ClusDur Hero HPL Report

Cluster Description

- NFS file share across all nodes.
- Infiniband network, all nodes had infiniband drivers installed and configured for mpi runs.
- Head node was a prometheus agent configured to export to Grafana.
- All compute nodes had a prometheus node exporter, only queried once every 15 minutes with a minimal set of metrics to minimise performance impact.
- Nodes with least memory or performance/mpi issues were iteratively detected and removed from our hostfiles.
- Images were used to enable consistency across compute nodes.

Software Stack

- Node reservations done through chameleon cloud web GUI due to issues with python chi in previous assignments.
- Node setup and config: Ansible (scripts used https://github.com/DUSCC/cluster-provisioning)
- HPL: OpenMPI, OpenBLAS as most consistent.
- Python + bash used for scripting during the runs
- Openstack CLI
- Prometheus + Grafana for observability of nodes (performance monitoring)

HPL Tuning

- Used our best HPL.dat from the HPL assignment for a single node as a baseline and the website https://www.advancedclustering.com/act_kb/tune-hpl-dat-file/.
- OpenMPI and OpenBLAS were chosen due to stability, as the Intel OneAPI toolchain was inconsistent in previous runs. These are widespread distributions of MPI and BLAS and have previously produced good benchmark scores using them in previous challenges and in our own experiments on other clusters.

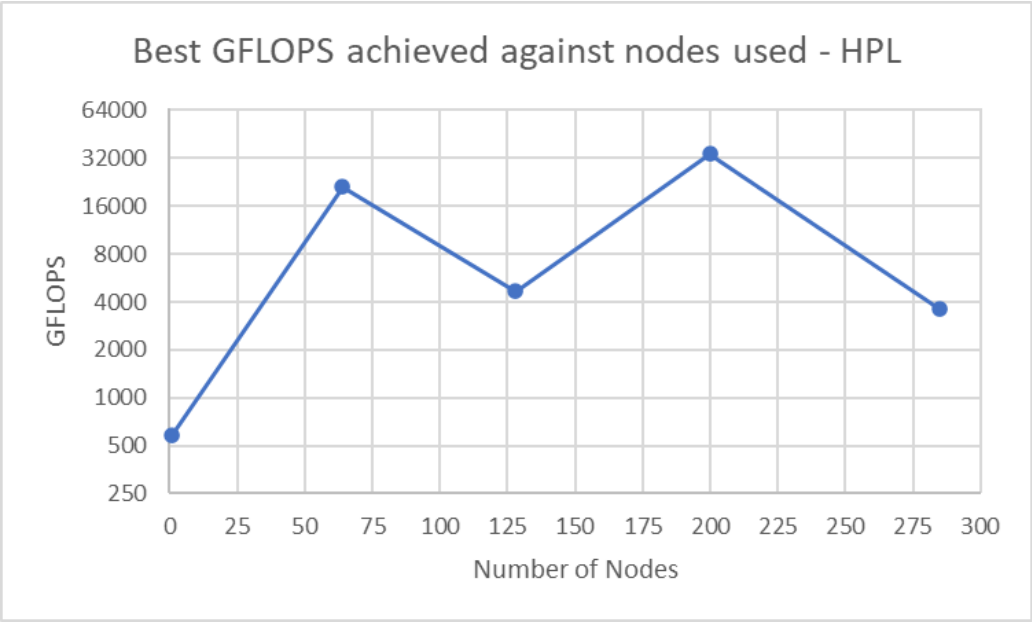
- We changed the makefile to remove the -g debug flag and change -O2 optimisation to -O3, as well as set -march=native. These compilation changes should yield the highest performance in nearly all cases, but by the time these were set we were facing severe problems with our benchmarks that will be discussed later. As a result, these perhaps had a 5% impact but negligible when in context of the poor scores.

- Used the same blocking factor (NBs) for all runs as the HPL assignment, based on evidence from those runs backed up by research on the processor used. These can be found here: https://www.intel.com/content/www/us/en/develop/documentation/onemkl-linux-developer-guide/top/intel-oneapi-math-kernel-library-benchmarks/intel-distribution-for-linpack-benchmark-1/configuring-parameters.html, which suggests that for Intel Haswell CPUs, NB=192.

- Tried to keep Ps and Qs as close as possible, and P < Q. For example, with 300 nodes of 20 processors (6000 processes), we would use 75x80. Intel also suggests this in the guide linked above.

- Dynamically changing our HPL.dat file as to fit what we found best through experimentation. There is some level of “randomness” between each run so we couldn’t fit to exact variables, but we played around with N, NBs, and the process grid until we were happy with performance.

Hpl plot (Log base 2 y-axis):



Hpl scores:

1 node - 584 GFLOPS
64 nodes - 21019 GFLOPS
128 nodes - 4640 GFLOPS (!)
200 nodes - 33914 GFLOPS
285 nodes - 3608 GFLOPS (!!)

