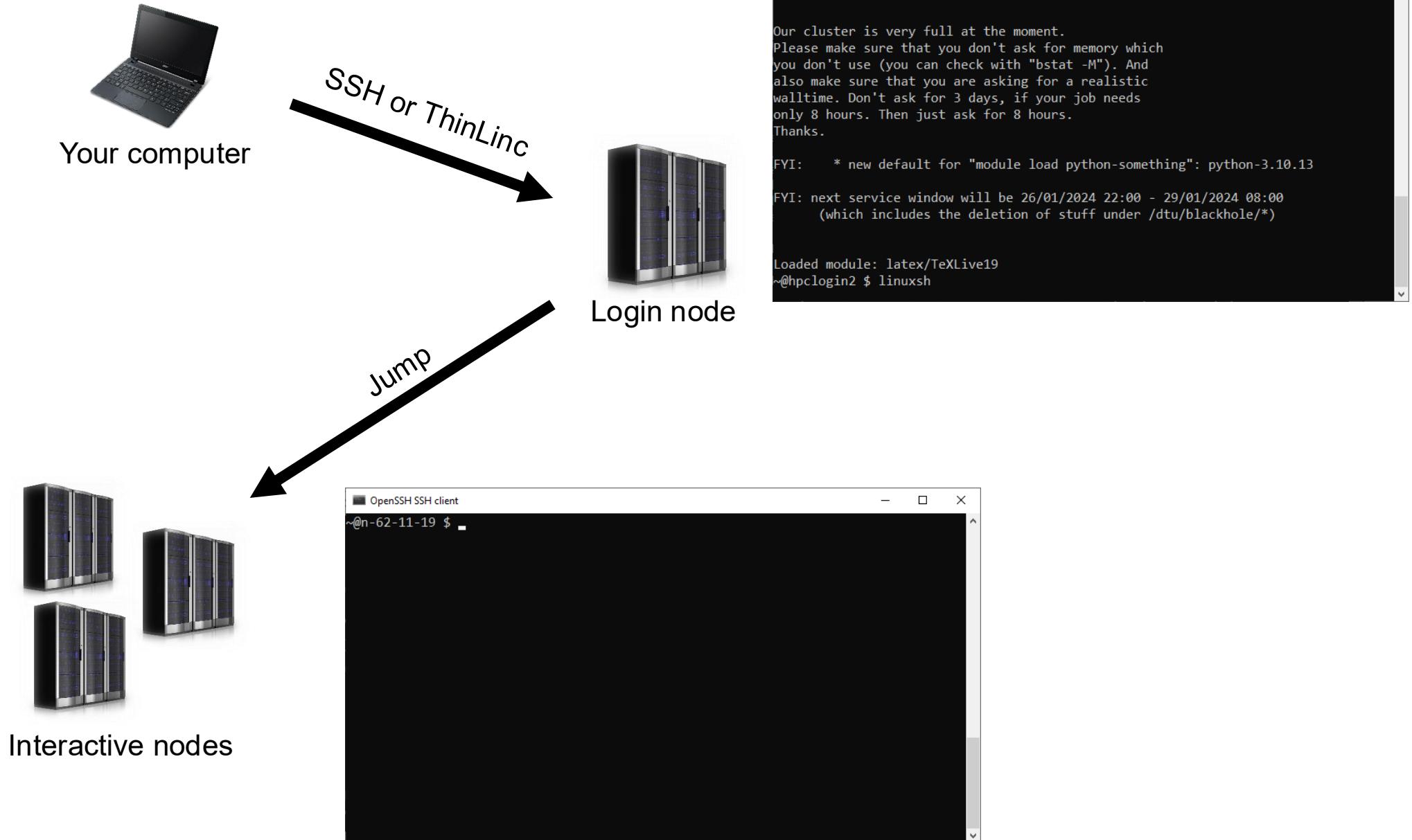


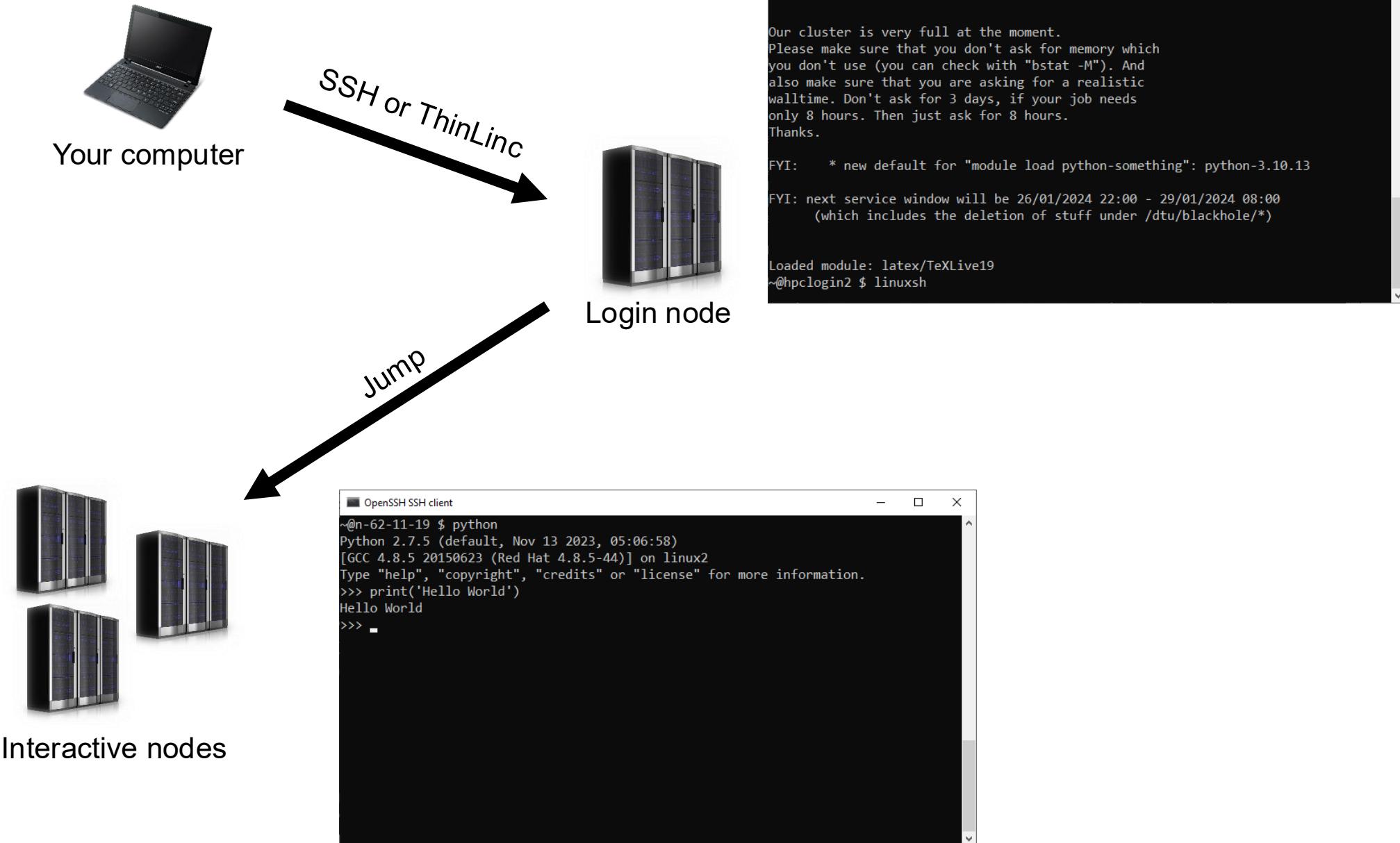
Login node

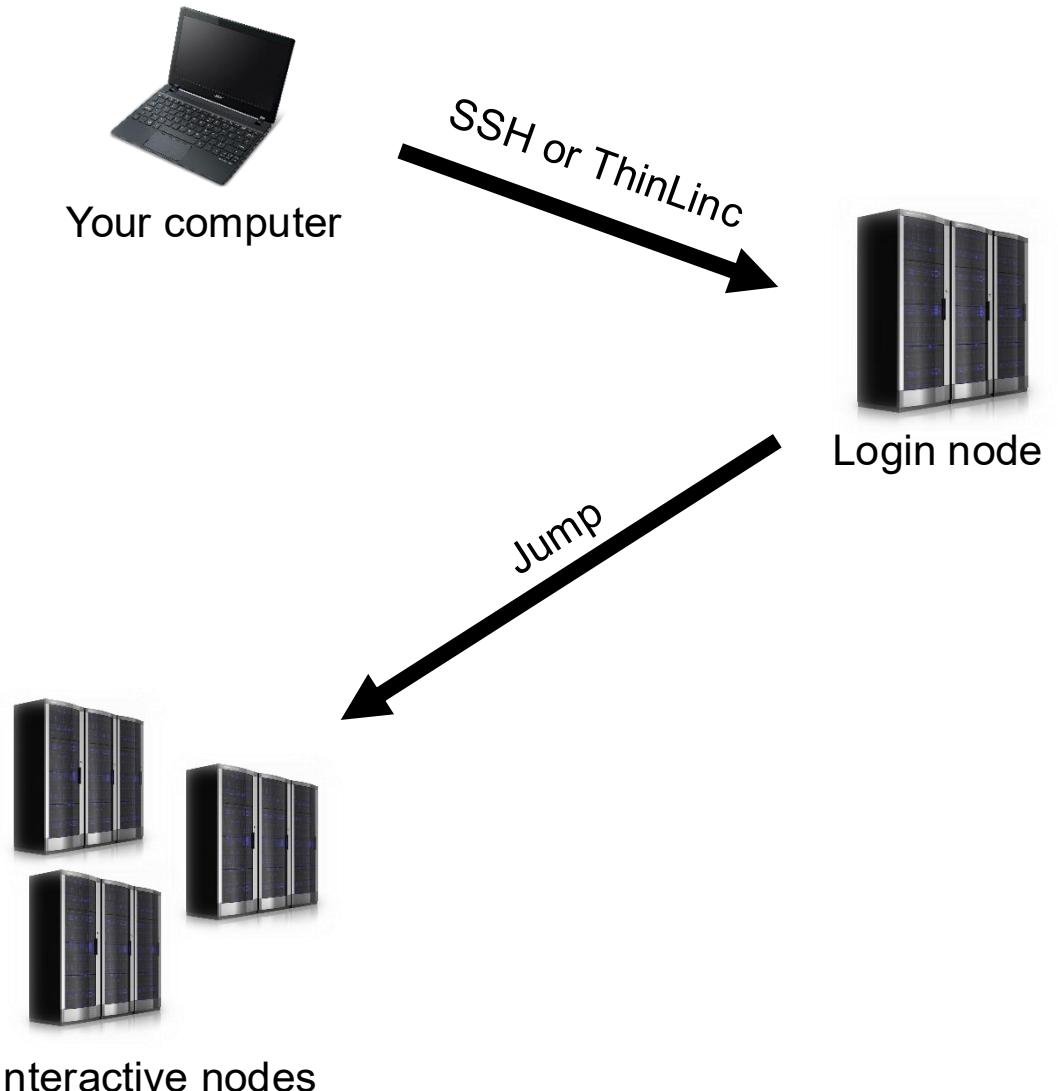
node = a computer

A screenshot of an OpenSSH SSH client window. The terminal output includes:

```
***  
Our cluster is very full at the moment.  
Please make sure that you don't ask for memory which  
you don't use (you can check with "bstat -M"). And  
also make sure that you are asking for a realistic  
walltime. Don't ask for 3 days, if your job needs  
only 8 hours. Then just ask for 8 hours.  
Thanks.  
  
FYI: * new default for "module load python-something": python-3.10.13  
  
FYI: next service window will be 26/01/2024 22:00 - 29/01/2024 08:00  
(which includes the deletion of stuff under /dtu/blackhole/*)  
  
Loaded module: latex/TeXLive19  
~@hpclogin2 $ linuxsh
```

