Swarming the SC'17 Student Cluster Competition

Petros Eskinder
Georgia Institute of Technology

Dr. Oded Green
8/5/2018 12:55:10 PM PDT

Dr. Jeffrey Young
8/7/2018 10:04:14 AM PDT

A thesis submitted to the Georgia Institute of Technology
for the degree of Bachelor's of Science

Swarming the SC'17 Student Cluster Competition

The Experience of Team Swarm from the Georgia Institute of Technology

Petros Eskinder
peskinder3@gatech.edu

***Abstract*—The Student Cluster Competition is a suite of challenges where teams of undergraduates design a computer cluster and then compete against each other through various benchmark applications. The present study will provide a select summary of the experiences of Team Swarm who represented the Georgia Institute of Technology at the SC'17 Student Cluster Competition. This report will first describe the competition and the members of Team Swarm. After this introduction, it focuses on three major aspects of the experience: the hardware and software architecture of the team's computer cluster, the team's system administration workflow and the team's usage of cloud resources. Additionally, the appendix provides a brief description of the team members and their method of preparation.**

I. The Student Cluster Competition

This section introduces the Student Cluster Competition.

A. Motivation for the Competition

In 2007, the Association for Computing Machinery (ACM) introduced the Student Cluster Competition as an avenue to expose undergraduates to High Performance Computing (HPC). [1].

In the years prior, several parties had identified a deficiency in undergraduate HPC education [2], despite HPC becoming an increasingly critical component of scientific and industrial research.

The Student Cluster Competition focuses on *experiential learning*. By using a competition style environment, it aims to expose students to the many issues and constraints found in production level HPC system, as well as the interdisciplinary nature of professional HPC environments [2], [3].

Prior initiatives like the Curriculum Initiative on Parallel and Distributed Computing focused on expanding the existing Computer Science and Computer Engineering degree programs to include more parallel programming and HPC architecture courses [2].

Because of its success, the initial competition has dramatically grown in popularity, spawning other Student Cluster Competitions at the International Supercomputing Conference (ISC) and the Asia Supercomputing Community (ASC). Because there are subtle differences between these competitions, subsequent sections will focus on the ACM/IEEE Supercomputing Conference.

B. Structure of Competition

The competition is one week long and is held at the annual ACM Supercomputing conference. During the week of the competition, teams of undergraduates run six different high performance applications aiming to achieve optimal performance on each applications.

In the months prior to the competition, students must do considerable preparation and legwork.

First, before they can compete, student teams must apply for acceptance, through a proposal. The proposal tabulates the members on their team, how they are preparing for the competition, which hardware vendors they have relationships with, and how they have architect their supercomputer.

If accepted, teams commence preparation by:

- obtaining their cluster from their vendor
- physically assembling their cluster
- setting up their operating system and other critical system software
- developing strategies for administering their cluster
- experimenting with different toolchains
- tuning each application to be as performant as possible
- preparing recovery mechanisms for system going down

C. Judging Metrics

Teams are judged against two orthogonal metrics, creating the potential for two "winners."

First, they are judged against their performance on the LINPACK and HPCG benchmarks. The team with the highest combined benchmark results wins this sub-competition.

Second, teams are judged against their aggregate performance across the six applications. This sub-competition also included the team's performance in interviews with the competition committee as well as the quality of the team's poster. Each application and interview was scored, and the team with the highest final score became the Overall Winner.

D. SC'17 Competition

The competition discussed in the present study was held during the SC'17 conference in Denver, Colorado. Sixteen different teams from various universities (and in some cases, high schools) across the world were selected to compete.

The competition, itself, lasted six days days: November 11 to November 16, 2017. For the first two days, teams set up and built their clusters. On the third day, November 13, teams ran the LINPACK and HPCG benchmarks. The remaining three days were dedicated to the four other applications. On the last day, the winner was awarded and teams cleaned up and packed.

II. Applications at SC'17

This section briefly describes the six different applications used in the SC'17 Student Cluster Competition.

A. Background

During the SC'17 Student Cluster Competition, teams were expected to run six different applications. Of these six applications, one was a "mystery" application that was revealed during the competition.In addition to optimizing their applications, teams were also expected to prepare for and recover from a power shut off activity, where every team's cluster is unexpectedly shut off. This was intended to simulate the uncommon occurrence of needing to recover a shut down system. Finally, teams were also expected to incorporate a cloud component to their general workflows.

B. The High Performance LINPACK Benchmark

The HPL benchmark is a parallel implementation of the LINPACK, a library of numerical linear algebra tools. It is currently used to rank the world's supercomputers on the Top500 list [4]. The benchmark measures the floating point rate for solving a dense linear system [4]. The fastest computers have the highest floating point rate.

During the competition, teams competed to attain the highest resulting floating point rate, while keeping their cluster's peak power consumption under 3000 Watts throughout the entire calculation. If at any point during an HPL run, teams exceeded the 3000 Watt limit, they were forced to restart [5].

C. The High Performance Conjugate Gradient Benchmark

The High Performance Conjugate Gradient (HPCG) benchmark serves as an alternate benchmark to the HPL benchmark. The benchmark synthesizes a sparse 3D linear system, and then over the span of 30 minutes, it executes a fixed number of iterations of the Gauss-Seidel conjugate gradient descent [6].

Compared to HPL, the HPCG has a significantly lower compute to data access ratio because it performs matrix vector multiplication on a *sparse* matrix rather than on a *dense* one like HPL.

Consequently, HPCG better assesses a system's ability to handle IO bound applications. Note this was by design. The HPCG benchmark was motivated by the growing belief that HPL did not accurately assess modern HPC systems [6]. Despite modern HPC applications are increasingly governed by high bandwidth and low latency [6], HPL does not assess these attributes. Instead it focuses on compute bound applications [6].

Similar to HPL, teams competed to obtain the highest floating point rate all while staying under the power limit [5].

D. Reproducibility Challenge

In recent years, there has been growing concern that large swathes of research results in computer science are not independently reproducible [7], [8]. The ACM, in response, has introduced an artifact review process that encourages paper authors to distribute a well-formed and documented digital artifact [7]. This would allow reviewers and other third party groups to reproduce and analyze the authors results.

The SC'17 competition committee selected *The Vectorization of the Tersoff Multi-Body Potential: An Exercise in Performance Portability* as the paper to reproduce [5], [9]. The paper introduced a vectorized implementation of the Tersoff Multibody Potential into the LAMMPS molecular dynamics library, documenting a measurable speedup over the existing LAMMPS implementation for multiple different CPU and accelerator combinations [9].

