POLAR SCIENCE
FOR PLANET EARTH

British Antarctic Survey

An Introduction to High Performance Computing 2022

Introduction

HPC Technical Staff

- Andrew England: HPC Service Owner
- Paul Sumption: HPC Engineer
- Jame Byrne: HPC Software / AI Lab

Your background

- Programmers (or not).
- Linux power users (or not).
- Researchers wishing to run large, parallel code.
- Researchers wishing to run many, non-parallel cases.
- Researchers interested in big data, machine learning, AI.
- Researchers requiring slightly more than an ordinary
- Workstation.

Many different disciplines and requirements!

Topics

- Part 1: HPC basic principals
  - What is HPC?
  - Types of problems
- Part 2: BAS HPC Resources
  - HPC Overview
  - Hardware
  - Storage
- Part 3: Access
  - Authentication
  - SSH / X2go
  - Data transfer

Topics

- Part 4: User Environment
  - Software
  - Modules
  - Shell
- Part 5: SLURM
  - What is a scheduler
  - Basic SLURM commands
- Part 6: Advanced
  - Containers
- Part 7: Final tips
  - Best practise
  - Help
  - Links

POLAR SCIENCE
FOR PLANET EARTH

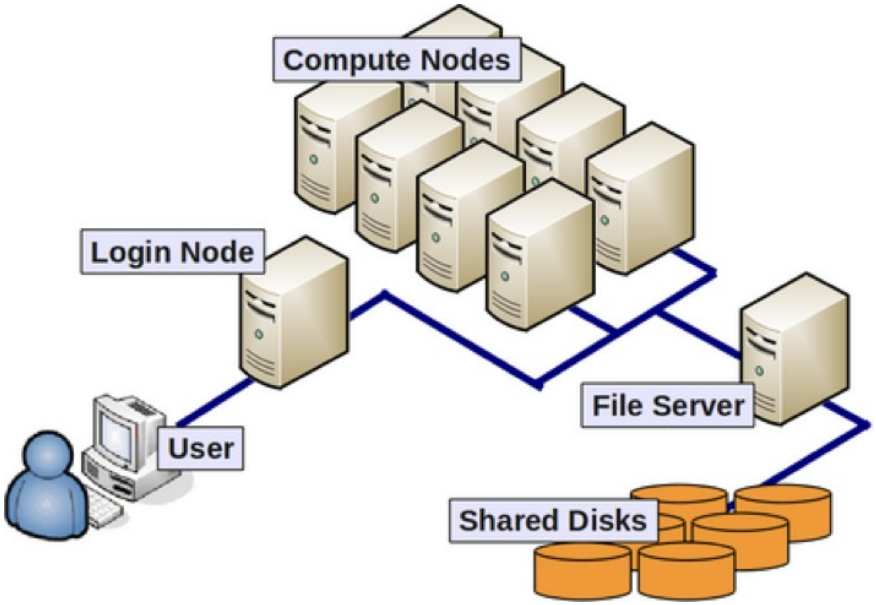
Part 1: HPC Basic Principals

Part 1: What is HPC?

HPC: High Performance Computing

- High powered workstation(s)
  - Lots of RAM
  - Lots of cores
  - High spec GPU cards for AI or 3D visualisation
  - Submit host for SLURM jobs
- A compute cluster
  - A large number of computers (worker nodes) usually commodity servers
  - A head node(s)
  - Configured to work as one system and share resources
  - A scheduler (SLURM) runs and prioritises jobs
  - Nodes can pass messages between each other (parallelisation)
- A National or TIER 1 centre
  - Dedicated data centres hosting large scale HPC clusters
  - Lots and lots of nodes!
  - Often used for scale out
  - JASMIN or ARCHER are facilities that you may have or need access to

Part 1: A simple cluster



- A cluster is a group of computers (worker nodes)
- A head node co-ordinates resources: RAM, storage, cores, software
- A login or submit node is used for sending jobs to the cluster
- Normally there is some shared storage (group drives / homes)
- Nodes might also have a small amount of local storage

Part 1: Why use HPC?

What types of big problem might require a “Big Computer”?