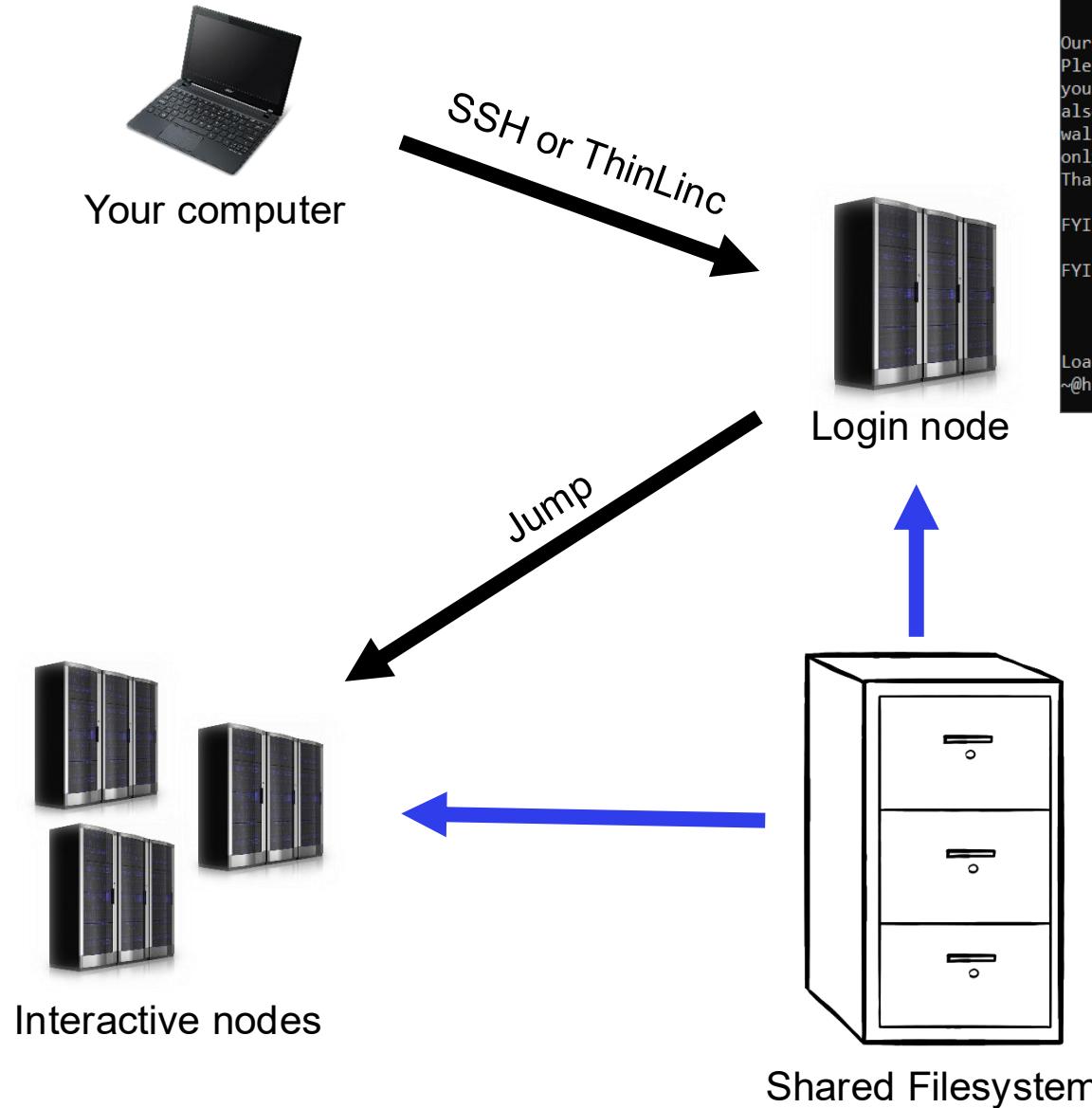






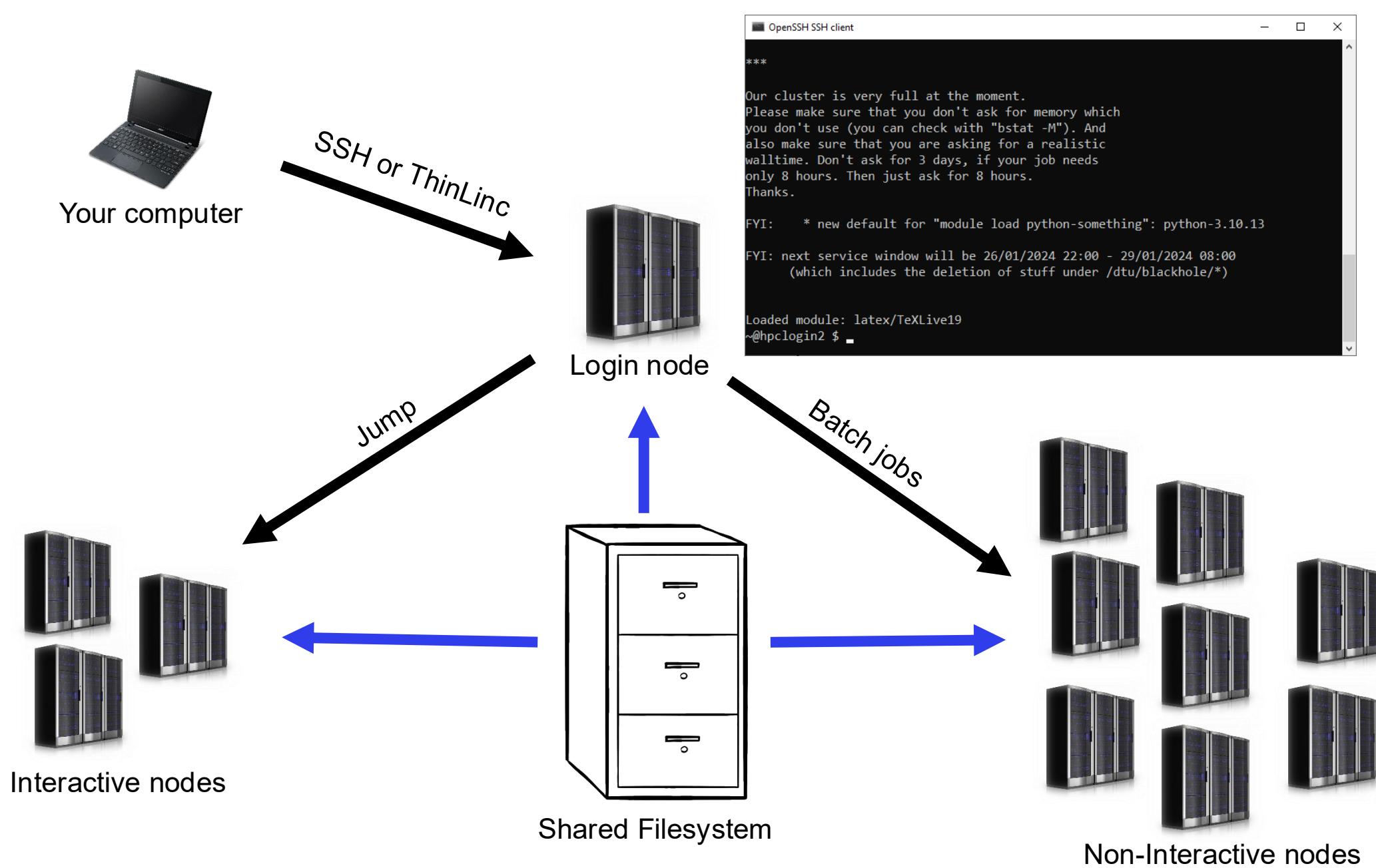
OpenSSH SSH client

```
***  
Our cluster is very full at the moment.  
Please make sure that you don't ask for memory which  
you don't use (you can check with "bstat -M"). And  
also make sure that you are asking for a realistic  
walltime. Don't ask for 3 days, if your job needs  
only 8 hours. Then just ask for 8 hours.  
Thanks.  
  
FYI: * new default for "module load python-something": python-3.10.13  
  
FYI: next service window will be 26/01/2024 22:00 - 29/01/2024 08:00  
(which includes the deletion of stuff under /dtu/blackhole/*)  
  
Loaded module: latex/TeXLive19  
~@hpclogin2 $ -
```



