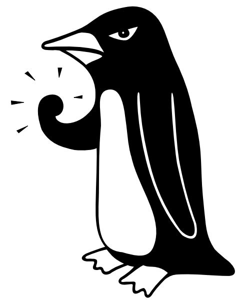
Big Computing



Warwick RSE

12/12/2022

About this Course

Big Computing

- This course is **not** aiming to have you walk out and be able to run your problem on a big compute systems
    - Too many different needs and approaches
- The aim of the course is that you should know what systems are available to you, what approach you might want to take and how to learn more about the approach that will work for you

Big Computing

- Day 1
  - Introduction and access to big computing
  - Software technologies in big computing
  - Cloud Computing and Containers
  - Details - Task based big computing

Big Computing

- Day 2
  - Challenges 1 - Working at scale
  - Challenges 2 - Robustness and Reproducibility
  - Challenges 3 - IO and Data processing
  - Where do I go next?

What is Big Computing?

Big Computing

- You might have heard of “Big Data”
  - Informal term meaning “data that is too large or complex to be processed by traditional means”
- Big Compute or Big Computing is the equivalent for calculations
  - Problems that are too numerous or too complex to be dealt with by traditional means
  - Can loosely be split into subgroups

High Performance Computing (HPC)

- This is the oldest of the types (but not the easiest!)
- Write computer programs to solve single large problems
- Programs are at least designed to use multiple processor cores of a single machine
- Often designed to run across multiple computers
- Sometimes use “accelerators” such as GPUs

High Throughput Computing (HTC)

- Rather than using many computers/processors to solve one large problem, use them to solve many small problems
    - Each problem is generally "small" although sometimes they can be small HPC jobs in their own right
- Related term Many Task Computing (MTC)
    - More of an idea of solving a single problem with multiple programs rather than solving multiple tasks

Do I want big computing?

- Major questions
  - Do I need to do an amount of work that will tie up my desktop or laptop for more than a couple of work days?
    - If so then probably yes
  - Do I need more memory than I have in my desktop or laptop
    - You can get 512GB to 2TB fairly easily but more will require special hardware or special programming

Do I want big computing?

- Major questions
  - Do I want or need specialised hardware?
    - Big systems often have specialist hardware and cloud systems have even more
  - Do I want or need specialised software?
    - Big computing systems often have preinstalled versions of major academic and scientific software

Do I NOT want big computing?

- Don’t be afraid of using big compute type resources
  - They aren’t that scary or hard to use
  - We’ll cover this later but mostly big computer systems are the same sort of thing as normal desktop PCs
  - They are enough extra work that if you can avoid them then it is probably easier to do so

Do I NOT want big computing?

- I want access to faster processors
  - Generally big computer systems have **more** processors that are **slower**, not fast processors
  - You can write software to use multiple processors on one task but that is hard
  - 4.5GHz gaming PC vs 2.4GHz processors in big computer systems

Do I NOT want big computing?

- Can I make my code faster on the computer that I have?
    - There is nothing wrong with using big computing rather than optimising your code massively
    - Developer time is important too!
    - But you may be able to improve the performance of your existing code so that you can avoid using big compute resources

Do I want big computing?

- Changes to algorithms
  - A different approach can massively improve performance
- Change to libraries
  - Switching to using a library from your own code or to using a high performance library can give substantial performance improvements
- Change to language
  - Some languages are faster than others!

What are we teaching?

- A bit of everything
    - What types of machines are available to you
    - Broadly how you get access to them
    - How you apply for access
    - Some general background on programming
        - There are general courses on parallel programming by us and other people as well
    - “Tips and Tricks” that are often not covered by normally big computing training courses
- Challenges and approaches to using big computing

Where do we run things?

Small

- Laptop or Desktop - Normal computer that you sit at and use
  - Typically 4 cores and 8 or 16GB of RAM
- Workstation - More powerful computer generally a supplement to a “normal” computer
  - Typically 8 or 16 (up to 128) cores and 128GB of RAM
  - Can sometimes be hard to justify to funding councils
- Advantage of small computers is that they are yours and you (can) have complete control

Medium

- Task Farm (Previously called the COW or Cluster of Workstations)
- Workstations rack mounted in a data centre
- Dedicated nodes for some research groups (some formally part of the task farm, some separate)
- Shared, managed resource
  - Often have to write a **submission script** to get your program running on the cluster
  - Some dedicated nodes you just log into directly

