Performance optimisation of research codes for the Supercomputing Wales programme: three case studies

Ed Bennett

@QuantumofEd

Swansea University
Prifysgol Abertawe

@SwanseaUni

SUPERCOMPUTING WALES
UWCHGYFRIFIADURA CYMRU

@SuperCompWales

SA2C

@sa2c_swansea

HPC-LEAP Conference, Cambridge University
11 July 2018

Slides: bit.ly/2zpSYQc

In this talk

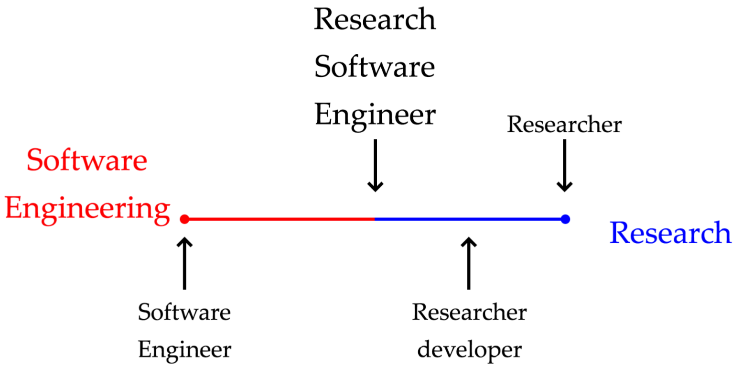
- The Supercomputing Wales programme
- What is a Research Software Engineer?
- Case studies:
  - Throughput optimization of a life sciences application
  - Improving parallel efficiency of a CFD application
  - Parallelising a lattice field theory code

Slides: bit.ly/2zpSYQc

The Supercomputing Wales programme

- £15m investment by EU, Welsh Government, and four Welsh Universities
  - Cardiff
  - Swansea
  - Aberystwyth
  - Bangor
- Two HPC facilities
  - 12,000 Skylake cores
  - 34 Nvidia V100 GPUs
  - 2 Bullion Analytics Factories
- 15 Research Software Engineers

What is a Research Software Engineer?

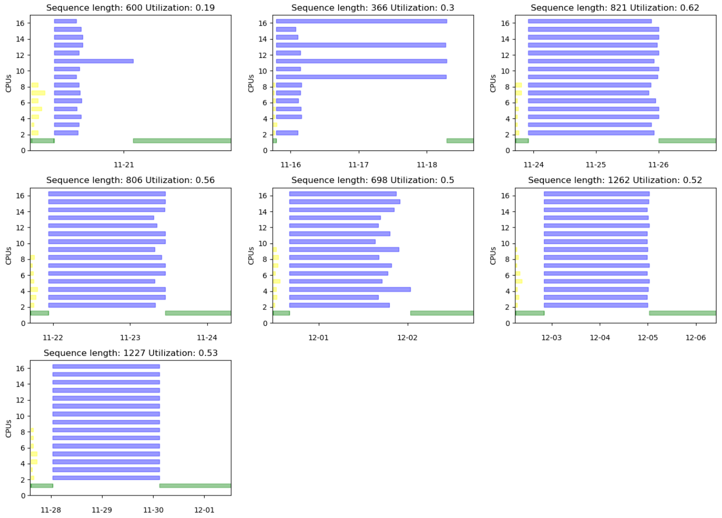


(from Simon Hettrick, https://goo.gl/znRRzb)

Supercomputing Wales RSEs

- RSEs:
  - Know software engineering
  - Know research
  - Develop software for research
- Supercomputing Wales RSEs:
  - Do all this
  - Know about HPC
  - Adapt, optimize, and benchmark research software for HPC

Throughput optimisation for life sciences



Slides: bit.ly/2zpSYQc

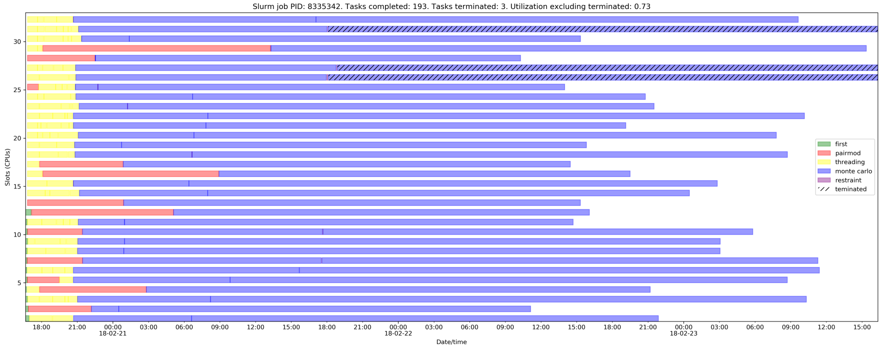
Options considered

- Serialise
- Separate jobs, SLURM dependencies
- Allocate a single job, run many jobs within it
  - Require a queue manager to organise this

METAQ (https://git.io/fNUxC)

- Written in bash
- Filesystem-based
- No support for dependencies
  - Now implemented

Initial results of throughput optimisation



Slides: bit.ly/2zpSYQc

Ongoing work

- Predictions of runtime for each job step
- Collecting data from current runs to allow this

Parallel optimisation of computational fluid dynamics software

- Solver for the 2D Boltzmann equation
  - BGK operator
  - Two-step discontinuous Taylor–Galerkin
- Space and velocity degrees of freedom
- Initially space parallelised (domain decomposition)
  - Memory bound
  - Performance not explicitly considered
- Fortran 77 code

Initial performance

Optimisations implemented

- Parallelise in velocity space
  - Less communication on this axis
- Optimise domain decomposition
  - Better load balancing
  - Less communication

Optimised performance

Parallelising lattice field theory code

- 2+1d Thirring model
- RHMC algorithm
- Serial Fortran 77 (and FORTRAN IV) code
- Domain Wall Fermions
  - Highly vectorisable
- Need for:
  - Larger volumes $\Rightarrow$ longer time per iteration
  - Stronger coupling $\Rightarrow$ more iterations
- $\Rightarrow$ Need parallelism

Initial tests

- Tune compiler options: 40% performance improvement
- Automatic multithreading (`-parallel`)
  - linear scaling to 4 threads
- Deliver $\sim 6\times$ improvement to researcher before starting MPI

Refactoring

- Implement regression testing
- Add per-site random numbers
- Remove indirection
- Move to free-form Fortran
- Replace loop operations with array operations where possible

  ```
        do 1000 i=1,10000
   1000    sum_x = sum_x + x(i)
  ```

  becomes

  ```
  sum_x = sum(x)
  ```

- `implicit none`
- `common` blocks → modules
- Global `parameters` → module

Adding MPI

- Add a 1-site halo around

Thanks for listening!

- Interested?
- Currently recruiting
- bit.ly/swansea-rse-2018
- Deadline: 20 July (next Friday)