- 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

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

Part 1: Memory Intensive

- A single problem that can requires a large amount of RAM

- Memory requirements are greater than your laptop/desktop
- Often a large amount of data needs to load into RAM before processing
- Might not require multiple cores/CPU's
- Limited by maximum RAM in a workstation or node
- Might consume a whole cluster node

Part 1: Compute Intensive

- A single problem that can be split up 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
- This type of job is the traditional domain of HPC clusters

Part 1: Compute Intensive (Cont)

- Parallelisation requires effort:

- There are libraries to help (e.g. OpenMP, MPI)
  - OpenMP: parallelize in a shared memory system (whole node)
  - MPI: parallelize over a distributed system (multiple nodes)
  - MPI and OpenMP can be used at the same time in a hybrid job
- Aim to make the the amount of code that can be parallelised and performance per CPU as large as possible
- The scaling limit: eventually using more CPUs becomes detrimental instead of helpful.

Part 1: Data Intensive

- Distributing a large data set 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 the most important requirement
- Flash or large amounts of storage are often required

Part 1: High Throughput

- Distribute independent, multiple problems across many 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
- A good example is running lots of the same R or python job, you are probably high throughput, and possibly data intensive or compute intensive as well

Part 1: Summary

- Think about the type of problems you will need to solve
- Learn how to run single core jobs first
- If a job requires more cores than a single node you will need to look at panellisation
  - Open Source software will sometimes have an option to compile it to run in parallel (less work for you)
  - Otherwise you'll need to start to write your own code (more work)
  - TensorFlow and Python both have good MPI support

POLAR SCIENCE
FOR PLANET EARTH

Part 2: BAS HPC Facilities

Part 2: Hardware (workstations)

- Gateway (Jump or Bastion) hosts (bslcenb & bslcenc)
  - Only use for access to BAS or transferring files, don't use for running programs
- General Use Workstations & Private Workstations
  - 12 workstations
  - 2 with GPU cards
- Development Workstation
  - No access for normal users
  - Used by IT to test and build modules
  - Used by IT to test storage and other changes

Part 2: Hardware (Cluster)

- Headnode(s)
  - No direct access, manages job queues and storage (/data/hpcdata)
- CPU Nodes
  - 20 CPU compute nodes
  - 88 CPU cores+threads and 512GB system memory per node
- GPU Nodes
  - Node021: 24 CPU cores (no threads), 512GB system memory and 2 x NVIDIA T4 GPU's each with 16GB GPU memory
  - Node022: 48 CPU cores+threads, 512GB System memory and 8 x NVIDIA V100S GPU's each with 32GB GPU memory
    - Only available for use BAS AI Lab members
- Development Node
  - No access, used for testing by IT

Part 2: Storage (homes)

There are two home areas:

User Area - /users/<username>

- Small, not intended for sharing data
- Space restricted via quotas
- Not accessible from the HPC Nodes!

HPC Storage - /data/hpcdata/users/<username>

- Accessible from nodes and workstations
- Space restricted via quotas

Part 2: Storage (group drives)

SAN Volumes

- Setup for projects and departments, eg: /data/cruise, /data/vlf
- Accessible from workstations, bslcenb, bslcenc
- Volume should be managed and curated by a data manager
- Space is not controlled by quota's
- Adding additional space depends availability of physical disk space
- Contact the data manager first if you think you require additional storage

Part 2: Storage policies

Quotas

- On HPC you can check your quotas using: myquota
- Make sure to check your storage quotas regularly
- Jobs can fail if you are out of space
- If you need more space then request it via the service desk

Backups

- Daily at 6pm
- All SAN and HPC volumes backed up
- Backups are both onsite and offsite, via tapes disk
- If you need a file restored, contact the service desk

Part 2: Fast Storage

HPC Storage - /data/hpcflash

- Will be accessible from nodes and workstations, bslcenb, bslcenc
  - Currently only accessible from test nodes (work in progress)
- For high throughput GPU and AI jobs
- Access managed by the AI lab
- Suitable for workflow that has fast I/O requirements

