High Performance Computing Challenge (HPCC)

Benchmark Optimisation

NOVEMBER 27, 2017

CENTRE FOR HIGH PERFORMANCE COMPUTING

UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

Proposed Schedule

- Monday – HPCC
- Tuesday – WRF
- Wednesday – LAMMPS
- Thursday/Friday – Start from scratch
  (reinstall OS, setup networking, compile benchmarks, etc.)
- Saturday – Shuttle to Pretoria, competition starts

HPCC

- The HPC Challenge benchmark consists of 7 tests:
  - **HPL** – the Linpack TPP benchmark which measures the floating point rate of execution for solving a linear system of equations.
  - **DGEMM** – measures the floating point rate of execution of double precision real matrix-matrix multiplication.
  - **STREAM** – a simple synthetic benchmark program that measures sustainable memory bandwidth (in GB/s) and the corresponding computation rate for simple vector kernel.
  - **PTRANS** (parallel matrix transpose) – exercises the communications where pairs of processors communicate with each other simultaneously. It is a useful test of the total communications capacity of the network.
  - **RandomAccess** – measures the rate of integer random updates of memory (GUPS).
  - **FFT** – measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform (DFT).
  - **Communication bandwidth and latency** – a set of tests to measure latency and bandwidth of a number of simultaneous communication patterns; based on *b_eff* (effective bandwidth benchmark).
- http://icl.cs.utk.edu/hpcc/

Compilation

- Use Intel Parallel Studio XE or gcc+openmpi+atlas.
  - **Some benchmarks may be faster with OpenMpi/MPICH, HPCC is not one of them**.
- The procedure for building changes based on your version of Intel Parallel Studio XE.
  - **2015** version is most stable, install it as a backup if things are going wrong.
  - **2016/17/18** versions work but have **different compilation flags** and multithreading issues.
  - Compare the performance of the 2015 version and whatever you already have installed.
- **Makefile.intel64** – Intel MKL ships with a Makefile for HPL, adapt it to see if it performs better.

WITS UNIVERSITY

HPL

High-Performance Linpack (HPL)

- Primary benchmark in HPCC and one of the only ones you can directly optimise.
- Solves a dense $n \times n$ system of linear equations, $Ax = b$.

- Measures floating point computing power (output in GFlops).
- Widely used benchmark; Top500 list is determined by HPL score.

WITS UNIVERSITY

Running HPL

- **hpccinf.txt** – input file, specifies problem settings for HPL.
  - Initially named "_hppcinf.txt", rename it.
- Four main settings you should optimise:

1. **N** – **Problem size** (size of matrix $N \times N$). Runtime scales quadratically.
   - Choose N to fit ~80-90% of your memory; too large will cause swapping to disk (very slow).
2. **NB – Block size.**
   - Choose NB primarily based on your CPU cache size.
   - Basically trial-and-error searching for right size. Start from 96 and work upwards in increments of 8: 96, 104, 112, 120, …
3. **P** – **number of rows**.
4. **Q** – **number of columns**.
   - P and Q determines the size of your process grid.
   - P×Q is the total number of processors used in mpirun.
   - P×Q should be as square as possible, so P and Q should be almost equal with Q slightly higher (e.g. 4x6 rather than 1x24)

- **Specify nodes and number of processors when running:**
  - mpirun -np 24 -hosts node0 node1 node2 ./hpcc

WITS UNIVERSITY

Running HPL

1	# of problems sizes (N)
10000	Ns
1	# of NBs
96	NBs
0	PMAP process mapping (0=Row-,1=Column-major)
1	# of process grids (P x Q)
8	Ps
9	Qs

WITS
UNIVERSITY

Running HPL

- Can specify multiple settings. HPL will try each combination in turn when run.

2	# of problems sizes (N)
10000 12000	Ns
2	# of NBs
96 104	NBs
0	PMAP process mapping (0=Row-,1=Column-major)
1	# of process grids (P x Q)
8	Ps
9	Qs

- In the above example, four combinations will be executed:
  - 10000/96, 12000/96, 10000/104, 12000/104

