Final Architecture Proposal

University of Washington

October 8, 2019

Hardware Configuration

Our primary hardware configuration consists of two nodes with four nVidia Tesla V100s each. Each node has 256GB of RAM with two Intel Xeon (Skylake) Gold 6130 processors. Our chassis system is the GIGABYTE HPC System G191-H44. Networking will be performed with Mellanox Infiniband 5.X (50GB/s)

One important note about this hardware architecture is that it is and was subject to significant constraints based on our team's operational budget. As the budget continues to evolve over the course of the next month our hardware configuration may slightly change, either adding or reducing our competition capability.

Accelerators

We chose to use the accelerators we did because they are very much the industry standard and Nvidia was offering them to competitors for free. The rest of our hardware configuration is built around utilizing these accelerators to the greatest possible extent.

CPUs

The CPUs we chose are Intel Xeon Gold 6130s (TDP 130w). These CPUs provide a robust computing backbone for HPC, as they each have 16 cores, giving us a sufficient number of threads to properly occupy our GPUs as well as high-performance parallelization in their own right. Furthermore, these CPUs also support high-end SIMD instructions like AVX-512, which gives them a competitive edge over other CPUs which do not support this feature such as older-generation Xeons. Furthermore, the 6130 provides an excellent performance-TDP balance as well as price point efficiency even among the new Xeon Gold Skylake-SP line. This chip should overall be a good fit for our cluster based on these factors.

Memory

Our memory is all registered DDR4-2666MHz in 32GB DIMMs, which should protect our runs from memory corruption to a large extent. We chose to have 256GB of RAM because budget constraints prevented us from getting 384GB per node (32*12) and 256GB is the next logical step down. We plan to experiment between using all 8 32GB DIMMs and just 6 because of memory bandwidth considerations given the lane setup of newer generation Xeon processors (3 lanes, 2 DIMMs each). Both configurations are balanced but it is possible that 3 DIMMs across 3 lanes is faster than 4 DIMMs across 2 in a live system, despite the overall smaller memory size.

Chassis

We chose to use a preconfigured chassis system because it guarantees part compatibility as well as providing us with the ultra-high efficiency power supplies necessary to have as much of our allotted 3000W limit as possible.

Networking

We chose to go with the Infiniband standard because our team was familiar with it and because it was among the most cost-effective ways to achieve a high performance interconnect between our nodes. Since we only have two nodes we do not need a switch, which further saves on total power.

Software Used

The following software decisions were made based on hardware choices, vendor partner feedback, and compatibility requirements for applications. Some software choices are subject to change based on further testing.

Operating System

We plan on using CentOS 7 as our choice of operating system, as recommended by our vendor partner and supported on AWS. Additionally, CentOS is a commonly used distribution by supercomputers on the TOP500, so this is a reflection of real world choices.

Compilers

Based on application requirements and dependencies, we plan on using a mix of Intel Parallel Studio, Clang/LLVM, and the GNU Compiler Collection as our choices for compilers. Multiple compilers and compiler versions can be installed with Spack and can be activated individually with Environment Modules.

Message Passing Interface

We plan on using a mix of Intel MPI, Open MPI, and MPICH, based on application requirements and additional testing.

Scheduler

We plan on using Slurm as our scheduler, as recommended by our vendor partner.

Package Manager

To simplify the installation of multiple compilers, MPI implementations, and dependencies, our vendor partner suggested that we use Spack. Spack can additionally build Environment Module files, for use by lmod.

Environment Modules

We plan on using lmod to manage environment variables for building and running applications. This allows applications to be built and run with specific tools, while avoiding the trouble of hardcoding paths.

Profiling/Tuning

We will use a mix of Intel Parallel Studio's tools (Intel vTune Amplifier), and Nvidia's Nsight, nvvp, and nvprof, to profile our applications to aid optimization.

Monitoring

We will use a combination of Ganglia, htop, and nvidia-smi. Ganglia gives us the big picture of what we've utilized across the entire cluster, htop is useful for seeing CPU utilization and which processes consume the most resources, and nvidia-smi tells us more about what's running on the GPUs.

Application Specific Dependencies

HPL and HPCG

Since we plan to utilize V100s for HPL and HPCG, we will be using binaries provided by Nvidia optimized for Volta architecture GPUs. Using these binaries requires CUDA 10.0 and Open MPI 3.1, and the binaries were compiled with GCC 4.8.5.

SST

SST additionally depends on Python 2.7.

NormalModes

The Artifact Description of the paper mentions the use of Intel 17.0.4 compilers and Intel MPI 17.0.3, so we will use Intel Parallel Studio to build and run the application. Additionally, we will have MATLAB and Paraview installed for the visualization portion of the reproducibility challenge.