Low-cost Cluster Computing Using Raspberry Pi with Mathematica

Blake Jacobus
bjacobus@millikin.edu

RJ Podeschi
rpodeschi@millikin.edu

Tabor School of Business
Millikin University
Decatur, IL 62522

Abstract

The costs of higher education continue to rise as budgetary and financial pressures strain universities and its students. In particular, students in the sciences rely on computational software like Wolfram Research's Mathematica in their research and studies. The software and its associated syntax are highly useful tools and are necessary skills in the areas of engineering, mathematics, physics, and data science. However, the software is costly and is difficult for colleges and students to afford. With the advent of inexpensive credit card-sized computing devices like the Raspberry Pi and its partnership with Mathematica, the software can now be used at no cost. However, processing and speed are limited and performance is affected on a Raspberry Pi. Through the use of cluster computing, execution times of algorithms using Mathematica can be decreased while maintaining a lower cost than Mathematica's traditional licensing model. This research reports the design and configuration of a Raspberry Pi cluster for use with Mathematica in addition to the results of performance benchmark tests between algorithms executed on one node and four nodes. This work makes an important contribution to both information systems and science disciplines to decrease software licensing costs without sacrificing performance. Conveniently, this research project provided an opportunity for an undergraduate information systems major to learn and understand cluster computing in an experiential learning independent study project.

Keywords: Raspberry Pi, Mathematica, cluster computing, high performance computing, remote kernel processing

1. INTRODUCTION

Small private universities appeal to students across the world for various reasons. Smaller classes, closer relationships, and condensed campuses are among the most common benefits. However, a bantam budget is not likely to make the list of appealing traits for any school. Unfortunately, many institutions are forced to make sacrifices when it comes to funding, and studies centered around the management of information systems and technology suffer greatly from a lack of physical assets. Likewise, the cost of higher education continues to rise as students carry higher debt load upon graduation. While the average annual cost of a four year university in 1985 was $18,910, students are paying at least $37,990 as of 2015, an increase that has trended upwards every single year for the past three decades (U.S. Department of Education, 2016). Fortunately, the advent of the Raspberry Pi computing device has allowed many students across the world to hold a multi-core system in the palm of their hand for far less than the price of a textbook. The Raspberry Pi is affordable and appealing to learners and enthusiasts alike, but their primary drawback is the inability to process complex computations

with any speed, allowing for faster execution times and opportunity for more iterative development.

Among those who depend on powerful processing units are mathematicians, engineers, and various scientists that utilize Mathematica, a "symbolic mathematical computation program" with the ability to process visual and aural data, generate 3D models, and a plethora of other functions used across a variety of fields (Trustees of the California State University, 2017). In the past, Mathematica has been a tool used only by very well-funded entities. The licensing for Mathematica, not unlike the equipment required to leverage its functionality, has a premium price tag. Many institutions with full funding for hardware may still find themselves restricted to licensing only a handful of workstations.

Raspberry Pi brought multicore computing to the eager hands of many financially challenged institutions and the implementation of a free Mathematica license for the Raspbian operating system allowed for the proliferation of computing research on a smaller budget. The Raspberry Pi was, however, unable to accommodate the resource-intensive features of Mathematica, requiring users to search for ways to optimize their small Raspberry Pi for more complex computations. This led to the utilization of Mathematica's remote kernel function, which allows Raspberry Pi devices to pool resources from other Raspberry Pi devices on the same network with a compatible Mathematica license to operate as a cluster. A cluster is defined as a group of similar or identical computer, connected by a computer network that pool resources to provide services or run applications (Burd, 2016). Configuring computers into a cluster can exponentially increase the processing speed. This clustering technique made it possible for students and professionals to distribute workloads across machines to solve complex equations and further expand the applications of the Raspberry Pi computer.

This research project was primarily conducted as an independent research project by a senior information systems major. This paper represents the outcome of a hands-on opportunity to better understand cluster computing and its potential benefits to higher education. This work presents the groundwork to build lab exercises to teach cluster computing in courses related to computer architecture, networking, and infrastructure.

This paper first reviews the Raspberry Pi, its uses, along with background information related to Mathematica. A detailed account of the process of constructing a multi-node Raspberry Pi cluster in conjunction with Mathematica's remote kernel function are provided. Results are shared to demonstrate the performance benefits achieved with Mathematica in a multi-node cluster environment. This work is an important contribution to the field for students and/or education institutions in search of a lower-cost, higher-performance environment for utilizing Mathematica and its computational features.

