UNIVERSITY OF CAMBRIDGE

An Introduction to High Performance Computing

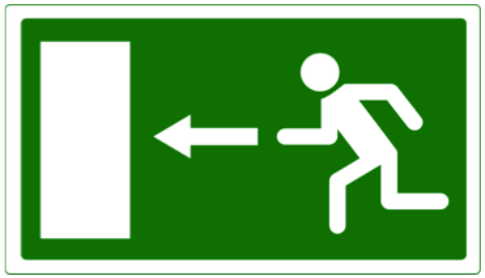
Stuart Rankin

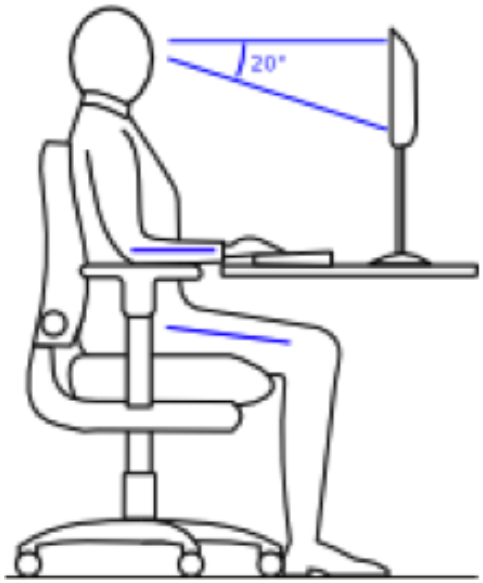
support@hpc.cam.ac.uk

Research Computing Services (http://www.hpc.cam.ac.uk/ http://www.csd3.cam.ac.uk/)
University Information Services (http://www.uis.cam.ac.uk/)

18th October 2017 / UIS Training

Health and Safety





Welcome

- Please sign in on the attendance sheet.
- Please fill in the online feedback at the end of the course:
  `http://feedback.training.cam.ac.uk/ucs/form.php`
- Keep your belongings with you.
- The printer will not work.
- Please ask questions and let us know if you need assistance.

Plan of the Course

Part 1: Basics

Part 2: High Performance Computing Service

Part 3: Using a HPC Facility

09:30	WELCOME
11:00-11:30	Practical and break
12:00-12:30	Practical
12:30-13:30	LUNCH
14:00-14:30	Practical
14:45-15:15	Practical
15:30-CLOSE	Further discussion

Plan of the Course

Part 1: Basics

Part 2: High Performance Computing Service

Part 3: Using a HPC Facility

09:30	WELCOME
11:00-11:30	Practical and break
12:00-12:30	Practical
12:30-13:30	LUNCH
14:00-14:30	Practical
14:45-15:15	Practical
15:30-CLOSE	Further discussion

Part I: **Basics**

Basics: Outline

Why Buy a Big Computer?

Inside a Modern Computer

How to Build a Supercomputer

Programming a Multiprocessor Machine

Basics: Why Buy a Big Computer?

What types of big problem might require a "Big Computer"?

Compute Intensive: A single problem requiring a large amount of computation.

Memory Intensive: A single problem requiring a large amount of memory.

Data Intensive: A single problem operating on a large amount of data.

High Throughput: Many unrelated problems to be executed in bulk.

Basics: Why Buy a Big Computer?

What types of big problem might require a "Big Computer"?

Compute Intensive: A single problem requiring a large amount of computation.

Memory Intensive: A single problem requiring a large amount of memory.

Data Intensive: A single problem operating on a large amount of data.

High Throughput: Many unrelated problems to be executed in bulk.

Basics: Why Buy a Big Computer?

What types of big problem might require a "Big Computer"?

Compute Intensive: A single problem requiring a large amount of computation.

Memory Intensive: A single problem requiring a large amount of memory.

Data Intensive: A single problem operating on a large amount of data.

High Throughput: Many unrelated problems to be executed in bulk.

Basics: Why Buy a Big Computer?

What types of big problem might require a "Big Computer"?

Compute Intensive: A single problem requiring a large amount of computation.

Memory Intensive: A single problem requiring a large amount of memory.

Data Intensive: A single problem operating on a large amount of data.

High Throughput: Many unrelated problems to be executed in bulk.

Basics: Why Buy a Big Computer?

What types of big problem might require a "Big Computer"?

Compute Intensive: A single problem requiring a large amount of computation.

Memory Intensive: A single problem requiring a large amount of memory.

Data Intensive: A single problem operating on a large amount of data.

High Throughput: Many unrelated problems to be executed in bulk.

Basics: Compute Intensive Problems

- Distribute the work for a single problem across multiple CPUs to reduce the execution time as far as possible.
- Program workload must be *parallelised*:

  Parallel programs split into copies (processes or threads).

  Each process/thread performs a part of the work on its own CPU, concurrently with the others.

  A well-parallelised program will fully exercise as many CPUs as there are processes/threads.
- The CPUs typically need to exchange information rapidly, requiring specialized communication hardware.
- Many use cases from Physics, Chemistry, Engineering, Astronomy, Biology...
- The traditional domain of HPC and the Supercomputer.

Basics: Compute Intensive Problems

- Distribute the work for a single problem across multiple CPUs to reduce the execution time as far as possible.
- Program workload must be *parallelised*:

  Parallel programs split into copies (processes or threads).
  Each process/thread performs a part of the work on its own CPU, concurrently with the others.
  A well-parallelised program will fully exercise as many CPUs as there are processes/threads.
- The CPUs typically need to exchange information rapidly, requiring specialized communication hardware.
- Many use cases from Physics, Chemistry, Engineering, Astronomy, Biology...
- The traditional domain of HPC and the Supercomputer.

Basics: Compute Intensive Problems

- Distribute the work for a single problem across multiple CPUs to reduce the execution time as far as possible.
- Program workload must be *parallelised*:

  Parallel programs split into copies (processes or threads).
  Each process/thread performs a part of the work on its own CPU, concurrently with the others.
  A well-parallelised program will fully exercise as many CPUs as there are processes/threads.
- The CPUs typically need to exchange information rapidly, requiring specialized communication hardware.
- Many use cases from Physics, Chemistry, Engineering, Astronomy, Biology...
- The traditional domain of HPC and the Supercomputer.

Basics: Compute Intensive Problems

- Distribute the work for a single problem across multiple CPUs to reduce the execution time as far as possible.
- Program workload must be *parallelised*:

  Parallel programs split into copies (processes or threads).
  Each process/thread performs a part of the work on its own CPU, concurrently with the others.
  A well-parallelised program will fully exercise as many CPUs as there are processes/threads.
- The CPUs typically need to exchange information rapidly, requiring specialized communication hardware.
- Many use cases from Physics, Chemistry, Engineering, Astronomy, Biology...
- The traditional domain of HPC and the Supercomputer.

Basics: Scaling & Amdahl's Law

- Using more CPUs is not necessarily faster.
- Typically parallel codes have a scaling limit.
- Partly due to the system overhead of managing more copies, but also to more basic constraints;
- Amdahl's Law (idealized):

$$S(N) = \frac{1}{(1 - p + \frac{p}{N})}$$

where

$S(N)$ is the fraction by which the program has sped up relative to $N = 1$

p is the fraction of the program which can be parallelized

N is the number of CPUs.

Basics: Scaling & Amdahl's Law

- Using more CPUs is not necessarily faster.
- Typically parallel codes have a scaling limit.
- Partly due to the system overhead of managing more copies, but also to more basic constraints;
- Amdahl's Law (idealized):

$$S(N) = \frac{1}{(1 - p + \frac{p}{N})}$$

where

$S(N)$ is the fraction by which the program has sped up relative to $N = 1$

p is the fraction of the program which can be parallelized

N is the number of CPUs.

Basics: Scaling & Amdahl's Law

- Using more CPUs is not necessarily faster.
- Typically parallel codes have a scaling limit.
- Partly due to the system overhead of managing more copies, but also to more basic constraints;
- Amdahl's Law (idealized):

$$S(N) = \frac{1}{\left(1 - p + \frac{p}{N}\right)}$$

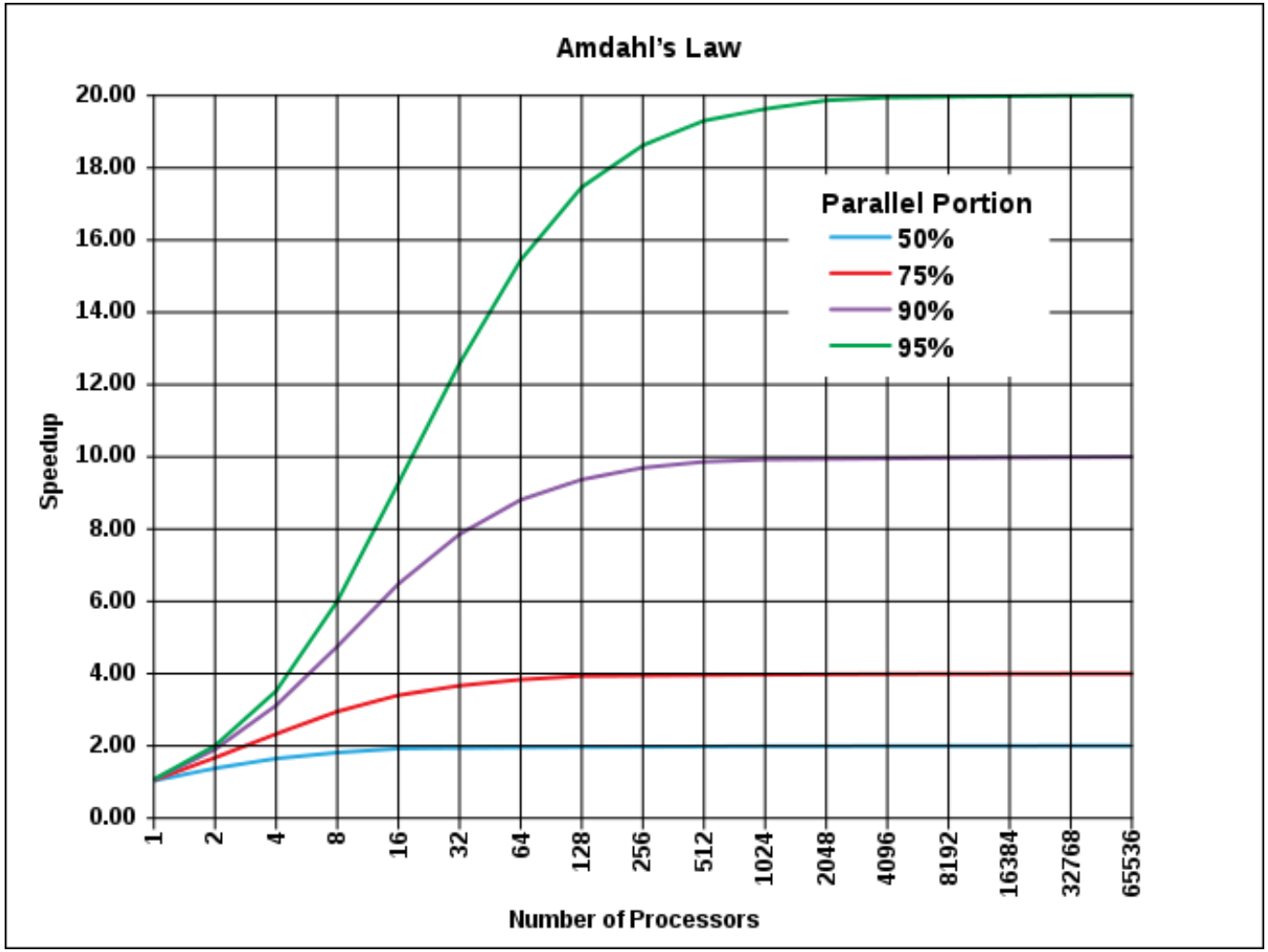
where

$S(N)$ is the fraction by which the program has sped up relative to $N = 1$

p is the fraction of the program which can be parallelized

N is the number of CPUs.

Basics: Amdahl's Law



http://en.wikipedia.org/wiki/File:AmdahlsLaw.svg

The Bottom Line

- Parallelisation requires effort:
  - There are libraries to help (e.g. OpenMP, MPI).
  - First optimise performance on one CPU, then make p as large as possible.
- The scaling limit: eventually using more CPUs becomes detrimental instead of helpful.

The Bottom Line

- Parallelisation requires effort:
  - There are libraries to help (e.g. OpenMP, MPI).
  - First optimise performance on one CPU, then make p as large as possible.
- The scaling limit: eventually using more CPUs becomes detrimental instead of helpful.

Basics: Data Intensive Problems

- Distribute the data for a single problem across multiple CPUs to reduce the overall execution time.
- The *same* work may be done on each data segment.
- Rapid movement of data to and from disk is more important than inter-CPU communication.
- Big Data problems of great current interest -
- Hadoop/MapReduce
- Life Sciences (genomics) and elsewhere.

Basics: Data Intensive Problems

- Distribute the data for a single problem across multiple CPUs to reduce the overall execution time.
- The *same* work may be done on each data segment.
- Rapid movement of data to and from disk is more important than inter-CPU communication.
- Big Data problems of great current interest -
- Hadoop/MapReduce
- Life Sciences (genomics) and elsewhere.

Basics: Data Intensive Problems

- Distribute the data for a single problem across multiple CPUs to reduce the overall execution time.
- The *same* work may be done on each data segment.
- Rapid movement of data to and from disk is more important than inter-CPU communication.
- Big Data problems of great current interest -
- Hadoop/MapReduce
- Life Sciences (genomics) and elsewhere.

Basics: Data Intensive Problems

- Distribute the data for a single problem across multiple CPUs to reduce the overall execution time.
- The *same* work may be done on each data segment.
- Rapid movement of data to and from disk is more important than inter-CPU communication.
- Big Data problems of great current interest -
- Hadoop/MapReduce
- Life Sciences (genomics) and elsewhere.

Basics: High Throughput

- Distribute independent, multiple problems across multiple CPUs to reduce the overall execution time.
- Workload is trivially (or *embarrassingly*) parallel:
  - Workload breaks up naturally into *independent* pieces.
  - Each piece is performed by a separate process/thread on a separate CPU (concurrently).
  - Little or no inter-CPU communication.
- Emphasis is on throughput over a period, rather than on performance on a single problem.
- Compute intensive capable $\Rightarrow$ high throughput capable
- Compute intensive capable $\not\Leftarrow$ high throughput capable

Basics: High Throughput

- Distribute independent, multiple problems across multiple CPUs to reduce the overall execution time.
- Workload is trivially (or *embarrassingly*) parallel:
  - Workload breaks up naturally into *independent* pieces.
  - Each piece is performed by a separate process/thread on a separate CPU (concurrently).
  - Little or no inter-CPU communication.
- Emphasis is on throughput over a period, rather than on performance on a single problem.
- Compute intensive capable $\Rightarrow$ high throughput capable
- Compute intensive capable $\not\Leftarrow$ high throughput capable

Basics: High Throughput

- Distribute independent, multiple problems across multiple CPUs to reduce the overall execution time.
- Workload is trivially (or *embarrassingly*) parallel:
  - Workload breaks up naturally into *independent* pieces.
  - Each piece is performed by a separate process/thread on a separate CPU (concurrently).
  - Little or no inter-CPU communication.
- Emphasis is on throughput over a period, rather than on performance on a single problem.
- Compute intensive capable $\Rightarrow$ high throughput capable
- Compute intensive capable $\not\Leftarrow$ high throughput capable

Basics: High Throughput

- Distribute independent, multiple problems across multiple CPUs to reduce the overall execution time.
- Workload is trivially (or *embarrassingly*) parallel:
  - Workload breaks up naturally into *independent* pieces.
  - Each piece is performed by a separate process/thread on a separate CPU (concurrently).
  - Little or no inter-CPU communication.
- Emphasis is on throughput over a period, rather than on performance on a single problem.
- Compute intensive capable $\Rightarrow$ high throughput capable
- Compute intensive capable $\not\Leftarrow$ high throughput capable

Basics: High Throughput

- Distribute independent, multiple problems across multiple CPUs to reduce the overall execution time.
- Workload is trivially (or *embarrassingly*) parallel:
  - Workload breaks up naturally into *independent* pieces.
  - Each piece is performed by a separate process/thread on a separate CPU (concurrently).
  - Little or no inter-CPU communication.
- Emphasis is on throughput over a period, rather than on performance on a single problem.
- Compute intensive capable $\Rightarrow$ high throughput capable
- Compute intensive capable $\not\Leftarrow$ high throughput capable

Basics: Memory Intensive Problems

- Require aggregation of large memory rather than multiple CPUs.

  NB Memory (fast, volatile) vs disk (slow, non-volatile).
- Technically more challenging to build machines (interconnecting memory and CPUs).
- Coding/porting easier (memory appears seamless, allowing a single system image).
- Optimisation harder (nonuniform memory produces latency).
- Historically, the arena of large SGI systems.
- Nowadays, similar techniques are applicable to commodity servers.

Basics: Memory Intensive Problems

- Require aggregation of large memory rather than multiple CPUs.

  NB Memory (fast, volatile) vs disk (slow, non-volatile).
- Technically more challenging to build machines (interconnecting memory and CPUs).
- Coding/porting easier (memory appears seamless, allowing a single system image).
- Optimisation harder (nonuniform memory produces latency).
- Historically, the arena of large SGI systems.
- Nowadays, similar techniques are applicable to commodity servers.

Basics: Memory Intensive Problems

- Require aggregation of large memory rather than multiple CPUs.

  NB Memory (fast, volatile) vs disk (slow, non-volatile).
- Technically more challenging to build machines (interconnecting memory and CPUs).
- Coding/porting easier (memory appears seamless, allowing a single system image).
- Optimisation harder (nonuniform memory produces latency).
- Historically, the arena of large SGI systems.
- Nowadays, similar techniques are applicable to commodity servers.

Basics: Memory Intensive Problems

- Require aggregation of large memory rather than multiple CPUs.

  NB Memory (fast, volatile) vs disk (slow, non-volatile).
- Technically more challenging to build machines (interconnecting memory and CPUs).
- Coding/porting easier (memory appears seamless, allowing a single system image).
- Optimisation harder (nonuniform memory produces latency).
- Historically, the arena of large SGI systems.
- Nowadays, similar techniques are applicable to commodity servers.

Basics: Memory Intensive Problems

- Require aggregation of large memory rather than multiple CPUs.

  NB Memory (fast, volatile) vs disk (slow, non-volatile).
- Technically more challenging to build machines (interconnecting memory and CPUs).
- Coding/porting easier (memory appears seamless, allowing a single system image).
- Optimisation harder (nonuniform memory produces latency).
- Historically, the arena of large SGI systems.
- Nowadays, similar techniques are applicable to commodity servers.

Basics: Memory Intensive Problems

- Require aggregation of large memory rather than multiple CPUs.

