



Investigations of Various HPC Benchmarks to Determine Supercomputer Performance Efficiency and Balance

Wilson Lisan

August 24, 2018

MSc in High Performance Computing

The University of Edinburgh

Year of Presentation: 2018

Abstract

This dissertation project is based on participation in the Student Cluster Competition (SCC) at the International Supercomputing Conference (ISC) 2018 in Frankfurt, Germany as part of a four-member Team EPCC from The University of Edinburgh. There are two main projects which are the team-based project and a personal project.

The team-based project focuses on the optimisations and tweaks of the HPL, HPCG, and HPCC benchmarks to meet the competition requirements. At the competition, Team EPCC suffered with hardware issues that shaped the cluster into an asymmetrical system with mixed hardware. Unthinkable and extreme methods were carried out to tune the performance and successfully drove the cluster back to its ideal performance.

The personal project focuses on testing the SCC benchmarks to evaluate the performance efficiency and system balance at several HPC systems. HPCG fraction of peak over HPL ratio was used to determine the system performance efficiency from its peak and actual performance. It was analysed through HPCC benchmark that the fraction of peak ratio could determine the memory and network balance over the processor or GPU raw performance as well as the possibility of the memory or network bottleneck part.

Contents

Chapter 1 Introduction	1
1.1. Report Organisation	3
Chapter 2 Background Overview	4
2.1. NVIDIA Volta Architecture and Tesla V100	5
2.2. Intel Xeon Scalable Processor (Skylake)	8
2.3. Obstacles and Deviation from the original plan.....	10
2.4. Final Selection of the HPC Systems for the Project	12
Chapter 3 Student Cluster Competition 2018 and Team EPCC	13
3.1. ISC-HPCAC SCC 2018 Rules	13
3.2. ISC-HPCAC SCC 2018 Benchmark.....	14
3.3. ISC-HPCAC SCC 2018 Awards.....	15
3.4. Team EPCC Cluster	15
3.5. Team EPCC Preparation and Hardware Issues.....	19
3.6. Troubleshooting at the SCC	20
3.7. Competition Conclusion	23
Chapter 4 HPL, HPCG, and HPCC	24
4.1. HPL NVIDIA Binary	24
4.1.1 The Binary	24
4.1.2 HPL Rules and Output.....	24
4.1.3 HPL Inputs and Parameters	25
4.1.4 HPL Benchmark on 1 Node for Maximum Performance	26
4.1.5 HPL Benchmark on 1 Node with 1500W Limit.....	30
4.1.6 HPL Benchmark on 2 Nodes with 3000W Limit	33
4.1.7 HPL Benchmark at the SCC	35
4.2. HPCG NVIDIA Binary	38
4.2.1 The Binary	38
4.2.2 HPCG Parameters, Rules, and Output.....	38
4.2.4 HPCG Benchmark on 1 Node.....	39
4.2.5 HPCG Benchmark at the SCC	39
4.3. HPC Challenge.....	40
4.3.1 HPCC Binary and Output	41
4.3.2 HPCC Parameters	42

4.3.3 HPCC Investigation on Old Boston Cluster	42
4.3.4 HPCC Modification and Porting to GPU	44
4.3.5 HPCC at the SCC.....	46
Chapter 5 Comparison of the Fraction of Peak Performance and System Balance..	49
5.1. Benchmark Comparison on Cirrus and Scafell Pike.....	50
5.1.1 Intel MKL Benchmark Background	50
5.1.2 HPCG Fraction of Peak over HPL.....	51
5.1.3 System Balance Analysis with HPCC	53
5.1.4 CP2K H2O-64 Comparison.....	59
5.2. Benchmark Comparison on NVIDIA P100 and V100.....	60
5.3. Comparison Conclusions and Future Probabilities	62
Chapter 6 Conclusion and Future Works.....	64
6.1. HPCG Fraction of Peak: The Future of HPC Hardware Evaluation.....	64
6.2. Student Cluster Competition: a Life-time Experience	64
6.3. Future Work and Recommendations.....	65
Appendix A Oddities of the KNL benchmarking at the ARCHER and Scafell Pike KNL Nodes	67
Appendix B Raw Benchmark Records – EPCC Cirrus and Hartree-Centre Scafell Pike	71
Appendix C Actual SCC Benchmark Scripts and Input Parameters	76
Bibliography	83

List of Tables

<i>Table 1 – P100 and V100 basic specification comparison.....</i>	<i>6</i>
<i>Table 2 – Performance comparison between V100 SXM2 NVLink and PCIe.....</i>	<i>8</i>
<i>Table 3 – Final selection of the HPC systems and tests</i>	<i>12</i>
<i>Table 4 – Cluster specification proposal specification</i>	<i>16</i>
<i>Table 5 – Cluster specification a day before the competition</i>	<i>16</i>
<i>Table 6 – Actual cluster specification during the competition</i>	<i>17</i>
<i>Table 7 – All software installed inside the cluster.....</i>	<i>19</i>
<i>Table 8 – Important HPL.dat parameters</i>	<i>25</i>
<i>Table 9 – HPL NVIDIA specific parameters</i>	<i>26</i>
<i>Table 10 – Starting point of HPL.dat parameters for the V100 peak performance test</i>	<i>26</i>
<i>Table 11 – Effect of CPU cores per GPU and HT</i>	<i>27</i>
<i>Table 12 – Performance investigation through different combination of Ns and NB</i>	<i>27</i>
<i>Table 13 – PFACT and RFACT investigation with Ns 122000</i>	<i>27</i>
<i>Table 14 – BCAST investigation with Ns 122000.....</i>	<i>28</i>
<i>Table 15 – SWAP investigation</i>	<i>28</i>
<i>Table 16 – Ns investigation for the peak performance of 6 V100s.....</i>	<i>29</i>
<i>Table 17 Starting point of HPL.dat parameters for the V100 power consumption limit test</i>	<i>30</i>
<i>Table 18 – Quick investigation of downclocking CPU and GPU clocks with NB 192, Ns 115000, and GPU_DGEMM_SPLIT 1.00.....</i>	<i>31</i>
<i>Table 19 – Comparison between NB 192 and 256 with clock 975 MHz and Ns 121344</i>	<i>32</i>

<i>Table 20 – Attempts to decrease the power consumption of NB 192. None of them hit below 1500W</i>	<i>32</i>
<i>Table 21 – DGEMM investigation with GPU clock 975 MHz, CPU clock 2.4 GHz, and NB 256</i>	<i>33</i>
<i>Table 22 – Five runs of NB 256, GPU clock 975 MHz, and DGEMM 0.95 begin from idle temperature between 36 – 37°C</i>	<i>33</i>
<i>Table 23 – Increasing Ns at 12 V100s with NB 192, DGEMM 0.95, and GPU clock 960 MHz.....</i>	<i>34</i>
<i>Table 24 – HPL results for 12 V100s with NB 256 and different DGEMM</i>	<i>34</i>
<i>Table 25 – HPL Configuration before the competition</i>	<i>34</i>
<i>Table 26 – Number of OMP threads and HT investigation over 9 mixed up V100s</i>	<i>36</i>
<i>Table 27 – Investigation of downclocking GPU clock with CPU_CORES_PER_RANK=4, CPU clock 2.4 GHz and DGEMM 1.0</i>	<i>36</i>
<i>Table 28 – Investigation of the highest Ns for the mixed-up 9 V100s</i>	<i>36</i>
<i>Table 29 – Using different GPU clock on both nodes to achieve higher performance with GPU temperature tweaking</i>	<i>37</i>
<i>Table 30 – Submitted HPL result at the SCC</i>	<i>37</i>
<i>Table 31 – Comparison of the HPCG result over 256³ grid size with different V100 configuration.....</i>	<i>39</i>
<i>Table 32 – Submitted HPCG result at the SCC</i>	<i>40</i>
<i>Table 33 – HPCC Benchmarks</i>	<i>41</i>
<i>Table 34 – HPCC website results</i>	<i>42</i>
<i>Table 35 – Results of HPCC benchmark on one node with Ns 75000 and additional 20000 for PTRANS</i>	<i>43</i>
<i>Table 36 – Comparison between one and two node performance in HPCC with OMP threads 4, Ns 95000 and additional 20000 for PTRANS.....</i>	<i>44</i>
<i>Table 37 – STREAM baseline versus optimised run at one node with 5 MPI processes and 4 OMP threads</i>	<i>45</i>
<i>Table 38 – Comparison of FFTE complete results between CPU and GPU version</i>	<i>46</i>

<i>Table 39 – Comparison between the result of the HPCC benchmark between E5-2630 v4 processors and E5-2650 v4 processors with similar MPI-OMP processes</i>	47
<i>Table 40 – Comparison between HPCC results at the SCC and the quick re-run with correct disabled HT</i>	48
<i>Table 41 – Homogeneous CPU systems for the comparison test</i>	49
<i>Table 42 – Heterogeneous GPU systems for the comparison test</i>	50
<i>Table 43 – List of the benchmarks in the Cirrus and Scafell Pike systems</i>	50
<i>Table 44 – List of the benchmark binaries that are used for the tests</i>	51
<i>Table 45 – HPCC results of the Cirrus system</i>	54
<i>Table 46 – HPCC results of the Scafell Pike system</i>	54
<i>Table 47 – Memory balance ratio between Cirrus and Scafell Pike system</i>	55
<i>Table 48 – Network balance ratio between Cirrus and Scafell Pike system</i>	56
<i>Table 49 – FFTE balance ratio between Cirrus and Scafell Pike system</i>	57
<i>Table 50 – CP2K results in the Cirrus and Scafell Pike systems</i>	59
<i>Table 51 HPL KNL benchmark result comparisons</i>	68
<i>Table 52 List of the HPCG results from various systems</i>	70

List of Figures

<i>Figure 1 – V100 full GPU architecture. It contains six GPU Processing Clusters (GPCs), each GPC has seven TPCs which each containing two SMs</i>	<i>6</i>
<i>Figure 2 – V100 Streaming Multiprocessor (SM)</i>	<i>7</i>
<i>Figure 3 – Three forms of V100 hardware spotted in the ISC 2018. (A) SXM2 for NVLink version (B) PCIe version (C) FHHL version.....</i>	<i>7</i>
<i>Figure 4 – Xeon Scalable motherboard from Supermicro X11 8U blade at the ISC 18 with dual LGA-3647 sockets</i>	<i>8</i>
<i>Figure 5 – Data center system from Lenovo displayed at the ISC 18 which uses Intel Xeon Scalable Platinum as its main processor.....</i>	<i>9</i>
<i>Figure 6 – Intel Xeon Scalable product numbering convention</i>	<i>10</i>
<i>Figure 7 – GPU NVLink node which contains six NVIDIA V100 NVLink version with silver passive heatsink</i>	<i>17</i>
<i>Figure 8 – Team EPCC cluster on Boston custom chassis</i>	<i>18</i>
<i>Figure 9 – The cluster has two grand nodes, each grand nodes contains separated CPU and GPU node</i>	<i>18</i>
<i>Figure 10 – Final GPU configuration on both nodes. N/D means Not Detected.</i>	<i>21</i>
<i>Figure 11 – Example of a valid HPL output which reads 19.81 Teraflops.....</i>	<i>25</i>
<i>Figure 12 – Comparison between 6 P100s and 6 V100s with the same RAM amount and processors</i>	<i>29</i>
<i>Figure 13 – Example of benchmark stdout of the NVIDIA HPL. Pay attention to the GF_per numbers, and also the time left where this benchmark will run for approximately 40 - 42 seconds.</i>	<i>30</i>
<i>Figure 14 – Power consumption usage during NVIDIA HPL run in the SCC</i>	<i>31</i>
<i>Figure 15 – Result of the HPCG inside the yaml output file</i>	<i>38</i>
<i>Figure 16 – Rankfile example to run the HPCG with mixed V100s</i>	<i>40</i>

<i>Figure 17 – Modification in the hpcc.c to replace the HPL 2.0 with HPL NVIDIA Binary.</i>	<i>45</i>
<i>Figure 18 – HPL results of the Cirrus and Scafell Pike systems across different number of nodes.....</i>	<i>51</i>
<i>Figure 19 – HPCG results of the Cirrus and Scafell Pike systems across different number of nodes.....</i>	<i>52</i>
<i>Figure 20 – Fraction of peak between Cirrus and Scafell Pike systems across different number of nodes</i>	<i>52</i>
<i>Figure 21 – Correlation balance between memory and processor performance.....</i>	<i>55</i>
<i>Figure 22 – Correlation balance between memory and processor performance.....</i>	<i>56</i>
<i>Figure 23 – Correlation balance between FFTE and processor performance</i>	<i>58</i>
<i>Figure 24 – Comparison of the CP2K speedup between the Cirrus and Scafell Pike</i>	<i>59</i>
<i>Figure 25 – Comparison of 1 node performance at the CP2K.....</i>	<i>60</i>
<i>Figure 26 – HPL results comparison of six P100s and V100s.....</i>	<i>61</i>
<i>Figure 27 – HPCG results comparison of six P100s and V100s</i>	<i>61</i>
<i>Figure 28 – Fraction of peak comparison of six P100s and V100s</i>	<i>61</i>
<i>Figure 29 – Comparison of the fraction of peak between all systems.....</i>	<i>62</i>
<i>Figure 30 – New Bull-Atos Sequana X1000 at the SCC 18. Now they are pairing the ARM processors with NVIDIA Tesla V100.....</i>	<i>62</i>
<i>Figure 31 HPCG KNL investigations at the Scafell Pike KNL one node.....</i>	<i>69</i>
<i>Figure 32 HPCG KNL 6 nodes results on the Scafell Pike KNL. N=256^3 failed to finish the benchmark.....</i>	<i>69</i>

Acknowledgements

First I would like to thank my amazing supervisor Fiona Reid for her total guidance, support, vast inspirations to improve my work into its new level, and willingness to proof-read my report. I would like to thank all of the Boston Limited and EPCC people at the SCC to keep our morale high during the chaotic hardware failures moment, especially Konstantinos Mouzakitis for the total supports.

I would also like to thank my family especially my father and mother for their pray and support that helped and motivated me to finish the dissertation.

At last, it was an honour to work with the best teammates and coach ever. Together we fought in the SCC and surely will carve my memory as the most unforgettable once in a life-time experience ever.



Team EPCC 2018

This dissertation work is dedicated to my amazing supervisors at Universitas Gadjah Mada, Indonesia who introduced me the world of high performance computing and once dreamed to study in the University of Edinburgh, Dr. Mardhani Riasetiawan.

And to my friends and colleagues at Laboratorium Sistem Komputer dan Jaringan UGM.

Chapter 1

Introduction

One of the main goals of the HPC world nowadays is trying to break the 1 Exaflop performance wall or over 1000 Petaflops which is dubbed as 'Exascale race' term. The Exaflop race began in 2008 when IBM Roadrunner broke the 1 Petaflops wall [1]. In the same year, the US DARPA programme summarized the requirements for 'ideal' Exaflop supercomputer [2]. One of the requirements is 1 Exaflop based from HPL (High Performance Linpack) benchmark with a maximum 66 MW power consumption and 200,000 core count. For comparison, the current fastest supercomputer, the IBM Summit, which has just replaced Sunway Taihulight is still at 187 Petaflops peak performance with two million core counts [3].

Regarding the Exascale race, there lies a fundamental problem which is the validation of so-called 1 Exaflop based on HPL. Jack Dongarra, the creator of HPL stated that HPL peak performance result is exaggerated and doesn't represent the real performance of parallel computing. There's no real parallel computation which could match the nearly-perfect parallel fraction from HPL. In scientific computing for example, 10 - 15% of peak HPL flops would be considered as good and effective coding result [4]. The next problem, is that the HPL benchmark is heavily loaded into processors and forgets about memory, network, storage, and I/O performance. A cluster with poor network and memory performance for parallel computing like a Raspberry Pi 2 or 3 cluster can still be able to gain good scalability result over different number of nodes with the HPL benchmark [5]. Meanwhile, supercomputers nowadays are suffering from a memory performance bottleneck if it is compared to processor and accelerator performance and that often limits the weak scaling on many scientific codes. That is not the problem for HPL benchmark since the number of memory operations on HPL is half of the floating point operations ratio [6]. An additional problem is that HPL benchmark is biased towards the number of processors and clock speed.

To replace HPL with a better benchmark that closely resembles actual computational power, Jack Dongarra created the HPCG (High Performance Conjugate Gradient) benchmark [7]. HPCG uses the PCG (Preconditioned Conjugate Gradient) which stresses not only processors but also memory and network performance. Nevertheless, the HPCG benchmark was criticized by numerous parties for its unrealistic benchmark results. Take one example at China's Sunway Taihulight with a huge 125 PFlops peak performance for HPL. Sunway Taihulight only has 481 TFlops for the HPCG benchmark. If HPCG is claimed to closely represent the actual performance, it means Sunway Taihulight can only utilize 0.4% from its peak performance, which is inefficient [8]. This phenomena leaves two possibilities for HPCG: the use of a fraction of performance between peak HPL and HPCG as a new parameter for supercomputer efficiency or there's something wrong with how the PCG algorithm measures flops performance on HPCG benchmark.

Meanwhile, DARPA has their own benchmark, not a single benchmark but a suite called the HPC Challenge (HPCC) [9]. It consists of seven different benchmarks with different roles to measure peak performance, scientific code performance, network, and memory. HPCC benchmarks are selected and assembled also by Jack Dongarra and Piotr Luszczek. Although HPCC is entirely composed of synthetic benchmarks, HPCC might be another good benchmark alternative to measure the efficiency and balance of supercomputer performance because it measures every important aspect of a parallel computer (apart from storage and I/O).

We may overlook the power efficiency between peak flops performance and power consumption, but performance efficiency is also an important aspect for supercomputer usage. Take an example from the future Exascale supercomputer. Are we walking in the right path to design the future supercomputer with the HPL benchmark as the goal? What if we ended up designing a supercomputer that has a poor fraction performance peak between HPL and actual code, or is difficult to use by various types of code? Nowadays, most scientific parallel codes and development are still favouring CPU computation rather than heterogeneous CPU-GPU computation which generates fantastic scores with the HPL benchmark (except for AI and deep learning computations). In the opposite point of view, a poor fraction of peak between HPL and other flops benchmarks may become a good indicator for the HPC industry to improve the architecture design of supercomputers hardware and parallel software.

This dissertation discusses the comparison between different benchmarks that are mentioned above over different machines with different architectures and generations. The first aim of this project is to augment the HPL benchmark with HPCG and HPCC. Then the result will be compared based on the fraction of peak flops between those benchmarks. Comparison of the fraction of peak between HPL, HPCG, and other flops benchmarks from HPCC will be the main focus for this project. There will be the head to head comparisons between CPU (homogeneous) machines and heterogenous (accelerator) machines with different architectures.

The second aim is to compare and balance of the HPC system with HPCC benchmark based on the network - memory benchmarks inside and benchmark correlation results between two related different benchmarks from HPCC and HPL - HPCG. In addition, a real scientific code benchmark named CP2K [10] is used to measure the system balance based on how well the machine scale over different number of nodes. All of the benchmarks used in this project will be tuned to explore the best performance for each machine. Because of several predicted obstacles, the selection of the machines will be described in Chapter 2 section 2.4.

Last but not least, the writer of this project involved in the Student Cluster Competition 2018 as a member of Team EPCC. The writer's area of responsibilities on the team are the HPL, HPCG, and HPCC benchmarks, hence give good opportunities to investigate and play with those benchmarks. This report will also contain the Team EPCC report through the preparation and performance at the SCC 2018. Apart from the writer's project, HPL is also included as the main explanation focus on this report because (as always) Team EPCC is aiming for the Best Linpack Award at the SCC.

1.1. Report Organisation

The report is composed in six chapters, as follows:

Chapter 1 contains background problems that inspire the writer's dissertation and sum up explanations for the project.

Chapter 2 describes the important background theories to be used as foundation for the project

Chapter 3 describes the SCC 2018 as well as Team EPCC's preparation and performance at the competition.

Chapter 4 presents the work of the writer (together with Team EPCC) on three SCC benchmarks which are HPL, HPCG, and HPCC.

Chapter 5 presents the comparison and analysis of different benchmarks on different platforms and various numbers of nodes for scalability testing.

Chapter 6 contains the conclusions of the project, together with future and recommendation works for next year's Team EPCC.

Chapter 2

Background Overview

The history of the supercomputers had been running together with parallel benchmarks to measure and rank the supercomputer. Probably it was Hans Meuer from Mannheim Supercomputer Seminar (now ISC Group) [11] who published the statistics of high performance computer rank since 1986 (vector computer era). Hans Meuer listed the machines based on its manufacturer. However, soon supercomputers saw the huge differences as public institutions and governments became aware of the functionality of the HPC for greater purposes. Each manufacturer began to produce the huge varieties, from low-end until high-end line-up, or specialized nodes. The parallel computing also advanced into MPP (Massively Parallel Processing) era.

In 1993, the Top500 group was founded in the same mind and objective with Hans Meuer: to form and update the statistics of the fastest supercomputer and rank. Top500 saw the rapid advancement of the supercomputer types and manufactures, moreover motivated them to design the new ranking method. Top500 decided to use the HPL (High Performance Linpack) benchmark as their main parameter to rank the supercomputer. HPL was selected because it is widely used and performance results are available to be compared in all relevant machines [12]. Hence HPL become one of the most popular parallel benchmarks and is used for many comparisons on supercomputer ranks and studies. The Top500 ranks the best 500 supercomputers around the world since it was first published list in June 1993. The rank became important when end-users started to use it as one of the main considerations when buying the machine that involves huge amount of budget and procurement time.

Nowadays, the next generation of supercomputers keep moving forward to deploy more cores, especially with large number of nodes and many-cores accelerator such as co-processor and GPGPU (General Purpose GPU). MPP machine have become bigger and bigger and 1,000,000 cores is ‘normal’ for a modern HPC machine. The architecture model is also slowly evolving from a CPU only system (homogeneous machine) into CPU-GPU machine (heterogeneous machine). All of those approaches easily turn the table of the HPL rankings with new fantastic results moreover if it’s backed with newer hardware architecture like duo NVIDIA Tesla V100 [13] and Intel Xeon Skylake [14]. Both of them drastically changes the Top500 ranking courses with better HPL score and efficiency. In the current top 10 Top500 list, six out of ten supercomputers use a heterogeneous architecture which produce high HPL benchmark result [3].

Over many years, supercomputer usage is always evolving into more sophisticated usage, practically to compute new technologies related. During this development phase, the HPC community began to acknowledge the weakness of the HPL and find new problems. The development of parallel software and libraries has become more complex than ever before to accommodate new kinds of simulation and scientific code

usage. Modern parallel applications are starting to depend more on memory operation, especially at irregular simulation model which use a temporal locality memory model. Unfortunately memory development is not as fast as processor and creates a performance bottleneck. Meanwhile, HPL fails to consider this problem due to the code design and the results keep growing up regardless of memory bottleneck problem (see Chapter 1). In short, actual scientific code nowadays behaves and performs very differently from synthetic benchmarks like HPL.

At first, the HPC Challenge which was made by Piotr Luszczek et.al was introduced by the DARPA as a benchmark suite to measure every important component of HPC hardware such as memory and network. HPCC was first introduced at SC 2006 in Tampa, Florida with a complete presentation to explain the benchmark [16]. The presentation itself came with the ideas to compare different benchmarks as performance correlation. One of the correlation group is HPL with another benchmarks that were designed to closely resemble the actual scientific program such as FFTE and DGEMM. In the following years, Jack Dongarra designed a better complete benchmark (not a suite) to augment HPL, which known as HPCG. The HPCG community began to augment the HPL benchmark with fraction of peak from the 2016 results list and onwards [17]. As for HPCC, although it is less popular than HPCG but still being be used at many Student Cluster Competition means HPCC is still considered as important benchmark for the HPC community.

2.1. NVIDIA Volta Architecture and Tesla V100

NVIDIA Volta microarchitecture is the successor of Pascal microarchitecture which was introduced in 2017. Unlike any previous NVIDIA instalments, Volta is only introduced on Tesla family for GPGPU and accelerator. Under the hood of Volta architecture, NVIDIA Tesla V100 was born with the codename GV100. V100 is also the successor of Pascal P100 and now is making its own way to fill up the next generation supercomputer hardware.

V100 offers some new features compared to P100. The main feature of the V100 is the new Streaming Multiprocessor (SM) which is optimized for deep learning and AI computation with a 50% improvement on performance and energy efficiency compared with the previous Pascal. The V100 uses a 12 nm FFN manufacturing process which provides the capability to have denser cores hence it has more cores inside. These new features allow the V100 to have a significant performance boost on both FP32 and FP64 performance. As for deep learning, the V100 has a specialized SM called Tensor Cores which offers 12x higher peak TFLOPS for training with a total of 125 Tensor TFLOPS for training and inference applications. V100 also uses the second-generation NVIDIA NVLink with a higher bandwidth speed (300 GB/sec compared to 160 GB/sec in the first generation). To reduce the memory transfer bottleneck between CPU and GPU, V100 uses newer and enhanced Unified Memory and Address Translation Service. NVIDIA cooperated with IBM in order to design those systems. Table 1 summarises the specifications of the V100 and the comparison with P100. Figure 1 describes the full architecture of V100 and *Figure 2* describes the SM architecture.

Components	Tesla P100	Tesla V100
GPU Architecture	GP100 Pascal	GV100 Volta
SMs (Streaming Multiprocessors)	56	80
TPCs (Texture Processing Clusters)	28	40
FP32 cores per SM / GPU	64 / 3584	64 / 5120
FP64 cores per SM / GPU	32 / 1792	32 / 2560
Tensor Cores per SM / GPU	N/A	8 / 640
GPU boost clock	1480 MHz	1530 MHz
Peak FP32 / FP64 TFLOPS	10.6 / 5.3	15.7 / 7.8
Peak Tensor TFLOPS	N/A	125
Texture Units	224	320
Memory Interface	4096-bit HBM2	4096-bit HBM2
Memory size	16 GB	16 / 32 GB
Transistors	15.3 billion	21.1 billion
Manufacturing process	16 nm FinFET+	12 nm FFN

Table 1 – P100 and V100 basic specification comparison



Figure 1 – V100 full GPU architecture. It contains six GPU Processing Clusters (GPCs), each GPC has seven TPCs which each containing two SMs. Source: [18]



Figure 2 – V100 Streaming Multiprocessor (SM). Source [18]

There are several types of V100 hardware form. As spotted in the ISC 2018, there are SXM2 V100, V100 PCIe, and V100 FHHL. *Figure 3* describes all of the possible hardware forms of the V100. The SXM-2 form is commonly used in HPC sector because of the NVLink enabled for higher interconnect bandwidth and faster performance than PCIe version. *Table 2* describes the comparison between V100 SXM2 NVLink and PCIe version. Despite of the difference of performance, V100 PCIe is still widely used because it has lower price together with the more common PCIe motherboard. Meanwhile V100 FHHL is the ‘small’ version of V100 PCIe with lower clock and power consumption (TDP 150 W). FHHL only takes one PCIe slot compared to PCIe version which takes 2 slots, and also uses low-profile bracket size.