2. REVIEW OF LITERATURE

Raspberry Pi

In 2009, the Raspberry Pi Foundation was established at Cambridge University by six scientists including its leader, Eben Upton, to develop an ultra-low-cost computing (ULCC) platform to address the lack of interest in programming from grade-school age students (Andrews, 2013; Heeks & Robinson, 2013). The goal for designers was to keep the price of the credit-card sized motherboard to $35 to make it affordable to get in the hands of children in British schools (Harris, 2015). The most recent version of the Raspberry Pi released in 2016, version 3 model B, consists of the following specifications as seen in Table 1 (Raspberry Pi Foundation, n.d.).

Raspberry Pi 3 Model B Specifications
• Quad Core 1.2GHz Broadcom BCM2837 64bit CPU • 1GB RAM • BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board • 40-pin extended GPIO • 4 USB 2 ports • 4 Pole stereo output and composite video port • Full size HDMI • CSI camera port for connecting a Raspberry Pi camera • DSI display port for connecting a Raspberry Pi touchscreen display • Micro SD port for loading your operating system and storing data • Upgraded switched Micro USB power source up to 2.5A

Table 1.

Using system on chip (SOC) technology, advances in integration have effectively enabled manufacturers the ability to shrink an entire desktop onto this handheld motherboard. The

motherboard, with the exception of the Broadcom graphics processor, is entirely open source and meant to run a slimmed down distributions of Debian or Arch-based Linux (Heeks, 2013). While requiring no fans or moving parts, the Raspberry Pi relies on a microSD card for its storage, and can connect to peripherals using USB or HDMI for graphics (Shuurman, 2015).

The Raspberry Pi, due to its low cost, has been successful in reaching underdeveloped countries by providing access to general computing and teaching programming skills to youth (Schuurman, 2015). British school systems now require that all primary school students learn key ideas of computer science and understand computational thinking to accommodate the increased demand for employees with technical acumen (Naughton, 2012). With millions of units purchased for the purpose of democratizing programing and computer science education, the Raspberry Pi's easy-of-use and small footprint has led to the unintended development of several initiatives. Home automation devices for security, digital signage, and a multitude of Internet of Things (IoT) are being tested and incubated through this low-cost platform (Edwards, 2013). As a result, the Raspberry Pi has become a lost-cost computer for teaching, research, and innovation.

Cluster Computing

A modern approach to systems architecture involves distributing computing resources (e.g. CPU and memory) across a set of nodes with identical or similar hardware, and communicate over a specialized network. Known as cluster computing, this methodology replaces previous-generation mainframe and high-cost supercomputers such as MPP (Massively Parallel Processors). Cluster systems are typically configured on commodity hardware, thereby creating cost efficiencies (Gropp, Lusk, & Sterling, 2003). Cluster systems have the advantage of being able to scale up as additional resources are needed as additional nodes can be added to the cluster. Cluster systems generally include six hierarchical layers which include: internetworking, computation, operating systems, compilers, distributing programming models, middleware, and applications (Fung, Li, & Myers, 2005).

Massive multi-processor servers require a large up-front investment while cluster configurations rely on off-the-shelf (or commodity) hardware and can be implemented for a fraction of the cost. Likewise, cluster systems scale well and can grow as resource needs (CPU and memory) grow. As such, the Raspberry Pi is an ideal candidate for architecting a cluster system for this particular project.

Mathematica

Wolfram Research, founded by Stephen Wolfram, developed and released Mathematica in 1988 to solve computational problems through the use of a graphical user interface (GUI) (Wolfram Research, 2017b). Wolfram's flagship product was developed so that the scientific, engineering, mathematical, and computing fields had a software platform to perform a vast array of computations, complete with its own syntax language. The software engine is capable of creating visualizations, performing data analysis, geometric computation or machine learning. Mathematica, as of July 2017, is available through a cloud subscription or a standalone desktop license and are priced at $575 per year or $1,150 per installed machine, which includes upgrades and support (Wolfram Research, 2017d).

The use of Mathematica is prevalent in higher education and can be found in a range of subjects including, but not limited to: computer science, engineering, mathematics, and physics. Previous research has cited its effectiveness in providing interactive simulations in chemical engineering courses (Falconer & Nicodemus, 2014), increased student outcomes in linear algebra (Rahmawati, et. al., 2017), and studying projectile motion in physics (Hutem & Kerdmee, 2013).