Teams were responsible for reproducing the results of the paper for at least one CPU architecture, and then submitting a report detailing their experiences [5].

E. MrBayes

MrBayes is a tool for Bayesian Phylogenetic analysis that is often used for species delimitation, the process of identifying the species of various organisms from their phylogenetic data [10].

Before the competition, teams were expected to identify and tune an efficient installation of MrBayes for their cluster. Additionally, they could integrate it with the Beagle library, a high performance library for performing Bayesian phylogentic calculations that can make use of ones GPUs [5], [11].

During the competition, teams were provided a dataset, for which they were expected to construct an accurate consensus tree [5].

F. Born Seismic Imaging

Born is a tool used by geologists to capture images of the earth's subsurface. It constructs seismic images by transmitting thousands of sound waves into the earth and recording the response signals [12]. In general, each of these thousands of sound waves are varied based on the position of the source and receiver, and the results are later stitched together.

During the competition, teams were provided 1,136 Born shots to process. They were responsible for processing as much of these shots as they could. Each shot could be run in parallel, though they could each potentially take up two to three hours to complete [5].

G. Mystery Application: MPAS

Unlike the other five applications, teams were unaware of this application until the second day of the application.

The mystery application at SC'17 involved using the Model for Prediction Across Scales (MPAS) Framework to model the atmospheric consequences of excess carbon in the Antarctic [13]. MPAS, itself, is an application framework for building weather simulations [13]. The Los Alamos National Laboratory and the National Center of Atmospheric Research use the framework to produce simulation components that are used in climate and weather studies [13].

H. Power Shutoff Activity

In addition to optimizing their applications, teams were also expected to prepare for and recover from a power shut off activity. During the activity, every team's cluster is unexpectedly shut off, and no team is allowed near their cluster until designated to do so [5]. This was intended to simulate the uncommon occurrence of needing to recover a shut down system [5].

I. Supplemental: Cloud Component

This section describes the competition's new cloud component. Although it was not required, teams were strongly encouraged to use the cloud. Additionally, they received points to the overall score based on how well they used their cloud resources.

Recently, within the HPC community, there has been increased interest in developing new hybrid HPC compute architectures and workflows that combine traditional owner-centric physical clusters with cloud and grid computing [14]. In an attempt to investigate these hybrid HPC and cloud workflows, a cloud component was introduced to the SC'17 competition [5].

Each team was given access to a cloud High Performance computing environment through CycleCloud on Microsoft Azure [5]. Teams could run the various competition applications in the cloud in addition to their physical compute cluster. Unlike the physical clusters, teams were allocated a specific *dollar cost budget* for their cloud instances [5].

III. Team Swarm

This section briefly introduces Team Swarm, Georgia Tech's inaugural team at the Student Cluster Competition.

A. Student Members

Team Swarm consisted of seven undergraduate members, six competitors and one alternate from the Georgia Institute of Technology. The competitors were Alok Tripathy, David Meyer, Dezhi "Andy" Fang, Jessica Rosenfield, Nicholas Fahrenkrog, and Petros Eskinder. The alternate was Manas George.

All members were Computer Science or Electrical Engineering upperclassmen, with prior exposure to HPC through research projects, internships and coursework.

B. Faculty Mentors

The team was mentored by Dr. Oded Green, Chirag Jain, and Will Powell, who are all affiliated with the School of Computational Science and Engineering at Georgia Tech. Dr. Green was a research scientist; Chirag a doctoral student; Will a research technologist.

C. Contributions By Writer

The writer of this report made four key contributions to Team Swarm: spearheading the initial team proposal as well as subsequent write ups, tweaking and experimenting with the system administration workflow, designing the team poster, and working on the reproducibility challenge.

Further information about our team, our preparation strategy, or our sponsorships is provided in the appendix.

IV. Hardware Design

Each team with the help of their industry sponsors was expected to design a compute cluster and justify why the cluster was suitable for the competition. Suitability is defined by performance on the HPL benchmark, HPCG benchmark, and the other four competition applications.

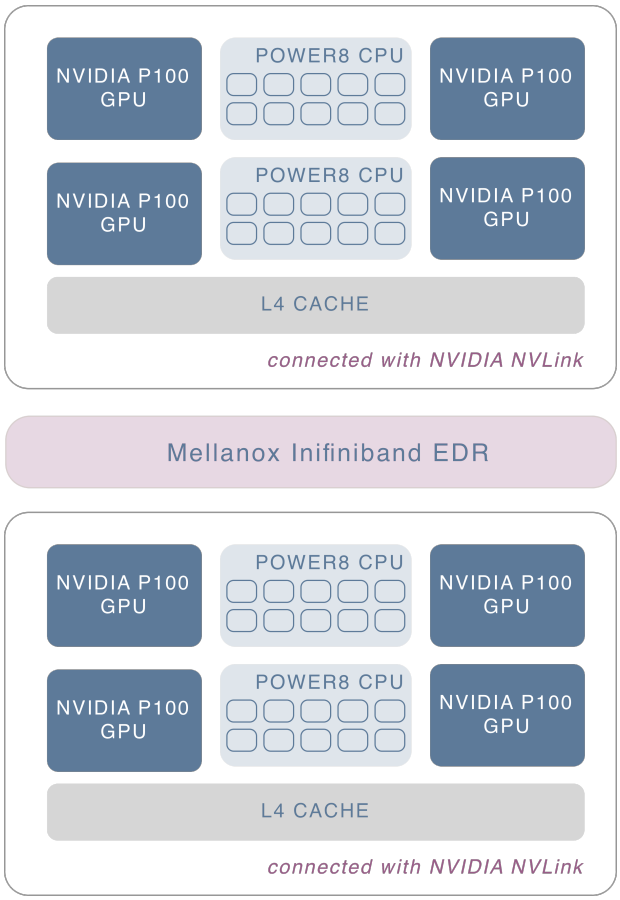


Fig. 1. Our Hardware Design

Below, we describe our hardware configuration, justify why we thought our system was well suited for the competition. We leave comments on its results to the next section.

A. Overview

Our team architected a cluster heavily based on the two node IBM Minsky system (Power S822LC). Each node was outfitted with two 10-core Power8 CPUs and four NVIDIA P100 GPUs. Between the two nodes, we used Infiniband EDR's as our interconnect. Additionally, we used NVIDIA's NVLink interconnect within each node for GPU-GPU and GPU-CPU communication.