HPL Output

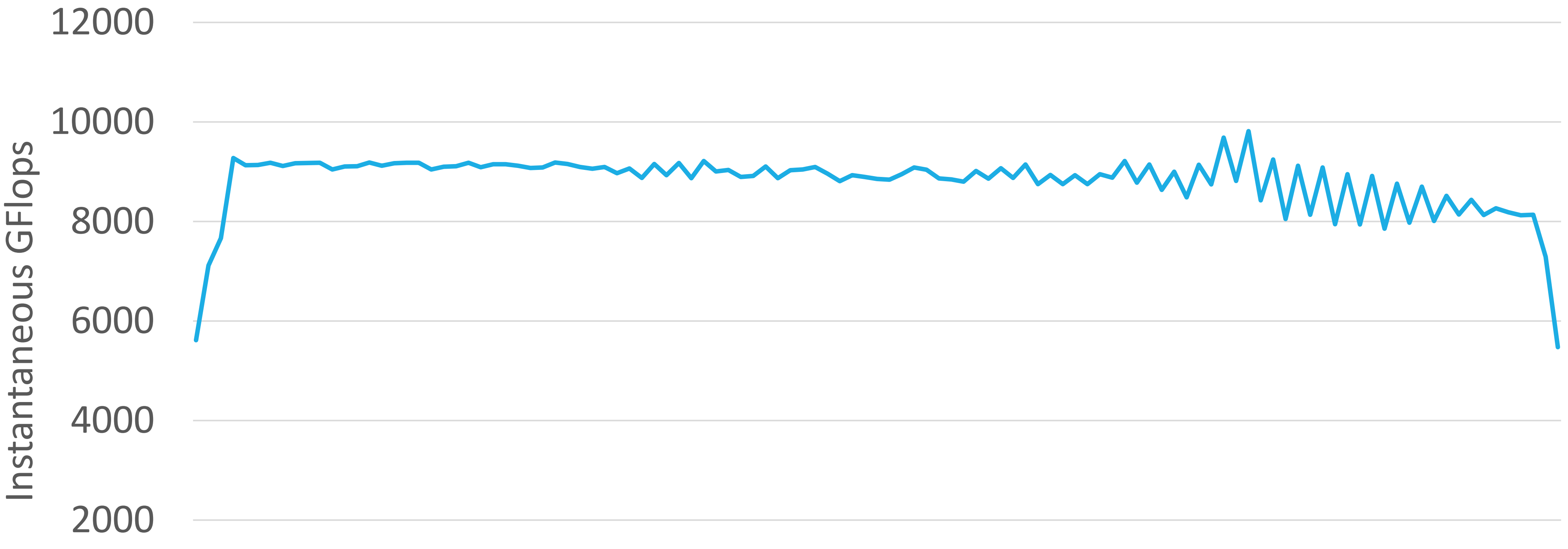
- **hpccoutf.txt** – results
- Look for HPL Gflops result near end of file:

```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR11C2R4      115200   192    16    18             180.95              5.633e+03
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0017589 ...... PASSED
================================================================================
```

WITS UNIVERSITY

HPL

- HPL ramps up in the beginning, warming up and distributing blocks.
- Ramps down at the end, when not all CPUs have work available.
- Optimise N to maximise the time that all CPUs spend doing useful work.



WITS UNIVERSITY

HPL Calculator

Top500 HPL Calculator

Please Submit Your System's Data:

(Numerical values only)

Number of Nodes (e.g. 3168): 3

Cores Per Node (e.g. 24): 24

Speed Per Core (GHz) (e.g. 2.5): 2.5

Memory Per Node (GB) (e.g. 96): 96

Instructions Per Cycle (e.g. 16): 16

Submit Data

Developed By: Mohamad Sindi

- Estimates your HPL performance based on your cluster hardware.

- http://hpl-calculator.sourceforge.net/

Below is the 100% theoretical system performance (Rpeak) in GFLOPS along with other common efficiencies:

70%	72%	74%	76%	78%	80%	82%	84%	86%	88%	90%	92%	94%	96%	98%	100% (Rpeak)
2015	2073	2131	2188	2246	2304	2361	2419	2476	2534	2592	2649	2707	2764	2822	2880

More Details (Running HPL)

WITS UNIVERSITY

HPL Calculator

- Also gives possible settings (P, Q, N, NB) for running HPL.
  - Use as starting point, optimize yourself.
  - Start with a lower N and work up to the recommended values.

Below is the 100% theoretical system performance (Rpeak) in GFLOPS along with other common efficiencies:

70%	72%	74%	76%	78%	80%	82%	84%	86%	88%	90%	92%	94%	96%	98%	100% (Rpeak)
2015	2073	2131	2188	2246	2304	2361	2419	2476	2534	2592	2649	2707	2764	2822	2880

More Details (Running HPL)

Possible combinations of how the HPL grid might look like in terms of P and Q:

P	x	Q
1	x	72
2	x	36
4	x	18
8	x	9

Possible memory percentages (i.e. N values) aligned with various NB values:

N/NB	96	104	112	120	128	136	144	152	160	168	176	184
80%	157248	157248	157248	157200	157184	157216	157248	157168	157280	157248	157168	1571
82%	161184	161200	161168	161160	161152	161160	161136	161120	161120	161112	161216	1611
84%	165120	165048	165088	165120	165120	165104	165024	165072	165120	165144	165088	165[illegible]
86%	169056	169000	169008	169080	168960	169048	169056	169024	168960	169008	168960	168[illegible]
88%	172992	172952	172928	172920	172928	172992	172944	172976	172960	172872	173008	172[illegible]

Theoretical Peak Performance

- Theoretical peak performance (TPP):

Node performance in GFlops = (CPU speed in GHz) x (number of CPU cores) x (CPU instruction per cycle) x (number of CPUs per node)

- **Efficiency = (HPL Score in GFlops) / (TPP in GFlops)**
- Actual GFlops score from HPL will always be lower.
  - Aim for ~70-80% of the cluster's TPP during the competition.
  - Efficiency will be lower on your current machines because 1GBe network is slow.

WITS UNIVERSITY

GOOD LUCK!