Raspberry Pi Meets Mathematica

In 2013, Wolfram Research released their Mathematica software as a free license when installed on a Raspberry Pi running an approved Linux distribution (Wolfram). This allowed anyone with the credit-card sized device access to the same computational platform as researchers in an effort to build the knowledge base in the educational systems. While Mathematica on a Raspberry Pi is functional, it has its limitations due to lower CPU power and available RAM than a typical desktop. To be able to perform computations with better response times, multiple Raspberry Pis are needed through Mathematica's clustering feature. Through the combination of Mathematica's feature-set and Raspberry Pis open and low-cost platform, the multi-note clusters can be built to increase processing and response times.

3. METHODOLOGY

Hardware Configuration

The Raspberry Pi cluster consists of four Raspberry Pi Model B devices configured on a vertical enclosure. The enclosure allows for proper cooling, secure transportation, and proper cable management (see Figure 1). The total cost of the four-note cluster procured through Amazon.com, including peripherals, cables, and display, was less than $500. The parts used in the four-node cluster are outlined in Table 2.

QTY	Item
4	Raspberry Pi 2 Model B V1.1 devices
1	GeauxRobot Raspberry Pi 4-layer Dog Bone Stack Enclosure
4	16GB microSD cards (FAT format)
1	microSD/SD Adapter
4	RPi AC/microUSB power adapters
1	6-outlet electrical strip
1	Encore 8 Port NWay 10/100 switch
5	CAT5e network cables
1	HDMI Cable
1	Monitor or television with HDMI port
1	Computer mouse
1	Computer keyboard

Table 2. Cluster Parts List.

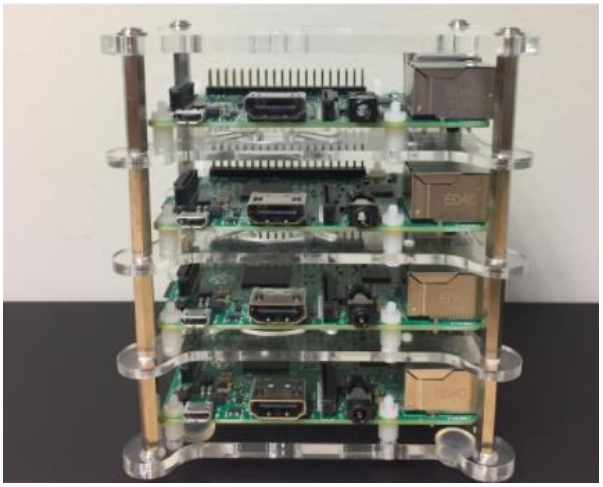

Figure 1. Raspberry Pi Enclosure

A Raspberry Pi device stores secondary data on one microSD card. The Raspberry Pi Foundation recommends using an 8GB SD card, but 16GB cards were used for this cluster to ensure ample storage space (Raspberry Pi Foundation, n.d.d). These cards must be formatted using FAT file system and an image of the latest Raspbian image must be installed on the SD cards. A table containing names, functions, and web sources for all software tools used to create this Raspberry Pi cluster is located in Table 7, found in Appendix A. The process for formatting and mounting the cards is as follows:

1. Format all four microSD cards using SDFormatter application
2. Download Raspbian image from Raspberry Pi Foundation
3. Write image to a single microSD card (known hereafter as RPi01) using Win32 Disk Imager
4. Configure Raspbian OS settings using `raspi-config` command (detailed in Software Configuration section below)
5. Read image from RPi01 using Win32 Disk Imager and write to Windows machine
6. Write image RPi01 to the three other microSD cards, known hereafter as RPi02, RPi03, and RPi04
7. Login to RPi02, RPi03, and RPi04 using PuTTY interface from remote workstation and use `raspi-config` command to change the machine names and passwords according to Table 3. All other configuration settings remain the same.

Software Configuration

8 Update	Wait for updates to finish
1 Change User Password	RPi01*, RPi02*, RPi03*, and RPi04*
2 Hostname	RPi01-04
3 Boot Options	B1 Desktop / CLI B4 Desktop Autologin Desktop GUI
5 Interfacing Options	Enable P2 SSH
6 Overclock	High*
A1 Expand Filesystem	
A3 Memory Split	16**
* Overclocking the CPU yielded minimal increases in processing speed. ** Memory Split allocates more or less RAM to GUI processing, allowing for more performance when processing equations and less graphical overhead.	