POLAR SCIENCE
FOR PLANET EARTH

Part 3: Access

Part 3: Access

Authentication

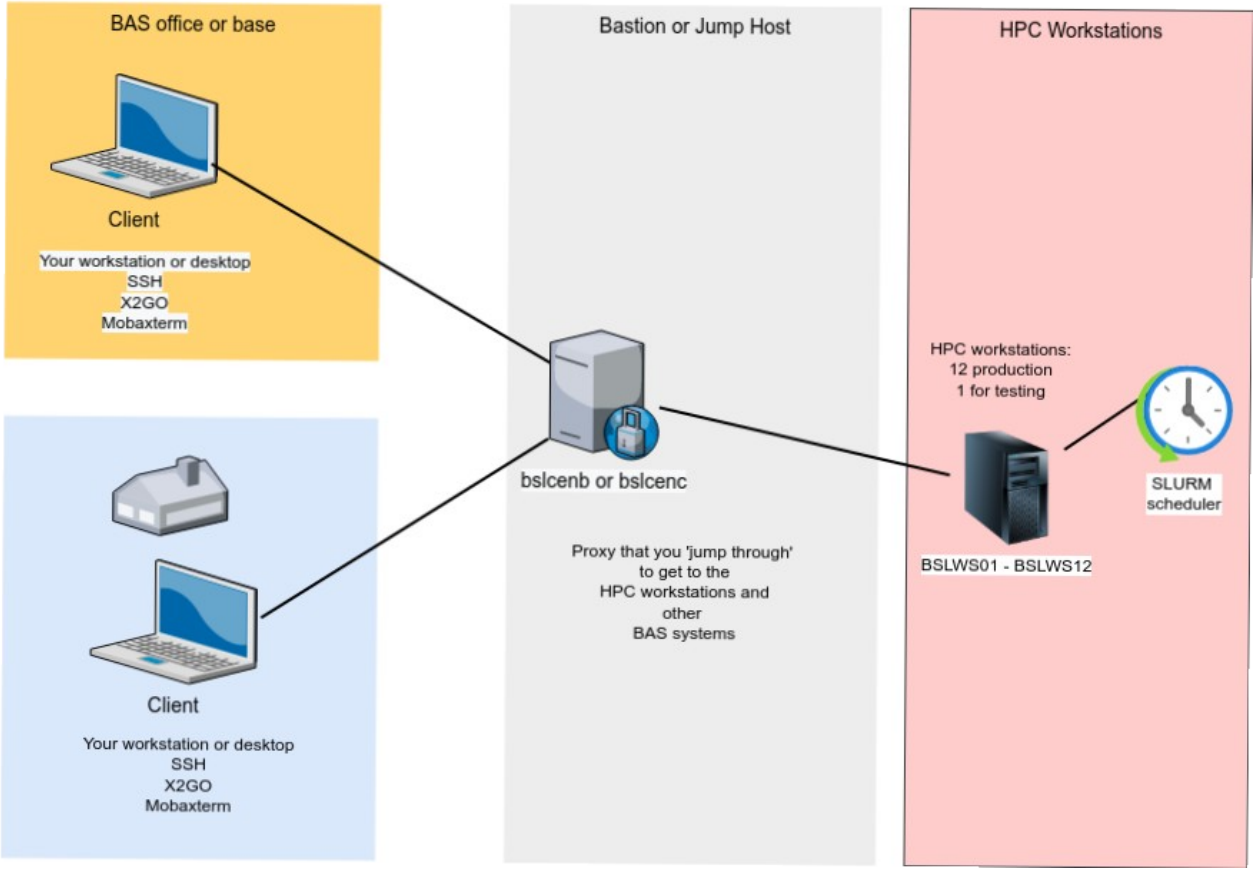
- Three passwords – UNIX (NIS), LDAP and Samba,
- UNIX for bslcenb / bslcenc and LDAP for HPC workstations
- Try to keep all these password synchronised
- We are working to simplify the situation

Please make sure you have had an IT Induction (or arrange one via the service desk)

You will need all of the above passwords to complete the exercises

Part 3: Jump or Bastion hosts

To access HPC service(s) you need to pass through a bastion host(s)



Part 3: SSH

The Secure Shell Protocol (SSH)

BSLCNENB, BSLCENC and the HPC workstations allow you to connect using SSH

- SSH is a cryptographic network protocol
- It allows us to send data securely over public networks (internet)
- Your data and commands are encrypted
- You can authenticate using a password (if the server allows this)
- You can authenticate using an SSH key
- Your key needs to be in a format that the remote host understands
- Older operating systems don’t always support the stronger key formats

SSH Keys

SSH client (your computer)

BAS servers and workstaions

1. Client sends its public key to the server when conecting.
2. Server checks the public key and that the client is in the authorized_hosts file
3. The clients private key is used to decrypt messages sent from the server

