PERFORMANCE, MANAGEMENT, AND MONITORING OF 68 NODE RASPBERRY PI 3 EDUCATION CLUSTER: BIG ORANGE BRAMBLE (BOB)

J. Parker Mitchell
The University of Tennessee
1520 Middle Drive, Knoxville, TN, USA
jmitch47@vols.utk.edu

Aaron R. Young
The University of Tennessee
1520 Middle Drive, Knoxville, TN, USA
ayoung48@vols.utk.edu

Jordan Sangid
The University of Tennessee
1520 Middle Drive, Knoxville, TN, USA
jsangid@vols.utk.edu

Kelley A. Deuso
The University of Tennessee
1520 Middle Drive, Knoxville, TN, USA
kdeuso@utk.edu

Patricia J. Eckhart
The University of Tennessee
1520 Middle Drive, Knoxville, TN, USA
pdraney@vols.utk.edu

Taher Naderi
The University of Tennessee
1520 Middle Drive, Knoxville, TN, USA
tnaderi@vols.utk.edu

Mark Dean
The University of Tennessee
1520 Middle Drive, Knoxville, TN, USA
markdean@utk.edu

ABSTRACT

High performance computing clusters have become the de facto standard for addressing large scientific and commercial applications, yet there continues to be many challenges with building and using these system structures including cost, power consumption, application scaling, systems management, and parallel programming. The Big Orange Bramble (BOB) project, influenced by the smaller-scale Tiny Titan of Oak Ridge National Laboratory, was created to provide students, faculty, and researchers a low-cost platform to study and find solutions to these challenges. BOB involved the design and construction of a high performance cluster composed of 68 quad-core ARMv8 64-bit Raspberry Pi 3s featuring one master node, 64 worker nodes, a monitor node, and two storage nodes. Beyond the primary target of delivering a functional system, efforts were focused on application development, performance benchmarking, and delivery of a comprehensive build and usage guide to aid those who wish to build upon the efforts of the project.

Keywords: Raspberry Pi, micro-node, ARM, cluster, education.

SpringSim-HPC, 2017 April 23-26, Virginia Beach, VA, USA;

1 INTRODUCTION

As current computing technology evolves, there is an ever increasing need for students in the computational sciences field to have more exposure to local test bed parallel architectures. The main hindrance to this can be attributed to accessibility, predominantly due to monetary constraints. The Big Orange Bramble (BOB) project was conceived with The University of Tennessee, Knoxville's Electrical Engineering and Computer Science (EECS) department's request for a low-cost, high performance cluster for students and faculty of the department to readily maintain and utilize. The initial design of BOB was inspired by a 2015 project from the Oak Ridge National Laboratory (ORNL) called Tiny Titan, which implemented 9 Raspberry Pis (Model B+) as a $1,000 "classroom" cluster (Simpson 2013). Expanding on this, BOB's construction incorporated 68 Raspberry Pi 3's, a cluster size that was deemed sufficient for the purposes of the EECS department. Additionally, BOB represents a departure from the typical model of high performance computing in that it relies upon what may be called a micro-node architecture. This architecture uses relatively computationally weak processing nodes in a large array in an attempt to achieve higher computational throughput for highly parallel tasks. This goal of this paper is to detail BOB's hardware and system components, performance evaluations, and its respective usefulness and viability in high performance computation education.

2 HARDWARE

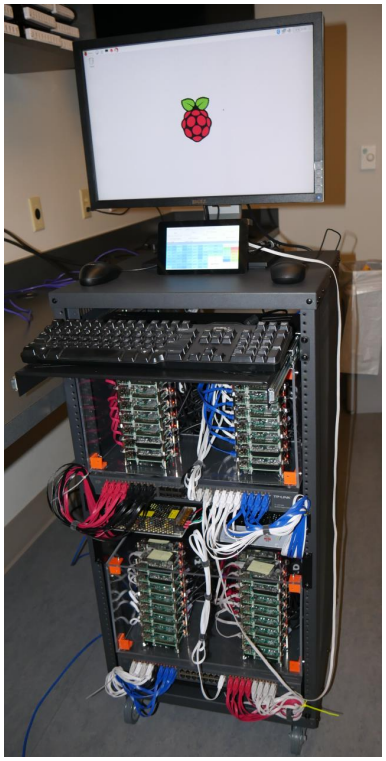

Figure 1: Picture of the BOB cluster

2.1 Hardware layout

The cluster consists of 68 Raspberry Pi 3's, of which 2 are for storage, 1 acts as a monitor node for the system, 64 are worker nodes, and 1 is the master (or head) node responsible for distributing jobs and resources. Each Raspberry Pi 3 has a 1.2 GHz quad core ARM Cortex A53 CPU along with 1 GB of RAM and 100

