Student cluster competition SC19
Final Architecture Proposal

Tartu Team

September 2019

1 Hardware Configuration

1.1 Hardware description

Our final cluster configuration is 6 nodes: 4 CPU nodes and 2 GPU nodes.

	CPU node	**GPU node**
Processor	2x AMD 7742 @2.25GHz (128 cores)	1x AMD 7742 @2.25GHz (64 cores)
Accelerator	-	4x NVIDIA Tesla V100 32GB
Server model	Gigabyte R282-Z91	Gigabyte G292-Z20 Rev.A00
Memory	DDR4 128GB	DDR4 256GB
Storage	500GB NVME M.2 SSD	1TB SSD SATA
Interconnect	EDR Infiniband ConnectX-5	

Table 1: Hardware specification

The nodes are interconnected with an InfiniBand (IB) switch "Mellanox MLNX-OS MSB7800" and an Ethernet Switch "CISCO Catalyst WS-C3560". The IB switch provides a high rate of communication and low latency between the nodes. After having evaluated our IB connection with Perftest, we've turned IB to prefer the cores logically in its NUMA domain, set linux kernel parameters for improved IPv4 traffic performance and initiated the high performance profile of Mellanox adapters. The Ethernet switch is just a tool to access the cluster due to no wireless connection.

1.2 Power consumption and its management

Component	**Name**	**TDP(W)**	**Amount**	**Total(W)**
Accelerator	NVIDIA Telsa V100	250	8	2000
Processor	AMD 7742 @2.25GHz	225	10	2250
Motherboard	R282-Z91/G291-Z20	80	6	480
IB card	EDR IB ConnectX-5	16	6	96
Storage	NVME M.2 SSD	10	4	40
Storage	SSD SATA	4	2	8
Memory	32GB DDR4 RAM	12	12	384
IB switch	MLNX-OS MSB7800 IB Switch	136	1	136
Ethernet Switch	CISCO Catalyst WS-C3560	75	1	75
		Total		**5469**

Table 2: Power consumption of hardware components.

Table 2 lists the majority of our cluster components and their TDP, hence our system is estimated to have the maximum power consumption around 5.5k. However, this number is only a theoretical consumption

with every component under full load. The competition experience showed us that an actual power load is typically lower, because no application takes up all the resources at the same time. Nevertheless, there are several strategies we've got in our arsenal to cope with high power consumption.
To eliminate most of unnecessary power fluctuation, we have prepared ipmitool scripts either to facilitate a smooth fan speeds adjustment according to the components temperature and avoid throttling or to set them manually. Also, we used Gigabyte's BMC fan profile tools to manipulate temperature and power use of the system.
CPU application have a tendency to have rather stable power consumption in comparison to benchmarks, for example HPL, which favours the power peaking at certain points of its run-time. Our general approach with such problems is to implement power cups to tell the system not to go over the competition limit. In addition, in the extreme case scenario, our hardware configuration and substantial core number allows us to be flexible with CPU frequencies and tailor it to fit our need for each individual application.
All announced applications are CPU based, while benchmarks mainly utilise GPUs. From table 2, it is clear GPUs have the second largest power consumption potential. To get control over this watt usage, we can set a power limit for GPUs via Nvidia system management interface (nvidia-smi) as well as control GPUs frequencies.

1.3 Hardware justification

The cluster configuration is suitable for a wide range of applications, especially the ones in this competition. Our chosen hardware has 3 main advantages:

- A large core number allows us to scale the problems efficiently.
- High bandwidth and fast interconnect facilitate memory accesses and inter-communications.
- GPU-s with Tensor cores boosts up calculations, especially for AI applications.

, Additionally, the design of the ROME CPU, 7nm I/O lithography, give us an advantage in lowering power consumption while still maintaining the efficiency of calculations.

Besides those advantages, the current setup is also good for each application in the competition in different aspects:

- HPCG, HPL: Tesla V100 is currently one of the best GPU in the market for these benchmarks. Besides, our setup (4V100/node * 2 nodes) can avoid overload of PCI lanes thus improve sending data speed between CPU and GPUs. The ROME CPU also give us advantage in these tests: each CPU has 4 NUMA nodes, allow non-overlapping binding; the FLOPS of 64 cores also can contribute to the final result.
- IO500: the NVMe SSDs and fast interconnect will be an advantage for this benchmark. Besides, GlusterFS file system with RDMA mode will give us a good result in this test.
- SST: The CPU has a high bandwidth capabilities as well as scalability, which play a crucial part for SST performance, especially at simulating a high scale systems, there is a large number of data shuffling occurs and calculation heavy scalable loads due to modular structure of the application.
- VPIC: The ROME CPU supports different types of vectorisation that is usable in this application, including SSE, AVX, AVX2. Although there is no AVX512, the processor supports native-256bits instructions, speeds up 2X performance comparing to the previous AMD CPU generations, which is also as good as the performance of Intel CPU with AVX512. Combining with fast and large memory, this cluster setup will be perfectly work out for this application.
- Normal Mode: The performance of this application is mainly determined by performance of matrix-vector multiplication which is limited by the memory bandwidth. The AMD Zen2 7742 supports up to two-way SMP and up to 4 TiB of eight channels DDR4-3200 memory per socket, performs up to 190.7 GiB/s in bandwidth. Besides, large CPU caches allow to load big blocks of matrix. Those above features speed up calculations, especially when we have a large problem/matrix.

