An Introduction to High Performance Computing

Serial Assignment Feedback

8 November 2018

General Feedback

- A great start to this year’s HPC course!
- Many of you clearly worked hard on your submissions
- Remember, the overall quality of your report is very important
- Many of you only tried the minimum expected optimisations
- Not everything here will apply to **you**

Report Feedback

Report Feedback

- State which compiler (and what version) you use

- Lack of explanation and analysis of results
  - “I did this and it was faster”… but **why**?!
  - Back up with evidence

- Don’t report the same data twice (in a table **and** a graph)

- Analyse **all** the problems, not just one of them
  - They show different performance characteristics
  - The small one is the least interesting for HPC

Report Feedback

- Screenshots, terminal output, etc. are data. You must interpret and present this – no screenshots!

- Proofread – check spelling and grammar

- Page limit is **strict,** and must include everything, including references
  - Next time you **will lose marks**
  - No exceptions because you think you did more than was asked so you needed more space to talk about it
  - Many of you had lots of whitespace in…

Report Feedback

- Some of you didn’t include a title, or even your name!
  - This was part of the spec!

- Use `fixed-width (monospace) font` for code
  - Otherwise text can get hard to read and it may not be obvious when you’re referring back to the code

- When reporting a run time of 596 seconds, does it matter that it is actually 596.2186785 seconds?
  - Don’t use more decimal places than necessary!
  - Remember there is noise, so there are error margins: 0.093 vs 0.098 s is not really a big improvement

Report Feedback

- Use subheadings to break up large sections of text
  - Don’t use strange subtitles either, e.g. Body
  - Don’t stuff a whole section into a single paragraph

- Put evidence, timings, speedups, etc. next to the relevant text; **not** in a summary table at the end

- But do include a summary table of final runtimes for all inputs
  - Make it clear what your final configuration was

Report Feedback

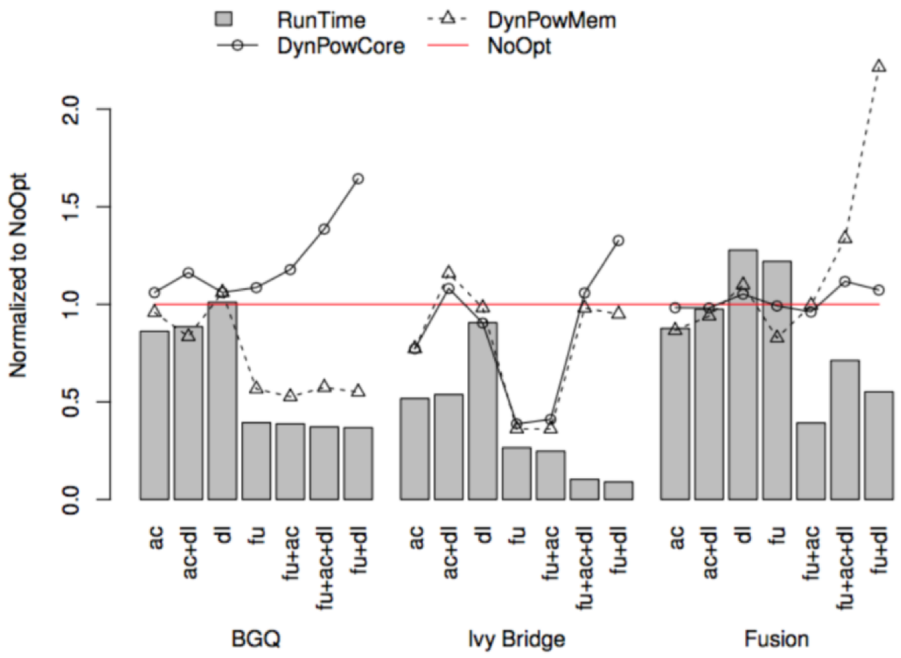
- Speedup as a % is confusing
  - 100% faster might be same speed, or half speed, or …

- Better to use *X* times faster: `old_time/new_time`
  - E.g. 100 s to 50 s results in a 2X speedup

- Useful to measure your achieved memory bandwidth and arithmetic performance
  - Compare to the hardware's peak to see how much room for improvement there is
  - But keep in mind that *peak* may not be *achievable*

Report Feedback

- **Don't** do a timeline of optimisation graph (or table)
  - Many optimisations depend on previous optimisations
  - This graph is a good one, for only 3 optimisations, but would be intractable for your coursework



Leon, E. et al. Optimizing Explicit Hydrodynamics for Power, Energy and Performance. IEEE Cluster, 2015.