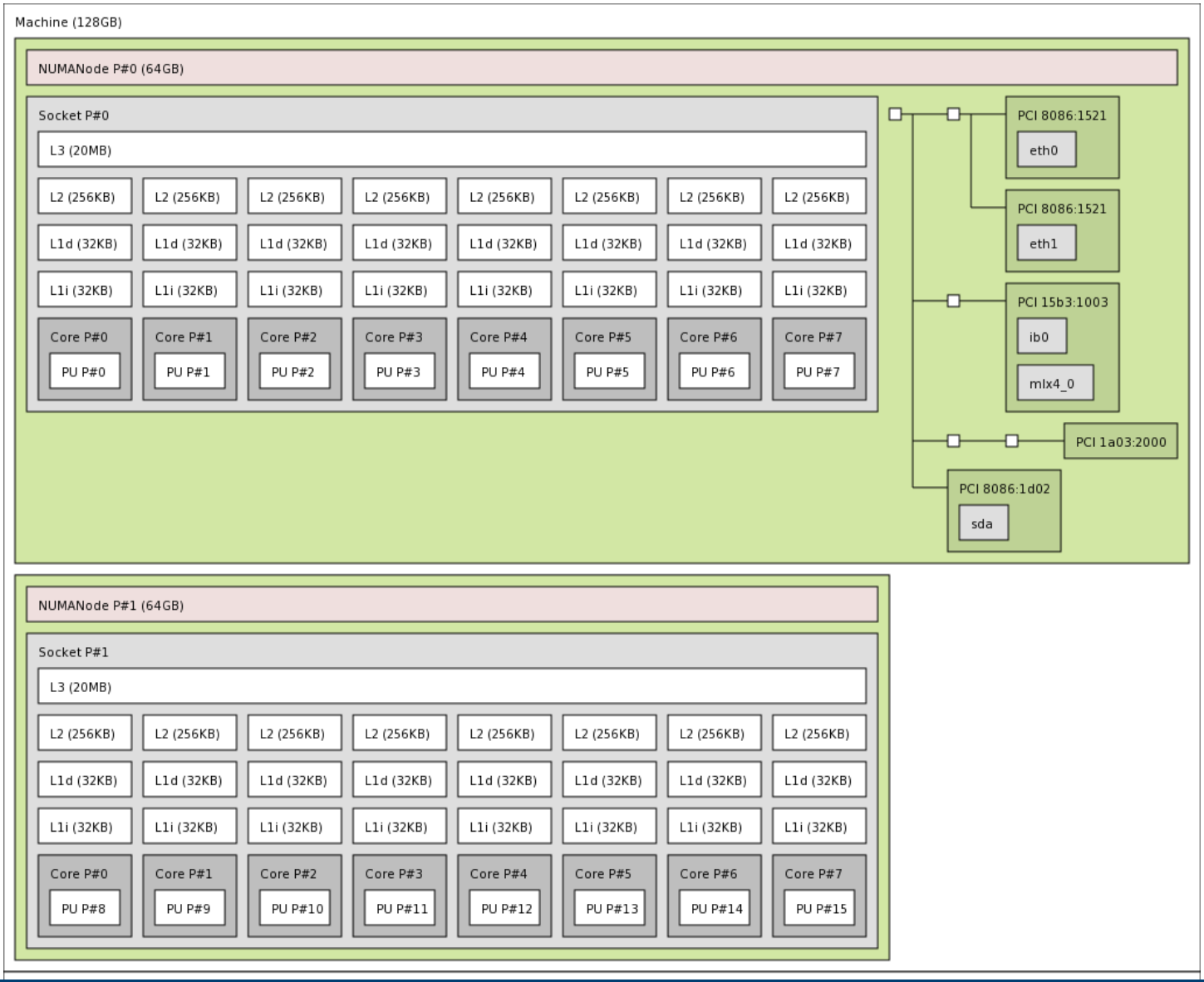
  NB Memory (fast, volatile) vs disk (slow, non-volatile).
- Technically more challenging to build machines (interconnecting memory and CPUs).
- Coding/porting easier (memory appears seamless, allowing a single system image).
- Optimisation harder (nonuniform memory produces latency).
- Historically, the arena of large SGI systems.
- Nowadays, similar techniques are applicable to commodity servers.

Basics: Memory Intensive Problems

- Require aggregation of large memory rather than multiple CPUs.

  NB Memory (fast, volatile) vs disk (slow, non-volatile).
- Technically more challenging to build machines (interconnecting memory and CPUs).
- Coding/porting easier (memory appears seamless, allowing a single system image).
- Optimisation harder (nonuniform memory produces latency).
- Historically, the arena of large SGI systems.
- Nowadays, similar techniques are applicable to commodity servers.

Basics: Inside a Modern Computer



Basics: Inside a Modern Computer

- Today's commodity servers already aggregate both CPUs and memory.
- Even small computers now have multiple CPU cores per socket
- Larger computers have multiple sockets (each with local memory): all CPU cores (unequally) share the node memory

Basics: Inside a Modern Computer

- Today's commodity servers already aggregate both CPUs and memory.
- Even small computers now have multiple CPU cores per socket
- Larger computers have multiple sockets (each with local memory): all CPU cores (unequally) share the node memory

Basics: Inside a Modern Computer

- Today's commodity servers already aggregate both CPUs and memory.
- Even small computers now have multiple CPU cores per socket
  $\Longrightarrow$ each socket is a Symmetric Multi-Processor (SMP).
- Larger computers have multiple sockets (each with local memory):
  all CPU cores (unequally) share the node memory

Basics: Inside a Modern Computer

- Today's commodity servers already aggregate both CPUs and memory.
- Even small computers now have multiple CPU cores per socket
  $\Longrightarrow$ each socket is a Symmetric Multi-Processor (SMP).
- Larger computers have multiple sockets (each with local memory):
  all CPU cores (unequally) share the node memory

Basics: Inside a Modern Computer

- Today's commodity servers already aggregate both CPUs and memory.
- Even small computers now have multiple CPU cores per socket
  $\Longrightarrow$ each socket is a Symmetric Multi-Processor (SMP).
- Larger computers have multiple sockets (each with local memory):
  all CPU cores (unequally) share the node memory
  $\Longrightarrow$ the node is a shared memory multiprocessor

Basics: Inside a Modern Computer

- Today's commodity servers already aggregate both CPUs and memory.
- Even small computers now have multiple CPU cores per socket
  $\Longrightarrow$ each socket is a Symmetric Multi-Processor (SMP).
- Larger computers have multiple sockets (each with local memory):
  all CPU cores (unequally) share the node memory
  $\Longrightarrow$ the node is a shared memory multiprocessor
  with Non-Uniform Memory Architecture (NUMA)

Basics: Inside a Modern Computer

- Today's commodity servers already aggregate both CPUs and memory.
- Even small computers now have multiple CPU cores per socket
  $\Longrightarrow$ each socket is a Symmetric Multi-Processor (SMP).
- Larger computers have multiple sockets (each with local memory):
  all CPU cores (unequally) share the node memory
  $\Longrightarrow$ the node is a shared memory multiprocessor
  with Non-Uniform Memory Architecture (NUMA)
  but users still see a single computer (single system image).

Basics: How to Build a Supercomputer

- A supercomputer aggregates contemporary CPUs and memory to obtain increased computing power.
- Usually today these are clusters.

Basics: How to Build a Supercomputer

- A supercomputer aggregates contemporary CPUs and memory to obtain increased computing power.
- Usually today these are clusters.

Basics: How to Build a Supercomputer

1. Take some (multicore) CPUs plus some memory.
   - Could be an off-the-shelf server, or something more special.
   - A NUMA, shared memory, multiprocessor building block: a node.

Basics: How to Build a Supercomputer

1. Take some (multicore) CPUs plus some memory.
   - Could be an off-the-shelf server, or something more special.
   - A NUMA, shared memory, multiprocessor building block: a node.

Basics: How to Build a Supercomputer

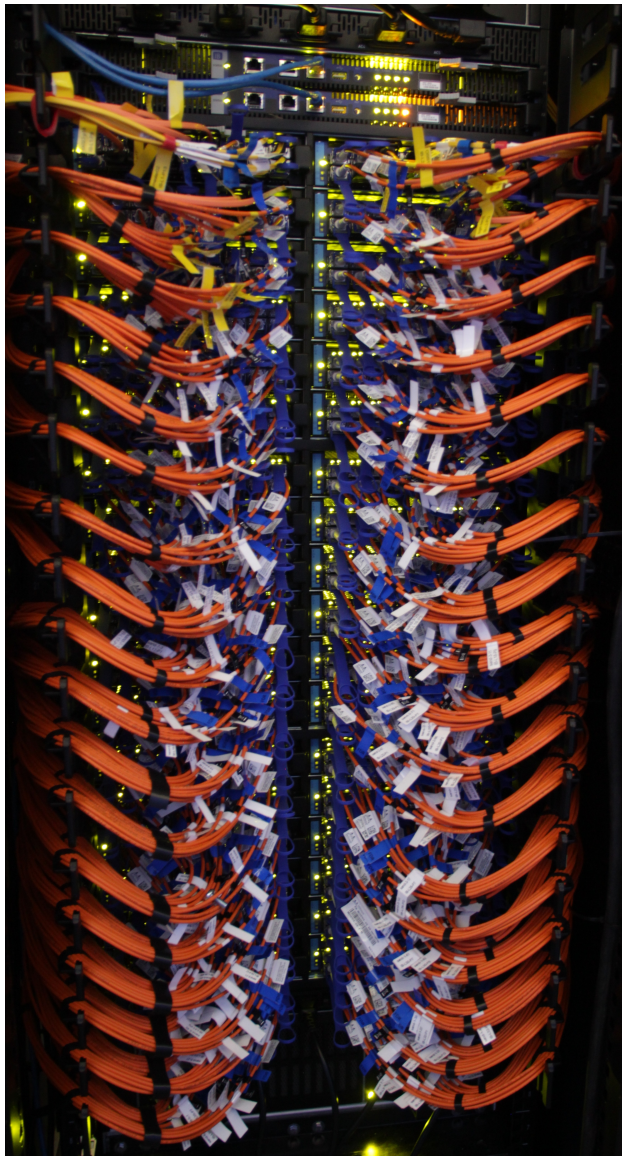
1. Take some (multicore) CPUs plus some memory.
   - Could be an off-the-shelf server, or something more special.
   - A NUMA, shared memory, multiprocessor building block: a node.

Basics: How to Build a Supercomputer

2. Connect the nodes with one or more networks. E.g.

   Gbit Ethernet: 100 MB/sec
   FDR Infiniband: 5 GB/sec

   Faster network is for inter-CPU communication across nodes.
   Slower network is for management and provisioning.
   Storage may use either.

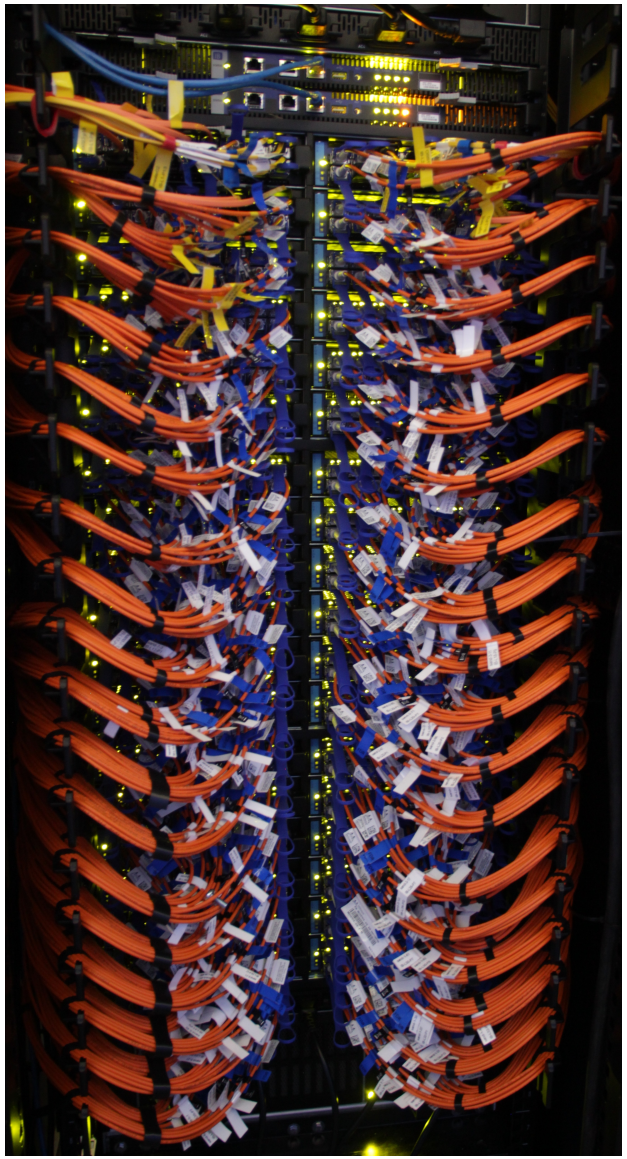


Basics: How to Build a Supercomputer

2. Connect the nodes with one or more networks. E.g.

   Gbit Ethernet: 100 MB/sec
   FDR Infiniband: 5 GB/sec

   Faster network is for inter-CPU communication across nodes.
   Slower network is for management and provisioning.
   Storage may use either.



Basics: How to Build a Supercomputer

3. Logically bind the nodes
   - Clusters consist of distinct nodes (i.e. separate Linux computers) on common private network(s) and controlled centrally.
     * Private networks allow CPUs in different nodes to communicate.
     * Clusters are distributed memory machines:
       Each process/thread sees only its local node's CPUs and memory (without help).
     * Each process/thread must fit within a single node's memory.
   - More expensive machines logically bind nodes into a single system i.e. CPUs and memory.
     * E.g. SGI UV (COSMOS system in DAMTP).
     * Private networks allow CPUs to see CPUs and memory in other nodes.
     * These are shared memory machines.
     * Logically a single system - 1 big node - but very non-uniform.
     * A single process can span the entire system.

Basics: How to Build a Supercomputer

3. Logically bind the nodes
   - Clusters consist of distinct nodes (i.e. separate Linux computers) on common private network(s) and controlled centrally.
     * Private networks allow CPUs in different nodes to communicate.
     * Clusters are distributed memory machines:
       Each process/thread sees only its local node's CPUs and memory (without help).
     * Each process/thread must fit within a single node's memory.
   - More expensive machines logically bind nodes into a single system i.e. CPUs and memory.
     * E.g. SGI UV (COSMOS system in DAMTP).
     * Private networks allow CPUs to see CPUs and memory in other nodes.
     * These are shared memory machines.
     * Logically a single system - 1 big node - but very non-uniform.
     * A single process can span the entire system.

Basics: How to Build a Supercomputer

3. Logically bind the nodes
   - Clusters consist of distinct nodes (i.e. separate Linux computers) on common private network(s) and controlled centrally.
     * Private networks allow CPUs in different nodes to communicate.
     * Clusters are distributed memory machines:
       Each process/thread sees only its local node's CPUs and memory (without help).
     * Each process/thread must fit within a single node's memory.
   - More expensive machines logically bind nodes into a single system i.e. CPUs and memory.
     * E.g. SGI UV (COSMOS system in DAMTP).
     * Private networks allow CPUs to see CPUs and memory in other nodes.
     * These are shared memory machines.
     * Logically a single system - 1 big node - but very non-uniform.
     * A single process can span the entire system.

Basics: How to Build a Supercomputer

3. Logically bind the nodes
   - Clusters consist of distinct nodes (i.e. separate Linux computers) on common private network(s) and controlled centrally.
     * Private networks allow CPUs in different nodes to communicate.
     * Clusters are distributed memory machines:
       Each process/thread sees only its local node's CPUs and memory (without help).
     * Each process/thread must fit within a single node's memory.
   - More expensive machines logically bind nodes into a single system i.e. CPUs and memory.
     * E.g. SGI UV (COSMOS system in DAMTP).
     * Private networks allow CPUs to see CPUs and memory in other nodes.
     * These are shared memory machines.
     * Logically a single system - 1 big node - but very non-uniform.
     * A single process can span the entire system.

Basics: How to Build a Supercomputer

3. Logically bind the nodes
   - Clusters consist of distinct nodes (i.e. separate Linux computers) on common private network(s) and controlled centrally.
     * Private networks allow CPUs in different nodes to communicate.
     * Clusters are distributed memory machines:
       Each process/thread sees only its local node's CPUs and memory (without help).
     * Each process/thread must fit within a single node's memory.
   - More expensive machines logically bind nodes into a single system i.e. CPUs and memory.
     * E.g. SGI UV (COSMOS system in DAMTP).
     * Private networks allow CPUs to see CPUs and memory in other nodes.
     * These are shared memory machines.
     * Logically a single system - 1 big node - but very non-uniform.
     * A single process can span the entire system.

Basics: Programming a Multiprocessor Machine

- Non-parallel (serial) code
  - For a single node as for a workstation.
  - Typically run as many copies per node as CPUs, assuming node memory is sufficent.
  - Replicate across multiple nodes.

Basics: Programming a Multiprocessor Machine

- Non-parallel (serial) code
  - For a single node as for a workstation.
  - Typically run as many copies per node as CPUs, assuming node memory is sufficent.
  - Replicate across multiple nodes.

Basics: Programming a Multiprocessor Machine

- Non-parallel (serial) code
  - For a single node as for a workstation.
  - Typically run as many copies per node as CPUs, assuming node memory is sufficent.
  - Replicate across multiple nodes.

Basics: Programming a Multiprocessor Machine

- Parallel code
  - Shared memory methods within a node.
    E.g. pthreads, OpenMP.
  - Distributed memory methods spanning multiple nodes.
    Message Passing Interface (MPI).

Basics: Programming a Multiprocessor Machine

- Parallel code
  - Shared memory methods within a node.
    E.g. pthreads, OpenMP.
  - Distributed memory methods spanning multiple nodes.
    Message Passing Interface (MPI).

Basics: Programming a Multiprocessor Machine

- Parallel code
  - Shared memory methods within a node.
    E.g. pthreads, OpenMP.
  - Distributed memory methods spanning multiple nodes.
    Message Passing Interface (MPI).

Basics: Summary

- Why have a supercomputer?
  - Big single problems, many problems, Big Data.
- Most current supercomputers are clusters of separate nodes.
- Each node has multiple CPUs and non-uniform shared memory.
- Parallel code uses shared memory (pthreads/OpenMP) within a node, distributed memory (MPI) spanning multiple nodes.
- Non-parallel code uses the memory of one node, but may be copied across many.

Basics: Summary

- Why have a supercomputer?
  - Big single problems, many problems, Big Data.
- Most current supercomputers are clusters of separate nodes.
- Each node has multiple CPUs and non-uniform shared memory.
- Parallel code uses shared memory (pthreads/OpenMP) within a node, distributed memory (MPI) spanning multiple nodes.
- Non-parallel code uses the memory of one node, but may be copied across many.

Basics: Summary

- Why have a supercomputer?
  - Big single problems, many problems, Big Data.
- Most current supercomputers are clusters of separate nodes.
- Each node has multiple CPUs and non-uniform shared memory.
- Parallel code uses shared memory (pthreads/OpenMP) within a node, distributed memory (MPI) spanning multiple nodes.
- Non-parallel code uses the memory of one node, but may be copied across many.