Mbps (million bits per second) Ethernet NIC. The 64 worker nodes are stacked together in groups of 8 in the enclosure. The monitor node has its own touchscreen interface which shows current CPU temperatures and loads for all nodes. The head node is connected to a monitor via HDMI to allow direct access to BOB as well as display any graphical data produced by the cluster. Each node is connected to one of the two Ethernet switches, and the 2 switches are connected to each other through a single Gigabit link. The maximum network throughput for any individual node is 100 Mbps.

2.2 Power and Daughter Card

The cluster is powered by ten 100W, 5V switch-mode power supplies. Each node has been fitted with a custom designed power monitoring daughter card. The daughter card monitors both the supply voltage and current of each node by using a high side shunt resistor in series with the power supply and the Raspberry Pi in addition to a Texas Instruments INA219 current monitoring integrated circuit (IC). The INA219 has an integrated ADC, registers, and I^2C interface. The information gathered from each node is sent to the monitoring node at predetermined intervals allowing for the information to be analyzed visually by the user. Because the daughter card powers the Raspberry Pi from the GPIO port and bypasses the input power conditioning, each node lacks over voltage/current as well as reverse polarity protection. However, the daughter card does include a supply voltage decoupling capacitor in order to filter some ripple and switching noise from the power supplies. The daughter card schematic diagram is located in **Appendix A.1**.

3 SYSTEMS AND FRAMEWORKS

A variety of software packages and frameworks were necessary to build, operate, and maintain BOB. In this section, some of the important systems are described.

3.1 Slurm

Slurm stands for Simple Linux Utility for Resource Management ("Slurm Workload Manager" n.d.). Slurm is an open source system designed for highly scalable cluster management and job scheduling. Job scheduling and resource management is necessary to afford multiple users the ability to use the cluster without interfering with each other. A job scheduler is also needed to efficiently manage resources so multiple jobs can run in parallel allowing for maximum usage of the cluster. There are many cluster management systems (CMS) available; including Slurm, openSSI, and TORQUE. After looking into the other options, Slurm was chosen for a variety of reasons. Slurm is a modern CMS and "as of the June 2016 Top 500 computer list, Slurm was performing workload management on five of the ten most powerful computers in the world including the number 2 system, Tianhe-2 with 3,120,000 computing cores" ("Slurm Commercial Support and Development" n.d.). Additionally, the Slurm daemon that runs on the worker nodes is very lightweight and only runs on job start and signaling. Using a lightweight manager like Slurm on the Raspberry Pis is particularly important as each node does not have a significant amount of extra RAM to spare. Furthermore, Slurm does not require any kernel modifications and is relatively self-contained. It is able to provide all the functionality expected from a CMS. There were no implementation challenges, and Slurm was installed using the Debian package manager.

Slurm has one centralized control manager which runs the `slurmctld` daemon. This daemon monitors the resources and work. An optional backup `slurmctld` daemon may be configured to act as the control manager in the event the main controller goes down. Furthermore, a dedicated database node that keeps up with accounting information for the cluster is also supported, but a separate accounting node is not required.

Slurm supports an arbitrary number of worker nodes, and the worker nodes can be configured in different network topologies. Each of the worker nodes runs the `slurmd` daemon.

3.2 Ansible

To quickly deploy configuration changes and facilitate initial setup, a method for parallel communication from the head node to the worker nodes is needed. Parallel ssh (pssh) was first option investigated. As its name implies, it allows for running ssh calls in parallel, but it lacks any built-in system management tools. The next method reviewed for parallel communication was Ansible ("Ansible Is Simple IT Automation" n.d.), and it was clearly the better tool for the job. Ansible runs over ssh connections, can run in parallel, and only needs to be installed on the head node. Additionally, Ansible is designed to deploy applications and manage systems. Ansible can be used to run ad-hoc commands to quickly run one-off commands such as copying an executable file or restarting the system. However, the real power of Ansible comes from the modular playbooks that define system roles. Using these playbooks, different configurations can be set up and applied to different components of the cluster. Each component of the cluster is set up as a role. For example, one role is the "head" node, which sets up the configuration needed to run the cluster. Another role is "worker," which performs all the setup unique to worker nodes in the cluster. Other roles can define optional setups such as using NFS or OrangeFS as the file system.

