Distributed Computing Systems with Raspberry Pi

Distributed Computing RPi Final Report

Georgia Institute of Technology – CX 4140

Shreyyas Vanarase – svanarase3@gatech.edu
Tommy Mark – thomas.mark@gatech.edu

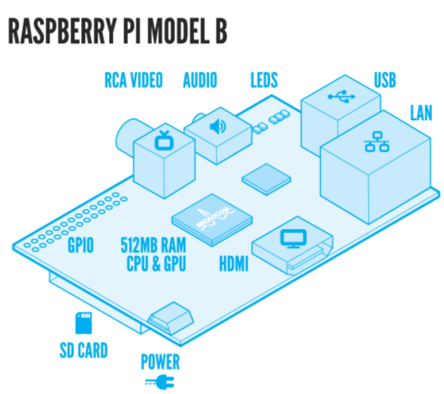
Introduction

Introduction

The purpose of this project is to analyze the performance and power consumption over varying raspberry pi clusters. Raspberry Pis are cheap (~$36) computers the size of a wallet. They have ports for connections to a computer monitor, keyboard, mouse and an Ethernet cable. They are particularly marketed for K-12 educational purposes but recently have been used in a wide variety of research applications from digital signs to entire home automation systems (Edwards 2013). The Ethernet port can be used to connect to both the internet and other Raspberry Pis.

By connecting multiple Raspberry Pis together, their computational power can be combined. Generally distributed clusters use message passing in order to communicate among its various machines. As such, messages can be sent across this distributed system between all the Pis so that parallel computation can be performed. The results of each individual computational result per pi are then sent to a "master" pi for aggregation and further processing. The effects of distributed computing on the clusters will be measured by comparing the running time, cost, and energy consumption of these clusters and benchmarking those results against a standard home computer.

By comparing these results, it can be demonstrated that raspberry pi clusters may provide varying benefits to computation over the standard performance of personal computers for some problem realms and situations. For instance, if the energy usage is much less on a given cluster of Pis, it may be preferable to run algorithms on the pi cluster rather than a PC to save money.



Literature Survey

Today there is a great need for efficient and inexpensive computing resources. As we tackle larger and increasingly difficult problems, the amount of resources and power required to compute solutions has increased dramatically. Up until recently, we have relied on significant growths in the clock speed of processors to improve single CPU performance. However issues relating to heat dissipation prevent us from continually improving clock speeds. Instead, distributed clusters are becoming more popular and seek to overcome the problems associated with single CPU processing. By taking advantage of parallel computing, distributed clusters aim to provide strong performance benefits over just single powerful machines.

Parallel cluster processing relies on the idea that machines in the cluster can be inexpensive commodity hardware. Running parallel programs on these cheap machines have shown that it is possible to radically increase one's computational capabilities. Also because we use slower processors they naturally run at lower voltages meaning that we would optimally obtain a much greater compute power at a much lower power consumption (Spyrou 2009). By maintaining a cluster of many easily replaceable machines, if a single node fails during execution of a program, the entire program does not fail. Comparatively, in a single computer model if the computer fails during execution of a program, the entire job fails.

Given these clear benefits, one might believe that using distributed systems always triumphs over single machines. On the contrary, distributed systems can be inefficient due to high network communication latency among the cluster machines (Dietz 2004). Thus for small input sizes, the distributed system can actually be inefficient since the overhead of communicating on the network is much higher than actually performing the computation itself. To improve the capabilities of these clusters, researchers are continually seeking not only new algorithms to reduce communication costs, but also alternative commodity hardware to provide the best performance boost to the overall distributed system. A few researchers have suggested using Raspberry Pis as hardware for a functional distributed system.

Researchers at the University of Glasgow created a 56-pi cluster and compared the cost and power consumption to a testbed server (Tso 2013). They pointed out that there is a lack of large-scale cost-effective cloud computing infrastructures and that raspberry pi clusters could be used to fulfill that need. Cloud data centers contain many servers that are extremely expensive, not affordable to rent, much less create, for many educators and researchers. Through some basic analysis they concluded that the servers total cost was $112,000 and consumed 10,080 W/h, needing cooling. The raspberry pi cluster only cost $1,960 and consumed 196 W/h, not needing cooling.

