ARM Cluster

Senior Design Final Documentation

Discoverment

Andrew Kenneth Hoover   Christine Norma Sorensen

April 7, 2016

Contents

List of Figures

Overview Statements

0.1 Mission Statement

To create the fastest and most cost effcient cluster of single-board computers.

0.2 Elevator Pitch

Our goal is to build a cluster of single-board computers that produces as many floating-point operations possible per watt per dollar. We are testing three computers: ODROID, Raspberry Pi, and PcDuino and finding alternative modes of communication outside of the traditional ethernet port in order to find what best fits our cluster.

Document Preparation and Updates

Current Version [3.2.0]

Prepared By:
Andrew Hoover
Christine Sorensen

Revision History

Date	***Author***	***Version***	***Comments***
09/14/15	*Christine Sorensen*	*1.0.0*	*Initial version*
10/02/15	*Christine Sorensen*	*1.1.0*	*Completion of Sprint # 1*
11/09/15	*Christine Sorensen*	*2.0.0*	*Updated Project Overview*
11/09/15	*Christine Sorensen*	*2.1.0*	*Completion of Sprint # 2*
12/09/15	*Christine Sorensen*	*3.0.0*	*Completion of Sprint # 3*
12/10/15	*Andrew Hoover*	*3.1.0*	*Design and implementation and Unit Testing documentation*
12/11/15	*Christine Sorensen*	*3.2.0*	*Added to log entries. Added resumes.*
1/21/16	*Christine Sorensen*	*4.0.0*	*Moved to new template. Cleaned up sections.*

1

Overview and concept of operations

1.1 Team Members and Team Name

1.2 Client

1.3 Project

1.3.1 Purpose of the System

1.4 Business Need

1.5 Deliverables

- ARM Cluster
- Research Symposium
- Design Fair
- Documentation

1.6 System Description

1.6.1 Major System Component #1

1.6.2 Major System Component #2

1.6.3 Major System Component #3

1.7 Systems Goals

1.8 System Overview and Diagram

1.9 Technologies Overview

2

User Stories, Requirements, and Product Backlog

2.1 Overview

2.2 User Stories

2.2.1 User Story #1

As a user, I want a cluster of at least 6 and no more than 12 single-board computers.

2.2.1.a User Story #1 Breakdown

The cluster will be made of ODROIDs, PcDuinos, or Raspberry Pi's, depending on which performs best in the benchmark tests.

2.2.2 User Story #2

As a user, I want the fasest, most efficient in both cost and operation cluster.

2.2.2.a User Story #2 Breakdown

Testing will be done on the single-board computers compared with prices to determine which will be best for the ARM cluster.

2.2.3 User Story #3

I want to the cluster to be at or below the maximum budget of $1,200.00.

2.2.3.a User Story #3 Breakdown

The budget must include all components of the cluster: the computer boards, cost of power, switch, memory, cables, and power strips.

2.2.4 User Story #4

I want to know which of the single-board computers is the fastest in GFlops/$/Watt.

2.2.4.a User Story #4 Breakdown

Testing will take place on the ODROID, PcDuino, and Raspberry Pi to determine which is the fastest in this metric.

2.2.5 User Story #5

I want a different communication mode beyond standard Ethernet.

2.2.5.a User Story #5 Breakdown

Utilize the other pins and ports to find an alternative form of communication.

2.2.6 User Story #6

Develop a message passing protocol for the communication.

2.2.6.a User Story #6 Breakdown

There is no message passing protocol for the other modes of communication. They must be developed and benchmarked.

2.3 Requirements and Design Constraints

2.3.1 System Requirements

2.3.2 Network Requirements

2.3.3 Development Environment Requirements

2.3.4 Project Management Methodology

Oral progress reports are due on Tuesdays and Thursdays at one o'clock in the afternoon. These reports are given to Dr. Karlsson.

- Trello is used to manage the backlog and status.
- All parties have access to the sprint and product backlogs.
- Six sprints will be completed this project
- The sprint cycles are a couple weeks long.
- No restrictions on source control.

2.4 Specifications

2.5 Product Backlog

2.6 Research or Proof of Concept Results

2.7 Supporting Material

3

Project Overview

3.1 Team Member's Roles

- Andrew Hoover - hardware/testing/software
- Christine Sorensen - team lead/parallel programmer/software/documentation

3.2 Project Management Approach

Project will be split into six sprints, each lasting two weeks. Items from the backlog are organized and assigned into these sprints.

Product backlog is located on the Trello board. Documents and source code is located on Github.

Formal meetings take place twice a week on Tuesdays and Thursdays at 1:00pm in advisor's, Dr. C. Karlsson's, office. Casual meetings are planned as needed.

3.3 Stakeholder Information

3.3.1 Customer or End User (Product Owner)

3.3.2 Management or Instructor (Scrum Master)

3.3.3 Investors

3.3.4 Developers –Testers

3.4 Budget

3.5 Intellectual Property and Licensing

3.6 Sprint Overview

3.7 Terminology and Acronyms

- Benchmarking - running tests in order to assess the perfomance of the computers
- iperf - tool in standard Debian repositories to test network speed

3.8 Sprint Schedule

3.9 Timeline

3.10 Backlogs

3.11 Burndown Charts

3.12 Development Environment

3.13 Development IDE and Tools

3.14 Source Control

3.15 Dependencies

3.16 Build Environment

3.17 Development Machine Setup

4

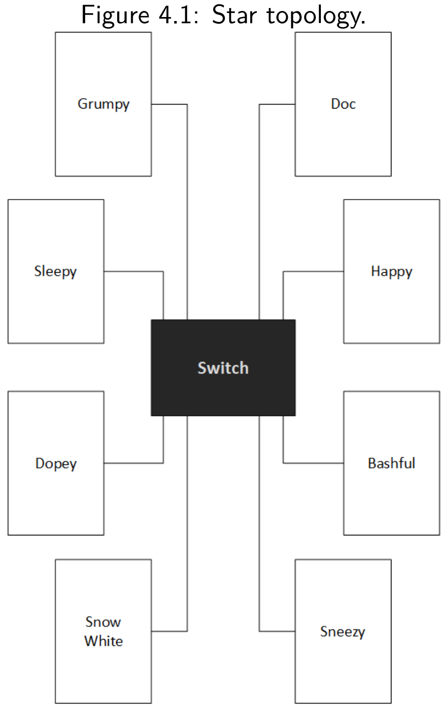
Design and Implementation

The design of the cluster is going change as we test different configurations to determine which is capable of producing the most gigaflops, and as we test different connection methods as we design our custom data transfer protocol. This section will outline the different designs we have created, how they were implemented, and our plans for implementing future designs going forward.

