BEAGLE BEOWULF CLUSTER

Arjun Vinod, 353925043

Nachiket Kulkarni, 798498504

Abstract

Beagle Beowulf Cluster is a high performing, scalable and homogeneous computing cluster composed by connecting multiple BeagleBone Black Rev-C boards through a very fast network.

Beowulf cluster is a class of computer clusters where large computing power is harvested by connecting commodity, off the shelf hardware usually connected by Ethernet. The term “Beowulf” was coined by Thomas Sterling and Donald Baker at NASA in 1994. Beowulf clusters are generally very cheap, extremely reliable, scalable and very powerful. Linux has an extensive support for such clusters. Special distribution like ClusterKnoppix, Kerrighed, Rocks Cluster Distribution are available which are dedicated to cluster computing.

In Beagle Beowulf Cluster, we are going to connect multiple BeagleBone Black Rev-C boards using a Gigabit Ethernet switch over a LAN. The cluster should be able to run parallel computing applications using tools and libraries like MPI and ScaLAPACK (Scalable Linear Algebra PACKage).

Introduction

The BeagleBone Black is a complete, low-cost, energy-efficient, multipurpose development system with onboard Ethernet, flash storage, video controller and much more. Its primary goal is to offer a true open hardware and community-supported embedded computer for developers and hobbyists. Compared to low-level embedded systems such as Arduino, BeagleBone Black is a fully functional standalone PC that has the size of a credit card.

Without any expansion cards, the power required by the board is 2.5W, which is around 5 percent of the power required by a typical light bulb. Compared to a Intel Pentium III computer with the same clock frequency of 1 GHz, it only needs about 2.5 percent of its power requirement, and even has a better graphics card onboard, but has slightly less memory.

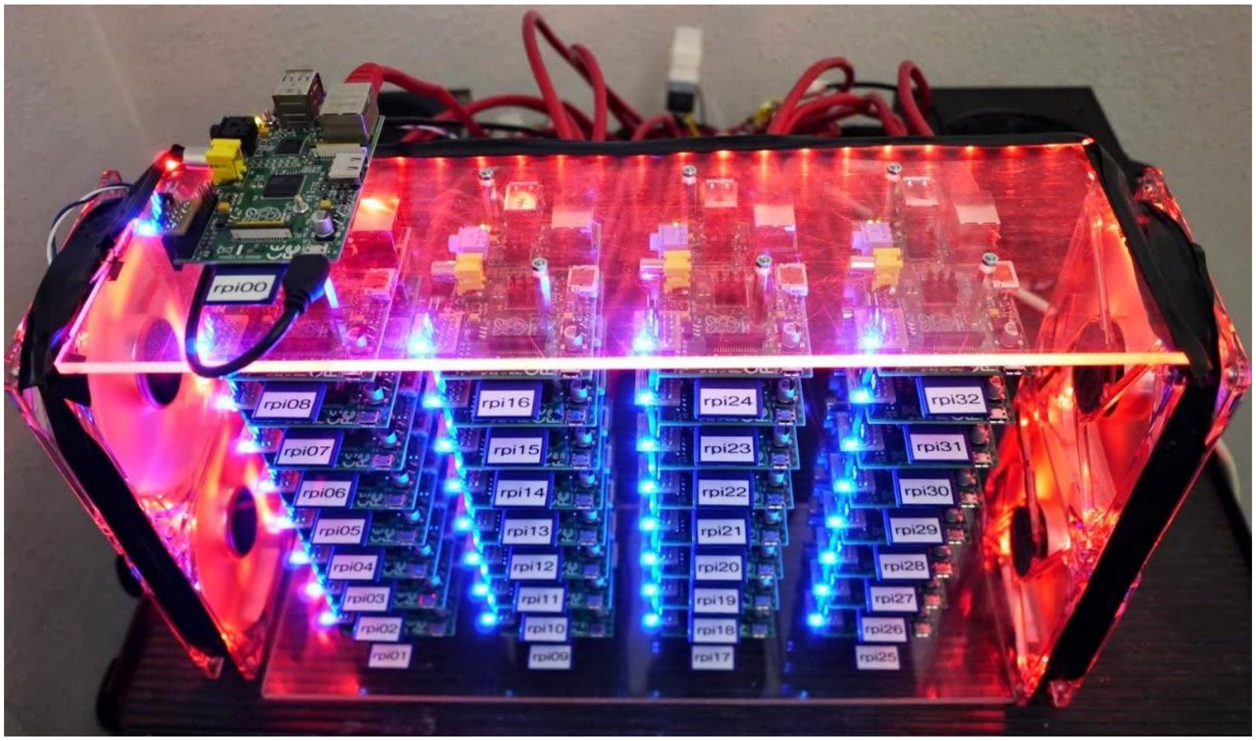
A Beowulf cluster is a computer cluster of what are normally identical, commodity-grade computers networked into a small local area network with libraries and programs installed which allow processing to be shared among them. The result is a high performance parallel computing cluster from inexpensive personal computer hardware.

