Answers to Consolidation Questions

These answers are correct, but often not the only correct answer. Think of them as hints as much as answers!

01 Background

- What is HPC?
  - *High Performance Computing. Anything that needs dedicated hardware, or uses so much compute that your local machine is slowed down for perhaps an hour and up.*
- What are the most important things to remember when using computing for research?
  - *Robustness, reproducibility, conscientious documentation. Being happy with the quality and provenance of your code and data. Being aware of your skills and knowing when you might reach your limits and risk missing mistakes in your work.*
- What is an OS? What are you running? What processor do you have?
  - *OS means Operating System, and is the system that runs the basic facilities of your computer. You are likely to be using a Linux variant, such as Ubuntu, Debian or Centos. At home you may have Windows, or you may have MacOS. Your processor may be many things, but is likely to have 2 or more cores. You may have an Intel or AMD one, or (less likely) another brand.*

02 Notebook to Script

- What are notebooks? How are they different from scripts?
  - *Notebooks are combined code and description, images etc, arranged like a labbook. Notebooks are basically scripts underneath.*
- Can notebooks have any unwelcome consequences?

- *Yes, definitely! All code can. For Notebooks or Interactive sessions, there is a risk of running things out of order or losing track of parts of your state, as well as all of the risks of unexpected behaviour etc that scripts have.*

- Can you do anything in a notebook that you can't do in a script? What about vice versa?
  - *Yes, and No. You can more easily have a nice labbook layout in a notebook, and it is a bit easier to use a script as a library, but ultimately they are not very different. You can use functions in both, you can generate images etc.*
- What changes might you make when moving from a notebook to a script?
  - *Often you will want to move user-interaction code into a single place to make this easier. You might also want to create image files rather than just display them, so that you can keep them for later. You might want more functions (although that is also a good way to write your notebooks). You might want a script that can be imported like a library.*
- How can you keep a library of code for tasks you do often?
  - *Write small script files containing functions with related purposes, with a sensible name. Document them well. Try to write more general things, instead of having one function for every single purpose - use parameters etc instead. But don't go mad - don't try to write a single function that does everything or something silly like that!*
- What habits do you have that might lead to buggy or poorly maintainable code? How can you remedy them?
  - *I can't really answer this for you - but any of the things we mentioned in "writing better code" might improve your code. Perhaps the strongest answer to this is "anything you feel the need to excuse to other people" is probably a bad habit, and you can certainly improve their understanding of your stuff by doing things better there.*

- 03 Clusters, Queues and Modules

- Which of the following are untrue? Which might get you a very upset email from a system administrator? *I have Bolded the Untrue and given an explanation*

- **You're allowed to do anything on a cluster - if you shouldn't do it, it wont be possible**
    - ***NO! You wouldn't be able to do a lot of useful things if we tried to work this way. You should always be aware of what is OK and what isn't. But don't panic! Just be careful!***
- **Clusters are great for all your small, short jobs so your laptop doesn't get warm**
    - ***NO! Clusters are usually busy and aren't a place to dump all of your tiny scripts! Use them wisely! If your laptop is struggling to keep up, that is a good use. But always try to suit your jobs to your resources, not vice-versa***
- **Documentation is mostly pointless and you don't need to read it**
    - ***NO! Nobody likes writing documentation, so if they wrote it, it needs to be written! Read the docs and be aware of what you are allowed to do and how you are supposed to work!***
- Running on a cluster is simple once you know how
    - *Yes, this one is true. It's not always easy, but it's simple if you follow instructions and work carefully.*
- **Queues are designed to slow down your workflow and annoy you**
    - ***NO! It feels like this sometimes, but queues are designed to get the most out of a cluster without having a free-for-all on resource usage! It might be annoying having to adapt your workflow to a system, but trust us, it is the better alternative.***
- **There's no such thing as fair sharing of resources - you should just get to use as much as you want**
    - ***NO! Again, this often feels unfair, but mostly fair sharing is trying to be fair. But do check if the way you are doing your jobs is disadvantaging them - for instance you can usually get smaller allocations faster, so consider asking for more smaller jobs if you can for instance***
- **Special resource nodes are a great way to jump the queue**

