ARM Cluster: A Research Tool

Andrew Hoover, Christine Sorensen, and Dr. Christer Karlsson
Mathematics and Computer Science Department
South Dakota School of Mines and Technology
501 East St. Joseph Street
Rapid City, SD 57701
christer.karlsson@sdsmt.edu

Abstract

Recently, a trend to create more computational power has been done by connecting computing platforms into cluster. The initial purpose of this project was to build an ARM cluster single-board computers to make it the fastest, most efficient. Three types of single board computers were tested: Raspberry Pi 2B, pcDuino, and ODROID-XU4. The ODROID-XU4 was chosen and eight were purchased connected. LINPACK, a software that performs numerical linear algebra, was used for benchmarking. The benchmark was designed to fill as much available memory on the eight devices as possible. The results were compared to other connections: USB and GPIO. Ring and hypercube topologies were also compared to the star topology. The purpose of this project has shifted into using the cluster as an education tool. Answering questions such as how computers work and how can we communicate between the computers and benchmark the performance?

1 Background

A computer cluster is a designed single logical unit that contains multiple computers. These computers work together through a network connection and essentially act as one machine with more power, speed, storage, and data reliability. One computer of the cluster holds the software that manages all of the computers. In high-performance clusters, each of the computers perform operations in parallel. This enhances the performance and speed of the applications being ran on them [3].

The TOP500 is a project that ranks the five hundred most powerful supercomputers in the world. These computers perform high-level computations compared the general purpose consumer computers. They rank these by using the LINPACK Benchmark. LINPACK is a software library that performs numerical linear algebra on a computer system, testing every combination of number of cores and size of matrices. It's designed to fill as much available memory on the eight devices as possible and use every core of the machine [1].

As of November 2015, the NUDT's Tianhe-2 (Milky-Way-2) has been placed at number one of the TOP500 list for the sixth time in a row. The Tianhe-2 holds about 3.1 million cores and performed at 33,862.7 teraflops from the LINPACK benchmark. Ten years prior, the top was the IBM's BlueGene/L with more than 65.5 thousand cores and running at 136.8 teraflops. In TOP500's first list released in 1993 placed Thinking Machines Corporation's CM-5/1024 as the top supercomputer with only 1024 cores and 59.7 gigaflops [1]. In just ten years, the speed of these supercomputers have increased by over two hundred fold. The competition for the fastest supercomputer is still a current, running race.

2 Description

The purpose of this project was to build an ARM cluster of 6-12 homogeneous single-board computers with the goal to make it the fastest and most efficient in cost and energy with the intention of showing a proof of concept.

2.1 Choosing a Computer

We decided the metric to measure efficiency in cost and energy would be the number of float-point operations per dollar per unit of power. Three types of single board computers were chosen to be tested: Raspberry Pi 2B, pcDuino, and ODROID-XU4. Before testing took place, the pcDuino was dropped due problems with the operating system. With this

knowledge and the cost of $160 per board led to the disqualification of the computer inferring that it would not excel in the tests.

An Open Multi-Processing (OpenMP) benchmark was installed on the ODROID-XU4 and Raspberry Pi 2B. This benchmark ran mathematical equations, addition, multiplication, division, and sine, on all cores. The results are as shown in Table 3**.**

Even though the Raspberry Pi 2B performed best in the benchmark, the ODROID-XU4 was chosen because the bandwidth potential was higher due to the USB 3.0 ports and it ran 7.4 times faster.

2.2 Building and Benchmarking the Cluster

Eight ODROID-XU4s were purchased and named after Snow White and the Seven Dwarfs, making Snow White the main ODROID. They were connected in a star topology using an unmanaged switch, portrayed in Figure 1. It was decided to use LINPACK, a software that performs numerical linear algebra that is commonly used for cluster benchmarking. However, LINPACK benchmarking software did not exist for an ARM cluster. An open source project High-Performance LINPACK, known as HPL was adjusted to perform on the ARM architecture This led to a side goal; to be able to release an ARM version of the HPL testing as a Debian package to facilitate easier installation in the future.