Private Key

Public Key

Part 3: SSH into BAS systems

SSH

- First connect to a gateway host(s): bslcenb.nerc-bas.ac.uk / bslcenc.nerc-bas.ac.uk
- Second connect to HPC workstations: bslws01...bslws12
- OpenSSH (available for Linux, Mac & windows), Putty, WSL, MobaXterm
- Linux and Mac users can modify the file *~/.ssh/config* then add aliases and jump hosts:

```
IdentityFile ~/.ssh/myusers_key



Host bslcenc
   Hostname bslcenc.nerc-bas.ac.uk
   User pasump
## cluster via jump host as pasump ###
Host bslws*
   Hostname %h
   User pasump
   ProxyCommand ssh -W %h:%p bslcenb
```

Demonstration of SSH

Generating an SSH key

To use an SSH key for accessing remote hosts:

- You will need to generate an SSH key pair
- You will need to transfer the public part of the key to the remote host you want to access:

i.e.ssh-copy-id -i ~/.ssh/mykey user@host

Linux and Mac users: Your operating system includes the commands: *ssh-keygen* and *ssh-copy-id*

Windows users: Putty can generate keys

Best practise:

- Make sure you keep the private part of the key safe!
- Set a passphrase on your key!
- Don’t store private keys in Onedrive or other cloud services
- Follow the Service desk solution on how to generate keys: https://servicedesk.bas.ac.uk/app/itdesk/ui/solutions/60624000019410097/details

SSH Agent

SSH agent is part of Open SSH

- ssh-agent is a key manager for SSH
- It holds your keys and certificates in memory, unencrypted, and ready for use by ssh
- It saves you from typing your passphrase every time you use your key
- You are only prompted to type the passphrase once (unless you remove the key from the agent)

Part 3: X2GO

X2GO allows you to access a desktop environment for interactive work

Part 3: Access (X2GO continued)

X2Go

- Access the HPC workstations desktop interface
- Works with or without VPN access
- Allows disconnecting and reconnecting to sessions
- Copy/paste (remote to local and vice versa)
- Sharing files from your laptop or PC
- More information: http://ictdocs/wiki/index.php/HPC:X2GO
- **Demonstration of X2GO**

Service Desk Solution:

Setting up X2go on Linux, Windows or Mac:
https://servicedesk.bas.ac.uk/app/itdesk/ui/solutions/60624000019605299/details

Part 3: Access (continued)

X2Go alternatives (for Windows users)

- putty, pscp, psftp: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
  - Good if you are mainly using SSH and terminals/command line
  - WinSCP: http://winscp.net/eng/download.php
- 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/
  - Probably the best alternative to X2GO

Part 3: Access (Mac / Linux)

- Both Mac (OS X) and Linux have:
  - Terminal applications installed as part of the OS
  - Common UNIX commands such as: *cp, mv, ls, ssh, scp, sftp, rsync*
- If you requite a GUI for using FTP:
  - GFTP: Installed from most Linux package managers
  - Cyberduck: for OS X https://cyberduck.io/
  - I would also recommend installing and learning a good text editor (useful when working on remote scripts or JASMIN/ARCHER)
    - Vi and vim are both powerful editors (but take some learning!)

Part 3: Data Access and Transfer

Samba

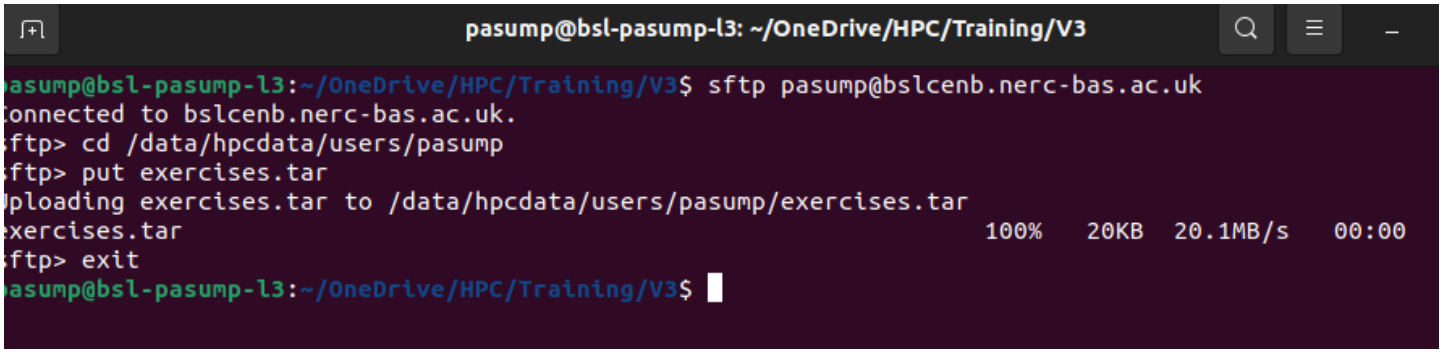
- Allows clients to connect to BAS UNIX storage servers as if they were a windows network share
- Allows access to SAN volumes, /users and /data/hpcdata
- There is no SAMBA access to /data/hpcflash
- It's advisable to set up SAMBA with the same password as that which you use for Windows
- There is a service desk solution for this:
  https://servicedesk.bas.ac.uk/app/itdesk/ui/solutions/60624000016015047/details