Researchers at the University of Southampton created a 64 raspberry pi cluster as an educational exercise (Cox 2013) to introduce students to the world of high performance computing. The power draw of their system was 810 mA (194 V) when idle and 900 mA (216 V) when busy. They performed some benchmark tests and concluded that there is a growing trend toward low-power ARM CPU architectures in the parallel computing world. They used the high performance benchmarking suite LINPACK to determine cluster performance by solving dense systems of linear equations.

The researchers at Southampton also performed I/O benchmarking on their 16 GB Class 10 Kingston SD cards using Hadoop version 1.1.1. 59 Pis were used for computation and 5 Pis were used as data aggregators or administrative nodes. The Hadoop benchmarking they performed used the TestDFSIO benchmark for read and write testing. They concluded that despite the slow I/O transfers (0.38 MB/s/pi), the process mimicked the larger more expensive data center clusters commonly used.

At Bradley University, Aaron Pfalzgraf and Joseph Driscoll benchmarked a cluster of 25 Raspberry Pis (Pfalzgraf 2014) for educational purposes. They conducted three simple tests with basic linear algebraic equations using scalars, vectors, and matrices. During the vector triad problem, they found that their cluster of Raspberry Pis performing in parallel out-performed a serial version of the code once the size of the vectors reached 45,000, highlighting a drawback of parallel computing: the simpler the dataset of a given problem, the more inefficient parallel computing becomes due to communication overhead between the nodes in the cluster. For the matrix-vector multiplication benchmark test, this line occurred at approximately the 1,000 x 1,000 size matrix.

Finally, Joshua Kiepert at Boise State University performed the most rigorous benchmarking found (at the time of this writing) with his cluster of 32 Raspberry Pis (Kiepert 2013) coming to a total of $1,155 for the Pis. Furthermore, he optimized his Pis performance in several ways. He installed Arch Linux instead of Raspbian Wheezy and overclocked his CPUs, increasing performance by ~30% and power consumption. He first performed a basic benchmark, calculating pi using the Monte Carlo method while varying the size of the cluster from 1 to 32 nodes. With one pi it took 456 seconds to calculate. At 3 Pis it took 150 seconds. With all 32 Pis, the time to compute was down to 15 seconds. The power consumed was 102.3 W when the cluster was idle and 127.5 W when it was running at peak performance.

With this project, we wanted to learn for ourselves about the functionality of distributed clusters, implement our own distributed cluster using raspberry pi's, and analyze the efficiency of that RPI cluster. To meet this need, we performed the aforementioned research, constructed an 8-node raspberry pi cluster, and compared the strength of this cluster to a standard laptop and a standard desktop computer.

Further along in this report, we will discuss the objectives of the project, detail the raspberry pi system we developed, explain the benchmarking problems, and analyze our results and understanding.

Objectives

The main goal of this project was to gain experience in distributed computing systems and to understand the performance of a Pi-based distributed cluster to a normal computer. To meet this goal, we set the following objectives for our project:

1) Gain strong understanding of the Raspberry Pi
2) Develop a functional RPI cluster
3) Provide a performance analysis to gain knowledge about distributed clusters

We additionally hoped to answer questions like: what are the performance vs cost tradeoffs between the Raspberry PI cluster and a normal home computer? How many networked Raspberry Pis are required to exceed the optimization capabilities of a standard computer? What are the energy requirements of the cluster and does it complete tasks at a quicker pace than regular computers?

Raspberry Pi Distributed Cluster

The RPI cluster we have developed is comprised of 8 Model B raspberry pi nodes. Each raspberry pi has an ARM1176JZFS @ 700 MHz processor, 512 MB RAM, and a 16 GB SD Card. Since RPis use these SD cards for their operating systems, the size of the SD card seems to work best when it is 8+ GB so that computational work can be properly performed. Each of the Pis run Raspbian Wheezy OS and contain a functional version of Apache Hadoop 2.5.1.

One node is designated as the master node and the remaining are worker nodes, pi01-pi07. While Hadoop computations are performed, every worker node is assigned 1 DataNode and 1 NodeManager. Simply put, DataNodes actually perform the computations being assigned and NodeManagers oversee the success of these computations on a per-node basis. The master node does not run DataNodes or NodeManagers. Instead, it runs the Resource Manager which is the ultimate authority in allocating resources among all the tasks running in the Hadoop cluster. Master node also runs NameNode and Job History Server. Though there may be more than one master node in some larger clusters, here we have decided to simply choose 1 master node.