- The secret application: Whereas the secret application have not been announced yet, we strongly believe that this application will involved GPU acceleration. There are some reasons for that. Firstly, three announced applications purely work with CPUs. Secondly, AI tasks have appeared in every SCC-s so far. Thus, having a good preparation for this case is a good strategy, in which we have Tesla V100 GPUs. Besides, many software has CUDA support, enable us to fork the application into GPU with less hurt.

2 Software

2.1 Admin tools

The operating system (OS) of our choice is CentOS 7.7 as we have already obtained a lot of experience with it and are aware of all potential pitfalls we might encounter at the competition. Although we are currently testing and comparing CentOS 8.0 on VM machine to 7.7 version to see if everything does work smoothly and reliably. The IB network takes the use of Mellanox OFED drivers. The network subnet manager is run on the IB switch releasing the extra workload from the nodes.

Having faced OS, drivers and CUDA failures that led to time shortages at the previous competitions, the automation was built up. It assures us from any issues with the core system due to reasons like transportation or power cuts. In case of such accidents the whole system can be brought up back in minutes from our own Github repository with all the necessary files. The whole process is run with Ansible, which is adjusted to be as versatile as possible, separating different roles,e.g. adding user groups, configuring IB.

Prometheus is established to be our main monitoring and alerting tool. The hardware and OS metrics is collected by node exporter providing a easily readable data on the actual CPU and memory utilisation. Currently we are working on exporting GPU metrics via nvidia-smi or NVML for better visualisation.

Our cluster also benefits from the parallel file system - GlusterFS + ZFS - to boost I/O bandwidth, avoid metadata bottlenecks and improve cluster's performance. Performance tests for lookups, open, close, random read, random rw, sequential read, sequential rw gave good results for a small cluster comparing to other file systems. Besides, GlusterFS is easy to setup and maintain.

We also are testing different solutions to make snapshots of the whole system in order to avoid any data loss due to the power cut at the competition. However, some application, e.g. VPIC, can create dump files and hence they do not require any system snapshots.

2.2 Compilers

- **C/C++/Fortran**:

  - AOCC v2.0: AOCC is a compiler system provided by AMD that has optimisation for performance improvement in AMD EPYC 7002 Series architecture. Besides, this compiler software also improved vectorisation and code generation towards the Rome CPUs which can give us advantages for application relates to the Linear Algebra. In fact, this compiler gave us an significant improvement in performance for the matrix vector multiplication in the Normal Mode application.

  - GCC-9.1.0: Besides AOCC, we also tested with the gcc-9. This is the mostly newest version of gcc with OpenMP support. This version also tunned for 'znver2' cpu type, gives a slightly enhancement in the performance of VPIC.

- **MPI**:

  - OpenMPI-4.0.2: OpenMPI is our first choice since it is optimal for many network routines and API. Additionally, it is easy to tweake with ompi parameters to get the best performance out of a specific application. Together with OpenMPI, HWLOC is used to facilitate controlling work over process topology; OpenSHMEM provides routines with low-latency, high-bandwidth communication thus improving performance of multinode-calculation. For the communication layer for MPI, we will alternatively use MXM, UCX, OFED verbs depending on each application.

  - OpenMPI-3.1.0: Beside the newest version of OpenMPI, version 3.1.0 will be used for HPL and HPCG benchmarks since the binaries provided by NVIDIA requires that version. GPUDirect RDMA coupled with UCX will be used as MPI communication layer for those benchmarks.

- MPICH-3.3.1: In fact, some MPI-3 features in OpenMPI are buggy. The OpenMPI did cause errors in MPI communicator initialization for the Normal Mode when it calls subroutines from the Parmetis library. Thus, we tested with other MPI packages, MPICH vs. MVAPICH. MPICH gave us the best results for the Normal Mode.

2.3 Libraries

- **SST**
  - Zoltan 3.83
  - HDF5 1.8
- **Normal Mode**
  - BLAS: We tested with several different derivatives of BLAS. BLIS+LibFLAME raised error in memory allocation. MKL2019 gave better results than OpenBLAS-0.3.6. Thus, MKL2019 will be used in the competition.
  - pEVSL and parmetis-4.0.3

2.4 Additional software

- SPACK - A package manager tools: This software helps with installing packages automatically and manage those software.
- Profiling tools: Besides tunning application's peformance with different compilers, compilers' FLAGs, and libraries, we also profiled applications to understand more about them. From the profiling graphs, we can detect bottlenecks of applications, then have suitable solutions to improve performance. HPC-ToolKit was used to provide general information about program's work, resource consumption, and inefficiency. Moreover, to analyze the effectiveness of MPI parallelism or Hybrid MPI + OpenMP, we used Paraver with Extrae library.
- Visualization: Paraview Server is used to visualize the results of VPIC and Normal Mode on the server as the output data is so large that visualization works could not be done on the local machine.