Proyecto de Grado

Instalación, Implementación y Uso de las librerías Boost y su aplicación para el centro de computación científica Apolo

Autor:

Sergio Andrés Monsalve–Castañeda

Código: 200410061010

smonsal3@eafit.edu.co

Asesor:

Juan Guillermo Lalinde–Pulido, Ph.D

jlalinde@eafit.edu.co

2619500 ext 9588

27 de noviembre de 2013

Instalación, Implementación y Uso de las librerías Boost y su aplicación para el centro de computación científica Apolo

Sergio Andrés Monsalve–Castañeda
smonsal3@eafit.edu.co

Asesor:
Juan Guillermo Lalind–Pulido, Ph.D

Ingeniería de Sistemas
Departamento de Informática y Sistemas
Escuela de Ingeniería
Universidad EAFIT
Medellín, Colombia.
2013

Índice general

Índice de figuras

Resumen

Este trabajo de grado tiene como objetivo consolidar del conocimiento del Ingeniero de sistemas a través de un proyecto que requiera el uso de diferentes áreas de su conocimiento tales como; telemática, organización de computadores, estructuras de datos y algoritmos, lenguajes de programación entre otras, de esta manera que sirva de referencia para quienes estén interesados en el tema.

Este documento esta compuesto por un marco de referencia que introduce a los diferentes conceptos que se interrelacionan en este proyecto, el procedimiento de creación de un ambiente de pruebas que simule un clúster de computo científico para la computación en paralelo, una guía para la instalación de las librerías de Boost y las instrucciones para el uso y la ejecución de programas que hagan uso de las librerías paralelas para grafos de Boost.[1]

[1]Para una versión electronica de este documento, actualizaciones y proyectos relacionados, visitar: `https://github.com/smonsalve/tesis.git`

Agradecimientos

Quiero agradecer al profesor Juan Guillermo Lalinde quien desde el inicio de mi carrera me inspiró y apoyó en grandes retos académicos, por su paciencia y confianza a la hora de explicarme cada una de las múltiples dudas que surgían en el proceso.

A Juan David Pineda por toda su paciencia para explicarme el funcionamiento de Apolo.

También deseo manifestarle mi gratitud a John Jairo Silva, Alejandro Gómez, Mateo Gómez, Jaime Pérez y John Mario Gutiérrez por sus múltiples explicaciones y colaboraciones.

Por último agradecer el apoyo que he tenido de mi madre y hermano, que son inspiración y ejemplo para el día a día.

Capítulo 1

Introducción

Dentro de los problemas que se estudian en la Ingeniería de Sistemas y sus aplicaciones existe una gran cantidad de problemas que se han planteado para resolver, continuamente incrementa esta cantidad y aun mas complejos, con mas interacciones y mas datos para analizar.

Entre estos existen algunos con una complejidad alta, donde de la solución de los mismos esta condicionada a mucho tiempo de computo, haciendo de tales investigaciones, trabajos y simulaciones algo inviable por su prolongada ejecución. Con el crecimiento de la capacidad computacional y los algoritmos paralelos ya es posible abordar algunos de los problemas que antes hubieran tomado mucho tiempo o simplemente no hubiese sido posible resolver, pasando de semanas, meses y años de computo, a solo semanas o incluso días.

De este conjunto de problemas, existen un subconjunto particular los cuales pueden ser modelados mediante la teoría de grafos, una abstracción que ofrece las ciencias de la computación para la solución de un conjunto determinado de problemas que cumplen con ciertas características, y aunque los algoritmos para grafos tienen una complejidad alta, existe la posibilidad de utilizar algoritmos paralelos.

En este proyecto se pretende introducir al lector en el amplio mundo de la computación de alto rendimiento, mediante el procedimiento de instalación y ejecución en un am-

biente de pruebas que pretende simular el funcionamiento del centro de computación científica "Apolo", el supercomputador de la Universiad EAFIT. La configuración y uso de la "infraestructura" básica para que un programador, utilizando las librerías BOOST para C++, pueda desarrollar aplicaciones que aprovechen su capacidad computacional.

1.1. Justificación

Dada la configuración de Sistema necesaria para la realización de este proyecto, los procedimientos y productos aquí planteados pueden ser de utilidad en diferentes campos y aplicaciones, entre ellas destacan las siguientes:

- Personas que requieran una introducción a la configuración de un sistema de pruebas para la computación paralela, en particular una simulación de la arquitectura y sistema utilizado por el supercomputador de la universidad EAFIT.

- En el mundo académico también puede ser importante para Investigación, Estudio y análisis de la Arquitectura de un clúster a través de un sistema virtualizado, con diferentes configuraciones de redes implementadas en el clúster, para analizar rendimiento, velocidad, eficiencia y diferentes características importantes.