In Figure 1, we can see the setup of our distributed cluster. From left to right, the first node is master node and the next seven are pi01-pi07.

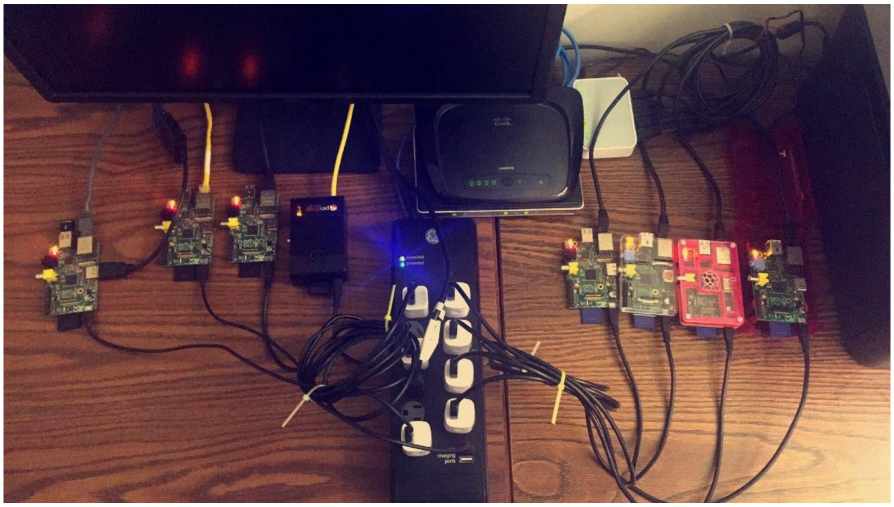


Figure 1: Distributed Raspberry Pi Cluster

Bill of Materials (BOM)

One of the biggest merits to a distributed system that uses commodity hardware is that they can be made relatively cheaply. For example, our 8 Raspberry Pi cluster cost a total of $60.87. For our cluster, we borrowed 3 Raspberry Pis from Prof. Bader and crowd sourced the remaining from members of the CS community here at Georgia Tech. Similarly, spare Ethernet cords, switches, power cables and power adapters cut down the cost as well. The following table shows our expenses for each one of the components involved in making the system.

Material	**Quantity Bought**	**Total Quantity**	**Total Cost (in Dollars)**
Raspberry Pi Model B	0	8	$0.00
Ethernet Cord (3 ft)	5	11	$0.00
Router	1	2	$28.99
Switch	0	1	$0.00
SD Card	3	8	$15.00
Power Cables (3 ft)	1 (Pack of 5)	8	$9.99
Power Adapter	1 (Pack of 5)	8	$6.89
Kill-a-Watt Meter	0	1	$0.00
			$60.87

Table 1: Bill of Materials for RPI

As researchers such as Keipert or Tso have previously demonstrated, with 32 and 56 Pis, respectively, it's quite possible to make a very powerful distributed computing cluster for under $2500.

Benchmarking Problems

Before we analyze our results, we should understand the benchmarking problems that were run on our machines. Benchmarking problems allowed us to determine the computational power of our raspberry pi cluster and compare its performance to that of the laptop and the desktop. By analyzing these benchmarking problems we can determine effective targets for future improvement and assess the factors that contribute to any differences. Each task was run multiple times to ensure the precision of the results. Here, we ran a total of 3 Apache Hadoop benchmarking problems and one CPU test:

1) *WordCount* – The WordCount example reads in text files and counts frequency of each word in the text files.

2) *TeraGen/TeraSort* – The TeraSort example seeks to sort a specific amount of data the user inputs. Generally, TeraSort is used to benchmark how quickly a cluster can sort 1 terabtye of data. Here, we've reduced that size to accommodate for our limited memory RPI cluster.

3) *PageRank* – The PageRank example tries to determine website rankings for search engine results. At a high level, the algorithm counts the number of links and the quality of those links to rank sites. The higher the quality of the site and the larger the number of links, the higher the site is ranked. For this Hadoop cluster, we ran a PageRank algorithm on Stanford University's web graph (courtesy of SNAP).

4) *SysBench* – SysBench is a benchmarking suite that allows one to quickly understand the performance of their computer. This measurement is only for a single node, so we measured the CPU performance of the Raspberry Pi.