Large

- Clusters
  - Collections of powerful workstation class machines in a data centre
  - Specific hardware to connect the machines together
- Local clusters at Warwick
  - Avon
  - New cluster to be purchased

Very Large

- Tier 2 Clusters
  - Several large clusters around the country
    - Different clusters intended for different purposes and designed for those purposes
    - Also linked to a region where you can get access for any purpose
  - Tier 2 for the Midlands plus region that includes Warwick is Sulis which is based at Warwick
    - Sulis is aimed at high throughput and ensemble computing
  - You have to apply for time on these or larger machines

Very Large

- National (Tier 1) clusters
    - DiRAC - STFC
    - EPSRC and NERC - ARCHER2
    - Other UKRI funded research can get access to ARCHER2 as well by contacting the EPSRC research infrastructure team
- International (Tier 0) clusters
    - PRACE - European systems
    - INSITE - American systems - largest public machines in the world are currently here

This Course

- Not small - that's in the title!
- Not very large - You should probably be beyond this course already if you need them
- Concentrate on Medium and Large scale
- Systems at this scale are generally
    - Freely available (money)
    - Not freely available (time)
- You have a responsibility to use the resources wisely

Cloud

- Cloud systems are technically very much like clusters
  - You access them by paying rather than by application
    - Cost can be **very** high and hard to predict
    - Costs for unexpected things like copying data off the system
  - Some researchers will have access to cloud time as part of their project and some cloud providers have mechanisms for researchers to apply for time
- Unless your problem is wanting unusual hardware cloud doesn't solve any problems on it's own

Selecting scale

- As a general rule you'll probably be working in a field that already "does" large computing if you need access to tier 1 or tier 0

- Main questions are

  - How long does your work take?

  - How much can it be split up?

Selecting scale

- If your problem is very quick to run (all of it, not just one particular run of your program) then it probably isn't worth the effort of moving away from just running it on your desktop
- If **each** run runs within a few hours but you have a lot of them then you are in the regime of high throughput or ensemble computing
- If **each** run is slow then you are in the regime of parallel or high performance computing

High throughput computing

- Task farm is really well suited to HTC
    - If you want to run 10s of tasks then this is the place to go
- Sulis is specifically set up for HTC (although you can use it for other things)
    - If you have hundreds of tasks to run **or** each individual task is a parallel program go here
- There are no larger systems **intended** for HTC but you can sometimes run on other systems

High Performance Computing

- **NOTE** - It is not easy to write HPC code that works well - if you can avoid it **do**
- All modern computer systems from the smallest to the largest can run parallel codes
- At Warwick initially consider Avon (and eventually the new cluster), then Sulis (which is **designed** for HTC but works for HPC) for larger jobs
- After that look at DiRAC or ARCHER2 (or others where provided) based on your research council

Applying for Access

Documentation

- **READ THE DOCUMENTATION**
- This is the most important rule for any shared system
- It will say how to sign up, how to access your chosen machine, how to use the software, how to report problems etc.
- It might not always be good (we think that ours is pretty good!) but it will always have the basics
- Always check here before you ask for help

Resources at Warwick

- The task farm and cluster (Avon) at Warwick are generally available resources available through the Scientific Computing Research Technology Platform (warwick.ac.uk/scrtp)
- Sign up first for an SCRTP desktop account https://warwick.ac.uk/research/rtp/sc/desktop/gettingstarted/
- That's it for the task farm! Follow the instructions to use it
- For Avon (or the new cluster when it arrives) need a separate account https://warwick.ac.uk/research/rtp/sc/hpc/register/ and then follow the instructions again

Other resources

- Access to most other systems is by application
  - Sometimes time on systems can be provided with grants but we will ignore this here
  - There can be a lead time of months for the application so don't try and do this at the last minute

Other resources

- For Sulis you can apply as a Warwick researcher by following the guidance at https://warwick.ac.uk/research/rtp/sc/hpc/hpcmidplus/
    - Note especially that only one application per research group is permitted so check if your group is already using Sulis and get onto their application and sort out everyone who wants to use it and make one application
- You can ask for small allocations on Sulis for testing proof of principle by applying on Bugzilla or by email to sulis@warwick.ac.uk

Other resources

- For larger systems (or tier 2 systems that aren't Sulis) you are generally applying to research councils to do work within their remits
- Rules for the Tier-2 systems or ARCHER2 are available at https://www.archer2.ac.uk/support-access/access.html
  - Mainly EPSRC and NERC but access is available for other researchers too
- STFC DiRAC https://dirac.ac.uk/callforproposals/
- If calls are not open you have to wait until one is (basically)

How to apply

- Application for time on these systems is primarily on the basis of academic justification
  - You have to show the work that you are going to do from the computer time that you are asking for
  - Much like applying for grants so follow general guidance there
- Technical assessments are less important and mostly are about feasibility
  - Still best if you use resources efficiently!