Table 3. Raspi-Config Settings.

Raspbian's operating system settings are configured using the raspi-config interface, which can be accessed by entering the `raspi-config`

command into the Raspbian terminal. The following settings in Table 3 must be adjusted to ensure that the individual devices are optimized for clustering:

Network Configuration
After securing each Raspberry Pi device to the enclosure and developing a system for powering each node using the USB cables, the Raspberry Pi devices must be connected to a central switch for remote management and remote kernel communication. Each of the four devices pictured in Figure 1 have an Ethernet port. The cluster network is constructed by attaching a CAT5e network cable to each network port and attaching the other end of each cable to ports 1-4 of the network switch. A fifth CAT5e cable is inserted into port 5 of the network switch, and the opposite end of that cable is inserted into a network router that facilitates the communication between all connected devices and the Internet.

Passwordless SSH Configuration
In order for Mathematica to issue commands across all four Raspberry Pi devices, SSH must be utilized. This requires the sharing of public SSH keys between all devices (Raspberry Pi Foundation, n.d.a). Public key sharing is accomplished by following these steps:

1. Create .ssh directory on each Raspberry Pi device
2. Generate SSH keys on each Raspberry Pi device
3. Copy the public key from each device to every other device's .ssh directory

All devices will be able to login to each other using SSH without entering a password due to the public key sharing. Commands from the Raspbian terminal and the Wolfram terminal (or Mathematica GUI) can be submitted without entering any passwords.

Mathematica Remote Kernel Configuration
Mathematica processes evaluations using a local kernel, a notebook, and an evaluation (Wolfram Research, 2017c). The local kernel represents the processor(s) present on the local machine. A notebook is used as a value processing interface for Mathematica evaluations, which are strings processed by the kernel. The purpose of building the cluster was to simulate a supercomputer, which relies on the computation of variables in tandem for increased efficiency through load balancing, otherwise known as parallel processing. Mathematica allows for the configuration of remote kernels, which can be used to perform parallel evaluations. Remote kernels must be able to communicate within a common network. Parallel evaluations take a single evaluation and delegate the process of evaluating the string to all processing units (or remote kernels) indicated in the evaluation. Splitting the processing task among multiple processors, or kernels, increases the efficiency and speed of the evaluation. The process for configuring remote kernels in Raspbian distribution of Mathematica is as follows:

Step	Command/Process
1	From a Mathematica notebook: `FrontEndTokenExecute ["PreferencesDialog"]`
2	Select 'Parallel' tab
3	Select 'Remote Kernels' under 'Parallel Kernel Configuration'
4	Select 'Add Host'
5	`Hostname`: <ip address of RPi>
6	`LaunchRemote`: default commands
7	Change 'Kernels' value to `4`
8	Select 'Enable'

Table 4. Remote Kernel Configuration Process

After completing these steps, Mathematica can then be directed to start the remote kernels and issue evaluation tasks using specific Parallel commands.

Mathematica Parallel Computing
Once the remote kernels are configured, evaluating commands in parallel are relatively simple through the use of the `ParallelCombine` command in the Wolfram Language (Wolfram Research, 2017e). This command evaluates a normal expression by distributing parts of the computation to every available remote kernel. The kernels process the computation in tandem and return the results as a single solution. The `ParallelCombine` command is added after the `AbsoluteTiming` command. The following functions were evaluated on the Local Kernel to establish a control by which the parallel evaluations could be compared:

Extract prime numbers from data set of 1 to 1000 (Local Prime)
```
AbsoluteTiming[Prime[Range[1000]]]
```

Extract prime numbers from data set of 1 to 1000 (Parallel Prime)
```
AbsoluteTiming[ParallelCombine[Prime[Range[1000]]]]
```

4. RESULTS

Each evaluation listed above was performed 10 times. Every time an evaluation was performed, the Raspberry Pi kernels were completely reset by closing the Mathematica application and restarting it. This was done to clear the internal system caches of stored results. Mathematica contains a feature that indexes evaluations temporarily so that repeated evaluations can be resolved faster (Wolfram Research, 2017e). However, this feature skewed the data originally collected for the Raspberry Pi cluster. While the first evaluation was a true test of processing speed, the subsequent evaluations all yielded equally fast resolution times, which were faster than the original evaluation. One proposed solution was to use the `ClearSystemCache[]` command to wipe the evaluations indexed in Mathematica, but this did not appear to serve its purpose (Wolfram Research, 2017a). Therefore the most efficient, though undoubtedly rudimentary, technique was to close Mathematica and restart it, forcing all kernels to close their connections and restart along with the graphical interface. The data collected while using this technique are located in Table 5.