In the following section, we will provide an analysis of our results after running these Hadoop benchmarking problems on our cluster (for sizes of 1-node, 4-node, and 8-node), laptop, and desktop. Note that SysBench can only be run on one machine within a cluster so it was simply run on the master RPI Node.

Cluster Performance Analysis

Our analysis of the raspberry pi cluster begins with an understanding of the machines involved in the comparisons. As Table 2 shows below, the desktop and laptop both contain Intel-i7 processors, 4 GB of RAM and run the latest Ubuntu distributions on virtual machines. The raspberry pi is running an ARM processor at 700MHz and has 512 MB of RAM with a Raspbian OS.

Computer Type	Number of Cores	Processor	RAM	Hard Drive Space	Operating System
Desktop Computer	4	Intel Core i7 - 3770K CPU @ 3.50 GHz	4 GB	20 GB	Ubuntu-14.04 (64 Bit VM)
Laptop Computer	2	Intel Core i7 - 4500U CPU @ 1.80 GHz	4 GB	20 GB	Ubuntu-14.04 (64 Bit VM)
Raspberry Pi	1	ARM1176JZFS @ 700 MHz	512 MB	16 GB (SD Card)	Raspbian Wheezy (32 Bit)

Table 2: Computer Specifications

Initially, the 8 node raspberry pi cluster ran without any optimizations and the WordCount benchmark was conducted, achieving a result of 16 min 58s. Then, based on discussions with an experienced raspberry pi user, a number of optimizations were introduced. First, the number of map tasks conducted was limited to one per core. The same was done for the number of reduce tasks. Then the block size for each pi was set to 5 MB (Widrikkson 2014) Finally, DFS replication was set to 1. Table 3 illustrates the optimized, pi results compared to the benchmark PCs.

Benchmarking Problem	Problem Size	RPI 1-Node	RPI 4-Node	RPI 8-Node	Standard Laptop	Standard Desktop
WordCount	35 MB	26m 42s*	21m 55s	8m 35s	18.074s	14.501s
TeraSort	50 MB	-	26m 28s*	7m 45s	10.475s	9.125s
PageRank	33 MB	-	16m 42s*	31m 56s	58.10s	51.43s
SysBench	-	21m 13s	-	-	29.03s	23.58s

Table 3: Optimized RPI Cluster vs Standard Computers

*Note: *Cluster crashed*

WordCount Analysis

Looking at results made by simply varying the size of the cluster, there is an exponential gain in performance the more Pis that are used. Figure 2 below shows an optimization increasing the number of reduces per job that corresponded to a nearly 15 minute decrease in runtime. These results matches up with some of the literature previously mentioned. As far as comparison with the PCs however, the results in Table 3 clearly show that the standard PCs completely out-perform the varying size cluster of Pis. This not only occurs with the WordCount benchmark, but also the TeraSort and PageRank results. This is possibly due to not having enough Pis in the cluster to compete with the processing power of the PCs. The eight Raspberry Pis give us the total processing power of 4 GB of RAM and 8 cores @ 700 MHz. Additionally, the Java heap size on the Pis are 272 MB, much smaller than the standard 512 MB – 1 GB for a RAM size of 4 GB+ on the PC. Finally, the size of the dataset may need to be larger in order to overcome the raspberry pi network communication overhead. This overhead is not present in the PCs since all the communication happens within the main system itself.

Name	User	Queue	State	Maps Total	Maps Completed	Reduces Total	Reduces Completed
word count	hduser	default	SUCCEEDED	7	7	9	9
word count	hduser	default	SUCCEEDED	7	7	11	11
word count	hduser	default	SUCCEEDED	7	7	12	12
word count	hduser	default	SUCCEEDED	7	7	8	8
word count	hduser	default	SUCCEEDED	7	7	10	10
word count	hduser	default	SUCCEEDED	7	7	1	1
word count	hduser	default	SUCCEEDED	7	7	1	1

Figure 2: WordCount Results for 8-Node Cluster

TeraSort Analysis

The results shown in Table 3 are for the optimized runtime of TeraSort. Originally, the un-optimized run of TeraSort yielded a runtime of 8 min 24 seconds for the 8-node cluster as shown in Figure 3. The shuffle time of the un-optimized run was 16, so the number of map tasks per job were decreased to 10. This allows for less time spent assigning work to the Raspberry Pis and less shuffling during the shuffling stage. The difference in shuffle time can easily be seen in Figures 3 and 4 on the following page.