The name Beowulf originally referred to a specific computer built in 1994 by Thomas Sterling and Donald Becker at NASA. The name "Beowulf" comes from the main character in the old English epic poem Beowulf, which, Sterling chose because the poem describes its eponymous hero as having "thirty men's heft grasp in the gripe of his hand".

One of the main difference between Beowulf cluster and a cluster of workstations is that Beowulf behaves more like a single system rather than many workstations. Beowulf nodes can be thought of as a CPU and memory package which can be plugged into the cluster, just like a CPU or memory module can be plugged into a motherboard.

One of the main motivation behind this project was to build a small supercomputer with high performance parallel computing with low cost and many commodity-grade computers networked into a small cluster. The first thing we did is connect the BeagleBone Black to the internet. By default, the beagle bone is not connected to the internet. We need to turn off the firewall or change the setting as the windows firewall keeps the PC from responding to pings. After the BeagleBone is connected to the internet we tried pinging Google and got response. Moving on, we connected the two BeagleBone Black's to a Local Area Network (LAN) in a star topology. We assign one node as the master node and the other one as a slave node. Both the BeagleBone's are connected to a switch. We use a T-link 8-port Gigabit desktop switch having a total bandwidth of 16Gbps. We are using high performance CAT6 4-pair UTP cables for connecting the BeagleBone's to the 8-port gigabit switch.

The below diagram is a Beowulf cluster made using Raspberry Pi. This was one of the motivating things that led us to build a Beowulf Cluster. We built it using BeagleBone Black Rev C with just two BeagleBone black's.



A Beowulf Cluster built with 33 Raspberry Pi (1 Master node & 32 Slave nodes)

We built a Beowulf Cluster with two BeagleBone Black Rev C where one acts as a master node and the other as a slave node. We can scale the clusters by adding more nodes to it and solve more complex problems quickly. After connecting the BeagleBone's to the LAN, we have to assign them a static IP address. We achieved communication between the BeagleBone Black using Message Passing Interface (MPI). We used openMPI, which is a implementation of MPI. openMPI is an open source. We execute 60x60 matrix multiplication and found the time taken for computation in parallel mode and also serial mode. We compared the computation time and found the serial to be approximately 1.5 times faster than the parallel. This is because we have used only two nodes and because of the communication overhead, the parallel-computing seems slower. With addition of more nodes, the computation time of the Beagle Beowulf cluster will be way faster than that of obtained in serial mode.

Background

A Beowulf cluster with 33 Raspberry Pi has been built before by Joshua Kiepert. A Raspberry Pi is an inexpensive, single-board computer, about the size of a credit card, that is capable of running Linux and other lightweight operating systems. Its processor is quite similar to those found in smartphones. The Raspberry Pi can be plugged into a monitor and keyboard and used to perform all of the popular functions of a regular desktop computer. Originally developed to make computing cheap and available to as many people as possible, the Raspberry Pi has found its way into Joshua Kiepert's research project, in the Electrical and Computer Engineering department at Boise State.

In spite of the obvious benefits, Kiepert's approach is not ideal for every application. Perhaps the biggest downside is that an RPi is nowhere near as powerful as a desktop PC. Also, because of the limited processing capability, the RPiCluster will not reliably support multiple users simultaneously. However, Kiepert believes these drawbacks do not have a negative impact on his current research. In fact, he has found the RPiCluster so successful that he now uses it exclusively for his dissertation work. He has found the performance perfectly acceptable for his simulation needs and now has the added benefit of being able to customize the cluster software to fit his exact requirements.

The RPi cluster built by Kiepert: https://www.youtube.com/watch?v=i_r3z1jYHAc , the video shows how Kiepert designed it and other valuable information regarding the Raspberry Pi-based Beowulf cluster.

We decided to try and understand how to build and configure such a supercomputer with the more modern BeagleBone Black's instead of Raspberry Pi. We succeeded in building our first self-built, low-cost **Beowulf** cluster using two BeagleBone Black's.

