

**University of Arkansas – CSCE Department**

**Capstone II – Preliminary Report – Spring 2014**

# Affordable Raspberry Pi Cluster

**Brenna Blackwell, Nicolas Edward, Joshua Ross**

## Abstract

Building a Raspberry Pi cluster computer can solve the problem of affordable performance based computing by combining multiple Pis, along with other peripherals, to create a self-contained computing system that demonstrates performance comparable to market laptops available. Additionally, the system would provide a portable, cost-effective teaching tool focused on distributed computing.

## 1.0 Problem

While there are products on the market geared towards affordable, ubiquitous computing, the market is sparse. The products available require the user to have a basic functional knowledge in order to create more powerful tools from these parts. Affordable distributed computing systems are even more sparse.

## 2.0 Objective

We are building a computer system using inexpensive parts, such as Raspberry Pis and other peripherals, in order to create a more powerful computing and teaching tool.

## 3.0 Background

### 3.1 Key Concepts

Networking - The four Pis are going to have to work together, and we intend to build the network using an Ethernet router. The individual Pis will be running their own instance of Raspian, the Debian-based Linux operating system optimized for the Raspberry Pi.

Concurrent/Parallel Computing - Because we want to demonstrate the parallel nature of the completed Raspberry Pi cluster, we intend to develop a problem for the cluster to solve concurrently. A message passing interface will be used in order to distribute the work, and an MPI will coordinate the activities for every Pi node in the cluster. The de facto standard approach for MPIs uses TCP/IP and socket connections.

Node Failure Management - The system must remain operational in the case of node failure. To keep the system operational, a node may be shut down or disallowed access to shared resources.

### 3.2 Related Work

Joshua Kiepert, a Ph. D. student at Boise State University’s Electrical and Computer Engineering Department, designed and built a Beowulf cluster consisting of Raspberry Pis in order to further his research. He published the method that he used, comparisons with other clusters computers, and the results of testing [1].

Additionally, Simon Cox, Professor of Computational Methods at Southampton University published his design on how to make a Raspberry Pi cluster, which can be found at the University’s website [2].

## 4.0 Design

### 4.1 Design Goals

The design goal is to build a more powerful computer using inexpensive, widely available components, such as Raspberry Pis, than can be achieved with those components individually.