Basics: Summary

- Why have a supercomputer?
  - Big single problems, many problems, Big Data.
- Most current supercomputers are clusters of separate nodes.
- Each node has multiple CPUs and non-uniform shared memory.
- Parallel code uses shared memory (pthreads/OpenMP) within a node, distributed memory (MPI) spanning multiple nodes.
- Non-parallel code uses the memory of one node, but may be copied across many.

Basics: Summary

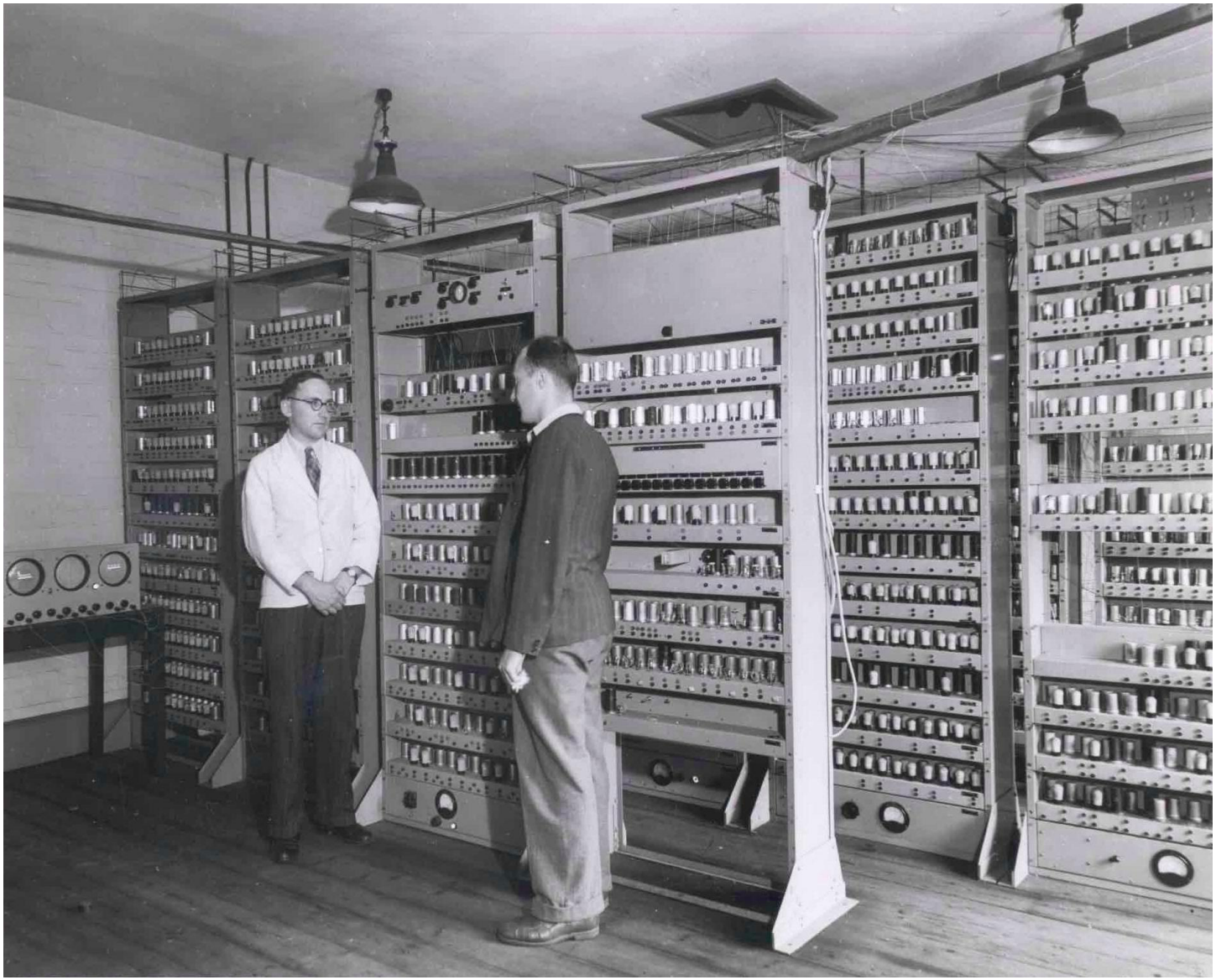
- Why have a supercomputer?
  - Big single problems, many problems, Big Data.
- Most current supercomputers are clusters of separate nodes.
- Each node has multiple CPUs and non-uniform shared memory.
- Parallel code uses shared memory (pthreads/OpenMP) within a node, distributed memory (MPI) spanning multiple nodes.
- Non-parallel code uses the memory of one node, but may be copied across many.

Basics: Summary

- Why have a supercomputer?
  - Big single problems, many problems, Big Data.
- Most current supercomputers are clusters of separate nodes.
- Each node has multiple CPUs and non-uniform shared memory.
- Parallel code uses shared memory (pthreads/OpenMP) within a node, distributed memory (MPI) spanning multiple nodes.
- Non-parallel code uses the memory of one node, but may be copied across many.

Part II: **The High Performance Computing**

Early History: EDSAC (1949–1958)



Early History: EDSAC (1949–1958)

- **E**lectronic **D**elay **S**torage **A**utomatic **C**alculator
- The second general use, electronic digital (Turing complete) stored program computer
- 3,000 valves
- 650 instructions per second
- 2KB memory in mercury ultrasonic delay lines
- One program at a time!
- Used in meteorology, genetics, theoretical chemistry, numerical analysis, radioastronomy.
- *"On a few occasions it worked for more than 24 hours."*

Early History: EDSAC (1949–1958)

- **E**lectronic **D**elay **S**torage **A**utomatic **C**alculator
- The second general use, electronic digital (Turing complete) stored program computer
- 3,000 valves
- 650 instructions per second
- 2KB memory in mercury ultrasonic delay lines
- One program at a time!
- Used in meteorology, genetics, theoretical chemistry, numerical analysis, radioastronomy.
- *"On a few occasions it worked for more than 24 hours."*

Early History: EDSAC (1949–1958)

- **E**lectronic **D**elay **S**torage **A**utomatic **C**alculator
- The second general use, electronic digital (Turing complete) stored program computer
- 3,000 valves
- 650 instructions per second
- 2KB memory in mercury ultrasonic delay lines
- One program at a time!
- Used in meteorology, genetics, theoretical chemistry, numerical analysis, radioastronomy.
- *"On a few occasions it worked for more than 24 hours."*

Early History: EDSAC (1949–1958)

- **E**lectronic **D**elay **S**torage **A**utomatic **C**alculator
- The second general use, electronic digital (Turing complete) stored program computer
- 3,000 valves
- 650 instructions per second
- 2KB memory in mercury ultrasonic delay lines
- One program at a time!
- Used in meteorology, genetics, theoretical chemistry, numerical analysis, radioastronomy.
- *"On a few occasions it worked for more than 24 hours."*

Early History: EDSAC (1949–1958)

- **E**lectronic **D**elay **S**torage **A**utomatic **C**alculator
- The second general use, electronic digital (Turing complete) stored program computer
- 3,000 valves
- 650 instructions per second
- 2KB memory in mercury ultrasonic delay lines
- One program at a time!
- Used in meteorology, genetics, theoretical chemistry, numerical analysis, radioastronomy.
- *"On a few occasions it worked for more than 24 hours."*

Early History: EDSAC (1949–1958)

- **E**lectronic **D**elay **S**torage **A**utomatic **C**alculator
- The second general use, electronic digital (Turing complete) stored program computer
- 3,000 valves
- 650 instructions per second
- 2KB memory in mercury ultrasonic delay lines
- One program at a time!
- Used in meteorology, genetics, theoretical chemistry, numerical analysis, radioastronomy.
- *"On a few occasions it worked for more than 24 hours."*

Early History: Mainframes (1958–1995)

EDSAC 2 (1958–1965) Complete redesign in-house: 10x faster, 80KB memory.

TITAN (1964–1973) Multiuser system, designed with Ferranti.

Phoenix (1971–1995) IBM mainframes, general purpose (including email).

Mainframe service morphs into distributed research computing support with central services.

Specialised research computing needs remain!

Early History: Mainframes (1958–1995)

EDSAC 2 (1958–1965) Complete redesign in-house: 10x faster, 80KB memory.

TITAN (1964–1973) Multiuser system, designed with Ferranti.

Phoenix (1971–1995) IBM mainframes, general purpose (including email).

Mainframe service morphs into distributed research computing support with central services.

Specialised research computing needs remain!

Early History: Mainframes (1958–1995)

EDSAC 2 (1958–1965) Complete redesign in-house: 10x faster, 80KB memory.

TITAN (1964–1973) Multiuser system, designed with Ferranti.

Phoenix (1971–1995) IBM mainframes, general purpose (including email).

Mainframe service morphs into distributed research computing support with central services.

Specialised research computing needs remain!

Early History: Mainframes (1958–1995)

EDSAC 2 (1958–1965) Complete redesign in-house: 10x faster, 80KB memory.

TITAN (1964–1973) Multiuser system, designed with Ferranti.

Phoenix (1971–1995) IBM mainframes, general purpose (including email).

Mainframe service morphs into distributed research computing support with central services.

Specialised research computing needs remain!

Early History: Mainframes (1958–1995)

EDSAC 2 (1958–1965) Complete redesign in-house: 10x faster, 80KB memory.

TITAN (1964–1973) Multiuser system, designed with Ferranti.

Phoenix (1971–1995) IBM mainframes, general purpose (including email).

Mainframe service morphs into distributed research computing support with central services.

Specialised research computing needs remain!

HPCS: A Brief History

Created: 1996 (as the HPCF).

Mission: Delivery and support of a large HPC resource for use by the University of Cambridge research community.

Self-funding: Paying and non-paying service levels.

User base: Includes external STFC & EPSRC plus industrial users.

Plus: Dedicated group nodes and research projects.

HPCS: A Brief History

Created: 1996 (as the HPCF).

Mission: Delivery and support of a large HPC resource for use by the University of Cambridge research community.

Self-funding: Paying and non-paying service levels.

User base: Includes external STFC & EPSRC plus industrial users.

Plus: Dedicated group nodes and research projects.

HPCS: A Brief History

Created:	1996 (as the HPCF).
Mission:	Delivery and support of a large HPC resource for use by the University of Cambridge research community.
Self-funding:	Paying and non-paying service levels.
User base:	Includes external STFC & EPSRC plus industrial users.
Plus:	Dedicated group nodes and research projects.

HPCS: A Brief History of Darwin

1997 76.8 Gflop/s

2002 1.4 Tflop/s

2006 18.27 Tflop/s

2010 30 Tflop/s

2012 183.38 Tflop/s

2013 183.38 CPU + 239.90 GPU Tflop/s

2017 $\sim$ 1.0 CPU + 1.193 GPU + 0.508 KNL Pflop/s

http://www.top500.org

HPCS: A Brief History of Darwin

1997 76.8 Gflop/s

2002 1.4 Tflop/s

2006 18.27 Tflop/s

2010 30 Tflop/s

2012 183.38 Tflop/s

2013 183.38 CPU + 239.90 GPU Tflop/s

2017 $\sim$ 1.0 CPU + 1.193 GPU + 0.508 KNL Pflop/s

http://www.top500.org

HPCS: A Brief History of Darwin

1997 76.8 Gflop/s

2002 1.4 Tflop/s

2006 18.27 Tflop/s

2010 30 Tflop/s

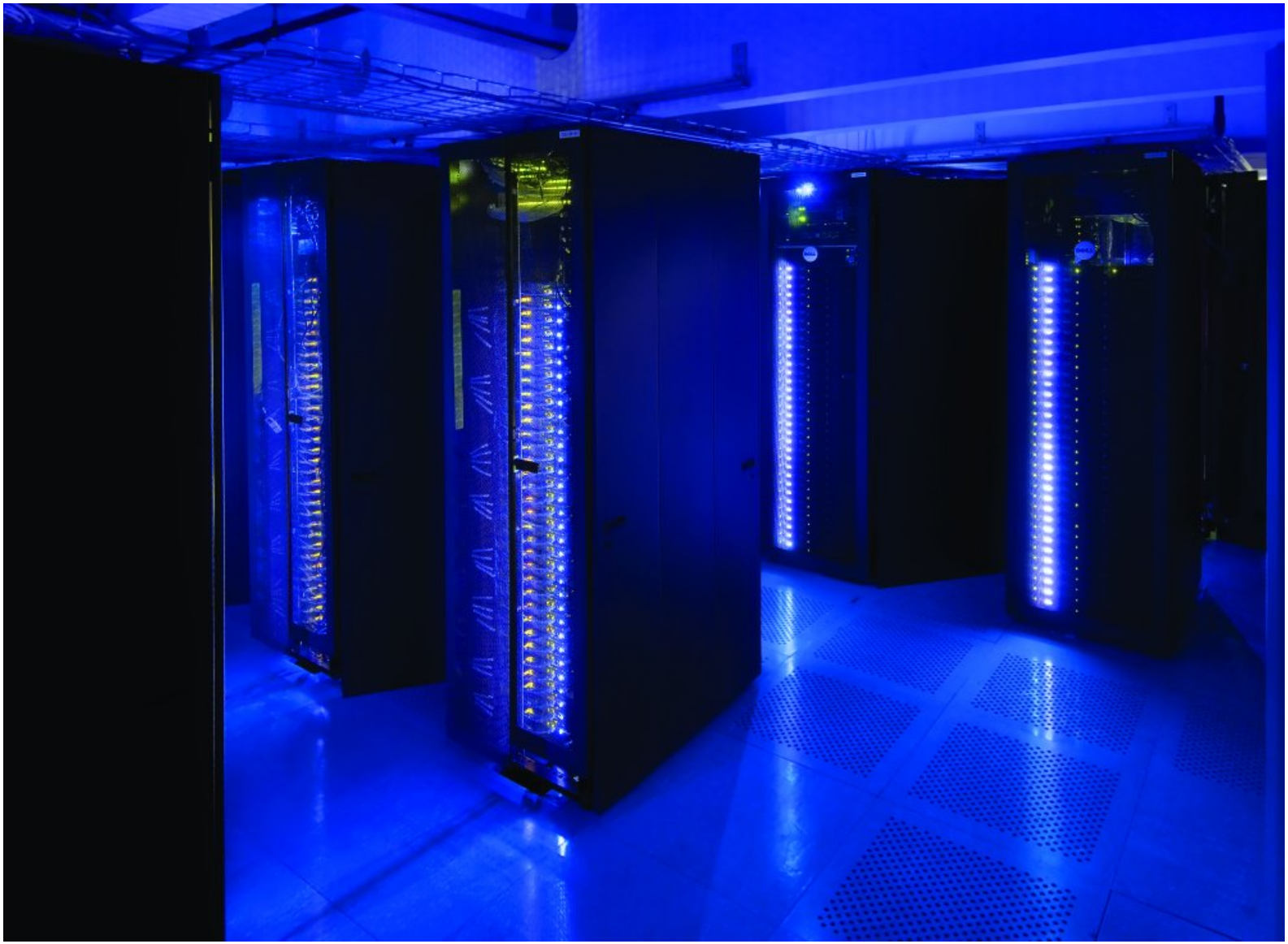
2012 183.38 Tflop/s

2013 183.38 CPU + 239.90 GPU Tflop/s

2017 $\sim$ 1.0 CPU + 1.193 GPU + 0.508 KNL Pflop/s

http://www.top500.org

Darwin1 (2006–2012)



Darwin3 (2012)(b) & Wilkes (2013)(f)

Darwin: an Infiniband CPU Cluster

- Each compute node:
  - 2x8 cores, Intel Sandy Bridge 2.6 GHz.
  - 64 GB RAM (63900 MB usable).
  - 56 Gb/sec (4X FDR) Infiniband.
- 600 compute nodes (300 belong to Cambridge).
- 8 login nodes (login.hpc.cam.ac.uk).

Darwin: an Infiniband CPU Cluster

- Each compute node:
  - 16 cores
  - 63900 MB
  - 5 GB/sec Infiniband (for MPI and storage)
- 600 compute nodes (300 belong to Cambridge).
- 8 login nodes (login.hpc.cam.ac.uk).

Darwin: an Infiniband CPU Cluster

- Each compute node:
  - 16 cores
  - 63900 MB
  - 5 GB/sec Infiniband (for MPI and storage)
- 600 compute nodes (300 belong to Cambridge).
- 8 login nodes (login.hpc.cam.ac.uk).
- 1 Petaflop upgrade availability October 2017

Wilkes: a Dual-Rail Infiniband GPU Cluster

- Each compute node:
  - $2 \times$ NVIDIA Tesla K20c GPU.
  - 2x6 cores, Intel Ivy Bridge 2.6 GHz.
  - 64 GB RAM (63900 MB usable).
  - 2×56 Gb/sec (4X FDR) Infiniband.
- 128 compute nodes.
- 8 login nodes (login.hpc.cam.ac.uk).
- Environment shared with Darwin (same filesystems, user environment, scheduler).
- 1 Petaflop upgrade (Wilkes2) early access June 2017

Wilkes: a Dual-Rail Infiniband GPU Cluster

- Each compute node:
  - 2 GPUs
  - 12 cores
  - 63900 MB
  - 2×5 GB/sec Infiniband (for MPI and storage)
- 128 compute nodes.
- 8 login nodes (login.hpc.cam.ac.uk).
- Environment shared with Darwin (same filesystems, user environment, scheduler).
- 1 Petaflop upgrade (Wilkes2) early access June 2017

Wilkes: a Dual-Rail Infiniband GPU Cluster

- Each compute node:
  - 2 GPUs
  - 12 cores
  - 63900 MB
  - 2×5 GB/sec Infiniband (for MPI and storage)
- 128 compute nodes.
- 8 login nodes (login.hpc.cam.ac.uk).
- Environment shared with Darwin (same filesystems, user environment, scheduler).
- 1 Petaflop upgrade (Wilkes2) early access June 2017

Wilkes: a Dual-Rail Infiniband GPU Cluster

- Each compute node:
  - 2 GPUs
  - 12 cores
  - 63900 MB
  - 2×5 GB/sec Infiniband (for MPI and storage)
- 128 compute nodes.
- 8 login nodes (login.hpc.cam.ac.uk).
- Environment shared with Darwin (same filesystems, user environment, scheduler).
- 1 Petaflop upgrade (Wilkes2) early access June 2017

HPCS: Storage

- Multi-petabytes split across multiple filesystems with tape.
- Lustre cluster filesystem:
  - Multiple RAID6 back-end disk volumes.
  - Multiple object storage servers.
  - Single metadata server.
  - Tape-backed HSM on newest filesystems.
  - 4 GB/sec overall read or write.
  - Prefers big read/writes over small.
- For active HPC work only.