Ansible works using modules that define an operation to be performed. The modules are basically small programs that are copied to the target machine over an ssh connection and then run to perform a task. There are many built-in modules to perform any common action and custom modules can also be created. For example, the copy module is used to copy files, the command module is used to run a shell command, and the apt module is used to configure packages. The modules are smart in that they only execute if there is a change to be made. After running a module the result is "OK" when there was no change to the system, "Changed" if a change was made to the system, and "failed" if the module execution encountered an error. Playbooks are used to organize and automate the modules to be run. The playbooks are written in YAML, a language which allows the playbooks to be stored in a human readable format. It also allows filtering on the target machines so that the playbook can be reran only on the machines that failed to run. The team uses Ansible scripts to set up all of the nodes in the cluster and also to restart or shutdown the cluster.

Ansible was simple to set up and can be installed via `apt` or `pip`. One caveat to note is that the version found in `pip` is much newer than the one found in `apt`. BOB was switched to the newer `pip` version after discovering that the desired modules were not available in the older `apt` version.

3.3 LDAP

One of the requirements for Slurm—and for most clusters—is to have a unified namespace for users, groups, and home areas. The cluster needed a way to synchronize accounts and to provide an easy structure for adding new users and groups. There are many ways to go about setting this up.

LDAP is a standard application protocol for accessing a distributed directory over the Internet. There are multiple back-end databases to choose from, however, to LDAP the database structure is organized into a tree with each entry in the tree having a unique identifier or distinguished name (DN). Each DN is a child of another DN with the top level being the database root. Each entry contains a set of attributes which defines the entry. Entries in the directory have different types; some are purely for organizing the directory, while others represent POSIX groups or users. The tools for modifying the LDAP directory directly are cumbersome and only need to be used to set up LDAP. Once established, scripts can be used to automate all

the commands needed to change the users and groups. With the scripts, it is as simple to use LDAP as it is to use the built-in commands for users and groups.

3.4 File Systems

Network File System (NFS) is a purely distributed file system protocol initially developed for Sun Microsystems. It has since become the standard file system in traditional Unix-based networks. Being purely distributed, the file system uses a protocol to access storage instead of sharing block level access to the storage. This access can be restricted on both clients and servers through the use of access control lists. Files are accessed using the same interfaces and semantics as local files. NFS aims to be transparent in access, location, concurrency, and failure while being heterogeneous and scalable. The most current version of NFS is NFSv4, which is designed to support clustered server deployments and has the ability to provide scalable parallel access to files distributed across multiples servers.

Raspbian Jessie, the operating system chosen for BOB's Raspberry Pis, uses the standard Linux kernel and is pre-installed with NFSv4. Because of this, NFS only needed to be instantiated on the Raspberry Pis to then become the implemented file system. NFS required no installation and minimal setup to become functional. Therefore, it was the file system deployed on the initial subset of operational nodes for BOB. The simplicity of the NFS setup allowed for application development and power analysis to begin almost immediately.

4 TESTING

In order to assess the usefulness of BOB, several HPC relevant benchmarks were used to evaluate the performance of the cluster. High Performance Linpack (HPL), High Performance Conjugate Gradient (HPCG), and the OSU MPI Benchmarks were used to create an overall picture of the expected performance of BOB.

4.1 HPL

HPL (High Performance Linpack) is a benchmark to determine the upper bound of double precision floating point performance on a distributed parallel system. It solves a random dense linear system $Ax = b$ with LU decomposition, and it checks the accuracy after the calculation to ensure a valid result. Dense linear algebra calculations are applicable to many problems and a good method to measure peak performance for a system (Petitet n.d.).

HPL requires an MPI and BLAS library. The BOB system is using MPICH for message passing and OpenBLAS as the linear algebra library. OpenBLAS was selected over some alternatives due to its high performance on ARM CPUs. In this case, OpenBLAS resulted in nearly an order of magnitude higher performance than ATLAS for the Raspberry Pi 3 SoC.