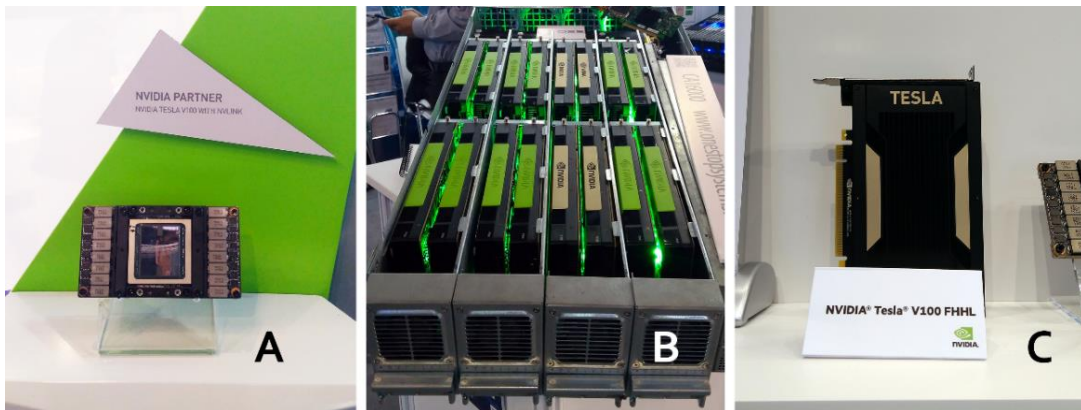


Figure 3 – Three forms of V100 hardware spotted in the ISC 2018. (A) SXM2 for NVLink version (B) PCIe version (C) FHHL version

Components	V100 SXM2 NVLink	V100 PCIe
Double Precision performance	7.8 TFlops	7 TFlops
Single Precision performance	15.7 TFlops	14 TFlops
Deep Learning performance	125 TFlops	112 TFlops
Interconnect bandwidth	300 GB/s	32 GB/s
TDP	300 W	250 W

Table 2 – Performance comparison between V100 SXM2 NVLink and PCIe

Following GeForce TITAN V, V100 comes with 16GB and 32GB VRAM variants. V100 32GB version is available in both SXM2 and PCIe forms. The 32GB variant is intended to provide higher performance for deep learning computations which has tendency to use more VRAM type memory. There is also a slightly improvement at synthetic benchmarks like HPL between V100 16GB and 32GB variant [19].

2.2. Intel Xeon Scalable Processor (Skylake)

Xeon Skylake, or officially named as Scalable is the answer from Intel to fight the AMD EYPC processors at server level. Intel Xeon Broadwell is becoming obsolete in term of performance. It was suggested that Xeon Scalable is to replace the previous Xeon Broadwell. Xeon Scalable is the ‘tock’ phase (Skylake generation) from Broadwell ‘tick’ (Haswell generation) therefore it offers a major improvement both in performance and architecture. Xeon Scalable was introduced in 2017 and currently the newer HPC machines are slowly beginning to use Xeon Scalable as their main processors both in homogeneous and heterogeneous systems. Many HPC companies in ISC 2018 displayed various Intel Scalable machines, like from Supermicro and Lenovo in *Figure 4* and *Figure 5*. Xeon Scalable is also a popular choice to pair NVIDIA V100 system from many HPC companies.



Figure 4 – Xeon Scalable motherboard from Supermicro X11 8U blade at the ISC 18 with dual LGA-3647 sockets

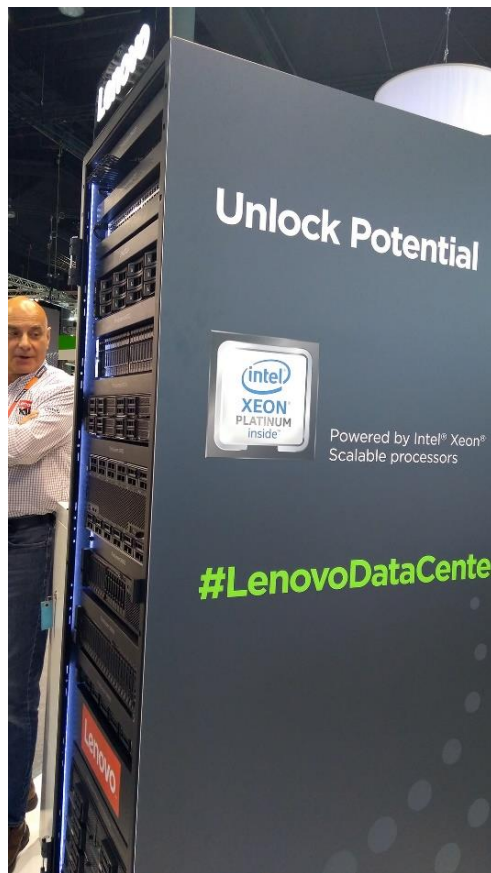


Figure 5 – Data center system from Lenovo displayed at the ISC 18 which uses Intel Xeon Scalable Platinum as its main processor

Xeon Scalable carries the feature of Intel Phi Knights Landing (KNL) Advanced Vector Extension 512 (AVX-512) with some improvements for HPC and big-data usage. The main purpose of AVX-512 is to accelerate performance on HPC workloads such as scientific simulation, data analytics, modelling, and deep learning AI. The applications can load both of 32 double precision and 64 single precision FLOPS within 512-bit length vectors and also up to two 512-bit FMA (Fused-Multiply Add) units. AVX-512 can also load the number of registers and FMA units twice that of the AVX-2 at Broadwell generation. Xeon Scalable is capable of using up to 4x10GbE Intel Ethernet to improve the transfer latency of large storage blocks, hence is well suited for big-data operations.

Xeon Scalable uses different naming convention system than the previous generation, as is shown in Figure 6. Rather than E3/E5/E7, Xeon Scalable is divided into four classes: Bronze, Silver, Gold, and Platinum. Bronze and Silver are equivalent with entry level Xeon E3 class which is used for mainstream HEDT usages such as office servers and data centres for mid-sized and growing IT institutions. Xeon Silver supports Intel hyper-threading (HT) and DDR4 2400 MHz memory. Xeon Bronze doesn't support Intel HT and can only use DDR4 2133 MHz memory. Both of them have 1 FMA per core and support up to 2 sockets per motherboard. The suffix names for Xeon Bronze and Silver are 3000 and 4000 series respectively.

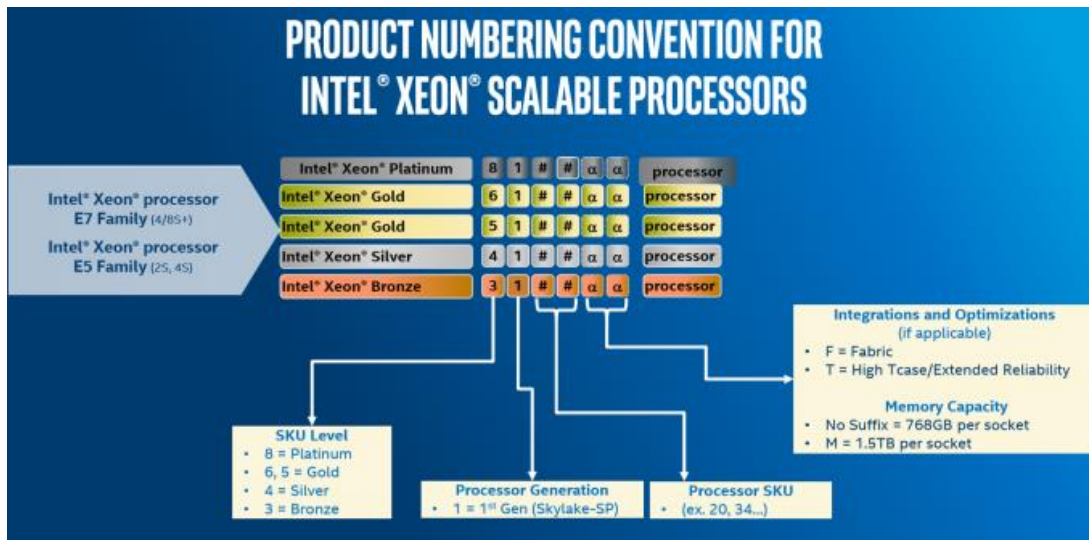


Figure 6 – Intel Xeon Scalable product numbering convention. Source [20]

Xeon Gold processors are equivalent to E5 and E7 class and they are suitable for large-scale HPC usage. Xeon Gold has 2 FMA per core which is likely to have twice the performance of Xeon Silver with the same number of cores. Xeon Gold processors support up to 2666 MHz DDR4 memory and 4 sockets per motherboard through 3 Intel UPI (Ultra Path Interconnect) links. The suffix for Xeon Gold is 5000 series (first generation and only supports 2 UPI links) and 6000 series (the improved version supports 3 UPI links). It is mentioned that if Xeon Gold 5000 processors with 2 UPI links are installed in quad-socket system, it will form a ring arrangement and therefore two hops are required if processor 1 wants to access processor 3 for example [20].

At last, Xeon Platinum holds the highest performance with the highest core count (up to 28 cores) and clock speed (up to 3.6 GHz). Xeon Platinum also has 48 PCIe 3.0 which is the highest among the Xeon Scalable families. Platinum supports up to 8 sockets per motherboard with 3 UPI links. Judged on the price, Xeon Platinum is intended to become the successor of Xeon E7 level, something like ‘E9’ class.

So far, Intel Platinum 8180 is the best Xeon Scalable class with highest number of core count (28 cores, 56 threads) and performance. In addition, Xeon Platinum 8180 broke the record of CPU clock in any Xeon family and 28 cores processor with 5 GHz overclocked at Taipei Computex 2018 exhibition with crunching number of Cinebench R15 benchmark score [21]. Although such of 5 GHz feat was achieved with 1000W sub-zero water-chiller cooling and 32 phases heavily-modified motherboard, Platinum 8180 shows the capability of newest Xeon Skylake architecture and opens the new possibilities to bring higher clock configuration into HPC world with good power efficiency.

2.3. Obstacles and Deviation from the original plan

All of the SCC related obstacles are described in Chapter 3 sections 3.5 and 3.6 give detailed explanation of the preparation phases and the competition. There were several deviations from the original plan. In this project, power consumption and power efficiency are not considered in order to focus only on the performance efficiency

investigation. The intended-machines for the project were also changed because of availability and benchmarking problems. The writer didn't see any possible attempt or way to measure the true power consumption in the final system selections.

As for the system selection itself, the original plan was to compare the Boston SCC cluster with Scafell Pike machine from the Hartree Centre because both of them has the Xeon Scalable Gold processor. In addition, EPCC's ARCHER KNL system will be compared too because it has different network layout (CRAY Aries) compared to others which use Mellanox Infiniband EDR. KNL nodes from ARCHER were chosen because it is commonly used in the HPC machines nowadays. All of the machines will run the full scale HPL, HPCG, HPCC, and CP2K benchmarks across the different numbers of nodes. The limit is 8 nodes since we can only use 8 nodes maximum at the ARCHER KNL.

About the Boston SCC cluster, the hardware specification was drastically diverted from the original hardware proposal, leaving the original project plan to compare it with Hartree Centre Scafell Pike system cancelled. Not to mention it was assembled again one month after the competition with only one node available and inadequate for the benchmarking tests which need at least 2 nodes for parallel benchmarking across the nodes.

As for the ARCHER KNL nodes, there were many oddities about the benchmark results. The benchmark results (both in the HPL and HPCG) were poor compared to the official result and last years dissertation who also benchmarked the ARCHER KNL [22]. Next attempt is to run the benchmark at Hartree Centre Scafell Pike KNL nodes. The result of the HPCG benchmark in the Scafell Pike KNL is better than the ARCHER KNL ones however the HPL performance is poor. It will be a waste to throw away the results of the ARCHER and Scafell Pike KNL benchmarks therefore all of the works there are recorded and explained in Appendix A.

At last, Cirrus from EPCC was decided upon in the place of the ARCHER KNL nodes. The idea to compare the Scafell Pike with other Xeon Gold system was changed into compare it with other system with same number of nodes which is 8 for this project. It is a coincidence that Cirrus uses Intel Broadwell processor, therefore it fits with Xeon Scalable from Scafell Pike as one Intel 'tick-tock' phase. Cirrus also uses different Infiniband interconnect from Scafell Pike system therefore meet the requirement to compare the systems for this project. There is also no official HPL and HPCG benchmark result on the Cirrus system either.

In addition, to augment the system comparison (which is only CPU based) with heterogeneous GPU system, NVIDIA P100 and V100 from the Boston SCC cluster will be compared to in terms of the fraction of peak. There was already the result of HPL and HPCG benchmarks at NVIDIA P100 and V100 during the competition preparation at the Boston SCC Cluster because we got an opportunity to use both GPUs.

2.4. Final Selection of the HPC Systems for the Project

Table 3 summarizes the final HPC systems selection and the benchmark test for this project.

Test	Platform
Comparison of HPL, HPCC, HPCG, and CP2K benchmarks	8 nodes of Hartree Centre Scafell Pike Xeon Gold with 8 nodes of EPCC Cirrus
Fraction of peak between HPL and HPCG only	6 NVIDIA P100s and 6 NVIDIA V100s from Boston SCC cluster

Table 3 – Final selection of the HPC systems and tests

Chapter 3

Student Cluster Competition 2018 and Team EPCC

The Student Cluster Competition (SCC) is a supercomputer competition between university students around the world. The goal of the competition is to build any cluster under 3 kW peak load and compete for the highest benchmark scores. The first SCC was held in the United States based Supercomputing Conference (SC) 2007. The next one is formed by the collaborative of ISC (International Supercomputer Conference) Group and HPCAC (HPC Advisory Council), known as ISC-HPCAC SCC. Meanwhile the newest one is Asian version of SCC, known as ASC (Asian Student Cluster Challenge). The three of them are known as 'Student Clustering Triple Crown' [23] and are held annually each year.

It has become a tradition for EPCC to participate in the SCC each year and participation along with additional work counts as a dissertation project for the students who join the team. This year, The University of Edinburgh under Team EPCC participated in the ISC-HPCAC SCC 2018 during the International Supercomputer Conference (ISC) 2018 in Frankfurt, Germany from the 25th until 27th June 2018. Our team is supervised by one coach and each member's dissertation supervisor. Our team is sponsored by London-based cluster vendor company Boston Limited [24] for hardware and technical support.

3.1. ISC-HPCAC SCC 2018 Rules

There are the rules used in the SCC 2018. They are mostly the same previous year rules but with slight changes and additions:

- The maximum peak power consumption is 3000W. Power consumption is monitored through a Power Distribution Unit (PDU) APC AP8588 [25]. All team PDUs are connected through ethernet cable and displayed online to be observed by the board committee and all teams. In addition there's also a small display in the PDU which also shows the power reading.
- If the power consumption exceeds 3000W, the PDU display will flash with a warning message and the board committee will come to the team and notify about it. Exceeding 3000W during the competition means penalized score. If the power consumption goes way over 3000W like 6000W for example, the breaker will trip and team will loss valuable time to boot up and start the benchmarks.
- In this year's SCC there is an additional rule: teams are allowed to exceed 3000W twice in the first day of the competition, after that penalized score will apply.

- The PDU power reading will refresh the readings each five seconds, hence ultimately gives all teams a 'little cheating' to go over 3000W during the five seconds interval as long as the PDU doesn't show the exceeded 3000W readings. It's possible in the HPL benchmark (further explanation will be carried at the fourth chapter about HPL investigations).
- Each team will be provided with two power budgets: PDU outlets and free outlets for another equipment. Anything related to the cluster should be connected to the PDU outlets. Meanwhile any laptops and monitors can use free outlets.
- No physically touch or hardware modification allowed to the cluster hardware once the competition start.
- No reboot allowed during the competition, it means no software modification that involves reboot like BIOS changing or NVIDIA ECC setting. Reboot is only allowed in significant condition like hardware crash / hang. The team has to notify the committee board to ask for permission to reboot.
- Like the other benchmark competition, teams have to provide the benchmark outputs and validations. Each teams is given a flashdisk by the board committee to collect the benchmark outputs there. The flashdisk also contains benchmark rules and output requirements, usually benchmark stdout, scores, readme, and software validations.
- Another special rules regarding of software library, Open MPI 1.10.3 until 1.10.7 are not allowed due to a bug time run which causes unrealistic benchmark score.

3.2. ISC-HPCAC SCC 2018 Benchmark

This year SCC uses the following benchmarks:

First day:

- **HPL, HPCG, and HPCC.** Note that teams can use any binaries (from CPU only into GPU) as long as it's not using Open MPI 1.10.3 until 1.10.7.

Second day:

- **Grid**, C++ library and framework developed by Peter Boyle from The University of Edinburgh and other contributors for Lattice Quantum Chromodynamics (Lattice QCD) [26]. The special input file is provided in the competition.
- **Nektar++**, Spectral/hp element framework used to the construction of high-performance scalable solvers of Partial Differential Equations [27]. The special input file is also provided during the competition.
- A Secret Application announced during the competition, which was **Nek5000**. Nek5000 is a computational fluid dynamics code that simulates fluid flow with thermal and passive scalar transport [28]. This application is highly scalable into large number of processors.

Third day:

- **AI application with Tensorflow.** This year Tensorflow computation is Deep Learning AI for image recognition model training with maximum accuracy within fixed 3 hours training time (this was announced at the competition).

3.3. ISC-HPCAC SCC 2018 Awards

There are five awards at the SCC this year:

- **Highest LINPACK Award**
The highest HPL score under power budget at the first day.
- **Fan Favorite Award**
To be given to the team which has highest count of unique votes of favorite fan.
- **1st, 2nd, and 3rd Place Overall Winners**
The main winners of the competition which is calculated by all benchmark scores except HPL with different weights: 10% HPCC, 10% HPCG, 15% Nektar++, 15% Grid, 15% Secret Application, 25% Tensorflow, and 10% team interviews by the representatives of the SCC board.

3.4. Team EPCC Cluster

The competition actually starts when each team designs the cluster specification. The hardware choice reflects the aim of the team for the competition. Our team selected the NVIDIA GPU-based cluster because of our aim is to achieve the Highest LINPACK Award and HPL is known to produce the best result with an NVIDIA binary. In addition, AI applications and HPCG will also intensively use the GPUs so a GPU-based cluster already gave us three benefits. Not to mention there's also many possibilities to port HPCC benchmark into GPU, for example linking HPCC HPL with NVIDIA binary and porting FFTE section into GPU with cu-FFT library.

At that moment there was a news about Nanyang National University (NTU) produced the LINPACK record at SC 2017 Denver with mind-blowing 51.2 TFlops under 3000W budget using 16 NVIDIA V100s [29]. After holding several meetings with our team coach, it was decided to use 12 V100s instead of 16 because at that moment we thought the regulation at SC 2017 Denver was average power consumption, not peak power consumption. 12 V100s was favoured over 16 V100s because sure it will involve underclocking but with reasonable clocks compared to 16 V100s. Two nodes configuration was also selected to compress the number of processors and keep power consumption as low as possible, so each node will contain six GPUs.

Grid and Nektar++ are CPU only applications and it seemed there was no way to port both of them to GPU, we decided to use powerful CPUs but with low Thermal Dissipation Specification (TDP) specs. Although our target is only the LINPACK award, we wanted to give a good fight for the other applications. Intel Xeon processor was selected because most Xeon motherboards supports NVIDIA NVLink which is

important for a GPU cluster. At that moment, Intel Xeon already had new architecture which was Skylake architecture under Xeon Scalable name. Xeon Scalable offers doubled performance than Broadwell in the same class with same power consumption. Xeon Gold was chosen because it hit the sweet-spot between high TDP Scalable Platinum and under-powered Scalable Silver. An early Xeon Gold review from Servethehome [30] showed the good result and power consumption of Xeon Gold compared to its counterpart AMD EYPC and power-hungry Xeon Broadwell 2699V4 class.

As for cooling we were agreed to go with liquid cooling because it had better cooling performance and more importantly fewer power consumption compared to traditional air cooling. Same with the previous teams, CoolIT was chosen. Together with another hardware, *Table 4* summed up our initial hardware proposal that was submitted to the board committee.

Two nodes in total. Each nodes contains:	
Hardware	Type
Processor	2x Intel Xeon Gold
Motherboard	Supermicro motherboard (any type)
RAM	128 GB DDR4
GPU	6x NVIDIA Tesla V100 16 GB HBM2
Storage	Intel Optane PCIe SSD
Network	Mellanox Infiniband
Cooling System	CoolIT DCLC AHx20

Table 4 – Cluster specification proposal specification

During our visits at Boston Limited HQ, London, their representative said that the team will go with old system Xeon Broadwell-E and air-cooling system only for unknown reasons. We ended up using the same processor of the previous team which was 10-core Xeon E5-2630V4 [22]. There were also another changes in the specs, including Intel P4600 NVMe SSD and upgraded NVIDIA V100 into 32GB version. The complete two nodes was only completed one day before the competition. *Table 5* describes the specifications that we used one day before the competition. *Figure 7* shows the GPU node only.

Two nodes in total. Each nodes contains:	
Hardware	Type
Processor	2x Intel Xeon E5-2630V4, 20 cores in total, 2.4 GHz
Motherboard	Supermicro motherboard
RAM	Micron 128 GB DDR4 ECC 2400 MHz
GPU	6x NVIDIA Tesla V100 32 GB HBM2 NVLink
Storage	Unknown 250 GB SSD for boot system Intel NVMe P4600 SSD 1 TB
Network	Mellanox Infiniband
Cooling System	Air cooling, 4 exhaust fans each CPU and GPU node

Table 5 – Cluster specification a day before the competition

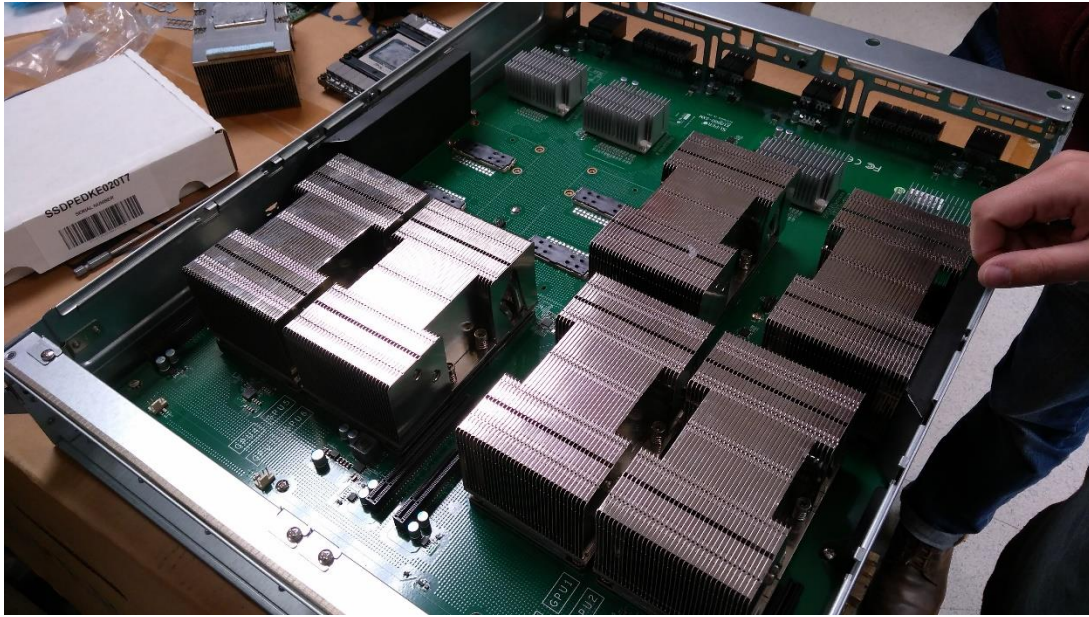


Figure 7 – GPU NVLink node which contains six NVIDIA V100 NVLink version with silver passive heatsink

On the first day of the competition, Boston Limited upgraded the processors into higher 12 cores E5-2650V4. Unfortunately the cluster suffered with so many hardware issues and human errors that will be explained further in the next section. The issues eventually changed the specs into asymmetric cluster and we had to notify the changes to the board committee. *Table 6* presents the final hardware specifications used at the competition. All of the hardware are installed on 2U rack size and Boston custom chassis. *Figure 8* and *Figure 9* show our cluster. We asked Boston Limited to plug out several cooling fans in order to reduce power consumption because the GPUs were not populating all of the slots, but it was not allowed.

Two nodes in total. Each nodes contains:	
Hardware	Type
Processor	2x Intel Xeon E5-2650V4, 24 cores in total, 2.4 GHz
Motherboard	Supermicro motherboard
RAM	Micron 128 GB DDR4 ECC 2400 MHz
GPU	Node 1: 5x NVIDIA Tesla V100 16 GB NVLink taken from NVIDIA DGX-1 Node 2: 2x NVIDIA Tesla V100 16 GB + 2x NVIDIA Tesla V100 32 GB
Storage	Unknown 250 GB SSD for boot system Intel NVMe P4600 SSD 1 TB
Network	Mellanox Infiniband
Cooling System	Air cooling, 4 exhaust fans each CPU and GPU node

Table 6 – Actual cluster specification during the competition



Figure 8 – Team EPCC cluster on Boston custom chassis



Figure 9 – The cluster has two grand nodes, each grand nodes contains separated CPU and GPU node

As for the software itself, *Table 7* presents all of the software, libraries, and compilers installed inside the cluster. Several system admin works were done to set up the cluster:

- SSH server to remote the cluster.
- Each team member has their own account and home directory plus dedicated root from Boston Limited.
- Module environment for each MPI, compilers, and libraries.
- Passwordless SSH and sudo for each nodes and accounts.
- Shared directory using CentOS NFS on team member home directories as well as /opt and /usr for the essential libraries and compilers.

Operating System	CentOS 7.5
MPI	Open MPI verbs (for Infiniband) 1.10.2, 3.1.0 Intel MPI 17.0.0, 17.0.4, 18.0.0 Optimized MPICH-3.2.1
Hardware drivers	NVIDIA graphics driver 396.26 Mellanox Infiniband
Compilers	GNU 4.9.3, 8.1.0 Intel Compilers 17.0.0, 17.0.4, 18.0.0 PGI 18.4
Libraries	<ul style="list-style-type: none"> • For GPU benchmarks: NVIDIA CUDA 9.0, 9.1, 9.2 • General benchmarks: Intel MKL 17.0.0, 17.0.4, 18.0.0 • Nectar++ only: Cmake 3.10, Boost 1.65, HDF5, PETSc, Scotch • Grid only: GMP, MPFR, C_LIME
Shared directory	CentOS NFS
Misc software	Screen
Monitoring	htop for hardware and OS usage monitoring PowerTool for power consumption monitoring

Table 7 – All software installed inside the cluster

3.5. Team EPCC Preparation and Hardware Issues

First of all, the cluster was in Boston Limited HQ London, it meant we had to remote login to the cluster from Edinburgh through SSH. Before our visit to London 2 weeks before the competition, we were provided with single test node containing dual Xeon E5-2630V4 (20 cores in total) and six NVIDIA P100 NVLink. All of the software and benchmarks installation plus optimisations were done based on this hardware. Last year Team EPCC's dissertation [22] was used as the result parameter to make sure everything was on the right track. In addition, several HPC companies testing results from DELL and Fujitsu were also used to compare the performance of the processors [31].

