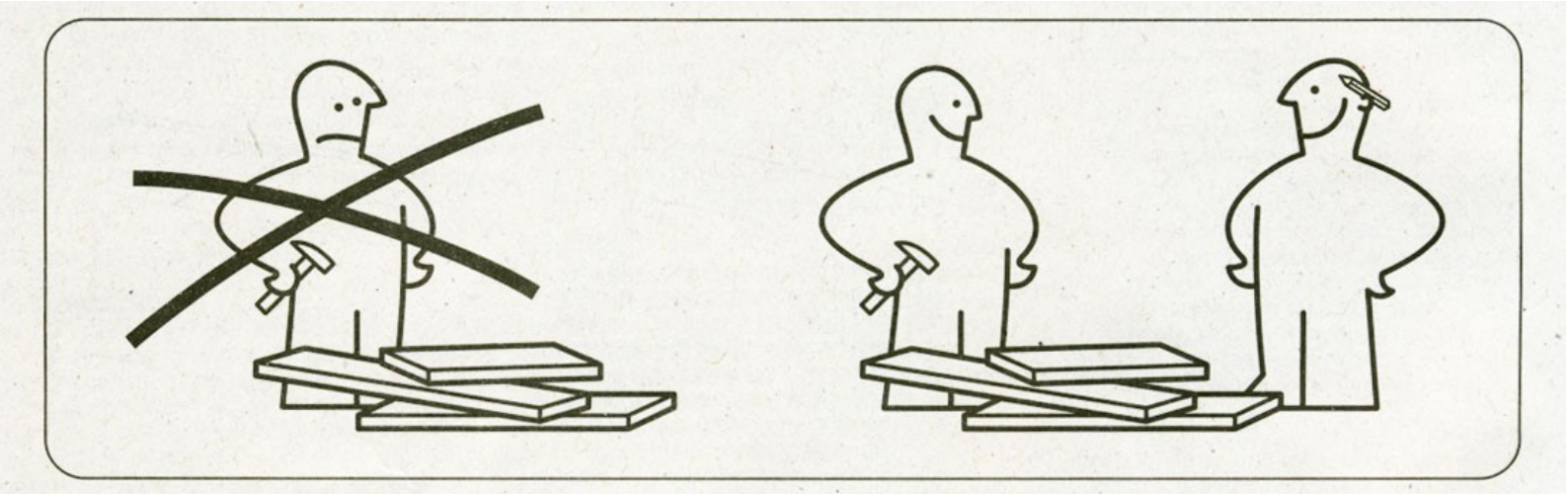
Ten* Tips for using HPC

Julien Sindt

EPCC, The University of Edinburgh

J.Sindt@ed.ac.uk

1. Don’t struggle alone



Service Desks

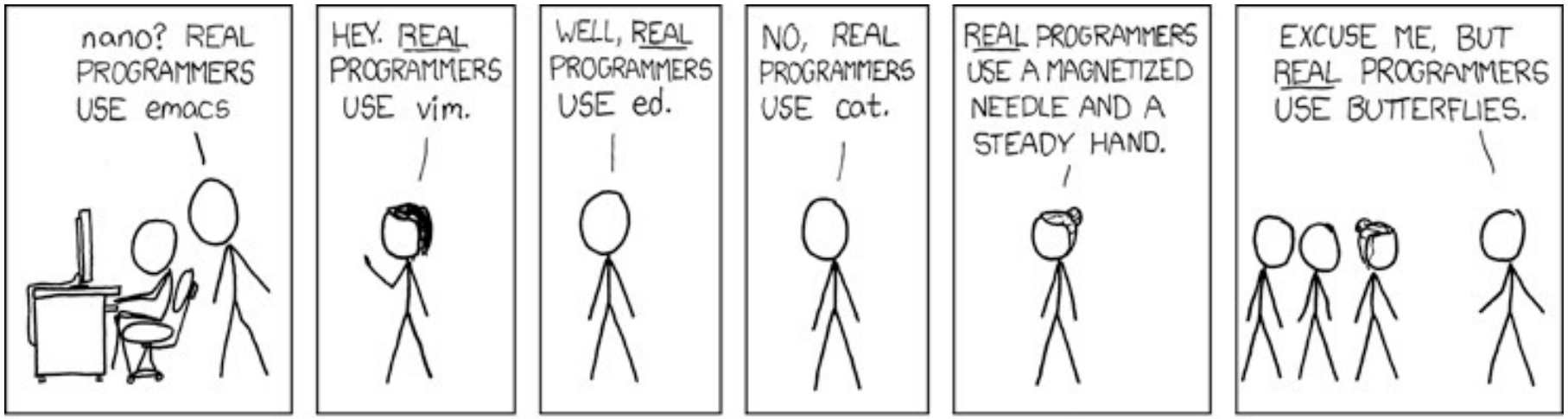
- Most HPC services have a service desk staffed by experts who are there to help you
- If something is not working as you expect then contact them