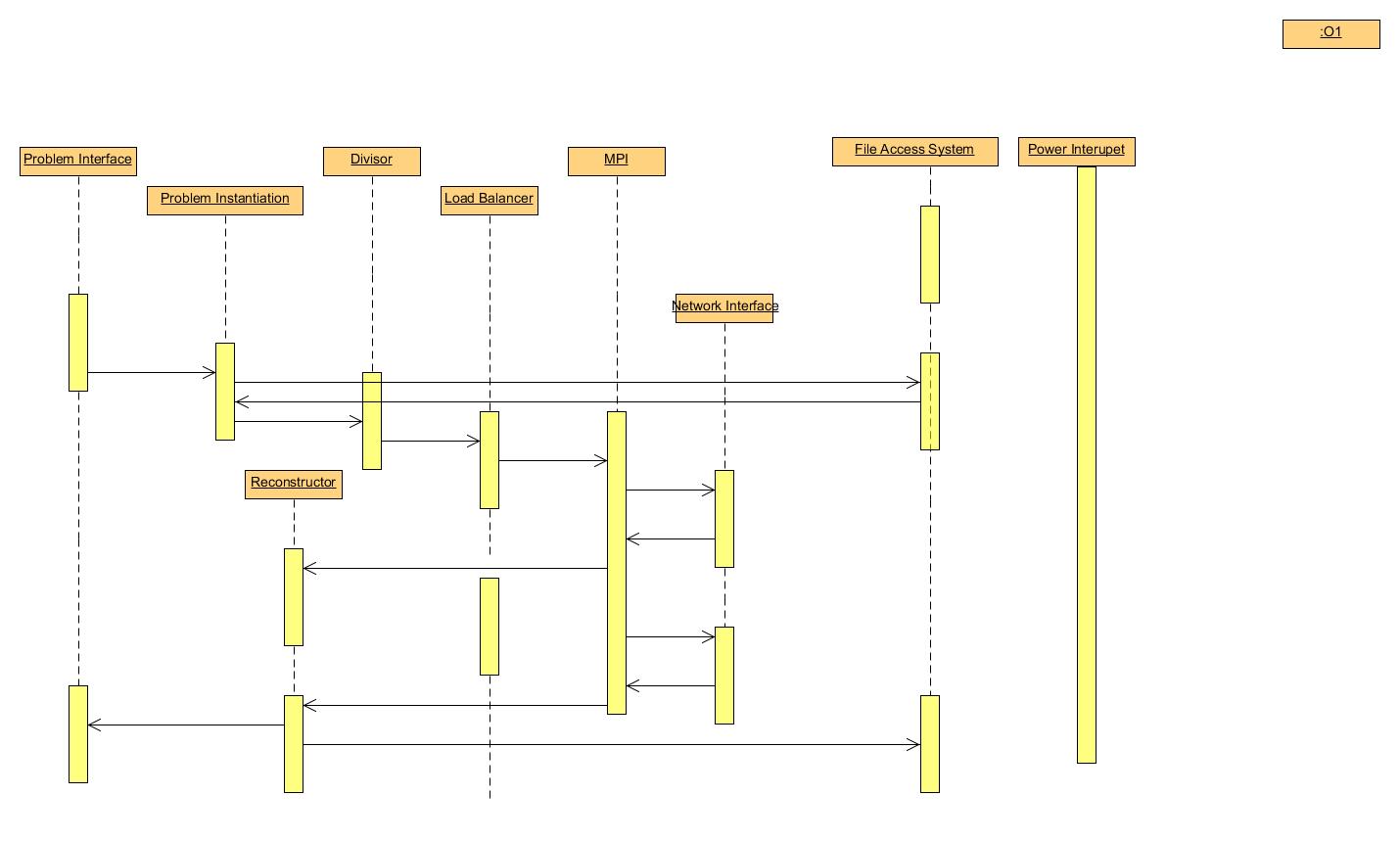
### 4.2 High Level Architecture



#### Hardware

The hardware architecture consists of one Pi acting as the head node, with three leaf node Pis connected to it via a switch. Power will be routed to each Pi using independent 5V power supplies in order to avoid relying on a single unit. Use of the GPIO pins may be done to integrate some data passage between Pis. Additionally, we have included a 7” 1280x800 (720p) display, powered by its own 9V power supply, can be connected to the head node via HDMI cable. While any HDMI display could be connected, this small, lightweight option enhanced the portability aspect of the project’s objective.

One additional hardware option we are considering is a separate USB port and hard drive in order to store the files which the Pis will need to complete their assigned tasks. The reason for this consideration is that if we move file access to USB communications, the Ethernet network connected via switch will be less congested with sending data. We hypothesize that this will free up the network, and speed up instruction communication.



#### Software:

The host Pi will be in charge of data and instruction communication. MPICH2 [3] has been installed on all Pis in order to facilitate intra-cluster communication. MPI4Py [4], a library of MPI bindings for the Python programming language has also been installed on each Pi in the cluster. Because the problem program will be one of image processing, Pillow, an open source fork of the Python Imaging Library, [5] has also been installed on all the Pis.

As we have not yet started tests on the USB hard drive connection described above, the process of distributing files across all the of the nodes is done by way of shell scripts we’ve written. These scripts distribute the files via the Ethernet network. The remaining software for the cluster is the problem programs we are writing for it described below.