The second phase of preparation was where Boston Limited installed the NVIDIA V100, first with 16 GB NVLink variant. The V100 has different specification with P100 so learning the data sheet and white paper is necessary. It was proved in the HPL benchmark for example that V100 uses different parameter configurations to P100 (will be explained further in Chapter 4). At this moment, most of the problems were only incorrect parameters in the benchmarks which gave us opportunity to analyse the benchmarks. The most important preparation was our two days visit to Boston Limited HQ London. Most of the works done in London were system admin related and setting up the two nodes with an Infiniband connection. The end result, we were able to run CPU only benchmarks on two nodes: Grid, Nektar++, Intel MKL based HPL, and HPCC. GPU benchmarks were not be tested yet because the second node had Tesla P100 and at that moment we didn't know that Open MPI verbs was needed to distribute the work within Infiniband network (NVIDIA HPL and HPCG use Open MPI).

The third phase and the real problem started when Boston Limited changed all of the GPUs into V100 32 GB variant on both nodes. The GPUs ran fine in the first node, but failed on the second node. The first symptom was only four or five out of six V100s were detected in the second node. Second, it took a long time to run `nvidia-smi` command (to see the NVIDIA installed GPU status). Linux `lspci` command (to detect all of the PCIe installed hardware) did detect all of the GPUs but the driver environment and benchmarks couldn't. These problems were (temporary) fixed in just one day before Boston Limited packed the cluster to be shipped to Frankfurt. In one long night we rushed all of the GPU benchmark testing on 12 V100 GPUs especially HPL which was our priority. Unfortunately the second node couldn't detect all of the GPUs again after the HPL testing finished, leaving Tensorflow, HPCG, and modified HPCC with NVIDIA binary benchmarks untested for two nodes running.

3.6. Troubleshooting at the SCC

We arrived at Frankfurt on Saturday to set up and fix the hardware issues together with Boston Limited (The competition started on the Monday). Hardware troubleshooting was eventually initiated when all of the system admin approaches didn't solve the problems. Our thoughts were that some broken PCIe slots in the GPU motherboard. The only way to troubleshoot it was remove all the GPUs one by one and swap it to all slots to locate the defect slots.

In such a critical phase, an unbelievable fatal human error mistake was occurred on the hardware. There was one GPU heatsink stuck because of a failed screw in the second node. We forcefully removed the screw with pliers, but ended up scratching the GPU motherboard right in the NVLink path. This small scratch killed half of the first GPU Island (two GPU slots) including the PCIe slots. It became worse when we knew the dead slots were not the defect ones, thus leaving only four working slots in the second node. Because of broken PCIe slots, the NVMe SSD and Infiniband network card were moved to the CPU motherboard.

As if not enough, on Sunday the first node showed the same symptoms with the second node. Gradually it only detected five out of six GPUs and ended up with only recognized three GPUs. Boston Limited finally carried out the very last attempt to fix the problem by took out all of eight V100 16 GB GPUs from NVIDIA DGX-1. We replaced all of the V100 32 GB GPUs from the first node with V100 16 GB GPUs from DGX-1. It worked and the OS detected the whole eight V100s. We finally thought that the motherboard is not meant to be paired with V100 32 GB. Now we had 10 V100s and intended to install five GPUs on each node and make it symmetric in term of GPU numbers. We took the risk to mix up V100 16 GB with detected 32 GB.

We knew the second node had two dead slots and two broken slots, but we wanted to test the broken slots with V100 16 GB. The second round of swapping GPU was initiated. In this step, the second human error mistake occurred. Probably due to pressured conditions and rushed installation, the pin socket in one of the V100 16 GB was bent and made it undetected on any working slots. At the end, there were only 9 GPUs available with mixed and asymmetric configuration as is described in *Figure 10*.

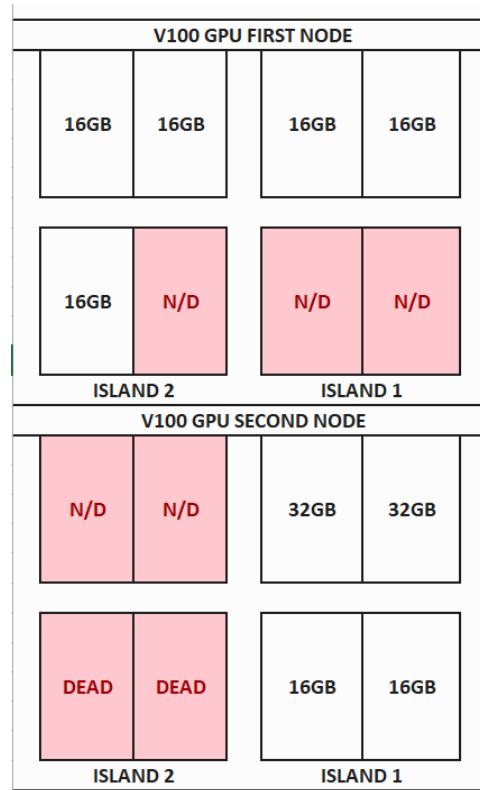


Figure 10 – Final GPU configuration on both nodes. N/D means Not Detected.

However, the show must go on for the competition was our priority. All of the HPL, HPCG, and HPCC testing with new hardware configurations were done on Monday morning, approximately six hours before the competition was commenced. HPL, which was our priority took most of the time to tweak the parameters and produced highest result below 3000W. The benchmark couldn't be run directly by the current script because of the different numbers of GPU on each node. The run scripts were

heavily modified (in such short time and tension) to make the HPL run. The second was HPCG with our very first run on two nodes. The run script was also modified especially as HPCG needs an older OpenMPI version which had different features with the OpenMPI used in the HPL benchmark. In addition, there was also several problems with the mpirun regarding the Infiniband network but we quickly managed to solve the problem. All of these work will be explained in the Chapter 4.

The last testing was the HPCC with NVIDIA-modified version. Before the competition started, we only tested the modified HPCC on one node because there was no time and the unavailability of the second node. It backfired our HPCC testing run and the benchmark failed to run on two nodes. With only a little time remaining, it was decided to run the HPCC in processor only and used the same parameters from the E5-2630V4 on E5-2650V4 processors except for the number of processes because the new processor has more cores.

After the competition started, the HPL, HPCC, and HPCG benchmarks were run. The HPL was only run once because it already hit the limit of 3000W and was considered as our best run. The next step was to run the rest of the benchmarks first, then came back to the HPL benchmark and rerun it with ‘suicidal’ clock, knew the committee allowed all of the teams to go over 3000W twice. HPCG was also run only once because it needed 30 minutes to finish and it was already the maximum performance for 9 GPUs (in comparison, the 12 V100s could run the HPCG at full clock for below 3000W).

Unexpectedly, the real problem lay in the HPCC benchmark. Although we had decided to run the HPCC with CPU only, we tried to fix the NVIDIA-binary version. There was a mistake in the NVIDIA HPL script section that made all of the GPUs to run at full clock. The power consumption rose into 3400W and we breached the first over 3000W opportunity. We urged to fix the benchmark, completely forgot to rerun the HPL benchmark and decreased the competition remaining time into one hour. As there were no possibilities left to run the HPCC NVIDIA version, the standard CPU version was executed. This was where the unexpected problem struck. HPCC with four Xeon E5-2630 v4 processors needed 20 – 21 minutes to finish, but the new and untested E5-2650 v4 needed almost 50 minutes to finish. The number of MPI processes should be correct to utilize all cores. The results of the HPCC benchmark was indeed overall faster than E5-2630 v4 but with awful result at the HPL. We almost failed to submit the HPCC benchmark when we had to prepare the desired output results and validations within 10 minutes remaining time.

The second day was executed with CPU only benchmarks which were Grid, Nektar++, and Nek5000 as the secret application. We were only able to submit Nektar++ because of some reasons:

- The input files provided by the committee had huge problem sizes which were incapable to be run by our cluster processors. For example, four Intel Xeon E5-2650V4 needed almost two hours to finish the Nektar++ benchmark with given input parameters.

- Somebody (or maybe visitors) accidentally unplugged the switch that was used to connect our laptop into the cluster, ultimately killed the benchmark run and we lost valuable time for the competition.
- The cluster storage couldn't accommodate all of Grid's benchmark outputs.
- There was no time left to run Nek5000 application although we managed to compile and run it on the cluster. We found out Nek5000 couldn't be compiled with Intel MPI and we used Open MPI 3.1.0 instead.

The third day was AI application only. First we tried to make the Tensorflow run on two nodes but it always stuck at warm-up stage. During the actual competition, the Tensorflow was only run on the first node which was five GPUs only. The benchmark finished with mediocre result, knowing the performance boundaries of five GPUs only although in overclocked mode.

3.7. Competition Conclusion

The aftermath was we didn't win any awards, however we were recognized by the other teams for our efforts to run all of the benchmarks with half-broken and asymmetrical hardware cluster especially in the HPL benchmark. We were also the only team that disassembled the hardware in the competition preparation. Unfortunately this year's SCC only mentioned the winners of the each award without stating benchmark rankings and results.

Perhaps the most important discovery from our works at the SCC is the way to run the HPL and HPCG NVIDIA binaries in such an asymmetric and mixed V100 GPUs. Our works also demonstrated how OpenMPI and the benchmark binaries could adapt with an asymmetric hardware configuration and run the binary properly. The last are several unique hardware tweaks like mixed GPU clocks and the importance of idle temperature in an air-cooling cluster to maintain the power consumption.

The highest power consumption for the submitted run was 2970W recorded in the PDU. The highest power consumption ever recorded in the testing run was 3600W, including the mistake run during the first day of the competition which was 3400W. The rest of the benchmarks ran below 3000W, with the highest achieved at Tensorflow run with 2500W on five overclocked GPUs. It's a great challenge to keep the power consumption as we used air cooling system and the ambient temperature inside the hall always changes depending on the time, weather, and visitors.

Chapter 4

HPL, HPCG, and HPCC

4.1. HPL NVIDIA Binary

HPL is a parallel distributed memory benchmark to measure the floating point operation per seconds. HPL is designed by Jack Dongarra, et al as the portable implementation of the previous Linpack benchmarks. HPL solves a random, double precision (IEEE 754 64 bits) $A(x) = B$ linear system with LU factorization with row partial pivoting containing multiple look-ahead depths [32]. In addition, various broadcast and swapping algorithms are used inside the benchmark. The number of operations are of the order of: $2n^3 / 3 + 2n^2$.

4.1.1 The Binary

For the SCC, we used the GPU-enabled, NVIDIA HPL binaries for NVIDIA Pascal and Volta GPUs since we used V100. The binary was strictly given from Boston Limited and NVIDIA. Normal users cannot obtain the HPL NVIDIA binary because it can only be given from NVIDIA itself and used for:

- Official benchmarking for Top500 ranking submission.
- Official benchmarking for official report.
- The Student Cluster Competition, through the sponsor or university.

There are six NVIDIA binaries available inside the package. All of them are already compiled with various OpenMPI and linked with various CUDA, as follows:

- OpenMPI 1.10.2 with CUDA 9.0
- OpenMPI 1.10.2 with CUDA 9.1
- OpenMPI 1.10.2 with CUDA 9.2
- OpenMPI 3.1.0 with CUDA 9.0
- OpenMPI 3.1.0 with CUDA 9.1
- OpenMPI 3.1.0 with CUDA 9.2

To accommodate all of the binaries, OpenMPI 1.10.2, 3.1.0, CUDA 9.0, 9.1, and 9.2 were installed.

4.1.2 HPL Rules and Output

There are two rules for the HPL benchmark in general:

- Use IEEE 754 double precision 64 bit operation [2]. No modification in the code which changes the precision.
- The output should pass the residual check for the accuracy as shown in *Figure 11*.

By default, most of the HPL binaries only display the output with stdout, hence the users should define an output file to save the stdout inside the running script. *Figure 11* describes the example of the HPL output with passed residual check.

```
=====
T/V          N    NB    P    Q          Time          Gflops
-----
WR01L2C8     90000  384    2    3          24.53          1.981e+04
2018-06-06 00:45:00.513
-----
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=      0.0034468 ..... PASSED
-----

Finished      1 tests with the following results:
              1 tests completed and passed residual checks,
              0 tests completed and failed residual checks,
              0 tests skipped because of illegal input values.
-----

End of Tests.
=====
```

Figure 11 – Example of a valid HPL output which reads 19.81 Teraflops

4.1.3 HPL Inputs and Parameters

An input file named HPL.dat is available to provide the parameters for the benchmark. *Table 8* summarizes the important parameters inside HPL.dat which visibly affects the benchmark performance.

Parameter	Explanation
Ns	The order problem size for the linear system $Ax = B$. Right Ns size (not too small or big) will bring the benchmark into its peak performance. Later we found out the good Ns should be able to be divided by NB, P, and Q
NB	NB decomposes the Ns into block size $NB \times NB$. This is also a very important parameter which is greatly affects the HPL performance.
P, Q	The parameter of two-dimensional block-cyclic distribution. The rule is $P \times Q$ should equal with the number of the processes. For example if we run the HPL with 8 processes, it leaves four possibilities for P and Q: 1-8, 8-1, 2-4, or 4-2. Each of them will produce different performance.
PFACT - RFACT	Panel Factorisation and Recursive Factorisation. Both of them have three variants: left, crout, and right.
BCAST	Broadcasting algorithm after PFACT performed. There are six variants available. Depends on the system, the right choice of the BCAST will boost the HPL performance.
SWAP	The algorithm for broadcast operation of U. There are three variants: bin executable, long, and mix.

Table 8 – Important HPL.dat parameters

Meanwhile there are other specific parameters for the NVIDIA binary only where all of them are defined in the running script (see Appendix A). *Table 9* summarizes the HPL NVIDIA parameters.

Parameter	Explanation
CPU_CORES_PER_RANK	This is a variable defined in the running script from NVIDIA to specify the number of OMP and MKL threads per GPU, or number of CPU cores per GPU.
GPU_DGEMM_SPLIT	Split ratio of LAPACK's DGEMM between CPU and GPU execution. Ratio 1 means 100% workload into GPU meanwhile ratio 0 into CPU. Another example, 0.95 means 95% GPU and 5% CPU.
CUDA_DEVICE_MAX_CONNECTIONS	Number of GPU connections allowed.

Table 9 – HPL NVIDIA specific parameters

4.1.4 HPL Benchmark on 1 Node for Maximum Performance

At this moment, the SCC cluster only had six V100s whereas the second node had six P100s. Therefore our first objective was to search the maximum performance of the six V100s and set up the proper of HPL.dat parameters for the rest of the benchmarking. The processors at this moment was dual Xeon E5-2630v4 which have 20 cores / 40 threads. HPL.dat parameters of the previous team was used [22] (which is designed for the P100s) as the first baseline but with some adjustments at the Ns, P, and Q for six GPUs. The important parameters are summarized in *Table 10*. To ensure the maximum performance, V100 turbo boost clock was used with GPU_DGEMM_SPLIT = 1. The binary OpenMP 1.10.2/CUDA 9.0 was used for the following tests.

Parameter	Value
Ns	110000
NB	384
P-Q	2-3
PFACT – RFACT	0-0
BCAST	1
SWAP	1

Table 10 – Starting point of HPL.dat parameters for the V100 peak performance test

The first investigation was to find the correct number of CPU cores per GPU with changing the CPU_CORES_PER_RANK parameter. Normally the number of cores per GPU should follows $[\text{number of GPU}] \times [\text{CPU_CORES_PER_THREAD}] \approx [\text{number of CPU cores or threads}]$ rule. The effect of the Hyper-Threading (HT) will be tested as well. *Table 11* presents the result of the tests.

CPU_CORES_PER_RANK	HT	TFlops	Wattage (Watt)
5	ON	24.69	2260
6	ON	25.89	2251
7	ON	18.87	2100
3	OFF	25.68	2201
4	OFF	16.85	1928

Table 11 – Effect of CPU cores per GPU and HT

The best performance results were achieved with 6 CPU cores per GPU with HT on (36 threads out of 40 threads) and 3 CPU cores per GPU with HT off (18 cores out of 20 cores). As expected when the number of CPU cores per GPU exceeds the CPU cores/threads count, the performance will horribly fall. 3 CPU cores per GPU-HT off has the same performance with 6 CPU cores per GPU-HT on but with lower power consumption and thus it was selected.

The next attempt was to find the correct value of NB, because 25.68 TFlops was considered as a quite poor result for six V100s (they were expected to have at least 27 or 28 TFlops). We found out that all of the V100s didn't work 100% when running the HPL through nvidia-smi monitoring. It should work nearly 100% all the way during the HPL initial run just like the P100s that we have tested before. The suspects was incorrect NB and Ns values. We tested the HPL with different combinations of NB and Ns values. The results are described in *Table 12*.

NB Ns	100000	110000	120000	120500	121000
128	25.95	26.76	27.72	27.32	27.54
192	26.2	27.32	28.12	28.13	28.36
256	25.2	26.58	27.53	27.8	27.41
384	23.29	25.43	26.5	27.17	27.12
512	19.25	21.37	23.15	22.94	23.23

Table 12 – Performance investigation through different combination of Ns and NB

NB 192 produced the best result meanwhile the performance with NB higher than 384 began to reduce, meaning setting NB to 640 or higher was out of the question. Nvidia-smi monitoring showed the V100s ran at nearly 100% load with NB 192. This test indicates each NVIDIA GPU has its own ideal NB on the HPL benchmark, for example P100 needs NB 384 for peak performance and older K40 with NB 1024 [34]. The next investigation was the PFACT and RFACT values. Usually we paired both of them with the same values. The results are presented in *Table 13*.

PFACT and RFACT	TFlops
0 (left)	29.41
1 (crout)	29.44
2 (right)	27.55

Table 13 – PFACT and RFACT investigation with Ns 122000

PFACT-RFACT with left and crout mode have similar performance, on the contrary right mode has lower performance. Because of the slightly higher result, crout mode

was selected. The next investigation is the BCAST which is also tends to visibly affect the HPL performance. *Table 14* summarizes the result of BCAST tests. BCAST 1 Ring modified, 2, and 2 ring modified give the best result from others.

BCAST	TFlops
0 (1 Ring)	26.90
1 (1 Ring modified)	29.41
2 (2 Ring)	29.42
3 (2 Ring modified)	29.40
4 (Long)	26.46
5 (Long modified)	28.92

Table 14 – BCAST investigation with N_s 122000

In practice, the three best BCAST tend not to give a significant difference and so we can assume all three of them have similar performance. We ended up selecting the 2 Ring BCAST. The last parameter to be investigated was the SWAP. There are three choices: bin exec, long, and mix. In the V100, all of the SWAP choices have similar performance as is shown in *Table 15* and thus we stuck with the long SWAP choice.

SWAP	TFlops
0 (bin exec)	29.36
1 (long)	29.41
2 (mix)	29.38

Table 15 – SWAP investigation

Apart from N_s , we finally had the best parameters which will be used for the rest of the performance investigation. The other parameters apart from what we tested before don't have any performance impact but rather than to keep the benchmark running without error and to ensure the validity of the benchmark result.

The last important parameter to improve the result is the right problem size or N_s . If the N_s is too little, the system will not produce its peak performance. If the N_s is too large, the performance will decrease. The ideal amount of theoretical N_s in NVIDIA HPL should fill about 80% of the system DRAM and GPU VRAM memories, as is summarized by the equation:

$$N_s = \sqrt{\frac{0.8 \times (DRAM \text{ in byte} + VRAM \text{ in byte} \times \text{amount of GPU}) \times \text{nodes}}{8 \text{ byte}}} \quad (1)$$

Given the system per node has 128GB of RAM and six V100s with 16GB VRAM, the ideal N_s based on equation (1) is 132664. However in the real benchmark, the system will achieve its peak performance with N_s below or higher than the theoretical N_s . The N_s was started from 120000 and gradually added until the performance began to drop (sign of too large N_s) or became close to the theoretical N_s . The results are shown in *Table 16*. The peak power consumption is variable but the average is between 2200 to 2300W for the entire node, so well below our 3000W limit, however we are only using one node here.

Ns	TFlops
120000	28.16
120320	28.66
122000	29.42
125000	30
125320	system crash

Table 16 – Ns investigation for the peak performance of 6 V100s

The highest result ever achieved in one node with six V100s was 30 TFlops with Ns 125000 and 2300W power consumption. Using Ns higher than 125000 like 125320 in the test will crash the system. This is one of the flaw in NVIDIA binary HPL. Most of the normal-distribution HPL binaries for processors will stop and exit the benchmark with MPI error code 11 if the Ns is too large. In the NVIDIA HPL binary however, the benchmark will crash and the system should be hard-rebooted (in this case, we have to wait for Boston’s people to reboot the system).

Figure 12 shows the difference between the peak performance of six P100s and six V100s based from our benchmark run (We got access to six P100s when Boston Limited gave us the temporary cluster for practice run). Six V100s have 42% performance speedup over six P100s. With such a noticeable performance gain, the Volta architecture demonstrates the capability of the new architecture design over the Pascal generation.

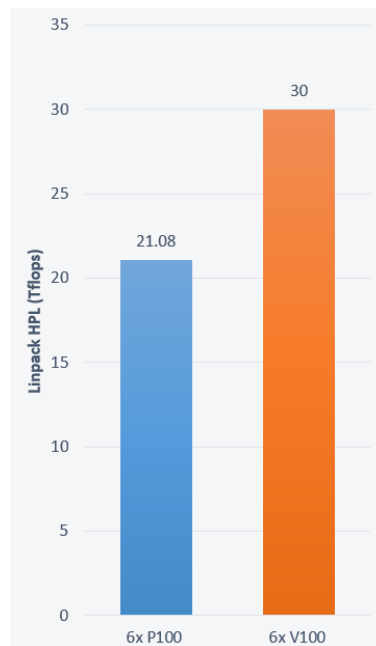


Figure 12 – Comparison between 6 P100s and 6 V100s with the same RAM amount and processors

Table 17 summarizes the important HPL.dat parameters baseline for the next test for power consumption limit.

Parameter	Value
Ns	110000 - 125000
NB	192
P-Q	2-3
PFACT – RFACT	1-1
BCAST	2
SWAP	1

Table 17 Starting point of HPL.dat parameters for the V100 power consumption limit test

4.1.5 HPL Benchmark on 1 Node with 1500W Limit

The next investigation was to keep the power limit below 3000W, however at this moment the second node only contained four V100s. We will carry the first power consumption investigation with one node only. Assuming each node draws the same power, 1500W will be the power limit per node. The purpose of this investigation is to find several ideal hardware parameters for 3000W limit while waiting for all 12 V100s to be ready. The power consumption was observed with SMCIPMITool 2.11.0 software which was set to refresh the reading each second.

Basically, the NVIDIA HPL benchmark run will follow these phases:

- Benchmark initialisation which is done by processors only.
- The actual benchmark run when it prints the status of benchmark progress, N left, time left, approximation of total performance, and GF (Gigaflops) per GPU as shown in *Figure 13*.
- The GF per GPU will reach its highest result in the middle of benchmark running then it will slowly decrease.
- Benchmark validation and print the final result

```
Per-Process Host Memory Estimate: 16.29 GB (MAX) 16.15 GB (MIN)
PCOL: 2 GPU_COLS: 44737 CPU_COLS: 0
PCOL: 1 GPU_COLS: 44737 CPU_COLS: 0
PCOL: 0 GPU_COLS: 43969 CPU_COLS: 960
2018-06-25 14:54:55.273
```

[0:31m Prog= 1.70%	N_left= 133632	Time= 0.70	Time_left= 40.12	IGF= 39649.96	GF= 39649.96	IGF_per= 4405.55	GF_per= 4405.55	^[[0m
[0:31m Prog= 3.39%	N_left= 132864	Time= 1.28	Time_left= 36.41	IGF= 46895.53	GF= 42948.78	IGF_per= 5210.61	GF_per= 4772.09	^[[0m
[0:31m Prog= 4.64%	N_left= 132288	Time= 1.69	Time_left= 34.82	IGF= 48562.22	GF= 44330.23	IGF_per= 5395.80	GF_per= 4925.58	^[[0m
[0:31m Prog= 6.29%	N_left= 131520	Time= 2.24	Time_left= 33.34	IGF= 49109.67	GF= 45492.14	IGF_per= 5456.63	GF_per= 5054.68	^[[0m
[0:31m Prog= 7.52%	N_left= 130944	Time= 2.65	Time_left= 32.62	IGF= 47986.08	GF= 45880.96	IGF_per= 5331.79	GF_per= 5097.88	^[[0m
[0:31m Prog= 9.14%	N_left= 130176	Time= 3.20	Time_left= 31.82	IGF= 47853.62	GF= 46218.35	IGF_per= 5317.07	GF_per= 5135.37	^[[0m
[0:31m Prog= 10.34%	N_left= 129600	Time= 3.61	Time_left= 31.31	IGF= 47421.39	GF= 46354.98	IGF_per= 5269.04	GF_per= 5160.55	^[[0m
[0:31m Prog= 11.92%	N_left= 128832	Time= 4.14	Time_left= 30.56	IGF= 48664.15	GF= 46649.22	IGF_per= 5407.13	GF_per= 5183.25	^[[0m
[0:31m Prog= 13.10%	N_left= 128256	Time= 4.53	Time_left= 30.07	IGF= 48023.05	GF= 46769.37	IGF_per= 5335.89	GF_per= 5196.60	^[[0m
[0:31m Prog= 14.26%	N_left= 127680	Time= 4.93	Time_left= 29.61	IGF= 47986.90	GF= 46866.55	IGF_per= 5331.88	GF_per= 5207.39	^[[0m
[0:31m Prog= 15.80%	N_left= 126912	Time= 5.44	Time_left= 28.99	IGF= 48360.28	GF= 47007.87	IGF_per= 5373.36	GF_per= 5223.10	^[[0m
[0:31m Prog= 16.94%	N_left= 126336	Time= 5.83	Time_left= 28.56	IGF= 47893.62	GF= 47066.51	IGF_per= 5321.51	GF_per= 5229.61	^[[0m

Figure 13 – Example of benchmark stdout of the NVIDIA HPL. Pay attention to the GF_per numbers, and also the time left where this benchmark will run for approximately 40 - 42 seconds.