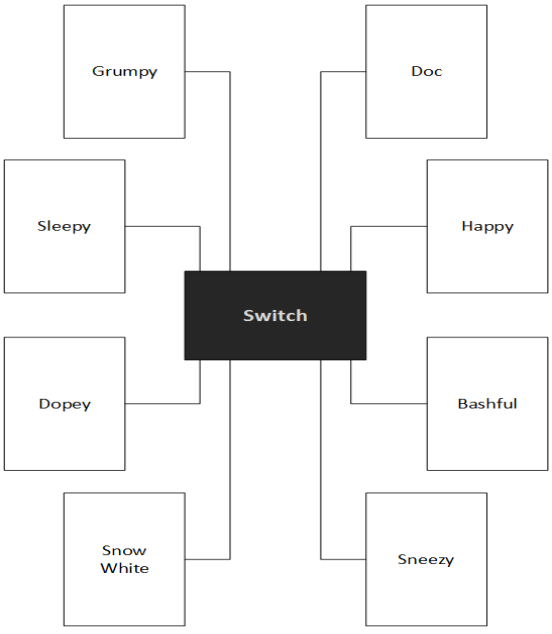

Figure 1: The Star.

To build HPL initially, we first downloaded and built the Automatically Tuned Linear Algebra Software, ATLAS, as well as the Basic Linear Algebra Subprograms, BLAS. One of those would be needed to be linked to build HPL, and we intended to use both and see which gave higher performance. However, BLAS did not build correctly with HPL, and was abandoned. Instead we created a Makefile for HPL to include the correct Message Passing Interface and ATLAS libraries and built the software. HPL includes a configuration file to allow the user to set the matrix size, block size and process grid dimensions. The file was edited such that the matrix size was large enough to use as much memory as possible without requiring swap space to be used. Due to the memory needs of the kernel, this was approximately 1600 megabytes per ODROID. The block size and grid dimensions were found to be most effective at 128 and 2x4 through experimentation [2].

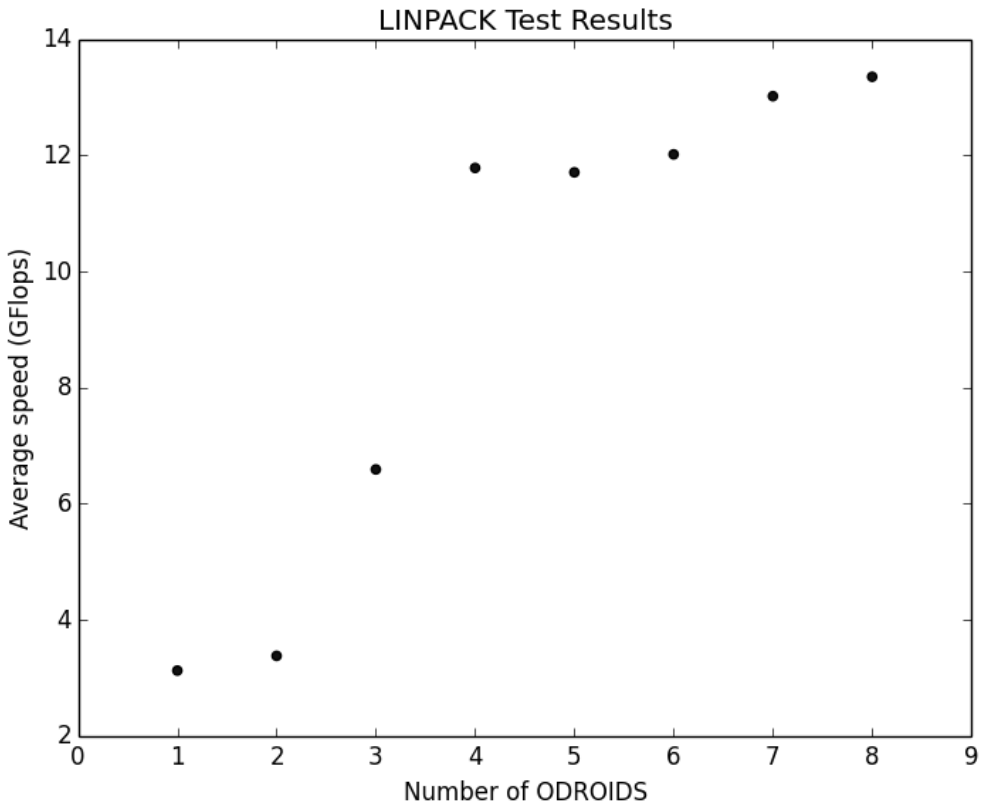

Figure 2: LINPACK results.