Part 3: Data Access and Transfer

SFTP

Secure File Transfer Protocol (FTP over SSH)

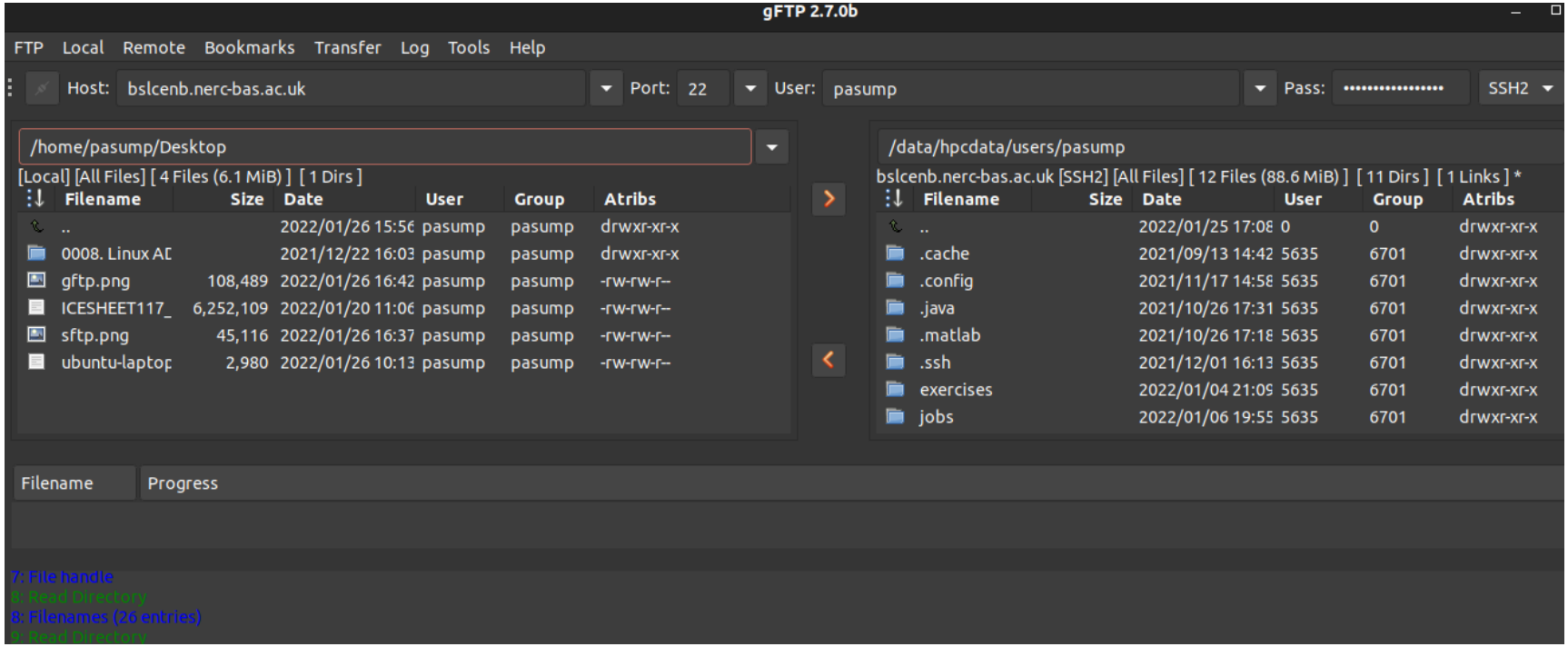
- Can be used with a GUI or command line
- The SAN volumes, */users* and */data/hpcdata* can be accessed from bslcenb.nerc-bas.ac.uk and bslcenc.nerc-bas.ac.uk
- You can use SFTP to connect to one of these hosts and transfer files in/out