Through constant SMCIPMITool readings, the power consumption during HPL runs varies according to the benchmark phases above. From idle, the power consumption will rise slightly in phase A and then significantly increase in phase B. The power consumption will reach its highest point for a short amount of time (usually 5 – 7 seconds) when the benchmark produces the highest peak GF per GPU. After that the power consumption decreases slowly toward the end of the benchmark. With 5 seconds interval from the SCC PDU, the power reading will go like as shown in *Figure*

14. During the power limit investigation we pay most of the attention to the power reading in the halfway of benchmark that always reaches the peak wattage

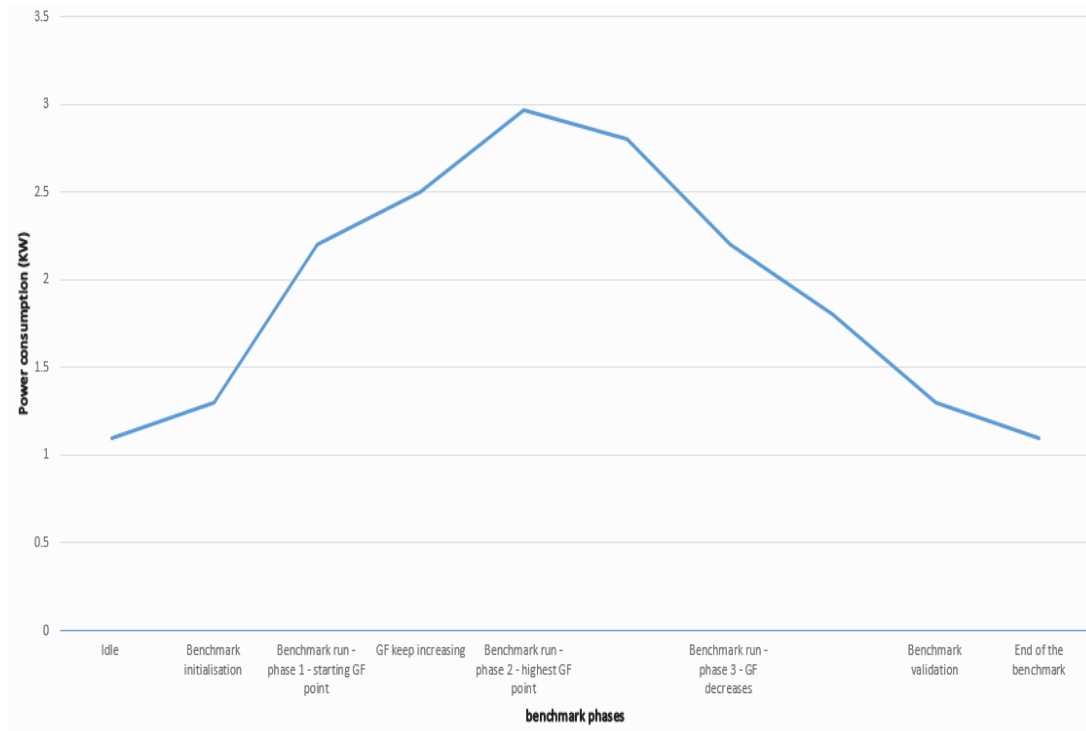


Figure 14 – Power consumption usage during NVIDIA HPL run in the SCC

There are several ways to decrease the power consumption. Based on the previous teams [22][34], GPU ECC should be turned off. Then the most significant way is with downclocking all of the GPUs core clock (note that the memory clock cannot be altered). Downclocking the CPU clock can also slightly decrease the power consumption but will decrease the HPL performance. By default Intel Xeon E5-2630 v4 has 2.2 GHz base clock but it will rise up to 3.1 GHz with turbo-boost when running the HPL. The first test is a quick investigation of downclocking with different combinations of GPU and CPU clocks. Table 18 summarizes the first quick tests with NB 192, Ns 115000, HT off, and GPU_DGEMM_SPLIT 1.0.

GPU clock (MHz)	CPU clock (GHz)	TFlops	Peak wattage
1200	default	26.73	1886W
1110	default	25.46	1685W
1005	default	24.06	1557W
982	default	23.68	1529W
975	2.4 GHz	23.67	1517W
	2.3 GHz	23.45	1501W
	2.2 GHz	23.42	1491W
960	2.4 GHz	23.36	1501W
	2.3 GHz	22.86	1484W
945	default	23.05	1477W
930	default	22.67	1459W

Table 18 – Quick investigation of downclocking CPU and GPU clocks with NB 192, Ns 115000, and GPU_DGEMM_SPLIT 1.00

The current best result was 23.42 TFlops for 6 V100s with GPU clock 975 MHz and CPU clock 2.2 GHz. Based on the first investigation, the sweet-spot clocks for 6 (and 12) V100s is between 960, 975, or 982 MHz. Downclocking the CPU clock can be used for fine-tuning the power consumption too. The second test is to vary the NB, Ns, and GPU_DGEMM_SPLIT parameters:

- Although NB 256 has lower performance than NB 192, it generates lower power consumption because the GPUs don't work at 100%. The performance decrease is not too far from NB 192 and still be able to produce good result.
- Later it was discovered that the ideal Ns should be able to be divided by NB, P, and Q. Bigger Ns also tend to use more power consumption because they produces higher GF per GPU with longer duration. Ns equal to 121344 was used for the following tests as we found it was the best for 6 V100s under 1500W.

Table 19 shows the comparison between NB 192 and 256 with Ns 121344 and clock 975 MHz. At this point we wanted to produce result higher than 24 TFlops. The idea was to fine-tune the NB 192 with DGEMM and CPU clock parameters. When altering the performance, DGEMM has the same act with CPU clock: decreasing DGEMM value will decrease both of the performance and power consumption. However this was not a case on V100s with NB 192 run, as is shown in Table 20. In common practice lower DGEMM means lower performance, yet DGEMM 0.9 in some cases produced higher performance than DGEMM 0.97 for example in our V100. After various attempts with different combinations of DGEMM and CPU clocks, we temporary concluded that there was no other way to reduce the power consumption of the NB 192 run as DGEMM and CPU clock were inconsistent of altering the performance and power consumption. There were possibilities that lower DGEMM will boost up the performance as well as keep the power consumption stable below limit.

NB	TFlops	Peak wattage
256	23.78	1494
198	24.44	1553

Table 19 – Comparison between NB 192 and 256 with clock 975 MHz and Ns 121344

CPU clock (GHz)	DGEMM	TFlops	Peak wattage
2.4	0.97	24.44	1553W
2.4	0.96	24.43	1552W
2.4	0.94	24.48	1543W
2.4	0.9	24.52	1550W
2.3	0.9	24.61	1560W
2.2	0.9	24.35	1530W
2	0.9	23.77	1520W
2.2	0.88	24.34	1530W
2.2	0.83	24.32	1543W

Table 20 – Attempts to decrease the power consumption of NB 192. None of them hit below 1500W

The next step was to fine-tune the performance of the NB 256 and aim for 24 TFlops performance but below the 1500W power limit. The first test was to find the DGEMM value to keep NB 256 run below 1500W, summarized in *Table 21*. DGEMM 0.95 was the best choice for NB 256. *Table 22* proved the performance of DGEMM 0.95 to keep the benchmark run below 1500W.

GPU clock	NB	DGEMM	TFlops	Peak wattage
975 MHz	256	0.97	23.94	1507
	256	0.96	23.95	1533
	256	0.95	23.77	1473

Table 21 – DGEMM investigation with GPU clock 975 MHz, CPU clock 2.4 GHz, and NB 256

Run	TFlops	Peak wattage
1	23.77	1473
2	23.71	1490
3	23.82	1499
4	23.93	1475
5	23.85	1480

Table 22 – Five runs of NB 256, GPU clock 975 MHz, and DGEMM 0.95 begin from idle temperature between 36 – 37°C

In addition, *Table 21* results has another important consideration for the benchmark running. Idle and full load temperature play an important part in maintaining the power consumption below the limit. Cooling fans are one of the major contributors of the power consumption (Our SCC cluster used 8 San Ace 80mm blower fans each node which were probably over 10,000 RPM speed with 40W wattage at full speed, it means they could consume between 250 – 300W alone during the HPL run). Those blower fans are equipped with PWM (Pulse-Width Modulation) sensors to control the RPM speed based on the hardware (mostly CPUs and GPUs) temperature.

For example if the benchmark is started when the GPUs are warm (over 40°C idle), all of the fans will blow harder during the peak GF per GPU benchmark phase (hottest temperature point for the GPUs) which cause to consume more power and eventually make the peak power consumption higher. After numerous experiments, we found out the benchmark run will not cause a sudden peak over 1500W when it was run with 36 - 37°C idle temperature at the first GPU. Basically after a HPL run, the cluster should be cooled down, waiting for the idle temperature to decrease until 36°C or below. Running the benchmark from cold, the GPU idle temperature tends to produce high results as is shown in *Table 21* with 23.93 TFlops

4.1.6 HPL Benchmark on 2 Nodes with 3000W Limit

We received the full 2-node 12 V100s cluster just one day before the cluster was shipped into Frankfurt for the SCC. In a short amount of time, several quick tests were conducted to find out several hardware settings and benchmark parameters for the competition. The first investigation was to find the proper Ns for twelve V100s. We

began with NB 192, slightly downclocked 960 MHz, and Ns 130560. The results are described in *Table 23* where all of them are below 3000W. After several investigations, Ns 156672 was the ‘near limit’ problem size before the cluster crashed with Ns 160656 (lucky one of the Boston’s employee at the moment was in the office and managed to hard-reboot the cluster). Didn’t want to take a risk for the second crash, we ended up at Ns 156672.

GPU clock	NB	Ns	TFlops	Peak wattage
960 MHz	192	130560	40.72	3028
		135168	41.07	3000
		140544	42.75	3020
		145152	42.30	3027
		153600	43.92	3049
		155136	44.43	3030
		156672	44.78	3043

Table 23 – Increasing Ns at 12 V100s with NB 192, DGEMM 0.95, and GPU clock 960 MHz

The next step was to go back to a GPU clock of 975 MHz. With DGEMM 0.95 our benchmark run achieved 43.73 TFlops with 2980W power consumption. So there was a little room before the 3000W limit, the DGEMM value was cranked up into 1.00 and the cluster gained a little speed-up to 44.26 TFlops with 2991W power consumption. Both of the results are described in *Table 24*. 44.26 TFlops was the highest peak result on that day thus we concluded the parameters to be used at the SCC in *Table 25*.

GPU clock	NB	DGEMM	TFlops	Peak wattage
975 MHz	256	0.95	43.73	2980
		1.00	44.26	2991

Table 24 – HPL results for 12 V100s with NB 256 and different DGEMM

Configuration	TFlops	Peak wattage
<ul style="list-style-type: none"> • CPU hyperthreading off, clock 2.4 GHz • GPU ECC disabled, persistence mode enabled • GPU clock memory/core 877/975 MHz. May be higher in the SCC • HPL.dat appendix B • OMP threads 3, GPU_DGEMM_SPLIT 1 	44.26	2991

Table 25 – HPL Configuration before the competition

Unfortunately there was no opportunity to use those configurations at the SCC because all of the hardware issues rose up right after the HPL testing finished, which are already explained in Chapter 3 Section 3.5 and finally led into several hardware issues at the SCC (see Chapter 3 Section 3.6).

4.1.7 HPL Benchmark at the SCC

In such a nerve-breaking 8 hours before the competition started, we had to run the HPL benchmark with an asymmetric and mixed-up 9 V100s cluster. The performance investigation was initiated from square one knowing 9 mixed V100s were used rather than 12 V100s or 6 V100s.

The most important step at the beginning is to run the NVIDIA HPL binary through a cluster which had different numbers of GPU per node (5 V100s at the first node and 4 V100s at the second node). There are two essential scripts to run the NVIDIA HPL binary. We named them ‘running script’ and ‘GPU affinity script’.

- The running script loads the module, defines library PATH, GPU clocks, CPU clock, and executes the mpirun command.
- The GPU affinity script contains the initialisation of all GPUs, together with the CPU binding through numactl. It is also where the GPU_DGEMM_SPLIT variable is defined.

The GPU affinity script should be modified first because each node had a different number of GPUs. IF logic was used to manage the GPU lists e.g. (if node 1 = 5 GPUs, else node 2 = 4 GPUs) but it didn’t work. A naïve way was developed with creating two different GPU affinity scripts for node 1 and 2.

The next modification was the mpirun command inside the running script. OpenMPI 3.1.0 has a flag -H to specify different MPI processes per node which made our job easier. However we still had to bind the two GPU affinity scripts. Another script named ‘linpack_9GPU_run’ was created to bind two GPU affinity scripts with some simple IF logics:

```
If node 1, run GPU affinity script for 5 GPUs. Else if node 2, run
GPU affinity script for 4 GPUs.
```

The mpirun command at the running script runs the ‘linpack_9GPU_run’ script with a complete command:

```
mpirun -np 9 -H epcc1:5,epcc2:4 -mca btl openip,self,vader -bind-
to-node -X LD_LIBRARY_PATH ./linpack 9GPU run
```

All of the attempts successfully ran the HPL benchmark across 2 asymmetric nodes with a decent result of GF per GPU (should be between 4000 – 6000 GF) and showed we were on the right track to investigate and improve the benchmark performance. First Ns 128256 and NB 192 were picked as the starting point. At this point it was important to consider that every parameter possibilities count and probably will defy the common sense for selecting the parameter since the HPL was run on a cluster with improper hardware configuration for GPU benchmarks.

The first investigation was to determine the number of CPU cores per GPU, because new processor 12 cores E5-2650 v4 processors were used instead of 10 cores E5-2630 v4. Judging by the number of GPU (9 V100s) and processor cores (48 cores total), the

number of the CPU_CORES_PER_RANK should be 4 or 5. It turned out that 4 CPUs cores per GPU generated the best result, meanwhile the performance became worse with 5 CPUs per GPU. The current system now favouring enabled HT compared to disabled HT as can be seen in *Table 26* the performance will reduce significantly with CPU_CORES_PER_RANK 5 although with HT on.

Ns	NB	CPU_CORES_PER_RANK	HT	TFlops
128256	192	3	OFF	38.26
		3	ON	38.25
		4	ON	39.01
		5	ON	28.01

Table 26 – Number of OMP threads and HT investigation over 9 mixed up V100s

Because the previous tests broke 3000W power limit, downclocking was necessary for all of the GPUs. The DGEMM value will be always set to 1.00 for simplifying the investigation. *Table 27* recaps the downclocking experiments where a GPU clock of 1155 MHz brought the GPU run power consumption below 3000W. After that the ideal Ns for our 9 mixed V100s was investigated. *Table 28* records all of the testing where Ns 134400 generated the best result with 39.08 TFlops performance for below 3000W.

Ns	NB	GPU Clock	TFlops	Peak Wattage
128256	192	1192 MHz	38.76	3.1 kW
		1170 MHz	38.44	3.01 kW
		1162 MHz	38.28	3.01 kW
		1155 MHz	38.24	2.97 kW

Table 27 – Investigation of downclocking GPU clock with CPU_CORES_PER_RANK=4, CPU clock 2.4 GHz and DGEMM 1.0

GPU clock	NB	Ns	TFlops	Peak wattage
1155 MHz	192	129792	38.31	2.95 kW
		130560	38.56	2.97 kW
		131328	38.55	2.95 kW
		132864	38.92	2.94 kW
		133632	38.39	3.00 kW
		134400	39.08	2.95 kW
		135936	38.71	2.97 kW

Table 28 – Investigation of the highest Ns for the mixed-up 9 V100s

Later we thought that the system still had some room for improvement. The idea was to balance the performance by slightly increasing the clock of the second node GPUs because it had fewer GPUs came up. The first node GPU clock was kept to 1155 MHz. This was not an easy thing to do because the cluster always broke the power limit when the second node GPUs had bigger clock. To counter the problem, the GPU should run from the lowest idle temperature possible to avoid the excessive power from cooling fans. Again it was not an easy feat as an air-cooling system is greatly

affected by the ambient temperature. The crowded hall at the opening of the SCC increased the ambient temperature thus made the cluster temperature hotter.

At this moment, Boston Limited gave us an access to control the PWM of the fans through the Supermicro web command. There are two options to control the fans: full speed mode and normal speed mode. The PWM setting fan was used to manage the cluster temperature on the fly. The idea is to set all of the fans into maximum speed mode before run of the benchmark until the end of the HPL initialisation phase. Shortly before the actual HPL phase run, all of the fans will be switched into normal speed mode to lower the RPM. After the benchmark finished, the fan mode is switched into maximum speed again to quickly cool down the cluster.

The goal is to ensure all of the GPUs run the benchmark with the lowest temperature possible and thus prevent the PWM fan speeding up too much during the peak GF and finally decrease the peak power consumption. Ultimately those attempts worked and the second node GPU clock could be increased without breaking the power limit. *Table 29* shows the HPL result with different GPU clock combination. The combination of 1155 and 1177 MHz at that moment slightly increased the peak performance into 39.44 TFlops but within the power limit.

CPU clock node 1	CPU clock node 2	TFlops	Peak wattage
1155 MHz	1200 MHz	39.54	3.02 kW
	1192 MHz	39.4	3.02 kW
	1177 MHz	39.44	2.99 kW
	1170 MHz	39.32	2.99 kW
	1162 MHz	39.01	2.94 kW

Table 29 – Using different GPU clock on both nodes to achieve higher performance with GPU temperature tweaking

It was decided to use the final configuration of combination 1155 and 1177 MHz for the competition run. Before the competition was started, there were opening ceremony and mini game to find the team's competition T-shirts. While the opening ceremony and mini game the cluster was left with maximum speed fan mode. Thirty minutes spent in the opening ceremony gave a decent amount of time to cool down the GPUs into the lowest temperature we had ever known which was 19°C. During the competition a brave decision was initiated to slightly increase the second node GPU clock from 1177 MHz into 1185 MHz. The end result is described in *Table 30* where we achieved 39.55 TFlops peak with 2970W power usage.

Configuration	TFlops	Peak wattage
<ul style="list-style-type: none"> • CPU hyperthreading on, clock 2.4 GHz • GPU clock 1155 MHz on the first node and 1185 MHz on the second node • Ns 134400, NB 192, and P-Q 3-3 • OMP threads 4, GPU_DGEMM_SPLIT 1 	39.55	2.97 kW

Table 30 – Submitted HPL result at the SCC

4.2. HPCG NVIDIA Binary

High Performance Conjugate Gradient (HPCG) is a relatively new benchmark in the supercomputer world which is designed to augment and change the HPL benchmark in Top500 ranking. HPCG uses Preconditioned Conjugate Gradient (PCG) algorithm to solve a sparse triangular system as part of Gauss-Seidel smoother with sparse matrix-vector multiplication, vector updates, and global dot products. HPCG is programmed in hybrid MPI and OpenMP C++.

The main purposes of HPCG is also to stress the memory and network alongside main processing performance which is intended to closely resemble actual scientific computing. The history of the HPCG benchmark in the SCC began at the SCC 2014 where it was introduced as the secret application [34]. Since then, HPCG become the compulsory benchmark alongside with HPL at any SCC. As for now, HPCG become an important benchmark for supercomputing community.

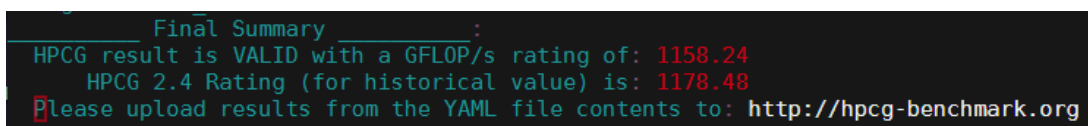
4.2.1 The Binary

HPCG 3.1 NVIDIA binary for Volta which is available at HPCG official website [35] was used for the SCC. CUDA 9.0 and OpenMPI 1.10.2 are needed to run the binary. Running the HPCG NVIDIA binary is simpler than the HPL NVIDIA binary since there is no affinity CUDA script to initialise the GPUs. A run script is necessary just to manage the hardware configuration before running the binary.

4.2.2 HPCG Parameters, Rules, and Output

There are two parameters for any HPCG binaries which are located inside an input file named hpcg.dat:

- Problem sizes for the three dimensional local domain. The problem size should be able to be divided by 8 and enough to fill at least 1/4 of the main memory.
- Benchmark runtime in seconds. The rule for any official submissions (including the SCC) is 1800 seconds minimum. For example we defined 1860 seconds for our benchmark run.



```
Final Summary :  
HPCG result is VALID with a GFLOP/s rating of: 1158.24  
HPCG 2.4 Rating (for historical value) is: 1178.48  
Please upload results from the YAML file contents to: http://hpcg-benchmark.org
```

Figure 15 – Result of the HPCG inside the yaml output file

Any HPCG binaries will automatically create two output files after the benchmark run finishes, there are a log-file and output yaml file. The yaml file is the one that will be used for the benchmark submission. *Figure 15* shows the result of the HPCG inside the yaml file which reads 1158.24 GFlops.

4.2.4 HPCG Benchmark on 1 Node

The first investigation was the problem size. Common problem sizes for the HPCG are the (power of two³) like 128 128 128 for example. After some tests, the V100 has the same ideal problem size with the previous P100 which is 256³. The first result was 841.6 GFlops for 6 V100s with 1583W power consumption which are described in *Table 31*. Any number of OMP threads didn't seem to affect the performance on our runs and it was decided to use 5 OMP threads.

The HPCG NVIDIA uses smaller hardware resource than the HPL NVIDIA. No matter how big the problem size is, the binary only uses approximately 8 GB of VRAM in both V100 16 GB and 32 GB variants. The HPCG NVIDIA is also consumes half power usage than the NVIDIA HPL in boost clock as for V100 it only needs 150W to run the binary.

During our visit to Boston Limited HQ, we noticed the status of the ECC was 0 (zero) and not OFF. When we tried to disable the ECC through nvidia-smi command, the status changed from 0 to OFF, meaning both of them are different (it helps to gain some performance improvement at the HPL). With ECC OFF, the HPCG performance decreased and it was noticeable to get lower GF per GPU. However, the power consumption also decreased to below 1500W and save for two nodes run with boosted clock. The processors HT were also turned off to ensure the power consumption was below the limit. *Table 32* shows the comparison between NVIDIA HPCG run with ECC status 0 and OFF.

Unfortunately there was no chance to run the NVIDIA HPCG with 12 V100s because the second node didn't detect all of the GPUs. Judged by HPCG has good scalability, our cluster should produce 1500-ish GFlops for 12 V100 GPUs with 2900W power usage.

Hardware	6x V100s 16GB	6x V100s 32GB
Result	841.6 GFlops	769 GFlops
GF per GPU	140.1 GFlops	128.2 GFlops
Peak wattage	1583W	1423W
ECC Status	0	OFF
OMP threads	5	3
Problem sizes	256 256 256	
HT	ON	OFF

Table 31 – Comparison of the HPCG result over 256³ grid size with different V100 configuration

4.2.5 HPCG Benchmark at the SCC

The run script to run the HPCG NVIDIA should be modified in order to load the job into asymmetric 9 V100s. OpenMPI 1.10.2 which was used to run the binary doesn't have the flag option to specify the number of MPI processes (which one MPI process equal to one GPU) per node. Alternatively 'rankfile' method was used to bind each GPUs with specific processor cores as is showed in *Figure 16*.

rank 0=epcc1 slot=0:0-2	#Node 1 GPU 1 with CPU 1 core 0-2
rank 1=epcc1 slot=0:3-5	#Node 1 GPU 2 with CPU 1 core 3-5
rank 2=epcc1 slot=0:6-11	#Node 1 GPU 3 with CPU 1 core 6-11
rank 3=epcc1 slot=1:0-5	#Node 1 GPU 4 with CPU 2 core 0-5
rank 4=epcc1 slot=1:6-11	#Node 1 GPU 5 with CPU 2 core 6-11
rank 5=epcc2 slot=0:0-5	#Node 2 GPU 1 with CPU 1 core 0-5
rank 6=epcc2 slot=0:6-11	#Node 2 GPU 2 with CPU 1 core 6-11
rank 7=epcc2 slot=1:0-5	#Node 2 GPU 3 with CPU 2 core 0-5
rank 8=epcc2 slot=1:6-11	#Node 2 GPU 4 with CPU 2 core 6-11

Figure 16 – Rankfile example to run the HPCG with mixed V100s

Due to an unknown network error, the binary didn't want to run when it was launched from the first node. We run the binary from the second node and it worked. The full mpirun command to run the benchmark is:

```
mpirun -np 9 -H epcc1,epcc2 -rf rankfile --mca btl openib,self -
bind-to none -x LD_LIBRARY_PATH=$LD_LIBRARY_PATH
$HPCG DIR/$HPCG BIN
```

A problem size of 256^3 was chosen with OMP threads set to 4 and HT on (same with the HPL NVIDIA). The competition result is summarized in *Table 32*. Surprisingly, despite the asymmetric and mixed V100s, the binary produced similar GF per GPU compared to the run before the competition which was between 128 – 129 GF. For valid submission, we run the HPCG at the SCC for 1860 seconds running time.

Hardware	9x mixed V100s
Result	1158.24 GFlops
GF per GPU	128.781 GFlops
Peak wattage	2.5 kW
ECC Status	OFF
Problem sizes	256 256 256
OMP threads	4
HT	ON

Table 32 – Submitted HPCG result at the SCC

4.3. HPC Challenge

The HPC Challenge (HPCC), mainly designed by Piotr Luszczek, et al was introduced in 2006 as part of the US DARPA, DOE, and NSF through the HPCCS (High Productivity Computing System) project [16]. HPCC is a suite of benchmarks composed of 7 different benchmarks to measure every kernel part of the supercomputer performance, network, and memory. The benchmarks that compose the HPCC suite are: HPL, PTRANS [36], RandomAccess, FFTE [37], STREAM [38], DGEMM, and Effective Bandwidth [39]. One Makefile and several C programs – objects are available inside the package to compile and combined all of them into one executable binary. *Table 33* describes the operation of each HPCC benchmark.

Benchmark	Operation	Objective
HPL	High Performance Linpack. Solves the linear problem $Ax=B$ on LU factorization with partial row pivoting.	Measures the peak performance of the system.
PTRANS	Parallel Matrix Transpose from PARKBENCH suite. Updates matrix alongside with the sum of its transpose and another matrix with $A = A^T + B$ operation.	During the benchmark, pairs of processors communicate with each other, hence it is a test of total network communication capacity.
Random Access	Calculates a series of randomly updated of integer to random locations in memory.	Measures the rate of random update per second (GUPS or Gigabyte Update Per Second) at memory.
FFTE	FFT from East, developed by Daisuke Takahashi. Computes the DFT in 1 dimensional sequence for the HPCC.	Measures the floating point of operation with different approach than HPL in term of memory access.
STREAM	Measures sustainable memory bandwidth between four operations copy ($c = a$), scale ($b = ac$), add ($c = a + b$), and triad ($a = b + ac$).	HPCC uses triad operation as its main result from STREAM.
DGEMM	Double-precision General Matrix-matrix Multiply. Computes the matrix $C \leftarrow \alpha AB + \beta C$	Measures the floating point operation of matrix multiplication.
Effective Bandwidth	Exchange 8 messages for latency and 2000 for bandwidth in ping-pong, random ring, and natural patterns.	Measures effective bandwidth and latency of the network interconnect based on number of communication patterns.

