TCO

I HPL Benchmark

The High-Performance Linpack Benchmark (HPL) executable was generated from source (HPL version 2.3) [9] using the Intel Parallel Studio XE Developer Suite (2018.1.163) [8]. The identical quasi-static[1]. binary (`xhpl`) has been used on one node of the CHPC notchpeak cluster and on one node of the Amazon Cloud Platform AWS [1].

II HPL Runs on CHPC's notchpeak cluster

The HPL Benchmark was run on the node `notch`139 which has the following computational characteristics:

- `OS`: Centos7 - Kernel: 3.10.0-957.27.2.el7.x86_64
- `Model Name CPU`: Intel(R) Xeon(R) Gold 6230 CPU @ 2.10 GHz - Dual socket - 40 physical cores
- `Memory`: 192 GB
- `Local Hard Drive`: 7200 RPM - 1.8 TB

A series of preliminary HPL iterations were performed on `notch`139 to optimize its HPL input parameters (maximization of the flop count). The energy used during each HPL run was recorded from the outlet feeding solely `notch`139, as reported by the Power Distribution Unit (PDU). In total 10 identical HPL runs using all 40 cores, were performed within the local scratch space of the node. The results for the HPL runs on `notch`139 are displayed in Table 1.

Run	Time (s)	Energy Used (kWh)	Tflops
1	1377.12	0.147	1.6686
2	1381.76	0.147	1.6656
3	1382.78	0.148	1.6610
4	1381.39	0.147	1.6628
5	1375.66	0.148	1.6704
6	1377.93	0.147	1.6677
7	1371.46	0.148	1.6738
8	1374.53	0.147	1.6701
9	1378.70	0.147	1.6666
10	1388.28	0.149	1.6532
Mean	1378.96	0.1475	1.6660

Table 1: Results for the HPL runs on `notch`139.

III HPL Runs on AWS

The identical HPL executable was run on a node comparable to `notch`139. We opted for a VM of the type `c5d.metal` in the region US West (Oregon) [11] (bare metal compute-heavy instance) with the following characteristics:

[1]The executable is still dependent on a few system libraries such `libpthread.so`, `libm.so`, etc.

- `OS`: RHEL8
- `Model Name CPU`: Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00 GHz - 48 physical cores
- `Memory`: 192 GB
- `Local Hard Drive`: 4 x 900 GB SSD

We proceeded in a similar fashion as in the previous case: we again optimized the HPL input parameters. We performed 6 identical HPL runs using all 48 cores. The results for the HPL runs on the AWS node (`EC2 & On-Demand approach`) of the type `c5d.metal` are displayed in Table 2.

Run	Time (s)	Tflops
1	874.38	2.6222
2	874.97	2.6211
3	889.43	2.5776
4	877.47	2.6135
5	873.34	2.6260
6	872.65	2.6282
Mean	877.04	2.6148

Table 2: Results for the HPL runs on an AWS node of the type `c5d.metal`.

IV Cost comparison

The HPL calculation on `notch139` lasted on average 1378.96 s. The node produced on average 1.6660 Tflops and consumed on average 0.1475 kWh (i. e. an average power consumption of 385.07 W). The cost-factors for the on-premise solution (`notch139`) are displayed in Table 3.

Cost Factor	Price ($)
Purchase cost per node	5,107.56
Rack Infrastructure Cost (5 years)	1,000
Monthly Power Infrastructure Cost	0.00
Monthly Data Center Cost	44,472.91
Expected Monthly Power Cost	29.24
Monthly Personnel Cost	45.40
Personnel Setup Cost	136.20

Table 3: Cost factors for the on-premise solution

The life expectancy of node `notch139` is 5 years. It was purchased on Aug. 2, 2019 for $5,107.56. The monthly purchase price amounts to $\frac{\$5,107.56}{60} = \$85.13$. The rack infrastructure cost per node over its life time is $1,000.00. The monthly rack infrastructure cost for node `notch139` is therefore $16.67. The latter number includes the monthly power infrastructure cost.