HPCS: Storage

- Multi-petabytes split across multiple filesystems with tape.
- Lustre cluster filesystem:
  - Multiple RAID6 back-end disk volumes.
  - Multiple object storage servers.
  - Single metadata server.
  - Tape-backed HSM on newest filesystems.
  - 4 GB/sec overall read or write.
  - Prefers big read/writes over small.
- For active HPC work only.

HPCS: Storage

- Multi-petabytes split across multiple filesystems with tape.
- Lustre cluster filesystem:
  - Multiple RAID6 back-end disk volumes.
  - Multiple object storage servers.
  - Single metadata server.
  - Tape-backed HSM on newest filesystems.
  - 4 GB/sec overall read or write.
  - Prefers big read/writes over small.
- For active HPC work only.

HPCS: Storage

- Multi-petabytes split across multiple filesystems with tape.
- Lustre cluster filesystem:
  - Multiple RAID6 back-end disk volumes.
  - Multiple object storage servers.
  - Single metadata server.
  - Tape-backed HSM on newest filesystems.
  - 4 GB/sec overall read or write.
  - Prefers big read/writes over small.
- For active HPC work only.

HPCS: Service Levels

Service Level 1 Paying, intended for large projects with long-term, consistent requirement.

Service Level 2 Paying, intended for medium-sized projects with irregular requirement.

Service Level 3 Non-paying, intended for interim or pump priming, small-scale use.

HPCS: Service Levels

Service Level 1 Paying, intended for large projects with long-term, consistent requirement.

- Guaranteed fraction of resources per quarter.

Service Level 2 Paying, intended for medium-sized projects with irregular requirement.

Service Level 3 Non-paying, intended for interim or pump priming, small-scale use.

HPCS: Service Levels

Service Level 1 Paying, intended for large projects with long-term, consistent requirement.

- Guaranteed fraction of resources per quarter.

Service Level 2 Paying, intended for medium-sized projects with irregular requirement.

- High priority, but no guarantees; *ad hoc*.

Service Level 3 Non-paying, intended for interim or pump priming, small-scale use.

HPCS: Service Levels

Service Level 1 Paying, intended for large projects with long-term, consistent requirement.

- Guaranteed fraction of resources per quarter.

Service Level 2 Paying, intended for medium-sized projects with irregular requirement.

- High priority, but no guarantees; *ad hoc*.

Service Level 3 Non-paying, intended for interim or pump priming, small-scale use.

- Low priority, limited usage (200,000 Darwin core hours) per quarter.

HPCS: Service Levels

Service Level 2 Paying, intended for medium-sized projects with irregular requirement.

- High priority, but no guarantees; *ad hoc*.

HPCS: Service Levels

Service Level 4 Non-paying, for when nothing else is available.

Very low priority, very restricted, very limited. Best efforts continuation.

HPCS: Service Levels

Service Level 4 Non-paying, for when nothing else is available.

Very low priority, very restricted, very limited. Best efforts continuation.

The West Cambridge Data Centre

HPCS: How To Apply

- Submit the online application form:

  https://www.hpc.cam.ac.uk/services/applying-for-resources/hpc-application

- The PI should be someone senior enough to have funding.
  E.g. supervisor, head of research group.
- Funding is not necessary.
- Please email support@hpc.cam.ac.uk for all support issues.
- Further information can be found on the web site:
  http://www.hpc.cam.ac.uk
- Imminent upgrades may introduce changes.

HPCS: How To Apply

- Submit the online application form:

  https://www.hpc.cam.ac.uk/services/applying-for-resources/hpc-application

- The PI should be someone senior enough to have funding.
  E.g. supervisor, head of research group.
- Funding is not necessary.
- Please email support@hpc.cam.ac.uk for all support issues.
- Further information can be found on the web site:
  http://www.hpc.cam.ac.uk
- Imminent upgrades may introduce changes.

HPCS: How To Apply

- Submit the online application form:

  https://www.hpc.cam.ac.uk/services/applying-for-resources/hpc-application

- The PI should be someone senior enough to have funding.
  E.g. supervisor, head of research group.
- Funding is not necessary.
- Please email support@hpc.cam.ac.uk for all support issues.
- Further information can be found on the web site:
  http://www.hpc.cam.ac.uk
- Imminent upgrades may introduce changes.

HPCS: How To Apply

- Submit the online application form:

  https://www.hpc.cam.ac.uk/services/applying-for-resources/hpc-application

- The PI should be someone senior enough to have funding.
  E.g. supervisor, head of research group.
- Funding is not necessary.
- Please email support@hpc.cam.ac.uk for all support issues.
- Further information can be found on the web site:
  http://www.hpc.cam.ac.uk
- Imminent upgrades may introduce changes.

Part III: **Using HPC**

Using HPC: Security

- Boring but very, very important . . .
- Cambridge IT is under constant attack by would-be intruders.
- Your data and research career is threatened by intruders.
- Big systems are big, juicy targets.
- Anything in the University of Cambridge is a big, juicy target.
- Don't let intruders in.

Using HPC: Security

- Boring but very, very important . . .
- Cambridge IT is under constant attack by would-be intruders.
- Your data and research career is threatened by intruders.
- Big systems are big, juicy targets.
- Anything in the University of Cambridge is a big, juicy target.
- Don't let intruders in.

Using HPC: Security

- Boring but very, very important . . .
- Cambridge IT is under constant attack by would-be intruders.
- Your data and research career is threatened by intruders.
- Big systems are big, juicy targets.
- Anything in the University of Cambridge is a big, juicy target.
- Don't let intruders in.

Using HPC: Security

- Boring but very, very important . . .
- Cambridge IT is under constant attack by would-be intruders.
- Your data and research career is threatened by intruders.
- Big systems are big, juicy targets.
- Anything in the University of Cambridge is a big, juicy target.
- Don't let intruders in.

Using HPC: Security

- Boring but very, very important . . .
- Cambridge IT is under constant attack by would-be intruders.
- Your data and research career is threatened by intruders.
- Big systems are big, juicy targets.
- Anything in the University of Cambridge is a big, juicy target.
- Don't let intruders in.

Using HPC: Security

- Boring but very, very important . . .
- Cambridge IT is under constant attack by would-be intruders.
- Your data and research career is threatened by intruders.
- Big systems are big, juicy targets.
- Anything in the University of Cambridge is a big, juicy target.
- Don't let intruders in.

Using HPC: Security

1. Keep your password (or private key passphrase) safe.
2. Choose a strong UIS password.
3. Keep the software on your laptops/tablets/PCs up to date.
4. Don't share accounts (this is against the rules anyway).

Using HPC: Security

1. Keep your password (or private key passphrase) safe.
2. Choose a strong UIS password.
3. Keep the software on your laptops/tablets/PCs up to date.
4. Don't share accounts (this is against the rules anyway).

Using HPC: Security

1. Keep your password (or private key passphrase) safe.
2. Choose a strong UIS password.
3. Keep the software on your laptops/tablets/PCs up to date.
4. Don't share accounts (this is against the rules anyway).

Using HPC: Security

1. Keep your password (or private key passphrase) safe.
2. Choose a strong UIS password.
3. Keep the software on your laptops/tablets/PCs up to date.
4. Don't share accounts (this is against the rules anyway).

Using HPC: Connecting

- SSH secure protocol only.
- HPCS allows access from registered IP addresses only.

Using HPC: Connecting

- SSH secure protocol only.
  Supports login, file transfer, remote desktop...
- HPCS allows access from registered IP addresses only.

Using HPC: Connecting

- SSH secure protocol only.
  Supports login, file transfer, remote desktop...
- HPCS allows access from registered IP addresses only.

Using HPC: Connecting

- SSH secure protocol only.
  Supports login, file transfer, remote desktop...
- HPCS allows access from registered IP addresses only.
  Almost all Cambridge University addresses already registered.

Using HPC: Connecting

- SSH secure protocol only.
  Supports login, file transfer, remote desktop...
- HPCS allows access from registered IP addresses only.
  Almost all Cambridge University addresses already registered.
  Connection from home possible via the VPN service
  http://www.ucs.cam.ac.uk/vpn

Using HPC: Connecting

- SSH secure protocol only.
  Supports login, file transfer, remote desktop...
- HPCS allows access from registered IP addresses only.
  Almost all Cambridge University addresses already registered.
  Connection from home possible via the VPN service
  http://www.ucs.cam.ac.uk/vpn
  or SSH tunnel through a departmental gateway.

Connecting: Windows Clients

- putty, pscp, psftp
  http://www.chiark.greenend.org.uk/ sgtatham/putty/download.html
- WinSCP
  http://winscp.net/eng/download.php
- TurboVNC (remote desktop, 3D optional)
  http://sourceforge.net/projects/turbovnc/files/
- Cygwin
  http://cygwin.com/install.html
- MobaXterm
  http://mobaxterm.mobatek.net/

Connecting: Windows Clients

- putty, pscp, psftp
  http://www.chiark.greenend.org.uk/ sgtatham/putty/download.html
- WinSCP
  http://winscp.net/eng/download.php
- TurboVNC (remote desktop, 3D optional)
  http://sourceforge.net/projects/turbovnc/files/
- Cygwin
  http://cygwin.com/install.html
- MobaXterm
  http://mobaxterm.mobatek.net/

Connecting: Windows Clients

- putty, pscp, psftp
  http://www.chiark.greenend.org.uk/ sgtatham/putty/download.html
- WinSCP
  http://winscp.net/eng/download.php
- TurboVNC (remote desktop, 3D optional)
  http://sourceforge.net/projects/turbovnc/files/
- Cygwin
  http://cygwin.com/install.html
- MobaXterm
  http://mobaxterm.mobatek.net/

Connecting: Windows Clients

- putty, pscp, psftp
  http://www.chiark.greenend.org.uk/ sgtatham/putty/download.html
- WinSCP
  http://winscp.net/eng/download.php
- TurboVNC (remote desktop, 3D optional)
  http://sourceforge.net/projects/turbovnc/files/
- Cygwin
  http://cygwin.com/install.html

- MobaXterm
  http://mobaxterm.mobatek.net/

Connecting: Windows Clients

- putty, pscp, psftp
  http://www.chiark.greenend.org.uk/ sgtatham/putty/download.html
- WinSCP
  http://winscp.net/eng/download.php
- TurboVNC (remote desktop, 3D optional)
  http://sourceforge.net/projects/turbovnc/files/
- Cygwin (provides an application environment similar to Linux)
  http://cygwin.com/install.html
  Includes X server for displaying graphical applications running remotely.
- MobaXterm
  http://mobaxterm.mobatek.net/

Connecting: Windows Clients

- putty, pscp, psftp
  http://www.chiark.greenend.org.uk/ sgtatham/putty/download.html
- WinSCP
  http://winscp.net/eng/download.php
- TurboVNC (remote desktop, 3D optional)
  http://sourceforge.net/projects/turbovnc/files/
- Cygwin
  http://cygwin.com/install.html

- MobaXterm
  http://mobaxterm.mobatek.net/

Connecting: Linux/MacOSX/UNIX Clients

- ssh, scp, sftp, rsync
  Installed (or installable).
- TurboVNC (remote desktop, 3D optional)
  http://sourceforge.net/projects/turbovnc/files/
- On MacOSX, install XQuartz to display remote graphical applications.
  http://xquartz.macosforge.org/landing/

Connecting: Linux/MacOSX/UNIX Clients

- ssh, scp, sftp, rsync
  Installed (or installable).
- TurboVNC (remote desktop, 3D optional)
  http://sourceforge.net/projects/turbovnc/files/
- On MacOSX, install XQuartz to display remote graphical applications.
  http://xquartz.macosforge.org/landing/

Connecting: Linux/MacOSX/UNIX Clients

- ssh, scp, sftp, rsync
  Installed (or installable).
- TurboVNC (remote desktop, 3D optional)
  http://sourceforge.net/projects/turbovnc/files/
- On MacOSX, install XQuartz to display remote graphical applications.
  http://xquartz.macosforge.org/landing/

Connecting: Linux/MacOSX/UNIX Clients

- ssh, scp, sftp, rsync
  Installed (or installable).
- TurboVNC (remote desktop, 3D optional)
  http://sourceforge.net/projects/turbovnc/files/
- On MacOSX, install XQuartz to display remote graphical applications.
  http://xquartz.macosforge.org/landing/

Connecting: Login

- From Linux/MacOSX/UNIX (or Cygwin):
  ssh -Y **abc123**@login.hpc.cam.ac.uk
- From graphical clients:
  Host: login.hpc.cam.ac.uk
  Username: **abc123** (your UCAM account name)
- login.hpc will map to a random login node
  i.e. one of login-sand1, login-sand2, . . . , login-sand8
  NB Not darwin.hpc (the head node).
- Non-registered addresses will fail with "Connection refused".
- Similarly for other systems (e.g. cardio-login.hpc, login-mrc-bsu.hpc, . . .).

Connecting: Login

- From Linux/MacOSX/UNIX (or Cygwin):
  ssh -Y **abc123**@login.hpc.cam.ac.uk
- From graphical clients:
  Host: login.hpc.cam.ac.uk
  Username: **abc123** (your UCAM account name)
- login.hpc will map to a random login node
  i.e. one of login-sand1, login-sand2, . . . , login-sand8
  NB Not darwin.hpc (the head node).
- Non-registered addresses will fail with "Connection refused".
- Similarly for other systems (e.g. cardio-login.hpc, login-mrc-bsu.hpc, . . .).

Connecting: Login

- From Linux/MacOSX/UNIX (or Cygwin):
  ssh -Y **abc123**@login.hpc.cam.ac.uk
- From graphical clients:
  Host: login.hpc.cam.ac.uk
  Username: **abc123** (your UCAM account name)
- login.hpc will map to a random login node
  i.e. one of login-sand1, login-sand2, . . . , login-sand8
  NB Not darwin.hpc (the head node).
- Non-registered addresses will fail with "Connection refused".
- Similarly for other systems (e.g. cardio-login.hpc, login-mrc-bsu.hpc,. . .).

Connecting: Login

- From Linux/MacOSX/UNIX (or Cygwin):
  ssh -Y **abc123**@login.hpc.cam.ac.uk
- From graphical clients:
  Host: login.hpc.cam.ac.uk
  Username: **abc123** (your UCAM account name)
- login.hpc will map to a random login node
  i.e. one of login-sand1, login-sand2, . . . , login-sand8
  NB Not darwin.hpc (the head node).
- Non-registered addresses will fail with "Connection refused".
- Similarly for other systems (e.g. cardio-login.hpc, login-mrc-bsu.hpc,. . .).

Connecting: Login

- From Linux/MacOSX/UNIX (or Cygwin):
  ssh -Y **abc123**@login.hpc.cam.ac.uk
- From graphical clients:
  Host: login.hpc.cam.ac.uk
  Username: **abc123** (your UCAM account name)
- login.hpc will map to a random login node
  i.e. one of login-sand1, login-sand2, . . . , login-sand8
  NB Not darwin.hpc (the head node).
- Non-registered addresses will fail with "Connection refused".
- Similarly for other systems (e.g. cardio-login.hpc, login-mrc-bsu.hpc,. . .).

Connecting: Login

- From Linux/MacOSX/UNIX (or Cygwin):
  ssh -Y **abc123**@login.hpc.cam.ac.uk
- From graphical clients:
  Host: login.hpc.cam.ac.uk
  Username: **abc123** (your UCAM account name)
- login.hpc will map to a random login node
  i.e. one of login-sand1, login-sand2, . . . , login-sand8
  NB Not darwin.hpc (the head node).
- Non-registered addresses will fail with "Connection refused".
- Similarly for other systems (e.g. cardio-login.hpc, login-mrc-bsu.hpc,. . .).

Connecting: First time login

- The first connection to a particular hostname produces the following:

```
The authenticity of host 'login-sand2.hpc.cam.ac.uk (131.111.1.214)'
can't be established.
RSA key fingerprint is
0b:ef:59:90:fb:13:4a:c9:56:82:7b:cd:4b:2b:e1:3b.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'login-sand2.hpc.cam.ac.uk' (RSA) to the
list of known hosts.
```

- One should always check the fingerprint before typing "yes".
- Graphical SSH clients *should* ask a similar question.
- Designed to detect fraudulent servers.

Connecting: First time login

- The first connection to a particular hostname produces the following:

```
The authenticity of host 'login-sand2.hpc.cam.ac.uk (131.111.1.214)'
can't be established.
RSA key fingerprint is
0b:ef:59:90:fb:13:4a:c9:56:82:7b:cd:4b:2b:e1:3b.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'login-sand2.hpc.cam.ac.uk' (RSA) to the
list of known hosts.
```

- One should always check the fingerprint before typing "yes".
- Graphical SSH clients *should* ask a similar question.
- Designed to detect fraudulent servers.

Connecting: First time login

- The first connection to a particular hostname produces the following:

```
The authenticity of host 'login-sand2.hpc.cam.ac.uk (131.111.1.214)'
can't be established.
RSA key fingerprint is
0b:ef:59:90:fb:13:4a:c9:56:82:7b:cd:4b:2b:e1:3b.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'login-sand2.hpc.cam.ac.uk' (RSA) to the
list of known hosts.
```

- One should always check the fingerprint before typing "yes".
- Graphical SSH clients *should* ask a similar question.
- Designed to detect fraudulent servers.