OpenSSH SSH client

```
***  
Our cluster is very full at the moment.  
Please make sure that you don't ask for memory which  
you don't use (you can check with "bstat -M"). And  
also make sure that you are asking for a realistic  
walltime. Don't ask for 3 days, if your job needs  
only 8 hours. Then just ask for 8 hours.  
Thanks.  
  
FYI: * new default for "module load python-something": python-3.10.13  
  
FYI: next service window will be 26/01/2024 22:00 - 29/01/2024 08:00  
(which includes the deletion of stuff under /dtu/blackhole/*)  
  
Loaded module: latex/TeXLive19  
~@hpclogin2 $ -
```



Batch jobs: the simplest script

```
#!/bin/bash  
sleep 60  
submit.sh
```



```
$ bsub < submit.sh
```

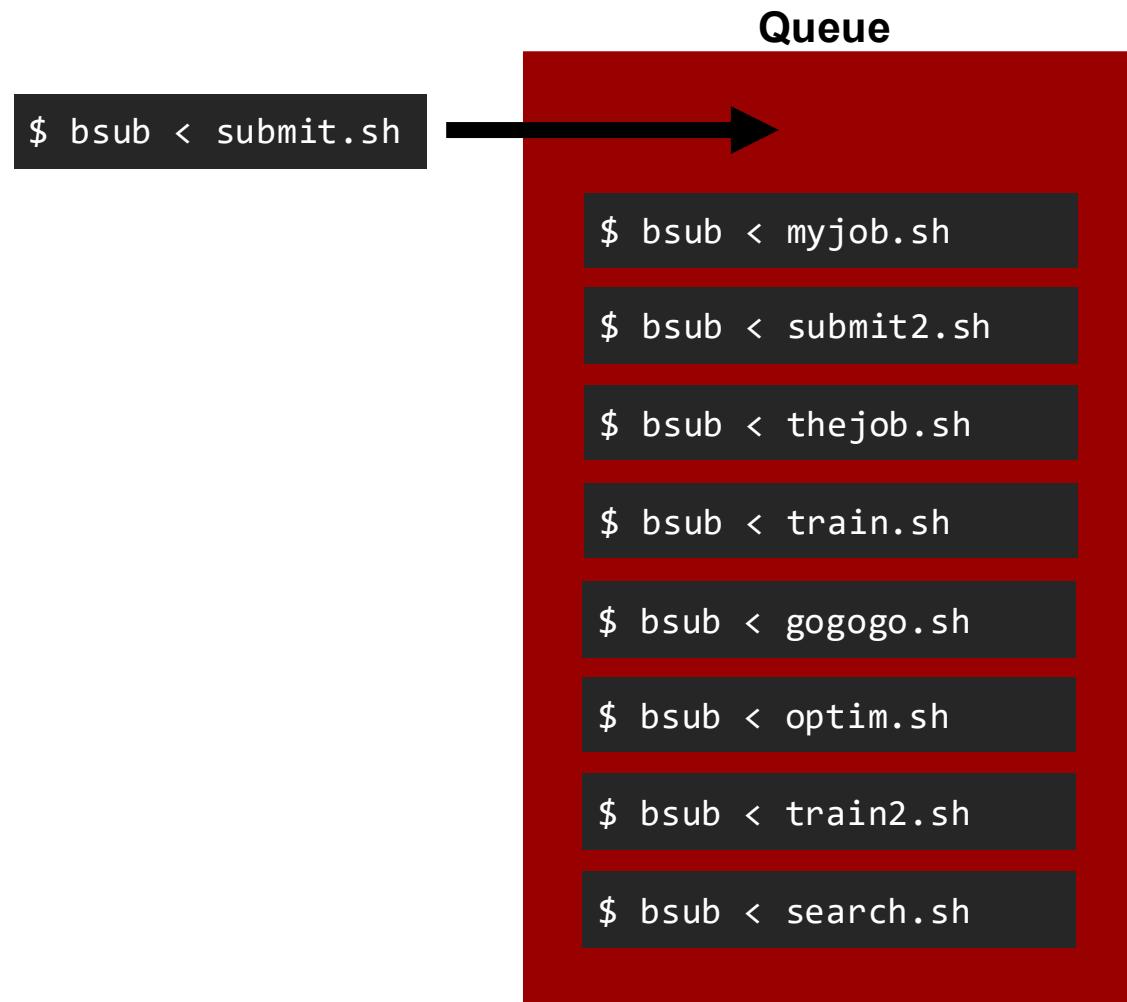
Batch jobs: the simplest script

```
#!/bin/bash  
sleep 60  
submit.sh
```

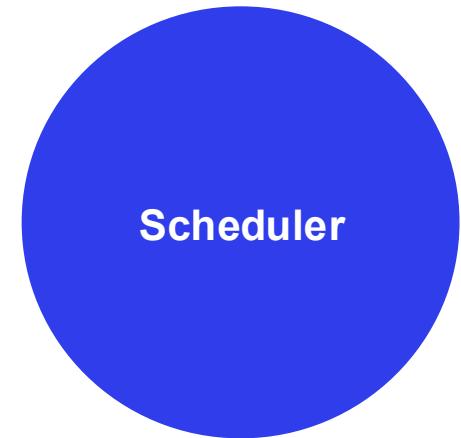
```
$ bsub < submit.sh  
Job <702572> is submitted to default queue <hpc>.  
$
```

What happens now?