Methodology/Technical Approach

In this, we will talk about the hardware components and the tools, libraries that we used to build a Beagle Beowulf cluster.

Hardware Configuration:

1) **BeagleBone Black Rev-C**

   BeagleBone Black Rev-C is the computing backbone of this Beowulf cluster. It is a very powerful, low cost, power efficient, ARM based embedded board that is community-supported development platform for developers and hobbyists. The salient features of the board are:

   i) AM335x 1GHz ARM Cortex A8 Processor
   ii) 512MB DDR3 RAM
   iii) 4GB 8-bit eMMC on-board flash storage
   iv) NEON floating point accelerator
   v) 2xPRU 32-bit microcontrollers
   vi) Comes with a built-in Debian Linux

2) **TP-LINK 8-Port Gigabit Desktop Switch, 10/100/1000Mbps, 16Gbps Switching**

   This great Gigabit Ethernet switch is the networking hub of this cluster. The salient features of this switch are:

   i) 15K Jumbo frame improves performance of large data transfers
   ii) 8 gigabit port switch Ethernet connection integrates 10/100/1000Mps with Auto-Negotiation
   iii) Energy efficient technology which saves up to 80% lower in power consumption
   iv) Non-blocking switching architecture that forwards and filters packets at full wire-speed for maximum throughput

3) **Cat6 Snagless Ethernet Patch Cable**

   These are high performing Cat6 Ethernet Patch Cables with RJ45 connectors providing universal connectivity for LAN network components. They provide much higher bandwidth as compared to CAT5E cables.

4) **SanDisk Extreme Pro 8GB MicroSDHC UHS-1 Flash Memory Card**

This blazing fast, extreme high performing MicroSD flash memory card was bought with an intent of setting up a file server on the master node. Following are a few performance statistics of this flash memory card from it's amazon.com page:

- Sequential Read : 95.325 MB/s
- Sequential Write : 90.660 MB/s
- Random Read 512KB : 89.314 MB/s
- Random Write 512KB : 75.971 MB/s

1) Software Configuration:

Linux Distribution:

While moving from Rev-B to Rev-C, BeagleBone community leaders made a revolutionary decision of replacing Angstrom by Debian as the default operating system of the BeagleBone Black Rev-C board. Debian is backed up by its abundant package repository making it one of the most stable Linux distribution.

2) OpenMPI:

OpenMPI is a High Performance Message Passing Library. It is a library implementation of Message Passing Interface(MPI) specifications. The library is used by many TOP500 supercomputers including ***Roadrunner***, which was the world's fastest supercomputer from June 2008 to November 2009 and ***K computer***, the fastest supercomputer from June 2011 to June 2013. The library is packed with plethora of features for high performance parallel computing like thread safety and concurrency, dynamic process spawning, support for all networks, support for multiple OS platforms etc.

3) MPICH

MPICH is a high-performance and widely portable implementation of the Message Passing Interface (MPI) standard (MPI-1, MPI-2 and MPI-3). The goals of MPICH are:

- To provide an MPI implementation that efficiently supports different computation and communication platforms including commodity clusters (desktop systems, shared-memory systems, multicore architectures), high-speed networks (10 Gigabit Ethernet, InfiniBand, Myrinet, Quadrics) and proprietary high-end computing systems (Blue Gene, Cray)
- To enable cutting-edge research in MPI through an easy-to-extend modular framework for other derived implementations.

OpenMPI and MPICH are designed to meet different needs. MPICH is supposed to be high-quality reference implementation of the latest MPI standard and the basis for derivative implementations to meet special purpose needs. OpenMPI targets the common case, both in terms of usage and network conduits.

4) ScaLAPACK

ScaLAPACK is a library of high-performance linear algebra routines for parallel distributed memory machines. ScaLAPACK solves dense and banded linear systems, least squares problems, eigenvalue problems, and singular value problems. The library is installed on every node of the cluster to solve computing intensive mathematical problems in a parallel manner on the cluster using OpenMPI.

