Rules and Regulations of the 3rd Annual IHPCSS Challenge

Trophy bears no relationship to reality.

Starting Point

- We give you working codes in MPI, OpenMP and OpenACC
  - copy “challenge.tar” from /home/dsh/ihpcss17/.
- Set up to tackle a problem of size 672x672
  - challenge is to run a much larger problem
- Straightforward solutions
  - no attempts at parallel (or serial) optimisation
  - each only uses a single model
- No compilation instructions
  - you decide how best to compile

General Rules

- Due Thursday midnight (!)
- 4 Nodes of Bridges
- Use any combination of MPI, OpenMP, OpenACC and Python
  - base versions (C and Fortran) for MPI, OpenMP and OpenACC on moodle
  - single tar file: challenge.tar
- How fast can you run a 10752 x 10752 Laplace code to convergence?
  - weird size chosen to decompose exactly on, e.g., 2, 4, 28 and112 procs
  - can use smaller size of 672 x 672 for development

Some Specifics

- Can't change kernel (Must retain two core loops source)
- Can change number of MPI processes (Does not have to be 112 or 4)
- 1 Source File
- 1 Combined Environment/Compile/Submit/Execute script
  - to make it easy for us to run your solutions!
- Mail to d.henty@epcc.ed.ac.uk by deadline

Rules For Lawyers

- No libraries
- Don’t mess with timer placement
- ?

Reality Checks

- Serial code converges at 3580 time steps. Yours should too.
- As we know, this is not enough to verify correctness. You should find point [8064][10702] in C and (10702,8064) in Fortran converges to 17.1 degrees.
- As discussed, the 10752 result differs from the 672 result.*
  - smaller problem converges in 3264 time steps
  - check values: [504][622] in C, (622,504) in Fortran = 15.5 degrees
  - Plugging in Gauss-Seidel or Successive Over Relaxation (SOR) would be easy and interesting. But, not for our contest.

http://www.cs.berkeley.edu/~demmel/cs267/lecture24/lecture24.html is a brief analysis of these issues.

Printing out the test point

- Straightforward in serial, OpenMP or OpenACC
  - single process, temperature stored in a single global array

- More complicated when you introduce MPI – must locate owning process
  - if (8064/columns == mype+1) then
  - write(*,*) 'PE ', mype, ': T(10702,8064) = ', temperature(10702,columns)
  - end if

  - if (8064/ROWS == my_PE_num+1)
  - printf("PE %d: T(8064,10702) = %f\n", my_PE_num,Temperature[ROWS][10702]);

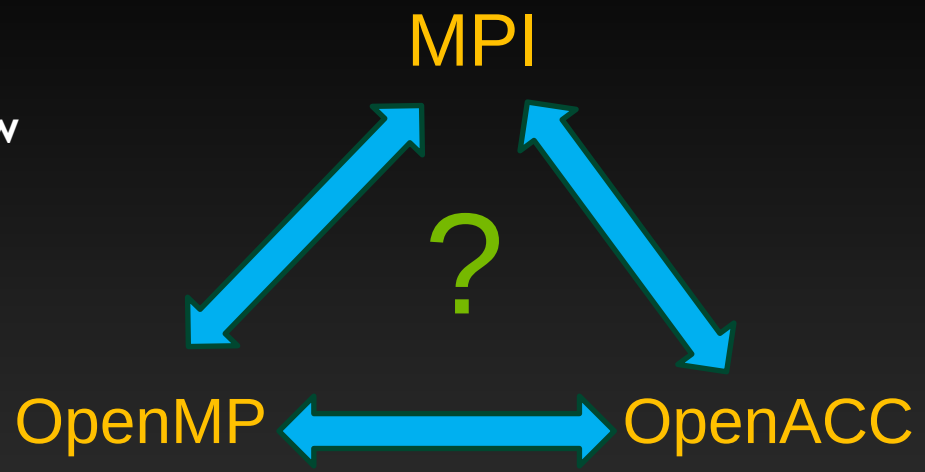
- This hacky piece of code requires at least 4 MPI processes!

http://www.cs.berkeley.edu/~demmel/cs267/lecture24/lecture24.html is a brief analysis of these issues.

Suggested Things to Explore

User Guide is your friend!

- Compiler flags
  - -O3
- Compiler
  - see Bridges documentation for how to use different modules
- MPI Environment Variables
  - man mpi
- Thread placement
  - google for KMP_AFFINITY



Decision

- On Friday morning we will take the top self-reported speeds and run them in an interactive session
- Timings not within 10% of self-reported time will be disqualified
- Codes should print out “test point” at conclusion of run.
- Best of two runs for each finalist will determine winner