4.1 Architecture and System Design

The first architecture we implemented was a star topology, with each device of the eight devices connected over Ethernet to a central switch. Each device was configured to be on the same network and capable of communicating over the switch via IP addresses. We configured the /home directory of our head node, Snow White, to be an NFS export that the seven other nodes would mount in their /home directory. This allowed for any change to be made on one device to be mirrored on the others. Also, it allowed for MPI programs located anywhere in the /home directory to be run, as all the devices would see the same executable in the same path.

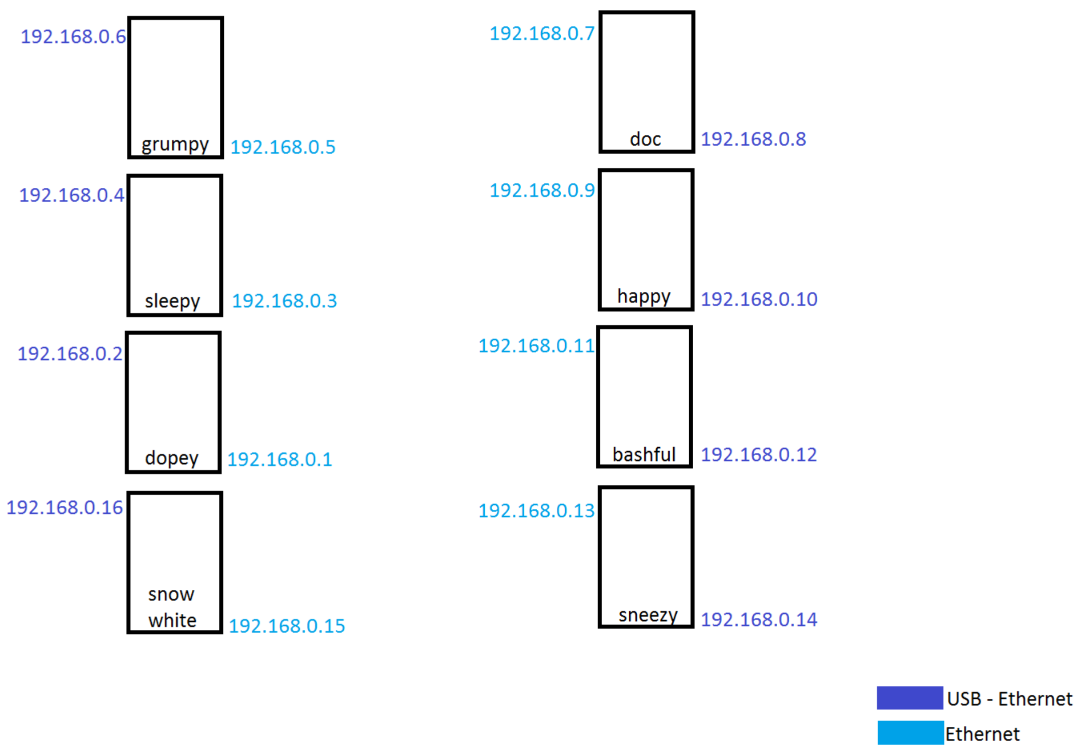
Figure 4.1: Star topology.



The next tolopogy we implemented was a ring. A USB to Ethernet device was attached to each ODroid, giving them a total of two network interfaces. Each interface was assigned a unique IP address, as shown in Figure Yada.

This allowed the cluster to be connected without the switch, and instead with each ODroid connected to the ajacent two forming a ring. For communication to be possible, routing information had to be specified to allow each device to know the path to follow to connect to the other seven. To do this, the /etc/networking/interfaces file was edited to add seven routes, four from one network interface and three from the other. Each route specified which port to use to next to go to any given ODroid. For instance, Snow White had a route with the information that if any program tries to access Sleepy on 192.168.0.3, it would go through Dopey at 192.168.0.1, and then Dopey had it's own routing table for how to access Sleepy. The routing tables were designed such that each ODroid would take the fewest possible hops to get to any other. When the ODroids were equidistant, such as Snow White to Doc where going clockwise or counter-clockwise would each take four hops, the convention used was to go clockwise. This would allow the network traffic to overlap as little as possible in order to maximize effeciency. Therefore, each ODroid had a unique routing table.

Figure 4.2: IP addresses for ring topology.



The third architecture we used was a hypercube topology. To implement it, a third network interface was added by connecting a second USB to Ethernet device on the final USB 3.0 port. An IP address was assigned to that interface as well, and more routing table information was added. This allowed more connections to be made, decreasing the maximum amount of hops needed to get from any one node to another, as shown in Figure Blada.

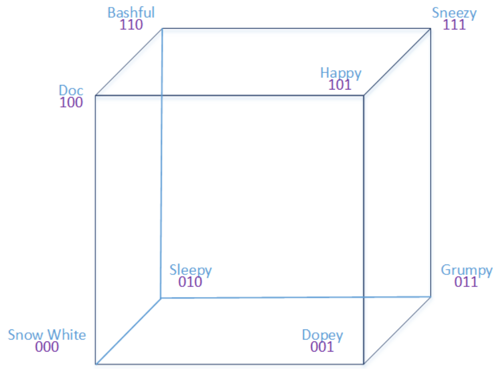
4.1.1 Design Selection

For the purposes of benchmarking the cluster, the star topology was used. It was found that MPI does not work with multiple network interfaces on the same subnet existing on the same node. Therefore, in order to be able to use the benchmarking tool Linpack the star topology had to be used, as it was the only design that had only one network interface per ODroid.