For each test, the block size (NB) was set to 100 which was found to offer favorable performance. The process grid ($P \times Q$) was set to be as close to square as possible. The matrix size (N) for each test was set to be approximately $\sqrt{10500^2 \times \#nodes}$. A matrix of 10500 per node was found to use most of the Pi's available RAM and offer the highest overall performance.

From the demonstrated performance in Graph 2, HPL successfully scales across all 64 nodes with progressively smaller gains as additional nodes are added to the application.

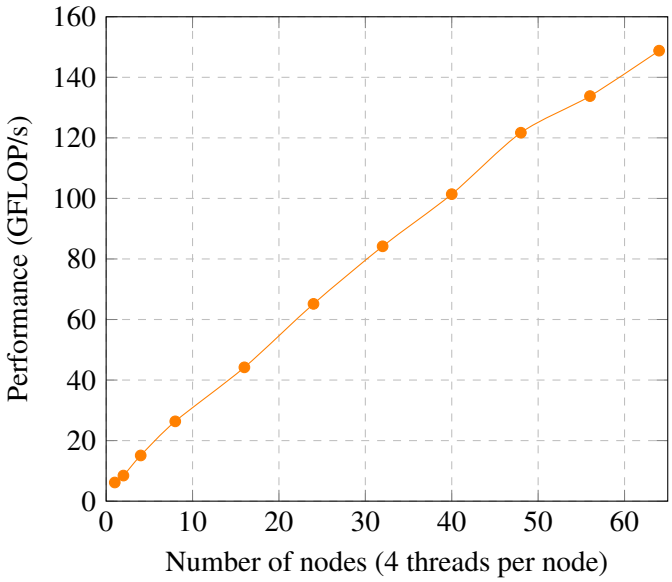

Figure 2: HPL Performance

Table 1 displays the detailed data from HPL testing. Limited by the slow 100 Mbps Ethernet interconnect, a scaling efficiency of 37.7% across the cluster is not surprising. However, a total performance of 148.8 GFLOPs (billions of floating point operations per second) does represent a useful amount of performance in line with or greater than many personal workstations.

Table 1: HPL Performance

# Nodes	N	Time (s)	GFLOPs	Scaling Efficiency
1	10500	125.21	6.165	100%
2	15000	266.35	8.449	68.5%
4	21000	409.25	15.09	61.2%
8	30000	682.86	26.36	53.4%
16	42000	1117.47	44.20	44.8%
24	51000	1357.19	65.16	44.0%
32	59000	1626.52	84.18	42.7%
40	66000	1890.25	101.4	41.1%
48	73000	2131.70	121.7	41.1%
56	79000	2454.77	133.8	38.8%
64	84000	2655.07	148.8	37.7%

4.2 HPCG

HPCG, or High Performance Conjugate Gradient, is a recent benchmark designed to be used in conjunction with HPL to provide a more complete picture of the performance of a cluster.

Like HPL, HPCG solves $Ax = b$; however, HPCG uses a sparse matrix representation using an iterative conjugate gradient method rather than LU decomposition. The end result of these changes is a benchmark which tries to capture lower bound performance rather than peak performance. Using both HPL and HPCG,

one can determine the upper and lower bounds for typical cluster applications. Most programs will not achieve HPL levels of performance but will be able to extract more performance than HPCG. This makes the combination of the two valuable when evaluating the performance of a cluster system (Dongarra n.d.).

In contrast with HPL, HPCG scaled very well as seen in Figure 3. The scaling was nearly linear as HPCG was run on anywhere from a single node to all 64 worker nodes.