Connecting: First time login

- You may be presented with any of the following fingerprints (depending on your client):

```
MD5:0b:ef:59:90:fb:13:4a:c9:56:82:7b:cd:4b:2b:e1:3b
SHA256:sSkVfzpwjwiFvxLcdPoDpN8IsN3kt0ZSywhDhPKZPAg

MD5:34:9b:f2:d2:c6:b3:5c:63:99:b7:27:da:5b:c8:16:fe
SHA256:HsiY1Oe0M8tS6JwR76PeQQA/VB7r8675BzG5OYQ4h34

MD5:64:7c:7c:ff:05:9d:0e:dc:06:fe:f1:c2:10:37:7a:85
SHA256:wq9ljBfPa7lXXpQq+rk5JTBXLJO/kXjOc5A7rp4ENzA
```

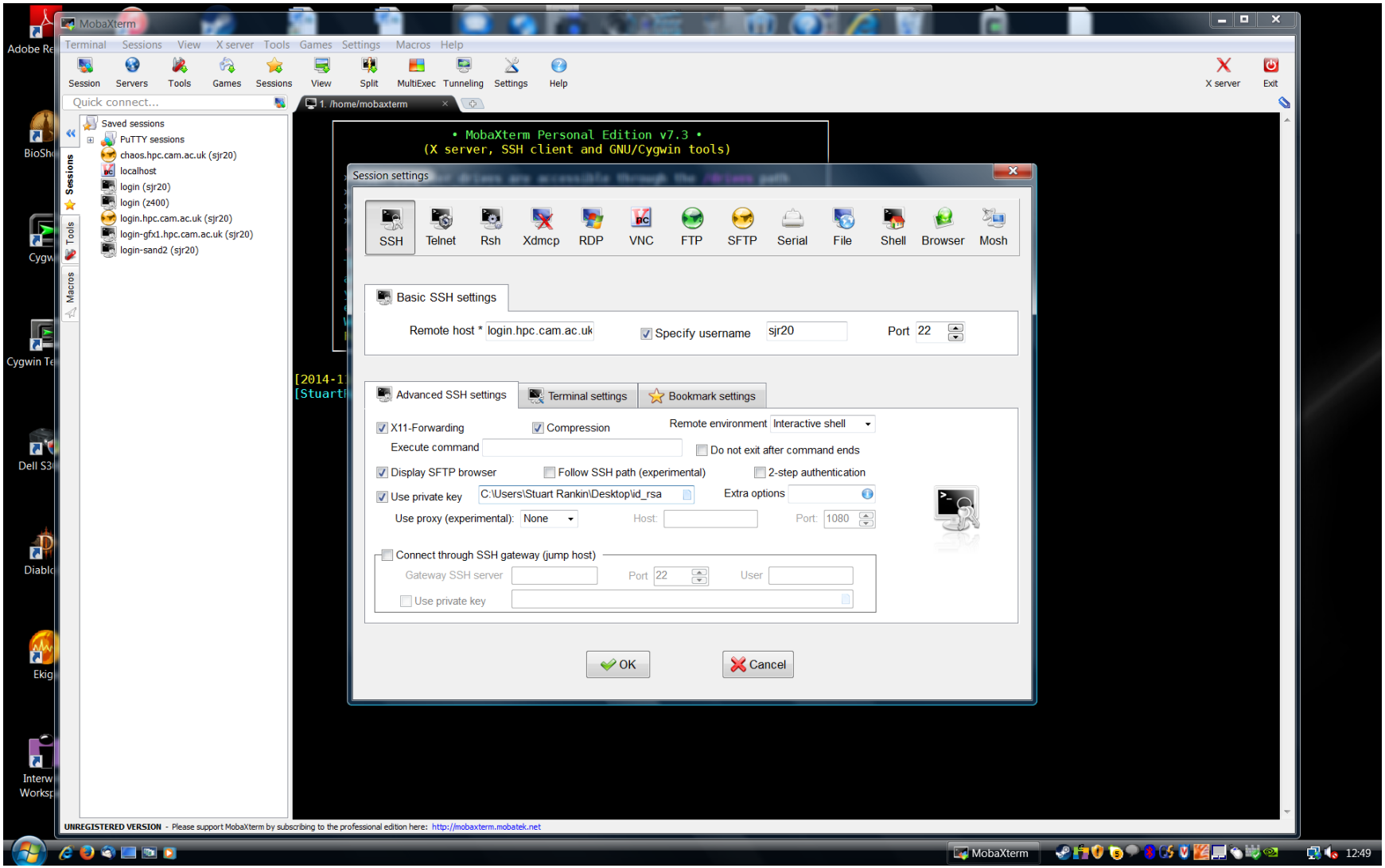
Connecting: First time login

- You may be presented with any of the following fingerprints (depending on your client):

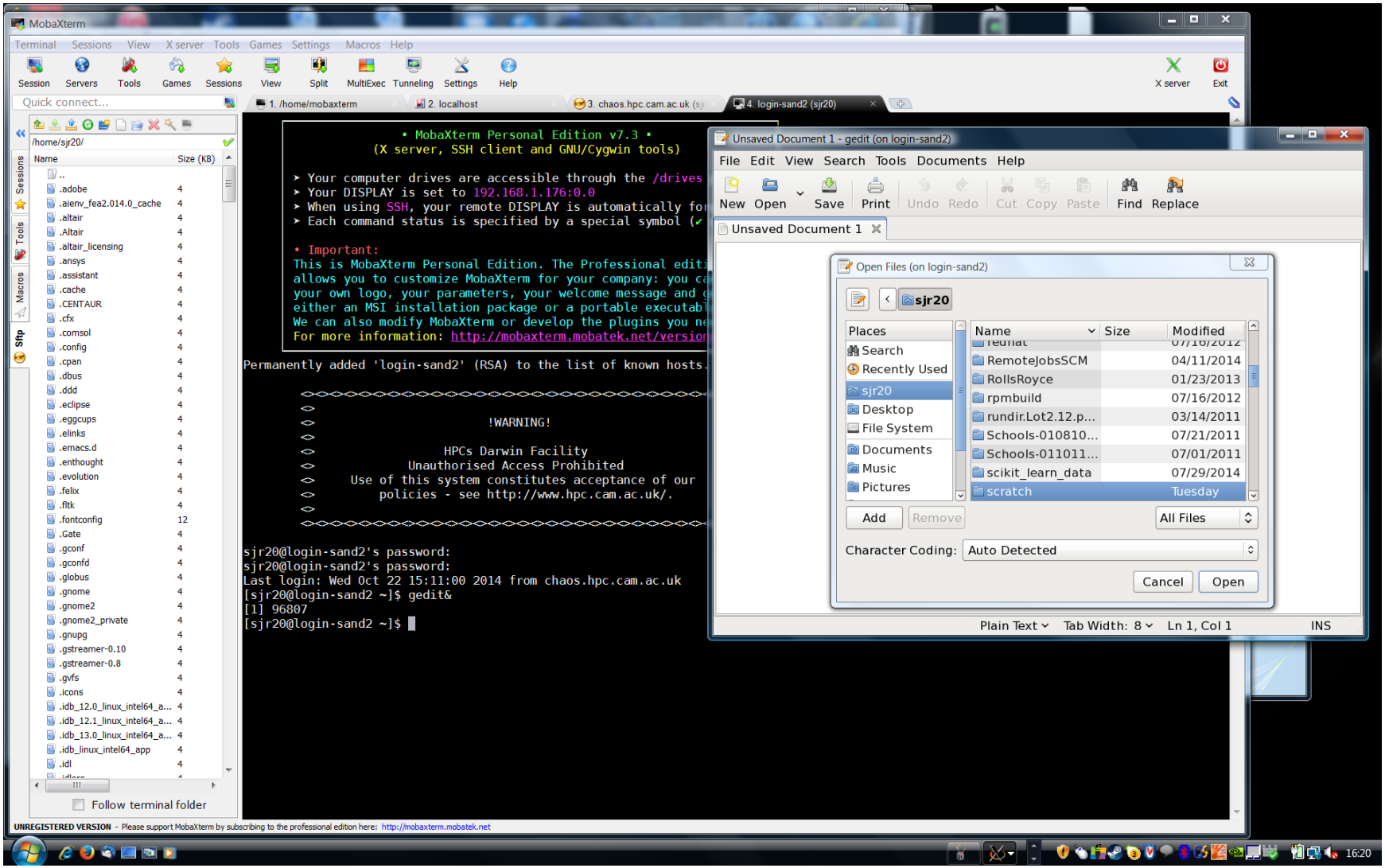
```
MD5:0b:ef:59:90:fb:13:4a:c9:56:82:7b:cd:4b:2b:e1:3b
SHA256:sSkVfzpwjwiFvxLcdPoDpN8IsN3kt0ZSywhDhPKZPAg

MD5:34:9b:f2:d2:c6:b3:5c:63:99:b7:27:da:5b:c8:16:fe
SHA256:HsiY1Oe0M8tS6JwR76PeQQA/VB7r8675BzG5OYQ4h34

MD5:64:7c:7c:ff:05:9d:0e:dc:06:fe:f1:c2:10:37:7a:85
SHA256:wq9ljBfPa7lXXpQq+rk5JTBXLJO/kXjOc5A7rp4ENzA
```

- Exercise 1 - Log into your HPCS training account.

MobaXterm SSH (Windows)



MobaXterm SSH (Windows)



Connecting: File Transfer

- From Linux/MacOSX/UNIX (or Cygwin):

  rsync -av **old_directory/** abc123@login.hpc.cam.ac.uk:scratch/new_directory

  copies contents of old_directory to ~/scratch/new_directory.

  rsync -av **old_directory** abc123@login.hpc.cam.ac.uk:scratch/new_directory

  copies old_directory (and contents) to ~/scratch/new_directory/old_directory.

  * Rerun to update or resume after interruption.
  * All transfers are checksummed.
  * For transfers in the opposite direction, place the remote machine as the first argument.

- With graphical clients, connect as before and drag and drop.
- Exercise 2 - File transfer.

Connecting: File Transfer

- From Linux/MacOSX/UNIX (or Cygwin):

  rsync -av **old_directory/** abc123@login.hpc.cam.ac.uk:scratch/new_directory

  copies contents of old_directory to ~/scratch/new_directory.

  rsync -av **old_directory** abc123@login.hpc.cam.ac.uk:scratch/new_directory

  copies old_directory (and contents) to ~/scratch/new_directory/old_directory.

  * Rerun to update or resume after interruption.
  * All transfers are checksummed.
  * For transfers in the opposite direction, place the remote machine as the first argument.

- With graphical clients, connect as before and drag and drop.
- Exercise 2 - File transfer.

Connecting: File Transfer

- From Linux/MacOSX/UNIX (or Cygwin):

  rsync -av **old_directory/** abc123@login.hpc.cam.ac.uk:scratch/new_directory

  copies contents of old_directory to ~/scratch/new_directory.

  rsync -av **old_directory** abc123@login.hpc.cam.ac.uk:scratch/new_directory

  copies old_directory (and contents) to ~/scratch/new_directory/old_directory.

  * Rerun to update or resume after interruption.
  * All transfers are checksummed.
  * For transfers in the opposite direction, place the remote machine as the first argument.

- With graphical clients, connect as before and drag and drop.
- Exercise 2 - File transfer.

Connecting: File Transfer

- From Linux/MacOSX/UNIX (or Cygwin):

  rsync -av **old_directory/** abc123@login.hpc.cam.ac.uk:scratch/new_directory

  copies contents of old_directory to ~/scratch/new_directory.

  rsync -av **old_directory** abc123@login.hpc.cam.ac.uk:scratch/new_directory

  copies old_directory (and contents) to ~/scratch/new_directory/old_directory.

  * Rerun to update or resume after interruption.
  * All transfers are checksummed.
  * For transfers in the opposite direction, place the remote machine as the first argument.

- With graphical clients, connect as before and drag and drop.
- Exercise 2 - File transfer.

Connecting: File Transfer

- From Linux/MacOSX/UNIX (or Cygwin):

  rsync -av **old_directory/** abc123@login.hpc.cam.ac.uk:scratch/new_directory

  copies contents of old_directory to ~/scratch/new_directory.

  rsync -av **old_directory** abc123@login.hpc.cam.ac.uk:scratch/new_directory

  copies old_directory (and contents) to ~/scratch/new_directory/old_directory.

  * Rerun to update or resume after interruption.
  * All transfers are checksummed.
  * For transfers in the opposite direction, place the remote machine as the first argument.
- With graphical clients, connect as before and drag and drop.
- Exercise 2 - File transfer.

Connecting: Remote Desktop

- First time starting a remote desktop:

```
[sjr20@login-sand2 ~]$ vncserver

You will require a password to access your desktops.

Password:
Verify:
Would you like to enter a view-only password (y/n)? n

New 'X' desktop is login-sand2:8

Starting applications specified in /home/sjr20/.vnc/xstartup.turbovnc
Log file is /home/sjr20/.vnc/login-sand2:8.log
```

- For 3D graphics sessions, use login-gfx1 or login-gfx2.

Connecting: Remote Desktop

- First time starting a remote desktop:

```
[sjr20@login-sand2 ~]$ vncserver

You will require a password to access your desktops.

Password:
Verify:
Would you like to enter a view-only password (y/n)? n

New 'X' desktop is login-sand2:8

Starting applications specified in /home/sjr20/.vnc/xstartup.turbovnc
Log file is /home/sjr20/.vnc/login-sand2:8.log
```

- For 3D graphics sessions, use login-gfx1 or login-gfx2.

Connecting: Remote Desktop

- Remote desktop already running:

```
[sjr20@login-sand2 ~]$ vncserver -list

TurboVNC server sessions:

X DISPLAY #     PROCESS ID
:8              12745
```

- Kill it:

```
[sjr20@login-sand2 ~]$ vncserver -kill :8
Killing Xvnc process ID 12745
```

- Typically you only need one remote desktop.
- Keeps running until killed, or the node reboots.

Connecting: Remote Desktop

- To connect to the desktop from Linux:

```
[sjr20@themis ~]$ vncviewer -via sjr20@login-sand2.hpc.cam.ac.uk localhost:8
Connected to RFB server, using protocol version 3.8
Enabling TightVNC protocol extensions
Performing standard VNC authentication
Password:
```

- Press F8 to bring up the control panel.
- Exercise 3 - Remote desktop

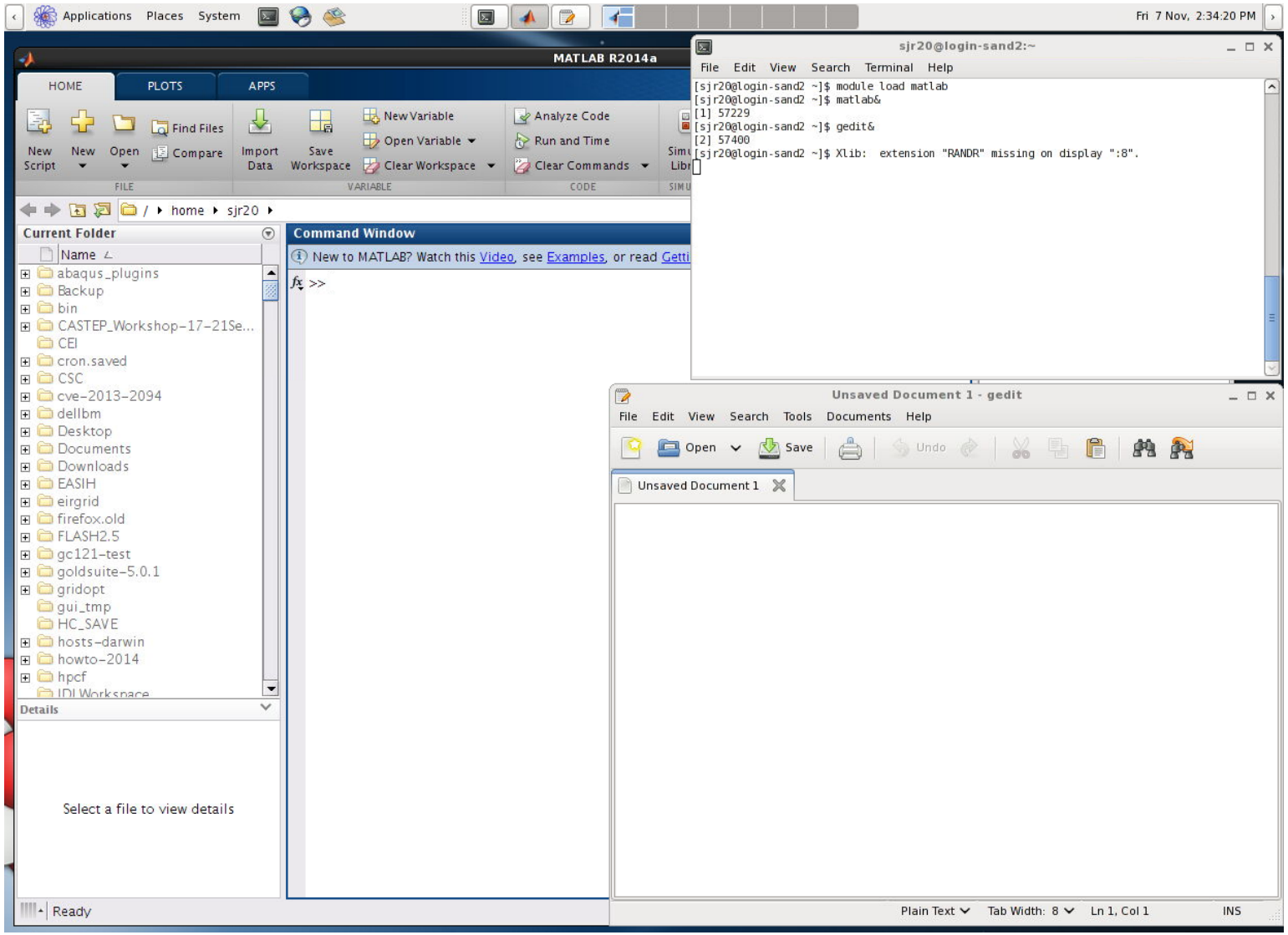
Connecting: Remote Desktop

- To connect to the desktop from Linux:

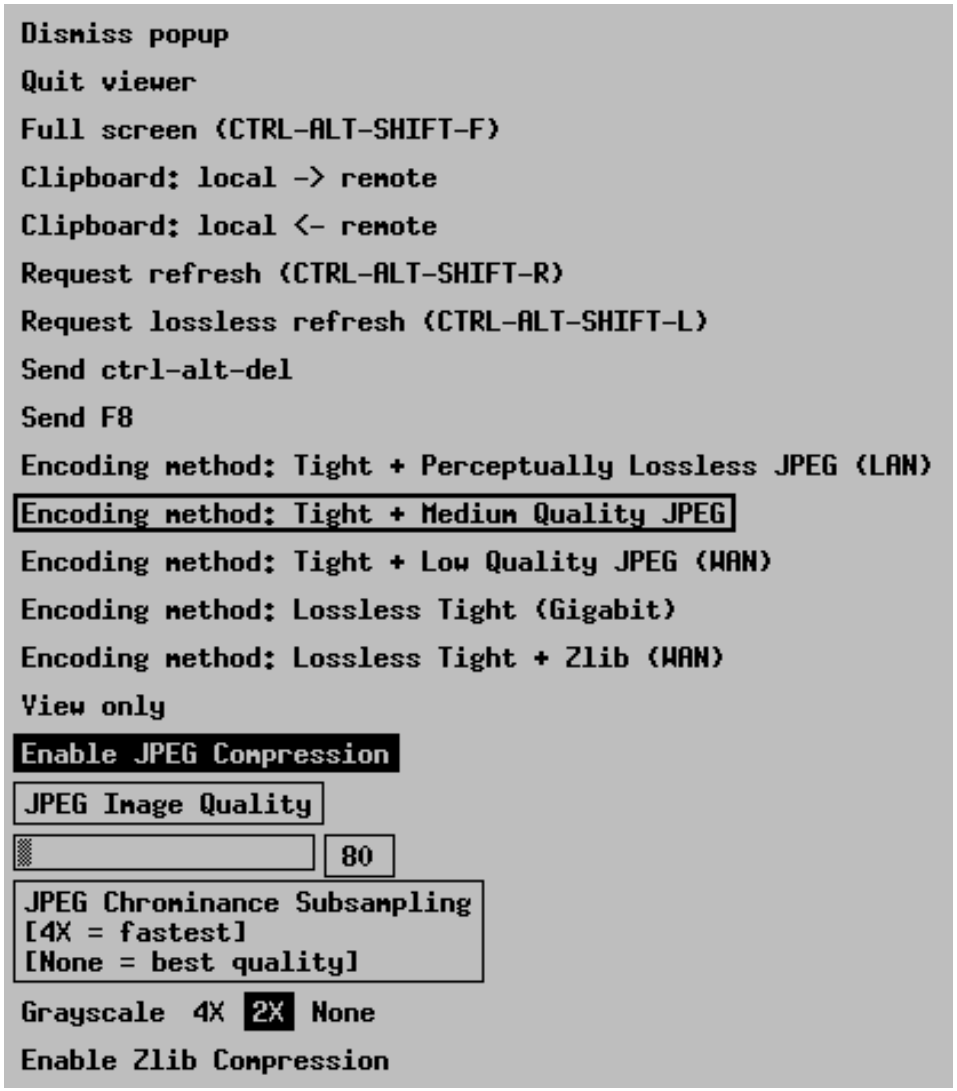
```
[sjr20@themis ~]$ vncviewer -via sjr20@login-sand2.hpc.cam.ac.uk localhost:8
Connected to RFB server, using protocol version 3.8
Enabling TightVNC protocol extensions
Performing standard VNC authentication
Password:
```