CHPC's monthly data center cost totals $44,472.91. CHPC's data center consists of 113 racks each having 42 rack units (RU) of space. The node `notch139` occupies 1 RU. Its monthly data center cost is thus $\frac{\$44,472.91}{113 \times 42} = \$9.37$.

The expected monthly power cost for the node is based on the current price of electricty: $0.08 per kWh. The monthly power cost for the node is therefore:

$$\text{Expected Power Cost/month} = \frac{\$0.08}{\text{kWh}} \frac{0.1475\,\text{kWh}}{1378.96\,\text{s}} \frac{3600\,\text{s}}{\text{h}} \frac{24\,\text{h}}{\text{day}} \frac{365\,\text{day}}{\text{year}} \frac{1\,\text{year}}{12\,\text{month}}$$

$$= \frac{\$0.08}{\text{kWh}} \frac{281.10\,\text{kWh}}{\text{month}}$$
$$= \$22.49$$

This number represents only the electricity consumed by the node itself. To account for the power used to cool it (and similar infrastructure power usage) we multiply the usage by the power usage effectiveness[2] (PUE) factor which is currently 1.3. The expected monthly power cost for `notch139` amounts to: $\$22.49 \times 1.3 = \$29.24$.

CHPC's yearly personnel cost for the compute service amounts to $\$741,000$. The monthly personnel cost per node is therefore: $\frac{\$741,000}{12 \times 1360} = \$45.40$ where the number 1360 stands for the total number of CHPC's compute nodes.

The personnel setup cost for a node is estimated to be on average 3 months of monthly personnel cost: $136.20. On a monthly basis the personnel setup cost amounts to $2.27.

The total cost to run `notch139` for a month is therefore:

$$\begin{aligned} \text{Tot. Cost/month} &= \$85.13 + \$16.67 + \$9.37 + \$29.24 + \$45.40 + \$2.27 \\ &= \$188.87 \end{aligned} \tag{1}$$

The cost to run a `c5d.metal` node in the `On-Demand` mode but **without** the consideration of data transfer, setup, external storage is currently valued at $\$4.608/\text{h}$ [3]. To run the `c5d.metal` node therefore would result in:

$$\begin{aligned} \text{Tot. Cost/month} &= \frac{\$4.608}{\text{h}} \frac{24\,\text{h}}{\text{day}} \frac{365\,\text{day}}{\text{year}} \frac{\text{year}}{12\,\text{month}} \\ &= \frac{\$3,363.84}{\text{month}} \end{aligned} \tag{2}$$

In order to make the On-Prem. vs. AWS cost comparison we need to take in account the time to perform the same amount of computational work. It would take the node `c5d.metal`, $\frac{877.07\,\text{s}}{1378.96\,\text{s}} = 0.64$ months to perform the same computational work (based on the HPL runs) as done by `notch139` in one month. The corresponding cost for the `c5d.metal` machine is therefore: $\frac{877.07\,\text{s}}{1378.96\,\text{s}} \times \$3363.84 = \$2,163.85$ (vs. $188.87).

The calculations above are based on Amazon's standard `On-Demand` pricing. They also offer discounts for long-term commitments. With a three-year contract Amazon's average discount[4] is 60 % off its `On-Demand` pricing. Therefore, the above number becomes: $0.4 \times \$2,163.85 = \$865.54$ (vs. $188.87).

V Conclusion

Based on the HPL runs the AWS approach is 11 times as expensive than the CHPC on-prem solution. With a three-year commitment it is still 5 times as expensive.

VI General Statements

The TCO project's goal is to compare the Total Cost of Running Jobs on On-Prem infrastructure (like the CHPC's) and jobs on the infrastructure of commercial Cloud Providers. In general, we have to make the following considerations:

- What are the optimal nodes (cost/benefit) to run the jobs on-prem and in the cloud. i.e. hardware
- What are good packages to tests certain specific scaling?
- What are the best compilation flags/modules to run a particular job. i.e. how do we get out most of a certain package on a certain node?

[2] Ratio which describes how efficiently a computer data center uses energy[10].