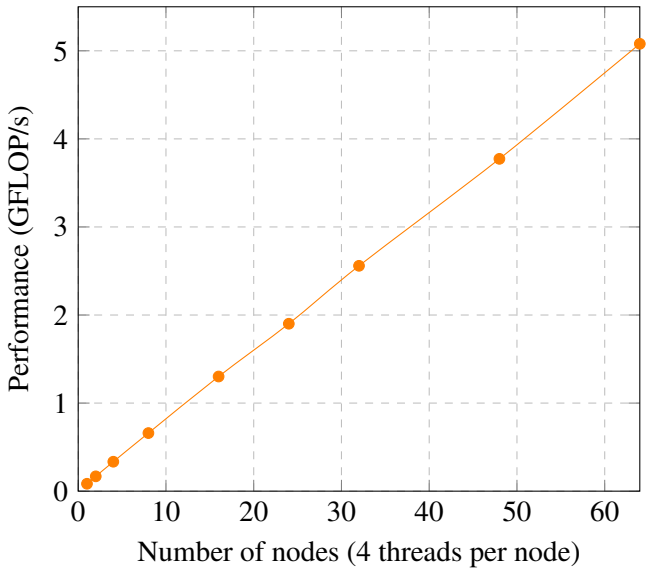

Figure 3: Reference Implementation HPCG Performance

Table 2 shows the full results of HPCG testing. Even across all 64 nodes, the scaling efficiency was in excess of 94% compared to just above 37% when using all 64 nodes in HPL. However, it is important to note that peak HPCG performance was much lower than peak HPL performance.

Table 2: Reference Implementation HPCG Performance

Nodes	GFLOPs	Scaling
1	0.08402	100.00%
2	0.16824	100.12%
4	0.33422	99.44%
8	0.65953	98.12%
16	1.30178	96.83%
24	1.90108	94.27%
32	2.55881	95.17%
48	3.77325	93.56%
64	5.07949	94.46%

This excellent, near linear scaling is possible when the computation and communication can be overlapped with the computation time exceeding the required communication. Thus, applications with this type of communication pattern are particularly suited to a system like BOB with several nodes but with limited networking performance between the nodes.

4.3 Networking Benchmarks

In order to characterize the performance of the 100 Mbps Ethernet interconnect and its impact on MPI performance, benchmarks from the Ohio State University were used to profile the MPI performance ("OSU Micro-Benchmarks" n.d.). In this report, data for the point to point bandwidth and the broadcast call latency is included.

Figure 4 demonstrates the throughput using point to point communication across the cluster interconnect. The maximum throughput correlates with the expected maximum throughput of the Ethernet interface. At transfer sizes of 2KB and larger, the transfer throughput is in excess of 11 MB/s.

Figure 5 also demonstrates the latency in a blocking broadcast collective communication. The graph shows the effect of running a broadcast across additional nodes. Because the cluster has limited bandwidth between the two collections of 32 nodes, the 64 node configuration demonstrated the lowest overall performance and also yielded some interesting artifacts in the latency curve.