- Press F8 to bring up the control panel.
- Exercise 3 - Remote desktop

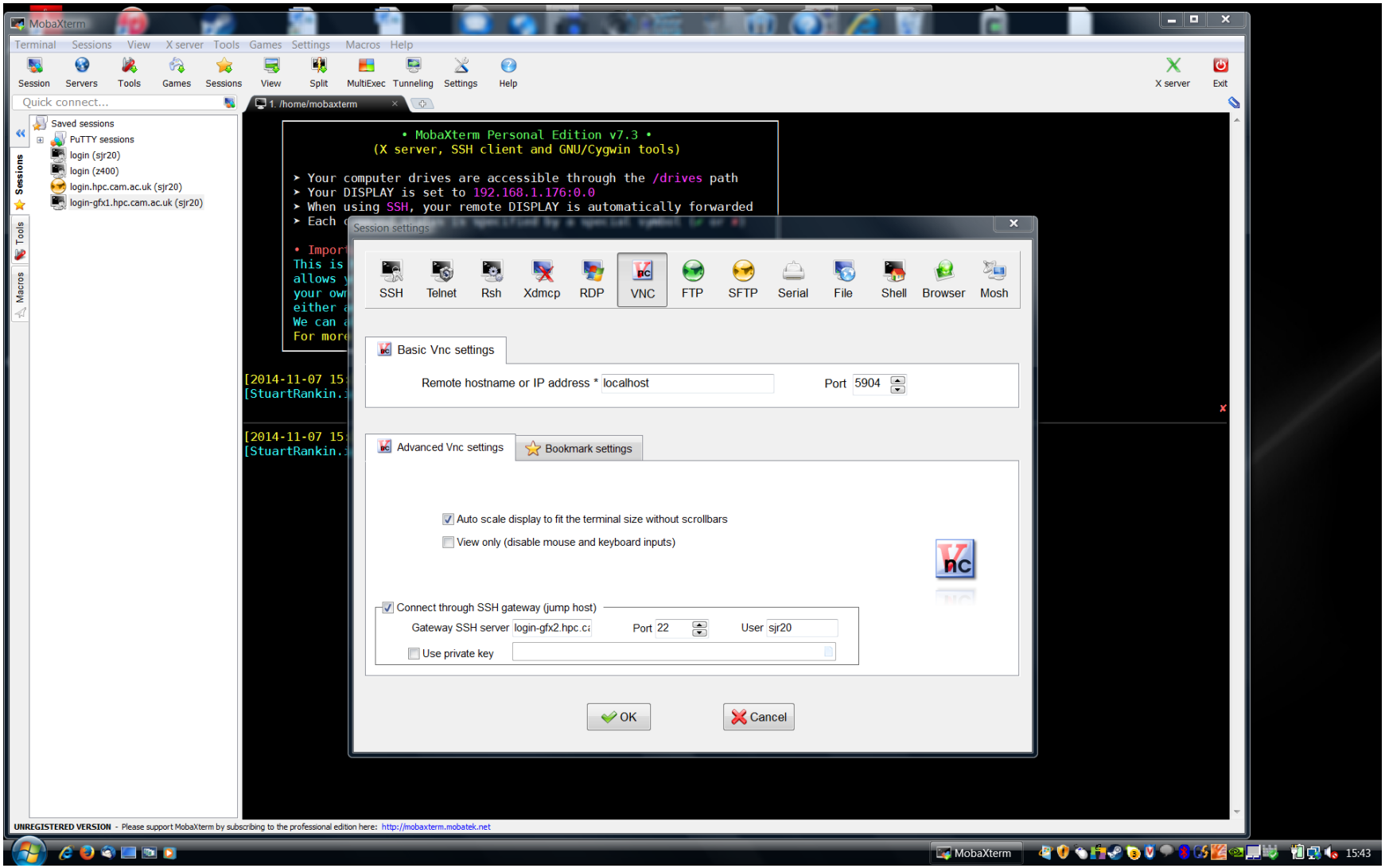
HPCS TurboVNC Session



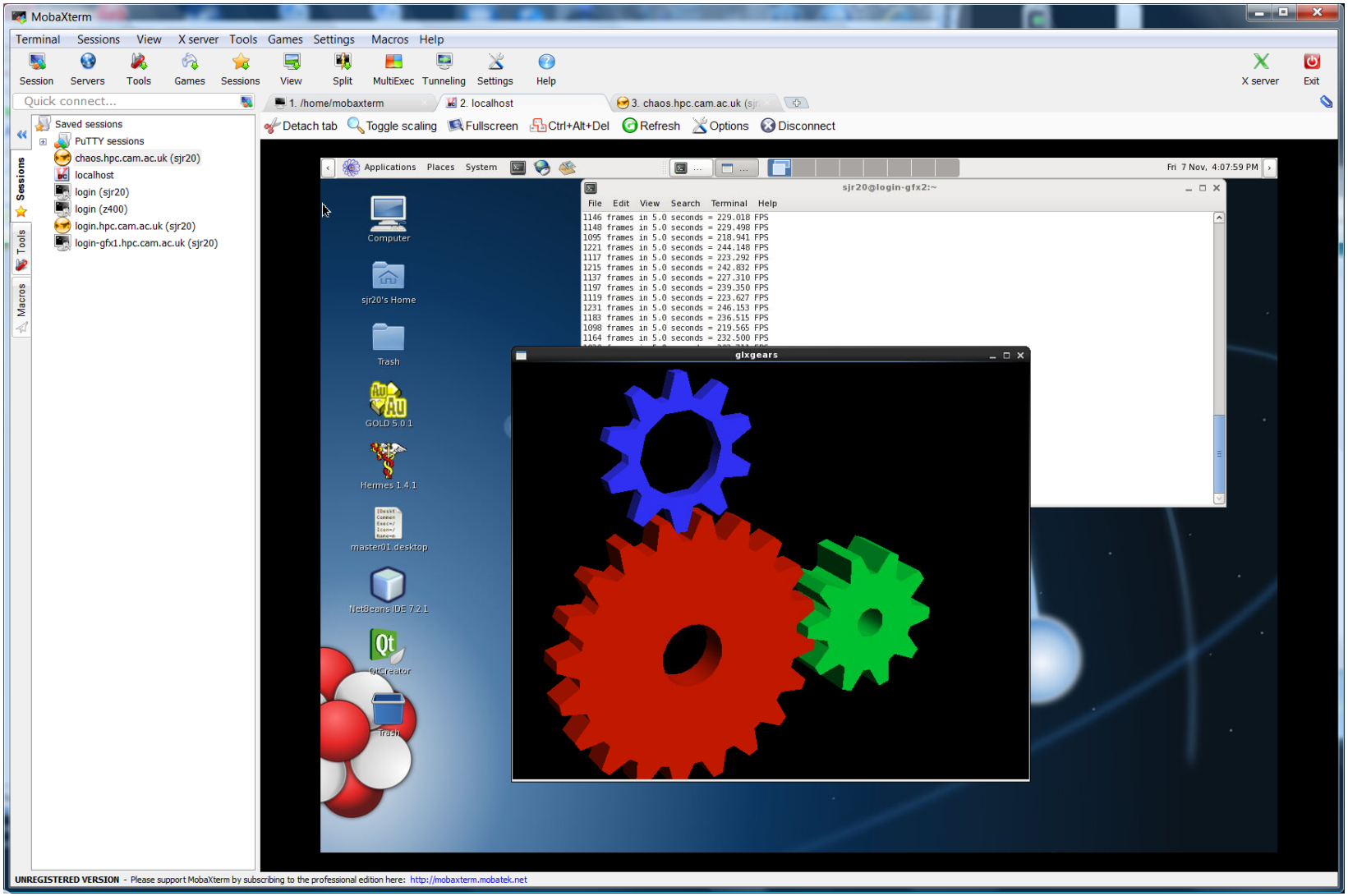
Linux TurboVNC Control Panel



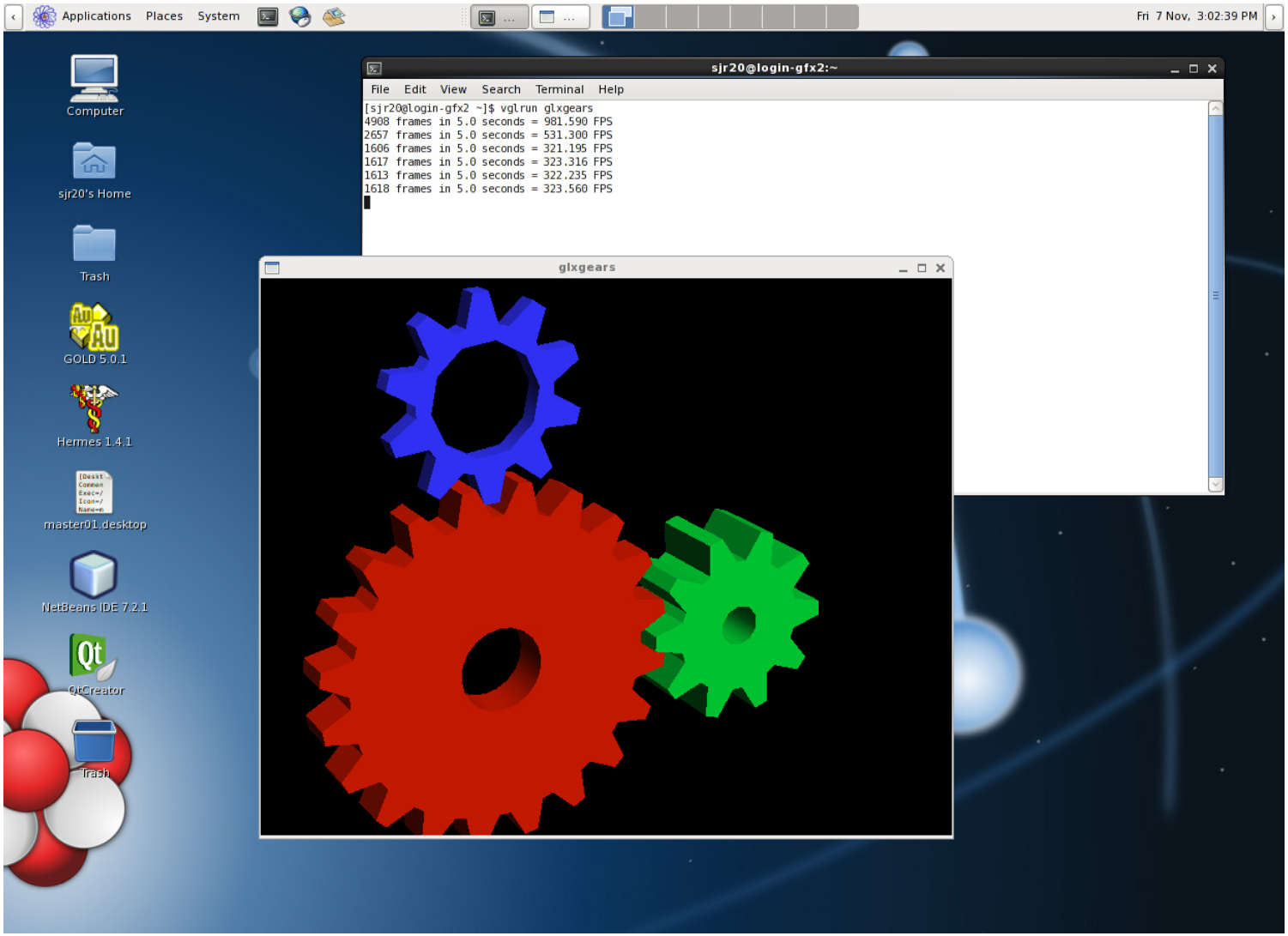
Connecting: Remote Desktop (MobaXterm)



Connecting: Remote Desktop (MobaXterm)



3D Remote Visualization



3D Remote Visualization

- Choose login-gfx2.
- Launch any application requiring 3D (OpenGL) with vglrun.
- May need to adjust the compression level for your network connection.

Using HPC: User Environment

- Scientific Linux 6.8 (Red Hat Enterprise Linux 6.8 rebuild)
  - bash
  - GNOME2 or XFCE4 desktop (if you want)
- Lustre 2.4.1 (patched), Mellanox OFED 3.3, CUDA 8.0
- But you don't need to know that.
- Upgrade to Scientific Linux/Red Hat Enterprise Linux 7 underway (Wilkes already upgraded).

Using HPC: User Environment

Red Hat Enterprise Linux 6

CUDA 8

- But you don't need to know that.
- Upgrade to Scientific Linux/Red Hat Enterprise Linux 7 underway (Wilkes already upgraded).

Using HPC: User Environment

Red Hat Enterprise Linux 6

CUDA 8

- But you don't need to know that. (Probably. . .)
- Upgrade to Scientific Linux/Red Hat Enterprise Linux 7 underway (Wilkes already upgraded).

Using HPC: User Environment

Red Hat Enterprise Linux 6

CUDA 8

- But you don't need to know that. (Probably...)
- Upgrade to Scientific Linux/Red Hat Enterprise Linux 7 underway (Wilkes already upgraded).

Using HPC: User Environment

Red Hat Enterprise Linux 6

CUDA 8

- But you don't need to know that. (Probably. . .)
- Upgrade to Scientific Linux/Red Hat Enterprise Linux 7 underway (Wilkes already upgraded).

User Environment: Filesystems

- /home/abc123
  - 40GB quota.
  - Visible equally from all nodes.
  - Single storage server.
  - Hourly, daily, weekly snapshots copied to tape.
  - Not intended for job outputs or large/many input files.
- /scratch/abc123
  - Visible equally from all nodes.
  - Larger and faster (1TB initial quota).
  - Intended for job inputs and outputs.
  - Not backed up.
  - Changes coming post-upgrade.
  - http://www.uis.cam.ac.uk/initiatives/storage-strategy/storage-services

User Environment: Filesystems

- /home/abc123
  - 40GB quota.
  - Visible equally from all nodes.
  - Single storage server.
  - Hourly, daily, weekly snapshots copied to tape.
  - Not intended for job outputs or large/many input files.
- /scratch/abc123
  - Visible equally from all nodes.
  - Larger and faster (1TB initial quota).
  - Intended for job inputs and outputs.
  - Not backed up.
  - Changes coming post-upgrade.
  - http://www.uis.cam.ac.uk/initiatives/storage-strategy/storage-services

User Environment: Filesystems

- /home/abc123
  - 40GB quota.
  - Visible equally from all nodes.
  - Single storage server.
  - Hourly, daily, weekly snapshots copied to tape.
  - Not intended for job outputs or large/many input files.
- /scratch/abc123
  - Visible equally from all nodes.
  - Larger and faster (1TB initial quota).
  - Intended for job inputs and outputs.
  - Not backed up.
  - Changes coming post-upgrade.
  - http://www.uis.cam.ac.uk/initiatives/storage-strategy/storage-services

Filesystems: Quotas

- quota

```
==========================================================================================
Usage on /scratch (lfs quota -u abc123 /scratch):
==========================================================================================
Disk quotas for user abc123 (uid 456):
      Filesystem  kbytes   quota   limit   grace   files   quota   limit   grace
 /scratch/abc123 9298708  1073741824 1181116006   -   165588       0       0       -
...
```

- Aim to stay below the soft limit (*quota*).
- Once over the soft limit, you have 7 days grace to return below.
- When the grace period expires, or you reach the hard limit (*limit*), no more data can be written.
- It is important to rectify an out of quota condition ASAP.

Filesystems: Quotas

- quota

```
==========================================================================================
Usage on /scratch (lfs quota -u abc123 /scratch):
==========================================================================================
Disk quotas for user abc123 (uid 456):
      Filesystem  kbytes   quota   limit   grace   files   quota   limit   grace
 /scratch/abc123 *1073742000  1073741824 1181116006   -   165588       0       0       -
...
```

- Aim to stay below the soft limit (*quota*).
- Once over the soft limit, you have 7 days grace to return below.
- When the grace period expires, or you reach the hard limit (*limit*), no more data can be written.
- It is important to rectify an out of quota condition ASAP.

Filesystems: Quotas

- quota

```
==========================================================================================
Usage on /scratch (lfs quota -u abc123 /scratch):
==========================================================================================
Disk quotas for user abc123 (uid 456):
      Filesystem  kbytes   quota   limit   grace   files   quota   limit   grace
 /scratch/abc123 *1073742000  1073741824 1181116006   -   165588       0       0       -
...
```

- Aim to stay below the soft limit (*quota*).
- Once over the soft limit, you have 7 days grace to return below.
- When the grace period expires, or you reach the hard limit (*limit*), no more data can be written.
- It is important to rectify an out of quota condition ASAP.

Filesystems: Quotas

- quota

```
==========================================================================================
Usage on /scratch (lfs quota -u abc123 /scratch):
==========================================================================================
Disk quotas for user abc123 (uid 456):
      Filesystem  kbytes   quota   limit   grace   files   quota   limit   grace
 /scratch/abc123 *1073742000  1073741824 1181116006   -   165588       0       0       -
...
```

- Aim to stay below the soft limit (*quota*).
- Once over the soft limit, you have 7 days grace to return below.
- When the grace period expires, or you reach the hard limit (*limit*), no more data can be written.
- It is important to rectify an out of quota condition ASAP.

Filesystems: Backups

- Disk snapshots and tape (as of May 2017).
- They are not an undelete - take care when deleting.
- Successful restoration depends on:
  - The file having existed long enough to have been backed up at all.
  - The last good version existing in a current backup.
  - Request restoration as soon as possible with *location* and *exact time of loss*.

Filesystems: Backups

- Disk snapshots and tape (as of May 2017).
- They are not an undelete - take care when deleting.
- Successful restoration depends on:
  - The file having existed long enough to have been backed up at all.
  - The last good version existing in a current backup.
  - Request restoration as soon as possible with *location* and *exact time of loss*.

Filesystems: Backups

- Disk snapshots and tape (as of May 2017).
- They are not an undelete - take care when deleting.
- Successful restoration depends on:
  - The file having existed long enough to have been backed up at all.
  - The last good version existing in a current backup.
  - Request restoration as soon as possible with *location* and *exact time of loss*.