#### Problem Program

After researching the Raspberry Pi, we decided to use Python in order to write our program. We are looking into image processing. Two initial examples we are working on are i) to send a whole image to each leaf Pi, with the individual leaves performing a unique operation on the image, so that when it returns to the head node, three different operations have been done to the image, and ii) divide a single image into three parts, sending a single part to each Pi in order for the same operation to be applied, at which point the parts are sent back to the head node and a single image is reconstructed. Currently, these applications are being developed by one team member, which a second member studies distributing tasks to the Pis via our chosen message passing interface.

### 4.3 Risks

|  |  |
| --- | --- |
| **Risk** | **Risk Reduction** |
| Overloaded Power Supply | Distributing power supply by using individual supplies instead of an entire-unit supply. |
| Overheating | Install a small fan unit if necessary. |
| Incorrect GPU Computations | Use unit testing and incremental development. |
| Idle CPU/GPU | Use a monitor program to load balance the Pis. |

### 4.4 Tasks

**Understand/gain background:** Read Raspberry Pi documentation. DONE

**Design:** Design system layout in order to integrate the four Pis and other necessary peripherals. Software design software flow to allow load balancing and data coherency among Pis when while the cluster is running the problem program. DONE

**Implementation:** Hook up the four Pis and peripherals according to designed specifications. DONE

**Test:** We are planning on testing each step of the implementation as we progress, so that when we move forward to the next step, the previous step has been thoroughly tested. Our major divisions in testing will occur when we go from coding for a single unit and move to concurrent computing in all units. IN PROGRESS

**Demonstrate:** Run the problem program on the cluster, and display the results. IN PROGRESS

**Document:** Like testing, we plan on writing documentation for the cluster as we are working on a specific task, so that by the time the task is complete, its documentation is written. IN PROGRESS

### 4.5 Schedule

|  |  |  |
| --- | --- | --- |
|  | **Fall** | **Spring** |
| 1. Understanding |  |  |
| 2. Design |  |  |
| 3. Implement |  |  |
| 4. Test |  |  |
| 5. Demonstrate |  |  |
| 6. Document |  |  |

### 4.6 Deliverables

* Design Document: Upon completion of the project, we will submit a final design document outlining the cluster.
* Code: Any code that we write to test or run on the cluster will be provided.
* Final Report: A final report will be written discussing the process of designing and implementing the cluster.

## 5.0 Key Personnel

**Brenna Blackwell:** Blackwell is a senior Computer Engineering major in the Computer Science and Computer Engineering Department of the University of Arkansas. She has completed or is enrolled in Computer Organization and Embedded Systems. She is currently an iOS developer for the Biological and Agricultural Engineering Department at the University, and she will be jointly responsible with Nicolas for the design and implementation of the hardware aspects of the Raspberry Pi cluster.

**Nicolas Edwards:** Edwards is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses in Embedded Systems, Computer Architecture, and Open Source Hardware. Edwards has gained experience in this area by using Arduino to detect door knocks and broadcast image from integrated camera to homeowners, as well as Light pulse detection was implemented as well to provide novel unlocking of door. He will be working on hardware mounting, GPU problem programming in OpenCL, and push button startup circuit.

**Joshua Ross:** Ross is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas.  He has completed courses in Computer Organization, Algorithms, and Operating Systems.  Ross interned at FIS at their Little Rock, AR campus under a Programming Analyst where he developed a productivity-enhancing file cross-checker for the technical reviewers.  He is responsible for implementation of the main software components of the cluster.

## 6.0 References

[1] Creating a Raspberry Pi Beowulf Cluster, http://coen.boisestate.edu/ece/files/2013/05/Creating.a.Raspberry.Pi-Based.Beowulf.Cluster\_v2.pdf

[2] Raspberry Pi at Southampton, http://www.southampton.ac.uk/~sjc/raspberrypi/

[3] MPICH, http://www.mpich.org/

[4] MPI4Py, http://mpi4py.scipy.org/

[5] Pillow, http://pillow.readthedocs.org/en/latest/