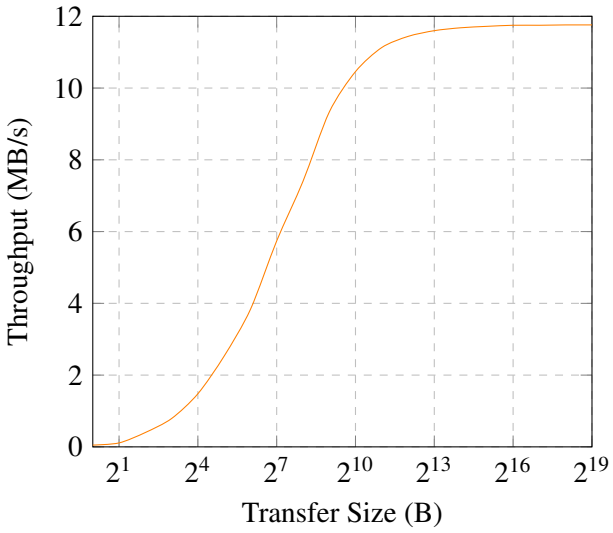

Figure 4: MPI Point to Point Performance

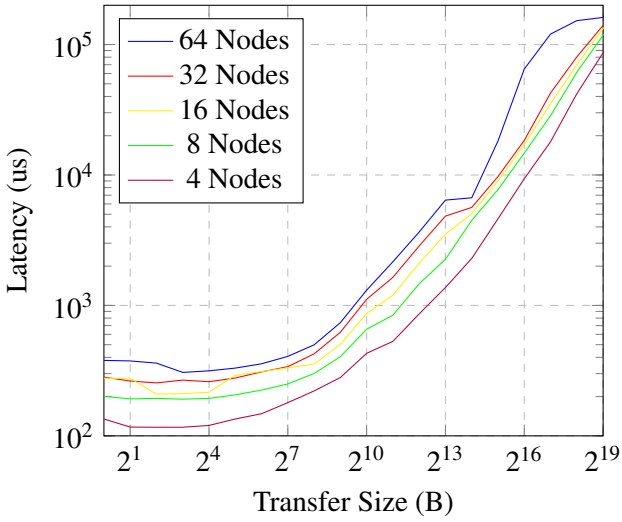

Figure 5: MPI Broadcast Performance

Due to the low performance of the 100 Mbps Ethernet networking available on each node, it necessary to be mindful of communication overhead when running and developing parallel applications for BOB. It is important to overlap communication and computation by using non-blocking MPI communication, and one should also take care to try to maximize the throughput of any communications. With point to point sends and receives, using 2 KB and larger messages appear to optimize the throughput of the interconnect. For collectives like `MPI_Bcast`, the performance is dependent on both the number of nodes participating and the size of the message.

4.4 Power Utilization

With the power monitoring capable daughter card described in section 2.2, power consumption data was collected both during idle and load states. This data was collected on a per-node basis and neglects the power of the network switches and power supply losses. In order to evaluate the power usage of the cluster, voltage and current measurements were collected from each node every second and aggregated back to the monitoring node. This simple method involved very low overhead both for each node and the network.

Table 3 displays the testing results for a single node. The idle situation is the node running only the standard background services. The HPL situation is the node individually running HPL. The typical power numbers represent the average of all samples collected over the duration of the test. For the idle test, this was arbitrarily selected to be one minute.

Table 3: Overview of Power Consumption Per Node

Situation	Typical (W)	Maximum (W)
Idle	1.48	–
HPL	5.16	7.38

Table 4 displays the testing results for all of the worker nodes. As with table 3, the idle test consists of all the nodes running only the standard background services. For the HPL test, all 64 worker nodes are collectively running HPL in an 8×8 process grid. It should be noted that the power used in this scenario is not simply 64 times the individual node power usage. Because HPL requires communication between nodes, there are periods of time in which the nodes are waiting for an MPI communication to complete which results in less CPU utilization and thus lower power consumption.

Table 4: Overview of Power Consumption For All Workers

Situation	Typical (W)	Maximum (W)
Idle	95.04	95.78
HPL	233.0	442.4

5 RELATED WORK

Similar projects include Oak Ridge National Laboratory's Tiny Titan project, the University of Southhampton's Iridis-Pi project, and the University of Maine's Pi 2 cluster. The Big Orange Bramble project attempts to build on these previous Raspberry Pi based clusters. By using Raspberry Pi 3 boards, BOB is able to offer an order of magnitude higher performance in HPL than the University of Maine cluster which used 24 Pi 2 nodes (Cloutier 2016). BOB also performs about two orders of magnitude better in HPL compared to the Iridis-Pi project which utilized 64 Pi nodes (Cox 2013).

6 CONCLUSION

In conclusion, the Big Orange Bramble cluster (BOB) offers a low cost educational platform for learning the details of high performance computing. The goal of this cluster was to learn how to build a parallel system and then offer that system to faculty and students for educational purposes. Through the construction of the cluster, the team gained many valuable lessons about cluster management and performance characteristics. Now, BOB has been successfully used for weather forecast modeling, facial recognition, neural network generation, stochastic equation simulation, and other uses. The success of BOB has also had led to the development of a second cluster titled ALICE which includes 32 PINE A64+ worker nodes and 12 Nvidia Jetson TX1 worker nodes. ALICE attempts to leverage the lessons learned through the development and analysis of BOB to create an improved cluster with higher RAM available per node, a higher speed interconnect (Gigabit Ethernet vs 100 Mbps Ethernet), and the opportunity for GPGPU programming via the Jetson TX1's GPU.

The findings and performance implications for BOB can be extended beyond just a bramble of Raspberry Pis. In the future, it is both possible and likely that some parallel systems will take advantage of a large number of inexpensive and low power ARM based nodes. Each of these nodes will, like BOB, have limited

resources in comparison to a more traditional Intel powered node. BOB demonstrates some feasibility for a simple Beowulf cluster of ARM cores, but this project also exhibited some of the potential drawbacks associated with such an approach. For BOB, the low amount of RAM and insufficient interconnect performance posed challenges for scaling some applications like the HPL benchmark. However, applications like the HPCG benchmark are able to scale at nearly linear rates across the cluster. One primary takeaway for this architecture is that performance is highly dependent on the application, and some applications can expose bottlenecks earlier and more quickly than with a more traditional system. Going forward, it is important to continue research on many-core ARM based HPC systems in order to identify best practices along with optimal applications to take advantage of the hardware.