4.1.2 Data Structures and Algorithms

4.1.3 Data Flow

Figure 4.3: Diagram of hypercube topology.



4.1.4 Communications

4.1.5 Classes

4.1.6 UML

4.2 Major Component #1

4.2.1 Technologies Used

4.2.2 Component Overview

4.2.3 Phase Overview

4.2.4 Architecture Diagram

4.2.5 Data Flow Diagram

4.2.6 Design Details

First, we made the devices able to communicate on the same network. We assigned each device a static IP address by editing the /etc/networking/interfaces file to incluce the IP address chosen for the device. All addresses were on the 192.168.1.X network, and the last number was 11 through 18 for the eight devices.

Next, we set each device to be able to use our naming convention in place of an IP address for any purpose, such as by using "ssh sleepy" instead of "ssh 192.168.1.13". To do this, we changed the /etc/hostname file to replace "odroid" with the name we wanted, and added entries to /etc/hosts to include the IP address of the other seven decives.

We then set the /home directory of Snow White to be an NFS share that could be mounted on the dwarfs. After nfs-kernel-server was installed on Snow White, we edited it's /etc/exports file to include /home as an export. The dwarfs then installed nfs-common and used the command "mount SnowWhite:/home /home" to mount the home directory of Snow White over their own.

To make this mount process automatic on boot, we edited the /etc/fstab file on each dwarf to make them mount Snow White's home directory as part of the boot process. This proved unsuccesful, however, on all devices except for on. As a work around, we wrote a Python tool that can be run from Snow White to mount it's home directory on the dwarfs.

```
#!/bin/usr/python

import os

def Main():
```

```
        hosts = [ 'snow_white', 'dopey', 'sleepy', 'grumpy', 'doc', 'happy',
              'bashful', 'sneezy' ]

        for host in hosts:
                if host != 'snow_white':
                        cmd = "ssh odroid@" + host + " 'sudo mount -t nfs snow_white:/
                        home /home'"
                        os.system( cmd )

if __name__ == '__main__':
        Main()
```

4.3 Major Component #2

4.3.1 Technologies Used

4.3.2 Component Overview

4.3.3 Phase Overview

4.3.4 Architecture Diagram

4.3.5 Data Flow Diagram

4.3.6 Design Details

4.4 Major Component #3

4.4.1 Technologies Used

We used several Linux system configuration tools to implement the cluster. We used the files

- etc/network/interfaces
- etc/exports
- etc/hostname
- etc/hosts
- etc/fstab

for several different configuration setting. We also used a few packages availible from the defual debian repositories.

- nfs-kernel-server
- nfs-common
- mpi-dev

Finally, we used SSH tools that are installed by default on Ubuntu 15.

4.4.2 Component Overview

The features implented by this configuration were:

- All devices recognizing the others over Ethernet.
- SSH without requiring a password.
- Mount home directory of Snow White on the dwarfs.
- Running MPI code on all devices.

4.4.3 Phase Overview

4.4.4 Architecture Diagram

4.4.5 Data Flow Diagram

4.4.6 Design Details

5

System and Unit Testing

Testing for our project inluded mainly testing aspects of our hardware. Future sprints will include testing for the software we design as our message passing protocol over USB or GPIO pins, but for the first three sprints our goal was to chose the most efficient hardware and benchmark the amount of gigaflops able to be produced by the cluster, and how much the Ethernet network slows down the system. To accomplish this, we tested the physical capabilities of the hardware.

5.1 Overview

We first tested the computational speed of the ODROID XU4 and Raspberry PI 2B. We then tested the amount of power used by each device with a voltimeter. From these tests, the ODROID produced more gigaflops per watt per dollar, influencing our choice to build our cluster out of ODROID XU4s.

5.2 Dependencies

To test the gigaflops of the cluster, we build and used LINPACK. To test the Ethernet speed, the tool iperf was used.

5.3 Test Setup and Execution

The first tests were to compare the ODROID XU4 and Raspberry Pi 2B. The computational speed was testing by writing a program in C++ to read two large arrays of floating point values into memory and compute four different computations accross the arrays; addition, multiplication, divison, and sine. We recorded how long it took each device to complete the calcuations, and used the timing as a point of comparision between the two devices. The power consumption during runtime was also tested using a voltimeter while the C++ code was executed. The amount of gigflops per watt per dollar was computed to favor the ODROID, influencing our decision to build the cluster out of them.

The next series of tests were the hardware capabilities of the ODROID xU4. The Ethernet speed was tested by using iperf on two nodes, which is a tool availible in the default debian repositories to test the connection speed over Ethernet. Also, a USB to Ethernet device was tested in the same way. Once we knew that information, we tested the amount of gigaflops as recorded by a reliable tool, LINPACK. To do this we had to download the source code, create a Makefile for the ARM architecture, and build the executable. Once done, we adjusted the settings to a square matrix of size 38,600, used 4 and 16 for the P and Q values that determine how many cores to run the code on, and ran the test.

5.4 System Testing

5.5 System Integration Analysis

5.6 Risk Analysis

5.6.1 Risk Mitigation

5.7 Successes, Issues and Problems

5.7.1 Changes to the Backlog

6

Prototypes

This chapter is for recording each prototype developed. It is a historical record of what you accomplished in 464/465. This should be organized according to Sprints. It should have the basic description of the sprint deliverable and what was accomplished. Screen shots, photos, captures from video, etc should be used.

6.1 Sprint 1 Prototype

The work completed in the first sprint was largely information gathering. The goal was to determine which single-board computer best matched our needs for the cluster. We accomplished this by measuring the speed of each device in terms of addition, multiplication, division and the sine function accross numberous floating point values.

6.1.1 Deliverable

- Mission Statement
- User Stories
- Number Generating Code
- Benchmark Code
- Benchmark Log
- Signed Software Contract
- Updated Design Document

6.1.2 Backlog

- Decide on a computer based on the results of the benchmarking
- Calculate prices on supplies and computers while maintaining below the budget
- Ordering said supplies and computers
- Build the cluster to perform floating-point operations
- Benchmark the cluster
- Experiment with different topologies
- Create a new mode of communication

6.1.3 Success/Fail

We successfuly found data for which device was faster, the cost of each device, and the power consumption while running.