Working with OpenMPI programs:

Compiling an OpenMPI program:

mpicc –o <executable_name> <program_name>

mpicc:

mpicc is a convenience wrapper for the underlying C compiler. Translation of an Open MPI program requires the linkage of the Open MPI-specific libraries which may not reside in one of the standard search directories of *ld(1)*. It also often requires the inclusion of header files what may also not be found in a standard location.

mpicc passes its arguments to the underlying C compiler along with the -I, -L and -l options required by Open MPI programs.

https://www.open-mpi.org/doc/v1.8/man1/mpicc.1.php

Executing the program on the cluster:

mpiexec –n X –f <network_config_file> <executable_file>

This will run X copies of *<program>* in your current run-time environment (if running under a supported resource manager, Open MPI's *mpirun* will usually automatically use the corresponding resource manager process starter, as opposed to, for example, *rsh* or *ssh*, which require the use of a host file, or will default to running all X copies on the localhost), scheduling (by default) in a round-robin fashion by CPU slot.

-n X:

Runs X copies of the executable

-f <network_config_file>

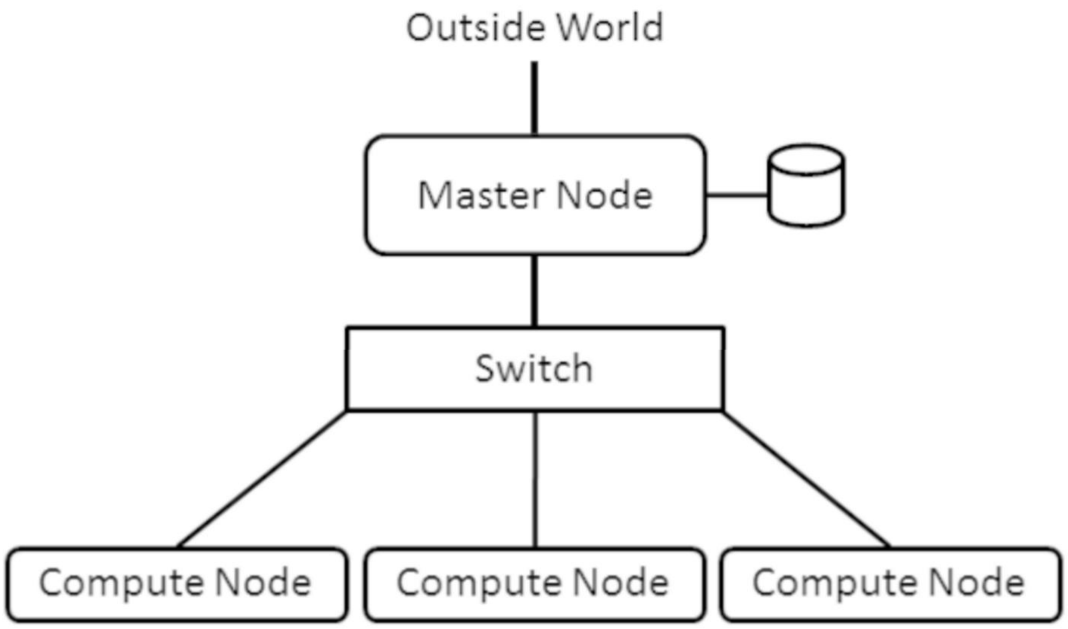
-f takes a normal text file where the user needs to enlist hostnames (or IP addresses) of all the nodes in the cluster

<executable_file>

The executable has to be at the same address on every participating node of the cluster. Typically, a NAS like shared file storage is configured and files are kept in the shared storage.