Local Prime	Parallel Prime
29.48	8.85
29.31	8.83
29.7	8.79
29.65	8.79
28.98	8.79
29.28	8.87
29.17	9.05
29.51	8.94
29.2	9.02
29.09	8.86

Table 5. Local and Parallel Evaluations

From these data sets, it can be concluded that parallel kernel evaluations are, on average, completed 70.42% faster than local kernel evaluations according to the `AbsoluteTiming` function of Mathematica.

In addition, ten parallel evaluations were performed on the Raspberry Pi cluster when the processors were overclocked (OC) and when the Memory Split function (MS) was adjusted to allocate more memory to graphical processing. The average evaluations are located in Table 6.

From these data sets, it was discovered that overclocking does increase processing speed, but only by 12.53%. Adjusting the Memory Split setting from a 16 MB allocation to 256 MB yielded a decrease in processing speed of 3.53%. However, it's interesting to note that Memory Split did not affect processing speed dramatically. Allocating additional memory to the GPU via the Memory Split settings will decrease the graphical processing overhead generated by Mathematica's interface. This may lead to a more optimized system if the CPU efficiency continues to be affected by a factor of less than 5% (Raspberry Pi Foundation, n.d.c).

Evaluation	Memory Split/ Overclock Setting	Time
Local	16/None	29.69
Parallel	16/None	8.78
Parallel w/OC	16/High	7.68
Parallel w/MS	256/None	9.09

Table 6.

5. DISCUSSION AND CONCLUSIONS

A key limitation to the development of the Raspberry Pi cluster was the gap in documentation between desktop versions of Mathematica and the Raspberry Pi distribution. The Mathematica user interface allowed for all of the function creation and evaluation features required to test the cluster, but both formal and informal sources (e.g. forums) lacked a standard direction for configuring remote kernels on Raspberry Pi devices. The Preferences Dialogue Menu, pictured in Figure 2 in Appendix A, is not accessible unless the user evaluates the following command in Mathematica: `FrontEndTokenExecute["PreferencesDialog"]`

Prior to discovering this command, the process of establishing remote kernel connections in Mathematica consisted of canvassing outdated Mathematica and Raspberry Pi forums for lengthy functions that attempted to activate remote kernels using commands in both the Wolfram CLI and Mathematica. One example of a remote kernel activation function that managed to generate an error message, an optimistic sign at the time, can be referenced in Appendix A, Figure 3 (Quantum, 2016).

A perpetual error message appeared following every attempt to evaluate the remote kernels afterwards:
"The kernel failed to connect to the front end. (Error = MLECONNECT). You should try running the kernel connection outside front end."

It was only by piecing together advice from various forums related to the Desktop version of Mathematica that the Preferences dialog menu became accessible.

Ultimately, this proof-of-concept was effective in building a working Raspberry Pi cluster to demonstrate the differences in processing times between a single node and multiple nodes. Effectively, the cluster could save an individual over $1,100 in software licensing costs.

There is a trait valued universally across various industries: the ability to do more with less. At the very least, this project demonstrates to the user that parallel processing can be used to leverage subpar resources to create something more valuable. This idea, and the proliferation of its practice, is vastly beneficial to the modern student. In a world where throwing something away is more commonplace than working to fix or improve it, the act of coupling processors together is a prime example of resourcefulness and ingenuity. Outside of the technical enhancements students may reap from this model of Mathematica processing, students may also find themselves extending these practices to their studies and eventual jobs. It's not difficult to write a business case for why spending should be allocated to a new resource, but it is uniquely valuable to have an employee that understands how to leverage depreciated resources to match current demand. That is one of the primary values behind this Raspberry Pi model.

An unintended consequence of this research project is the value this cluster has in I.S. education to teach and demonstrate how cluster computing can be applied. Previous work by Doucet and Zhang outline the benefits of learning cluster computing through the use of Raspberry Pis. Learning outcomes include a hands-on experiential learning opportunity along with learning a better understanding of how cluster computing could be used to solve computational problems (2017). Their research, along with this work, could be used to build a series of labs for student learning in a networking or I.T. infrastructure course to further connect theory to practice.