6.2 Sprint 2 Prototype

This sprint also feature information gathering and preliminary work for knowlege we would need in creating the cluster. The decision was made to build the cluster out of ODroids, and the devices were ordered. The speed of the Ethernet port was tested while we waited for the devices to come in.

6.2.1 Deliverable

- Budget
- Hardware Test
- Switch Benchmark
- Ethernet Benchmark
- USB to Ethernet Benchmark
- MPI Code
- Message-Passing Protocol

6.2.2 Backlog

- Build the cluster
- Code for the cluster
- Benchmark the cluster
- Experiment with different topologies
- Create a new mode of communication

6.2.3 Success/Fail

The ODroids were backordered, and took two weeks longer than expected to arrive. In the meantime, we were able to test the speed of the Ethernet port with the ODroid that we had, and we tested the speed of a USB to Ethernet device.

6.3 Sprint 3 Prototype

6.3.1 Deliverable

- Built cluster
- MPI code
- Mounted home directory
- Shutdown script
- Mounting script
- LINPACK and ATLAS installed on ODROIDs

- MPI installed on ODROIDs
- Hostnames
- Fixed IP Addresses
- SSH configuration

6.3.2 Backlog

- Research new connection methods
- Benchmark the cluster
- Experiment with different topologies
- Create a new mode of communication
- Design documentation
- Research symposium
    1. Complete abstract
- Design Fair

6.3.3 Success/Fail

6.4 Sprint 4 Prototype

6.4.1 Deliverable

- Graphs of total gigaflops performed depending on amount of devices used.
- Debian package of LINPACK for ARM.
- Found USB communication to not be feasible.
- Able to send bits over GPIO between ODroid devices.
- MICS abstract.

6.4.2 Backlog

- MICS presentation.
- SDSMT Research Symnposium.
- Design Documentation.
- Design Fair.

6.4.3 Success/Fail

6.5 Sprint 5 Prototype

6.5.1 Deliverable

- Design for the hypercube and ring topology.
- Routing tables for each of the ODROIDs.
- The cluster connected with new topology.
- Able to send bits over GPIO between ODroid devices.
- Acceptance into MICS.
- First draft of MICS paper.
- SDSMT Research Symposium abstract.

6.5.2 Backlog

- MICS presentation.
- SDSMT Research Symnposium.
- Completed hypercube cluster.
- Conglomerate data results.
- Design Documentation.
- Design Fair.

6.5.3 Success/Fail

7

Release – Setup – Deployment

7.1 Deployment Information and Dependencies

7.2 Setup Information

7.3 System Versioning Information

8

User Documentation

8.1 User Guide

8.2 Installation Guide

8.3 Programmer Manual

9

Business Plan

9.1 Business Model

9.2 Market and Competition

9.3 Regulatory environment

9.4 Intellectual Property and Freedom to Operate

9.5 Management Team and Advisors

9.6 Sources and Uses of Capital

9.7 Financial Statements

9.8 Metrics and Milestones

9.9 Exit Plan

10

Experimental Log

For research projects one needs to keep a log of all research/lab activities.

10.1 Benchmarking the Individual Computers

9/17/15 PcDuino isn't working according to Dr. Karlsson. The PcDuinos are about $160 each, so chances were that the PcDuino wasn't going to be selected for the cluster. We will not put the PcDuino in consideration with our benchmarking.

9/17/15 PcDuino isn't working according to Dr. Karlsson. The PcDuinos are about $160 each, so chances were that the PcDuino wasn't going to be selected for the cluster. We will not put the PcDuino in consideration with our benchmarking.

9/22/15 We begin work on benchmarking the remaining candidates. The code we will be using to benchmark the two devices will test the addition, multiplication, division, trigonmetric function in single and double point precision of two massively large arrays filled with random numbers.

9/29/15 OpenMP is added to the benchmark code so the program runs on all cores. Results are as follows:

Length of Time (seconds)				
Device	Addition	Multiplication	Division	Sine
ODroid 4xU	29.925	31.341	37.032	227.40
Raspberry Pi 2B	221.645	221.034	297.204	1468.63

9/30/15 The gigaflops are calculated.

Gigaflops				
Device	Addition	Multiplication	Division	Sine
ODroid 4xU	0.311	0.297	0.251	0.0410
Raspberry Pi 2B	0.0420	0.0421	0.0313	0.00634

10/1/15 The wattage is measured when the devices are running these operations. Using the wattage, the metric of GFlops/Dollar/Watt is calculated.

Gigaflops per Dollar per Watts				
Device	Addition	Multiplication	Division	Sine
ODroid 4xU	0.00028	0.000268	0.000226	0.0000369
Raspberry Pi 2B	0.0003	0.0003	0.000224	0.0000453

10/1/15 The results show that the Raspberry Pi and the ODroid perform nearly the same. The Raspberry Pi in our benchmarking proved the best. However, it is inconclusive as to which computer will be used.

10/24/15 Decided to go with the ODroid. Performance and the number of ports outweighted the cost of the Raspberry Pi.

10.2 Ethernet Benchmark

10/25/15 Created network between a machine with gigabit ethernet port and ODroid. Installed n both the server machine and ODroid.

Ethernet Speed	
Device	Speed (Mbps)
ODroid XU4	615 - 625

10.3 Hardware Test

11/03/15 All eight ODroids are functional. To test them, each device was connected to a router with internet access via ethernet. A monitor was connected through HDMI, and the mouse and keyboard were connected to both USB 3.0 ports to ensure they worked. The packages mpi-default-dev and openmpi-bin were installed. The test mpi code found this director and successfully compiled and executed. No issues found.

10.4 Switch Benchmark

11/03/15 With two ODroid devices attached to the switch via direct ethernet (no USB 3.0 to ethernet adapter), the connection speed was tested with iperf.

Switch Speed	
Device	Speed (Mbps)
ODroid XU4	775 - 800

This is notably faster than the previous benchmark between one ODroid and a non-ODroid machine not using a switch.