Common Mistakes

Common Mistakes

- `cc` compiler is the system default (GCC 4.4.7)

- Incorrect filenames and build errors
  - Check what you submit!
  - Be careful with whitespace issues

- Test **all** inputs
  - Must complete running and validate
  - Don’t do your work based on a single one

- Next time you will lose **more marks** for build and run errors

Common Mistakes

- The provided code doesn’t trash the cache
  - It just makes very poor use of it

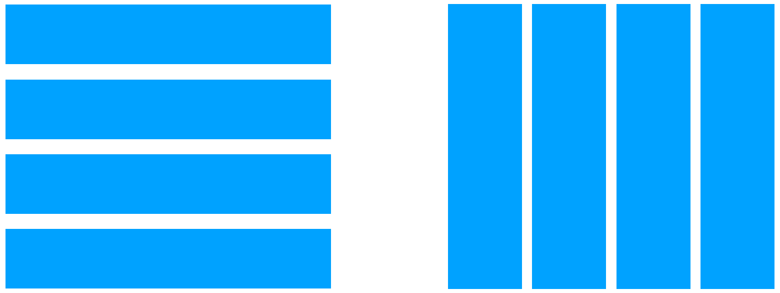
- A lot of you mentioned that the compiler does plenty of optimisations…
  - ... but attempted to do them all by hand before using optimisation flags…
  - … or used an old compiler version
  - You can tell if the compiler has done an optimisation you would have done by hand by looking at the assembly code

- The stencil code is **memory-bandwidth-bound**
  - Not memory-bound

Row/Column-Major

Lots of mistakes in this sentence!

- “C is a row-major language, so I changed from A being column-major to row-major”
- Arrays are on the heap and use pointer arithmetic to access data, so C stack arrays are irrelevant here
- Row-major   Column-major



- What’s important: ensure data order and access pattern are the **same**

Optimisation Feedback

Optimisation Feedback

- Start with changing the compiler and flags
  - It’s no-effort optimisation!
  - Has a big effect which nullified other manual changes later

- Intel compiler defaults to `-O2`
  - Compare to GCC fairly by turning off optimisations: `-O0`

- If –O3 is not faster than –O2, then your code likely can’t take advantage of some hardware features, e.g. vectorisation
  - Similarly, if you do an optimisation by hand and you don’t observe the expected result, double-check that you haven’t made an implemention mistake

Optimisation Feedback

- Vectorisation is **really** important for performance
  - Check the compiler vector reports `–qopt–report=5`
  - Many of you simply checked the loop vectorised at all, and didn't investigate further
  - Look at vector width, alignment, access pattern, types of load/stores, etc.
  - Did the most performance important loops vectorise, not just unimportant ones?

Optimisation Feedback

- Removal of branches in main loop:
  - Many of you said this was due to branch misprediction causing bubbles in the pipeline
    - Some of you said they cause branch predictions, which should be avoided
    - Nobody presented any evidence that this is an issue
  - The compiler might generate masked vector operations, or it might produce a peel loop
  - Your aim is to get the compiler to understand what you’ve done, so that it can enable **effective vectorisation**
    - Some of you noticed that the compiler sometimes understands you code better if you leave the branches in!

MPI Assignment

- Deadline is 5pm, Friday 14th December (week 11)
- Take our feedback into account (both individual and from today’s lab)
- Start with your optimised serial code
  - Make sure it works on non-square grids, as the grid is unlikely to be decomposed to a square on every rank
- Do include any new serial optimisations you find, but don’t repeat yourself from Assignment 1
- Remember, this is a report marked coursework too
  - Leave enough time to write it