Similar to the WordCount benchmark, the jump from the 4-node cluster to the 8-node cluster showed a large speedup in time. It is unknown how large however because the cluster crashed at the time listed in the above table. Larger datasets were attempted, however the cluster was not able to handle the larger size and crashed shortly after attempting sizes of 100 MB, 500 MB, and 1 GB. These clusters might have crashed because of the set blocksize for each of the Pis. Considering we had a 5 MB blocksize, increasing the block size to potentially 32 MB or 64 MB might be an option to consider for future work on the project.

Job Name:	TeraSort
User Name:	hduser
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Fri Dec 05 02:39:24 EST 2014
Started:	Fri Dec 05 02:40:27 EST 2014
Finished:	Fri Dec 05 02:48:52 EST 2014
Elapsed:	8mins, 24sec
Diagnostics:	
Average Map Time	2mins, 51sec
Average Shuffle Time	5mins, 13sec
Average Merge Time	21sec
Average Reduce Time	28sec

Figure 3: Un-optimized TeraSort on 8-node Cluster

Job Name:	TeraSort
User Name:	hduser
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Thu Dec 04 19:49:47 EST 2014
Started:	Thu Dec 04 19:51:05 EST 2014
Finished:	Thu Dec 04 19:58:50 EST 2014
Elapsed:	7mins, 45sec
Diagnostics:	
Average Map Time	2mins, 37sec
Average Shuffle Time	4mins, 50sec
Average Merge Time	23sec
Average Reduce Time	23sec

Figure 4: Optimized TeraSort on 8-node Cluster

PageRank Analysis

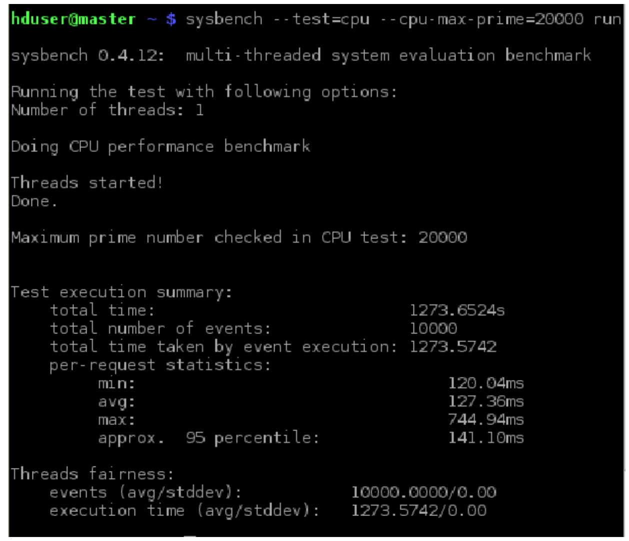
```
hduser@pi04: ~
File  Edit  Tabs  Help
nmon-13g------[H for help]---Hostname=pi04--------Refresh= 2secs ---13:57.39
 CPU Utilisation
                                     +--------------------------------------------------+
 CPU  User%  Sys%  Wait%  Idle|0          |25          |50          |75          100|
  1   88.7   11.3    0.0   0.0|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUsssss>
                                     +--------------------------------------------------+
 Memory Stats
                   RAM      High      Low      Swap
 Total MB         485.0     -0.0     -0.0     100.0
 Free  MB          12.3     -0.0     -0.0       0.0
 Free Percent       2.5%   100.0%   100.0%      0.0%
              MB                  MB                   MB
                      Cached=     2.3      Active=    228.3
 Buffers=      0.1 Swapcached=   20.2   Inactive =    228.5
 Dirty  =      0.2 Writeback =    0.0   Mapped   =      1.6
 Slab   =      5.2 Commit_AS =  706.6  PageTables=      2.1
```

Figure 5: Pi04 Pagerank Runtime Utilization

While neither the 1-node nor the 4-node clusters could complete the PageRank algorithm, the 8-node cluster completed it in 31 minutes 56 seconds. Again, significantly later than either of the benchmarks, however it is interesting to note that the slowdown between TeraSort and PageRank results was less for the 8-node cluster than for either of the PCs (~4.1x slower for the cluster vs ~5.5x slower for the PCs). Figure 5 illustrates an example of a nodes usage during the runtime of this algorithm, highlighting what is expected, the RAM and CPU are fully utilized across the cluster.