10.5 USB to Ethernet Benchmark

11/03/15 Tested speed of USB 3.0 to ethernet adapter using iperf. Found slower than direct ethernet conection. Test speeds varied greatly. The USB ethernet adapted was faster acting as a server than a client.

USB 3.0 to Ethernet Adapter Speed	
Device	Speed (Mbps)
ODroid XU4	300 - 700

In contrast, the ethernet connections were consistent reguardless of which device was the client or server.

11/05/15 There isn't a found way to connect two devices over USB-ethernet to ethernet directly. When attached to the switch, the devices can communicate. IF using the USB to ethernet adapter, they would be directly connected without the switch. Therefore, it was unable to directly connect devices.

The drastically lower speed of the USB to ethernet adapters and the inability to directly connect the devices means that the devices are very unlikely to be useful for this project.

11

Research Results

11.1 Result 1

11.2 Result 2

11.3 Conclusions

11.4 Further work

SDSMT SENIOR DESIGN
SOFTWARE DEVELOPMENT AGREEMENT

This Software Development Agreement (the “Agreement”) is made between the SDSMT Computer Science

Senior Design Team ____,
(“Student Group”)

consisting of team members ____,
(“Student Names”)

and Sponsor ____,
(“Company Name”)

with address: ____.

[Note: Bracketed material is included to suggest content that will vary with each agreement. I STRONGLY SUGGEST THAT THE INSTRUCTOR LOOK AT THE COMPLETED AGREEMENT BEFORE YOU SIGN IT!!]

1 RECITALS

1. Sponsor desires Senior Design Team to develop software [for use in Sponsor's simulation platform for optical fiber transmissions of digitized video signals] (the ”Field”).
2. Senior Design Teams willing to develop such Software.

NOW, THEREFORE, in consideration of the mutual covenants and promises herein contained, the Team and Sponsor agree as follows:

2 EFFECTIVE DATE

This Agreement shall be effective as of ____________________ (the “Effective Date”).

3 DEFINITIONS

1. “Software” shall mean [the computer programs in machine readable object code form and any subsequent error corrections or updates supplied to Sponsor by Senior Design Team pursuant to this Agreement.] [Depending on the particulars of each agreement, any or all of the following may need to be specified. If they are relevant, they should be used throughout, modifying the standard form as appropriate.]
2. “Acceptance Criteria” means the written technical and operational performance and functional criteria and documentation standards set out in the [project plan.]
3. “Acceptance Date” means [the date for each Milestone when all Deliverables included in that Milestone have been accepted by Sponsor in accordance with the Acceptance Criteria and this Agreement.]
4. “Deliverable” means a deliverable specified in the [project plan.]
5. “Delivery Date” shall mean, [with respect to a particular Milestone,] the date on which University has delivered to Sponsor all of the Deliverables [for that Milestone] in accordance with [the project plan and] this Agreement.

6. “Documentation” means the documents, manuals and written materials (including end-user manuals) referenced, indicated or described in [the project plan] or otherwise developed pursuant to this Agreement.
7. “Milestone” means the completion and delivery of all of the Deliverables or other events which are included or described in [the project plan] scheduled for delivery and/or completion on a given target date; a Milestone will not be considered completed until the Acceptance Date has occurred with respect to all of the Deliverables for that Milestone.

4 DEVELOPMENT OF SOFTWARE

1. Senior Design Team will use its best efforts to develop the Software described in [the project plan.] The Software development will be under the direction of or his/her successors as mutually agreed to by the parties (“Team Lead”) and will be conducted by the Team Lead. The Team will deliver the Software to the satisfaction of the course instructor that reasonable effort has been made to address the needs of the client. The Team understands that failure to deliver the Software is grounds for failing the course.
2. Sponsor understands that the Senior Design course’s mission is education and advancement of knowledge, and, consequently, the development of Software must further that mission. The Senior Design Course does not guarantee specific results or any results, and the Software will be developed only on a best efforts basis. The Software is considered PROOF OF CONCEPT only and is NOT intended for commercial, medical, mission critical or industrial applications.
3. The Senior Design instructor will act as mediator between Sponsor and Team; and resolve any conflicts that may arise.

5 COMPENSATION

[This is entirely subject to negotiation. Normally NO COMPENSATION occurs in a Senior Design Project. On occasion an intern status and wage is appropriate.]

6 CONSULTATION AND REPORTS

1. Sponsor’s designated representative for consultation and communications with the Team Lead shall be

   ________________________________ or such other person as Sponsor may from time to time designate to the Team Lead (“Designated Representative”).
2. During the Term of the Agreement, Sponsor’s representatives may consult informally with course instructor regarding the project, both personally and by telephone. Access to work carried on in University facilities, if any, in the course of this Agreement shall be entirely under the control of University personnel but shall be made available on a reasonable basis.
3. The Team Lead will submit written progress reports. At the conclusion of this Agreement, the Team Lead shall submit a comprehensive final report in the form of the formal course documentation at the conclusion of the Senior Design II course.

7 CONFIDENTIAL INFORMATION

1. The parties may wish, from time to time, in connection with work contemplated under this Agreement, to disclose confidential information to each other (“Confidential Information”). Each party will use reasonable efforts to prevent the disclosure of any of the other party’s Confidential Information to third parties for

a period of three (3) years after the termination of this Agreement, provided that the recipient party's obligation shall not apply to information that:

  (a) is not disclosed in writing or reduced to writing and so marked with an appropriate confidentiality legend within thirty (30) days of disclosure;
  (b) is already in the recipient party's possession at the time of disclosure thereof;
  (c) is or later becomes part of the public domain through no fault of the recipient party;
  (d) is received from a third party having no obligations of confidentiality to the disclosing party;
  (e) is independently developed by the recipient party; or
  (f) is required by law or regulation to be disclosed.

2. In the event that information is required to be disclosed pursuant to subsection (6), the party required to make disclosure shall notify the other to allow that party to assert whatever exclusions or exemptions may be available to it under such law or regulation.