How do we run things?

Queuing Systems

- Pretty much all larger computer systems have some kind of scheduling system
  - On cloud systems it is often combined with the charging system
- You need to have some kind of system to give people resources to use and deal with oversubscription of resources
- Queuing systems

Queuing Systems

- All systems have some mechanism of running "interactively"
  - You can issue commands to the computer immediately
- Generally your life is easiest if you run your problem as a batch job
  - Write your problem(s) to run unattended
  - Also has real advantages for reproducible research

Queuing Systems

- By far the most popular queuing system (at the moment) is called SLURM
  - Originally stood for Simple Linux Utility for Resource Management but now just accepts that they wanted to name their tool after something from Futurama so is just a name
- We will describe SLURM because it is very common and most other systems work in the same general way
- What we are describing is common and the way we do it but there can be slight variations between machines so read the documentation for the machine you are using

Core Element

- The core of a job script is a simple shell script
    - The more you know of Linux bash scripting the more you can do but you only **need** the very basics
- Each line in the file is run one after the other and your job finishes when the last line completes
- You have to do a bit more if your program has to be spun up across multiple computers

Simplest bash script

```
#!/usr/bin/env bash
./service1&
./myprogram
```

Resource Requests

- Once you have a bash script that can run your program you want to add elements requesting the resources that your program wants
  - Technically you can submit just the simple script and supply these commands manually when you submit the script but this generally isn't done
- All of these extra elements are special comments in the script that SLURM knows how to read

Resource Requests

- All of the special commands start with
- #SBATCH
  - This just stands for SLURM batch commands
- You can combine on a single line but normally one SBATCH line per specifier
- After which you put the request for the resource. There are a **lot** of possibilities (https://slurm.schedmd.com/sbatch.html) But typically they might come from the following list

Resource Requests

- --nodes - Number of individual computers to request on the cluster
- --ntasks - The number of individual tasks that **can** be spun up by SLURM
    - We'll come back to this but you can use **srun** to spin up individual tasks
- --cpus-per-task - Number of CPUs that will be set aside for each task

Resource Requests

- You may also be required to specify an account when running on a cluster, especially if you can have different accounts for different projects

- If you want specialised resources like GPUs, accelerators etc. you might have to put in a request for these

- If you are running on a machine that has internal partitions you might have to put in a partition name

    - Partitions are used for things like development queues, different types of hardware etc.

Simple SLURM script

```
#!/usr/bin/env bash

#SBATCH --nodes=1
#SBATCH --ntasks= 1
#SBATCH –cpus-per-task=128

./myprogram
```

- This script requests 1 node, with 1 task and 128 CPUs for that task
- This will work on Sulis to run one program and giving it 128 CPUs

srun

- While the examples above don't use it, you should run programs in Slurm scripts through srun
- It makes it easier for Slurm to check on your program and if your program uses MPI it automatically tells it how many processors to run on etc.
- **srun my_program**
- You can also use it to run individual tasks by passing extra parameters to **srun**
  - Usually better to use a secondary program to do this than roll your own

sbatch

- Once you have your script you can submit it to the queuing system
- This is done using the command **sbatch**
    - **sbatch {scriptname}**
    - You get back a **job number** that you can use to look up the state of the job or cancel it
- Important to note that a **copy** of your script is taken as soon as you submit the job - you can't change it without **scancel**-ing the job and submitting another
- Copies are **not** taken of your actual program though - if you want to queue different versions they have to be different files

squeue

- You can check the state of jobs using **squeue**
- Commonly you will either use **squeue {jobid}** to look up the state of a given job
- You will also often use **squeue -u {username}** to look at the status of all of your jobs

That's all folks!

- At this point you can choose a system and run your jobs on it
- The rest of this course is about meeting your requirement to be responsible with the resources
  - Make efficient use of the resources
  - Do good work
  - Be able to reproduce your work if needed
  - How to deal with things not working
  - How to good relationship with the support staff