B. Strength of Hardware: Why POWER8?

The POWER8 architecture is the first processor to come out of the OpenPOWER consortium, and has been designed for computationally intensive applications. We see this in part with the NVLink protocol, showing how the POWER8 system is optimized to work well with NVIDIA GPUs. A similar synergy is seen in the choice of interconnect which supports some of the fastest interconnects, like Mellanox InfiniBands.

Here is a listing of the advantages the Power8 possesses:

- **Fast Memory Subsystem**: The Power8's biggest attractor's is its very fast memory subsystem. The POWER8 memory

subsystem series shows consistently high memory bandwidth (∼90GB/s) on the Stream bandwidth benchmark, staying in the high nineties even as the number of threads varies from 20 to 80 [15]. These numbers were obtained using the GCC compiler, which is free and much easier for most developers to obtain and integrate into their toolchains, as opposed to vendor-specific compilers, and is therefore more representative of real world performance. The POWER8 processor also has big, high-bandwidth caches at each level of abstraction, further strengthening the memory subsystem.

- **SMT**: The per-core 8-way simultaneous multi-threading provides an incredible boost to memory-heavy applications, allowing the processor to hide memory latency over many threads. This means improved performance in applications like graph traversal. Thread scaling is further helped by support for hardware transactional memory, which removes the overhead of locking mechanisms in software.
- **Embedded NVLink**: Currently, POWER is the only architecture that supports the NVLink protocol, providing fast data transfer between GPUs. NVLink supports transfer rates 5 and 12 times faster than the more common PCIe3 protocol. Combined, the four NVLink ports on the Minsky system provide a 80GB/s full duplex link between the GPUs, minimizing the bottleneck presented by CPU to GPU transfers.
- **CAPI**: The Coherent Accelerator Processor Interface allows for tighter integration between the POWER8 processors and the P100 accelerators, allowing for better heterogeneous performance [16, 17]. It accomplishes this through two means: (1) providing a unified virtual memory space; meaning that the accelerators and the CPUs can use the same memory addresses, reducing device driver overhead, (2) permitting accelerators to behave like normal threads, reducing overhead with respect to cache coherency, as that is now taken care of in hardware [16].

C. The GPU Advantage

When designing our supercomputing cluster, we made several key decisions regarding (1) the composition of each node and (2) the number of compute nodes. We chose to build a cluster consisting of a small number of nodes based on a combined CPU/GPU platform as opposed to a larger cluster of smaller purely CPU based nodes.

Our preliminary literature survey suggested that using GPUs as computation accelerators provided significant performance improvements at a lower cost and power than adding another purely CPU-based node. For instance, Matsuoka et al. [18] showed that GPU's accounted for 66% of their LINPACK performance while only accounted for 15% of their total power consumption.

Consequently, we elected to build a cluster consisting of a small number of nodes, each based on a combined CPU/GPU platform.

Our experimental results corroborate our literature survey. Using only two of our eight P100 GPU's, we were able to achieve dramatic speedups using GPU optimized HPL and HPCG binaries. For example with HPL, each GPU, at its peak usage, provides 4-5 TFLOP/s.

Additionally, we found that these GPU based speedups have *not* come at a significant hit on power efficiency. From our experiments with HPL, we found that each GPU consumes 240-270 Watts of power when active, and 30 Watts when idle. In contrast, we found that one 10-core Power8 CPU consumes roughly 250-280 Watts, that is while providing a *fraction* of the computational benefit.

Given the P100's massive 10.6 TeraFLOPS peak single-precision performance, fed by 720GB/s memory bandwidth and backed by 16GB of high-bandwidth memory, using NVLink allowed us to minimize the bottleneck presented by CPU to GPU data transfers. Forgoing the incredible transfer speeds NVLink affords us would mean sacrificing performance to shuttling data between the host and the device, which is something we want to avoid at all costs.

V. Software Selection

In addition to designing their cluster's hardware, teams were also responsible for outfitting their machine with appropriate and highly performant system software, e.g. operating system, file system, toolchains, and scheduler [19].

In this section, we describe our system software and why we designed it as we did.

A. Operating System

For our operating system (OS), we elected to use CentOS 7.

When choosing an OS, we had two major constraints. First, our OS needed to be some Linux variant because Linux is the lingua franca of server side operating systems. Second, the OS needed to be supported by the Power8 architecture. At the time, Power8 supported two Linux variants: Red Hat Enterprise Linux (RHEL) 7.1+ and Ubuntu 16.04+ [20]. Though we were more experienced with Debian-based systems, like Ubuntu, several key system libraries, such as recent versions of ESSL, only offered support for RHEL-based systems like CentOS 7.

B. File system

For our file system, we used the Network File System or NFS, a distributed file system allowing us to share data between our two nodes, ramblin and wreck [21]. We were initially inclined to using IBM Generation Parallel File System (GPFS) as our file system. We posited that GPFS would be a great selection because it provided the performance characteristics matched by few file systems [22].

Additionally, our sponsor, IBM, as well as numerous Georgia Tech researchers had ample experience with tuning the performance of GPFS. Unfortunately, we found that GPFS was primarily designed for high-performance clusters with numerous nodes as opposed to our 2-node cluster [23]. Additionally, we found we did not have sufficient time before the competition to configure it.

C. Toolchain

In computing, a toolchain describes one's collection of system libraries and compilers. The section below describes our system's toolchains.

In general, we elected to use vendor specific tools, as they are clearly fine-tuned for our machines. Since our machine consists of

IBM Power8 CPUs with NVIDIA P100 GPUs, this translated to IBM's proprietary toolkit and NVIDIA's CUDA toolkit.

- **Compilers**: We used the standard GNU compilers as well as IBM's XL C/C++ compiler and CUDA C/C++.
- **BLAS**: We used the Engineering and Scientific Subroutine Library (ESSL), IBM's BLAS library, for math applications like LINPACK. We were motivated to use a vendor-specific BLAS after comparing the performance differences between them and the standard HPL implementation from the University of Tennessee, Knoxville on appropriate systems.
- **Message Passing Interface**: We used both OpenMPI and MVAPICH. Both MPI implementations are CUDA-aware, and work with InfiniBand. Initially, we had expected to use IBM Spectrum MPI. However, we found that support for Power8 systems was dropped.
- **System Monitoring**: To monitor our system for good load balancing, we used *Datadog*, a monitoring service. To monitor our GPUs' performance, we used the NVIDIA visual profiler as well as `nvidia-smi`.
- **Profilers and System Monitors**: We used a variety of miscellaneous tools to fine tune the performance of our system. For instance, to diagnose performance bottlenecks (e.g. communication latency, thread creation) in MPI and OpenMP applications, we used *Allinea*, an OpenMP and MPI profiler. For other tasks less suited for Allinea, for example, CPU intensive tasks executed on a single machine, we used Linux performance tools such as `perf` and flame graphs [24].