Table 33 – HPCC Benchmarks

4.3.1 HPCC Binary and Output

The HPCC source code can be obtained from the official website [40]. Version 1.5.0 was used for the SCC. The HPCC source code is compiled with the same way as building the HPL Netlib version, meaning the user has to link the Arch file with MPI, BLAS/LAPACK libraries, and define the compiler flag optimisations. For the SCC we used Intel MPI 18.0.0 with MKL 18.0.0 to compile the HPCC.

There are two categories of the HPCC: baseline run and optimised run. A baseline run means the optimisation is only through the selection of the MPI and libraries to compile the source code. On the other hand, the optimised version involves code modification. There are specific rules about which function that can be modified from the official website [41] but the SCC committee permits us to do any modification. It means a total modification like to port the HPCC code to heterogeneous CPU-GPU code is allowed for the SCC.

The HPCC benchmark will automatically create a complete output file after the benchmark run finishes. Because HPCC is a benchmark suite, the output file will contain various results from each benchmark, not to mention that each benchmark has its own sub-results like STREAM have copy, scale, add, and triad results. We didn't know how the committee will evaluate the results so we follow the HPCC website submission results which are described in *Table 34*.

Benchmark	Output Name	Variable in the output file
HPL	G-HPL	HPL_Tflops
PTRANS	G-PTRANS	PTRANS_GBs
Random Access	G-RandomAccess	MPIRandomAccess_GUPs
FFTE	G-FFT	MPIFFT_Gflops
STREAM	EP-STREAM Sys	StarSTREAM_Triad × MPI processes
	EP-STREAM Triad	StarSTREAM_Triad
DGEMM	EP-DGEMM	StarDGEMM_Gflops
Effective Bandwidth	RandomRing Bandwidth	RandomlyOrderedRingBandwidth_GBytes
	RandomRing Latency	RandomlyOrderedRingLatency_usec

Table 34 – HPCC website results

4.3.2 HPCC Parameters

The input parameters for HPCC are exactly same as for HPL, except there are additional Ns and NB for PTRANS. In HPCC, the values of Ns and NB affects HPL, PTRANS, FFT, and DGEMM. Usually PTRANS needs to be either the same size as Ns and NB used with HPL or bigger.

4.3.3 HPCC Investigation on Old Boston Cluster

The first optimisation was to compile the HPCC with Intel compiler optimisation flags, Intel OpenMP hybrid, and MKL libraries for FFT:

```
-O3 -fopenmp -ipo -w -ansi-alias -i-static -z noexecstack -z relro
-z now -nocompchk -Wall -restrict -D FFTW3 -mt mpi
```

The baseline run will be investigated first. Next, we carried out the investigations for searching the HPL peak performance of the processors with HPL MKL 18.0.0 distribution binary. The HPL peak performance result for dual E5-2630 v4 was 717 GFlops with Ns 55000, meanwhile the quad processors (two nodes) was 1377 GFlops with Ns 75000. Those peak results and Ns will be the reference to evaluate the efficiency of our HPCC results.

There are some interesting points for HPL on Intel processors. HPL version Netlib 2.0 (and slightly modified for the HPCC [16]) is used in HPCC. The ideal amount for Ns in the HPL Netlib is bigger than the ideal Ns in Intel-optimised HPL from MKL package. For example, Ns 75000 is where the HPL in HPCC stops to gain more performance on one node. The peak HPL result from the HPCC is 624 GFlops, 10% lower than the HPL MKL result. We checked the result by compiled the HPL 2.0 from

Netlib and it gave us similar result. The author also tried to compare the result between HPL from MKL and Netlib in the author's PC (Core i5 2400) and it showed the same pattern where HPL from Netlib produced around 10% lower peak performance than HPL from MKL distribution. Ns 75000 was also the best size for the FFT, DGEMM, and PTRANS (with additional Ns 20000 for PTRANS only).

All of the HPCC benchmarks are hybrid MPI-OpenMP benchmark thus we varied the combination of MPI and OMP thread values to fill all of the processor cores. We did the test with HT off, knowing hyperthreading didn't offer additional performance for all of the benchmarks. Intel KMP affinity was set to scatter to make sure each threads run at unique cores. *Table 35* summarizes the investigation of different MPI-OMP thread combinations.

Benchmark	20 MPI 1 OMP	10 MPI 2 OMP	5 MPI 4 OMP	4 MPI 5 OMP	2 MPI 10 OMP
G-HPL (TFlops)	0.596	0.606	0.612	0.589	0.624
G-PTRANS (GB/s)	7.505	5.356	3.437	1.506	2.279
G-Random Access (GUPS)	0.0032	0.0037	0.002	0.001	0.001
G-FFT (GFlops)	10.803	8	5.401	5.77	3.271
EP-STREAM Sys (GB/s)	59.82	57.02	43.15	36.36	20.82
EP-STREAM Triad (GB/s)	2.991	5.702	8.63	9.09	10.411
EP-DGEMM (GFlops)	32.251	66.068	139.555	167.88	336.01
Random Ring Bandwidth (GB/s)	1.906	3.97	9.247	11.045	12.482
Random Ring Latency (usec)	0.869	0.75	0.702	0.564	0.726

Table 35 – Results of HPCC benchmark on one node with Ns 75000 and additional 20000 for PTRANS

Each benchmark has different approach to exploit its best performance. They are divided into two classes: favouring MPI processes and favouring OMP threads. Based on the results in *Table 35*:

- Favouring MPI: PTRANS, FFTE, Random Access, and accumulated STREAM system
- Favouring OMP: HPL (Intel-optimised), STREAM, DGEMM, Effective Bandwidth.

Possibly, there are three ways to run the HPCC benchmark: pure MPI processes, OMP threading with minimal MPI processes, or balanced with equal number of MPI and OMP processes. A balanced run was selected because we thought the committee probably will judge the results by how balanced they are (since one of the HPCC

purposes is to evaluate the HPC system balance). ‘The median’ for all the benchmark results based on *Table 35* is 5 MPI processes with 4 OMP threads. The next test with 2 nodes will be carried out doubling the number of MPI processes but still with 4 OMP threads per MPI. The results and the speedup are presented in *Table 36*.

Benchmark	1 node	2 node	Speedup
G-HPL (TFlops)	0.612	1.201	1.96x
G-PTRANS (GB/s)	3.437	5.918	1.72x
G-Random Access (GUPS)	0.002	0.012	6x
G-FFT (GFlops)	5.401	9.44	1.74x
EP-STREAM Sys (GB/s)	43.15	85.87	1.99x
EP-STREAM Triad (GB/s)	8.63	8.587	- 1.01x
EP-DGEMM (GFlops)	139.555	138.018	- 1.01x
Random Ring Bandwidth (GB/s)	9.247	2.116	- 4.37x
Random Ring Latency (usec)	0.702	1.123	- 1.59x

Table 36 – Comparison between one and two node performance in HPCC with OMP threads 4, Ns 95000 and additional 20000 for PTRANS

The benchmarks that favouring MPI processes had reasonably good speedup except for the Random Access which had a super-linear speedup. Several re-run was conducted and all of the results showed the same speedup pattern (including the super-linear speedup on Random Access, probably because the benchmark perfectly fits the cache level memory). Meanwhile the benchmarks that favouring OMP threads suffered with performance decrease because of doubled MPI processes where they probably caused load imbalance and communication overhead among expensive computation sections (typically parallel loops that contain lot of small calculations). STREAM Triad and DGEMM only have a little performance decrease, in the other hand the Effective Benchmark has visibly speed-down.

4.3.4 HPCC Modification and Porting to GPU

The next investigation was to modify the HPCC code for the SCC and port some benchmarks to the GPU. Because of the limited time and competition priority to the HPL, we only selected some benchmarks to be modified which were PTRANS, STREAM, HPL, and FFTE. Among the selected benchmarks, HPL and FFTE were ported into NVIDIA GPU. CUDA cu-FFT was used to port to load the FFTE code into the GPU. As for the HPL, the idea was to replace the HPL 2.0 with the HPL NVIDIA binary.

The first benchmark to be optimised was STREAM. There are several optimisation flags from the Intel compiler for STREAM. To achieve better performance, STREAM should have sufficient array size and good decomposition value of the array. The choice of the array size depends on the hardware architecture. For Xeon Broadwell processors, a 64MB array size with 64 bits precision is the best. The complete flags for the STREAM are:

```
-DSTREAM_ARRAY_SIZE=64000000 -qopt-streaming-cache-evict=0 -qopt-streaming-stores always -qopt-prefetch-distance=64,8 -DMKL_INT=long -DLONG IS 64BITS
```

Comparison of STREAM Triad and Sys are presented in *Table 37*. The amount of the speedup is quite good at 1.75x.

	Baseline	Optimised	Speedup
EP-STREAM Sys (GB/s)	43.15	75.55	1.75x
EP-STREAM Triad (GB/s)	8.63	15.11	

Table 37 – STREAM baseline versus optimised run at one node with 5 MPI processes and 4 OMP threads

There were some difficulties to modify the PTRANS benchmark because the code is written in an old school C style with minimum documentation. There was no speedup gained after several changes to the source code like unrolling the loops manually, function inlining, or removing redundant variables. Perhaps the most difficult part to modify the code is confusing variable and function names (like f__1, f__2, f__3, f__4, and so on). However further investigation was not initiated because the main focus of the HPCC modification was to replace HPL with the NVIDIA binary version.

To replace the HPL 2.0 with HPL NVIDIA, Makefile modification is necessary to make the HPCC package run the HPL NVIDIA. First, the HPL NVIDIA package is copied into the HPCC package. There is one C file named hpcc.c that contains all of the scripts required to compile each of the HPCC benchmarks, including the HPL. The line to compile HPL was replaced with a C system command to run the HPL NVIDIA binary. *Figure 17* shows the modified HPL section inside hpcc.c file. Rank 0 is specified because if the HPCC benchmark is run with 5 MPI processes, the HPL NVIDIA will run 5 times too. The HPL output was set to be recorded into the HPCC output which is hpccoutf.txt.

```
// if (params.RunHPL) HPL_main( argc, argv, &params.HPLrdata,
&params.Failure );

// HPL NVIDIA Binary
if (myRank == 0){
    #pragma omp single
    {
        printf("===== Running the NVIDIA binary HPL! =====");
        if (params.RunHPL){
            system("./linpack_run_EPCCTeam >> hpccoutf.txt");
        }
    }
}
```

Figure 17 – Modification in the hpcc.c to replace the HPL 2.0 with HPL NVIDIA Binary.

Now there are two input files to run the HPCC with HPL NVIDIA, HPL.dat for the NVIDIA binary and hpccinf.txt for the rest of the HPCC benchmarks. The HPCC HPL NVIDIA successfully ran on one node, however for unknown reason it didn't want to run with 2 nodes at the SCC because of network issue.

The works of porting the FFT into GPU was done by another team member, Linda Koletsou Koutsu. The FFTE main file tstfft.c was modified by replacing the FFT functions with CUDA functions from cu-FFT library. The main obstacle of porting the FFT code is the modified original FFTE code for the HPCC package itself where several functions are redefined for the HPCC package linking.

The ported code could be compiled (by NVIDIA PGI) and run across the V100 GPUs. When the HPCC benchmark entered the FFTE part, all of the GPUs worked 100%. *Table 38* shows the huge difference between FFTE performance on CPU and GPU. The GPU run with 4 V100s produced 880.51 GFlops compared to 2 Xeon E5-2630 v4 which is only 2.61 GFlops (337.36x speedup). But, it seems the GPU result has an error validation, where the result of $\max(|x-x_0|)$ is NAN or not a number. The generation time is also longer than the CPU version. Although there is an output about zero error in the node(s), we decided not to use the FFT GPU version because of the NAN validation result.

FFTE CPU result with 2 E5-2630 v4	FFTE GPU result with 4 V100s
Begin of StarFFT section. Vector size: 1048576 Generation time: 0.035 Tuning: 0.000 Computing: 0.068 Inverse FFT: 0.083 $\max(x-x_0)$: 3.265e-15 Node(s) with error 0 Minimum Gflop/s 1.469459 Average Gflop/s 1.548555 Maximum Gflop/s 1.717283 Current time (1533410078) is Sat Aug 4 20:14:38 2018 End of StarFFT section. Begin of SingleFFT section. Node(s) with error 0 Node selected 33 Single FFT Gflop/s 2.612225 Current time (1533410079) is Sat Aug 4 20:14:39 2018	Begin of StarFFT section. Vector size: 134217728 Generation time: 4.043 Tuning: 0.270 Computing: 0.022 Inverse FFT: 5.052 $\max(x-x_0)$: nan Node(s) with error 0 Minimum Gflop/s 362.322559 Average Gflop/s 652.260824 Maximum Gflop/s 826.499083 Current time (1529510523) is Wed Jun 20 17:02:03 2018 End of StarFFT section. Begin of SingleFFT section. Node(s) with error 0 Node selected 4 Single FFT Gflop/s 880.516316 Current time (1529510537) is Wed Jun 20 17:02:17 2018

Table 38 – Comparison of FFTE complete results between CPU and GPU version

4.3.5 HPCC at the SCC

As it is already mentioned before, Boston Limited changed the processors from Xeon E5-2630 v4 to Xeon E5-2650 v4 which has 12 core and 24 threads, meaning the number of the MPI processes and OMP threads should be adjusted. Because of the troubleshooting the HPL and HPCG before the competitions, 1 hour was the only remaining time to test the modified HPCC. That moment was also the first time to test

the HPCC NVIDIA with 2 nodes mixed V100s. Unexpectedly, the benchmark failed to run and we rushed to fix the problem in the middle of the competition after finishing the HPL and HPCG benchmark runs.

Because of the intense HPL NVIDIA troubleshooting, the remaining time was reduced into 1 hour before the competition ended. The final decision was to run the HPCC benchmark with processors only. The balanced HPCC run was selected with 12 MPI processes with 4 OMP threads per MPI. The same input parameters from E5-2630 v4 were used for the E5-2650 v4 except the number of P and Q due to different amount of MPI processes. Unexpectedly the benchmark run was almost 40 minutes long which was supposed to run with 30 minutes long only.

Table 39 compares the 12 MPI – 4 OMP E5-2650 v4 run at the SCC with 10 MPI – 4 OMP from E5-2630 v4 run with non-optimised HPCC before the competition. Overall the E5-2650 v4 results are better except for HPL which is awful compared to E5-2630 v4 and FFTE which is supposed to gain more speedup. The HPL section also consumes most of the running time with 822 seconds or almost 14 minutes. Later the peak performance of the dual E5-2650 v4 was investigated with HPL MKL distribution and the result was 823 GFlops. With four processors under HPL 2.0 from HPCC it should be 1.4 TFlops at minimum.

Benchmark	E5-2630 v4 with 10 MPI – 4 OMP. Non-optimised version	E5-2650 v4, with 12 MPI – 4 OMP at the SCC	Speedup
G-HPL (TFlops)	1.201	0.694	- 1.7x
G-PTRANS (GB/s)	5.27	6.3	1.19x
G-Random Access (GUPS)	0.012	0.013	1.08x
G-FFT (GFlops)	9.25	8.65	- 1.06x
EP-STREAM Sys (GB/s)	86.77	131.4	1.5x
EP-STREAM Triad (GB/s)	8.67	10.95	1.26x
EP-DGEMM (GFlops)	138.685	114.952	- 1.2x
Random Ring Bandwidth (GB/s)	1.62	1.47	- 1.1x
Random Ring Latency (usec)	1.132	1.08	1.04x

Table 39 – Comparison between the result of the HPCC benchmark between E5-2630 v4 processors and E5-2650 v4 processors with similar MPI-OMP processes

After the competition finished, we didn't have an access to Boston SCC Cluster with two nodes until one week before the dissertation deadline came up. Other team members also had to use the cluster for their own projects. The HPCC re-

benchmarking investigation was initiated only in a very short amount of time to give the availability of the cluster for others. It turned out that the reason for the poor HPCC performance is the wrong hyperthread enable/disable script. The hyperthread script at the competition was designed for 10 cores Xeon E5-2630 v4 and forgot to be re-adjusted for the 12 cores Xeon E5-2650 v4. This caused 4 additional threads running in the CPU with HT off and utterly messed up the KMP affinity spreading (high possibility of two processes were run in the same core) thus caused very poor performance in the HPCC benchmark. *Table 40* shows the comparison of the re-run results over the SCC result where overall the re-run results are better than the SCC results. Once the input parameters were correctly adjusted, the HPCC benchmark will gain more speedup than the quick investigation results with correct HT script.

Benchmark	Results at the SCC	Re-run with correct disabled HT
G-HPL (TFlops)	0.694	1.133
G-PTRANS (GB/s)	6.3	6.641
G-Random Access (GUPS)	0.013	0.013
G-FFT (GFlops)	8.65	9.3
EP-STREAM Sys (GB/s)	131.4	152.04
EP-STREAM Triad (GB/s)	10.95	12.67
EP-DGEMM (GFlops)	114.952	133.703
Random Ring Bandwidth (GB/s)	1.47	1.68
Random Ring Latency (usec)	1.08	1.03

Table 40 – Comparison between HPCC results at the SCC and the quick re-run with correct disabled HT

Chapter 5

Comparison of the Fraction of Peak Performance and System Balance

This chapter describes the comparison of the fraction of peak performance and system balance analysis based on the HPCC benchmark over different HPC systems. There are five methods for the comparison test:

- The fraction peak of HPCG result over HPL peak performance.
- Network interconnect performance evaluation with the correlation between Random Ring Bandwidth from HPCC and HPL performance.
- Memory performance evaluation with the correlation between Parallel STREAM Triad from HPCC and HPL performance.
- Correlation between G-FFT from HPCC and HPL performance for another synthetic scientific benchmark than HPCG.
- Speedup investigation of the CP2K H2O-64 benchmark for a real scientific benchmark.

The main objective is to see how well the HPCG and HPCC benchmarks augment the HPL benchmark in terms of performance efficiency and balance between different HPC systems. The comparison tests are carried out over HPC systems with different manufacturers and architecture generations. The first comparison is a homogeneous CPU system only where *Table 41* details the HPC machines to be compared. The benchmarks are tested in 1,2,4,6, and 8 nodes for the scalability test.

Platform	System 1 ‘Cirrus’	System 2 ‘Scafell Pike’
HPC name	Cirrus [42]	Scafell Pike [43]
Institution	EPCC	Hartree Centre
Machine	SGI/HPE ICE XA Cluster	Bull-Atos Sequana X1000
Node count	280	846 for CPU node
CPU per node	2x Intel Xeon Broadwell E5-2695 v4, 2.1 GHz 36c/72t.	2x Intel Xeon Gold 6142, 2.6 GHz 32c/64t
Memory per node	128GB DDR4	192GB DDR4
Network	Mellanox Infiniband FDR	Mellanox Infiniband EDR
Cooling	Air-cooling	Bull custom water-cooling

Table 41 – Homogeneous CPU systems for the comparison test

The second comparison is heterogeneous accelerator machines but only with the HPL – HPCG fraction of peak test. *Table 42* lists the accelerators to be compared, all of them are installed in the Boston SCC Cluster.

Platform	System 1 ‘P100’	System 2 ‘V100’
HPC name	Boston SCC Cluster	
Machine	Custom build with Supermicro X10DGO-SXM NVLink motherboard	
Accelerator count	6	
CPU	2x Intel Xeon Broadwell E5-2630 v4, 2.2 GHz 20c/40t.	
Memory	128GB DDR4	
Accelerator	NVIDIA Tesla P100 16GB NVLink SXM2	NVIDIA Tesla V100 16GB NVLink SXM2
Cooling	Air-cooling	

Table 42 – Heterogeneous GPU systems for the comparison test

5.1. Benchmark Comparison on Cirrus and Scafell Pike

Intel MPI and MKL are used to compile and run all of the benchmarks since both of the systems are equipped with Intel processors. *Table 43* lists the detail of the benchmarks and binary for the performance testing. Usually the ideal comparison tests should use the same version of the compilers and libraries, but in this case the MPI and MKL versions used depend on what was available on the particular machine.

Benchmark	Cirrus	Scafell Pike
HPL	Intel Parallel Studio MKL binary 17.4.196	Intel Parallel Studio MKL binary 18.1.163
HPCG		
HPCC	Version 1.5.0. Compiled with Intel MPI and MKL 17.2.096	Version 1.5.0. Compiled with Intel MPI and MKL 18.1.163
CP2K	Self-compiled O2 optimisation with Intel MPI 17.2.096 and MKL 16.0.0	Self-compiled O2 optimisation with Intel MPI and MKL 18.1.163

Table 43 – List of the benchmarks in the Cirrus and Scafell Pike systems

5.1.1 Intel MKL Benchmark Background

The MKL benchmark from Intel Parallel Studio consists of HPCG and HPL benchmarks that are already compiled and optimised against Intel MKL. All of them are hybrid MPI-OpenMP. The general rule to run the benchmarks are the number of MPI processes is number of CPU sockets meanwhile the OMP threads is the amount of cores each in processor. For example to run the benchmarks in the dual Xeon Gold where each socket has 16 cores, 2 MPI processes with 16 OMP threads will run the benchmark with the correct 100% utilisation. The rules remain the same if the processor has hyperthreading enabled. The OS will read 50% load usage although the benchmarks run with 100% utilisation. Doubling the number of OMP threads to fill the HT enabled processor load into 100% doesn’t improve the performance. The ideal environment to run the MKL benchmarks should be with hyperthreading off, but here the author doesn’t have sudo access to disable the hyperthreading on both machines.

There are several compiled binaries inside the HPL and HPCG package. The very first investigation was to choose the correct binary to carry out the benchmark test. *Table 44* describes the HPL and HPCG binaries used on each platform.

Benchmark	Cirrus	Scafell Pike
HPL	xhpl static	xhpl dynamic
HPCG	xhpcg_avx2 with AVX2 enabled for Broadwell	xhpcg_skx optimised for AVX512 Skylake

Table 44 – List of the benchmark binaries that are used for the tests

5.1.2 HPCG Fraction of Peak over HPL

The steps to seek the peak performance are same with the investigation of HPL and HPCG benchmarks in the previous chapter. The same HPL.dat parameters from HPL NVIDIA were used except the values of Ns, P, and Q because of different number of cores and hardware (processor). Both of the Broadwell and Skylake run best with NB 384 and BCAST 2. All that remained was to find the correct size of Ns. Appendix B records all of the HPL and HPCG benchmark steps for both Cirrus and Scafell Pike. The same steps were applied to HPCG by testing various problem sizes. *Figure 18* and *Figure 19* show the result of the HPL and HPCG benchmarks on Cirrus and Scafell Pike machine.

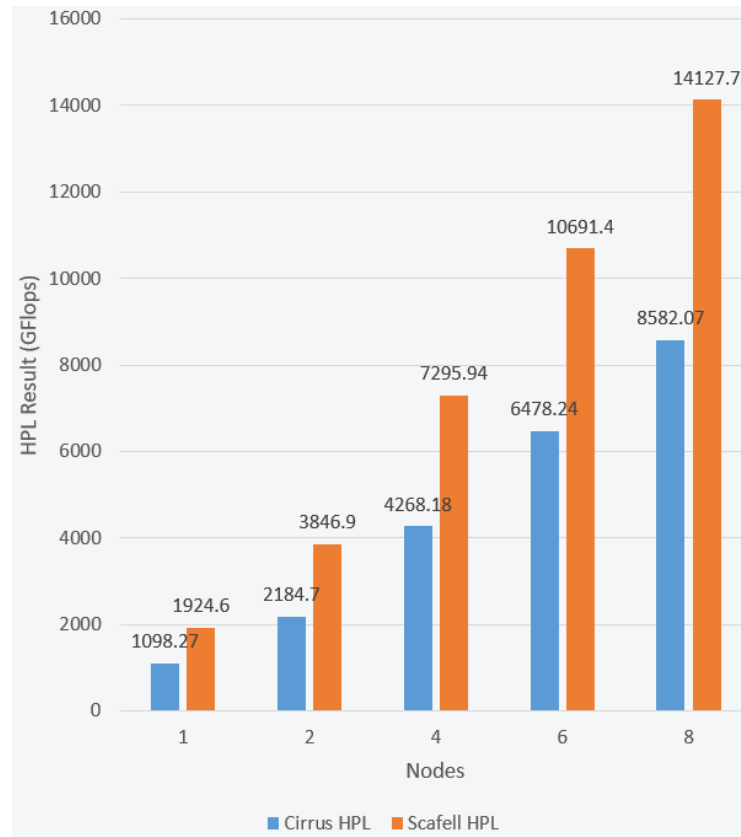


Figure 18 – HPL results of the Cirrus and Scafell Pike systems across different number of nodes

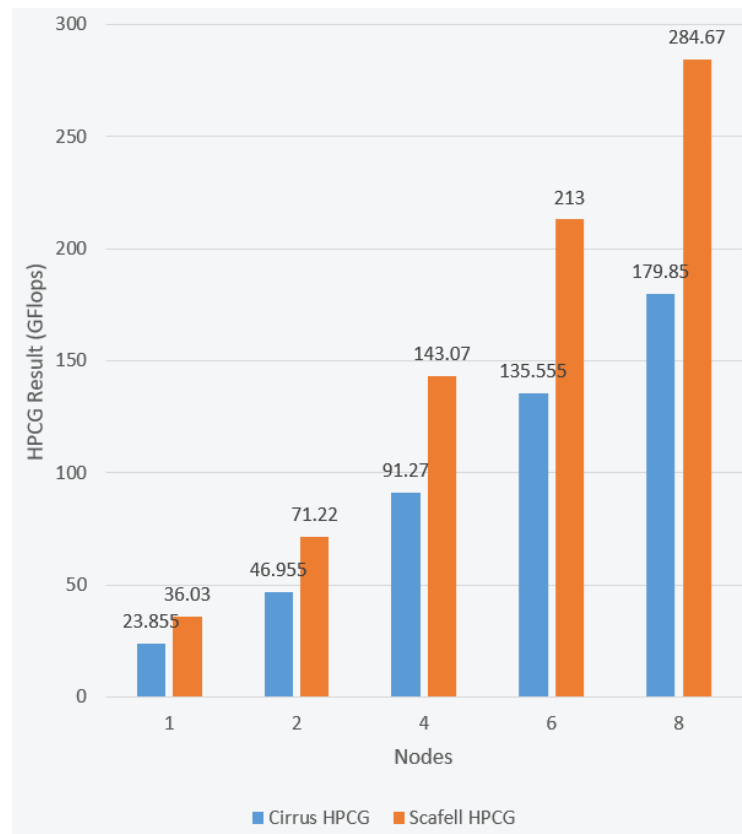


Figure 19 – HPCG results of the Cirrus and Scafell Pike systems across different number of nodes

On average, Xeon Gold 6142 is 58% better than Broadwell E5-2695 v4 for the HPL benchmark and 65% better for the HPCG benchmark. The comparison of the HPCG - HPL fraction of peak is presented in Figure 20.