VII Multi-Node Parallel Simulations

Our goal is to check the use of the following packages:

- HPL
- LAMMPS (Molecular Dynamics)
- VASP (Ab initio Solid State package)

The On-Prem calculations have been performed on the Ash cluster. The used nodes were of the type `Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz` containing 24 physical cores and 264 GB of RAM running on a Centos 7 OS. The network infastructure was of the Infiniband type. The multi-node parallel calculations on the AWS have been performed on computational nodes of the type `c5n.18xlarge`. The latter nodes possess enhanced network capabilities i. e. Amazon's Elastic Fabric Adapter (EFA) [2] network interface. We started with a Centos7 image on the AWS cluster. Due to its really poor performance we switched to the Amazon Linux 2 OS [3] The codes which ran on the `C5n.18xlarge` nodes were compiled using the Intel Compiler Suite v. 2020.1 and Amazon's libfabric module.

VII.1 LAMMPS

VII.1.1 General

All LAMMPS simulations were performed using version $07Aug19$. The calculations were either run in a hyperthreaded mode (`HT`) i. e. running 2 MPI processes per physical core, or non-hyperthreaded mode (`NoHT`) i. e. running 1 MPI process per physical core. To run the simulations the following input files/systems were used:

- polymer: bead-spring polymer melt of 100-mer chains, FENE bonds and LJ pairwise interactions with a $\sqrt[6]{2}\,\sigma$ cutoff (5 neighbors per atom), NVE integration.
- metallic: metallic solid, Cu EAM potential with 4.95 Å cutoff, NVE integration.
- lj: atomic fluid, Lennard-Jones potential with $2.5\,\sigma$ cutoff (55 neighbors per atom), NVE integration.

Using the aforementioned 3 systems the strong and the weak-scaling were tested. The strong scaling curves (Fig. 1, 2, 3) were generated using a fixed size version for each system (864,000 atoms and 5000 integration steps).

In the weak-scaling situation (Fig. 4, 5, 6) the "unit" system (864,000 atoms) was scaled/replicated by the number of MPI processes and ran for 100 integation steps. Each simulation was ran for 6 times as well.

VII.1.2 Cost of the LAMMPS runs

The `c5n.18xlarge` nodes are rated at $3.888/h[3].

[3] We discussed the poor performance of the scaling of the LAMMPS simulations with the Amazon engineers. They discovered 2 bugs i. e.:

1. Kernel bug in RHEL7/Centos7.
2. Bug in Amazon Linux/RHEL/CentOS with EFA - we used this workaround:
   `sudo bash -c 'echo 5128 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages'`

and suggested the switch to the Amazon Linux 2 OS.