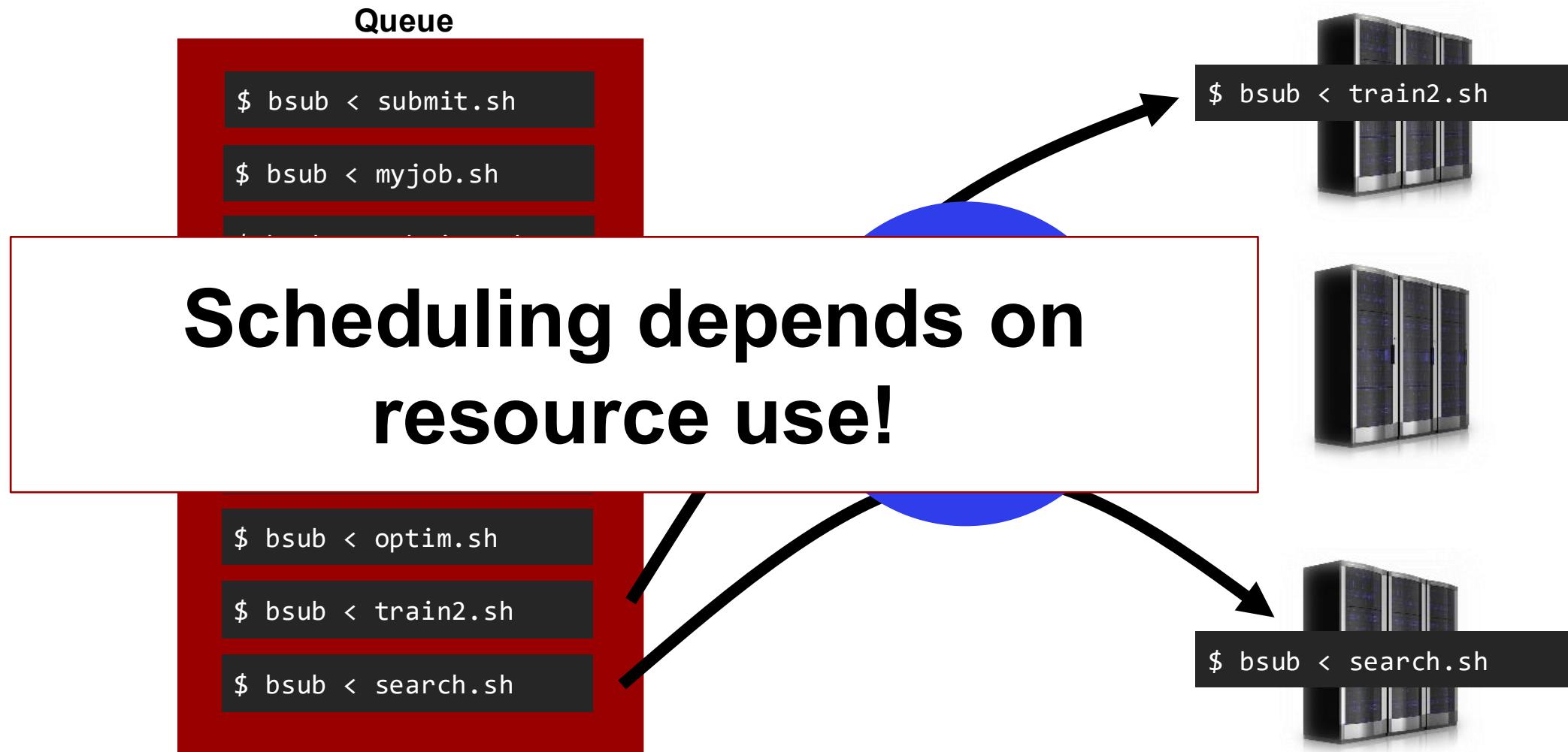
Interlude: queues and resource managers



Interlude: queues and resource managers



Interlude: queues and resource managers



Interlude: queues and resource managers

#	CPU	RAM	GPU	
31 x	2x Xeon E5 2650 v4	252 – 504 GB		hpc
46 x	2x Xeon E5 2660 v3	126 GB		
24 x	2x Xeon Gold 6126	189 – 756 GB		
4 x	2x Xeon Gold 6142	378 GB		
26 x	2x Xeon Gold 6226R	378 – 756 GB		
4 x	2x Xeon Gold 6342	504 GB		
8x	2x Xeon Gold 6126	189 – 378 GB	2x NVIDIA V100 16-32GB	gpuv100
3x	2x Xeon Gold 6142	378 GB	4x NVIDIA V100 32GB	
4x	2x Xeon Gold 6242	378 GB	2x NVIDIA V100 32GB	
5x	2x Xeon Gold 6226R	756 GB	2x NVIDIA A100 40GB	gpua100
6x	2x Xeon Gold 6326	1008 GB	2x NVIDIA A100 80GB	

... and more

Batch jobs: the simplest script

```
#!/bin/bash  
sleep 60  
submit.sh
```

```
$ bsub < submit.sh  
Job <702572> is submitted to default queue <hpc>.  
$
```

Batch jobs: the simplest script

```
#!/bin/bash  
sleep 60  
submit.sh
```

```
$ bsub < submit.sh  
Job <702572> is submitted to default queue <hpc>.  
$ bstat  
JOBID USER QUEUE JOB_NAME SLOTS STAT START_TIME ELAPSED  
702572 patmjen hpc NONAME 1 PEND - 0:00:00  
$
```

Batch jobs: the simplest script

```
#!/bin/bash  
sleep 60  
submit.sh
```

```
$ bsub < submit.sh  
Job <702572> is submitted to default queue <hpc>.  
$ bstat  
JOBID USER QUEUE JOB_NAME SLOTS STAT START_TIME ELAPSED  
702572 patmjen hpc NONAME 1 RUN Dec 13 12:17 0:00:00  
$
```

Batch jobs: the simplest script

```
#!/bin/bash  
sleep 60  
submit.sh
```

```
$ bsub < submit.sh  
Job <702572> is submitted to default queue <hpc>.  
$ bstat  
JOBID USER QUEUE JOB_NAME SLOTS STAT START_TIME ELAPSED  
702572 patmjen hpc NONAME 1 RUN Dec 13 12:17 0:00:00  
$ bjobs  
JOBID USER QUEUE JOB_NAME SLOTS STAT START_TIME TIME_LEFT  
702572 patmjen hpc NONAME 1 RUN Dec 13 12:17 00:02:00 L  
$
```

Batch jobs: the simplest script

```
#!/bin/bash  
sleep 60  
submit.sh
```

```
$ bsub < submit.sh  
Job <702572> is submitted to default queue <hpc>.  
$ bstat  
JOBID USER QUEUE JOB_NAME SLOTS STAT START_TIME ELAPSED  
702572 patmjen hpc NONAME 1 RUN Dec 13 12:17 0:00:00  
$ bjobs  
JOBID USER QUEUE JOB_NAME SLOTS STAT START_TIME TIME_LEFT  
702572 patmjen hpc NONAME 1 RUN Dec 13 12:17 00:02:00 L  
$ ls -1  
NONAME_702572.out  
submit.sh  
$
```

Batch jobs: the simplest script

```
#!/bin/bash  
sleep 60  
submit.sh
```

```
$ bsub < submit.sh
```

Batch jobs: the simplest script

```
#!/bin/bash  
sleep 60  
submit.sh
```

```
$ bsub < submit.sh  
bsub info: Job has no name! Setting it to NONAME!  
bsub info: Job has no wall-clock time! Setting it to 15 minutes!  
bsub info: Job has no output file! Setting it to NONAME_%J.out!  
bsub info: Job has no memory requirements! Setting it to 1024 MB!  
bsub info: You need to specify at least -R "rusage[mem=...]"!  
Job <702572> is submitted to default queue <hpc>.  
$
```

Need to specify resources!