8 INTELLECTUAL PROPERTY RIGHTS

[Negotiated on a case-by-case basis. This must address who owns the algorithms and who owns the source code. For example: All deliverables become property of the Sponsor. Roughly: If the idea originates with the sponsor, or if a sponsor pays you to develop an idea, then they have legitimate claim to the IP. If the idea originates from the University (through faculty or staff) then the University has legitimate claim. If the idea is yours (student) and you develop it without external compensation then you have legitimate claim.]

9 WARRANTIES

The Senior Design Team represents and warrants to Sponsor that:

1. the Software is the original work of the Senior Design Team in each and all aspects;
2. the Software and its use do not infringe any copyright or trade secret rights of any third party.

No agreements will be made beyond items (1) and (2).

10 INDEMNITY

1. Sponsor is responsible for claims and damages, losses or expenses held against the Sponsor. [Sponsor may have something to add here.]
2. Sponsor shall indemnify and hold harmless the Senior Design Team, its affiliated companies and the officers, agents, directors and employees of the same from any and all claims and damages, losses or expenses, including attorney's fees, caused by any negligent act of Sponsor or any of Sponsor's agents, employees, subcontractors, or suppliers.
3. NEITHER PARTY TO THIS AGREEMENT NOR THEIR AFFILIATED COMPANIES, NOR THE OFFICERS, AGENTS, STUDENTS AND EMPLOYEES OF ANY OF THE FOREGOING, SHALL BE LIABLE TO ANY OTHER PARTY HERETO IN ANY ACTION OR CLAIM FOR CONSEQUENTIAL OR SPECIAL DAMAGES, LOSS OF PROFITS, LOSS OF OPPORTUNITY, LOSS OF PRODUCT OR LOSS OF USE, WHETHER THE ACTION IN WHICH RECOVERY OF DAMAGES IS SOUGHT IS BASED ON CONTRACT TORT (INCLUDING SOLE, CONCURRENT OR OTHER NEGLIGENCE AND STRICT

LIABILITY), STATUTE OR OTHERWISE. TO THE EXTENT PERMITTED BY LAW, ANY STATUTORY REMEDIES WHICH ARE INCONSISTENT WITH THE PROVISIONS OF THESE TERMS ARE WAIVED.

11 INDEPENDENT CONTRACTOR

For the purposes of this Agreement and all services to be provided hereunder, the parties shall be, and shall be deemed to be, independent contractors and not agents or employees of the other party. Neither party shall have authority to make any statements, representations or commitments of any kind, or to take any action which shall be binding on the other party, except as may be expressly provided for herein or authorized in writing.

12 TERM AND TERMINATION

1. This Agreement shall commence on the Effective Date and extend until the end of classes of the second semester of Senior Design (CSC 467), unless sooner terminated in accordance with the provisions of this Section ("Term").
2. This Agreement may be terminated by the written agreement of both parties.
3. In the event that either party shall be in default of its materials obligations under this Agreement and shall fail to remedy such default within thirty (30) days after receipt of written notice thereof, this Agreement shall terminate upon expiration of the thirty (30) day period.
4. Any provisions of this Agreement which by their nature extend beyond termination shall survive such termination.

13 ATTACHMENTS

Attachments A and B are incorporated and made a part of this Agreement for all purposes.

14 GENERAL

1. This Agreement constitutes the entire and only agreement between the parties relating to the Senior Design Course, and all prior negotiations, representations, agreements and understandings are superseded hereby. No agreements altering or supplementing the terms hereof may be made except by means of a written document signed by the duly authorized representatives of the parties.
2. This Agreement shall be governed by, construed, and enforced in accordance with the internal laws of the State of South Dakota.

15 SIGNATURES

Replace with name of student #1	Date
Replace with name of student #2	Date
Replace with name of student #3	Date
Replace with name of sponsor's representative	Date

A

Product Description

Write a description of the product to be developed. Use sectioning commands as neccessary.

NOTE: *This is part of the contract.*

B

Publications

C

Sprint Reports

1 Sprint Report #1

Overview

Deliverables

- Mission Statement
- User Stories
- Number Generating Code
- Benchmark Code
- Benchmark Log
- Signed Software Contract
- Updated Design Document

Work for this sprint included:

- Wrote Mission Statement and Elevator Speech
- Drew up Software Contract
- Wrote user stories
- Obtained ODroid 4xU, Raspberry Pi 2B, and PcDuino 8 single-board computers
- Wrote number generating code
- Wrote benchmark code that ran addition, multiplication, division, and sine floating point operations
- Added OpenMP to run the benchmark code on all cores
- Ran the code each of the single-board computers
- Logged times
- Calculated the GFlops
- Calculated the GFlops/Dollar/Watts
- Determined best computer

Work that is carried over into Sprint 2 is as follows:

- Using the benchmark results to determine which computer to use
- Order more of the computers that proved best from Sprint 1 and maintain the given budget of $1,200
- Find a topology that best fits the cluster

Backlog

- Decide on a computer based on the results of the benchmarking
- Calculate prices on supplies and computers while maintaining below the budget
- Ordering said supplies and computers
- Build the cluster to perform floating-point operations
- Benchmark the cluster
- Experiment with different connections
- Create a new mode of communication

2 Sprint Report #2

Deliverables

- Budget
- Harware Tests
- Switch Benchmark
- Ethernet Benchmard
- USB to Ethernet
- MPI Code
- Message-Passing Protocol Design
- Updated Design Document

Overview

During this sprint, our team first decided to use the ODROID devices instead of Raspberry PIs based on our findings from sprint one. We roughly planned out our budget for ethernet cables, the switch, varius USB cables, power strips, and the devices themselves and extra memory for them. At this stage, we did not chose exactly what we would buy, but we narrowed our budget enough to know that the number of devices we could have without going over the limit set by our sponser was eight. Dr. Karlsson then ordered seven more ODroids, an eight port switch, eight ethernet cables, and one USB to RJ45 device to benchmark USB to ethernet speeds. We encountered an issue where the order was backordered, and we did not receive the devices until over two weeks laters. Once we did get them, we benchmarked the speed over ethernet between two devices through the switch, and the speed when one device was using a USB to RJ45 to connect to the switch. We also attempted to benchmark the direct connection between RJ45 on one device connected directly to the ethernet of another device, but found that we would need a crossover cable. We concluded our sprint by setting up all eight devices by installing MPI and giving them static IP addresses on the network over the switch.

