

**University of Arkansas – CSCE Department**

**Capstone I – Final Proposal– Fall 2013**

**Raspberry Bramble**

**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.

1. **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.

1. **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 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 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].

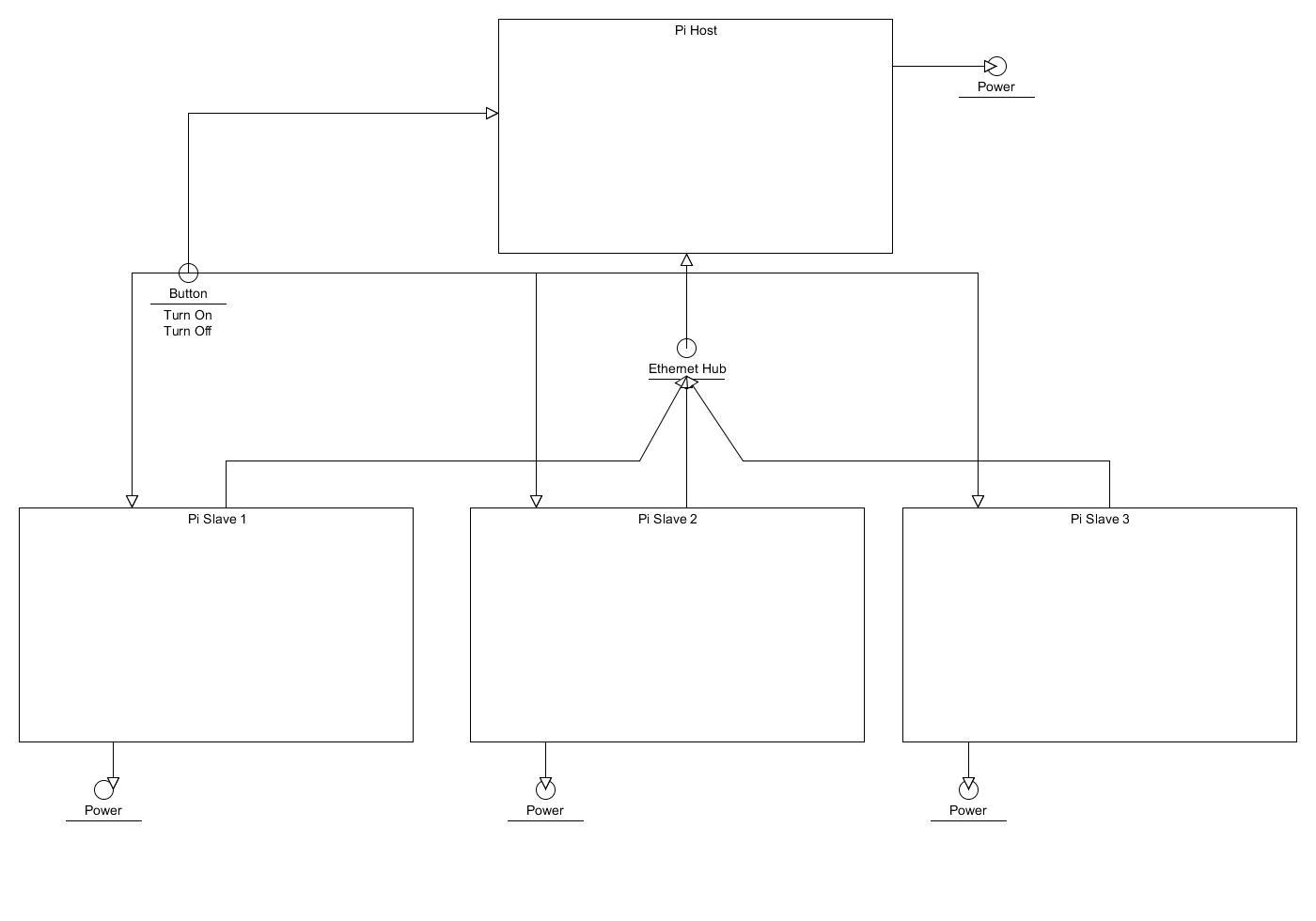
Additionally, Simon Cox, Professor of Computational Methods at Southhampton University published his design on how to make a Raspberry Pi cluster, which can be found at the University’s website [2].