SysBench Analysis

This benchmark consisted of verifying all prime numbers up to 20,000 on a single thread. When comparing the SysBench results across the PCs and a raspberry pi there is a clear winner. As expected, the low-cost ARM processor running at 700 MHz is dominated by the higher-end CPUs. The total running time for the Pis was approximately 20 minutes, while on the PCs it took only a few seconds.



```
hduser@master ~ $ sysbench --test=cpu --cpu-max-prime=20000 run

sysbench 0.4.12:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 1

Doing CPU performance benchmark

Threads started!
Done.

Maximum prime number checked in CPU test: 20000


Test execution summary:
    total time:                          1273.6524s
    total number of events:              10000
    total time taken by event execution: 1273.5742
    per-request statistics:
         min:                                120.04ms
         avg:                                127.36ms
         max:                                744.94ms
         approx.  95 percentile:             141.10ms

Threads fairness:
    events (avg/stddev):           10000.0000/0.00
    execution time (avg/stddev):   1273.5742/0.00
```

Figure 6: Output from SysBench for Master Node

Power Consumption Analysis

In addition to calculating runtime performance of a cluster of Raspberry Pis versus standard PCs, power consumption was also measured. The results are illustrated in Table 4 below. The resting state was the power consumption measured with each operating system sitting idle with no actively running benchmark computations.

State	*Machine*	*Volts*	*Amps*	*Watts*	*Hz*
Powered Off	RPI Cluster	119.40	0.00	0.00	59.97
	Laptop	122.40	0.04	1.90	60.00
	Desktop	122.50	0.06	0.50	60.00
Resting	RPI Cluster - 1 Node	119.70	0.09	6.07	59.98
	RPI Cluster - 4 Node	119.70	0.19	13.28	59.97
	RPI Cluster - 8 Node	119.79	0.29	23.50	59.98
	Laptop	121.50	0.24	11.40	60.00
	Desktop	122.10	0.69	80.90	60.00
Peak Computation	RPI Cluster - 1 Node	119.70	0.09	6.28	60.00
	RPI Cluster - 4 Node	119.68	0.20	14.70	59.97
	RPI Cluster - 8 Node	120.40	0.32	26.30	59.95
	Laptop	121.80	0.48	27.70	59.90
	Desktop	122.60	0.92	108.00	60.00

Table 4: Power Consumption across machines

While the voltage and Hertz are the same across the board for both clusters and PCs, differences truly appear when comparing Amps and Watts.

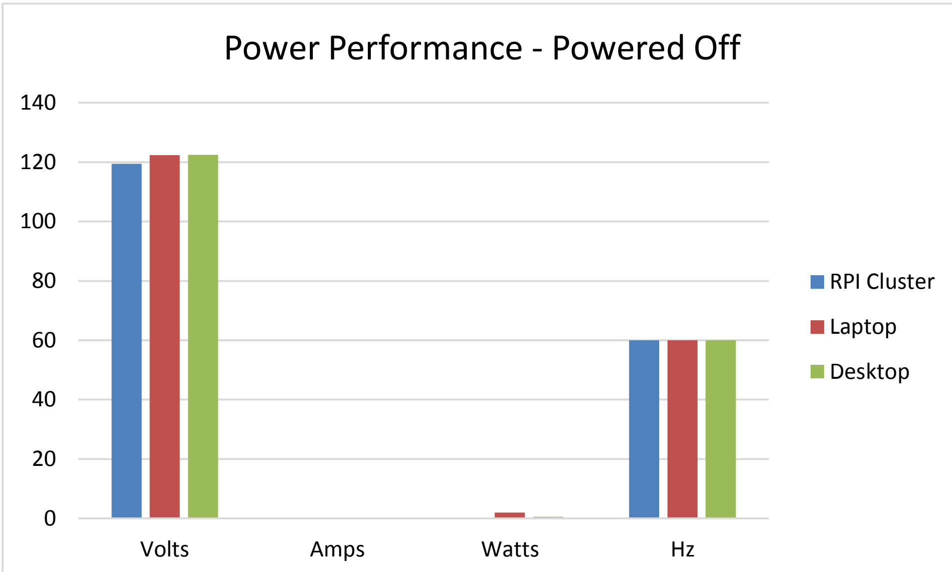


Figure 7: Results for Powered Off Machines

With the machines powered off, the only noteworthy result from Figure 7 is that the laptop consumed 1.9 W despite having a full charge. This is most likely due to powering of a few lights and some interaction with the battery.