Challenges

- Some nodes would have less memory (32GB or 48GB vs 64GB) or be unreachable after allocation, scripts were used to exclude these nodes from HPL runs.
- Rate limiting when trying to incrementally lease all the nodes.
- Slurm set up issues in the weeks leading up to the hero run. Had to use automatically generated and maintained host files, and use mpirun directly (rather than srun)
- (!) Mysteriously ran into problems getting results with any number of nodes after the cluster was fully set up, even when running the same HPL.dat as before with the same nodes. We were

not able to track down the reason behind this issue during the competition. We will follow up with a section on this (!)

Mysteriously poor results (out of nowhere!)

To put yourself in our shoes, we'll give a quick run-down of the rough timeline of our challenge:

- **2pm** - Challenge starts!
- **4pm** - People's lectures start to finish so they start to work. We have about 40 nodes up and more spinning up every few minutes. We have a 28 node run going (with suboptimal parameters). Using the same executable that we compiled in the HPL challenge (pre-infiniband).
- **6pm** - We've finished multiple runs by now. Everything is going as expected. 28 node run provided good results - 12.5 TFlops with suboptimal parameters (reasonably low N at 160k). The optimal value we'd expect (based on single node runs) would be 16.2 TFlops roughly so scaling well!
- **7pm** - Continue running tests - We run a 50 node and break our record with 27 TFlops! Still scaling nicely!
- **8pm** - Eat dinner and run a 200 node test over dinner with a low N (100k) as to get a short test. Get 34 TFlops which ended up being our highest score. This was much lower than we expected but put it down to low N so wanted to scale. By now, we had nearly 300 nodes up.
- **9pm** - Things start going wrong. We try to do a few 300 node tests but they fail due to some nodes not working (expectedly). We kick them out of the hostfile and continue booting up more to replace them. We continue doing extra tests but realise we aren't scaling upwards and start getting significantly poorer results (like 7 TFLOPS)
- **9pm** - We're rerunning the exact same tests as before (same HPL.dat file, using roughly the same subset of nodes as previously) and rather than say getting high 20s TFLOPS, we're getting ~6 TFLOPS.
- **9pm** - Recompile in a separate directory - realise maybe we haven't compiled since we've had infiniband so try this. No improvement so go back to old executable. Note that we did try this IB executable again later on with still no improvement.
- **10pm** - Starting to get lost. We create scripts to run HPL on all the nodes we have available (1 node per run, but all in parallel) and append the GFLOPS to a file. Scan through the file and kick out poor performing ones or ones that failed to run. By this point, we have about 285 nodes left as some are also down for maintenance so we continue testing for these
- **12am** - People go home from campus but half the team continue working on it. Realise we need to get runs for 64 node and 128 node too so we do this.
- Realise something is odd - performance hasn't increased, but things actually scale (more) like they should when we have less nodes. We try again on a few nodes (ie 20 nodes) and get 7.78 TFLOPS which isn't awful. We try smaller tests (N=100k) on 280 nodes and get 5.6 TFLOPS (is awful for number of nodes). Try scaling up N, value only increases to about 7 TFLOPS
- **1am** - Restart the first 100ish nodes and try using only these - no better.
- **3am** - Still trying to figure this out. Runs scale up to about 30 nodes and then level out and then start dropping again. We can't break the 7-TFLOP barrier that was once so easy to do. We have to go to bed by now, but one can hope things are better in the morning
- **11am** - Members of the team are waking up and trying runs again. Performance is still exactly the same and high-node runs yield useless results.
- **1pm** - End of challenge. We've submitted what we can, proved we could do runs on 285 nodes but not before we were getting poor performance. Sadly, the best we could show was 34 TFLOPS which should have been much higher (with the way we were scaling initially, 100-150 TFLOPS)

- **Now** - Still have no clue what was up. In hindsight, perhaps we should have reprovisioned everything through the night but that would have taken hours. We do honestly believe that the issue lies server side as we were rigorous in our methods, did not change anything or much at all and double, triple and quadruple checked as much as we could. That's part of the challenge, but we hope you can see our position - there wasn't really anything we could have done differently.

Take-Aways

- The main challenge takeaways were sticking to a systematic methodology when trying to sort our unexpected issues, which were sadly so strange we were unable to reproduce our original better results.
- Another key take-away was our approach to experimentally vary different factors affecting performance so we could have an idea of what factors made the biggest differences to performance.
- Our ability to think of solutions to problems faced was also a major take-away and although we perhaps weren't as successful as hoped we believe it to be an incredibly valid real world experience facing issues similar to those faced in real HPC scenarios.
- A larger emphasis on actual task scheduling (such as Slurm) would have made experimenting significantly easier with scale.