Part 3: Data Access and Transfer

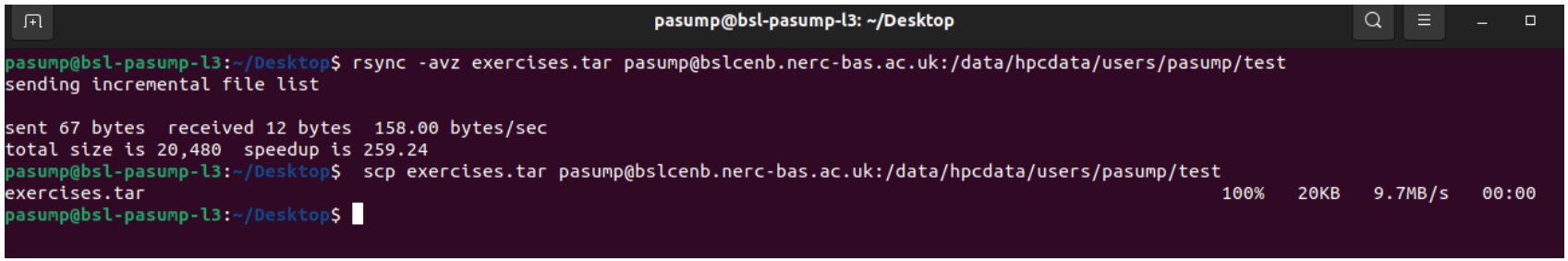
SFTP

File transfer using a graphical client: files can be dragged from one host to another.



Part 3: Data Access and Transfer (Cont.)

- SCP
  - Secure Copy, good for transferring single files
  - Works over SSH
- rsync
  - Perfect tool for transferring files locally and securely over the internet
  - Options to resume, reconnect, compress data and limit transfer rates
  - Works over SSH



Part 3: Data Access and Transfer (Cont.)

- sshfs
  - Allows a filesystem to be mounted over SSH
  - Doesn't deal with file locking!
  - Access can be quite slow
- Summary
  - However you transfer files you should use a secure protocol!
  - Be careful with the / in your rsync commands see: http://qdosmsq.dunbar-it.co.uk/blog/2013/02/rsync-to-slash-or-not-to-slash/

POLAR SCIENCE
FOR PLANET EARTH

Part 4: User Environment

Part 4: User Environment

- The Linux Shell
  - The shell is the Linux command line interpreter
  - It is the interface between the user and the operating system (kernel) and executes programs that we refer to as commands
  - Our default shell is tcsh
  - If you prefer something different such as bash, contact the service desk
  - To check which shell you are using type: *echo "$SHELL"*
  - If you raise an service desk ticket it is helpful to include details such as which shell you are using

Part 4: User Environment

Shell Scripts

A shell script is a file containing code that is interpreted by your shell (sometimes called a bash or tcsh script).

- Shell scripts are useful for cleaning up data and performing small functions
- It is worth investing some time in learning some basic shell scripting
- You can run a shell script you have written as part of your HPC job

Example:

A repeated tasks is to replace some text in the file ‘input.txt’. Rather than type the command each time we write a bash script: *replace.sh* to do this

```
#!/bin/bash
sed -i 's/old-text/new-text/g' input.txt
```

Part 4: User Environment

- Tmux
  - Keeps long running command line sessions running
  - Allows disconnecting and reconnecting
  - Multiple command line sessions and console splitting
  - More information: http://ictdocs/wiki/index.php/tmux

Part 4: Software

Operating System software

- The operating system install (nodes and workstations) is just the base packages
- This includes some graphical packages (accessible from the menus when connected via X2GO)
- Linux commands such as *cp, mv, scp, rsync etc*
- The software can be be run from the command line and desktop interface
- The core system is kept in a ‘vanilla’ state to stop packages that might conflict causing problems

Part 4: Software

Modules

The purpose of the modules system is to allow us to load a software that is not part of our base OS i.e. a different version of Python