Dr. Karlsson as been informed of a research synposium and has suggested our team take part of it. We are adding this to our goals for Senior Design to take part in that.

Work for this sprint included:

Andrew Hoover

- Benchmarked:
  - Switch
  - Ethernet
  - USB to Ethernet
- Tested hardware
- Tested MPI code on ODROIDs

Samantha Krantz

- Researched supplies
- Finialized budget
- Researched patents and other clusters

Christine Sorensen

- Designed message-passing protocol
- Sampled MPI code
- Updated Design Document
- Sprint report document

Work that is carried over into Sprint 3 is as follows:

- Design the cluster
- Build the cluster
- Benchmark cluster
- Test message-passing using the other pins and ports

Backlog

- Build the cluster to perform floating-point operations
- Benchmark the cluster
- Experiment with different connections
- Create a new mode of communication

Goals

- Design Fair
- Research Symposium

3 Sprint Report #3

Deliverables

- Built cluster
- MPI code
- Mounted home directory
- Shutdown script
- Mounting script
- LINPACK and ATLAS installed on ODROIDs
- MPI installed on ODROIDs
- Hostnames
- Fixed IP Addresses
- SSH configuration

3.1 Overview

- Built cluster
  - All parts for the cluster were ordered. Parts included:
    - 8 ODROID XU4's
    - Switch
    - Power box
    - Ethernet cables
    - Acrylic board
    - Accessories, such as handles and power switches
  - The ODROIDs, switch and power box were mounted onto the acrylic board. ODROIDs were connected to the switch and the power box.
- Set up cluster
  - Fixed IP addresses to each ODROID
  - Assigned a hostname to each ODROID
  - Configured cluster network
  - NFS share
  - Configured sudo
  - Set-up ssh between ODROIDS
- Benchmarked cluster
  - Installed LINPACK
  - Wrote MPI code to run on the cluster
  - Ran the MPI code with LINPACK on the cluster
  - Gathered data
- Setbacks
  - Backorder

    * The remaining ODROIDs were backordered, delaying to assembling of the cluster.
  - Broken ODROIDs
    * Two of the ODROIDs needed to be replaced. It was believed that the two ODROIDs were placed on the power supply without covering which exposed the soldering on the bottom to sepatate it from the metal which might have crossed circuits causing the ODROIDs to not power up. Ordering the new ODROIDs put us behind in our timeline.
  - Installing LINPACK
    * The installation of LINPACK was complicated and prompted issues. It took longer than expected to complete.

Work for this sprint included:

Andrew Hoover

- Installed LINPACK and ATLAS
- Assembled cluster
- Replaced broken ODROIDS
- Fixed IP address on the ODROIDS
- Connected all ODROIDS over network
- Accessed the internet through the ODROIDS
- Mounted home directory of Snow White on the other ODROIDS
- Wrote script to shut down all ODROIDs
- Wrote script to run LINPACK on ODROIDs for specified number of processes
- Removed the sudo password
- Wrote script to change network configuration to all access to the internet or local network
- Assembled cluster
- Debugged boot-up error
- Benchmarked cluster using LINPACK and MPI code

Samantha Kranstz

- Created client presentation
- Assembled cluster
- Debugged boot-up error
- Benchmarked cluster using LINPACK and MPI code
- Researched patents

Christine Sorensen

- Assigned hostnames to each of the ODROIDs
- Configured ssh on ODROIDs
- Replaced broken ODROIDs
- Wrote MPI code to run on all cores of the clusters
- Updated design documentation
- Wrote script to mount home directory of Snow White onto all ODROIDS at once
- Added ODROID's hostnames to the others' known hosts list
- Wrote sprint report
- Assembled cluster
- Debugged boot-up error
- Benchmarked cluster using LINPACK and MPI code

Work that is carried over into Sprint 4 is as follows:

- Research new methods of connection
- Take action on these new methods
- Benchmark
- Complete abstract for research symposium

Backlog

- Research new connection methods
- Benchmark the cluster
- Experiment with different topologies
- Create a new mode of communication
- Design documentation
- Research symposium
  - Complete abstract
- Design Fair

4 Sprint Report #4

5 Sprint Report #5

6 Sprint Report #6

D

Industrial Experience and Resumes

1 Resumes

Andrew Hoover
3616 Michigan Ave
Manitowoc, WI 54220
920-629-3227
Andrew.Hoover@mines.sdsmt.edu

Objective

Seeking position as a software engineer working with storage and kernel level technologies.

Work Experience

August 2015 – Present. Software Engineering Intern. Nexenta.

- Continuing to work for Nexenta remotely during the school year.
- Tasked with organizing, documenting and modifying system build procedures.
- Continued improving monitoring and notification tools for the Long Running Test System.

May 2015 – August 2015. Software Engineering Intern. Nexenta.

- Worked with sustainability and kernel engineering teams.
- Created and implemented monitoring and notification tools for the Long Running Test System.
- Created and implemented tools to drive the CIFS and NFS activity.
- Added new features and new compatibility to tools created by other team members.
- Communicated with team members in several countries to complete projects.

May 2014 - August 2014. Wal-Mart associate

- Worked during the summer part time as a Wal-Mart associate.

Education

- Currently attending South Dakota School of Mines and Technology, graduating May 2016. 112 credits complete, current GPA: 3.2
- Lincoln High School in Manitowoc, WI, 2012

Skills

- Familiarity with Ubuntu, Fedora, Illumos, NexentaStor
- Telecommuting
- Able to communicate effectively to complete tasks with teammates.
- Experience in completing code based projects with teams.

Technologies

- VMWare Workstation and VSphere
- Github
- Visual Studio
- Shell scripting
- Python scripting
- Languages:
    - C and C++
    - ARM assembly
    - Java and C#
    - Python, SQL, lisp

References available upon request.

CHRISTINE SORENSEN

619.5 Main Street
Apt 15
Rapid City, SD 57701

(605) 670-9808
www.linkedin.com/in/sorensenc
christine.sorensen@mines.sdsmt.edu

Objective

Senior Computer Science Undergraduate seeking **full-time opportunities.** Passionate about back-end software development. Specialized in working in a Linux environment with programming languages C++, C, and Python.