Research Software Engineers

- Many RSE groups around the UK that can provide advice and support
- https://society-rse.org/community/rse-groups/

2. Learn an in-terminal text editor

But which one?



https://xkcd.com/378/

- vim is very powerful and available everywhere
- emacs is more intuitive to use than vi and almost always available
- nano is simple to use but less powerful, not always available

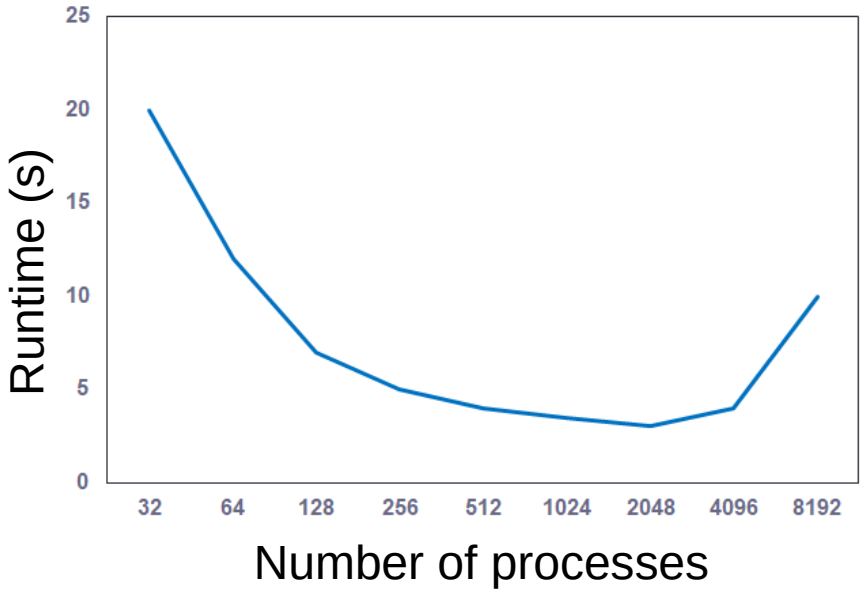
3. Submit test jobs when scaling up

- Running at large scale can have unexpected consequences.
- These can often be captured with a short test run.
- Most HPC facilities have a short/debug queue – use them!

4. Run basic benchmarking

Or: Why does it run slower when I used more cores?

- 100 node-hours of benchmarking can save 1,000's of node-hours.
  - Definitely true for hybrid systems.
- Popular HPC programs will have benchmarking data.
- Again, short tests can go a long way.
- This is a must on new architecture.

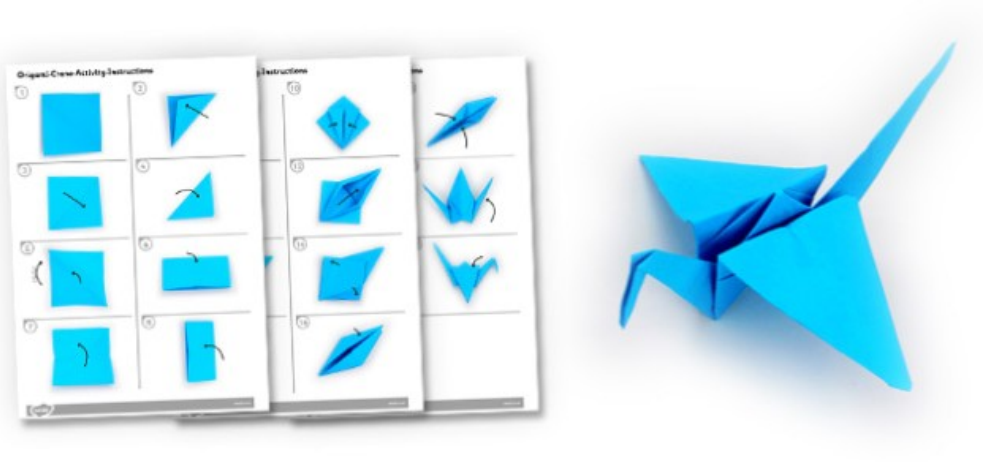


5. Plan for the future

It will be here sooner than you think.