    - ***NO! Special resource nodes such as GPU nodes or High-memory nodes are NOT to be used to try and skip queues on normal resources. Doing this is really going to upset people.***
  - If you can arrange your work to run when systems aren't busy, or exploit underused resources, you can get resources faster
    - *This is true. If you really can make good use of a GPU, for instance, those can be less busy. And often there are busy times of year (just before holidays, before major conferences etc), so if you can do more work at other times, you might make your life easier*
- What sorts of parallelism are there? How can you run a program across multiple compute nodes (machines)? How can you run a bunch of copies of a code with different inputs?
  - *Shared and distributed memory is the main one. Also, internal paralellism within a code, and running multiple independent tasks at once. Distributed memory is needed to run across nodes. Running a bunch of copies can be done using some task systems, such as GNU parallel or a job scheduler*
- What should you do if you need a new module installed for your work?
  - *You should know this, because you read the docs! Usually there is a place to request modules etc. What you mostly shouldn't do is email-bomb all the people you can think of demanding action now.*
- What queueing systems are in use on systems you might use? Can you use the same scripts between them (if more than one)?
  - *Slurm is a popular one. PBS is another and there are more.*
- How can you keep track of when your jobs run and whether they succeed or fail?
  - *Many schedulers can be set to email you with completion and failure messages. The docs should say.*
- (Humour) What might be a sys-admin's favourite email subject/bug title?
  - *Based on my experience, it is either "My code doesn't work" (with no details or even a hint as to which code that might be) or "Need help URGENTLY NOW" (usually at 5pm on a Friday for a problem that has existed for weeks)*

- 04 - Skills - Checkpoints

  - What things are important to keep in mind when checkpointing code?

    - *What data do you need to preserve? Are there any "hidden" things you might not have thought of? Can you reduce the amount of data? Is checkpointing worth it, or might it cost so much in time or storage that it wont improve your ability to run code?*

  - Complete this - if you can't continue from a checkpoint it is .....

    - *Many answers to this, but for instance, "redundant" "broken" or "useless". A checkpoint is only useful if it can be used, just like a backup is only a backup if it can be restored*

  - What sorts of problems might need "exact" continuation? What sorts only need statistical correctness?

    - *This has many answers. Many problems have already got some degree of statistical randomness in them and these can often be statistically continued. It really depends on the problem. Consider whether it matters to get the same result twice, or whether all answers are "right enough"*

  - How can you automate checkpoint-and-continue workflows using the job scheduler? Is this a fair workflow?

    - *Usually there is a way to make one job depend on another, so it runs only when the former finishes. This is (usually) a fine way to work, and is even recommended.*

- 05 - Skills - Containers

  - What are containers for? Why are they worthwhile?

    - *Packaging software to transfer between machines. Allowing absolute, exact reproducibility (although this is rarely needed). Easing difficult builds with a lot of dependencies, or at least easing the load on end users.*

  - Where can you find pre-built containers?

- *The Singularity Container Library, or DockerHub*

- What is provisioning?
  - *Putting the necessary programs etc into a container, so that it can do its job.*

- 06 - Skills - Numba

  - What is compiling for? What does JIT mean?
    - *Compiling is the converting of code from source text to machine-runnable instructions. JIT means "Just-In-Time" and refers to compiling as code runs*
  - What can the jit do for your code?
    - *In Numba, the jit can produce faster code. It can optimise some things, sometimes. It can also cache the compiled versions for future use. For your specific code, there may be many benefits*
  - What tricks are there for paralellism in numba?
    - *Parallelising loops is quite easy. You can also use some things like stencil for some array operations. Some functions in numba and other libraries can take advantage of using multiplt threads if you allow them too also.*

- 07 - Skills - High Performance Libraries

  - What reasons are there to use libraries?
    - *To perform tasks that you could't otherwise - to do them faster - to avoid re-inventing the wheel each time a task comes up - to do a more general task than you would be able or have time to write*
  - What problems can they cause?
    - *Dependencies make building code more involved. If a library changes you may need to track a specific version. You can't control fixes, just have to wait for the developer to do it. Popular libraries are well tested, but some aren't and you can miss important bugs, especially if you couldn't have written the code yourself*
  - How can you make your life easier if you have to change out a library for a different one in future?

- *Carefully using "shim" or "wrapper" functions so that the library itself only appears in a small number of places, limiting the number of things you have to change. To swap out a library, you change the inside of the wrapper only*

- 08 - Skills - GPUs

  - What is a GPU? Why are they good for parallel computing?
    - *The graphics hardware used orignally for rendering images. These have a lot of relatively weak cores, with little coupling between them. They are especially good for matrix problems which are a bit like images, after all.*
  - What sort of problems are suited for GPUs?
    - *Problems with a lot of independent tasks to run, such as applying a transform to every pixel of an image. Some GPU hardware is dedicated to more specific tasks, such as the Tensor cores in some modern GPUs*
  - Are there ways to seamlessly "roll-over" between using a GPU and not using one?
    - *Sometimes, especially if you can use a library. Otherwise, maybe*
  - Suppose I have a lot of passes to do over a small amount of data. What is likely to be the bottleneck in running this on a GPU?
    - *Getting the data on and off, assuming I need to look at it, or do something off the GPU in between passes. Although this question is very vague and doesn't have a real strict answer*