D. Scheduler

We used Slurm for our scheduler. However, we did not invest much energy to ensure it was performant or to automate many of our jobs. This was due to three reasons:

- **Limited Number of Nodes**: Our cluster only contained two nodes. A scheduler is considerably more useful when one has numerous nodes to manage.
- **Limited Number of Users**: We knew going in that we would only have six potential users and that during the competition, we would be in close proximity from one another. Consequently, if there were competing applications running, we could prioritize them through conversation and planning.
- **Time Constraints**: We gained access to our second node just a few weeks before the competition. a time when we were occupied with other aspects of the competition.

In the end, we took the approach of manually scheduling our applications, using system monitors to ensure that no two competing applications ran concurrently.

VI. System Administration Workflow

Teams were responsible for managing their system administration and application workflow. In this area, we invested considerable attention to automating our system environment. In particular, we make use of various tools so that we can deploy our system environment and be fully functional within only a few commands.

This section describes the key components of our system.

A. Chef

Chef is a configuration management tool used for dealing with machine setup on physical servers, virtual machines and in the cloud. It is primarily used as a provisioning tool, i.e. managing the software available on a cluster.

With our chef configuration, we are able to quickly set-up a machine and load it with our desired environmental state, i.e. all relevant libraries and tools installed and loaded.

As far as our workflow, we host our chef configuration in a git repository. Team members can make changes to the configuration of the cluster via submitting Pull Requests. Configurations are then converged into each of the hosts. This process is *idempotent* and *revertible*. Consequently, we are able to design reproducible experimentations of software configurations.

We are also able to make use of chef on the cloud. For example, through chef, we are able to share a set of common tools on both our Power 8 cluster and our cloud clusters.

B. Environment Module

A significant portion of our competition preparation involved determining the most optimal set of toolchains for each application. This meant experimenting on different applications with different system configurations.

For example, some applications can only be compiled in certain versions of GCC, whereas other applications can benefit from Power 8 specific optimizations present in IBM XLC. Unfortunately, because of conflicting libraries and environment variabls, manually setting up all the environment variables to switch compilers is rather cumborsome.

To circumvent this tedium of managing conflicting libraries and toolchain, we make use of environment modules. Environment modules allow us to dynamically load and unload different versions of toolchains, while ensuring no two conflicting modules are loaded.

Additionally, whenever possible, we packaged custom built system libraries and toolchain components into environment modules.

Finally, we store our entire set of module configurations in a git repository. This helps improve the reproducibility of our entire environment.

C. RPM Package and Artifact Store

As a part of the competition, we acquired and built numerous system libraries and components; for example, the Intel C compiler (ICC) and the IBM Engineering and Scientific Subroutine Library (ESSL).

To version control and to avoid recompiling system components when deploying to a new machine, we perform three major actions:

1) we package each component into its own RPM package
2) we version control each RPM package by storing its configuration into our Chef git repository
3) we store dependency and "file ownership" information for each RPM module

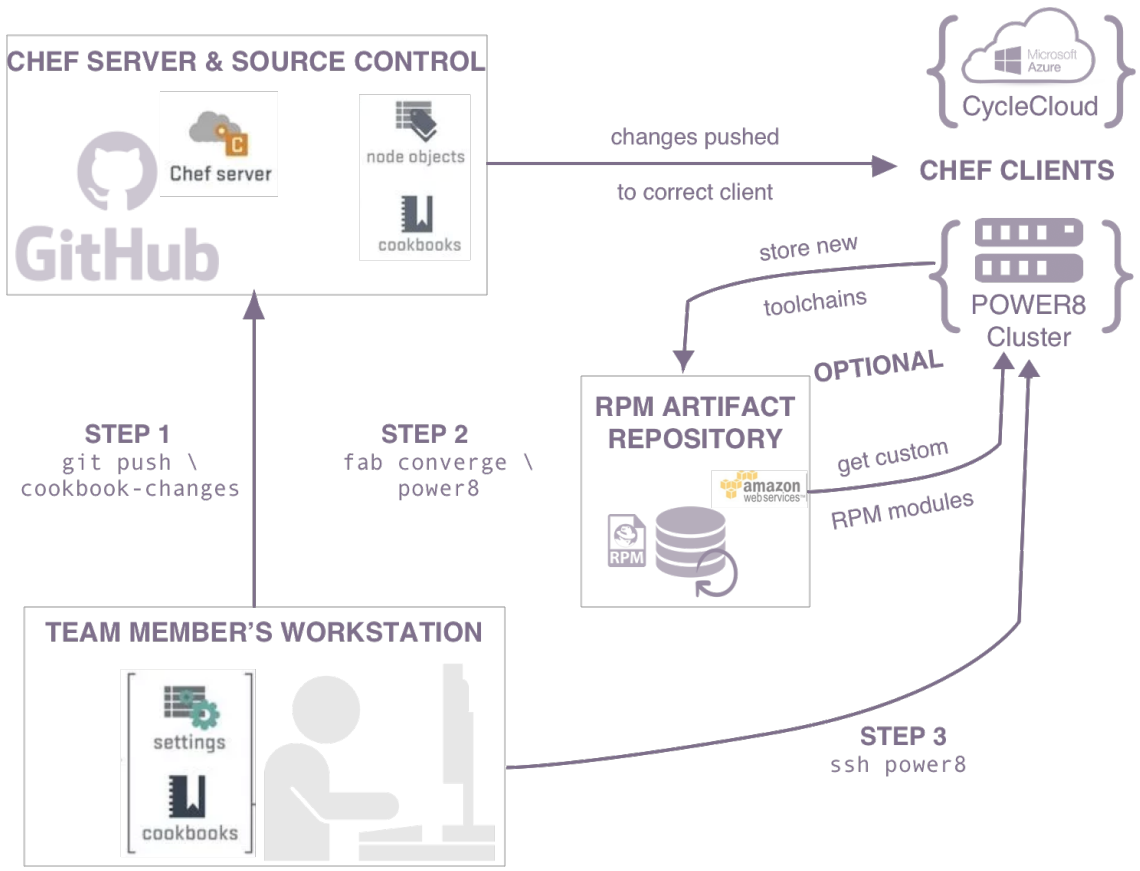


Fig. 2. Our System Administration Workflow

4) we publish the compiled artifact or binary into our internal RPM repository that we host in Amazon Simple Storage Service (S3)

Then when redeploying, we simply fetch from our internal RPM repository.