The benchmark was then properly configured to fill as much available memory on the eight devices as possible and use every core on each machine as efficiently as possible. We ran the test on every number of nodes from one to eight, and graphed the resulting gigaflops. The results of the initial benchmarking with communication over Ethernet and unmanaged switch is shown in Figure 2**.**

2.3 Further Experiments

The next stage of the project was to compare the results using Ethernet connection against connections using the other means of communications, specifically the Universal Serial Bus (USB) and General-Purpose Input/Output (GPIO). Communication via USB was deemed unattainable. It didn't cooperate with connecting two logic boards together and research wasn't found to troubleshoot the issue. GPIO was successful for single-bit communication, first by editing the system files in the Linux file system associated with the GPIO pins, and by using the WiringPi library in C code. However, GPIO communication was discontinued. The speed of GPIO didn't perform any better than the star topology and connection was far too complex for the timeframe of this project. This was determined by using a short piece of code to write integer values as 64 high or low values on a pin. The integer was converted to an array of binary values first, and then the values were written 10,000 times in a loop. The time required to write the 64,000 total bits was recorded, and found to take roughly 0.127 seconds. For comparison, the Ethernet connection only takes 8.1x10-5 seconds.

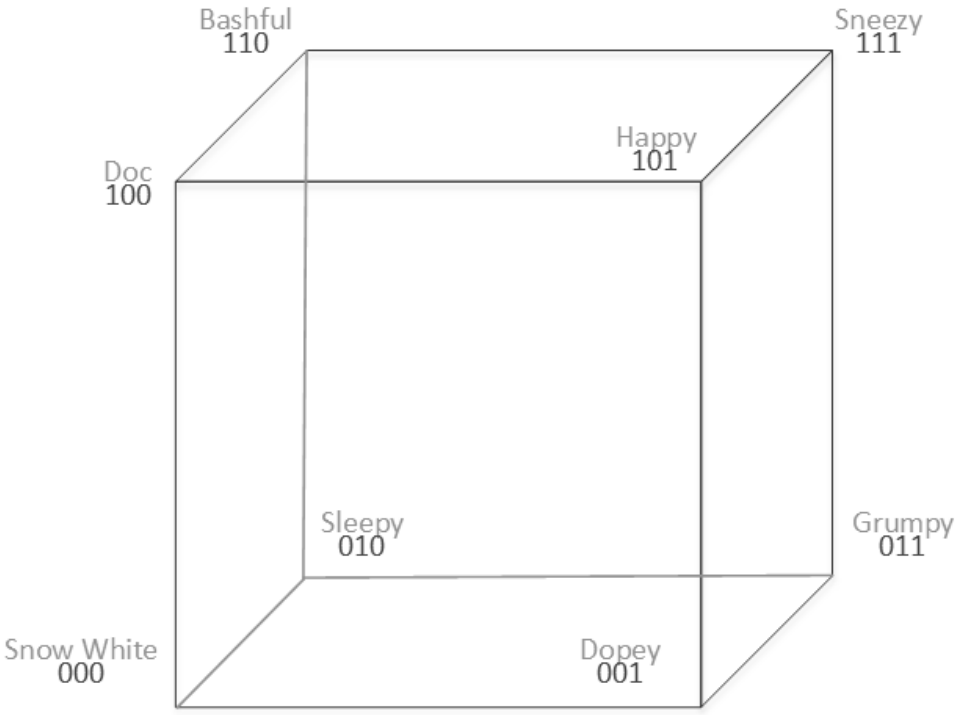

Figure 3: The Hypercube.