- The modules system is not installed on and will not work on bslcenb or bslcenc
- There are two module repositories: /packages/modules & /hpcpackages/modules
- /hpcpackages/modules - works with nodes and workstations
- Modules sometimes include the compiler used in their name eg. hpc/netcdf/intel/4.4.1.1
- Works by adjusting shell variables eg. PATH, LD_LIBRARY_PATH
- Loaded modules only affect the terminal (or SLURM job) you load them in

Part 4: Software (continued)

Modules

- Useful module commands:

```
module avail                          module display name/version

module load name/version              module list

module unload name/version            module purge
```

- Common mistakes
  - Forgetting to use hpc modules on nodes
  - Mixing modules created using different compilers
  - Loading clashing modules
- More information: http://ictdocs/wiki/index.php?title=HPC:User_Guide
- **Demonstration of the module system**

Part 4: Jupyter Notebooks

- Jupter notebooks running on workstations: http://jupyterhub.nerc-bas.ac.uk
- More information: http://ictdocs/wiki/index.php/HPC:JupyterHub

POLAR SCIENCE
FOR PLANET EARTH

Part 5: SLURM

Part 5: SLURM

- What is it?
  - Simple Linux Utility for Resource Management:
    - It manages jobs submitted to the cluster
    - It determines which HPC resources you can access
  - It applies limits such as:
    - Maximum job run time (wall time)
    - How many cores you can use
    - How much RAM you can use
  - The scheduler looks at a job script and then prioritises them based on:
    - Resources required
    - Resources available
    - Backfill and fair share take care of slotting jobs into the queue
    - Cancelling and resubmitting jobs won't help move them through the queue!

Part 5: SLURM

Different queues

- Short queue (nodes 2 to 4)
    - Maximum run time of 30 minutes
    - Intended for short jobs and debugging which complete quickly
- Medium (nodes 12 to 17)
    - Maximum run time of 48 hours
    - Intended for large MPI jobs
- Long (nodes 17 to 20)
    - Maximum run time of 2 weeks
    - Intended for long running jobs
- GPU
    - node021: 2 x NVIDIA T4 GPU's each with 16GB GPU memory
    - Node022: 512GB System memory and 8 x NVIDIA V100S GPU's each with 32GB GPU memory

Part 5: Fair Share

Fair Share

- Ensures each user gets fair usage of each HPC queue
- A user that has run a lot jobs will get a lower priority than a user than has run none
- Adjusts priorities of submitted jobs based on previous usage
- Fair share is applied to all submitted jobs

The HPC user guide has some example job scenarios with explanations of how fair share would be applied

See: http://ictdocs.nerc-bas.ac.uk/wiki/index.php/HPC:User_Guide#SLURM_Fair_Usage_Scheduler

Part 5: SLURM

Job Types

Jobs may be either non-interactive (batch) or interactive:

- Batch jobs run a shell script on the first of a list of allocated nodes
  - The user submits a job script into a queue
  - The job runs when resources are available (user has no interaction)
- Interactive jobs provide a command line on the first of a list of allocated nodes.
  - The user requests a node or nodes: *sintr -A short -N1 -n8 -t 1:0:0*
  - You load modules and type commands on the allocated node(s)
  - Good if you have to write a complex SLURM job scripts (test on one node!)

Part 5: SLURM

Job Types (cont.)

- MPI
  - Good for compute intensive jobs with a related workflow
  - Jobs that require more cpu cores or RAM than a single node
  - MPI has a dedicated infiniband network messaging
  - All nodes need to be in the same queue.
- Array: http : //slurm.schedmd.com/job array .html
  - Used for submitting and managing large sets of similar jobs
  - Maximum of 1000 jobs
  - Each job in the array has the same initial options
- GPU - To use GPU's you must include the –gres option with the number of GPU's you require:

```
#SBATCH --gres=gpu:2
```

Part 5: SLURM

- Job submission scripts – example:

```
#!/bin/bash
#SBATCH –output=/data/hpcdata/users/pasump/jobs/matlab.%j.%N.out
#SBATCH –error=/data/hpcdata/users/pasump/jobs/matlab.%j.%N.err
#SBATCH -J matlab#SBATCH –mem=4gb
#SBATCH –time=00:10:00
#SBATCH –mail-type=begin,end,fail,requeue
#SBATCH --mail-user=pasump@bas.ac.uk
#SBATCH –partition=short
#SBATCH --account=short



source /etc/profile.d/modules.sh

module purge
module load hpc/matlab/R2021a

SCRIPTNAME=/data/hpcdata/users/pasump/jobs/matlabplot.m

matlab -nodesktop -nosplash -nodisplay -r "try run('${SCRIPTNAME}'); catch exit; end; exit"
```

