Team SwanGeese

University of Science and Technology of China

General Information

A Brief Introduction to USTC

The University of Science and Technology of China (USTC) is a prominent university in China and enjoys an excellent reputation worldwide. It was established by the Chinese Academy of Sciences (CAS) in 1958 in Beijing, and in 1970, USTC moved to its current location in Hefei, Anhui.

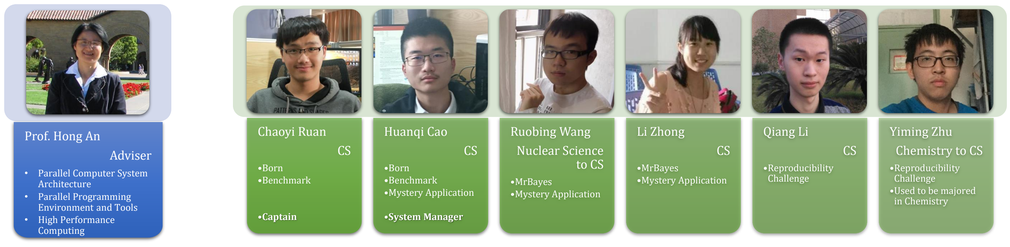
USTC actively participates in international cooperation in various ways. It has engaged in multiple joint research and educational activities with internationally acclaimed organizations.

Team Information

Team SwanGeese, naming "鸿雁" in China, founded in 2012, have participated in various HPC-related competitions 22 times, with 121 students involved and won many prizes. Among those, we have participated in the Student Cluster Competition in SC thrice, ISC thrice and ASC twice.

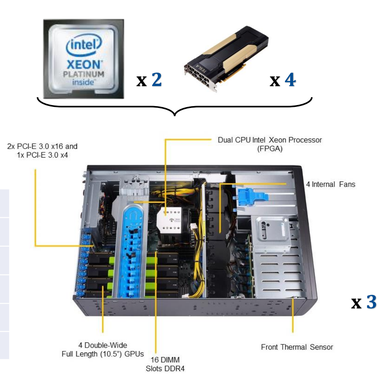
In SC16 10th anniversary commemoration for the SCC , we've won both LINPACK and Overall winner in the SCC. We have endlessly enhanced our technical strength through successful cooperation with our sponsors on designing, constructing and testing the competition systems and optimizing applications on it.

This year, we've formed a new team with five new members and one experienced, of which are from different regions in the large territory of China, including Yiming Zhu being one of Chinese minority, Hui. Li Zhong and Huanqi Cao are from the School of the Gifted Young, who are 1~3 years younger than students of the same grade: Li Zhong born in 1998 and Huanqi Cao born in 1999. We prove our attempt on diversity by our team founding. Also, two of us used to be majoring in Chemistry and Nuclear Science and learnt knowledges of other fields, providing meaningful help during our application understanding. Currently, they are studying in Computer Science.



System Configuration

Last year, we made our double championship with a GPU-oriented cluster brought by Supermicro and NVIDIA. This time, we again come with such a system.



System	**Supermicro SuperServer 7049GP-TRT** x 3 nodes
CPU	**Intel Xeon Scalable Processor 8176** @2.1GHz, 28 Cores x 2 CPUs/node
GPU	**NVIDIA Tesla V100 for PCIe** x 16slots, 4 cards/node, at least 8, prefer 12 in total
Memory	DDR4 RDIMMs@2666MHz, 256GB/node
Interconnect	**Mellanox EDR 100Gb/s** Switch
IO	1.2TB MLC SSD on each node

We build our system with 8 to 12 V100s, the latest NVIDIA Tesla GPU, 4 per node in 3 nodes. As the successor model of Tesla P100 which we used 10 of it last year, the Tesla V100 has much higher computing ability and almost the same power consumption compared to P100: 900 GB/s HBM2 compared to P100's 732 GB/s, 80 SMs compared to P100's 54. As all known applications can benefit from GPU, we choose NVIDIA Tesla cards with no hesitate; our tests shows tens times of acceleration against CPU under same power consumption.

