

ARCHER Software Carpentry boot camp and Introduction to Scientific Programming in Python

Mike Jackson, Andrew Turner, Arno Proeme, ARCHER CSE Team
 michaelj@epcc.ed.ac.uk
 aturner@epcc.ed.ac.uk
 aproeme@epcc.ed.ac.uk



Reusing this material

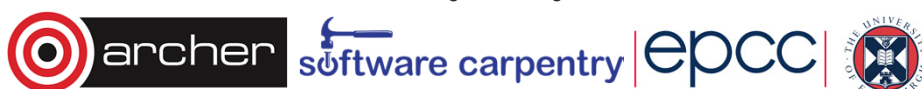


This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

http://creativecommons.org/licenses/by-nc-sa/4.0/deed.en_US

This means you are free to copy and redistribute the material and adapt and build on the material under the following terms: You must give appropriate credit, provide a link to the license and indicate if changes were made. If you adapt or build on the material you must distribute your work under the same license as the original.

Note that this presentation contains images owned by others. Please seek their permission before reusing these images.









<http://www.archer.ac.uk>
support@archer.ac.uk









<http://www.training.prace-ri.eu>

ARCHER in a nutshell

- UK National Supercomputing Service
 - Replacement for HECToR (they overlap by ~4 months)
- Cray XC30 Hardware
 - Nodes based on 2×Intel Ivy Bridge 12-core processors
 - 64GB (or 128GB) memory per node
 - 3008 nodes in total (72162 cores)
 - Linked by Cray Aries interconnect (dragonfly topology)
- Cray Application Development Environment
 - Cray, Intel, GNU Compilers
 - Cray Parallel Libraries (MPI, SHMEM, PGAS)
 - DDT Debugger, Cray Performance Analysis Tools



Performance

- HECToR #50 in top 500 with 830 TFlop/s
- ARCHER
 - Designed to provide 3-4 times scientific throughput of HECToR
 - #19 in November 2013 top 500 list with 1.65 PFlop/s
 - Fastest (known) computer in the UK
- Extract the best performance possible from the hardware



Optimization

- “More computing sins are committed in the name of efficiency (without necessarily achieving it) than for any other single reason ..
- ...Including blind stupidity” (Wulf)
- “We should forget about small efficiencies, say about 97% of the time:
- ...Premature optimization is the root of all evil.” (Knuth 1974)
- “performance variability that derives from differences among programmers of the same language ... is on average as large or larger than the variability found among the different languages.” (Prechelt 2000)



Three rules of optimization

- Don't – optimised code is not readable or maintainable code
- Don't....yet – designing code is not the same as optimising code
- Profile first – anything else is just a guess!
- How do we prevent ourselves introducing bugs when we're optimizing and parallelizing?
- Isn't our time more valuable than a computer's time?
- Hardware life < software life < our life



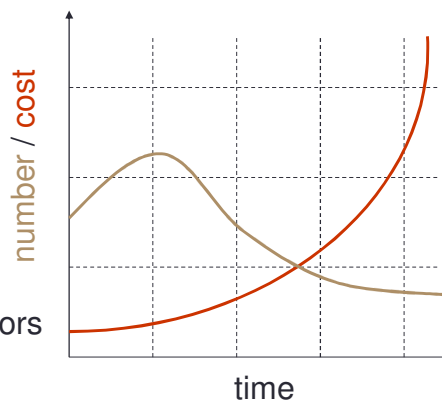
Optimize ourselves

50%-80% of errors
in
15%-20% of modules
(Davis 1995 quoting Endres
1975, Weinberg 1992)

50% of modules are error free
(Boehm and Basili 2001)

design errors \gg # coding errors
(Boehm et al. 1975)

Technical debt
1-10-100 rule



software carpentry

epcc



What about correctness?

Science

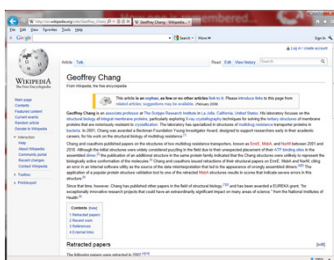


AAAS

We wish to retract our research article ... and both of our Reports...

An **in-house data reduction program** introduced a **change in sign** for anomalous differences...

Unfortunately, the use of the multicopy refinement procedure still allowed us to obtain reasonable refinement values for the **wrong** structures.



epcc



A skills gap

...SCIENTISTS AND THEIR SOFTWARE

A survey of nearly 2,000 researchers showed how coding has become an important part of the research toolkit, but it also revealed some potential problems.

> **45%** said scientists spend more time today developing software than five years ago."

> **38%** of scientists spend at least one fifth of their time developing software.

> Only **47%** of scientists have a good understanding of software testing.

> Only **34%** of scientists think that formal training in developing software is important.

Hannay et al, "How Do Scientists Develop and Use Scientific Software?"

SECSE '09 Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering, IEEE Computer Society, 2009. DOI: 10.1109/SECSE.2009.5069155

Images courtesy of Greg Wilson and Neil Chue Hong



software carpentry | epcc



Filling the gap



Software Sustainability Institute



DiRAC



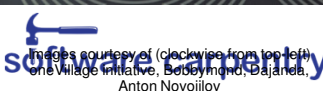
mozilla Science Lab



software carpentry | epcc



Important stuff



Images courtesy of (clockwise from top-left):
one village initiative, Bob Raymond, Dajana,
Anton Novojilov

How to ask for help

- Flag down a helper
- Stick up your sticky note
 - Red – I'm stuck, help!
 - Green – I'm fine