Batch jobs: the simple script

```
#!/bin/bash  
  
Job name #BSUB -J sleeper  
Queue name #BSUB -q hpc  
Wall-clock time #BSUB -W 2  
Resources (mem) #BSUB -R "rusage[mem=512MB]"  
Output file (stdout) #BSUB -o sleeper_%J.out  
Output file (stderr) #BSUB -e sleeper_%J.err  
  
sleep 60  
submit.sh
```

```
$ bsub < submit.sh  
Job <702573> is submitted to queue <hpc>.  
$
```

Batch jobs: the simple script

```
#!/bin/bash  
  
Job name #BSUB -J sleeper  
Queue name #BSUB -q hpc  
Wall-clock time #BSUB -W 2  
Resources (mem) #BSUB -R "rusage[mem=512MB]"  
Output file (stdout) #BSUB -o sleeper_%J.out  
Output file (stderr) #BSUB -e sleeper_%J.err  
  
sleep 60  
submit.sh
```

```
$ bsub < submit.sh  
Job <702573> is submitted to queue <hpc>.  
$ ls -1  
sleeper_702573.out  
sleeper_702573.err  
submit.sh  
$
```

Batch jobs: output files

```
#!/bin/bash
Job name #BSUB -J sleeper
Queue name #BSUB -q hpc
Wall-clock time #BSUB -W 2
Resources (mem) #BSUB -R "rusage[mem=512MB]"
Output file (stdout) #BSUB -o sleeper_%J.out
Output file (stderr) #BSUB -e sleeper_%J.err

sleep 60
submit.sh
```

<printed output if any>

Sender: LSF System <lsfadmin@hpc.dtu.dk>
Subject: Job 19953623: <sleeper> in cluster <dcc> Done

Job <sleeper> was submitted from host <n-62-11-19> by user <patmjen> in cluster <dcc> at Mon Jan 22 11:11:23 2024
Job was executed on host(s) <n-62-31-4>, in queue <hpc>, as user <patmjen> in cluster <dcc> at Mon Jan 22 11:11:25 2024
</zhome/9d/d/98006> was used as the home directory.
</zhome/9d/d/98006/Documents/02613> was used as the working directory.

Started at Mon Jan 22 11:11:25 2024
Terminated at Mon Jan 22 11:12:27 2024
Results reported at Mon Jan 22 11:12:27 2024

... more

sleeper_702573.out

Batch jobs: output files

```
#!/bin/bash  
  
Job name #BSUB -J sleeper  
Queue name #BSUB -q hpc  
Wall-clock time #BSUB -W 2  
Resources (mem) #BSUB -R "rusage[mem=512MB]"  
Output file (stdout) #BSUB -o sleeper_%J.out  
Output file (stderr) #BSUB -e sleeper_%J.err  
  
sleep 60  
  
submit.sh
```

...
Your job looked like:

```
-----  
# LSBATCH: User input  
#!/bin/bash  
#BSUB -J sleeper  
#BSUB -q hpc  
#BSUB -W 2  
#BSUB -R "rusage[mem=512MB]"  
#BSUB -o sleeper_%J.out  
#BSUB -e sleeper_%J.err  
  
sleep 60
```

... more

sleeper_702573.out

Batch jobs: output files

```
#!/bin/bash
Job name #BSUB -J sleeper
Queue name #BSUB -q hpc
Wall-clock time #BSUB -W 2
Resources (mem) #BSUB -R "rusage[mem=512MB]"
Output file (stdout) #BSUB -o sleeper_%J.out
Output file (stderr) #BSUB -e sleeper_%J.err

sleep 60
submit.sh
```

...

Successfully completed.

Resource usage summary:

CPU time :	0.82 sec.
Max Memory :	4 MB
Average Memory :	4.00 MB
Total Requested Memory :	512.00 MB
Delta Memory :	508.00 MB
Max Swap :	-
Max Processes :	4
Max Threads :	5
Run time :	62 sec.
Turnaround time :	64 sec.

sleeper_702573.out

Batch jobs: multiple cores

```
#!/bin/bash  
  
Job name #BSUB -J sleeper  
Queue name #BSUB -q hpc  
Wall-clock time #BSUB -W 2  
Resources (mem) #BSUB -R "rusage[mem=512MB]"  
Number of cores #BSUB -n 4  
Number of hosts #BSUB -R "span[hosts=1]"  
Output file (stdout) #BSUB -o sleeper_%J.out  
Output file (stderr) #BSUB -e sleeper_%J.err  
  
sleep 60  
submit.sh
```

```
$ bsub < submit.sh  
Job <702574> is submitted to queue <hpc>.  
$
```

Note: memory usage is *multiplied* by number of requested cores!
Actual requested memory: 2048 MB = 2GB

Interlude: what is my actual hardware?

```
$ lscpu
Architecture:          x86_64
CPU op-mode(s):       32-bit, 64-bit
Byte Order:           Little Endian
CPU(s):               48 ← Number of threads
On-line CPU(s) list: 0-47
Thread(s) per core:   2
Core(s) per socket:   12 ← Number of cores / socket
Socket(s):            2 ← Number of sockets
NUMA node(s):         2
Vendor ID:            GenuineIntel
CPU family:           6
Model:                42
Model name:           Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz
Stepping:              1
CPU MHz:              2600.000
BogoMIPS:             5200.00
Virtualization:        KVM
L1d cache:            32K
L1i cache:            32K
L2 cache:              1024K
L3 cache:              19712K
NUMA node0 CPU(s):    0-11,24-35
NUMA node1 CPU(s):    12-23,36-47
Flags:                <omitted...>
```

**Note: shows hardware cores.
Not what you requested!**

Interlude: what is my actual hardware?

```
$ cpucount  
48  
$ echo $CPUTYPEV  
XeonGold6226
```

Batch jobs: selecting hardware

```
#!/bin/bash  
  
Job name #BSUB -J sleeper  
Queue name #BSUB -q hpc  
Wall-clock time #BSUB -W 2  
Resources (mem) #BSUB -R "rusage[mem=512MB]"  
CPU model #BSUB -R "select[model==XeonGold6226R]"  
Output file (stdout) #BSUB -o sleeper_%J.out  
Output file (stderr) #BSUB -e sleeper_%J.err
```

```
lscpu  
sleep 60  
submit.sh
```

```
$ bsub < submit.sh  
Job <702575> is submitted to queue <hpc>.  
$
```

Batch jobs: selecting hardware

```
#!/bin/bash
Job name #BSUB -J sleeper
Queue name #BSUB -q hpc
Wall-clock time #BSUB -W 2
Resources (mem) #BSUB -R "rusage[mem=512MB]"
CPU model #BSUB -R "select[model==XeonGold6226R]"
Output file (stdout) #BSUB -o sleeper_%J.out
Output file (stderr) #BSUB -e sleeper_%J.err
```

```
lscpu
sleep 60
submit.sh
```

```
$ bsub < submit.sh
Job <702575> is submitted to queue <hpc>.
$ cat sleeper_702575.out
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:             Little Endian
CPU(s):                 48
On-line CPU(s) list:   0-47
Thread(s) per core:    2
Core(s) per socket:    12
Socket(s):              2
NUMA node(s):           2
Vendor ID:              GenuineIntel
CPU family:             6
Model:                  85
Model name:             Intel(R) Xeon(R) Gold 6226
...
-----

```

Sender: LSF System <lsfadmin@hpc.dtu.dk>

...