Design Rationale

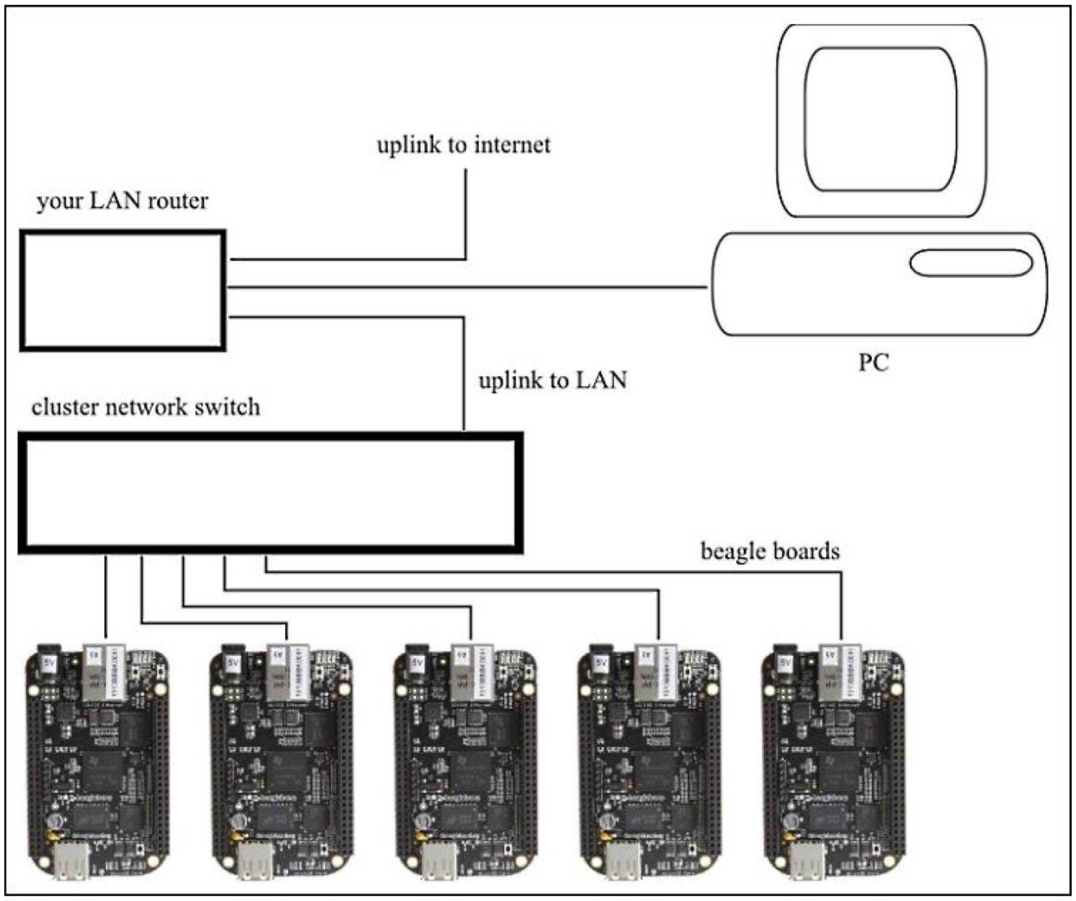
The architecture of a cluster is pretty straightforward. You have some servers (nodes) that serve various roles in a cluster and that are connected by some sort of network. That's all. It's that simple. Typically, the nodes are as similar as possible, but they don't have to be; however, its better if they are similar as possible because it will make your life much easier. The figure below is a simple illustration of the basic architecture.



Generic Cluster Layout

Almost always you have a node that serves the role of a "master node" (sometimes also called a head node). The master node is the "controller" node or "management" node for the cluster. It controls and performs the housekeeping for the cluster and many times is the login node for users to run applications. For smaller clusters, the master node can be used for computation as well as management, but as the cluster grows larger, the master node becomes specialized and is not used for computation.

The typical network topology is a star configuration. Every BeagleBone Black board has its own connection to the switch, and the switch itself is connected to the local area network. The master node is used to bridge between the cluster and the rest of the LAN. Regarding security, we can manage everything with SSH login rules and kernel firewall. Every BBB has its own IP address, and you have to reserve the required amount of IP addresses in your LAN. They do not have to be successive but it is easier if they are. The below diagrams shows the network topology.



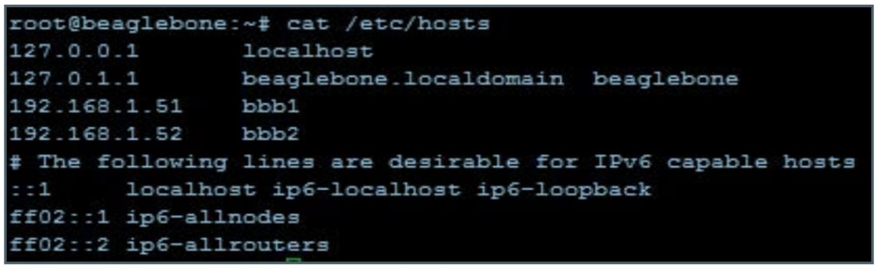
The Network topology

Every BeagleBone has its own IP address. We gave both the BeagleBone hostname “bbb1” and “bbb2”. We configured the network as given below:

Node	IP	Hostname
Master	192.168.1.51	bbb1
Slave	192.168.1.52	bbb2

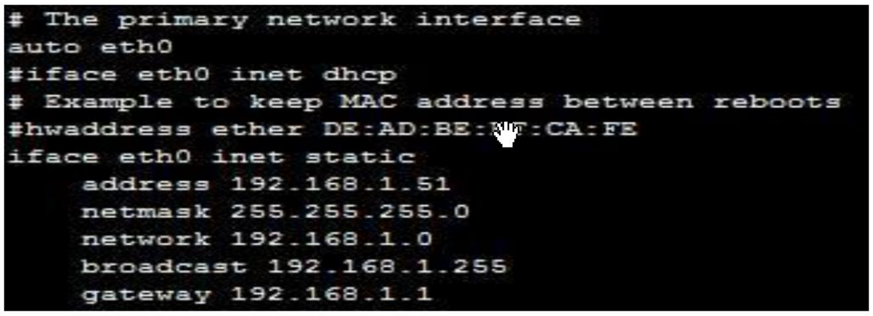
Here are some screenshots depicting how the network is configured

- **/etc/hosts** file of master and slave



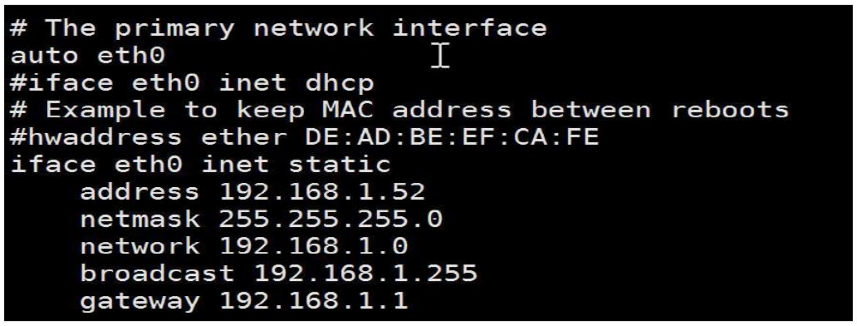
```
root@beaglebone:~# cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       beaglebone.localdomain  beaglebone
192.168.1.51    bbb1
192.168.1.52    bbb2
# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

- **/etc/network/interfaces** of the master



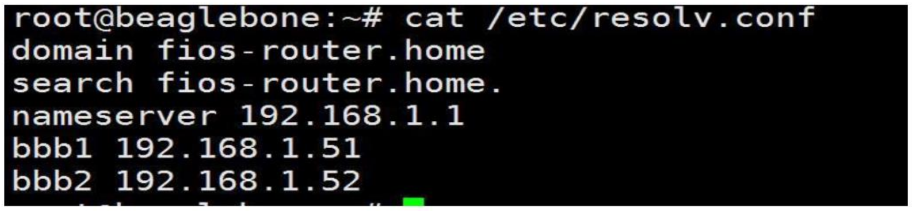
```
# The primary network interface
auto eth0
#iface eth0 inet dhcp
# Example to keep MAC address between reboots
#hwaddress ether DE:AD:BE:EF:CA:FE
iface eth0 inet static
    address 192.168.1.51
    netmask 255.255.255.0
    network 192.168.1.0
    broadcast 192.168.1.255
    gateway 192.168.1.1
```

- **/etc/network/interfaces** of the slave



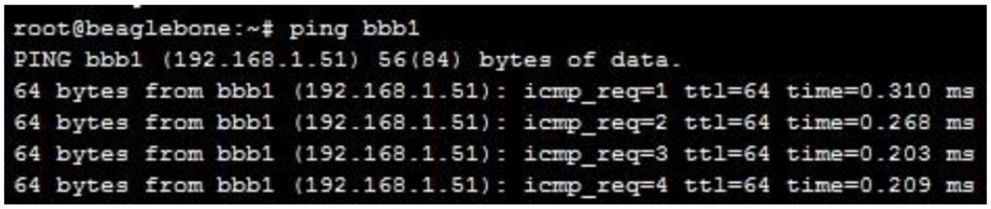
```
# The primary network interface
auto eth0
#iface eth0 inet dhcp
# Example to keep MAC address between reboots
#hwaddress ether DE:AD:BE:EF:CA:FE
iface eth0 inet static
    address 192.168.1.52
    netmask 255.255.255.0
    network 192.168.1.0
    broadcast 192.168.1.255
    gateway 192.168.1.1
```

- **/etc/resolv.conf** of slave



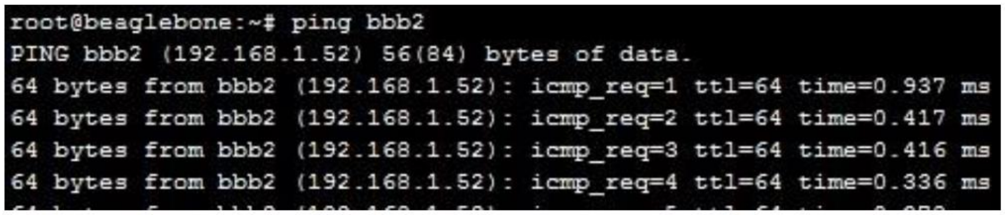
```
root@beaglebone:~# cat /etc/resolv.conf
domain fios-router.home
search fios-router.home.
nameserver 192.168.1.1
bbb1 192.168.1.51
bbb2 192.168.1.52
```

- Master Pinging itself