It is conceivable that the configuration of a Raspberry Pi/Mathematica cluster could be more automated to lower the overhead for institutions and students alike. Alternatively, clusters could be managed by the institution or IS/IT students for remote access by users. The technology allows for the use of Mathematica at a lower cost than purchasing a standalone license, which can ease the financial burden from departments or students. More research is needed to test other features of Mathematica including advanced algorithms, visualizations, and data analysis. In addition, benchmark tests need ran comparing processing times between the cluster and various models of typical workstations. Ideally, the next study would place the Raspberry Pi cluster in the hands of instructors and students to mimic applicable workloads.

6. REFERENCES

Andrews, C. (2013). Easy as Pi. *Engineering and Technology*, 8(3), 34-37. doi:10.1049/et.2013.0302

Burd, S. (2015). Systems Architecture 7e. Boston, MA: Cengage Learning

Doucet, K., Zhang, J. (2017). Learning Cluster Computing by Creating a Raspberry Pi Cluster. *Proceedings of the SouthEast Conference on - ACM SE 17,* 191-194. doi:10.1145/3077286.3077324

Edwards, C. (2013). Not-so-humble Raspberry Pi Gets Big Ideas. *Engineering and Technology*, 8(3), 30-33. doi:10.1049/et.2013.0301

Falconer, J., Nicodemus, G. (2014). Interactive Mathematica Simulations. *Chemical Engineering Education*, 48(3). Pp. 165-174

Fung, C., Li, J., & Myers, D. (2005). Evaluation of a Small Scale Cluster Computing System for Parallel Intelligent Technique Applications. *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, 388-393.

Gropp, W., Lusk, E., & Sterling, L. (2003). *Beowulf cluster computing with Linux.* Cambridge, MA: MIT Press.

Harris, S. (2015). Faster Raspberry Pi brings low-price computing power to education. *Engineer*. (Online Edition), 1.

Heeks, R., Robinson, A. (2013). Emerging Markets Ultra-Low-Cost Computing and Developing Countries. *Communications of the ACM.* 56(8), pp. 22-24.

Hutem, A., Kerdmee, S. (2013). Physics Learning Achievement Study: Projectile, using Mathematica program of Faculty of Science and Technology Phetchabun Rajabhat

University Students. *European Journal of Physics Education*, 4(3), pp. 22-33.

Naughton, J. (2012, March 31). Why all our kids should be taught how to code. *The Guardian*.

Quantum, The Physicist. *Remote Kernel through SSH*. (2016, February 3). Retrieved from https://mathematica.stackexchange.com/questions/65953/remote-kernel-through-ssh

Rahmawati, N.D., Nugroho, A.A., & Harun, L. (2017). *Proceedings of the International Conference on Mathematics: Education, Theory, and Application*. 1(1), pp. 157-164.

Raspberry Pi Foundation (n.d.a). *Passwordless SSH Access.* Retrieved from https://www.raspberrypi.org/documentation/remote-access/ssh/passwordless.md

Raspberry Pi Foundation (n.d.b). *Raspberry Pi 3 Model B*. Retrieved from https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Raspberry Pi Foundation (n.d.c). *Raspi-Config*. Retrieved from https://www.raspberrypi.org/documentation/configuration/raspi-config.md

Raspberry Pi Foundation (n.d.d). *SD Card*. Retrieved from https://www.raspberrypi.org/learning/hardware-guide/components/noobs-card/

Schuurman, D. (2015). Introducing Open Source and the Raspberry Pi to Schools in Developing Nations. *Perspectives on Science and Christian Faith*. 6(1). pp. 50-53.