Batch jobs: checking the output

```
#!/bin/bash  
  
Job name #BSUB -J sleeper  
Queue name #BSUB -q hpc  
Wall-clock time #BSUB -W 2  
Resources (mem) #BSUB -R "rusage[mem=512MB]"  
CPU model #BSUB -R "select[model==XeonGold6226R]"  
Output file (stdout) #BSUB -o sleeper_%J.out  
Output file (stderr) #BSUB -e sleeper_%J.err  
  
lscpu  
sleep 60  
submit.sh
```

```
$ bsub < submit.sh  
Job <702575> is submitted to queue <hpc>.  
$ bpeek  
<< output from stdout >>  
Architecture: x86_64  
CPU op-mode(s): 32-bit, 64-bit  
Byte Order: Little Endian  
CPU(s): 48  
On-line CPU(s) list: 0-47  
Thread(s) per core: 2  
Core(s) per socket: 12  
Socket(s): 2  
NUMA node(s): 2  
...  
<< output from stderr >>  
$
```

Batch jobs: checking the output

```
#!/bin/bash  
  
Job name #BSUB -J sleeper  
Queue name #BSUB -q hpc  
Wall-clock time #BSUB -W 2  
Resources (mem) #BSUB -R "rusage[mem=512MB]"  
CPU model #BSUB -R "select[model==XeonGold6226R]"  
Output file (stdout) #BSUB -o sleeper_%J.out  
Output file (stderr) #BSUB -e sleeper_%J.err  
  
lscpu  
sleep 60  
submit.sh
```

```
$ bsub < submit.sh  
Job <702575> is submitted to queue <hpc>.  
$ bpeek 702575 ←  
<< output from stdout >>  
Architecture: x86_64  
CPU op-mode(s): 32-bit, 64-bit  
Byte Order: Little Endian  
CPU(s): 48  
On-line CPU(s) list: 0-47  
Thread(s) per core: 2  
Core(s) per socket: 12  
Socket(s): 2  
NUMA node(s): 2  
...  
<< output from stderr >>  
$
```

What if I make a mistake?

Batch jobs: invalid jobs

```
#!/bin/bash
Job name #BSUB -J sleeper
Queue name #BSUB -q hpc
Wall-clock time #BSUB -W 2
Resources (mem) #BSUB -R "rusage[mem=512MB]"
Number of cores #BSUB -n 99
Number of hosts #BSUB -R "span[hosts=1]"
Output file (stdout) #BSUB -o sleeper_%J.out
Output file (stderr) #BSUB -e sleeper_%J.err

sleep 60
submit.sh
```

```
$ bsub < submit.sh
```

Batch jobs: invalid jobs

```
#!/bin/bash
Job name #BSUB -J sleeper
Queue name #BSUB -q hpc
Wall-clock time #BSUB -W 2
Resources (mem) #BSUB -R "rusage[mem=512MB]"
Number of cores #BSUB -n 99
Number of hosts #BSUB -R "span[hosts=1]"
Output file (stdout) #BSUB -o sleeper_%J.out
Output file (stderr) #BSUB -e sleeper_%J.err

sleep 60
submit.sh
```

```
$ bsub < submit.sh
Job <702576> is submitted to queue <hpc>.
$
```

Batch jobs: invalid jobs

```
#!/bin/bash  
  
Job name #BSUB -J sleeper  
Queue name #BSUB -q hpc  
Wall-clock time #BSUB -W 2  
Resources (mem) #BSUB -R "rusage[mem=512MB]"  
Number of cores #BSUB -n 99  
Number of hosts #BSUB -R "span[hosts=1]"  
Output file (stdout) #BSUB -o sleeper_%J.out  
Output file (stderr) #BSUB -e sleeper_%J.err  
  
sleep 60  
submit.sh
```

```
$ bsub < submit.sh  
Job <702576> is submitted to queue <hpc>.  
$ bstat  
JOBID      USER      QUEUE  JOB_NAME NALLOC STAT START_TIME   ELAPSED  
702576    patmjen    hpc    sleeper        0 PEND      -      0:00:00  
$
```

Job will never start!

Batch jobs: invalid jobs

```
#!/bin/bash
Job name #BSUB -J sleeper
Queue name #BSUB -q hpc
Wall-clock time #BSUB -W 2
Resources (mem) #BSUB -R "rusage[mem=512MB]"
Number of cores #BSUB -n 99
Number of hosts #BSUB -R "span[hosts=1]"
Output file (stdout) #BSUB -o sleeper_%J.out
Output file (stderr) #BSUB -e sleeper_%J.err

sleep 60
submit.sh
```

```
$ bsub < submit.sh
Job <702576> is submitted to queue <hpc>.
$ bstat
JOBID      USER      QUEUE  JOB_NAME NALLOC STAT START_TIME   ELAPSED
702576    patmjen    hpc    sleeper        0 PEND          -       0:00:00
$ bjobs -p
JOBID      USER      STAT  QUEUE  FROM_HOST EXEC_HOST JOB_NAME ...
702576    patmjen    PEND  hpc    hpclogin2           sleeper ...
Not enough job slot(s): 10 hosts;
$
```

Batch jobs: invalid jobs

```
#!/bin/bash
Job name #BSUB -J sleeper
Queue name #BSUB -q hpc
Wall-clock time #BSUB -W 2
Resources (mem) #BSUB -R "rusage[mem=250GB]"
Number of cores #BSUB -n 8
Number of hosts #BSUB -R "span[hosts=1]"
Output file (stdout) #BSUB -o sleeper_%J.out
Output file (stderr) #BSUB -e sleeper_%J.err

sleep 60
submit.sh
```

```
$ bsub < submit.sh
Job <702577> is submitted to queue <hpc>.
$ bstat
JOBID      USER      QUEUE  JOB_NAME NALLOC STAT START_TIME   ELAPSED
702577    patmjen    hpc    sleeper        0 PEND          - 0:00:00
$
```

Batch jobs: invalid jobs

```
#!/bin/bash
Job name #BSUB -J sleeper
Queue name #BSUB -q hpc
Wall-clock time #BSUB -W 2
Resources (mem) #BSUB -R "rusage[mem=250GB]"
Number of cores #BSUB -n 8
Number of hosts #BSUB -R "span[hosts=1]"
Output file (stdout) #BSUB -o sleeper_%J.out
Output file (stderr) #BSUB -e sleeper_%J.err

sleep 60
submit.sh
```

```
$ bsub < submit.sh
Job <702577> is submitted to queue <hpc>.
$ bstat
JOBID    USER      QUEUE  JOB_NAME NALLOC STAT START_TIME   ELAPSED
702577  patmjen   hpc    sleeper       0 PEND          - 0:00:00
$ bjobs -p
JOBID    USER      STAT  QUEUE  FROM_HOST EXEC_HOST JOB_NAME ...
702577  patmjen  PEND   hpc    hpclogin2           sleeper ...
Job's requirements for resource reservation not satisfied: 10
hosts;
$
```

Note: sometimes you just have to wait!
Help out by giving good resource requirements.