D. Nebula

Nebula is a command line tool we built to easily manage all of the machines in our cluster as well as in the cloud.

It automates updating process of each hosts, and ensures each host is up to date with the module files and general chef configurations. Additionally, it allows us to switch between different chef roles and branches, making for much easier experimentation.

To accomplish this, underneath the hood, we developed a service discovery system that indexes each of our machines, storing informations such as IP address and architecture. Additionally, we also defined a collection of tasks to help manage the state of our machines; convergence to a specific version, updating module configuration, etc.

Combining these parts, we can easily manage all of the machines in a cluster via a simple command-line tool. For example

```
fab converge -R power8
```

converges all Power8 machines to the latest configuration.

```
fab set_chef_branch:test-intel-cuda  converge -R
intel
```

converges all Intel machines to the branch "`test-intel-cuda`" for testing configuration changes made in that branch.

VII. Cloud Strategy

Below we document our strategy for using the cloud. Overall, we used the cloud as a backup for our physical servers. We relied on the cloud, whenever we were unable to overcome compatibility issues between an application on our local servers toolchains. Since we viewed it primarily as a backup, we did not invest considerable attention to optimally utilizing the allocated budget, nor did we actually finish our budget during the competition.

A. Background

In general, cloud computing can be thought of as on-demand virtual servers and infrastructure services available over the internet [25]. Instead of setting up and maintaining a physical compute cluster, one can request computational resources from a cloud provider like Amazon Web Services, Google Cloud Provider or Microsoft Azure [25]. Then, the cloud providers generally charge based off of the amount actually used.

Both in the research community and in industry, many organizations are moving their high-performance and high-throughput

workflows to the cloud. The cloud serves as an attractive alternative to physical clusters and data centers; eliminating the cost of setting up and maintaining potentially idle physical clusters, replacing it with dynamic and scalable compute configurations [26].

B. Cloud Orchestration

Cloud orchestration refers to the different processes involved with deploying services on a cloud environment [27], [28]. For example, what resources are allocated, what workloads are shared between different resources, and what services are deployed on what cloud environments [27], [28].

One of the early decisions we made in preparation of the competition was to use Chef to manage the configuration of our cluster. Naturally, Chef can also be extended for use on the cloud, so we were able to reuse our different chef recipes to orchestrate our cloud environment. There was one minor complication. To initially configure the cloud clusters, CycleCloud's cloud instances their own Chef configuration. This could have meant conflicts had our configurations differed in a meaningful way. However, we were able to have our Chef setup run side-by-side without any compatibility issues. By using our own Chef recipes, we were able to version-control our server configurations which increased the reproducibility of our entire software stack.

C. Architectural Considerations

Since the start of the century, Intel has stronghold on the HPC community. As of 2016, more than 90 percent of HPC systems are equipped with Intel processors, including those using NVIDA GPU accelerators [29]. Consequently, HPC applications are often designed assuming they'll be run on x86_64 processors, the CPU architecture often used by Intel processors. Our cluster had a pp64le CPU architecture, a CPU architecture unique to IBM POWER machines [30]. Infrequently, this difference resulted in compatibility issues with an application's library. For example, the library or application would expect an Intel specific library on our path, or it would use Intel's vector ISA rather than POWER's. We viewed the cloud as a backup if ever we were unable to overcome a compatibility issue on our local server.

D. Application Specific Strategies

We primarily used the cloud for the Mystery Application and the Born Seismic Imaging Application. The following section explains our approach for each application utilizing the cloud.

1) MPAS Mystery Application: Entering the competition, we reserved the cloud first and foremost to the mystery application. From our collective experiences working with the other applications, we were apprehensive that there would be compatibility issues between the Power 8 servers and the libraries used by the mystery application. As expected, there were numerous compatibility issues between our physical servers' toolchains and the libraries needed by MPAS. On the cloud, however, there were no such issues. We dedicated a H16r cluster and were able to successfully compile MPAS on the cloud.

2) Born Application: We offloaded Born to the cloud so it wouldn't compete for our cluster's limited compute resources.

For the Born application, teams tried to complete as many "shots" from the provided 1,136 shots [5]. Without considerable modification, the Born application only ran on the CPU, and each shot took two to three hours to complete.

We did not want the Born application to prevent other applications from using of our server's resources. We also were constrained on time before the competition, so we did not write the requisite CUDA code to exploit our GPUS.

Consequently, the cloud seemed like a naturally a fit for Born workloads. As far as our cloud configuration, we chose one of the most high-CPU instances (D64s) for born. Additionally, we utilized vectorization as well as Link Time Optimizations available on the Intel Platform. Since Born only ran on a single-node, we didnt need the RDMA related benefits from the H-series instance types.

VIII. Acknowledgments

We thank our advisors for challenging us to write this proposal, as well as for mentoring us throughout this process.

References

[1] S. L. Harrell, P. M. Smith, D. Smith, T. Hoefler, A. A. Labutina, and T. Overmyer, "Methods of creating student cluster competition teams," in *Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery*, ACM, 2011, p. 50.

[2] M. Sahami, M. Guzdial, A. McGettrick, and S. Roach, "Setting the stage for computing curricula 2013: Computer science–report from the acm/ieee-cs joint task force," in *Proceedings of the 42nd ACM technical symposium on Computer science education*, ACM, 2011, pp. 161–162.

[3] S. L. Harrell, H. A. Nam, V. G. V. Larrea, K. Keville, and D. Kamalic, "Student cluster competition: A multi-disciplinary undergraduate hpc educational tool," in *Proceedings of the Workshop on Education for High-Performance Computing*, ACM, 2015, p. 4.

[4] J. J. Dongarra, P. Luszczek, and A. Petitet, "The linpack benchmark: Past, present and future," *Concurrency and Computation: Practice and experience*, vol. 15, no. 9, pp. 803–820, 2003.