```
root@beaglebone:~# ping bbb1
PING bbb1 (192.168.1.51) 56(84) bytes of data.
64 bytes from bbb1 (192.168.1.51): icmp_req=1 ttl=64 time=0.310 ms
64 bytes from bbb1 (192.168.1.51): icmp_req=2 ttl=64 time=0.268 ms
64 bytes from bbb1 (192.168.1.51): icmp_req=3 ttl=64 time=0.203 ms
64 bytes from bbb1 (192.168.1.51): icmp_req=4 ttl=64 time=0.209 ms
```

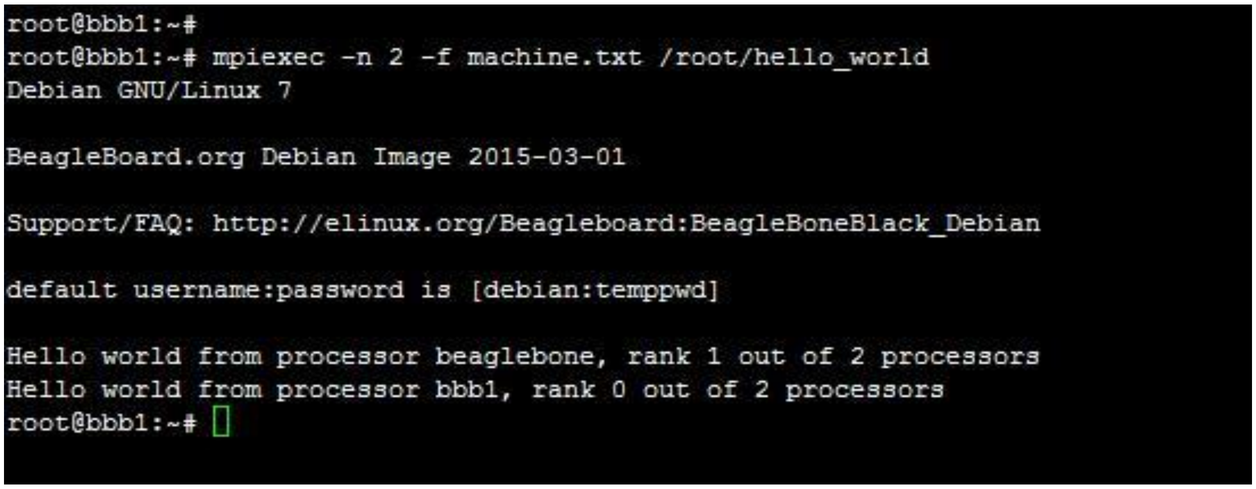
- Master pinging the slave



```
root@beaglebone:~# ping bbb2
PING bbb2 (192.168.1.52) 56(84) bytes of data.
64 bytes from bbb2 (192.168.1.52): icmp_req=1 ttl=64 time=0.937 ms
64 bytes from bbb2 (192.168.1.52): icmp_req=2 ttl=64 time=0.417 ms
64 bytes from bbb2 (192.168.1.52): icmp_req=3 ttl=64 time=0.416 ms
64 bytes from bbb2 (192.168.1.52): icmp_req=4 ttl=64 time=0.336 ms
```

Results and Analysis

We executed a simple hello world program using MPI API's. The program runs simultaneously on both master and slave node. The master node is bbb1 and the salve node is bbb2. The below screenshot shows the output we got. Rank 1 is the slave node and the rank 0 is the slave node.