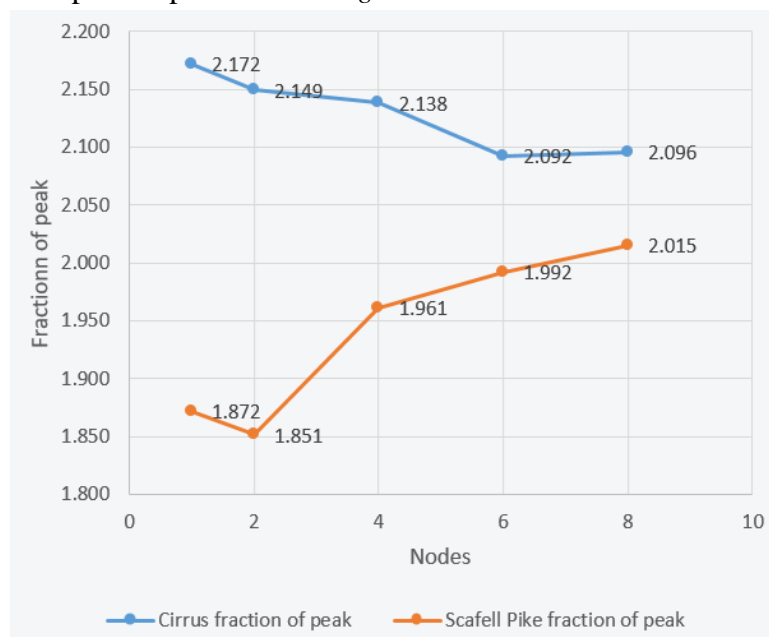


Figure 20 – Fraction of peak between Cirrus and Scafell Pike systems across different number of nodes

In one node, the Intel Broadwell processors from Cirrus has better fraction of peak than the Intel Scalable processors from Scafell Pike. This indicates that a dual socket Intel Broadwell node has better memory bandwidth performance than a dual socket Intel Scalable node since the HPCG benchmark stresses the memory. This also suggests that the DDR4 memory (should be 2400 MHz) is a bottleneck factor for Xeon Gold and hence it needs better a IMC (Integrated Memory Controller) design to accept higher DDR4 clock than 2400 and 2666 MHz (Xeon Platinum IMC).

Meanwhile, across 2 nodes and more the fraction of peak in the Scafell Pike tends to speed up but in the Cirrus it begins to decrease although overall Cirrus's fraction of peak is still higher than Scafell Pike. Probably this is because Scafell Pike machine has a better network interconnect than Cirrus since the HPCG benchmark also stresses the network component. Based on the specification in *Table 41*, Scafell Pike uses Infiniband EDR which indeed better than Infiniband FDR both in latency and bandwidth performance [44]. It means that the Infiniband FDR is the bottleneck factor in the Cirrus machine.

So far HPCG is able to recognize the memory and network bottleneck effect with a single node and scalability benchmark tests on homogeneous CPU system. The next part is to verify the memory and network performance with HPCC and CP2K benchmarks.

5.1.3 System Balance Analysis with HPCC

There are several modifications for the following HPCC tests:

- HPL 2.0 from the HPCC benchmark is replaced by the MKL binary to ensure the peak performance. Although it is run with hybrid MPI-OMP fashion, the following analysis will pretend it run with pure MPI process.
- The Ns size is adjusted to make sure every HPCC benchmarks reaches its peak performance per node.

Table 45 and *Table 46* present the HPCC benchmark results of Cirrus and Scafell Pike on different numbers of nodes. The HPCC was run with pure MPI process (one OMP thread per node) in order to scale better over large numbers of nodes. Overall, Scafell Pike shows better performance than Cirrus due to newer hardware and system.

Benchmark	1 node	2 node	4 node	6 node	8 node
HPL from MKL binary (TFlops)	1.1	2.18	4.27	6.48	8.58
MPI Processes	36	72	144	216	288
HPL MKL per MPI (GF)	30.51	30.34	29.64	29.99	29.8
G-PTRANS (GB/s)	12.19	16.88	17	30.19	36.34
G-Random Access (GUPS)	0.004	0.048	0.059	0.065	0.054
G-FFT (GFlops)	23	32.05	39.21	45.34	38.22
EP-STREAM Sys (GB/s)	115.2	248.4	495.36	743.04	990.72
EP-STREAM Triad (GB/s)	3.2	3.45	3.44	3.44	3.44
EP-DGEMM (GFlops)	25.02	28.7	28.61	28	26.83
Random Ring Bandwidth (GB/s)	1.38	0.35	0.19	0.18	0.14
Random Ring Latency (usec)	0.92	1.502	1.497	1.69	1.58

Table 45 – HPCC results of the Cirrus system

Benchmark	1 node	2 node	4 node	6 node	8 node
HPL from MKL binary (TFlops)	1.92	3.85	7.3	10.69	14.13
MPI Processes	32	64	128	192	256
HPL MKL per MPI (GF)	60.14	60.11	57	55.68	55.19
G-PTRANS (GB/s)	15.41	17.76	30.78	51.06	51
G-Random Access (GUPS)	0.015	0.038	0.056	0.065	0.07
G-FFT (GFlops)	30.02	48.72	74.25	74.47	138.995
EP-STREAM Sys (GB/s)	182.4	364.8	725.76	1065.6	1428.48
EP-STREAM Triad (GB/s)	5.7	5.7	5.67	5.55	5.58
EP-DGEMM (GFlops)	58.23	57.72	44.91	44.28	53.91
Random Ring Bandwidth (GB/s)	1.187	0.79	0.42	0.37	0.362
Random Ring Latency (usec)	0.92	1.469	1.655	1.71	1.6

Table 46 – HPCC results of the Scafell Pike system

The first investigation is the memory balance with embarrassingly parallel STREAM Triad (EP-STREAM) results. The memory balance is analysed through the correlation between HPL performance and EP-STREAM divided by HPL per MPI in TB/s – TFlops unit to express the memory balance ratio. The memory balance ratios for both systems are presented in *Table 47*, meanwhile *Figure 21* shows the correlation graph.

Node	Memory balance ratio EP-STREAM / HPL per MPI, TF/s - TFlops	
	Cirrus	Scafell Pike
1	3.776	3.033
2	8.186	6.069
4	16.712	12.733
6	24.775	19.136
8	33.247	25.885

Table 47 – Memory balance ratio between Cirrus and Scafell Pike system

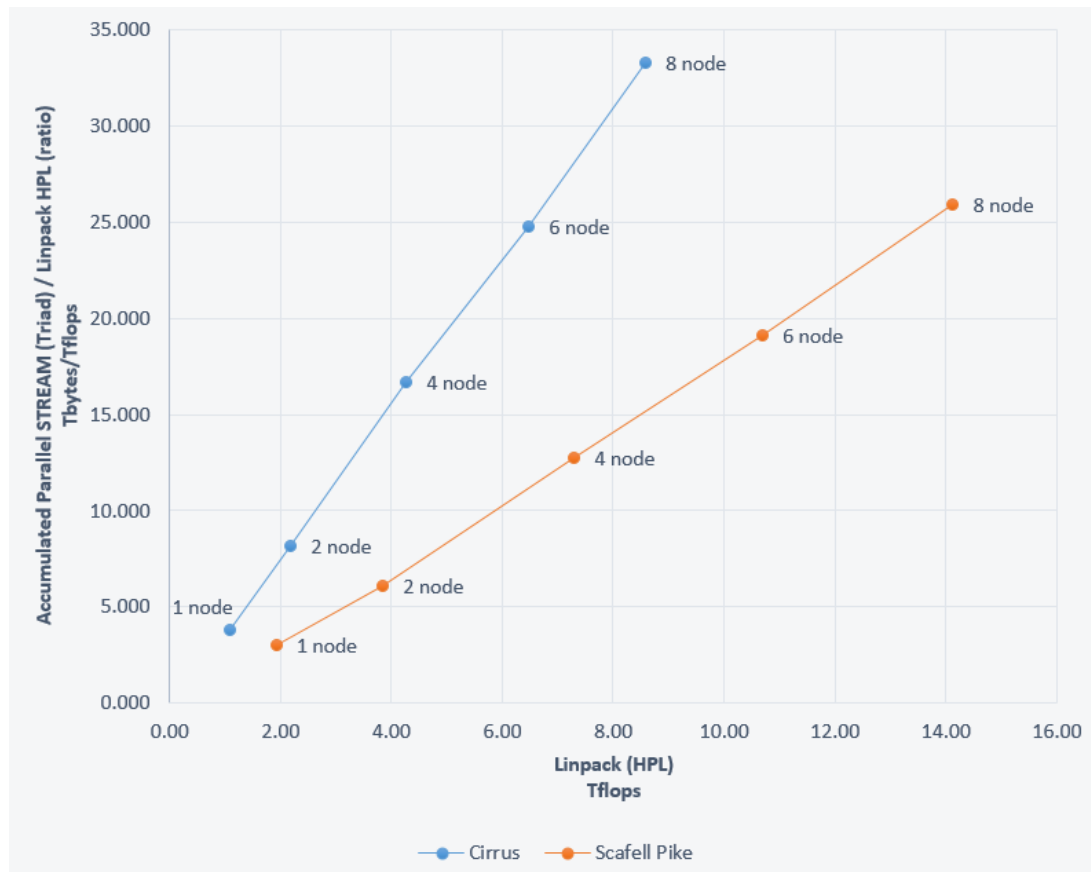


Figure 21 – Correlation balance between memory and processor performance

Overall, the Cirrus system shows a better memory balance ratio than the Scafell Pike system. The speedup of both systems are steady but Cirrus has better speedup begin from 2 nodes and so on. To perform below the perfect linear speedup, it seems DDR4 memory cannot follow the Xeon Gold HPL performance, in the other hand it's perfect for the Xeon Broadwell performance. At this point, both of the HPCG and STREAM

balance tests confirm that the Cirrus system has better memory performance and balance than the Scafell Pike system.

The next investigation is the network interconnect balance with Random Ring Bandwidth results. The correlation between HPL performance and Random Ring Bandwidth divided by HPL per MPI in GB/s – TFlops unit is formed to express the memory balance ratio. The network balance ratios for both systems are presented in *Table 48*, meanwhile *Figure 22* shows the correlation graph.

Node	Network Interconnect balance ratio Random Ring Bandwidth / HPL per MPI. GB/s - TFlops	
	Cirrus	Scafell Pike
1	45.235	19.736
2	11.535	13.143
4	6.41	7.368
6	6.002	6.645
8	4.698	6.56

Table 48 – Network balance ratio between Cirrus and Scafell Pike system

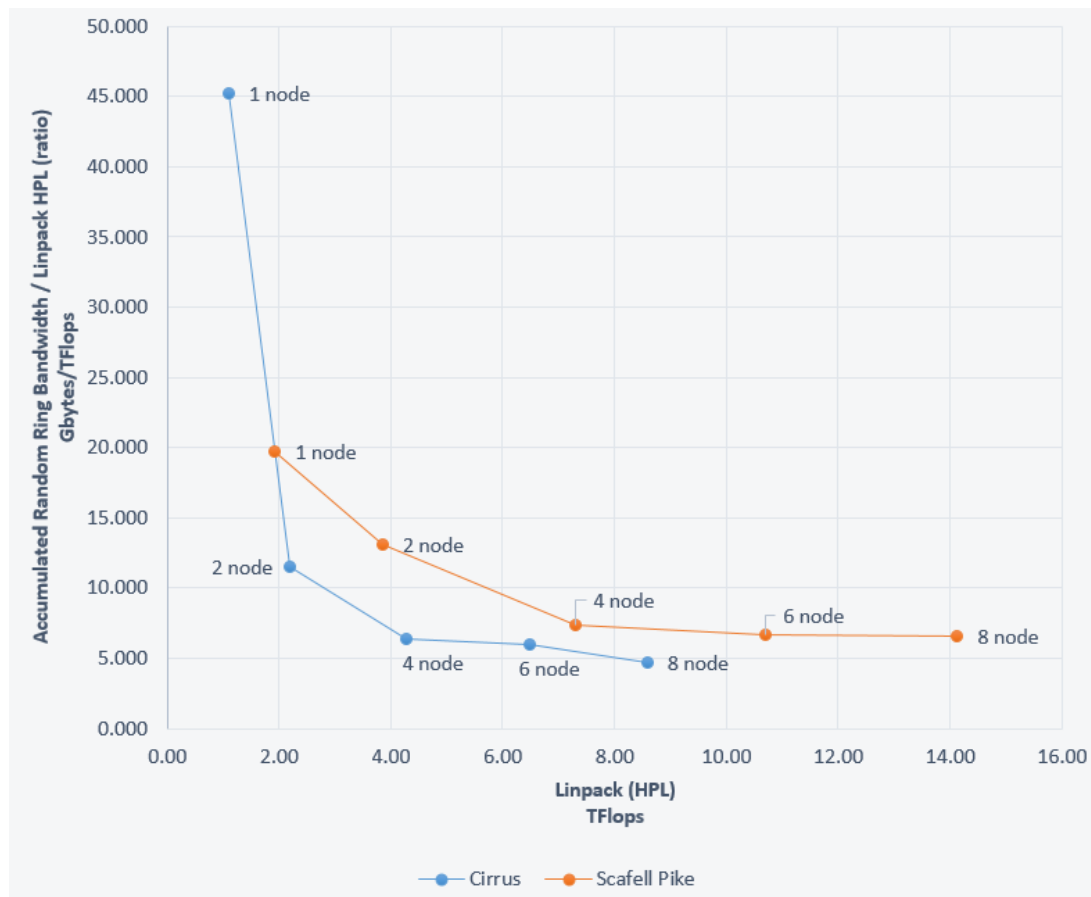


Figure 22 – Correlation balance between memory and processor performance

The Random Ring Bandwidth interconnect bandwidth will decrease as the number of nodes that communicate each other is increasing. On one node, the Cirrus system has a much better interconnect bandwidth than the Scafell Pike system (several re-runs confirmed the result is not a fluke) but it begins to dramatically decrease on two nodes and towards. The Scafell Pike system has a smooth bandwidth decrease which doesn't go down sharply like the Cirrus system. Begins from two nodes, the Scafell Pike system maintains higher network balance than the Cirrus system.

Assuming the Cirrus network balance ratio will fall along with the increase of HPL TFlops, means the Scafell Pike system has a better network balance than the Cirrus system. This correlation shows the capability of the Infiniband EDR interconnect performance compared to the previous Infiniband FDR. This correlation also can be used to explain the speedup pattern difference of the HPCG fraction of peak in *Figure 20* is because of the network balance difference among the two systems.

The last investigation is the FFTE balance over HPL performance with the correlation between HPL performance and FFTE divided by HPL per MPI in GB/s – TFlops. FFTE was chosen over DGEMM because this HPCC tests were run with pure MPI processes and FFTE favours MPI processes run (meanwhile DGEMM favours OMP threads). FFTE stresses both of the bandwidth - latency interconnect and memory between local and remote area which are similar to HPCG. The network balance ratios for both systems are presented in *Table 49*, meanwhile *Figure 23* shows the correlation graph.

Node	FFTE balance ratio FFTE / HPL per MPI. GB/s - TFlops	
	Cirrus	Scafell Pike
1	754.04	599.14
2	1056.45	810.54
4	1322.87	1302.64
6	1511.74	1337.36
8	1282.60	2518.65

Table 49 – FFTE balance ratio between Cirrus and Scafell Pike system

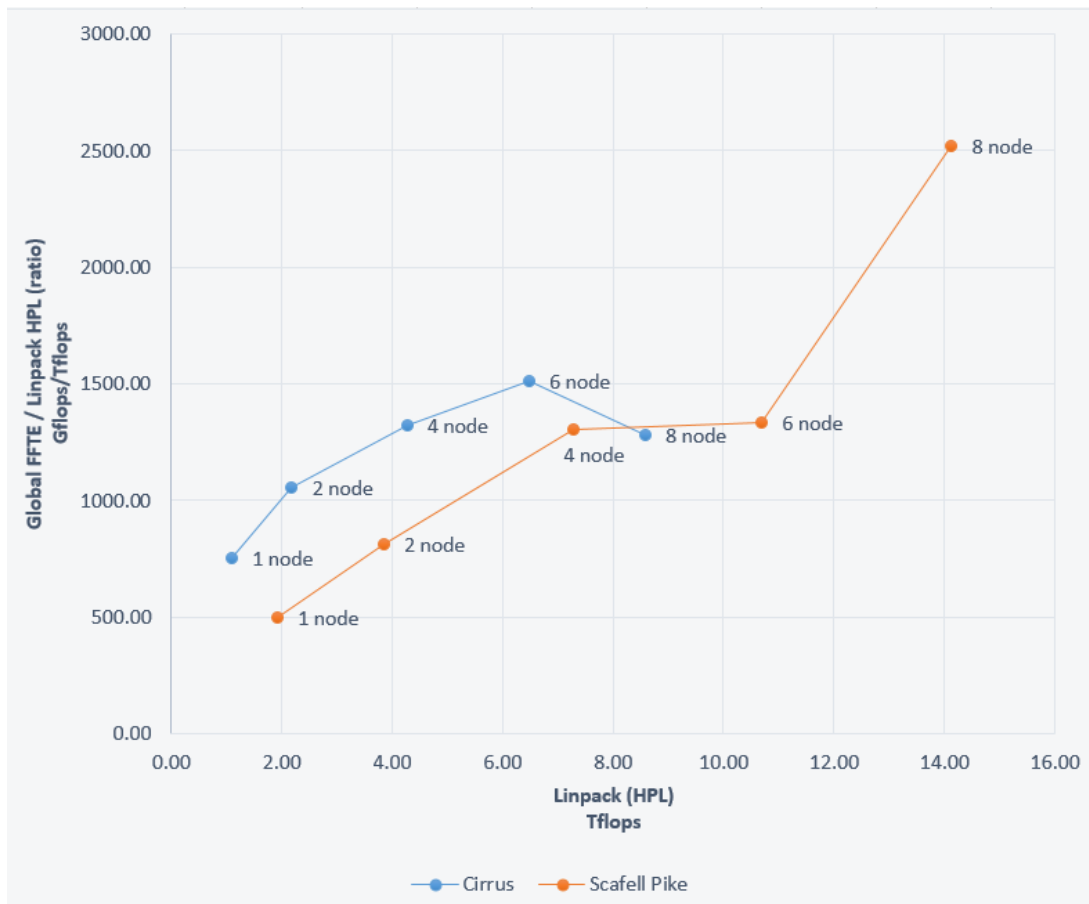


Figure 23 – Correlation balance between FFTE and processor performance

Between 1 to 6 nodes, the Cirrus system has a better FFTE balance than the Scafell Pike system but the speedup drops between 6 – 8 nodes. On the contrary, the Scafell Pike system gained steady speedup until 8 nodes. These patterns indicate FFTE benchmark is strongly related to the memory benchmark where network performance also plays an important role for multiple nodes run. In short, the FFTE balance test is the combination between network and memory balance tests. The Scafell Pike may gain steady speedup because of the better network balance, as opposite for the Cirrus which causes it to drop between 6 – 8 nodes. In contrast, the Cirrus keeps the FFTE balance higher than Scafell Pike in 1 – 6 nodes where the network bottleneck still doesn't affect the benchmark performance.

Both of the HPCG fraction of peak over HPL results and system balance investigations with HPCC conclude the same patterns: The Cirrus system has a better memory and balance performance whereas the Scafell Pike system is better in the network interconnect performance over large number of nodes. HPCG and the correlation between EP-STREAM from HPCC and HPL results suggest that DDR4 memory is the bottleneck factor for Intel Xeon Gold (and probably also Platinum since it's faster than Gold).

5.1.4 CP2K H2O-64 Comparison

CP2K is a quantum chemistry package and framework that performs atomic simulations, liquid, molecular dynamic, material, crystal, and biological system computations. CP2K is included in the PRACE benchmark suite and has the characteristic for heavily stressing the memory. This project used CP2K version 5.1 with psm (hybrid MPI-OpenMP) version. There are several benchmarks included inside the CP2K package where H2O-64 was chosen because of good scalability and fairly quick processing time.

CP2K on the Scafell Pike machine was compiled with Intel MPI and MKL 18.1.163 meanwhile in the Cirrus it was a mixed up between Intel MPI 17.2.096 and MKL 16.0.0 because of the problem with MKL 17.2.096 (the CP2K benchmark couldn't be run over 36 MPI processes and returned with 'KS energy is an abnormal value (NaN/Inf)' error). The benchmark was run three times in a row to ensure the consistency of results. Both of Cirrus and Scafell Pike ran the benchmark with the power of two number of MPI processes to maintain the benchmark parallel decomposition without any error. OMP threads more than one didn't improve the performance so it was set to one. The results for both systems are presented in *Table 50* where *Figure 24* illustrates the speedup.

Nodes	MPI Procs	Running Time (second)	
		Cirrus	Scafell Pike
1	32	63.65	55.25
2	64	44.62	30.72
4	128	30.78	23.81
6	192	Fail	23.88
8	256	52.08	19.10

Table 50 – CP2K results in the Cirrus and Scafell Pike systems

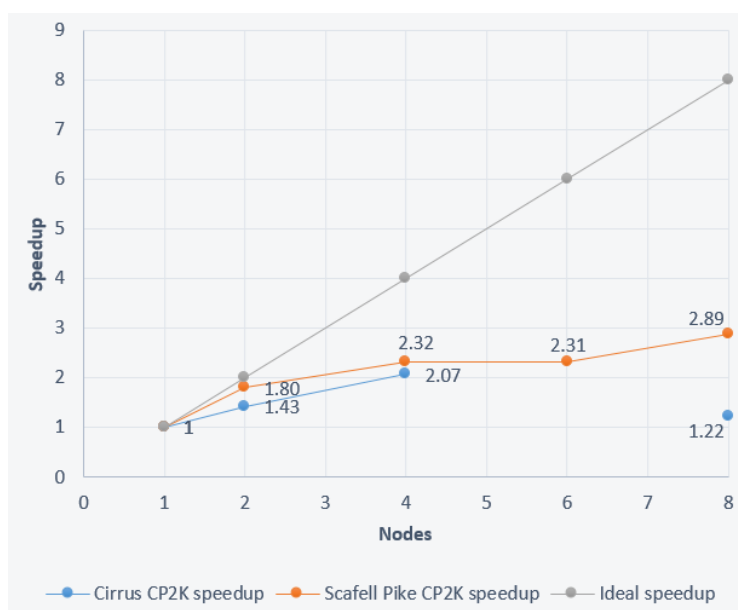


Figure 24 – Comparison of the CP2K speedup between the Cirrus and Scafell Pike

The Scafell Pike has better results and scalability than Cirrus. The benchmark at six nodes of Cirrus always return an error: “WARNING in qs_scf.F:539 :: SCF run NOT converged”, probably due to the MKL error. Cirrus with 8 nodes performed CP2K with inconsistent results. Possibly the different MPI-MKL versions at the Cirrus causes the benchmark to run with poor performance over multiple nodes.

However if we evaluate the benchmark result with one node only, there is only a little gap between a 36 MPI run at the Cirrus with a 32 MPI run at the Scafell Pike. *Figure 25* presents the comparison of the benchmark performance within 1 node for both systems.

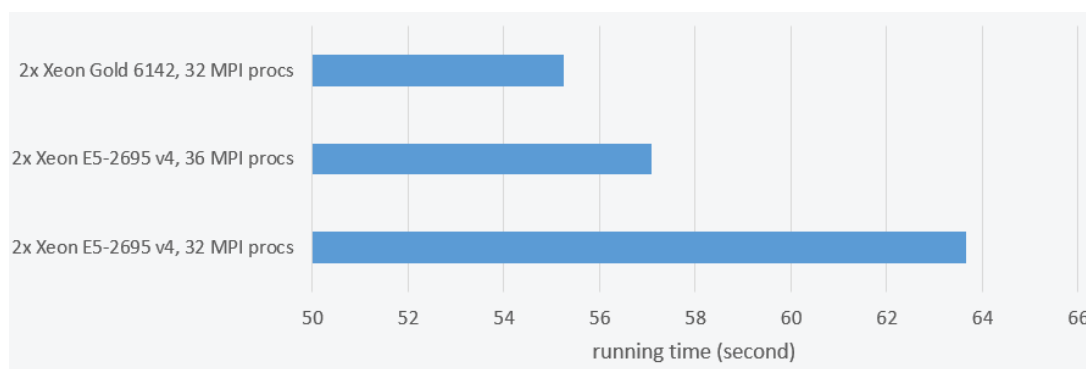


Figure 25 – Comparison of 1 node performance at the CP2K. N.B: the x-axis running time does not start at zero.

Intel Xeon Gold 6142 only gained a little speedup over Intel Xeon Broadwell E5-2695 v4 in the CP2K benchmark at one node despite of the huge difference on the HPL and HPCG results between the two processors. This little speedup result suggests the memory bottleneck at the Xeon Gold greatly affects the CP2K performance. The Scafell Pike systems had a good scalability performance more likely because of the correct MPI-MKL version and better interconnect performance.

5.2. Benchmark Comparison on NVIDIA P100 and V100

The benchmarks in the P100 GPUs were run before the competition when we were given an access from Boston Limited to the temporary one node server with six P100 GPUs. Here the results of the P100 HPL and HPCG benchmarks are compared with the V100 results. To ensure the fairness of the benchmark, same binaries were used: HPL OMP 1.10.2 – CUDA 9.0 and HPCG 3.1 NVIDIA. NVIDIA driver 396.26 was installed for both GPUs.

The P100 HPL benchmark run was investigated with the same fashion of the V100 benchmark investigations before the SCC. Ns 110000 with NB 384 was the best settings for six P100s which made them to run with 100% load and finally produced the peak performance. As for the HPCG, size problem 256^3 was used in both GPUs.

Figure 26 and Figure 27 show the HPL and HPCG results respectively whereas Figure 28 shows the fraction of peak result.