[5] (2017). 2017 competition applications, [Online]. Available: http://www.studentclustercompetition.us/2017/applications.html.

[6] J. Dongarra, M. A. Heroux, and P. Luszczek, "High-performance conjugate-gradient benchmark: A new metric for ranking high-performance computing systems," *The International Journal of High Performance Computing Applications*, vol. 30, no. 1, pp. 3–10, 2016.

[7] (2018). Artifact review and badging, [Online]. Available: https://www.acm.org/publications/policies/artifact-review-badging.

[8] I. Jimenez, C. Maltzahn, A. Moody, K. Mohror, J. Lofstead, R. Arpaci-Dusseau, and A. Arpaci-Dusseau, "The role of container technology in reproducible computer systems research," in *Cloud Engineering (IC2E), 2015 IEEE International Conference on*, IEEE, 2015, pp. 379–385.

[9] M. Höhnerbach, A. E. Ismail, and P. Bientinesi, "The vectorization of the tersoff multi-body potential: An exercise in performance portability," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE Press, 2016, p. 7.

[10] J. P. Huelsenbeck and F. Ronquist, "Mrbayes: Bayesian inference of phylogenetic trees," *Bioinformatics*, vol. 17, no. 8, pp. 754–755, 2001.

[11] beagle-dev, *Beagle-lib*, https://github.com/beagle-dev/beagle-lib, 2018.

[12] J. Hudson and J. Heritage, "The use of the born approximation in seismic scattering problems," *Geophysical Journal International*, vol. 66, no. 1, pp. 221–240, 1981.

[13] D. Heinzeller, M. Duda, and H. Kunstmann, "Towards convection-resolving, global atmospheric simulations with the model for prediction across scales (mpas) v3. 1: An extreme scaling experiment," *Geoscientific Model Development*, vol. 9, no. 1, p. 77, 2016.

[14] G. Mateescu, W. Gentzsch, and C. J. Ribbens, "Hybrid computing where hpc meets grid and cloud computing," *Future Generation Computer Systems*, vol. 27, no. 5, pp. 440–453, 2011.

[15] J. D. Gelas, "Assessing IBM's power8, part 1: A low level look at little endian," Tech. Rep.

[16] J. Stuecheli, "Power8," in *Hot Chips*, vol. 25, 2013, p. 2013.

[17] J. Stuecheli, B. Blaner, C. Johns, and M. Siegel, "Capi: A coherent accelerator processor interface," *IBM Journal of Research and Development*, vol. 59, no. 1, pp. 7–1, 2015.

[18] S. Matsuoka, T. Aoki, T. Endo, A. Nukada, T. Kato, and A. Hasegawa, "GPU accelerated computing—from hype to mainstream, the rebirth of vector computing," *Journal of Physics: Conference Series*, vol. 180, no. 1, p. 012 043,

[19] J. Pješivac-Grbović, T. Angskun, G. Bosilca, G. E. Fagg, E. Gabriel, and J. J. Dongarra, "Performance analysis of mpi collective operations," *Cluster Computing*, vol. 10, no. 2, pp. 127–143, 2007.

[20] (2015). Supported linux distributions for power8 linux on power systems.

[21] D. Arthursson, *Network file system*, US Patent 8,156,146, Apr. 2012.

[22] F. B. Schmuck and R. L. Haskin, "Gpfs: A shared-disk file system for large computing clusters.," in *FAST*, vol. 2, 2002.

[23] F. Schmuck and R. Haskin, "GPFS: A shared-disk file system for large computing clusters," *Proceedings of the Conferenced on File and Storage Technologies*, 2002.

[24] B. Gregg, "Linux profiling at netflix," 2015.

[25] E. Knorr and G. Gruman, "What cloud computing really means," *InfoWorld*, vol. 7, pp. 20–20, 2008.

[26] A. Gupta and D. Milojicic, "Evaluation of hpc applications on cloud," in *Open Cirrus Summit (OCS), 2011 Sixth*, IEEE, 2011, pp. 22–26.

[27] A. Parameswaran and A. Chaddha, "Cloud interoperability and standardization," *SETlabs briefings*, vol. 7, no. 7, pp. 19–26, 2009.

[28] D. Baur, D. Seybold, F. Griesinger, A. Tsitsipas, C. B. Hauser, and J. Domaschka, "Cloud orchestration features: Are tools fit for purpose?" In *Utility and Cloud Computing (UCC), 2015 IEEE/ACM 8th International Conference on*, IEEE, 2015, pp. 95–101.

[29] M. Feldman. (2016). Hpc in 2016: Hits and misses, [Online]. Available: https://www.top500.org/news/hpc-in-2016-hits-and-misses/.

[30] D. Henderson, *Power8 processor-based systems ras*, 2016.

[31] D. Fang and D. Chau, "M3: Scaling up machine learning via memory mapping.," *Proceedings of the 2016 International Conference on Management of Data.*, 2016.

[32] S. C. Competition. (2017). 2017 submission guidelines.

[33] W. M. DuBow, "Diversity in computing: Why it matters and how organizations can achieve it," *Computer*, vol. 46, no. 3, pp. 24–29, 2013.

[34] D. G. Johnson and K. W. Miller, "Is diversity in computing a moral matter?" *ACM SIGCSE Bulletin*, vol. 34, no. 2, pp. 9–10, 2002.

[35] A. W. Ervin. (2017). Office of institute diversity strategic plan.

[36] G. I. of Technology. (2015). Intel, georgia tech partner to diversify workforce, [Online]. Available: http://www.news.gatech.edu/2015/08/04/intel-georgia-tech-partner-diversify-workforce.

APPENDIX I. TEAM SWARM

We document who was on our team members and what our strengths as a team were. Additionally, we comment on our team diversity and our inclusive recruitment practices.

A. Strength of Team

Our mentors recruited our team with the intent of assembling an excellent group of passionate and capable students and faculty.