While total TDP goes far over 3000 Watts limit, we estimated the actual power consumption in different applications; as a result, we found that in memory-intensive applications on GPU, like HPCG and our transplanted Born, shows relatively low power consumption about 150W per card, which won't break the power limit; as to compute-intensive applications like HPL, they show better power efficiency with graphics clock limited to 900 - 1000 MHz than highest clock.

While we choose GPUs as our main computing hardware, we also prepare those 6 Intel Scalable Processor Platinum 8176, for the possibly CPU-featured mystery application, as they provides high CPU performance within the only 3 nodes. We'll slow them down a little when our GPUs are running at full speed to cover the power limit.

Operating System	GNU/Linux CentOS 7.4, x86_64
Compiler	GNU C/C++/Fortran Compiler 4.8.5 & Intel C/C++/Fortran Compiler 18.0 (for different applications)
Resource Managers	htop 2.0.2 & NVIDIA-SMI 384.81
MPI	OpenMPI 1.10.2 & Intel MPI Version 2018 (for different applications)
Power Monitor	Power Monitoring Toolkit (Self developed via IPMI & PDU querying)
Parallel File System	OrangeFS v2.9
GPU Runtime	CUDA Toolkit 9.0
Profilers	Intel VTune Amplifier 2018; NVIDIA Visual Profiler 9.0.176

As to software system, we have the above environments backing our whole competition. Used software include reliable open source projects, components in Intel Parallel Studio 18.0 and CUDA Toolkit 9.0.
To be mentioned, with the help of Intel VTune Amplifier and NVIDIA Visual Profiler, we profiled our applications, took a glance at applications' runtime features and got hands on analyzing and optimization quickly, which are really helpful.

Why we will win?

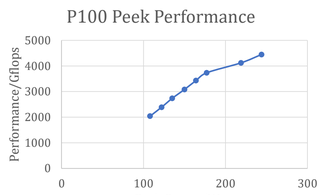
- **Strong Team Cooperation**
  Working in groups on each task, we've done our best in cooperation.
- **Working with Domain Experts**
  Prof. Zhaolei Zhang from University of Toronto helped on MrBayes application;
  Institute of Geology and Geophysics – CAS helped on Born application.
- **Working with Engineers from Supermicro and NVIDIA**
  Test system with Supermicro engineers in the US remotely against 16 hours time difference;
  Get help from NVIDIA engineers when testing benchmarks.
- **Love, Faith and Fight**
  For the competition and, also, the HPC.

Our Strategies

Benchmarks: HPL and HPCG

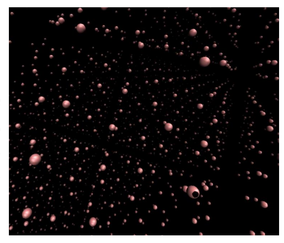
As the new Volta architecture Tesla V100s used heavily in our system, one of our sponsor, NVIDIA, has provided their latest optimized HPL and HPCG implementation with support for Volta with CUDA 9.0.

In HPL testing, we found that the final performance scales non-linearly while we tune the power limit. To gain best performance, we use 12 V100s, each working at relatively lower clock, to run Linpack within 3kW power limit. Our rough test of peak performance versus power consumption result on P100 on our testing platform shows as the graph in the right side.



In HPCG testing, we observed about 150 Watts power consumption per card, which won't touch the power limit even with 12 cards.

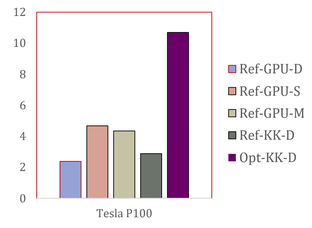
Reproducibility Challenge



LAMMPS is a classical molecular dynamics code, and an acronym for Large-scale Atomic/Molecular Massively Parallel Simulator. And in molecular dynamics simulation, compared with well-studied pair potentials, multibody potentials deliver increased simulation accuracy but are too complex for effective compiler optimization. There is a paper focusing on multi-body potentials in which the authors develop a vectorization scheme applicable to both CPUs and accelerators for LAMMPS.