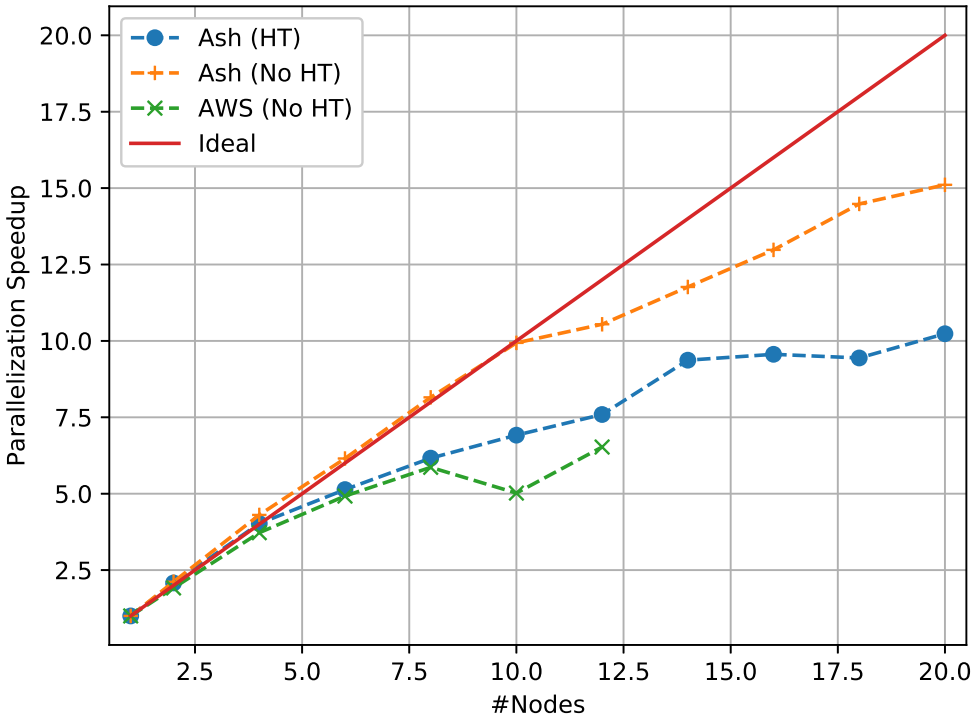

Figure 1: Strong scaling for polymer chain melt

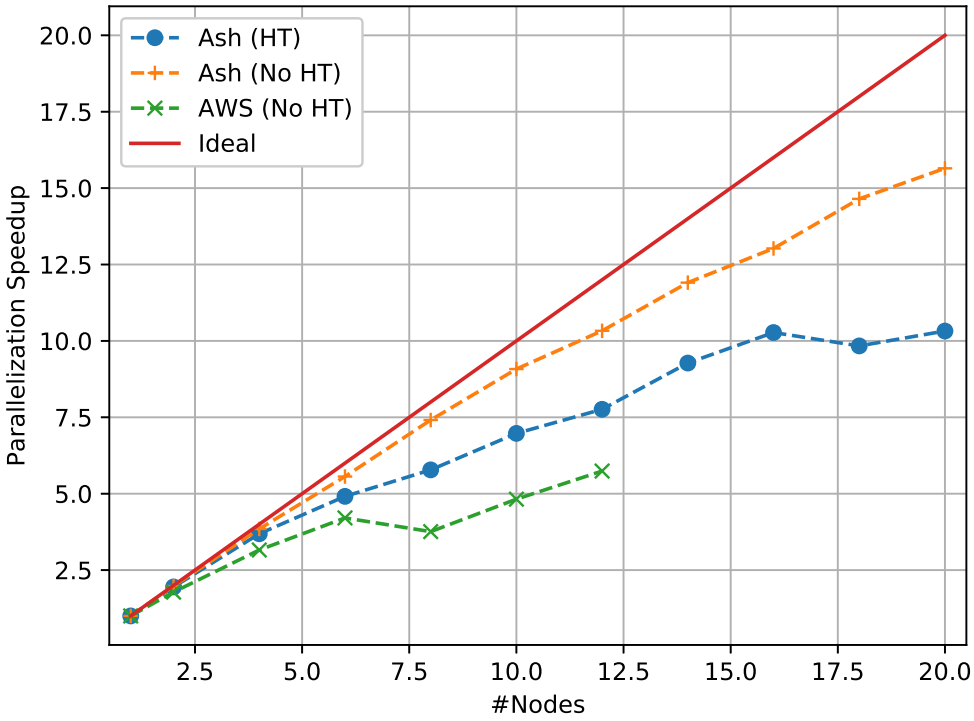