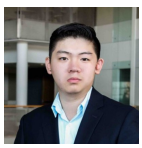
Since the SC'17 Student Cluster Competition was Georgia Tech's first student cluster competition, to ensure our team was competitive, we were selected on the following criteria:

- **Strong Academic Background.** Through our coursework, we each shared knowledge of computer architecture, operating systems, data structures and algorithms, and parallel programming. Members on our team have also taken graduate level courses in Numerical Linear Algebra, Operating Systems, Distributed Systems, and Parallel Computing Architecture.
- **Experienced with Linux.** All team members had prior experience using Linux through courses, internships, and personal usage. Some members had even worked as administrators. For instance, Andy administered the clusters for both his research lab and the company he previously led. Other members were active competitors of capture-the-flag (CTF) competitions, where they exploited vulnerabilities in Linux ELFs or binaries.
- **Extensive Research and Industry Experience.** We had a broad range of research and internship experiences relevant for the competition. Our shared research experience ranged from working on parallel streaming graph algorithms, to accelerating genomic applications using GPUs, to integrated circuit fabrication, to data analytics and parallel programming instrumentation. Additionally, we had also interned at a number of companies, providing us the opportunity to further improve our software engineering skills and face a wider breadth of programming and engineering problems. These companies included Apple, Bloomberg, Facebook, Google, Sandia National Labs, Square, Symantec, and Texas Instruments.
- **Leaders in Our Community.** Every one of us had loved some course or club enough that we wanted to take a leadership position to make a positive impact on it. For example, 5 of our 7 members had served as undergraduate teaching assistants for our computing courses. Jessica had served as the president of College of Computing's Undergraduate Council, and had been recognized as Outstanding Senior in Computing. Manas led our school's Computer Security club, where he organized weekly CTF competitions.
- **Passionate Programmers.** We all also programmed extensively outside of school, largely for fun. Several of our team members had competed in and won at hackathons. Others had competed in algorithmic programming competitions. Some members had also started and ran their own technology companies.

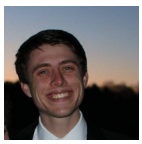
We believed that by coupling our shared experiences and talents with duly applied hard work, we would be competitive at the 2017 competition as well as many future ones to come.

B. Our Members

Petros Eskinder was a fourth year computer science major whose interests lie in graph algorithms and high performance computer architecture. As an undergrad at Georgia Tech, he served as a teaching assistant for courses on algorithms and computer systems. He previously interned at Google and Facebook, where he developed tools to automate build-file creation and made *pretty* mobile screens. Last year, he launched a viral mobile application that was featured on various media publications like Business Insider, CNN, and NBC News.



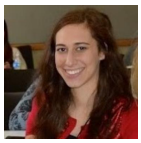
Dezhi "Andy" Fang was a second year computer science major. He was first exposed to HPC through a data analytics research project where he reimplemented the lower-level communication of a big data framework to use memory-mapped I/O. The results of his work were published in SIGMOD'16 [31]. Andy has interned with Symantec where he created a malware visualization pipeline using Hive and Hadoop. He has also served as technical lead for a Chinese startup. He will be responsible for managing our usage of Docker.



Nick Fahrenkrog was a fourth year student and the only electrical engineer on the team. He possesses a wide breadth of experiences, ranging from conducting integrated circuit fabrication research, squashing security issues at Texas Instruments, and developing full stack web applications. He is an avid programmer, frequently attending and winning hackathons. He was first introduced to HPC through a class on MPI, pthreads, CUDA, and OpenGL. He intends to leverage his experiences by focusing on power management and providing a holistic perspective from the transistor level upwards.

Manas George was a third year computer science student. He is a member of Georgia Tech's High Performance Computing Lab, where he works on cuStinger, a GPU-based dynamic graph processing framework. He was a research intern at Apple where he built a data processing pipeline for the iTunes app submission process. He leads the computer security club at Tech, giving presentations on security vulnerabilities, as well as setting up competitions that involve administering multiple systems under attack over a period of 24-48 hours.

David Meyer is a third year CS major, who became interested in HPC through reading about Beowulf clusters. For the past two years, he has worked as a teaching assistant for courses on Object Oriented Programming and Computer Organization. He is currently developing a web application to elegantly visualize genealogy data. Ever since, he has been searching for a ways to expand his knowledge on HPC.



Jessica Rosenfield was a third year computer science major who has served as the president of the College of Computing's Undergraduate Council and was recently awarded Outstanding Senior in Computing. This semester she is researching high performance computing algorithms to improve DNA alignment and assembly software. Jessica has interned twice with Square where she worked in distributed systems and customer insights. She plans to work at Snapchat this summer in data engineering.

Alok Tripathy was a second year CS major. His research focuses on designing and implementing parallel streaming graph analytics for massively multi-threaded systems. He has interned at Sandia National Labs where he implemented a distributed cache coherency protocol in Go, and at Bloomberg designing and implementing machine learning features to extract tables from PDF files. He is a recipient of the Sidney Goldin scholarship for outstanding leadership abilities. In his free time, he likes to play capture-the-flag (CTF) computer security and competitive programming competitions.

C. Our Advisors

Chirag Jain was a third year PhD student in the School of Computational Science and Engineering at Georgia Tech. His research interest lies in designing scalable serial and parallel algorithms for combinatorial problems in genomics. In 2016, he was a summer fellow at the National Institutes of Health. He is the recipient of the Reproducibility Initiative Award at Supercomputing'16.

Oded Green was a research scientist in the School of Computational Science and Engineering at Georgia Institute of Technology, where he also received his PhD. Oded received his MSc in electrical engineering and his BSc in computer engineering, both from the Technion, Israel Institute of Technology. Oded's research focuses on improving performance and increasing scalability for large-scale data analytics using a wide range of high performance computing platforms.

Will Powell was a research technologist in the School of Computational Science and Engineering at Georgia Institute of Technology. Will has over 20 years experience in the IT industry including almost 12 years at IBM. Will is an expert in designing high performance servers and clusters. He assisted the team with setting up the team's cluster and teaching the students good system administration techniques.

D. SC'17 on Diversity

When evaluating potential teams, the Student Cluster Competition committee included team member diversity and inclusive recruitment practices in their evaluation [32]. Due to the international nature of the competition, there was flexibility in terms of what constituted diversity. However, in general, diversity was broken down into two components: (1) academic diversity and (2) ethnic and gender diversity.

Academic diversity focused on the student's academic majors; in particular, whether there were students from each of the computational sciences, the domain sciences like physics and chemistry, as well as traditional engineering. The intent was to simulate the collaboration habits of traditional HPC environments. Ethnic and gender diversity, at least for American teams, primarily focused on ensuring women and traditionally underrepresented minority groups within computing had the opportunity to join and compete with the team. This was intended to ameliorate the pipeline crisis within HPC and computing, more broadly, where there are limited numbers at all stages of computing [33],[34].