```
root@bbb1:~#
root@bbb1:~# mpiexec -n 2 -f machine.txt /root/hello_world
Debian GNU/Linux 7

BeagleBoard.org Debian Image 2015-03-01

Support/FAQ: http://elinux.org/Beagleboard:BeagleBoneBlack_Debian

default username:password is [debian:temppwd]

Hello world from processor beaglebone, rank 1 out of 2 processors
Hello world from processor bbb1, rank 0 out of 2 processors
root@bbb1:~# 
```

We performed multiplication of 2 matrices on 2 computing nodes in parallel and profiled the performance and compared it with the multiplication of two matrices of the same order. To our surprise, the multiplication on a single computing node took significantly less time as compared to the parallel counterpart of the experiment.

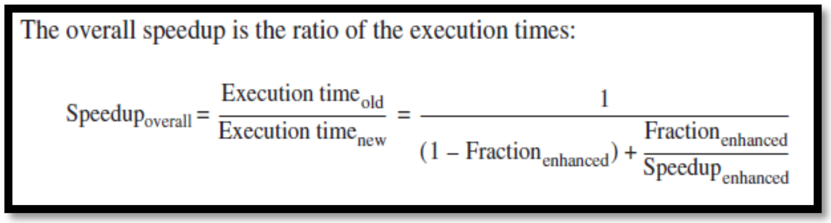
Here is the quantification of the experiments:

Order of the matrices	Parallel Multiplication (Computing on 2 nodes)	Serial Multiplication (Computing on 1 node)	Time $_{parallel}$ / Time $_{serial}$
10 X 10	0.009975	0.0001012	98.3678
50 X 50	0.100459	0.0390455	2.5726
60 x 60	0.102868	0.0618829	1.6623
100 x 100	0.822944	0.0919324	8.9516

Retrospective Analysis of the results:

- Optimal performance for multiplication 60 X 60 matrices
- The performance of parallel multiplication decreases with increases in the order of matrices.
- The main reason is the increased data and message transfers between the nodes in the cluster

- According to Amdahl's formula, the performance will be radically better if the cluster is scaled up and the parallelization is promoted



The overall speedup is the ratio of the execution times:

$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \dfrac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

Better communication medium like fiber optics or infiniband will also enhance the cluster performance drastically.

Future work

Building a Beowulf cluster out of BeagleBone Black embedded boards can be a stepping stone in the journey of low cost supercomputing. This work can be expanded in so many ways:

1) Scaling the cluster
   We, during this course could not scale the cluster well because of scarcity of resources. The real power of cluster computing can be enjoyed with a cluster having dozens of such nodes. As of now, the networking overhead is bogging down the benefits of parallel computing; but with enough processors working together will be experienced.

2) Building a heterogeneous cluster
   A potpourri of heterogeneous small computing nodes will cater the needs of different users. Such a cluster will be immensely powerful as compared to a homogeneous cluster.

3) Performing computing intensive scientific computing
   The real world applications of such a teeny tiny, yet powerful computer are fascinating. The scientific computing world can make extensive use of such a low cost cluster to run their simulations.

4) Building a private cloud on this cluster
   A small cloud built on a powerful cluster can definitely cater to the needs of a sizable organizations. The cloud will be very secure and can be used for small distributed data backups and other services.

5) Parallel graphics computing
   BeagleBone Black has an amazing support for graphics processing. Such a cluster can provide a great computational platform for parallel and distributed image processing, parallel rendering of images, live video streaming etc.

Conclusion/Summary

We successfully built a Beowulf cluster using BeagleBone Black Rev C. We used two BeagleBone's, where one acts as a master node and other as a slave. We successfully established a network with each BeagleBone Black connected to a Local Area network (LAN) in a star configuration. We successfully gave each BeagleBone a static IP address and hostname, "bbb1" for master node and "bbb2" for slave node.

We implemented a simple **hello world** program that runs simultaneously on master and slave node. The screenshot is given in the **Results and Analysis** section. Then, we executed a 60 x 60 matrix multiplication and also noted the computation time. Then we compared that with the result obtained with serial computation. We found that for 60 x 60 the parallel was approximately 1.6 slower than serial. This is because of the communication overhead. This would improve if there were many nodes instead of just two nodes.

References

[1] Building a BeagleBone Black Super Cluster by Andreas Josef Reichel; Publisher: PACKT Publishing

[2] https://www.youtube.com/watch?v=i_r3z1jYHAc

[3] http://wallfloweropen.com/?project=beaglebone-black-cluster-demo-build

[4] http://coen.boisestate.edu/ece/files/2013/05/Creating.a.Raspberry.Pi-Based.Beowulf.Cluster_v2.pdf

[5] http://www.admin-magazine.com/HPC/Articles/Building-an-HPC-Cluster

[6] http://likemagicappears.com/projects/raspberry-pi-cluster/

[7] https://www.linux.com/blog/building-compute-cluster-beaglebone-black