Filesystems: Backups

- Disk snapshots and tape (as of May 2017).
- They are not an undelete - take care when deleting.
- Successful restoration depends on:
  - The file having existed long enough to have been backed up at all.
  - The last good version existing in a current backup.
  - Request restoration as soon as possible with *location* and *exact time of loss*.

Filesystems: Backups

- Disk snapshots and tape (as of May 2017).
- They are not an undelete - take care when deleting.
- Successful restoration depends on:
  - The file having existed long enough to have been backed up at all.
  - The last good version existing in a current backup.
  - Request restoration as soon as possible with *location* and *exact time of loss*.
  - Scratch files are not backed up.

Filesystems: Automounter

- Directories under /scratch are automounted.
- They only appear under /scratch when explicitly referenced.
- Thus when browsing /scratch may appear too empty
  — use *ls* or *cd* to reference /scratch/abc123 explicitly.

Filesystems: Permissions

- Be careful and if unsure, please ask support@hpc.cam.ac.uk.
  - Can lead to accidental destruction of your data or account compromise.
- Avoid changing the permissions on your home directory.
  - Files under /home are particularly security sensitive.
  - Easy to break passwordless communication between nodes.

Using HPC: Software

- Free software accompanying Red Hat Enterprise is (or can be) provided.
- Other software (free and non-free) is available via modules.
- Some proprietary software may not be generally accessible.
- See http://www.hpc.cam.ac.uk/using-clusters/software.
- New software may be possible to provide on request.
- Self-installed software must be properly licensed.
- *sudo will not work.* (You should be worried if it did.)

Using HPC: Software

- Free software accompanying Red Hat Enterprise is (or can be) provided.
- Other software (free and non-free) is available via modules.
- Some proprietary software may not be generally accessible.
- See http://www.hpc.cam.ac.uk/using-clusters/software.
- New software may be possible to provide on request.
- Self-installed software must be properly licensed.
- *sudo will not work.* (You should be worried if it did.)

User Environment: Environment Modules

- Modules load or unload additional software packages.
- Some are required and automatically loaded on login.
- Others are optional extras, or possible replacements for other modules.
- Beware unloading default modules in ~/.bashrc.
- Beware overwriting environment variables such as PATH and LD_LIBRARY_PATH in ~/.bashrc. If necessary append or prepend.

User Environment: Environment Modules

- Currently loaded:

```
module list
Currently Loaded Modulefiles:
  1) dot                       6) intel/impi/4.1.3.045   11) default-impi
  2) scheduler                 7) global
  3) java/jdk1.7.0_60          8) intel/cce/12.1.10.319
  4) turbovnc/1.1              9) intel/fce/12.1.10.319
  5) vgl/2.3.1/64             10) intel/mkl/10.3.10.319
```

- Available:

```
module av
```

User Environment: Environment Modules

- Show:

```
module show castep/impi/7.0.3
-------------------------------------------------------------------
/usr/local/Cluster-Config/modulefiles/castep/impi/7.0.3:

module-whatis    adds CASTEP 7.0.3 (Intel MPI) to your environment

Note that this software is restricted to registered users.

prepend-path     PATH /usr/local/Cluster-Apps/castep/impi/7.0.3/bin:/usr/local/...
-------------------------------------------------------------------
```

- Load:

```
module load castep/impi/7.0.3
```

- Unload:

```
module unload castep/impi/7.0.3
```

User Environment: Environment Modules

- Matlab

```
module load matlab/r2015b
```

- Invoking matlab in batch mode:
  matlab -nodisplay -nojvm -nosplash command
  where the file command.m contains your matlab code.
- The University site license contains the Parallel Computing Toolbox.

User Environment: Environment Modules

- Matlab

```
module load matlab/r2015b
```

- Invoking matlab in batch mode:

  matlab -nodisplay -nojvm -nosplash command

  where the file command.m contains your matlab code.
- The University site license contains the Parallel Computing Toolbox.

User Environment: Environment Modules

- Matlab

```
module load matlab/r2015b
```

- Invoking matlab in batch mode:
  matlab -nodisplay -nojvm -nosplash command
  where the file command.m contains your matlab code.
- The University site license contains the Parallel Computing Toolbox.

User Environment: Environment Modules

- Purge:

```
module purge
```

- Defaults:

```
module show default-impi
module unload default-impi
module load default-impi-LATEST
```

- Run time environment must match compile time environment.

User Environment: Compilers

Intel: icc, icpc, ifort (recommended)

```
icc -O3 -xHOST -ip code.c -o prog
mpicc -O3 -xHOST -ip mpi_code.c -o mpi_prog
```

GCC: gcc, g++, gfortran

```
gcc -O3 -mtune=native code.c -o prog
mpicc -cc=gcc -O3 -mtune=native mpi_code.c -o mpi_prog
```

PGI: pgcc, pgCC, pgf90

```
pgcc -O3 -tp=sandybridge code.c -o prog
mpicc -cc=pgcc -O3 -tp=sandybridge mpi_code.c -o mpi_prog
```

Exercise 4: Modules and Compilers

User Environment: Compilers

Intel: icc, icpc, ifort (recommended)

```
icc -O3 -xHOST -ip code.c -o prog
mpicc -O3 -xHOST -ip mpi_code.c -o mpi_prog
```

GCC: gcc, g++, gfortran

```
gcc -O3 -mtune=native code.c -o prog
mpicc -cc=gcc -O3 -mtune=native mpi_code.c -o mpi_prog
```

PGI: pgcc, pgCC, pgf90

```
pgcc -O3 -tp=sandybridge code.c -o prog
mpicc -cc=pgcc -O3 -tp=sandybridge mpi_code.c -o mpi_prog
```

Exercise 4: Modules and Compilers

Using HPC: Job Submission

Using HPC: Job Submission

- Compute resources are managed by a scheduler:
  SLURM/PBS/SGE/LSF/...
- Jobs are submitted to the scheduler
  — analogous to submitting jobs to a print queue
  — a file (*submission script*) is copied and queued for processing.

Using HPC: Job Submission

- Jobs are submitted from the login nodes
  — not themselves managed by the scheduler.
- Jobs may be either non-interactive (batch) or interactive.
- Batch jobs run a shell script on the first of a list of allocated nodes.
- Interactive jobs provide a command line on the first of a list of allocated nodes.

Using HPC: Job Submission

- Jobs are submitted from the login nodes
  — not themselves managed by the scheduler.
- Jobs may be either non-interactive (batch) or interactive.
- Batch jobs run a shell script on the first of a list of allocated nodes.
- Interactive jobs provide a command line on the first of a list of allocated nodes.

Using HPC: Job Submission

- The HPCS has used SLURM since February 2014.
- Jobs may use part or all of one or more nodes
  — the owner can specify `--exclusive` to force exclusive node access (automatic on Wilkes but may change post-upgrade).
- Template submission scripts are available under /usr/local/Cluster-Docs/SLURM.

Job Submission: Using SLURM

- Prepare a shell script and submit it to SLURM:

```
[abc123@login]$ sbatch slurm_submission_script
Submitted batch job 790299
```

Job Submission: Show Queue

- Submitted job scripts are copied and stored in a queue:

```
[abc123@login]$ squeue -u abc123
             JOBID PARTITION     NAME     USER  ST       TIME  NODES NODELIST(REASON)
            790299 sandybrid     Test3  abc123 PD       0:00      8 (Priority)
            790290 sandybrid     Test2  abc123  R   27:56:10      8 sand-6-[38-40],sand-7-[27-31]
```

Job Submission: Show Queue

- Submitted job scripts are copied and stored in a queue:

```
[abc123@login]$ squeue -u abc123
             JOBID PARTITION     NAME     USER  ST       TIME  NODES NODELIST(REASON)
            790299 sandybrid     Test3  abc123 PD       0:00      8 (Resources)
            790290 sandybrid     Test2  abc123  R   27:56:10      8 sand-6-[38-40],sand-7-[27-31]
```

Job Submission: Show Queue

- Submitted job scripts are copied and stored in a queue:

```
[abc123@login]$ squeue -u abc123
             JOBID PARTITION     NAME     USER  ST       TIME  NODES NODELIST(REASON)
            790299 sandybrid     Test3  abc123 PD       0:00      8 (AssocGrpCPUMinsLimit)
            790290 sandybrid     Test2  abc123  R   27:56:10      8 sand-6-[38-40],sand-7-[27-31]
```

Job Submission: Monitor Job

- Examine a particular job:

```
[abc123@login]$ scontrol show job=790299
```

Job Submission: Cancel Job

- Cancel a particular job:

```
[abc123@login]$ scancel 790299
```

Job Submission: Scripts

- SLURM

  See slurm_submit.darwin, slurm_submit.wilkes.

```
#!/bin/bash
#! Name of the job:
#SBATCH -J darwinjob
#! Which project should be charged:
#SBATCH -A CHANGEME
#! How many whole nodes should be allocated?
#SBATCH --nodes=1
#! How many tasks will there be in total? (<= nodes*16)
#SBATCH --ntasks=1
#! How much wallclock time will be required?
#SBATCH --time=02:00:00
#! Select partition:
#SBATCH -p sandybridge
...
```

- #SBATCH lines are *structured comments*

  — correspond to sbatch command line options.

- The above job will be given 1 cpu on 1 node for 2 hours (by default there is 1 task per node, and 1 cpu per task).

Job Submission: Scripts

- SLURM

  See slurm_submit.darwin, slurm_submit.wilkes.

```
#!/bin/bash
#! Name of the job:
#SBATCH -J darwinjob
#! Which project should be charged:
#SBATCH -A CHANGEME
#! How many whole nodes should be allocated?
#SBATCH --nodes=1
#! How many tasks will there be in total? (<= nodes*16)
#SBATCH --ntasks=1
#! How much wallclock time will be required?
#SBATCH --time=02:00:00
#! Select partition:
#SBATCH -p sandybridge
...
```

- #SBATCH lines are *structured comments*

  — correspond to sbatch command line options.

- The above job will be given 1 cpu on 1 node for 2 hours (by default there is 1 task per node, and 1 cpu per task).

Job Submission: Scripts

- SLURM

  See slurm_submit.darwin, slurm_submit.wilkes.

```
#!/bin/bash
#! Name of the job:
#SBATCH -J darwinjob
#! Which project should be charged:
#SBATCH -A CHANGEME
#! How many whole nodes should be allocated?
#SBATCH --nodes=1
#! How many tasks will there be in total? (<= nodes*16)
#SBATCH --ntasks=1
#! How much wallclock time will be required?
#SBATCH --time=02:00:00
#! Select partition:
#SBATCH -p sandybridge
...
```

- #SBATCH lines are *structured comments*

  — correspond to sbatch command line options.

- The above job will be given 1 cpu on 1 node for 2 hours (by default there is 1 task per node, and 1 cpu per task).

Job Submission: Scripts

- SLURM

  See slurm_submit.darwin, slurm_submit.wilkes.

```
#!/bin/bash
#! Name of the job:
#SBATCH -J darwinjob
#! Which project should be charged:
#SBATCH -A CHANGEME
#! How many whole nodes should be allocated?
#SBATCH --nodes=1
#! How many tasks will there be in total? (<= nodes*16)
#SBATCH --ntasks=16
#! How much wallclock time will be required?
#SBATCH --time=02:00:00
#! Select partition:
#SBATCH -p sandybridge
...
```

- #SBATCH lines are *structured comments*

  — correspond to sbatch command line options.

- The above job will be given 16 cpus on 1 node for 2 hours (by default there is 1 task per node, and 1 cpu per task).

Job Submission: Accounting Commands [HPCS]

- How many core hours available do I have?

```
mybalance

User            Usage |        Account     Usage | Account Limit Available (CPU hrs)
---------- --------- + ------------- --------- + ------------- ---------
abc123             18 |          STARS       171 |       100,000    99,829
abc123             18 |      STARS-SL2        35 |       101,000   100,965
abc123            925 |         BLACKH    10,634 |       166,667   156,033
```

- How many core hours does some other project or user have?

```
gbalance -p HALOS

User           Usage |   Account     Usage | Account Limit Available (CPU hrs)
---------- --------- + --------- --------- + ------------- ---------

pq345              0 |     HALOS   317,656 |       600,000   282,344
xyz10         11,880 |     HALOS   317,656 |       600,000   282,344

(Use -u for user.)
```

- List all jobs charged to a project/user between certain times:

```
gstatement -p HALOS  -u xyz10 -s "2014-01-01-00:00:00" -e "2014-01-20-00:00:00"
       JobID      User   Account  JobName  Partition                End      NCPUS CPUTimeRAW ExitCode      State
------------ --------- ---------- -------- ---------- ------------------- ---------- ---------- -------- ----------
14505             xyz10     halos       help sandybrid+ 2014-01-07T12:59:40         16         32      0:9  COMPLETED
14506             xyz10     halos       help sandybrid+ 2014-01-07T13:00:11         16         48      2:0     FAILED
...
```

Job Submission: Single Node Jobs

- Serial jobs requiring large memory, or OpenMP codes.

```
#!/bin/bash
...
#SBATCH --nodes=1
#SBATCH --ntasks=1
# Default is 1 task per node
#SBATCH --cpus-per-task=

#SBATCH --mem=3994
# Memory per node in MB - default is pro rata by cpu number
# Increasing --mem or --cpus-per-task implicitly increases the other
...
export OMP_NUM_THREADS=  # For OpenMP across  cores
$application $options
...
```

Job Submission: Single Node Jobs

- Serial jobs requiring large memory, or OpenMP codes.

```
#!/bin/bash
...
#SBATCH --nodes=1
#SBATCH --ntasks=1
# Default is 1 task per node
#SBATCH --cpus-per-task=

#SBATCH --mem=3994
# Memory per node in MB - default is pro rata by cpu number
# Increasing --mem or --cpus-per-task implicitly increases the other
...
export OMP_NUM_THREADS=  # For OpenMP across  cores
$application $options
...
```

Job Submission: Single Node Jobs

- Serial jobs requiring large memory, or OpenMP codes.

```
#!/bin/bash
...
#SBATCH --nodes=1
#SBATCH --ntasks=1
# Default is 1 task per node
#SBATCH --cpus-per-task=1
# Default is 1 cpu (core) per task
#SBATCH --mem=3994
# Memory per node in MB - default is pro rata by cpu number
# Increasing --mem or --cpus-per-task implicitly increases the other
...
export OMP_NUM_THREADS=  # For OpenMP across  cores
$application $options
...
```

Job Submission: Single Node Jobs

- Serial jobs requiring large memory, or OpenMP codes.

```
#!/bin/bash
...
#SBATCH --nodes=1
#SBATCH --ntasks=1
# Default is 1 task per node
#SBATCH --cpus-per-task=1
# Default is 1 cpu (core) per task
#SBATCH --mem=3994
# Memory per node in MB - default is pro rata by cpu number
# Increasing --mem or --cpus-per-task implicitly increases the other
...
export OMP_NUM_THREADS=  # For OpenMP across  cores
$application $options
...
```

Job Submission: Single Node Jobs

- Serial jobs requiring large memory, or OpenMP codes.

```
#!/bin/bash
...
#SBATCH --nodes=1
#SBATCH --ntasks=1
# Default is 1 task per node
#SBATCH --cpus-per-task=1
# Default is 1 cpu (core) per task
#SBATCH --mem=3994
# Memory per node in MB - default is pro rata by cpu number
# Increasing --mem or --cpus-per-task implicitly increases the other
...
export OMP_NUM_THREADS=  # For OpenMP across  cores
$application $options
...
```

Job Submission: Single Node Jobs

- Serial jobs requiring large memory, or OpenMP codes.

```
#!/bin/bash
...
#SBATCH --nodes=1
#SBATCH --ntasks=1
# Default is 1 task per node
#SBATCH --cpus-per-task=16 # Whole node

#SBATCH --mem=3994
# Memory per node in MB - default is pro rata by cpu number
# Increasing --mem or --cpus-per-task implicitly increases the other
...
export OMP_NUM_THREADS=16  # For OpenMP across 16 cores
$application $options
...
```

Job Submission: Single Node Jobs

- Serial jobs requiring large memory, or OpenMP codes.

```
#!/bin/bash
...
#SBATCH --nodes=1
#SBATCH --ntasks=1
# Default is 1 task per node
#SBATCH --cpus-per-task=8  # Half node

#SBATCH --mem=3994
# Memory per node in MB - default is pro rata by cpu number
# Increasing --mem or --cpus-per-task implicitly increases the other
...
export OMP_NUM_THREADS=8   # For OpenMP across 8 cores
$application $options
...
```

Job Submission: Single Node Jobs

- Serial jobs requiring large memory, or OpenMP codes.

```
#!/bin/bash
...
#SBATCH --nodes=1
#SBATCH --ntasks=1
# Default is 1 task per node
#SBATCH --cpus-per-task=16 # Whole node

#SBATCH --mem=3994
# Memory per node in MB - default is pro rata by cpu number
# Increasing --mem or --cpus-per-task implicitly increases the other
...
export OMP_NUM_THREADS=8   # For OpenMP across 8 cores (using all memory)
$application $options
...
```

Job Submission: MPI Jobs

- Parallel job across multiple nodes.

```
#!/bin/bash
...
#SBATCH --nodes=4
#SBATCH --ntasks=64      # i.e. 16x4 MPI tasks in total.
#SBATCH --cpus-per-task=2
...
mpirun -np 64 $application $options
...
```

- SLURM-aware MPI launches remote tasks via SLURM.
- The template script uses $SLURM_TASKS_PER_NODE to set PPN.

Job Submission: MPI Jobs

- Parallel job across multiple nodes.

```
#!/bin/bash
...
#SBATCH --nodes=4
#SBATCH --ntasks=32     # i.e.  8x4 MPI tasks in total.
#SBATCH --cpus-per-task=2
...
mpirun -ppn 8 -np 32 $application $options
...
```

- SLURM-aware MPI launches remote tasks via SLURM.
- The template script uses $SLURM_TASKS_PER_NODE to set PPN.

Job Submission: MPI Jobs

- Parallel job across multiple nodes.

```
#!/bin/bash
...
#SBATCH --nodes=4
#SBATCH --ntasks=32     # i.e.  8x4 MPI tasks in total.
#SBATCH --cpus-per-task=2
...
mpirun -ppn 8 -np 32 $application $options
...
```

- SLURM-aware MPI launches remote tasks via SLURM.
- The template script uses $SLURM_TASKS_PER_NODE to set PPN.

Job Submission: Hybrid Jobs

- Parallel jobs using both MPI and OpenMP.

```
#!/bin/bash
...
#SBATCH --nodes=4
#SBATCH --ntasks=32      # i.e.  8x4 MPI tasks in total.
#SBATCH --cpus-per-task=2
...
export OMP_NUM_THREADS=2   # i.e. 2 threads per MPI task.
mpirun -ppn 8 -np 32 $application $options
...
```

- This job uses 64 CPUs (each MPI task splits into 2 OpenMP threads).