Additional details about the Big Orange Bramble project are available on Dr. Mark Dean's web page (http://web.eecs.utk.edu/~markdean/).

ACKNOWLEDGMENTS

This paper was made possible through support from the University of Tennessee and Dr. Mark Dean. Dr. Dean provided the necessary funding, vision, and guidance to successfully build BOB. Additionally, BOB would not be possible without the full team of graduate and undergraduate students who collaborated over the Summer of 2016. The full team includes Gregory Simpson, Shawn Cox, Kelley Deuso, Patricia Eckhart, Chencheng Li, Liu Liu, Parker Mitchell, Taher Naderi, Jordan Sangid, Sepeedeh Sepehr, Caleb Williamson, and Aaron Young.

A APPENDICES

A.1 Daughter Card Schematic

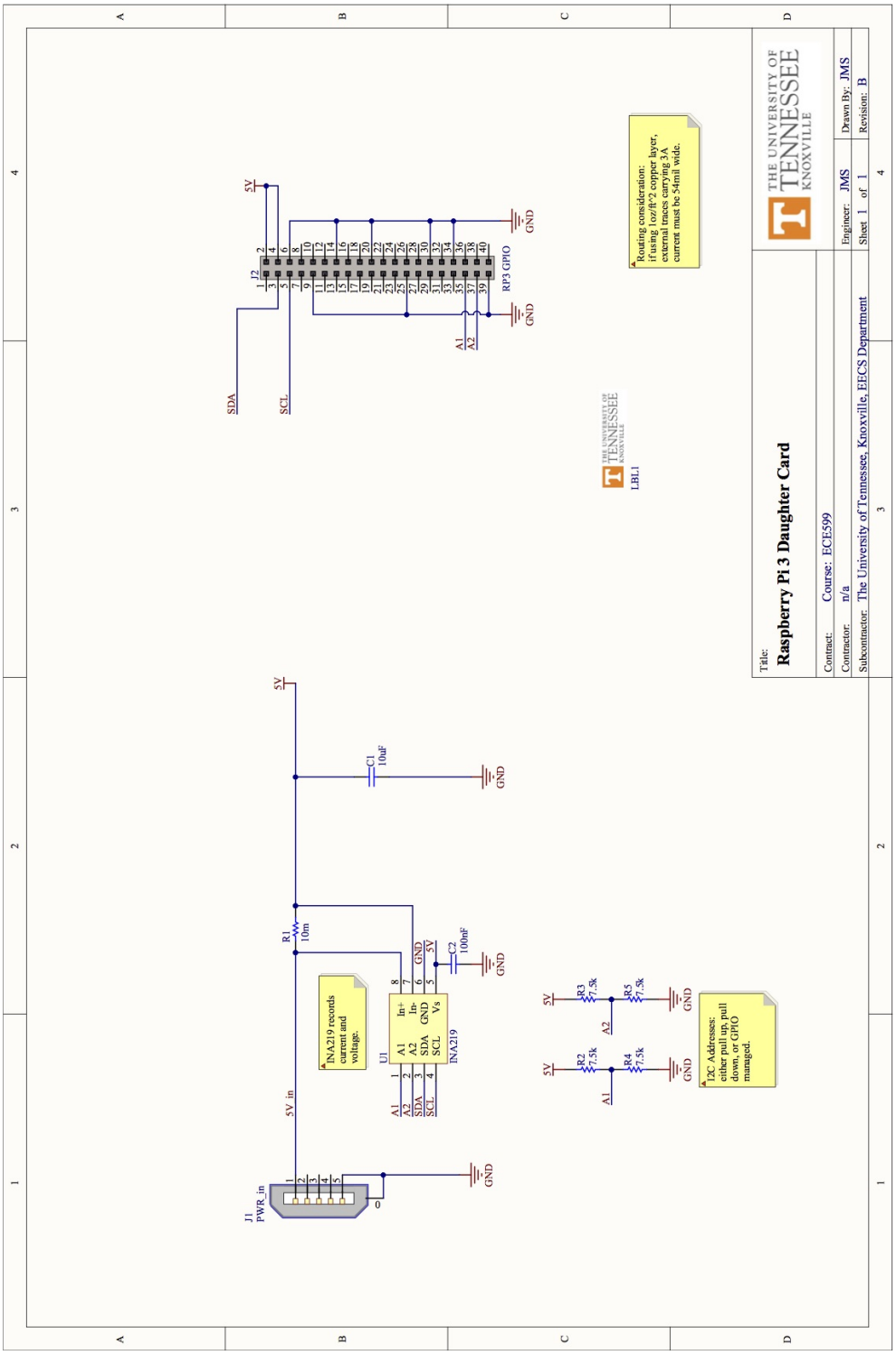

Figure 6: The Daughter Card Schematic Diagram

REFERENCES

Cloutier, Michael, Chad Paradis, and Vincent Weaver. "A Raspberry Pi Cluster Instrumented for Fine-Grained Power Measurement." Electronics 5, no. 4 (September 23, 2016). doi:10.3390/electronics5040061.

Cox, Simon J., James T. Cox, Richard P. Boardman, Steven J. Johnston, Mark Scott, and Neil S. O'Brien. "Iridis-pi: a low-cost, compact demonstration cluster." Cluster Computing 17, no. 2 (June 22, 2013): 349-58. doi:10.1007/s10586-013-0282-7.

Dongarra, Jack, Michael Heroux, and Piotr Luszczek. "HPCG Benchmark." Innovative Computing Laboratory and Sandia National Laboratories. Accessed December 15, 2016. http://www.hpcg-benchmark.org/.

Petitet, A., R. C. Whaley, J. Dongarra, and A. Cleary. "HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers." Accessed December 15, 2016. http://www.netlib.org/benchmark/hpl/.

Red Hat, Inc. "Ansible Is Simple IT Automation." Accessed December 15, 2016. https://www.ansible.com/.

SchedMD. "Slurm Commercial Support and Development." Accessed December 15, 2016. http://www.schedmd.com/.

SchedMD. "Slurm Workload Manager." Accessed December 15, 2016. http://slurm.schedmd.com/.

Simpson, Adam. "Tiny Titan." GitHub. 2013. Accessed December 15, 2016. https://github.com/tinytitan/.