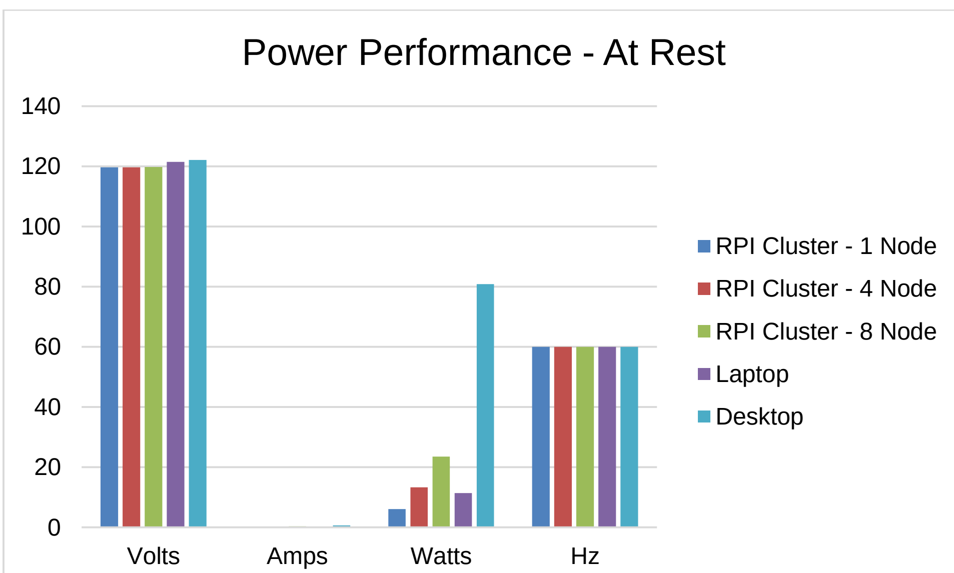


Figure 8: Results for Machines at Rest

Figure 8 shows the results of power consumption at peak computations. One bar immediately stands out. The desktop power consumption in watts is much greater than any of the other machines tested. As expected, the desktop is a power hog. Another interesting result is that the power required for the laptop is much less at idle than the 8 and 4 node clusters. As an Ultrabook it is expected that the laptop will have a relatively low power consumption in comparison to other laptops, but it was not expected to be lower than these pi clusters.

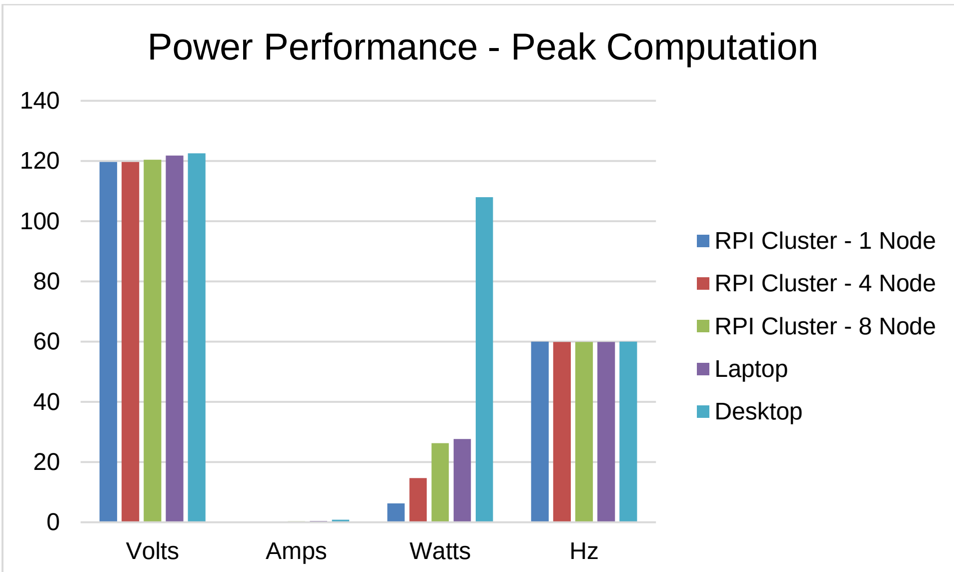


Figure 9: Results for Machines At Peak Computation

The results presented in Figure 9 are very similar to those for the resting state. Peak consumption of power was recorded during one of the benchmarking tests. The only exception is that the laptop consumed a little bit more power (27.3 W vs 26.3 W) than any of the clusters. As expected, the desktop consumed much more power than any other machine during its most computationally intensive moment. Overall, the clusters consume much less power than the standard desktop PC but are on par with the laptop.