**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.

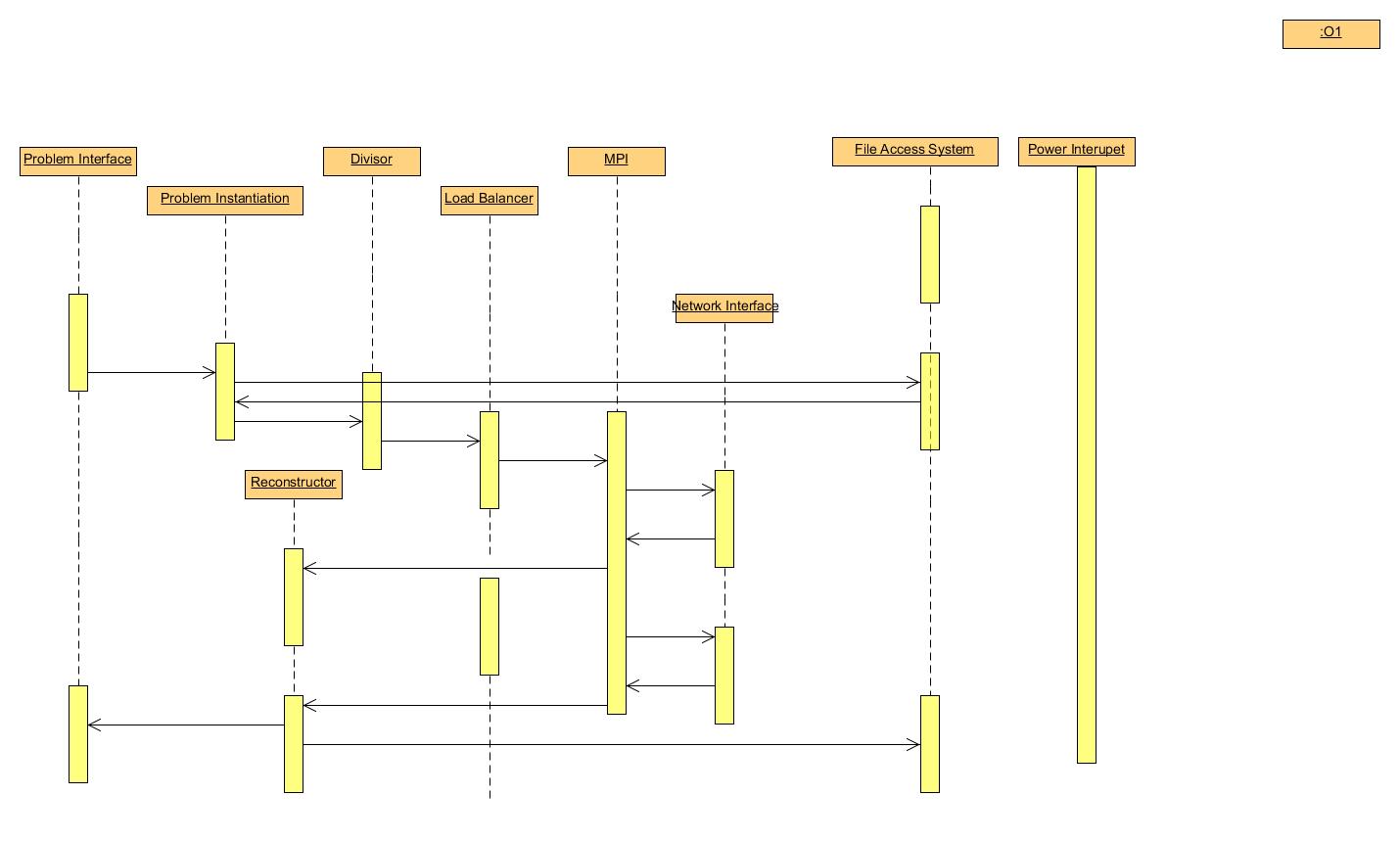
**4.2 High Level Architecture**



Hardware:

The hardware architecture consists of one Pi acting as a host to the three Pis connected to it. The connection will be done through a standard Ethernet router. Power will be routed to each Pi using independent power supplies in order to avoid relying on a single unit. A button will be routed to each Pi to feed a power on signal in parallel to allow for easy start up. Use of the GPIO pins may be done to integrate some data passage between Pis.

In order to make the process of turning the entire cluster on at once, as opposed to powering up/down the individual Pis one by one, we have looked into method applicable to the Pis. One would think, with the plethora of GPIO pins, that this task would be simple, barring that, one would think Wake on LAN would be available. However, neither of these options are available on the Pi. As such, we are intending to design a button that disconnects the 5 volts as a 1 to 4 parallel disconnection. This will give the appearance of pulling or reinserting the plug.



Software:

The host Pi will be in charge of data and instruction communication. Some form of MPI will be needed to allow them to talk to each other in a concurrent environment. The host will set up a problem and then contact each Pi with their instructions and data. As each Pi runs low on data or instructions it will use the MPI to request more data/instructions. Once the Pis receive there initial instructions they will have to go through a set up routine to format the data and instructions. As results become available they will be passed back to the Host Pi.

While each Pi is working, the host computer will monitor the Pis’ resources and load balance as needed.

For the message passing interface, we are specifically looking into two options, OpenMPI and MPICH. OpenMPI is a message passing interface written by the MPI Forum [3]. The OpenMPI documentation describes their software as an API used for parallel or distributed computing systems. MPICH is a “high performance, widely portable implementation” [4] software available. Both software options are open source with freely available licenses, and have been tested on the Linux platform. In the course of research in this area, it was found that Boston University has published a comparison of various message passing interface softwares [5], which we are studying to determine which would work best for our purposes.

Problem Program:

During initial research, it was discovered that the Broadcom VideoCore IV was designed for embedded home application use, such as for a DVD player, flat panel television, etc. Additional, BroadCom has not designed OpenCL APIs compatible with the chip. Many individuals or enthusiast groups have attempted to create an OpenCL API, but none have completed this task, for various reasons, such as the closed source intellectual property rights of Broadcom’s chip. As a result, our initial plan of writing a program to access the GPU and solve a graphics based problem is no longer possible. Thus, we are currently researching parallel/concurrent computing problems capable of being solved on a 4 Pi cluster.

**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. |
| 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.

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.

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

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.

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

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.

**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 Southhampton, <http://www.southampton.ac.uk/~sjc/raspberrypi/>

[3] OpenMPI, <http://www.open-mpi.org/>

[4] MPICH, <http://www.mpich.org/about/overview/>

[5] Boston University, How to Select MPI Implementation and Compilers, http://www.bu.edu/tech/about/research/computation/katana-cluster/mpi-implementations/