"OSU Micro-Benchmarks." MVAPICH. Accessed December 15, 2016. http://mvapich.cse.ohio-state.edu/benchmarks/.

AUTHOR BIOGRAPHIES

J. PARKER MITCHELL is an undergraduate in Computer Engineering at The University of Tennessee, Knoxville. He holds interests in Computer Architectures, Neuromorphic Computing, High Performance Computing, Embedded Systems, and Robotics. His email address is jmitch47@vols.utk.edu.

AARON R. YOUNG is a Ph.D. student at The University of Tennessee, Knoxville majoring in Computer Engineering. His areas of interest include Computer Architectures, Neuromorphic Computing, High Performance Computing, and Embedded Systems. His email address is ayoung48@vols.utk.edu.

JORDAN SANGID is currently a Masters candidate in Electrical Engineering at The University of Tennessee, Knoxville with a specialization in Wide Bandgap Power Electronics. Areas of interest include audio electronics, circuit theory, and semiconductor device physics. His email address is jsangid@vols.utk.edu.

KELLEY A. DEUSO received a Masters degree in December 2016 in Computer Science from the University of Tennessee, Knoxville. Her research interests include data center design and maintenance, high performance computing, and supercomputing. Her email address is kdeuso@utk.edu.

PATRICIA J. ECKHART is a Masters candidate in Computer Engineering at The University of Tennessee, Knoxville and also hold a Bachelors degrees in Mathematics. Her current research focuses on Neuromorphic Architecures and networks of Neuromorphic devices. Her email address is pdraney@vols.utk.edu.

TAHER NADERI is a Ph.D. candidate in Electrical Engineering at The University of Tennessee, Knoxville. His areas of interest are Control Theory and Power Electronics. His email address is tnaderi@vols.utk.edu.

MARK DEAN is a John Fisher Distinguished Professor at the University of Tennessee. He holds a Ph.D. in Electrical Engineering from Stanford University. His research focus is in advanced computer architecture, data centric computing, and computational sciences. His email address is markdean@utk.edu.