- Have a data management plan
  - What files need transferring?
  - Does compression help or hinder?
  - How long will it take to transfer data?
- Think carefully before limiting your code.
  - A “quick fix” can have costly consequences!
- Consider future collaborators.

6. Read the docs...

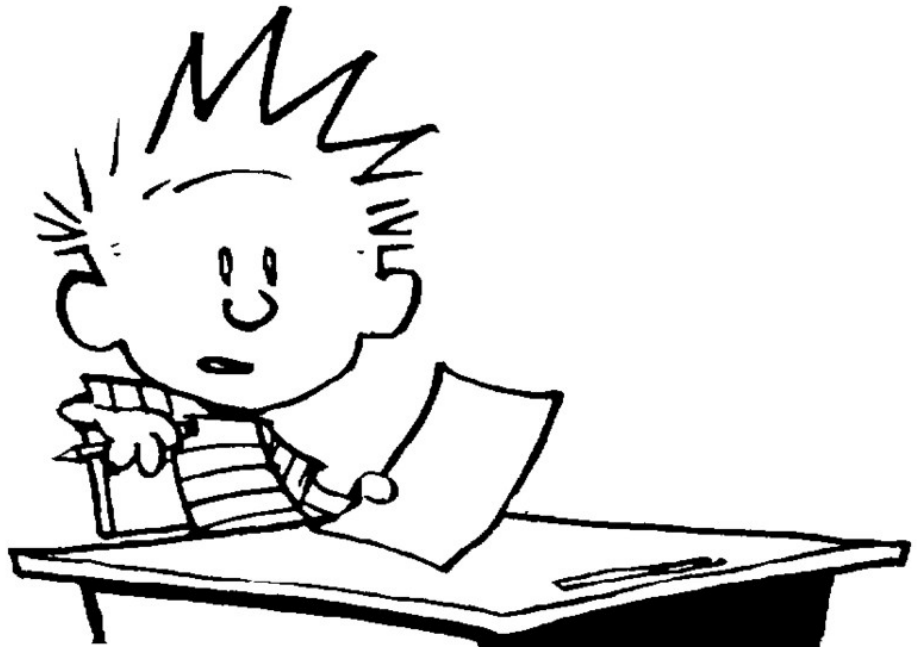


- Most HPC facilities provide good documentation.
- Documentation is the first port of call when providing assistance.
- More and more, the user community is welcome to join in improving the documentations:

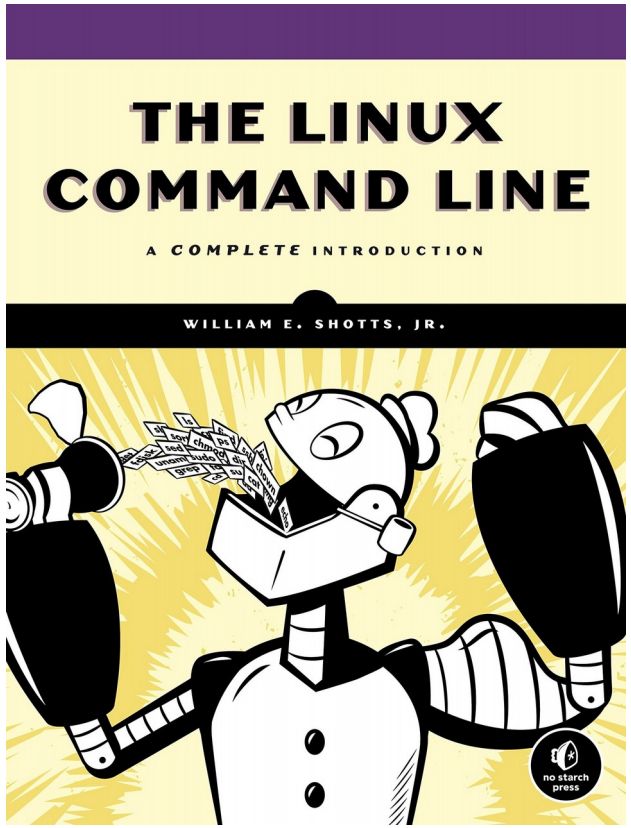
  https://github.com/ARCHER2-HPC/archer2-docs