E. Strength of Our Diversity

At Georgia Tech, we pride ourselves on our institution-wide commitment to support and foster diversity in all of its manifestations [35]. Each year, we are consistently rated among the top universities in the nation for graduation of women and underrepresented minorities in engineering, and computer science. More recently, in 2015, we received a $5 million grant from Intel to expand our existing initiatives to recruit and retain qualified underrepresented students in STEM [36]. These initiatives include summer research programs for undergraduates, mentorship programs at the K-12 level as well as at the graduate and undergraduate levels, and outreach programs attracting bright women and underrepresented minorities to attend graduate school [36].

Our team was a direct reflection of this commitment to inclusive excellence. Early January 2017, our advisors posted an announcement on our college's news and events mailing list inviting students to join GT's inaugural Student Cluster Competition team. Soon after, they held an information session describing the competition and expected level of commitment. From the attendees, seven members, six competitors and one alternate, were then selected based on their enthusiasm and prior experience with HPC. What resulted was a very diverse group of students.

Geographically, our team members' backgrounds span four states, six countries, and three continents. Manas grew up in Dubai. Chirag is from Haryana, India. Dr. Green stems from Haifa, Israel. Andy is from Ürümqi, China. Petros was born in Ethiopia. Jessica was raised in Farmington, Connecticut, Alok grew up in Princeton, New Jersey. Nick is from Boulder, Colorado, and David is from Buford, Georgia. Beyond this variety in background, we also share an eclectic mix of interests outside of our work in computing. Members of our team can be found running marathons, playing sports such as squash and baseball, designing fashionable clothing, and hacking gaming consoles.

Appendix II. Sponsorship

F. Vendor Relationship

Our company sponsors were IBM, Avnet, and Flagship. They provided our cluster hardware and funded hardware shipment to Georgia Tech. We received our first node in August and the second in late September. In the ensuing time, IBM also loaned us a

Fig. 3. Map of our Team Members' Backgrounds

Power8 cluster until June. We were able to using this machine to investigate optimizations for HPL and HPCG.

During the months preceding the competition, we worked with IBM, Avnet and Flagship to ensure we utilized our systems to peak capacity. Concretely, this meant receiving

- Binaries for the LINPACK and HPCG challenge that were highly optimized
- Guidance selecting and setting up our file system
- Suggestions on how to reduce power consumption

Appendix III. Team Preparation

We describe how we prepared for the competition.

G. Prior Coursework

As apart of our degrees, we each took coursework on computer architecture, operating systems, data structures and algorithms, and parallel programming. Several members, through their own choosing, also took advanced courses in Numerical Linear Algebra, Operating Systems, Distributed Systems, Processor Design, Compilers, High Performance Computing Algorithms and Parallel Computing Architecture. Note, we not only *enrolled* in these courses, we also *excelled* in them. Most notably, 5 of our 7 members had served as teaching assistants for relevant courses like Data Structures and Algorithms, Introductory Computer Architecture, Introductory Computer Systems and Networks, and Algorithm Design. Other members also privately tutor the above coursework.

H. Available HPC Hardware Resources

We describe the hardware we used to prepare for the competition.

1) When we expected competition cluster: With the support of our industry sponsors, we planned to assemble a two node Power8 cluster. We expected the first node to arrive in early August, right before the start of the Fall 2017 semester. Then, between mid-September and early October, we expected our second node. Because of these arrival dates, we expected at most two months of access to our competition machine.

2) Additional Clusters: To prepare during the ensuing waiting period, we planned various alternative clusters. The College of Computing at Georgia Tech provided us with two Intel Xeon based clusters, PACE and Jinx. PACE is a four node research cluster with each node outfitted with 24 core Intel Xeon E5-2680 CPU. Jinx is a 24 node instructional cluster with each node containing two Intel Xeon X5650 processors and two NVIDIA Tesla GPUs. PACE is shared amongst members of a research group at Georgia Tech and Jinx is shared with various classes. Additionally, we were given access to Top500 machine Comet@SDSC. These servers were used for practicing our skills and developing an intuition for tuning the different applications.

In addition to the aforementioned clusters, IBM loaned us a Power8 server. This server was an older generation of the Minsky server we used at the competition. We used this system to get accustomed to the PowerPC architecture and to properly tune each application for our future server. This way, during the fall, when our server arrives, we will be productive. Note, IBM loaned us this cluster explicitly to prepare for the Student Cluster Competition. Consequently, the team has exclusive access to the server, and each member has sudo access.

I. Preparation for Applications

1) Spring Semester: After forming our team in early January, we held weekly meetings throughout the spring semester. Since

TABLE I
SUPERCOMPUTING 2017 TRAVEL BUDGET

Description	Estimated Cost	Quantity	Frequency	Total Cost
Airport Transport	$10	7-people	4-trips	$280
Airfare	$550	7-people	1-flight	$3850
Per diem	$70	7-people	6-days	$2940
Hotel Room (provided by SCC)	$0	7-people	7-nights	$0
Conference Registration (provided by SCC)	$0	7-people	7-nights	$0
Hardware Transport	$5000	1	1	$5000
Onsite expenses	$1500	1	1	$1500
Total Cost				**$13,570**

many of our members had strong fundamental HPC knowledge, our primary focus during the spring was to familiarize everyone with the applications, and with the practice of tuning an application. Consequently, our meetings largely consisted of discussions on applications and our experiences tuning them. On a number of occasions, faculty and research scientists at Georgia Tech have given talks on topics like the mathematics behind the LINPACK algorithm and the basics of distributed memory parallel computing using Message Passing Interface (MPI).

2) Fall Semester: Once our first cluster node arrived, during the fall, we split our team into groups of two or three members. Each group was responsible for becoming technical leads on a specific application or tool. Each week, subteams worked together to experiment, build tools and conduct investigations. Then, during weekly team meetings, they presented their results and learnings to the larger team. This weekly communication practice served as excellent preparation for the application and team specific interviews at the competition.

3) Communication Channels: Outside of meetings, we communicated using Slack, an instant messaging and collaboration tool. We segmented our Slack into different channels such as proposal writing, HPL benchmark results, Reproducibility Challenge and general announcements. Within each channel, we communicate topically relevant questions, progress and concerns.

J. Institutional Support

Our team's travel expenses were supported by Georgia Tech's School of Computational Science and Engineering (CSE) as well as it's Institute of Data Engineering and Science (IDEaS). This included support shipping our machine to the conference. We provide our projected budget in Table I.