Trustees of the California State University (2017). *What is Mathematica?* Retrieved from http://www.calstatela.edu/its/services/software/mathematica.php

Wolfram Research (2017a). *ClearSystemCache*. Retrieved from http://reference.wolfram.com/language/ref/ClearSystemCache.html.en

Wolfram Research (2017b). *Company Information.* Retrieved from https://www.wolfram.com/company/

Wolfram Research (2017c). *Documentation Center*. Retrieved from http://reference.wolfram.com/language/

Wolfram Research (2017d). *Mathematica Pricing*. Retrieved from https://www.wolfram.com/mathematica/pricing/colleges-universities-individuals.php

Wolfram Research (2017e). *ParallelCombine*. Retrieved from http://reference.wolfram.com/language/ref/ParallelCombine.html

Wolfram, Stephen. (2013). *Putting the Wolfram Language (and Mathematica) on Every Raspberry Pi*. Retrieved from http://blog.stephenwolfram.com/2013/11/putting-the-wolfram-language-and-mathematica-on-every-raspberry-pi/

U.S. Department of Education, National Center for Education Statistics. (2016). Digest of Education Statistics, 2015 (NCES 2016-014),Chapter 3. https://nces.ed.gov/fastfacts/display.asp?id=76

Appendix A

Resource/Tool		Function
7-Zip	http://www.7-zip.org/download.html	A file archiver used to unzip Raspbian image file for distribution to microSD cards.
PuTTY	https://www.putty.org	A SSH and Telnet client that allows the user to issue commands simultaneously to Raspberry Pi devices from a Windows workstation.
Raspbian image	https://www.raspberrypi.org/downloads/raspbian/	Operating system for Raspberry Pi
SDFormatter	https://www.sdcard.org/downloads/formatter_4/eula_windows/index.html	Tool used to format the microSD card used by the Raspberry Pi for secondary storage of Raspbian operating system and all other system files
Win32 Disk Imager	https://sourceforge.net/projects/win32diskimager/	Allows the user to read and write Raspbian images to and from microSD cards for uniform image provisioning throughout cluster

Table 7. Raspberry Pi Cluster Software Tools

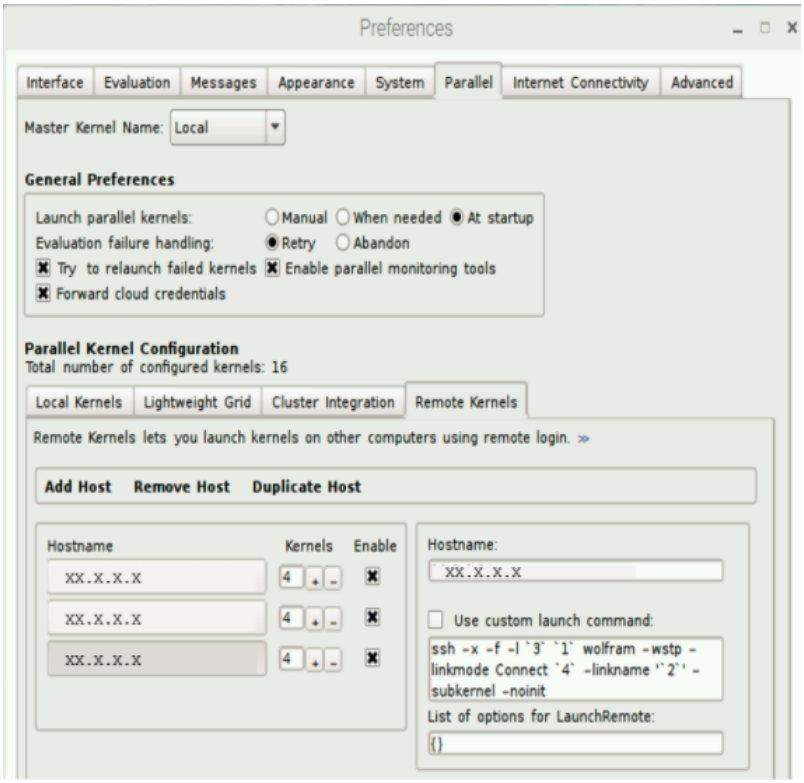

Figure 2. Mathematica Preferences Dialog Menu

```
Needs["SubKernels`RemoteKernels`"]
Parallel`Settings`$MathLinkTimeout = 100;
       user = "pi";
password = "RPi02*";
ssh = "export LD_LIBRARY_PATH=;ssh";
math = "MathKernel" <> " -wstp -linkmode Connect `4` linkname `2` -subkernel -noinit
>& \ /dev/null &";
number = 4;
machine = "XX.X.X.X";
remote = SubKernels`RemoteKernels`RemoteMachine[machine, ssh <> " " <> user <> "@" <>
machine <> " \"" <> math <> "\"", number]

Print[remote // InputForm]
kerns = LaunchKernels[remote]

ParallelEvaluate[$MachineName]
(*CloseKernels[]*)
```

Figure 3. Sample Function to Activate Remote Kernels