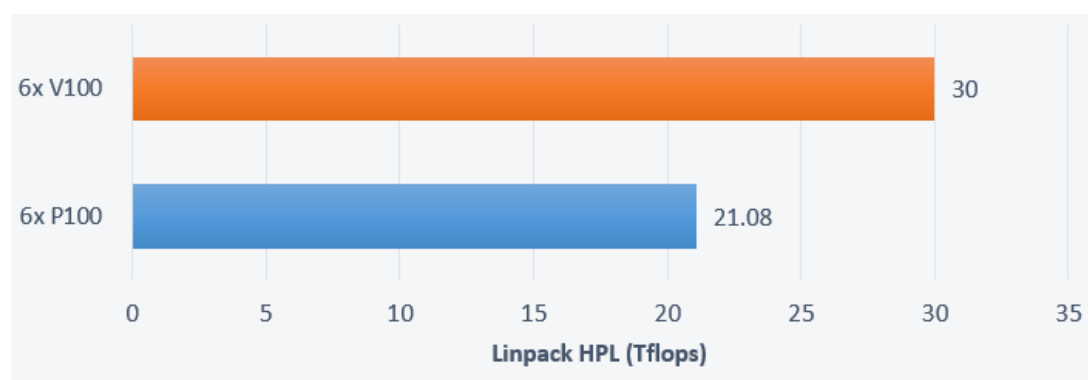


Figure 26 – HPL results comparison of six P100s and V100s

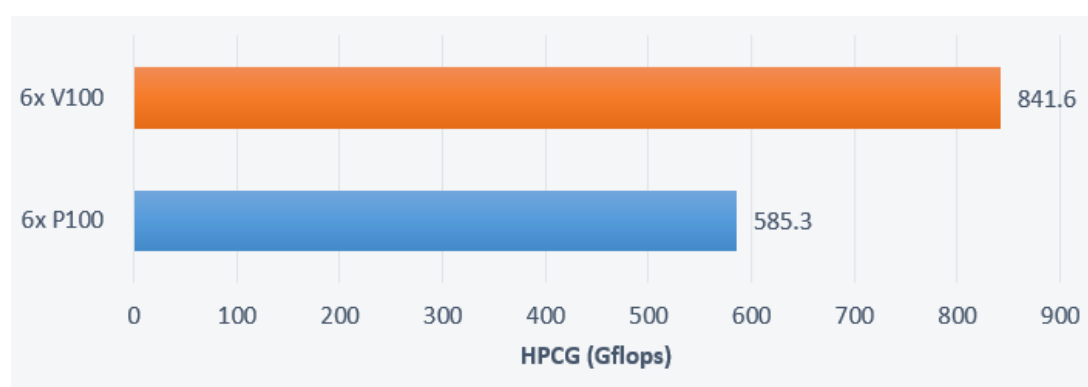


Figure 27 – HPCG results comparison of six P100s and V100s

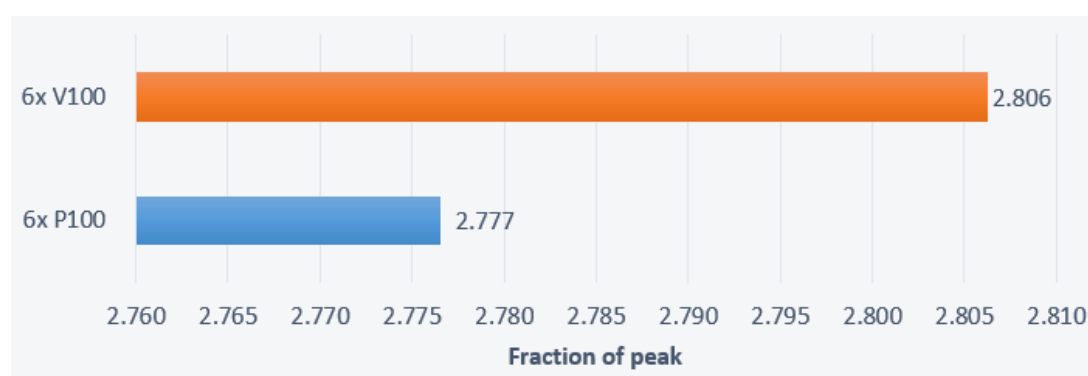


Figure 28 – Fraction of peak comparison of six P100s and V100s. N.B: the x-axis fraction of peak does not start at zero.

The NVIDIA V100 has better performance than P100 with 1.42x HPL speedup and 1.43x HPCG speedup. The V100 has a bigger fraction of peak than P100, meaning V100 is more efficient than P100 in terms of performance efficiency and demonstrates the NVLink interconnect improvement. In addition, the same HBM2 memory is still be able to support the Volta architecture performance without causing any bottleneck effect or any means that could reduce the fraction of peak compared to the P100.

Now all of the fraction of peak data is complete. The last investigation is to compare the fraction of peak between homogeneous CPU and heterogeneous GPU systems. For fair comparisons without any cluster interconnect effect, all of the systems are compared in one node only. *Figure 29* shows the comparison of the fraction of peak of all systems that have been benchmarked before.

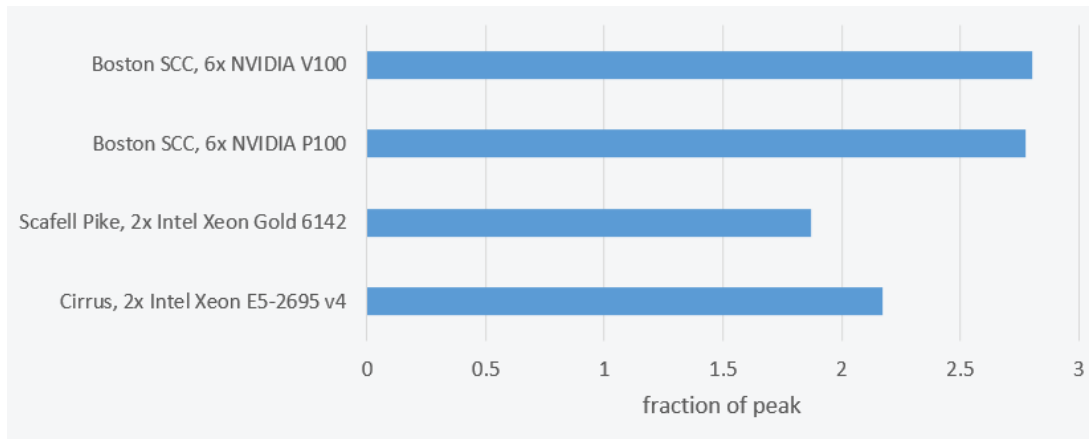


Figure 29 – Comparison of the fraction of peak between all systems

Both of the GPU systems have a better fraction of peak than both of the CPU systems. The NVIDIA GPU systems may have better efficiency because of the NVIDIA NVLink interconnect performance compared to the Intel UPI (Ultra Path Interconnect) in Xeon Scalable and QPI (Quick-Path Interconnect) in Xeon Broadwell. We are probably now going to the right track with the current trends where newer HPC systems slowly shifting from homogeneous CPU systems to heterogeneous CPU-GPU systems (including ARM system as is shown in *Figure 30*).



Figure 30 – New Bull-Atos Sequana X1000 at the SCC 18. Now they are pairing the ARM processors with NVIDIA Tesla V100

5.3. Comparison Conclusions and Future Probabilities

Through some analysis from the HPCC correlation benchmarks and CP2K it can be concluded that HPCG fraction of peak over HPL ratio is proven to become a new evaluation for the HPC system balance. In addition, the comparison of the fraction of peak between one node and scalability testing is able to determine the possibilities of the bottleneck part. There are several important observed points and early assumptions from the project:

- DDR4 memory more likely couldn't follow the performance of Xeon Scalable Gold (and Platinum) processors. The next Xeon Scalable development should consider to equip the Scalable processors with better IMC and ECC RAM with higher clock / lower latency.
- Memory and network together as one unity contribute to balance the system performance between synthetic and actual scientific code performance.
- GPU-based systems overall have better performance, power efficiency, and performance efficiency than a CPU-based systems. The HPC world will have a turning-point in the future when heterogeneous systems dominate homogeneous systems. The other possibility is that SoC (System-on-Chip) CPU-GPU takes over processor as an alternative from GPGPU or accelerator in the HPC system to improve the power and performance efficiency.
- The current HPC community is moving in the right track to fully develop the heterogeneous system for the next HPC systems generation and Exaflop supercomputer.

Chapter 6

Conclusion and Future Works

6.1. HPCG Fraction of Peak: The Future of HPC Hardware Evaluation

In this project, various homogeneous CPU and heterogeneous CPU-GPU systems are evaluated with the HPL benchmark as a ‘compulsory’ benchmark for any HPC system. In addition, HPCG, HPCC, and CP2K benchmarks are also tested in order to augment the HPL benchmark. Based on the results and comparisons, the three benchmarks served their purposes to augment the HPL result. This project was considered successful in the early stage where the tests produced several interesting points about performance balance other than ‘which system is faster’ and contradict results that may be useful for the next hardware development or investigation.

Benchmark correlations and analysis from the HPCC benchmark between the interconnect (Random Ring Bandwidth) and memory (EP-STREAM) performance over HPL result conclude that the HPCG fraction of peak over HPL ratio is reasonable to evaluate the HPC system balance. HPCG is proved as a suitable one binary benchmark to augment HPL benchmark. The author summarises simple steps to use HPCG for evaluating the HPC system balance:

- The HPL and HPCG should be the optimised binary for the specific hardware.
- Comparison of the HPCG fraction of peak over HPL at one node only to see the balance between the CPU/GPU interconnect (if one node contains multiple processor sockets or GPUs) and memory performance.
- Scalability test over multiple nodes to see the interconnect performance. If the fraction of peak scales up it means the system has good interconnect performance.

Perhaps this is the time to promote a Top500 based on the HPCG fraction of peak to recognize the efficiency of the system and effort of the HPC companies behind the system design. HPCG website quietly posts the HPCG ranking results with the fraction of peak since 2016 but it seems doesn’t gain much attention in the HPC community.

6.2. Student Cluster Competition: a Life-time Experience

Being part of the Team EPCC to endeavour the Student Cluster Competition awards and come from a country without any proper HPC facility was truly a once in a life-time experience. Vast amount of experience and knowledges were gained from

configuring and optimising the cluster from zero. Boston Limited allegedly gave us the opportunity to configure the two nodes cluster together. Be able to work with a high-end NVIDIA Tesla V100 cluster with sudo access was also a great amount of experience since normally it is restricted to the system admin only. It was also a great opportunity to attend the ISC and observe all of the current and newest HPC hardware. The author managed to visit all of the booths and gained so many important knowledges from the HPC experts and industries there. The most important thing were to make connections to the HPC actors and know how the HPC industry works.

Perhaps the true definition for once in a life-time experience at the SCC was to re-configure and optimise an asymmetrical cluster with mixed hardware (V100 16GB and 32GB). The greatest moment is where we could manage to optimise the benchmarks in the half-broken cluster during a chaotic 8 hours before the competition began. We initiated many unthinkable attempts to make the benchmarks work at the SCC including setting up two different clocks in the GPUs, manually adjusting the PWM fan speed to tame the peak power consumption, binding the MPI processes into specific number of cores, and modifying the run scripts. We observed and were surprised at how well an asymmetrical and mixed hardware cluster was still be able to produce similar performance with a normal cluster, together with how supple the OpenMPI was to distribute the works in such a cluster.

6.3. Future Work and Recommendations

There are several recommendations for the next year Team EPCC:

- It would be interesting to fully tune the HPCC benchmark such as porting the FFTE code into NVIDIA GPU with correct validation result, tuning the Random Access and Effective Bandwidth according to the cluster network, re-investigate about swapping the HPL HPCC into HPL NVIDIA binary, and if possible tuning the PTRANS.
- If NVIDIA V100 is still being used for the next year SCC, 16 GPUs should be able to bring the cluster into at least 50 TFlops at below 3000W. The ‘extreme’ downclocking from 1380 MHz to 900 MHz for example doesn’t sacrifice the V100 performance too much.
- Use a job batch scheduler such as SLURM so all of the team members could work without asking the availability of the cluster first. However the login node should be a low-power PC like an Intel NUC.
- Based on the experience with GRID at the competition, make sure to double the storage capacity to 2 TB per node at least in order to accommodate the benchmarks with huge output results.

The author believes that the performance efficiency experiments above are still in the early stage without more in-depth tests to support the current results. There are several recommendations for the future projects:

- Use the PRACE benchmark suite that consists of various real scientific benchmarks to augment these project results.
- Compare the fraction of peak in the HEDT platform such as Core i9 or AMD Threadripper with various RAM frequencies. The test should be done on X299 or X399 motherboard which is able to tune the RAM frequency and latencies. The goal is to see the effect of RAM performance in regards of the fraction of peak ratio. Samsung B-die DRAM (commonly found in 2x8 GB DDR4 RAM with XMP over 3000 MHz) is recommended for this experiment because of the capability to be tuned into any frequency and timings.

Two methods are available to tune the RAM latency: higher frequency with loose timings (e.g. 3866 MHz 15-15-15-30) or lower frequency with tight timings (e.g. 3600 MHz 12-12-12-28) for X299/399 platforms.

Appendix A

Oddities of the KNL benchmarking at the ARCHER and Scafell Pike KNL Nodes

A.1 HPL Benchmark

One of the original plans was to include the Intel KNL from EPCC ARCHER in the accelerator comparisons with NVIDIA GPUs. ARCHER is a CRAY XC-30 system which has eight KNL nodes with Intel KNL 7210 1.3 GHz, 64c/256t at each node. HPL and HPCG from Intel MKL 17.4.196 were used because ARCHER KNL only has Intel environment 2017 installed. KNL nodes from the Scafell Pike system was also tested because it has the same KNL type (7210). The Scafell Pike KNL run the HPL and HPCG from Intel MKL 18.1.163 version. Both of the KNL systems were run in flat memory mode to ensure the maximum performance.

The first investigation was to test the HPL in one node. HPL.dat below was used for a quick test with NB 332 (said to be the best for the KNL [46]) and Ns 100,000. The benchmark was run with pure 64 OMP threads with 1 MPI process (the ideal run for the Intel HPL). The result should be somewhere near 1900 GFlops [46], however the result was only 1500 – 1520 GFlops. In the Scafell Pike system it was even worse which was only 1000 – 1100 GFlops. 128 and 256 OMP threads were tested but the result was lower than the 64 OMP threads run. Based on the HPCG test, pure OMP run at the Intel MKL benchmarks is not always bring the KNL into its peak performance. Various combinations of the MPI processes and OMP threads were tested but the pure OMP run was the best for HPL.

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out          output file name (if any)
6                device out (6=stdout,7=stderr,file)
1                # of problems sizes (N)
100000           Ns
1                # of NBs
332             NBs
0                PMAP process mapping (0=Row-,1=Column-major)
1                # of process grids (P x Q)
3               Ps
2               Qs
16.0            threshold
1                # of panel fact
0               PFACTs (0=left, 1=Crout, 2=Right)
1                # of recursive stopping criterium
8               NBMINs (>= 1)
1                # of panels in recursion
2               NDIVs
1                # of recursive panel fact.
0               RFACTs (0=left, 1=Crout, 2=Right)
1                # of broadcast
2               BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
```

1	# of lookahead depth
0	DEPTHS (>=0)
0	SWAP (0=bin-exch,1=long,2=mix)
192	swapping threshold
1	L1 in (0=transposed,1=no-transposed) form
1	U in (0=transposed,1=no-transposed) form
0	Equilibration (0=no,1=yes)
8	memory alignment in double (> 0)

Other investigations were to tune the NB and BCAST variables. BCAST tuning didn't offer additional performance. NB tuning however, the ARCHER KNL gained more performance with NB 256 to 1650 – 1660 GFlops. Another NB tested like 220, 280, and 300 but turned out the best was NB 256. In the Scafell Pike KNL, NB 256 decreased the performance to below 1000 GFlops and utterly out of the question. The problem size of Ns was also tested to below and over 100,000 but no additional performance gained (100,000 – 110,000 is the maximum Ns for one KNL).

Other HPL versions from various MKL distributions were tested (from 2017 update 0, 2, 4 and 2018 update 0, 1, 3) but none of them could bring the KNL to the expected 1800 GFlops. The author also tried to compile the HPL-2.2 Netlib version with the same Intel MPI/MKL versions and optimisation flag from the HPCC but the results were even worse.

From interactive session at both systems, both of the KNL in the ARCHER and Scafell Pike has four-hyperthreads enabled per core. The Scafell Pike KNL has turbo-boost enabled to 1.5 GHz meanwhile in the ARCHER it is disabled (which adds more oddities where the HPL result in the ARCHER is higher than Scafell Pike). Intel OMP KMP affinity was set to scatter to make sure each thread run on a unique core. The big probability of the poor HPL result is because the hyperthreading is on. Usually the ideal environment to run the HPL benchmark at Intel processors is with hyperthreading off.

Under an assumption if the turbo-boost in the ARCHER KNL is enabled (1.5 GHz), compared to the 1.3 GHz HPL result it will become 1900-ish GFlops and close to the expected 1900 GFlops as is shown below:

$$1.5 \text{ GHz GFlops perf} = \frac{1.5 \text{ GHz}}{1.3 \text{ GHz}} \times 1653.59 \text{ Gflops} = 1907.99 \text{ GFlops}$$

On Scafell Pike the suspect is a combination of hyperthreading enabled or possibly the binary is not using the KNL AVX-512 instruction.

The initial plan was to use the HPL 1.6 TFlops performance at the ARCHER and run it within six nodes. However the run failed with error messages that stated it needs six processes to run the benchmark although it was already run with six MPI processes with P - Q = 2 – 3. This error happened at all of the HPL binaries including the self-compiled HPL 2.2 Netlib version, nonetheless the HPL benchmark could not be continued. *Table 51* summarises the best result of the HPL in both of the KNL systems.

System	HPLResult in one KNL (GFlops)
Ideal performance [46]	1895
ARCHER KNL Node	1653.59
Scafell Pike KNL Node	1151.11

Table 51 HPL KNL benchmark result comparisons

A.2 HPCG Benchmark

First, the HPCG test was initiated in one node only. Both of the systems run the Intel HPCG binary which was optimised for AVX-512 KNL. In contrast with HPL, HPCG in the KNL needs either 2 or 4 MPI processes to achieve peak result. The number of OMP threads should be adjusted so the benchmark run could fill 64 cores or 128 threads at maximum.

The same thing happened at the HPCG benchmark when the ARCHER KNL produced poor results. After various combinations of the MPI processes – OMP threads and various problem size, the ARCHER KNL could only produce 32.86 GFlops at its best. There is a previous member of the last year Team EPCC who did the HPCG benchmark at the ARCHER KNL and the result was 40.82 GFlops [22]. Based on Intel [47], the HPCG result for the KNL 7210 in flat mode should be 46.7 GFlops.

The result was contrast with the Scafell Pike KNL which was able to produce 54.91 GFlops with 4 MPI processes and 16 OMP threads, problem size 128^3 . The HPCG benchmark was run with 6 nodes at the Scafell Pike KNL and produced 251.094 GFlops, higher than the last year dissertation project who also did the HPCG benchmark in the 6 ARCHER KNL nodes with 224.848 GFlops result [22].

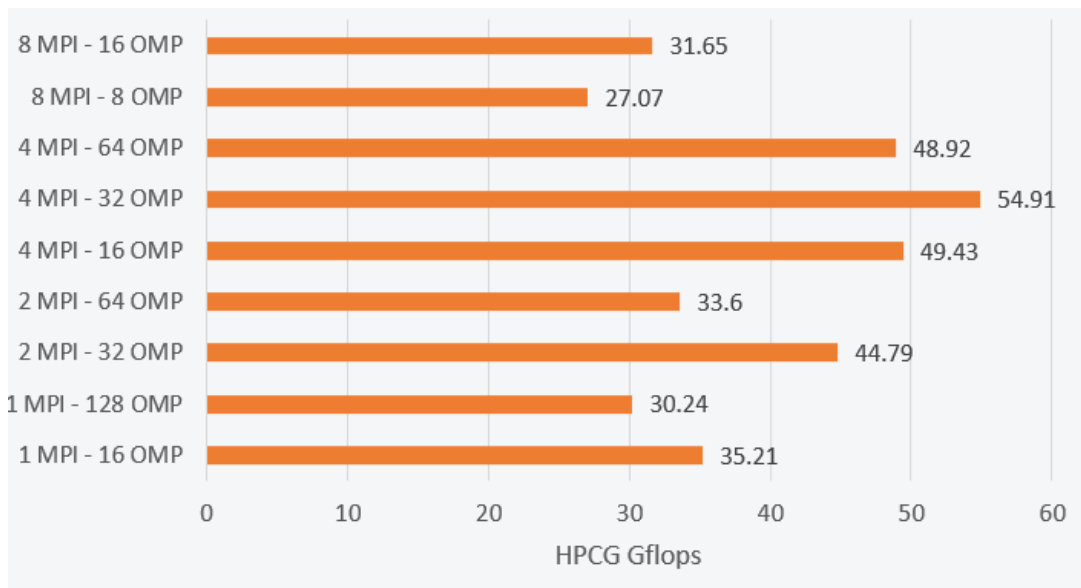


Figure 31 HPCG KNL investigations at the Scafell Pike KNL one node

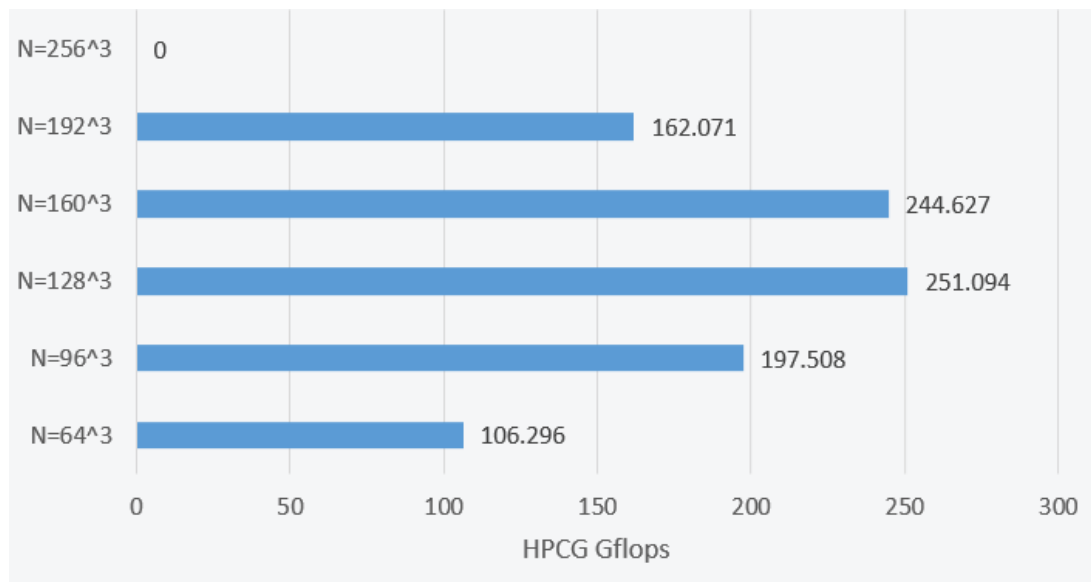


Figure 32 HPCG KNL 6 nodes results on the Scafell Pike KNL. $N=256^3$ failed to finish the benchmark

Table 52 summarises all of the HPCG results from various systems and sources.

System	HPCG result (GFlops)
Intel official result [46] 1 KNL	46.7
Last year dissertation result [22] 1 KNL	40.82
ARCHER 1 KNL	32.86
Scafell Pike 1 KNL	54.91
Last year dissertation result 6 KNL	224.848
Scafell Pike KNL 6 KNL	251.097

Table 52 List of the HPCG results from various systems

Appendix B

Raw Benchmark Records – EPCC Cirrus and Hartree-Centre Scafell Pike

B.1 Initial HPL.dat parameters

HPLinpack benchmark input file	
Innovative Computing Laboratory, University of Tennessee	
HPL.out	output file name (if any)
6	device out (6=stdout,7=stderr,file)
1	# of problems sizes (N)
65280	Ns
1	# of NBs
384	NBs
0	PMAP process mapping (0=Row-,1=Column-major)
1	# of process grids (P x Q)
2	Ps
1	Qs
16.0	threshold
1	# of panel fact
0	PFACTs (0=left, 1=Crout, 2=Right)
1	# of recursive stopping criterium
8	NBMINs (>= 1)
1	# of panels in recursion
2	NDIVs
1	# of recursive panel fact.
0	RFACTs (0=left, 1=Crout, 2=Right)
1	# of broadcast
2	BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1	# of lookahead depth
0	DEPTHS (>=0)
0	SWAP (0=bin-exch,1=long,2=mix)
192	swapping threshold
1	L1 in (0=transposed,1=no-transposed) form
1	U in (0=transposed,1=no-transposed) form
0	Equilibration (0=no,1=yes)
8	memory alignment in double (> 0)

B.2 HPL results

Testing the binary

Cirrus	Scafell Pike
starting point	starting point
xhpl_intel64_static	xhpl_intel64_static
Gflops 1090.72	Gflops 1821.09
Time 170.04	Time 101.84
xhpl_intel64_dynamic	xhpl_intel64_dynamic
Gflops 1078.5	Gflops 1921.08
Time 171.97	Time 96.54

BCAST investigations with the best binary version

Cirrus		Scafell Pike	
N = 55296		N = 55296	
BCAST	Gflops	BCAST	Gflops
0	1090.72	0	1805.29
1	1094.18	1	1913.82
2	1095.95	2	1922.11
3	1094.9	3	1842.57
4	1095.64	4	1818.78
5	1095.24	5	1912.66

NB with the best BCAST

Cirrus			Scafell Pike		
BCAST = 2			BCAST = 2		
NB	Gflops	Time	NB	Gflops	Time
128	938.82	197.55	128	1567.9	71.9
192	1029.58	180.14	192	1745	64.6
256	1009.14	183.79	256	1892.9	59.55
384	1095.95	169.23	384	1922.11	58.64
512	1018.65	182.07	512	1857.03	60.7

HPL run

Cirrus			Scafell Pike		
1 NODE					
Theoretical peak 1209.6			Theoretical peak 2663		
Best result 1098.27			Best result 1924.6		
1 node, P-Q = 2-1			1 node, P-Q = 2-1		
Ns	Gflops	Time	Ns	Gflops	Time
40704	1050.72	42.79	40704	1766.75	25.45
45312	998.15	62.14	45312	1874.53	33.09
50688	1087.34	79.85	50688	1791.98	48.45
55296	1077.91	104.57	55296	1922.11	58.64
60672	1053.82	141.29	60672	1924.6	77.37
65280	1095.95	169.23	65280	1921.08	96.54
70656	1074.41	218.88	70656	1818.37	129.33
75264	1091.6	260.39	75264	1923.7	147.76
80640	1077.69	324.4	80640	1886.72	185.3
85248	1095.01	377.18	85248	1844.11	223.97
90624	1098.27	451.79			
95232	1075.55	535.35			
2 NODE					
Perfect S(N) 2196.54			Perfect S(N) 3849.2		
Best result 2184.7			Best result 3846.9		
2 node, P-Q = 2-2			2 node, P-Q = 2-2		
Ns	Gflops	Time	Ns	Gflops	Time
65280	2125.63	87.25	60672	3492.56	42.63
70656	2100.46	111.96	65280	3512.13	52.81
75264	2127.93	133.58	70656	3571.23	65.85

80640	2119.07	164.98	75264	3625.34	78.4
85248	2114.55	195.32	80640	3577.95	97.71
90624	2154.49	230.31	85248	3616.26	114.21
95232	2154.13	267.3	90624	3651.04	135.9
100608	2118.27	320.5	95232	3718.51	154.85
105216	2161.82	359.21	100608	3665.82	185.2
110592	2184.7	412.76	105216	3665.84	211.83
115200	2148.46	474.4	110592	3846.9	234.41
120576	2133.64	547.74	115200	3662.07	278.32
			120576	3765.91	310.33
			125184	3673.29	356.05
4 NODE					
Ns = 110592			Ns = 110592		
P-Q	Gflops	Time	P-Q	Gflops	Time
2-4	4107.15	216.24	2-4	6947.13	129.8
4-2	4231.46	213.11	4-2	7133.05	126.42
Perfect S(N)		4393.08	Perfect S(N)		7698.4
Best result		4268.18	Best result		7295.94
4 node, P-Q = 4-2			4 node, P-Q = 4-2		
Ns	Gflops	Time	Ns	Gflops	Time
110592	4231.46	213.11	110592	7133.05	126.42
115200	4240.51	240.36	115200	6988.19	145.85
120576	4268.18	477.69	120576	7175.04	162.88
125184	4266.56	477.81	125184	7106.9	184.03
130560	4236.04	481.31	130560	7295.94	203.36
135168	4243.28	388	135168	7236.87	227.5
140544	4241.02	436.4	140544	7263.86	254.79
145152	4264.88	478.05	145152	7224.7	282.21
150528	4239.49	536.36	150528	7258.37	313.28
155136	4183.03	595.06	155136	7275.5	342.13
160512	4203.95	655.81	160512	presumably error	
6 NODE					
Ns = 120576			Ns = 130560		
P-Q	Gflops	Time	P-Q	Gflops	Time
3-4	6089.2	191.93	3-4	10049.2	147.67
4-3	6064.11	192.72	4-3	10218.2	145.2
Perfect S(N)		6589.62	Perfect S(N)		11547.6
Best result		6478.24	Best result		10691.4
6 node, P-Q = 3-4			6 node, P-Q = 4-3		
Ns	Gflops	Time	Ns	Gflops	Time
120576	6089.2	191.93	130560	10218.2	145.2
125184	6089.38	214.78	135168	10542.0	156.18
130560	6336.96	234.13	140544	10513.7	176.03
135168	6478.24	254.14	145152	10627.8	191.84
140544	6343.16	291.77	150528	10507.9	216.4
145152	6312.93	322.96	155136	10465.8	237.84
150528	6449.12	352.59	160512	10691.4	257.87
155136	FAIL		165120	10457.1	287.01
160512	6446.38	424.81	170496	10593.8	311.89
165120	6119.76	490.43			

8 NODE					
Perfect S(N)			8786.16		
Best result			8582.07		
8 node, P-Q = 4-4					
Ns	Gflops	Time			
135168	8362.27	196.89			
140544	8341.63	221.87			
145152	8370.24	243.58			
150528	8340.96	272.62			
155136	8508.27	292.56			
160512	6404.35	328.04			
165120	8532.03	351.77			
170496	8407.34	393.01			
175104	8559.94	418.15			
180480	8431.63	464.83			
185088	8499.88	497.32			
190464	8582.07	536.74			
195072	8288.57	597.06			
200448	8558.04	627.40			
205056	8290.00	693.39			
Perfect S(N)			15396.8		
Best result			14127.7		
8 node, P-Q = 4-4					
Ns	Gflops	Time			
160512	13739.7	200.66			
165120	14053.2	213.57			
170496	13588.0	243.16			
175104	13818.4	259.03			
180480	13928.9	281.37			
185088	14127.7	299.21			
190464	12744.9	361.42			

B.3 HPCG results

For an unknown reason, the HPCG in the Cirrus didn't want to run at the back-end with problem size other than 128. However after several tests in the front-end run, 128 was the best problem size for the Cirrus.

Cirrus						
Node	N=128^3		Average			
	1 st run	2 nd run				
1	23.82	23.89	23.855			
2	47.01	46.9	46.955			
4	91.37	91.17	91.27			
6	135.65	135.46	135.555			
8	179.41	180.29	179.85			
Scafell Pike						
Node	N=160^3		N=192^3		N=256^3	
	1 st run	2 nd run	1 st run	2 nd run	1 st run	2 nd run
1	35.94	35.4	36.32	35.74	35.21	35.24
2	69.6	70.22	71.19	71.24	70.25	70.26
4	141.31	141.56	141.19	141.16	143.04	143.1
6	202.17	204.22	207.64	207.29	213.09	212.9
8	277.33	278.32	277.34	276.65	284.89	284.45
Node	Gflops Average					
	N=160^3	N=192^3	N=256^3			
1	35.67	36.03	35.23			
2	69.91	71.22	70.26			
4	141.44	141.18	143.07			
6	203.20	207.47	213.00			
8	277.83	277.00	284.67			

B.4 CP2K results

Cirrus					
36 MPI per node					
Running time (sec)					
Nodes	average	1 st run	2 nd run	3 rd run	
1	57.090	57.14	56.98	57.15	
2	40.420	44.34	38.46	38.46	
4	28.330	28.37	28.05	28.57	
6	66.203	36.76	64.8	97.05	
8		fail			
32 MPI per node					
Nodes	average	1 st run	2 nd run	3 rd run	
1	63.65	64.85	63.05	63.05	
2	44.62333	41.1	46.42	46.35	
4	30.78	31.42	30.41	30.51	
6		fail, error below			
8	52.08333	65.36	45.82	45.07	
WARNING in qs_scf.F:539 :: SCF run NOT converged					
Scafell Pike					
Running time (sec)					
Nodes	average	1 st run	2 nd run	3 rd run	
1	55.253	55.465	55.062	55.231	
2	30.722	30.814	30.786	30.565	
4	23.812	24.769	23.392	23.274	
6	23.880	23.854	23.892	23.895	
8	19.104	19.014	19.087	19.212	

Appendix C

Actual SCC Benchmark Scripts and Input Parameters

C.1 HPL NVIDIA binary run scripts

Main run script - linpack_run_EPCCteam
<pre>#!/bin/bash #location of HPL HPL_DIR=`pwd` cp HPL.dat_2x1_1_node_dgx_p100x2 HPL.dat TEST_NAME=SCC18 DATETIME=`hostname`.`date +"%m%d.%H%M%S"` mkdir ./scc_output/HPL-\$TEST_NAME-results-\$DATETIME echo "Results in folder ./output/HPL-\$TEST_NAME-results-\$DATETIME" RESULT_FILE=./scc_output/HPL-\$TEST_NAME-results-\$DATETIME/HPL-\$TEST_NAME- results-\$DATETIME-out.txt # Make sure HT on both node is on sudo hyperthread 1 ssh epcc2 "sudo hyperthread 1" # Persitence mode enabled sudo nvidia-smi -pm 1 ssh epcc2 "sudo nvidia-smi -pm 1" # Node 1 clock 1155, Node 2 clock 1185 sudo nvidia-smi -ac 877,1155 # base clock for PCIE V100 ssh epcc2 "sudo nvidia-smi -ac 877,1185" # May the force be with you! /opt/openmpi_3.1.0_verbs/bin/mpirun -np 9 -H epcc1:5,epcc2:4 --mca btl openib,self,vader -bind-to none -x LD_LIBRARY_PATH ./runLinpack_meta 2>&1 tee \$RESULT_FILE</pre>

To bind two affinity GPUs - runLinpack_meta
<pre>#!/bin/sh if [\$(hostname) == epcc1] then ./run_linpack_aff_EPCCTeam else ./run_linpack_aff_EPCCTeam_2 Fi</pre>

GPU affinity script, node 1 with 5 GPUs – run_linpacck_aff_EPCCTeam