7. Make use of training courses

- Lots of HPC, software, data analysis training material available
- ARCHER2: https://www.archer2.ac.uk/training/
  - Lots of free online training
  - Repository of past materials: https://www.archer2.ac.uk/training/materials/
- The Carpentries: https://carpentries.org/
- CodeRefinery: https://coderefinery.org/



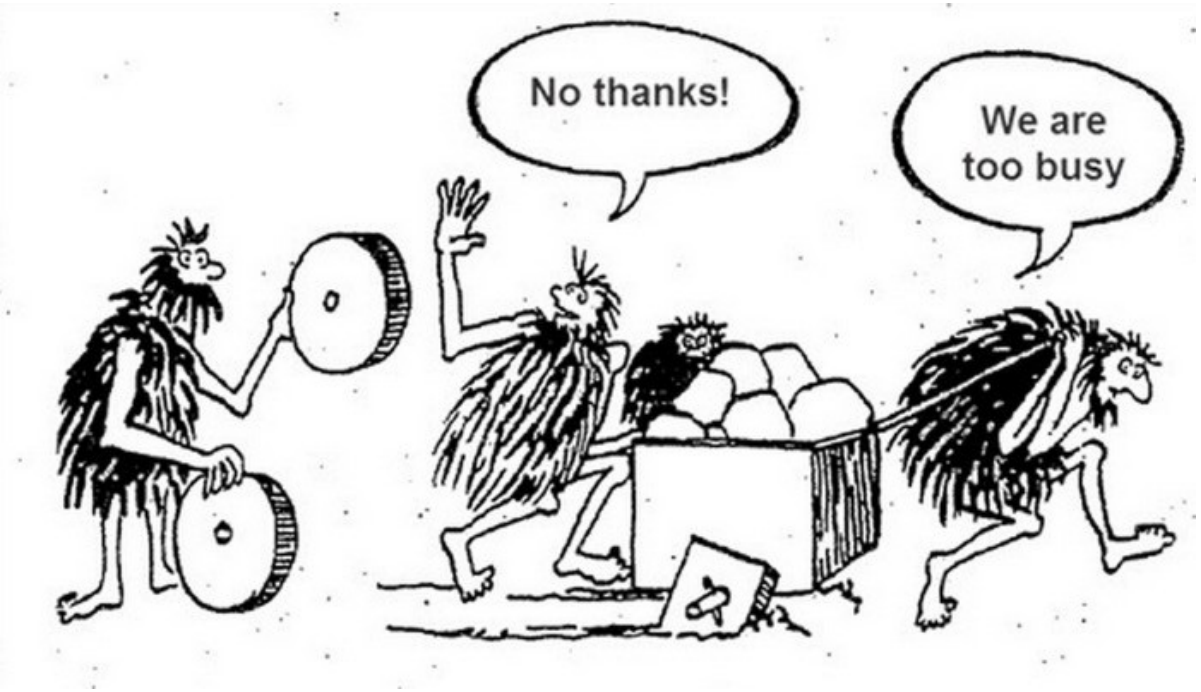
8. Learn Linux command line



- You will need basic knowledge to use HPC facilities.
- Modern Linux command line has many useful, powerful features:
  - E.g. sed, awk, paste, uuidgen
- Combining bash and Python can lead to very powerful capabilities

9. Don’t reinvent the wheel

- Build on top of existing and tested libraries as these are often faster.
  - HPC facilities often come with optimised libraries (as do some compilers).
- Where possible, use centrally-maintained software.



10. Learn Pandas or R

- Manipulating data is key to almost all research
- Pandas:
  - https://pandas.pydata.org/pandas-docs/stable/getting_started/intro_tutorials/index.html
  - Pandas can be parallelised using Dask
- R:
  - https://education.rstudio.com/learn/beginner/

11. Some that didn’t make the list

1. Understand your black boxes.
2. sudo will not solve anything.
3. Think before running though Python.
4. Consider visualising locally.
5. Don’t underestimate I/O costs.
6. Know the consequences of the code change you’re about to make

Ten tips

1. Don’t struggle alone
2. Learn an in-terminal text editor
3. Submit test jobs before scaling up
4. Run basic benchmarking
5. Plan for the future
6. Read the docs…
7. Take advantage of training courses
8. Learn Linux command line
9. Don’t reinvent the wheel
10. Learn Pandas or R