Part 5: SLURM

- Job submission scripts – example:

- Each of the #SBATCH directives are interpreted by SLURM to setup your job

- In my example I set:

1. A job error and job ouput file. %j appends the job number to the files, %N the nodename.
2. -J sets my jobname, I will see this when I check the queue
3. --mem sets the amount of RAM I require per core
4. --time sets the maxmimum runtime of my job, the format is days, hours and minutes
5. --mail we set when SLURM should email us, in my example I ask to for an email when my job starts, ends, fails or gets requeued*
6. --mail-user this should be your @bas email address.
7. --partition My job only needs 10 mins so I an using the short run partition
8 --acount The account needs to have access to the partion, in our case the name matches the partition (short for short, medium or medium etc)
9. The source command sets up our module environment by reading a file that contains some settings
10. To be sure we are running in a clean environment we purge any loaded modules
11. We then add any module load commands for modules we want our job to load
12. The have user a variable SCRIPTNAME to set the full path to my matlab script
13. The final line is the program we wish to run and some options we wish to send to it such as the matlab scriptname an exit command.

*re-queing can cause problems such as data being overwritten.

Useful Options

Exclusive node: `#SBATCH -exclusive`
Specific node: `#SBATCH --nodelist=node022`

Part 5: SLURM (continued)

- Job monitoring
  - squeue -u <username>
  - sacct -j <jobid>
  - To see details on resources used by all running jobs: `scontrol show jobid -dd <jobid>`
  - To see all your recent jobs: `sacct -u <username>`
  - To check memory and cpu usage on a node: `scontrol show node <node>`

See the service desk solution on SLURM commands (queues and jobs)

https://servicedesk.bas.ac.uk/app/itdesk/ui/solutions/60624000019535035/details

Part 5: SLURM (continued)

- Troubleshooting failed jobs
  - Set an output file, often has useful information when jobs fail
- Common mistakes
  - Forgetting to load modules
  - Using storage which is not visible to the HPC nodes (use either /data/hpcdata or /data/hpcflash)
  - Avoid using symlinks
  - Did you load any require modules
  - Check your quota
- More information: http://ictdocs/wiki/index.php?title=HPC:User_Guide
- **Demonstration:** single core SLURM job

POLAR SCIENCE
FOR PLANET EARTH

Part 6: Advanced

Part 6: Containers

- Containers at BAS are still a work in progress
- What are containers?
- Podman
  - To be able to use, you need to contact the service desk
  - Container images must be downloaded to each node or workstation
- Singularity
  - Designed with HPC usage in mind
  - Ready to use on workstations and nodes
- For more information: http://ictdocs/wiki/index.php/HPC:Containers

POLAR SCIENCE
FOR PLANET EARTH

Part 7: Final tips

Developer Environment

- Use git!
  - If you are writing code it is a good idea to put it onto github
  - You can git pull the code onto remote machines (and reverse changes)
- Try to make your code:
  - Repeatable: can another lab member download and run your code?
  - Reproducible: could some else repeat your workflow?
  - Shareable: put your code in a group drive or repository such as github

If you need to move work between clusters containers can help. All of the required software is within the container.

Best Practice

- User Policy
  - Link:
- Do!
  - Test small jobs first (one core) before submitting a large array
  - Ask for help:
    - Preferably submit a help desk ticket
    - Give us much detail as possible
- Don't!
  - Submit jobs which use more than 4 nodes at a time
  - Cancel and reschedule jobs
  - Request more RAM/Cores than you need

HELP!

- Service desk: servicedesk@bas.ac.uk
  - Send in a ticket if you are having problems
  - Try to give some detail of the problem
  - For HPC jobs the name/location of your job script, what shell you are using, modules you are trying to use
- HPC User Guide: http://ictdocs/wiki/index.php?title=HPC:User_Guide
- Service Desk Solutions
  - Short solutions to common problems i.e. generating an SSH key
- Email List: bashpc@exchange.nerc.ac.uk
  - You get added to this list when we create your account

POLAR SCIENCE
FOR PLANET EARTH

Any Questions?