Figure 2: Strong scaling for EAM metallic solid

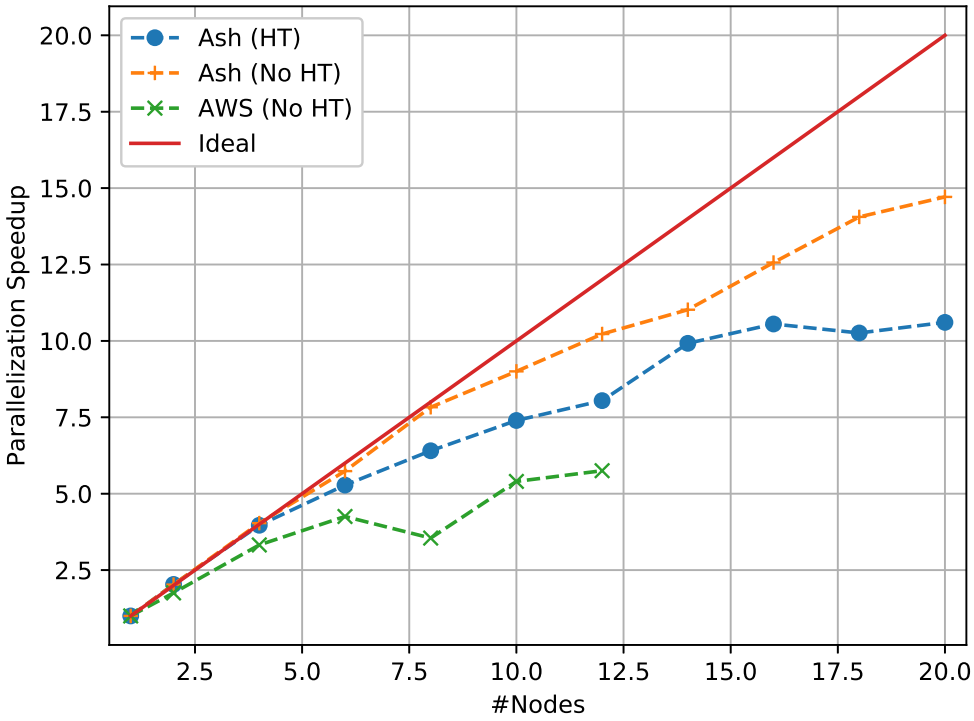

Figure 3: Strong scaling for Lennard-Jones atomic liquid.