Job Submission: Hybrid Jobs

- Parallel jobs using both MPI and OpenMP.

```
#!/bin/bash
...
#SBATCH --nodes=4
#SBATCH --ntasks=32      # i.e.  8x4 MPI tasks in total.
#SBATCH --cpus-per-task=2
...
export OMP_NUM_THREADS=2   # i.e. 2 threads per MPI task.
mpirun -ppn 8 -np 32 $application $options
...
```

- This job uses 64 CPUs (each MPI task splits into 2 OpenMP threads).

Job Submission: High Throughput Jobs

- Multiple serial jobs across multiple nodes.
- Use srun to launch tasks (job steps) within a job.

```
#!/bin/bash
...
#SBATCH --nodes=4
...
cd directory_for_job1
srun --exclusive -N 1 -n 1 $application $options_for_job1 > output 2> err &
cd directory_for_job2
srun --exclusive -N 1 -n 1 $application $options_for_job2 > output 2> err &
...
cd directory_for_job64
srun --exclusive -N 1 -n 1 $application $options_for_job64 > output 2> err &
wait
```

- Exercise 5 & 6 - Submitting Jobs.

Job Submission: High Throughput Jobs

- Multiple serial jobs across multiple nodes.
- Use srun to launch tasks (job steps) within a job.

```
#!/bin/bash
...
#SBATCH --nodes=4
...
cd directory_for_job1
srun --exclusive -N 1 -n 1 $application $options_for_job1 > output 2> err &
cd directory_for_job2
srun --exclusive -N 1 -n 1 $application $options_for_job2 > output 2> err &
...
cd directory_for_job64
srun --exclusive -N 1 -n 1 $application $options_for_job64 > output 2> err &
wait
```

- Exercise 5 & 6 - Submitting Jobs.

Job Submission: High Throughput Jobs

- Multiple serial jobs across multiple nodes.
- Use srun to launch tasks (job steps) within a job.

```
#!/bin/bash
...
#SBATCH --nodes=4
...
cd directory_for_job1
srun --exclusive -N 1 -n 1 $application $options_for_job1 > output 2> err &
cd directory_for_job2
srun --exclusive -N 1 -n 1 $application $options_for_job2 > output 2> err &
...
cd directory_for_job64
srun --exclusive -N 1 -n 1 $application $options_for_job64 > output 2> err &
wait
```

- Exercise 5 & 6 - Submitting Jobs.

Job Submission: High Throughput Jobs

- Multiple serial jobs across multiple nodes.
- Use srun to launch tasks (job steps) within a job.

```
#!/bin/bash
...
#SBATCH --nodes=4
...
cd directory_for_job1
srun --exclusive -N 1 -n 1 $application $options_for_job1 > output 2> err &
cd directory_for_job2
srun --exclusive -N 1 -n 1 $application $options_for_job2 > output 2> err &
...
cd directory_for_job64
srun --exclusive -N 1 -n 1 $application $options_for_job64 > output 2> err &
wait
```

- Exercise 5 & 6 - Submitting Jobs.

Job Submission: High Throughput Jobs

- Multiple serial jobs across multiple nodes.
- Use srun to launch tasks (job steps) within a job.

```
#!/bin/bash
...
#SBATCH --nodes=4
...
cd directory_for_job1
srun --exclusive -N 1 -n 1 $application $options_for_job1 > output 2> err &
cd directory_for_job2
srun --exclusive -N 1 -n 1 $application $options_for_job2 > output 2> err &
...
cd directory_for_job64
srun --exclusive -N 1 -n 1 $application $options_for_job64 > output 2> err &
wait
```

- Exercise 5 & 6 - Submitting Jobs.

Job Submission: High Throughput Jobs

- Multiple serial jobs across multiple nodes.
- Use srun to launch tasks (job steps) within a job.

```
#!/bin/bash
...
#SBATCH --nodes=4
...
cd directory_for_job1
srun --exclusive -N 1 -n 1 $application $options_for_job1 > output 2> err &
cd directory_for_job2
srun --exclusive -N 1 -n 1 $application $options_for_job2 > output 2> err &
...
cd directory_for_job64
srun --exclusive -N 1 -n 1 $application $options_for_job64 > output 2> err &
wait
```

- Exercise 5 & 6 - Submitting Jobs.

Job Submission: Interactive [HPCS]

- Compute nodes are accessible via SSH while you have a job running on them.
- Alternatively, submit an interactive job:

```
sintr -A MYPROJECT -N2 -n16 -t 2:0:0
```

- Within the window (screen session):
  - Launches a shell on the first node (when the job starts).
  - Graphical applications should display correctly.
  - Create new shells with ctrl-a c, navigate with ctrl-a n and ctrl-a p.
  - ssh or srun can be used to start processes on any nodes in the job.
  - SLURM-aware MPI will do this automatically.

Job Submission: Interactive [HPCS]

- Compute nodes are accessible via SSH while you have a job running on them.
- Alternatively, submit an interactive job:

```
sintr -A MYPROJECT -N2 -n16 -t 2:0:0
```

- Within the window (screen session):
  - Launches a shell on the first node (when the job starts).
  - Graphical applications should display correctly.
  - Create new shells with ctrl-a c, navigate with ctrl-a n and ctrl-a p.
  - ssh or srun can be used to start processes on any nodes in the job.
  - SLURM-aware MPI will do this automatically.

Job Submission: Interactive [HPCS]

- Compute nodes are accessible via SSH while you have a job running on them.
- Alternatively, submit an interactive job:

```
sintr -A MYPROJECT -N2 -n16 -t 2:0:0
```

- Within the window (screen session):
  - Launches a shell on the first node (when the job starts).
  - Graphical applications should display correctly.
  - Create new shells with ctrl-a c, navigate with ctrl-a n and ctrl-a p.
  - ssh or srun can be used to start processes on any nodes in the job.
  - SLURM-aware MPI will do this automatically.

Job Submission: Array Jobs

- This feature varies between versions.
- *http://slurm.schedmd.com/job_array.html*
- Used for submitting and managing large sets of similar jobs.
- Each job in the array has the same initial options.
- SLURM

```
[abc123@login]$ sbatch --array=1-7 -A STARS-SL2 submission_script
Submitted batch job 791609
```

```
[abc123@login-sand2]$ squeue -u abc123
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          791609_1 sandybrid      hpl    abc123  R       0:06      1 sand-6-32
          791609_3 sandybrid      hpl    abc123  R       0:06      1 sand-6-37
          791609_5 sandybrid      hpl    abc123  R       0:06      1 sand-6-59
          791609_7 sandybrid      hpl    abc123  R       0:06      1 sand-7-27
```

791609_1, 791609_3, 791609_5, 791609_7

i.e. ${SLURM_ARRAY_JOB_ID}_${SLURM_ARRAY_TASK_ID}

SLURM_ARRAY_JOB_ID = SLURM_JOBID for the first element.

Job Submission: Array Jobs

- This feature varies between versions.
- *http://slurm.schedmd.com/job_array.html*
- Used for submitting and managing large sets of similar jobs.
- Each job in the array has the same initial options.
- SLURM

```
[abc123@login]$ sbatch --array=1-7:2 -A STARS-SL2 submission_script
Submitted batch job 791609
```

```
[abc123@login-sand2]$ squeue -u abc123
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          791609_1 sandybrid      hpl    abc123  R       0:06      1 sand-6-32
          791609_3 sandybrid      hpl    abc123  R       0:06      1 sand-6-37
          791609_5 sandybrid      hpl    abc123  R       0:06      1 sand-6-59
          791609_7 sandybrid      hpl    abc123  R       0:06      1 sand-7-27
```

791609_1, 791609_3, 791609_5, 791609_7

i.e. ${SLURM_ARRAY_JOB_ID}_${SLURM_ARRAY_TASK_ID}

SLURM_ARRAY_JOB_ID = SLURM_JOBID for the first element.

Job Submission: Array Jobs

- This feature varies between versions.
- *http://slurm.schedmd.com/job_array.html*
- Used for submitting and managing large sets of similar jobs.
- Each job in the array has the same initial options.
- SLURM

```
[abc123@login]$ sbatch --array=1,3,5,7 -A STARS-SL2 submission_script
Submitted batch job 791609
```

```
[abc123@login-sand2]$ squeue -u abc123
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          791609_1 sandybrid      hpl    abc123  R       0:06      1 sand-6-32
          791609_3 sandybrid      hpl    abc123  R       0:06      1 sand-6-37
          791609_5 sandybrid      hpl    abc123  R       0:06      1 sand-6-59
          791609_7 sandybrid      hpl    abc123  R       0:06      1 sand-7-27
```

791609_1, 791609_3, 791609_5, 791609_7

i.e. ${SLURM_ARRAY_JOB_ID}_${SLURM_ARRAY_TASK_ID}

SLURM_ARRAY_JOB_ID = SLURM_JOBID for the first element.

Job Submission: Array Jobs

- This feature varies between versions.
- *http://slurm.schedmd.com/job_array.html*
- Used for submitting and managing large sets of similar jobs.
- Each job in the array has the same initial options.
- SLURM

```
[abc123@login]$ sbatch --array=1,3,5,7 -A STARS-SL2 submission_script
Submitted batch job 791609

[abc123@login-sand2]$ squeue -u abc123
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          791609_1 sandybrid      hpl    abc123  R       0:06      1 sand-6-32
          791609_3 sandybrid      hpl    abc123  R       0:06      1 sand-6-37
          791609_5 sandybrid      hpl    abc123  R       0:06      1 sand-6-59
          791609_7 sandybrid      hpl    abc123  R       0:06      1 sand-7-27
```

791609_1, 791609_3, 791609_5, 791609_7

i.e. ${SLURM_ARRAY_JOB_ID}_${SLURM_ARRAY_TASK_ID}

SLURM_ARRAY_JOB_ID = SLURM_JOBID for the first element.

Job Submission: Array Jobs

- This feature varies between versions.
- *http://slurm.schedmd.com/job_array.html*
- Used for submitting and managing large sets of similar jobs.
- Each job in the array has the same initial options.
- SLURM

```
[abc123@login]$ sbatch --array=1,3,5,7 -A STARS-SL2 submission_script
Submitted batch job 791609

[abc123@login-sand2]$ squeue -u abc123
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          791609_1 sandybrid      hpl    abc123  R       0:06      1 sand-6-32
          791609_3 sandybrid      hpl    abc123  R       0:06      1 sand-6-37
          791609_5 sandybrid      hpl    abc123  R       0:06      1 sand-6-59
          791609_7 sandybrid      hpl    abc123  R       0:06      1 sand-7-27
```

791609_1, 791609_3, 791609_5, 791609_7

i.e. ${SLURM_ARRAY_JOB_ID}_${SLURM_ARRAY_TASK_ID}

SLURM_ARRAY_JOB_ID = SLURM_JOBID for the first element.

Job Submission: Array Jobs

- This feature varies between versions.
- *http://slurm.schedmd.com/job_array.html*
- Used for submitting and managing large sets of similar jobs.
- Each job in the array has the same initial options.
- SLURM

```
[abc123@login]$ sbatch --array=1,3,5,7 -A STARS-SL2 submission_script
Submitted batch job 791609

[abc123@login-sand2]$ squeue -u abc123
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          791609_1 sandybrid      hpl    abc123  R       0:06      1 sand-6-32
          791609_3 sandybrid      hpl    abc123  R       0:06      1 sand-6-37
          791609_5 sandybrid      hpl    abc123  R       0:06      1 sand-6-59
          791609_7 sandybrid      hpl    abc123  R       0:06      1 sand-7-27
```

791609_1, 791609_3, 791609_5, 791609_7

i.e. ${SLURM_ARRAY_JOB_ID}_${SLURM_ARRAY_TASK_ID}

SLURM_ARRAY_JOB_ID = SLURM_JOBID for the first element.

Job Submission: Array Jobs

- This feature varies between versions.
- *http://slurm.schedmd.com/job_array.html*
- Used for submitting and managing large sets of similar jobs.
- Each job in the array has the same initial options.
- SLURM

```
[abc123@login]$ sbatch --array=1,3,5,7 -A STARS-SL2 submission_script
Submitted batch job 791609

[abc123@login-sand2]$ squeue -u abc123
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          791609_1 sandybrid      hpl    abc123  R       0:06      1 sand-6-32
          791609_3 sandybrid      hpl    abc123  R       0:06      1 sand-6-37
          791609_5 sandybrid      hpl    abc123  R       0:06      1 sand-6-59
          791609_7 sandybrid      hpl    abc123  R       0:06      1 sand-7-27
```

791609_1, 791609_3, 791609_5, 791609_7

i.e. ${SLURM_ARRAY_JOB_ID}_${SLURM_ARRAY_TASK_ID}

SLURM_ARRAY_JOB_ID = SLURM_JOBID for the first element.

Job Submission: Array Jobs (ctd)

- Updates can be applied to specific array elements using ${SLURM_ARRAY_JOB_ID}_${SLURM_ARRAY_TASK_ID}
- Alternatively operate on the entire array via ${SLURM_ARRAY_JOB_ID}.
- Some commands still require the SLURM_JOB_ID (sacct, sreport, sshare, sstat and a few others).

Scheduling

- SLURM scheduling is multifactor:
  - QoS — payer or non-payer?
  - Age — how long has the job waited?
    Don't cancel jobs that seem to wait too long.
  - Fair Share — how much recent usage?
    Payers with little recent usage receive boost.
  - sprio -j jobid
- Backfilling
  - Promote lower priority jobs into gaps left by higher priority jobs.
  - Demands that the higher priority jobs not be delayed.
  - Relies on reasonably accurate wall time requests for this to work.
  - Jobs of default length will not backfill readily.

Scheduling

- SLURM scheduling is multifactor:
  - QoS — payer or non-payer?
  - Age — how long has the job waited?
    Don't cancel jobs that seem to wait too long.
  - Fair Share — how much recent usage?
    Payers with little recent usage receive boost.
  - sprio -j jobid
- Backfilling
  - Promote lower priority jobs into gaps left by higher priority jobs.
  - Demands that the higher priority jobs not be delayed.
  - Relies on reasonably accurate wall time requests for this to work.
  - Jobs of default length will not backfill readily.

Scheduling

- SLURM scheduling is multifactor:
  - QoS — payer or non-payer?
  - Age — how long has the job waited?
    Don't cancel jobs that seem to wait too long.
  - Fair Share — how much recent usage?
    Payers with little recent usage receive boost.
  - sprio -j jobid
- Backfilling
  - Promote lower priority jobs into gaps left by higher priority jobs.
  - Demands that the higher priority jobs not be delayed.
  - Relies on reasonably accurate wall time requests for this to work.
  - Jobs of default length will not backfill readily.

Scheduling

- SLURM scheduling is multifactor:
  - QoS — payer or non-payer?
  - Age — how long has the job waited?
    Don't cancel jobs that seem to wait too long.
  - Fair Share — how much recent usage?
    Payers with little recent usage receive boost.
  - sprio -j jobid
- Backfilling
  - Promote lower priority jobs into gaps left by higher priority jobs.
  - Demands that the higher priority jobs not be delayed.
  - Relies on reasonably accurate wall time requests for this to work.
  - Jobs of default length will not backfill readily.

Scheduling

- SLURM scheduling is multifactor:
  - QoS — payer or non-payer?
  - Age — how long has the job waited?
    Don't cancel jobs that seem to wait too long.
  - Fair Share — how much recent usage?
    Payers with little recent usage receive boost.
  - sprio -j jobid
- Backfilling
  - Promote lower priority jobs into gaps left by higher priority jobs.
  - Demands that the higher priority jobs not be delayed.
  - Relies on reasonably accurate wall time requests for this to work.
  - Jobs of default length will not backfill readily.

Scheduling

- SLURM scheduling is multifactor:
  - QoS — payer or non-payer?
  - Age — how long has the job waited?
    Don't cancel jobs that seem to wait too long.
  - Fair Share — how much recent usage?
    Payers with little recent usage receive boost.
  - sprio -j jobid
- Backfilling
  - Promote lower priority jobs into gaps left by higher priority jobs.
  - Demands that the higher priority jobs not be delayed.
  - Relies on reasonably accurate wall time requests for this to work.
  - Jobs of default length will not backfill readily.

Wait Times

- The cluster is currently very busy and we do not preempt.
- 36 or 12 hour job walltimes are permitted.
- This sets the timescale at busy times (*without* backfilling).
- Use backfilling when possible.
- Short (1 hour or less) jobs have higher throughput (dedicated nodes).
- Each user can submit one very high priority job with --qos=INTR.
- We are upgrading significantly in 2017!

Wait Times

- The cluster is currently very busy and we do not preempt.
- 36 or 12 hour job walltimes are permitted.
- This sets the timescale at busy times (*without* backfilling).
- Use backfilling when possible.
- Short (1 hour or less) jobs have higher throughput (dedicated nodes).
- Each user can submit one very high priority job with --qos=INTR.
- We are upgrading significantly in 2017!

Wait Times

- The cluster is currently very busy and we do not preempt.
- 36 or 12 hour job walltimes are permitted.
- This sets the timescale at busy times (*without* backfilling).
- Use backfilling when possible.
- Short (1 hour or less) jobs have higher throughput (dedicated nodes).
- Each user can submit one very high priority job with --qos=INTR.
- We are upgrading significantly in 2017!

Wait Times

- The cluster is currently very busy and we do not preempt.
- 36 or 12 hour job walltimes are permitted.
- This sets the timescale at busy times (*without* backfilling).
- Use backfilling when possible.
- Short (1 hour or less) jobs have higher throughput (dedicated nodes).
- Each user can submit one very high priority job with --qos=INTR.
- We are upgrading significantly in 2017!

Checkpointing

- Insurance against failures during long jobs.
- Restart from checkpoints to work around finite job length.
- Application native methods are best. Failing that . . .
- Darwin & Wilkes nodes currently provide Berkeley Lab Checkpoint/Restart (BLCR)

  cr_run, cr_checkpoint, cr_restart
- sbatch --checkpoint=minutes
- scontrol checkpoint restart jobid

Checkpointing

- Insurance against failures during long jobs.
- Restart from checkpoints to work around finite job length.
- Application native methods are best. Failing that ...
- Darwin & Wilkes nodes currently provide Berkeley Lab Checkpoint/Restart (BLCR)

  cr_run, cr_checkpoint, cr_restart
- sbatch --checkpoint=minutes
- scontrol checkpoint restart jobid

Checkpointing

- Insurance against failures during long jobs.
- Restart from checkpoints to work around finite job length.
- Application native methods are best. Failing that ...
- Darwin & Wilkes nodes currently provide Berkeley Lab Checkpoint/Restart (BLCR)

  cr_run, cr_checkpoint, cr_restart
- sbatch --checkpoint=minutes
- scontrol checkpoint restart jobid

Job Submission: Scheduling Top Dos & Don'ts

- **Do . . .**
  - Give reasonably accurate wall times (allows backfilling).
  - Check your balance occasionally (mybalance).
  - Test on a small scale first.
  - Implement checkpointing if possible (reduces resource wastage).
- **Don't . . .**
  - Request more than you need
    — you will wait longer and use more credits.
  - Cancel jobs unnecessarily
    — priority increases over time.