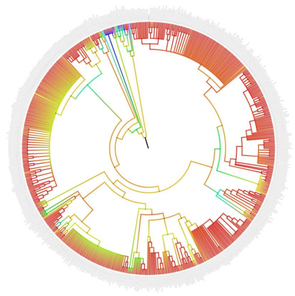
Our job this year is to reproduce the result described in the paper according to our cluster architecture:

- We read the paper and documents of LAMMPS, and understand how the algorithms function.
- Then we solved several compiling problem, because Makefile of LAMMPS has not been prepared for our architecture yet.
- Finally we got similar accelerate ratios like the results in the paper.



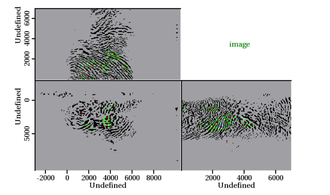
Application: MrBayes

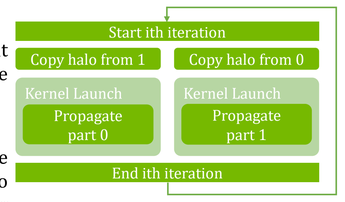
- **CPU-GPU Heterogeneous System Exploitation**
  The most time-consuming part, by our profiling, is the calculation of likelihood. After reading many publications about it and testing their actual performance on our cluster, we found the GPU accelerator of Beagle, which exploits our CPU-GPU system, is the most efficient option for our cluster.
- **Dynamic Parameter Strategy**
  Based on solid understanding of the application, we choose the MCMC parameters as efficient as we can. Familiar with the functions of those parameters, we dynamically adjust them in accordance to the running state instead of maintaining the same parameter.
- **Architecture-Based Optimization**
  Taking the traits of high performance clusters into consideration, we make use of optimizing options of INTEL compiler. Moreover, we also place the strong-related chains in the same CPU to decrease time of communication.
- **Interdisciplinary Communication**
  Consulting and communicating with Prof. Zhong Wang from JGI and molecular geneticist Prof. Zhaolei Zhang from University of Toronto, we make sure we understand the underlying knowledge and what the application is really doing when running.

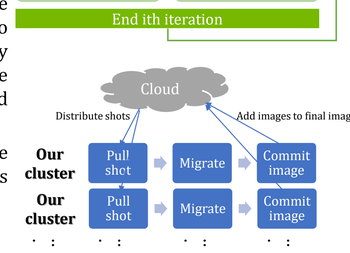


Application: Born

- **Fully GPU Transplant via CUDA Programming**
  With runtime profiling using Intel VTune Amplifier, we pointed it out that the propagation takes the most of the time; with statically analyzing the code, we stated that this kernel computing best to be executed on GPU. To avoid the slow data exchanging over PCIe between CPU and GPU, we transplanted all the kernels to GPU with the help of CUDA, in our own hands.
- **Scalable Multi-GPU Computing**
  While finding the dataset too large to fit in a single GPU's memory, we decided to split the data space into multiple part, and use CUDA Peer to Peer Memcpy to exchange data between GPUs before each iteration effectively.
- **Asynchronous Streaming Execution and Cloud Hosted Result Server**
  To deal with large amount of data (1000 to 10000 shots, 2.5 GiB per shot), we designed an asynchronous stream to download a single shot data, process it into migrated image and integrate the image to final result. Also, to avoid the unreliability of our own cluster (the power shut-off activity), we choose the cloud system to serve the final result: our cluster commit migrated images to the cloud, and the cloud add this shot's image to the final image.







In this application, we will make use of most GPUs; thus the power limit is to be considered. Observing that the power consumption of a single V100 stays below 150W, It's Okay for this application to fully use the cluster.

Time Arrangement

Our arrangement on running those applications are as following:

- As Born and MrBayes both needs relatively long execution and MrBayes is shorter compared to Born, they will share the whole cluster and running on separate nodes.
- Reproducibility challenge will finish their execution as soon as possible once the competition starts.
- Mystery application compiling and execution will take on once MrBayes gets an acceptable result and we team members finished basic analysis on it.

Sponsors