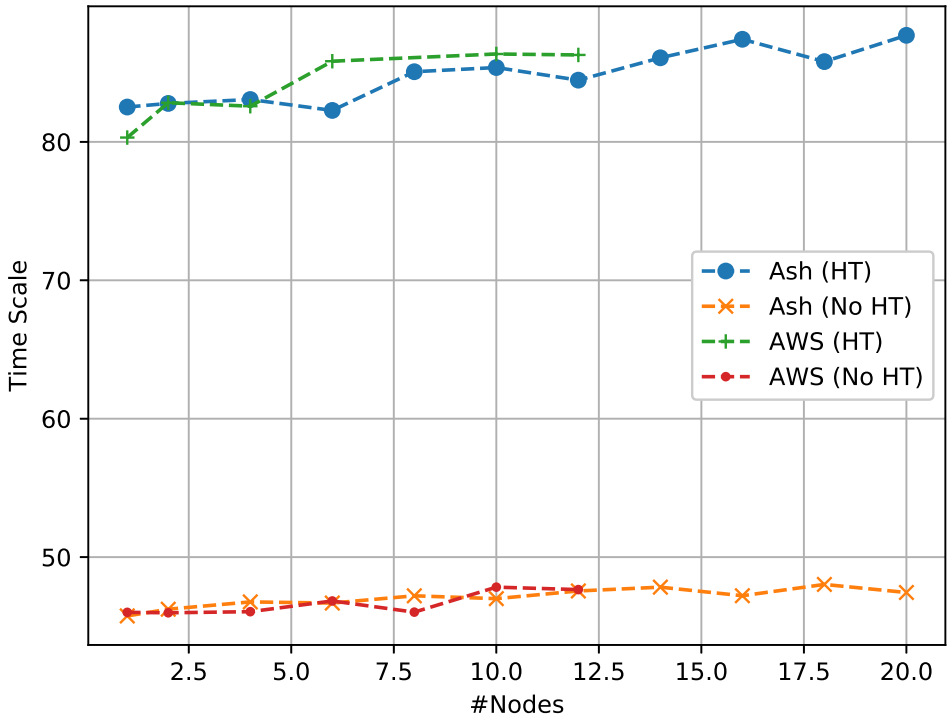

Figure 4: Weak scaling for polymer chain melt

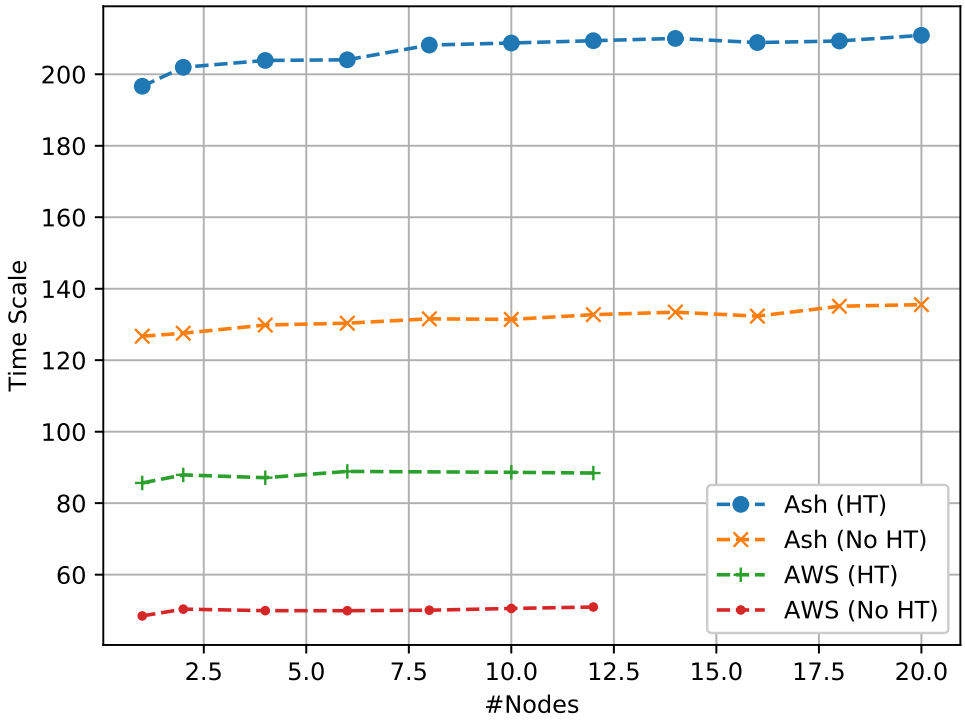

Figure 5: Weak scaling for EAM metallic solid

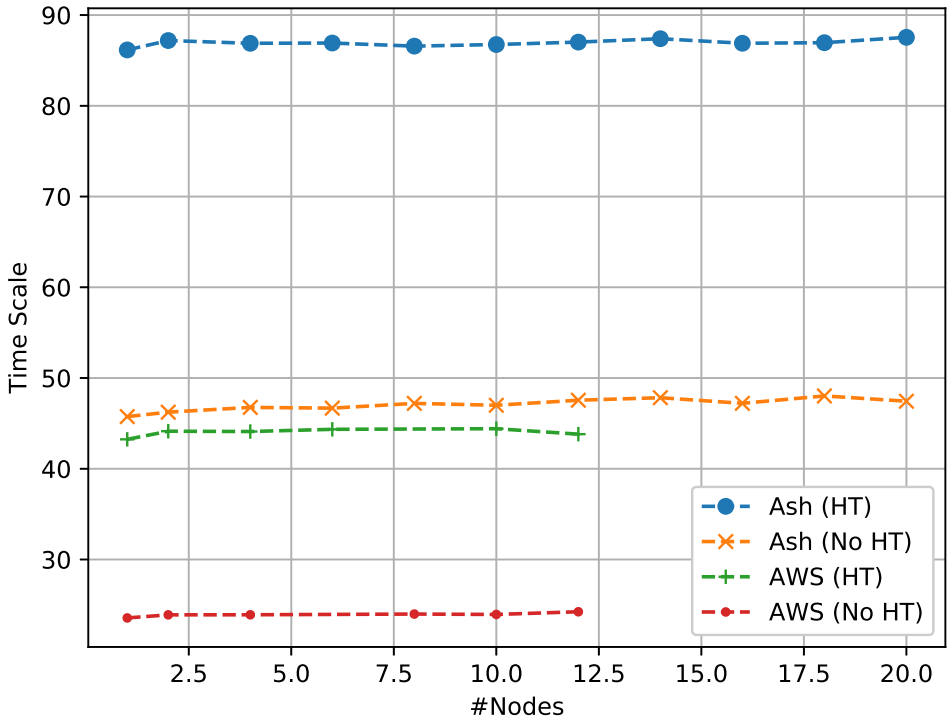

Figure 6: Weak scaling for Lennard-Jones atomic liquid.