As is clear Figure 2**,** the cluster seemed to reach a peak amount of gigaflops with around four nodes. Adding more devices increased the performance, but not by nearly as much

after that point. So we concluded that the bottleneck was in the switch, and adding more connections beyond four was overloading the maximum traffic it could support. Therefore, we moved on to reshaping the topology; first into an eight-point ring, then into a hypercube, shown in Figure 3. In order to accomplish this, we used USB 3.0 to Ethernet devices to add new network interfaces to the ODROIDs. We sought networking advice from Dr. Mengyu Qiao, a professor at South Dakota School of Mines and Technology. He taught us about setting up multiple IP addresses onto a single ODROID and routing tables, allowing us to successfully communicate to each of the ODROIDs in the new topology. We did this by first setting up IP addresses as seen in Figure 4 to the Ethernet port and the USB ports and then creating a routing table for each ODROID.

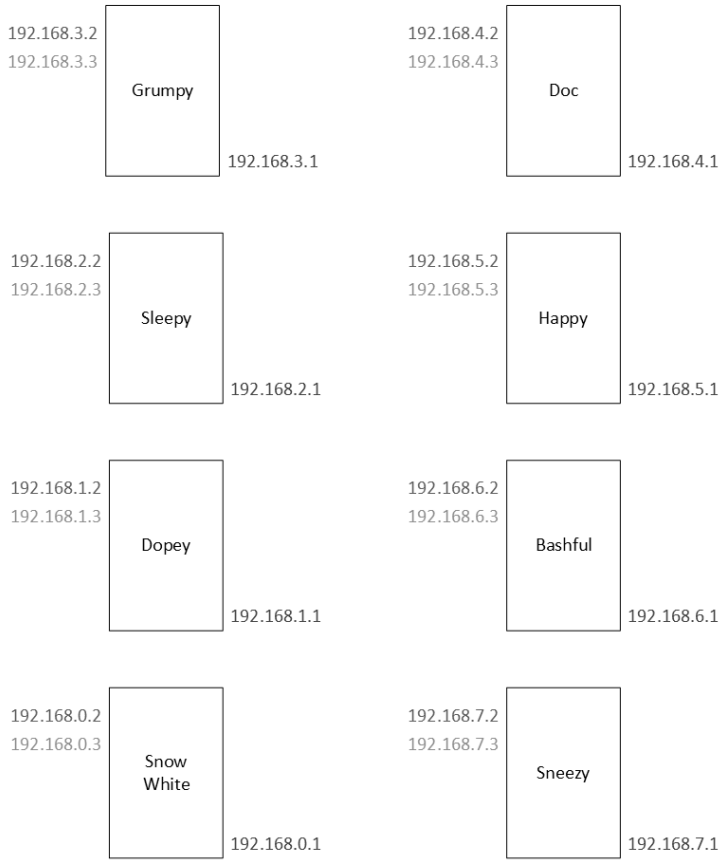

Figure 4: The Ring with IP Addresses.

3 Results

Over the course of completing this project, we used various tests to gather benchmark results in order to determine the clustering method that best matched our criteria. We first benchmarked each device individually that was available to us to decide which to use for our cluster.

3.1 The First Benchmark

The first benchmark we ran was a simple test of how many gigaflops the Raspberry Pi 2B and ODROID-XU4 were capable of. We wrote a benchmark that created a matrix of random floating point values and ran four calculations across all of the numbers, addition, multiplication, division, and sine using OpenMP to utilize all cores. Since our goal was to determine the device that produced the most amount of gigaflops per dollar per watt, we also measured the amount of watts of power used by each device while running these calculations, and found that the Pi used 4 watts, and the ODROID 15. As seen in Table 3 Raspberry Pi 2B and ODROID-XU4 were extremely close in performance due to the Pi being roughly half as expensive and using a quarter of the power, but the ODROID being roughly eight times faster. In the end, the fact that the ODROID had two USB 3.0 ports allowing us more options if how to cluster the devices drove our decision to build our cluster out of ODROIDs.