Future Work

Hopefully this research inspires others as well to continue with our work. Future work would involve running these benchmarks with larger raspberry pi clusters and larger datasets to fully illustrate the parallel computing advantage of a cluster. Obtaining more Pis would allow for a precise determination of when these pi clusters become as computationally powerful as both the laptop and desktop. Based on past research, it seems that approximately 25 Pis are needed to overcome this difference. In addition, further optimizations of the Pis could be made such as increasing the blocksize for larger data inputs. As Kiepert performed with his Beowulf 32 node cluster, installing the bare bones Arch Linux OS over the more bloated Raspbian Wheezy might free up a little more RAM for processing power. Further optimizations could also be made with tuning Hadoop and benchmark algorithm parameters.

Conclusion

The purpose of this project was to compare low-cost raspberry pi clusters with standard PCs. We benchmarked these clusters using multiple algorithms, including: Hadoop WordCount, Hadoop TeraSort, Hadoop PageRank and SysBench. The PCs completely outperformed even the largest cluster tested (8 nodes) by several orders of magnitude. Based on current literature, in order to scale to a standard PC a cluster of approximately 25 Pis would be needed. While the power consumption of the 8 node cluster was much less than the power consumption of the desktop, the 8 node cluster was on par with the laptop. For distributed clusters in general, we can say with some certainty that a cluster will win out in performance compared to a single standalone computer due to its ability to scale dynamically with larger datasets and offload parallel computations to its nodes, decreasing the communication overhead between nodes.

References

1. Brian, Dye. "Distributed Computing with Raspberry Pi." *Kansas State University*. N.p., n.d. Web. 14 Sept. 2014. <https://krex.k-state.edu/dspace/bitstream/handle/2097/17612/BrianDye2014.pdf?sequence=1>.

2. Cox, Simon, James Cox, Richard Boardman, Steven Johnston, Mark Scott, and Neil O'Brien. "Iridis-pi: A Low-cost, Compact Demonstration Cluster." *The Journal of* Networks, Software Tools and Applications 10.1007 (2013). Springer.com. Springer Science and Business Media. Web. 4 Dec. 2014. http://link.springer.com/article/10.1007/s10586-013-0282-7/fulltext.html

3. Dietz, Hank. "Linux Parallel Processing HOWTO." Linux Parallel Processing HOWTO. 8 June 2004. Web. 1 Dec. 2014. <http://www.tldp.org/ HOWTO/Parallel-Processing-HOWTO.html>.

4. Edwards, C. (2013). “Not-So-Humble Raspberry Pi Gets Big Ideas”. *Engineering & Technology* 8.3, 30-33.

5. Kiepert, Joshua. "Creating a Raspberry Pi-Based Beowulf Cluster." 22 May 2013. Web. 14 Sept. 2014. <http://coen.boisestate.edu/ece/files/2013/05/Creating.a.Raspberry.Pi-Based.Beowulf.Cluster_v2.pdf>.

6. Leskovek, Jure. "Stanford Web Graph." *SNAP: Network Datasets:*. Stanford University, 8 Oct. 2008. Web. 2 Dec. 2014. <http://snap.stanford.edu/data/web-Stanford.html>.

7. Pfalzgraf, A.M.; Driscoll, J.A., "A low-cost computer cluster for high-performance computing education," *Electro/Information Technology (EIT), 2014 IEEE International Conference on* , vol., no., pp.362,366, 5-7 June 2014

8. Tso, F.P., White, D.R., Jouet, S., Singer, J., and Pezaros, D.P. “The Glasgow raspberry pi cloud: a scale model for cloud computing infrastructures.” First International Workshop on Resource Management of Cloud Computing (CCRM), 8-11 Jul 2013, Philadelphia, PA, USA.