```
#!/bin/bash

#location of HPL
HPL_DIR=`pwd`

CPU_CORES_PER_RANK=4

export OMP_NUM_THREADS=$CPU_CORES_PER_RANK
export MKL_NUM_THREADS=$CPU_CORES_PER_RANK
export LD_LIBRARY_PATH=$HPL_DIR:$LD_LIBRARY_PATH

export TRSM_CUTOFF=1000000
export GPU_DGEMM_SPLIT=1.0

APP=$HPL_DIR/xhpl_cuda9.2.88_mkl_2018_ompi_3.1.0_gcc485_sm35_sm60_sm70_5_18_18

echo "xhpl binary= $APP"

case ${lrank} in
0)
#ldd $APP
echo "host=$me rank= $OMPI_COMM_WORLD_RANK lrank = $lrank
cores=$CPU_CORES_PER_RANK bin=$APP"

#set GPU and CPU affinity of local rank
export CUDA_VISIBLE_DEVICES=0; numactl --cpunodebind=0 $APP

;;
1)
echo "host=$me rank= $OMPI_COMM_WORLD_RANK lrank = $lrank
cores=$CPU_CORES_PER_RANK bin=$APP"

#set GPU and CPU affinity of local rank
export CUDA_VISIBLE_DEVICES=1; numactl --cpunodebind=0 $APP

;;
2)
echo "host=$me rank= $OMPI_COMM_WORLD_RANK lrank = $lrank
cores=$CPU_CORES_PER_RANK bin=$APP"

#set GPU and CPU affinity of local rank
export CUDA_VISIBLE_DEVICES=2; numactl --cpunodebind=0 $APP

;;
3)
echo "host=$me rank= $OMPI_COMM_WORLD_RANK lrank = $lrank
cores=$CPU_CORES_PER_RANK bin=$APP"

#set GPU and CPU affinity of local rank
export CUDA_VISIBLE_DEVICES=3; numactl --cpunodebind=1 $APP

;;
4)
echo "host=$me rank= $OMPI_COMM_WORLD_RANK lrank = $lrank
cores=$CPU_CORES_PER_RANK bin=$APP"

#set GPU and CPU affinity of local rank
export CUDA_VISIBLE_DEVICES=4; numactl --cpunodebind=1 $APP

;;
esac
```

GPU affinity script, node 1 with 4 GPUs – run_linpack_aff_EPCCTeam_2

```
#!/bin/bash

#location of HPL
HPL_DIR=`pwd`

CPU_CORES_PER_RANK=4

export OMP_NUM_THREADS=$CPU_CORES_PER_RANK
export MKL_NUM_THREADS=$CPU_CORES_PER_RANK
export LD_LIBRARY_PATH=$HPL_DIR:$LD_LIBRARY_PATH

export TRSM_CUTOFF=1000000
export GPU_DGEMM_SPLIT=1.0

APP=$HPL_DIR/xhpl_cuda9.2.88_mkl_2018_ompi_3.1.0_gcc485_sm35_sm60_sm70_5_18_18

echo "xhpl binary= $APP"

case ${lrank} in
0)
#ldd $APP
echo "host=$me rank= $OMPI_COMM_WORLD_RANK lrank = $lrank
cores=$CPU_CORES_PER_RANK bin=$APP"

#set GPU and CPU affinity of local rank
export CUDA_VISIBLE_DEVICES=0; numactl --cpunodebind=0 $APP

;;
1)
echo "host=$me rank= $OMPI_COMM_WORLD_RANK lrank = $lrank
cores=$CPU_CORES_PER_RANK bin=$APP"

#set GPU and CPU affinity of local rank
export CUDA_VISIBLE_DEVICES=1; numactl --cpunodebind=0 $APP

;;
2)
echo "host=$me rank= $OMPI_COMM_WORLD_RANK lrank = $lrank
cores=$CPU_CORES_PER_RANK bin=$APP"

#set GPU and CPU affinity of local rank
export CUDA_VISIBLE_DEVICES=2; numactl --cpunodebind=1 $APP

;;
3)
echo "host=$me rank= $OMPI_COMM_WORLD_RANK lrank = $lrank
cores=$CPU_CORES_PER_RANK bin=$APP"

#set GPU and CPU affinity of local rank
export CUDA_VISIBLE_DEVICES=3; numactl --cpunodebind=1 $APP

;;
esac
```

C.2 HPCG NVIDIA binary run scripts

Main run script – boston_run_6v100.1

```
#!/bin/bash

# TeamEPCC, Student Cluster Competition 2018
# This script is to run the NVIDIA binary HPL

# Location of HPL
HPCG_DIR=`pwd`

export PATH=/usr/local/cuda-9.0/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-9.0/lib64:$LD_LIBRARY_PATH

echo " ----- Loaded module ----- "
module list
which mpirun

echo " ----- GPU: -----"
nvidia-smi
ssh epcc1 "nvidia-smi"

TEST_NAME=SCC18
DATETIME=`hostname`.`date +"%m%d.%H%M%S"`
mkdir ./scc-output/HPCG-$TEST_NAME-results-$DATETIME
echo "Results in folder ./scc-output/HPCG-$TEST_NAME-results-$DATETIME"
RESULT_FILE=./scc-output/HPCG-$TEST_NAME-results-$DATETIME/HPCG-
$TEST_NAME-results-$DATETIME-out.txt

# ----- HPCG.dat input -----
# Make sure to use up to 1800 sec running time for official run
cp hpcg.dat_256x256x256_3660 hpcg.dat
#cp hpcg.dat_256x256x256_60 hpcg.dat

# Don't forget to turn on HT
sudo hyperthread 1
ssh epcc1 "sudo hyperthread 1"

ssh epcc1 "export OMP_NUM_THREADS=4"
export OMP_NUM_THREADS=4

# ----- GPU clock settings -----
sudo nvidia-smi -pm 1
ssh epcc1 "sudo nvidia-smi -pm 1"

sudo nvidia-smi -acp 0
ssh epcc1 "sudo nvidia-smi -acp 0"

sudo nvidia-smi -ac 877,1380 # boost clock for NVLINK V100 32GB
ssh epcc1 "sudo nvidia-smi -ac 877,1380"

ssh epcc1 "module load mpi/openmpi-1.10.2-verbs cuda/9.0"
HPCG_BIN=xhpcg-
3.1_gcc_485_cuda90176_ompi_1_10_2_sm_35_sm_50_sm_60_sm_70_ver_10_8
_17

echo " ----- Running the HPCG benchmark! -----"
# Too many problem arghhhhhh holy molyyyyyy >_<

/opt/openmpi_1.10.2_verbs/bin/mpirun -np 9 -H epcc1,epcc2 -rf rankfile --
mca btl openib,self -bind-to none -x LD_LIBRARY_PATH=$LD_LIBRARY_PATH
$HPCG_DIR/$HPCG_BIN | tee ./scc-output/HPCG-$TEST_NAME-results-
$DATETIME/SCC18-$DATETIME-output.txt
```

C.3 HPCC 1.5.0 run scripts

Main run script – run_hpcc

```
#!/bin/bash

# TeamEPCC, Student Cluster Competition 2018
# This script is to run the NVIDIA binary HPL

# ----- Disable HT -----
# Please see hyperthread script to run this command
sudo hyperthread 0
ssh epcc2 "sudo hyperthread 0"

# ----- Env settings -----
MPI_PROCESSES=12
RANK_PER_NODE=6
export OMP_NUM_THREADS=4

export KMP_AFFINITY=scatter

/opt/intel_2018/compilers_and_libraries_2018.1.163/linux/mpi/intel64/bin/
mpirun -f host_2 -np ${MPI_PROCESSES} -ppn ${RANK_PER_NODE} ./hpcc

# -----Finish and turn on HT again -----
echo "HPCC benchmark finished!"
sleep 1
sudo hyperthread 1
ssh epcc2 "sudo hyperthread 1"
```

C.4 HPL NVIDIA binary input parameters (HPL.dat)

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
1            # of problems sizes (N)
134400      Ns
1            # of NBs
192         NBs
0           PMAP process mapping (0=Row-,1=Column-major)
1           # of process grids (P x Q)
3           Ps
3           Qs
16.0        threshold
1           # of panel fact
1           PFACTs (0=left, 1=Crout, 2=Right)
1           # of recursive stopping criterium
8           NBMINs (>= 1)
1           # of panels in recursion
2           NDIVs
1           # of recursive panel fact.
1           RFACTs (0=left, 1=Crout, 2=Right)
1           # of broadcast
2           BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1           # of lookahead depth
0           DEPTHs (>=0)
0           SWAP (0=bin-exch,1=long,2=mix)
192         swapping threshold
1           L1 in (0=transposed,1=no-transposed) form
0           U in (0=transposed,1=no-transposed) form
1           Equilibration (0=no,1=yes)
8           memory alignment in double (> 0)
```

C.5 HPCC input parameters (hpccinf.txt)

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
1            # of problems sizes (N)
110532       Ns
1            # of NBs
384          NBs
0            PMAP process mapping (0=Row-,1=Column-major)
1            # of process grids (P x Q)
3            Ps
4            Qs
16.0         threshold
1            # of panel fact
0            PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
8            NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
0            RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
1            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
0            DEPTHs (>=0)
1            SWAP (0=bin-exch,1=long,2=mix)
192          swapping threshold
1            L1 in (0=transposed,1=no-transposed) form
0            U in (0=transposed,1=no-transposed) form
0            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
##### This line (no. 32) is ignored (it serves as a separator). #####
0            Number of additional problem sizes for PTRANS
10000        values of N
0            number of additional blocking sizes for PTRANS
64           values of NB
```

C.6 Hyperthreading script

```
#!/bin/bash

if [ "$1" != "1" ] && [ "$1" != "0" ]
then
    echo "Select argument: 1 for HT on, 0 for HT off"
    exit
fi

for cpuid in {24..35}
do
    FILENAME="/sys/devices/system/cpu/cpu${cpuid}/online"
    temp=$(cat $FILENAME)
    if [ "$temp" != "1" ]
    then
        echo "$1" > $FILENAME
    fi
    echo "cpu${cpuid}: $(cat $FILENAME)"
done

for cpuid in {36..47}
do
    FILENAME="/sys/devices/system/cpu/cpu${cpuid}/online"
    temp=$(cat $FILENAME)
```

```

        if [ "$temp" != "$1" ]
        then
            echo "$1" > $FILENAME
        fi
        echo "cpu$cpuid: $(cat $FILENAME)"
    done

```

C.7 Quick script to read the HPCC results

```

# Ultra simple batch file to read HPCC benchmark result
# Made by Wilson Lisan
# v1.0
# 29 May 2018

echo '----- HPCC Result -----'
echo 'G-HPL (GFLOPS)'
grep "HPL_Tflops" hpccoutf.txt
echo ''
echo 'G-PTRANS (GB/s)'
grep "PTRANS_GB" hpccoutf.txt
echo ''
echo 'G-RandomAccess (GUP/s)'
grep "MPIRandomAccess_GUPs" hpccoutf.txt
echo ''
echo 'G-FFT (GFLOPS)'
grep "MPIFFT_Gflops" hpccoutf.txt
echo ''
echo 'EP-STREAM Triad (GB/s)'
grep "StarSTREAM_Triad" hpccoutf.txt
grep "CommWorldProcs" hpccoutf.txt
echo ''
echo 'EP-DGEMM (GFLOPS)'
grep "StarDGEMM_Gflops" hpccoutf.txt
echo ''
echo 'RandomRing Bandwidth (GB/s)'
grep "RandomlyOrderedRingBandwidth_GBytes" hpccoutf.txt
echo ''
echo 'RandomRing Latency (usec)'
grep "RandomlyOrderedRingLatency_usec" hpccoutf.txt
echo '-----'

```

Bibliography

- [1] IBM, “Breaking the Petaflop Barrier,” 2009. [Online]. Available from: <http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/petaflopbarrier/>.
- [2] DARPA, “ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems,” 2008. [Online]. Available from: <http://staff.kfupm.edu.sa/ics/ahkhan/Resources/Articles/ExaScale%20Computing/TR-2008-13.pdf>.
- [3] Top500.org, “Top500 List for July 2018,” 2018. [Online]. Available from <https://www.top500.org/lists/2018/06/>.
- [4] Jackson, A., “HPCG: Benchmarking Supercomputers,” 2015. [Online]. Available from <https://www.epcc.ed.ac.uk/blog/2015/07/30/hpcg>.
- [5] Lisan, W., “Rancang Bangun Prototipe Super Komputer Mini Dengan Kluster Raspberry Pi (bachelor final assignment, Universitas Gadjah Mada),” 2016.
- [6] HPCG, “HPCG, HPL, STREAMS,” [Online]. Available from: <http://www.hpcg-benchmark.org/custom/index.html?lid=158&slid=281>.
- [7] HPCG, “HPCG Benchmark,” [Online]. Available from: <http://www.hpcg-benchmark.org/index.html>.
- [8] HPCG, “June 2018 HPCG Results,” 2018 [Online]. Available from: <http://www.hpcg-benchmark.org/custom/index.html?lid=155&slid=295>
- [9] HPCC, “HPC Challenge Benchmark,” [Online]. Available from: <http://icl.cs.utk.edu/hpcc/>
- [10] CP2K.org, “About CP2K,” [Online]. Available from <https://www.cp2k.org/>.
- [11] Top500, “INTRODUCTION AND OBJECTIVES,” [Online]. Available from: <https://www.top500.org/project/introduction/>
- [12] Top500, “THE LINPACK BENCHMARK,” [Online]. Available from: <https://www.top500.org/project/linpack/>
- [13] NVIDIA, “NVIDIA Tesla V100 Tensor Core GPU,” [Online]. Available from: <https://www.nvidia.com/en-us/data-center/tesla-v100/>
- [14] “Intel,” [Online]. Available: <https://www.intel.co.uk/content/www/uk/en/processors/xeon/scalable/xeon-scalable-platform.html>
- [15] Dongarra, J., “An Overview of High Performance Computing and Benchmark Changes for the Future,” 2016. [Online]. Available from: https://www.nist.gov/sites/default/files/documents/2016/12/07/dongarra_nist_nsci_seminar.pdf
- [16] Luszczek, P. et al, “The HPC Challenge (HPCC) Benchmark Suite,” 2006. [Online]. Available from: <http://www.icl.utk.edu/files/publications/2006/icl-utk-290-2006.pdf>
- [17] HPCG, “June 2016 HPCG Results,” 2016. [Online]. Available from: <http://www.hpcg-benchmark.org/custom/index.html?lid=155&slid=288>
- [18] NVIDIA, “NVIDIA Tesla V100 GPU Architecture,” [Online]. Available from: <http://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf>
- [19] DELL, [Online]. Available from: http://en.community.dell.com/techcenter/high-performance-computing/b/general_hpc/archive/2018/06/22/new-nvidia-v100-32gb-gpus-initial-performance-results

- [20] Gelas, D.J and Cutress, I., “Sizing Up Servers: Intel’s Skylake-SP Xeon versus AMD’s EPYC 7000 – The Server CPU Battle of the Decade?,” 2017. [Online]. Available from: <https://www.anandtech.com/show/11544/intel-skylake-ep-vs-amd-epyc-7000-cpu-battle-of-the-decade/7>
- [21] Shrout, R., “Computex 2018: Intel demos 28-core Processor at 5 GHz,” 2018. [Online]. Available from: <https://www.pcper.com/news/Processors/Computex-2018-Intel-demos-28-core-processor-5-GHz>
- [22] Mappoura, A., “Comparing Performance and Power Consumption on Different Architectures,” 2017. [Online]. Available from: https://static.epcc.ed.ac.uk/dissertations/hpc-msc/2016-2017/Andriani_Mappoura-Andriani_Mappoura-MSc-Dissertation-Report.pdf
- [23] Olds, D., “What are Student Cluster Competition, anyway? A Field Guide”, [Online]. Available from: <http://www.studentclustercomp.com/history/>
- [24] “Boston,” [Online]. Available: <https://www.boston.co.uk/default.aspx>
- [25] HPC.AI Advisory Council, “Rules,” [Online]. Available from: <http://hpcadvisorycouncil.com/events/student-cluster-competition/Rules/>
- [26] “GitHub,” [Online]. Available: <https://github.com/paboyle/Grid>
- [27] “Nektar++,” [Online]. Available: <https://www.nektar.info/features/>
- [28] “NEK5000,” [Online]. Available: <https://nek5000.mcs.anl.gov/>
- [29] Olds, D., “Student Cluster LINPACK Record Shattered! More LINs Packed Than Ever Before!” 2017 [Online]. Available from: <https://www.hpcwire.com/2017/11/16/student-cluster-linpack-record-shattered-lins-packed-ever/>
- [30] Kennedy, P., “Dual Xeon Gold 6150 Initial Benchmarks – Delving into Gold 6100” 2017 [Online]. Available from: <https://www.servethehome.com/dual-intel-xeon-gold-6150-initial-benchmarks/>
- [31] DELL, “DELL Solution Performance Analysis, Whitepaper: Performance and Energy Efficiency of Dell PowerEdge Servers with E5-2600 v4.” 2016 Version 1.1
- [32] Dongarra, J.J., et al, “The LINPACK Benchmark: Past, Present, and Future” 2001 [Online]. Available from: <http://www.netlib.org/utk/people/JackDongarra/PAPERS/hpl.pdf>
- [33] “HPCG,” [Online]. Available: <http://www.hpcg-benchmark.org/>
- [34] Mouzakitis, K., “High Performance LINPACK and GADGET 3 Investigation” 2014 [Online]. Available from: <https://static.epcc.ed.ac.uk/dissertations/hpc-msc/2013-2014/High%20Performance%20LINPACK%20and%20GADGET%203%20investigation.pdf>
- [35] HPCG, “HPCG Software Releases,” [Online]. Available from: <http://www.hpcg-benchmark.org/software/index.html>
- [36] “PARKBENCH MATRIX KERNEL BENCHMARKS,” [Online]. Available from: <http://www.netlib.org/parkbench/html/matrix-kernels.html>
- [37] “FFTE: A Fast Fourier Transform Package,” [Online]. Available: <http://www.ffte.jp/>
- [38] McCalpin, J.D., “STREAM: Sustainable Memory Bandwidth in High Performance Computers,” [Online]. Available from: <http://www.cs.virginia.edu/stream/>
- [39] “Effective Bandwidth (b_eff) Benchmark,” [Online]. Available: https://fs.hlrs.de/projects/par/mpi/b_eff/

- [40] <http://icl.cs.utk.edu/hpcc/software/index.html>
- [41] HPCC, “Rules for Running the Benchmark,” [Online]. Available from: <http://icl.cs.utk.edu/hpcc/overview/index.html>.
- [42] “Cirrus,” [Online]. Available: <http://www.cirrus.ac.uk/about/hardware.html>
- [43] Hartree Centre, “Scafell Pike,” [Online]. Available from: <http://community.hartree.stfc.ac.uk/wiki/site/admin/sequana%20-%20further%20info.html>
- [44] DELL, [Online]. Available from: http://en.community.dell.com/techcenter/high-performance-computing/b/general_hpc/archive/2016/02/02/need-for-speed-comparing-fdr-and-edr-infiniband-part-1
- [45] PRACE, “Unified European Applications Benchmark Suite,” [Online]. Available from: <http://www.prace-ri.eu/ueabs/>
- [46] Colfaxresearch, “HPLinpack Benchmark on Intel Xeon Phi Processor Family x200 with Intel Omni-Path Fabric 100,” [Online]. Available from: <https://colfaxresearch.com/hpl-on-xeon-phi-x200/>
- [47] Kleymenov, A. and Park, J., “HPCG on Intel Xeon Phi 2nd Generation, Knights Landing,” 2016 [Online]. Available from: <http://www.hpcg-benchmark.org/downloads/sc16/sc16-hpcg-bof.pdf>