EDUCATION

South Dakota School of Mines and Technology—Rapid City, SD (Graduation: **May 2016**)
Computer Science B.S. • Cumulative GPA: 3.1 • Major GPA: 3.5

RELEVANT COLLEGE COURSEWORK

Data Structures • Computer Organization & Architecture • Software Engineering • Parallel Computing • Computer Networks • Assembly Language • Digital Systems • Database and Management Systems • Analysis of Algorithms • Computer Graphics • Technical Communications

University of Regina Regina, SK, Canada (January 2012 – May 2012)
National Student Exchange

University of South Dakota Vermillion, SD (August 2010 – December 2012)

EXPERIENCE

Black Hills Information Security Rapid City, SD (October 2015 – Present)
Software Developer Intern

Sencore Inc. Sioux Falls, SD (May 2015 – August 2015)
World class technology company focused on innovating products and services for professional content providers to enable efficient, high quality video delivery.

Software Engineer Intern

- Wrote a utility in Python that processed VOD streaming on Sencore's monitoring probes.
- Added an expiration license to software that monitors RF measurements using C#.
- Used JavaScript to add features on the web side of the development team's debug page.

Projects

- Playlist generator using C++
- Packet sending simulations using SIMSCRIPT
- Photo-manipulation program in ARM Assembly
- Entrepreneurial presentation competing for the Butterfield Cup
- Solar System simulator in OpenGL
- Microarchitecture emulator in C++

SKILLS

Programming Languages: C++ • C • Python • ARM Assembly • SIMSCRIPT • C# • Common Lisp
Database: MySQL
Web Application: PHP • JavaScript • HTML • CSS
Graphics: OpenGL

**In order of proficiency*

E

Acknowledgment

Thanks

LaTeX Example

LaTeX sample file: Remove from submitted materials

1 Introduction

This is a sample input file. Comparing it with the output it generates can show you how to produce a simple document of your own.

2 Ordinary Text

The ends of words and sentences are marked by spaces. It doesn't matter how many spaces you type; one is as good as 100. The end of a line counts as a space.

One or more blank lines denote the end of a paragraph.

Since any number of consecutive spaces are treated like a single one, the formatting of the input file makes no difference to TeX, but it makes a difference to you. When you use LaTeX, making your input file as easy to read as possible will be a great help as you write your document and when you change it. This sample file shows how you can add comments to your own input file.

Because printing is different from typewriting, there are a number of things that you have to do differently when preparing an input file than if you were just typing the document directly. Quotation marks like "this" have to be handled specially, as do quotes within quotes: " 'this' is what I just wrote, not 'that' ".

Dashes come in three sizes: an intra-word dash, a medium dash for number ranges like 1–2, and a punctuation dash—like this.

A sentence-ending space should be larger than the space between words within a sentence. You sometimes have to type special commands in conjunction with punctuation characters to get this right, as in the following sentence. Gnats, gnus, etc. all begin with G. You should check the spaces after periods when reading your output to make sure you haven't forgotten any special cases. Generating an ellipsis . . . with the right spacing around the periods requires a special command.

TeX interprets some common characters as commands, so you must type special commands to generate them. These characters include the following: $ & % # { and }.

In printing, text is emphasized by using an *italic* type style.

A long segment of text can also be emphasized in this way. Text within such a segment given additional emphasis with Roman *type. Italic type loses its ability to emphasize and become simply distracting when used excessively.*

It is sometimes necessary to prevent TeX from breaking a line where it might otherwise do so. This may be at a space, as between the "Mr." and "Jones" in "Mr. Jones", or within a word—especially when the word is a symbol like *itemnum* that makes little sense when hyphenated across lines.

Footnotes[1] pose no problem.

TeX is good at typesetting mathematical formulas like $x - 3y = 7$ or $a_1 > x^{2n}/y^{2n} > x'$. Remember that a letter like x is a formula when it denotes a mathematical symbol, and should be treated as one.

[1] This is an example of a footnote.

3 Displayed Text

Text is displayed by indenting it from the left margin. Quotations are commonly displayed. There are short quotations

> This is a short a quotation. It consists of a single paragraph of text. There is no paragraph indentation.

and longer ones.

> This is a longer quotation. It consists of two paragraphs of text. The beginning of each paragraph is indicated by an extra indentation.
>
> This is the second paragraph of the quotation. It is just as dull as the first paragraph.

Another frequently-displayed structure is a list. The following is an example of an *itemized* list.

- This is the first item of an itemized list. Each item in the list is marked with a "tick". The document style determines what kind of tick mark is used.
- This is the second item of the list. It contains another list nested inside it. The inner list is an *enumerated* list.
  1. This is the first item of an enumerated list that is nested within the itemized list.
  2. This is the second item of the inner list. LaTeX allows you to nest lists deeper than you really should.

  This is the rest of the second item of the outer list. It is no more interesting than any other part of the item.
- This is the third item of the list.

You can even display poetry.

> There is an environment for verse
> Whose features some poets will curse.
>
> For instead of making
> Them do *all* line breaking,
> It allows them to put too many words on a line when they'd rather be forced to be terse.

Mathematical formulas may also be displayed. A displayed formula is one-line long; multi-line formulas require special formatting instructions.

$$x' + y^2 = z_i^2$$

Don't start a paragraph with a displayed equation, nor make one a paragraph by itself.

4 Build process

To build LaTeX documents you need the latex program. It is free and available on all operating systems. Download and install. Many of us use the TexLive distribution and are very happy with it. You can use a editor and command line or use an IDE. To build this document via command line:

```
alta>  pdflatex SystemTemplate
```

If you change the bib entries, then you need to update the bib files:

```
alta>  pdflatex SystemTemplate
alta>  bibtex SystemTemplate
alta>  pdflatex SystemTemplate
alta>  pdflatex SystemTemplate
```

The template files provided also contain a Makefile, which will make things much easier.

Acknowledgment

Thanks to Leslie Lamport.