Batch jobs: invalid jobs

```
#!/bin/bash
Job name #BSUB -J sleeper
Queue name #BSUB -q hpc
Wall-clock time #BSUB -W 2
Resources (mem) #BSUB -R "rusage[mem=250GB]"
Number of cores #BSUB -n 8
Number of hosts #BSUB -R "span[hosts=1]"
Output file (stdout) #BSUB -o sleeper_%J.out
Output file (stderr) #BSUB -e sleeper_%J.err

sleep 60
submit.sh
```

```
$ bsub < submit.sh
Job <702577> is submitted to queue <hpc>.
$ bstat
JOBID    USER      QUEUE  JOB_NAME NALLOC STAT START_TIME   ELAPSED
702577  patmjen   hpc    sleeper       0 PEND        -      0:00:00
$ bjobs -p
JOBID    USER      STAT  QUEUE  FROM_HOST EXEC_HOST JOB_NAME ...
702577  patmjen  PEND   hpc    hpclogin2          sleeper ...
Job's requirements for resource reservation not satisfied: 10
hosts;
$
```

Batch jobs: invalid jobs

```
#!/bin/bash
Job name #BSUB -J sleeper
Queue name #BSUB -q hpc
Wall-clock time #BSUB -W 2
Resources (mem) #BSUB -R "rusage[mem=250GB]"
Number of cores #BSUB -n 8
Number of hosts #BSUB -R "span[hosts=1]"
Output file (stdout) #BSUB -o sleeper_%J.out
Output file (stderr) #BSUB -e sleeper_%J.err

sleep 60
submit.sh
```

```
$ bsub < submit.sh
Job <702577> is submitted to queue <hpc>.
$ bstat
JOBID      USER      QUEUE  JOB_NAME NALLOC STAT START_TIME   ELAPSED
702577    patmjen    hpc    sleeper        0 PEND          -       0:00:00
$ bjobs -p
JOBID      USER      STAT  QUEUE  FROM_HOST EXEC_HOST JOB_NAME ...
702577    patmjen    PEND  hpc    hpclogin2           sleeper ...
    Job's requirements for resource reservation not satisfied: 10
hosts;
$ bkill 702577
Job <702577> is being terminated
$
```

Batch jobs: invalid jobs

```
#!/bin/bash  
  
Job name #BSUB -J sleeper  
  
Queue name #BSUB -q hpc  
  
Wall-clock time #BSUB -W 2  
  
Resources (mem) #BSUB -R "rusage[mem=250GB]"  
  
Number of cores #BSUB -n 8  
  
Number of hosts #BSUB -R "span[hosts=1]"  
  
Output file (stdout) #BSUB -o sleeper_%J.out  
  
Output file (stderr) #BSUB -e sleeper_%J.err  
  
sleep 60  
submit.sh
```

```
$ bsub < submit.sh  
Job <702577> is submitted to queue <hpc>.  
$ bstat  
JOBID      USER      QUEUE  JOB_NAME NALLOC STAT START_TIME   ELAPSED  
702577    patmjen    hpc    sleeper        0 PEND          -       0:00:00  
$ bjobs -p  
JOBID      USER      STAT  QUEUE  FROM_HOST EXEC_HOST JOB_NAME ...  
702577    patmjen    PEND  hpc    hpclogin2           sleeper ...  
  Job's requirements for resource reservation not satisfied: 10  
hosts;  
$ bkill 702577  
Job <702577> is being terminated  
$
```

Please clean after yourselves with bkill

Batch jobs: invalid jobs

```
#!/bin/bash  
Job name #BSUB -J sleeper  
Queue name #BSUB -q hpc  
Wall-clock ti  
Resources (m)  
Number of c  
Number of hosts #BSUB -R "span[hosts=1]"  
Output file (stdout) #BSUB -o sleeper_%J.out  
Output file (stderr) #BSUB -e sleeper_%J.err  
  
sleep 60  
submit.sh
```

```
$ bsub < submit.sh  
Job <702577> is submitted to queue <hpc>.  
$ bstat  
JOBID      USER      QUEUE  JOB_NAME NALLOC STAT START_TIME   ELAPSED  
702577  natmjen  hpc    sleeper       0 PEND          - 0:00:00
```

More available – Links to guides on Learn

```
702577  
Job's requirements for hosts;  
$ bkill 702577  
Job <702577> is being terminated  
$
```

Today's exercise

Get set up and get familiar:

1. File upload and download
2. Python scripts on interactive nodes
3. Simple batch jobs

Python on HPC: shared Anaconda env

```
$
```

Python on HPC: shared Anaconda env

```
$ source /dtu/projects/02613_2025/conda/conda_init.sh  
$ conda activate 02613
```

Python on HPC: shared Anaconda env

```
$ source /dtu/projects/02613_2024/conda/conda_init.sh
$ conda activate 02613
(02613) $
```

Python on HPC: shared Anaconda env

```
$ source /dtu/projects/02613_2024/conda/conda_init.sh
$ conda activate 02613
(02613) $ python -c "import sys; print(sys.executable)"
/dtu/projects/02613_2025/conda/miniconda3/envs/02613/bin/python
(02613) $
```

Python on HPC: shared Anaconda env

```
$ source /dtu/projects/02613_2024/conda/conda_init.sh  
$ conda activate 02613  
(02613) $ python -c "import sys; print(sys.executable)"  
/dtu/projects/02613_2024/conda/miniconda3/envs/02613/bin/python  
(02613) $
```

**Must do this everytime you
change node!**

Python on HPC: shared Anaconda env

```
$ source /dtu/projects/02613_2024/conda/conda_init.sh  
$ conda activate 02613  
(02613) $ python -c "import sys; print(sys.executable)"  
/dtu/projects/02613_2024/conda/miniconda3/envs/02613/bin/python  
(02613) $
```

Check "Initializing the 02613 Conda Environment" in Learn for known issues

Useful commands

Log on to HPC

```
ssh <username>@login.hpc.dtu.dk  
ssh <username>@login2.hpc.dtu.dk
```

Initialize course Anaconda environment

```
source /dtu/projects/02613_2024/conda/conda_init.sh  
conda activate 02613
```

Change to work node

```
linuxsh
```

Terminal commands (directory == folder)

```
pwd          # print working directory  
ls           # list contents of working directory  
ls <path>    # list contents of path  
cd <path>    # change directory to path  
mkdir -p <path> # make new directory
```

Check CPU

```
lscpu
```

Submit job script

```
bsub < submit.sh
```

Job status

```
bstat  
bjobs  
bjobs -p  # pending reason
```

Check job output

```
bpeek  
bpeek <JOBID>
```

Kill job

```
bkill <JOBID>
```