LENGTH OF TIME (SECONDS)				
DEVICE	**Addition**	**Multiplication**	**Division**	**Sine**
ODROID-XU4	29.925	31.341	37.032	227.40
RASPBERRY PI 2B	221.645	221.034	29.204	1468.63

Table 1: Calculation Speed

GIGAFLOPS				
DEVICE	**Addition**	**Multiplication**	**Division**	**Sine**
ODROID-XU4	0.311	0.297	0.251	0.0410
RASPBERRY PI 2B	0.0420	0.0421	0.0313	0.00634

Table 2: Total Gigaflops

GIGAFLOPS PER DOLLAR PER WATTS				
DEVICE	**Addition**	**Multiplication**	**Division**	**Sine**
ODROID-XU4	0.00028	0.000268	0.000226	0.0000369
RASPBERRY PI 2B	0.0003	0.0003	0.000224	0.0000453

Table 3: Final Gigaflops / Dollar / Watt

2.2 LINPACK Benchmarking

We then implemented the High-Performance LINPACK benchmark in order to use a standard benchmarking method. After installing HPL, we ran it first on a single ODROID-XU4 using all eight cores. In order to get results that were as accurate as possible, HPL was configured to create a matrix that filled almost all of the free memory in order to maximize the amount of calculations done without having to use the swap space. It was found that each device was capable of roughly 3.15 gigaflops.

The next step was to cluster the eight devices. First, a switch was used connected to the 1 gigabit Ethernet ports on the ODROIDs, creating a star topology. Iperf was used to test the data transfer speed of the Ethernet connections, and though there was relatively high variance, the average result was found to be roughly 750-800 megabits per second. The clustered eight ODROIDs were then all used to run HPL. The maximum amount of gigaflops recorded was 13.34 using all 64 cores and HPL re-configured to use all the free memory of all eight of the devices.

2.3 GPIO Connection

With that result in mind, the goal was then to test different clustering methods to try and find a way to produce more gigaflops. First GPIO connections were analyzed. Using WiringPi to read from and write to the pins was successful in transmitting data. To test the speed, we wrote a short program to send a 64 bit integer value from one ODROID to another over a single pin 10,000 times. It was found that even without a delay or waiting for the receiving device to signal that it read the pin, the C code with WiringPi was only able to send about 500,000 bits per second, or 0.5 megabits per second. This means that even using all 42 pins, any communication protocol that could be made would be several orders of magnitude slower than the 750-800 megabits per second that Ethernet was found to be.

2.4 Other Topologies

The creation of different topologies required more testing. Before even creating the topologies, we tested the speed of the USB 3.0 to Ethernet devices we were using. They were tested with the same method using Iperf, with one device connected by its Ethernet port and the other by the USB 3.0 to Ethernet device. It was found that the speed was only 475 megabits per second, significantly slower than 1 gigabit Ethernet to Ethernet. However, due to the speed of the switch slowing the cluster by such a significant degree,

we moved forward with the ring and hypercube topologies, keeping the decreased transfer speed in mind. We would use the results from running HPL on the new cluster designs to determine which topology was superior.

4 Conclusion

The purpose of this project has over time drifted into using the cluster as a hands-on educational tool. The cluster answered questions such as how computers work, how do we build and setup networks, and how can we communicate between computers and benchmark the performance? Also, how can we make this a cost efficient educational tool for teaching parallel computing, networks, or assembly language, for institutions and organizations that normally do not have access to clusters? This cluster will continue on to another group of students to expand it. They will gain more education and be able to answer the questions we weren't able to.

References

[1] Top 500. (n.d.). Retrieved March 20, 2016, from http://www.top500.org/.
[2] Automatically Tuned Linear Algebra Software (ATLAS). (n.d.). Retrieved March 21, 2016, from http://math-atlas.sourceforge.net/.
[3] Computer Cluster. (n.d.). In Techopedia. Retrieved March 19, 2016, from https://www.techopedia.com/definition/6581/computer-cluster.