VIII GPU Simulations

VIII.1 HPCG

The High-Performance Conjugate Gradients package (HPCG) [6], [5] was used to test out the performance of the GPU Devices. We used the HPCG 3.1 Binary [7] (the CUDA source not being available) which was compiled with GCC 4.8.5 on a Centos7 OS and required the presence of CUDA 10.0.130 as well as OpenMPI 3.1.x.

VIII.1.1 General

On the CHPC clusters we installed OpenMPI 3.1.6. The required version of CUDA was already installed.

On AWS we started with an existing AWS Parallel Cluster Image `ami-067d7a961841473e2` (Centos 7). We downscaled its existing CUDA version 10.2 to CUDA 10.0.130. We also installed OpenMPI 3.1.6 with libfabrics and CUDA support to generate a new image to run our new simulations.

On the CHPC cluster we ran the HPCG code 6 times on 2 types of GPU devices: `Tesla P100-PCIE-16GB` and `Tesla V100-PCIE-16GB`.

On AWS we ran the tests on a `p3.2xlarge` node 6 times. The type of node contains 8 logical CPUs and 1 GPU device (`Tesla V100-SXM2-16GB`). The `p3.8xlarge` nodes which have 32 logical CPUs and 4 GPU devices (`Tesla V100-SXM2-16GB`). The tests on the `p3.8xlarge` nodes were repeated 3 times.

In order to run valid simulations we had to run for at least 1 h. For all the runs a domain size of $256 \times 256 \times 256$ was chosen. The number of MPI tasks equals the number of used GPU devices.

VIII.1.2 Results

Node	Device Type	#Devices Used	Time/s	GFlop/s
`kp359`	`Tesla P100-PCIE-16GB`	1	3665.24	94.18
`kp360`		2	3671.14	199.73
`notch003`	`Tesla V100-PCIE-16GB`	1	3670.61	137.42
`notch001`		2	3678.45	290.34
`p3.2xlarge`	`Tesla V100-SXM2-16GB`	1	3661.14	138.25
`p3.8xlarge`	`Tesla V100-SXM2-16GB`	1	3661.10	138.18
		2	3661.81	291.03
		4	3660.04	575.28

Table 4: HPCG: Results for different nodes.

VIII.1.3 Pricing

The `p3.2xlarge` is priced at $3.06/h. The use of a node of the type `p3.8xlarge` is billed at $12.24/h. Both prices were recorded at 05/14/2020.

VIII.2 Amber

References

[1] AWS Portal. https://aws.amazon.com/. Accessed: 2020-01-03.

[2] Elastic Fabric Adapter. https://aws.amazon.com/hpc/efa/. Accessed: 2020-06-04.

[3] Amazon EC2 Pricing. https://aws.amazon.com/ec2/pricing/on-demand/. Accessed: 2020-01-03.

[4] Amazon EC2 Reserved Instances. https://aws.amazon.com/ec2/pricing/reserved-instances/. Accessed: 2020-01-23.

[5] M. Fatica. Optimizing the High Performance Conjugate Gradient Benchmarks for GPUs. https://devblogs.nvidia.com/optimizing-high-performance-conjugate-gradient-benchmark-gpus/. Accessd: 2020-06-04.

[6] M. A. Heroux and J. Dongarra. Toward a new metric for ranking high performance computing systems. 6 2013.

[7] HPCG 3.1 Binary for NVIDIA GPUs including Volta based on CUDA 10. https://www.hpcg-benchmark.org/software/index.html. Accessed: 2020-06-04.

[8] Intel Parallel Studio XE. https://software.intel.com/en-us/parallel-studio-xe. Accessed: 2020-01-03.

[9] A. Petitet, C. Whaley, J. Dongarra, A. Cleary, and P. Luszczek. HPL 2.3 - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers, December 2018. [Online; updated December 2, 2018].

[10] Power Usage Effectiveness. https://en.wikipedia.org/wiki/Power_usage_effectiveness. Accessed: 2020-01-03.

[11] J. Simon. Now Available: New C5d Instance Sizes and Bare Metal Instances. https://aws.amazon.com/blogs/aws/now-available-new-c5d-instance-sizes-and-bare-metal-instances/. Published: 2019-11-05.