- Análisis y practicas de telemática y comunicación en el clúster.

- Toda aquella persona que necesite un ambiente de pruebas de un clúster de computación de tipo HPC y HTC.

- Quien necesite probar el uso de algoritmos con MPI, Boost u otro ambiente de computación en paralelo.

- Algoritmos para el procesamiento de Grafos en paralelo (tales como PBGL).

1.2. Aplicaciones

Como posibilidades para la aplicación de este proyecto se mencionan las siguientes áreas:

- En la Industria: Análisis de malla vial de la ciudad a través de un grafo, análisis de rutas, vías alternas en caso de cierre de vías, camino más corto entre dos puntos, optimización de recorrido para ambulancias, bomberos, policías o atención de emergencias, análisis de la estructura de distribución de mallas eléctricas, análisis de la estructura de distribución de la infraestructura hídricas.

- Para Educación e Investigación: Simulaciones de mecánica de materiales, mecánica de fluidos, Ingeniería Civil, optimización de redes, patrones de distribución de la información. Aplicación en la enseñanza en materias de algoritmia y los algoritmos paralelizados, seguridad informática, criptoanálisis, genómica, matemática, física y muchos mas.

Los usuarios potenciales de este proyecto son: comunidad Científica y Académica, interesados en el desarrollo de software paralelo, simulaciones de diferentes tipos, investigación en diferentes campos que requieran la ejecución de software en paralelo, simulaciones o procesos de la industria, estudiantes de pregrado de instituciones académicas, estudiantes de posgrado de instituciones académicas.

1.3. Objetivos

- Generar documentación sobre el manejo de las librerías de Boost.
- Construir un manual para la instalación de las librerías de Boost.
- Correr códigos de ejemplo que hagan uso de las librerías Paralelas de Boost para el recorrido de grafos.

1.4. Alcance

- Introducción y descripción de las librerías de Boost.
- Introducción a las librerías de utilización de grafos a ser instaladas.
- Introducción al manejo de grafos con Boost (representación conceptual).

1.5. Productos

- Manual de creación de un ambiente de pruebas para la instalación de las librerías de Boost.
- Maquina Virtual para la creacion de un ambiente de pruebas.
- Manual de instalación de librerías Para Boost.
- Manual de ejecución para el uso de las librerías de Boost para el grafos en paralelo.

Capítulo 2

Marco de Referencia

En este capitulo se brinda una explicación de Apolo, su arquitectura, sus las especificaciones técnicas y como este soporta la computación paralela.

Se brinda también una descripción de lo que es un grafo y su importancia, para modelarlos se utilizan las librerías paralelas de Boost, por lo que se explica que es Boost, las cuales están escritas en C++ y utilizan MPI como mecanismo de comunicación.

2.1. Apolo

Apolo es el Centro de Computación Científica de la Universidad EAFIT, el cual tiene por objetivo principal apoyar la investigación y el desarrollo del país y de la región.

Para cumplir con tal objetivo, presta sus servicios para la investigación, tanto en el mundo académico como en la industria, mediante la simulaciones computacionales en un equipo de altos recursos, utilizando técnicas de computación de alto rendimiento, también conocida como computación paralela.

Apolo se compone de un equipo físico (todos el hardware y la infraestructura necesaria de un clúster para el computo en paralelo) y un equipo humano capacitado tanto para la administración y mantenimiento del software y hardware, como para la capacitación

de nuevos usuarios y el apoyo a los usuarios existentes.

Especificaciones Técnicas

La infraestructura física de Apolo esta compuesta por:

- 40 Servidores de 4 núcleos: Blade HP ProLiant DL140 Generation 3 con 8 GB de RAM cada uno.[1]

Figura 2.1:

- 6 servidores de 8 núcleos: Blade HP Proliant BL460c G1 con 8 GB de RAM cada uno.[2]

Figura 2.2:

- 40 servidores de 8 núcleos: Blade Dell PowerEdge 1950 con 16 GB de RAM cada uno.[3]

Figura 2.3:

Para un total de 86 servidores y 528 núcleos, con 2x TB de almacenamiento.

Adicionalmente cuenta con:

- Red de 1 GBps.
- Sistema eléctrico con 5 modulos de UPS de respaldo para 18 minutos de autonomia.
- Aire acondicionado especializado para la refrigeración de los equipos.
- Equipo para el control de incendios.

2.2. Computación Paralela

La computación paralela es el uso de múltiples recursos computacionales para resolver un problema o una necesidad de computo en particular. La computación paralela surge como respuesta ante la necesidad de incrementar los recursos, sea en procesador, memoria y así mejorar el tiempo de respuesta para problemas con alta complejidad computacional o alto volumen de análisis de datos.

El paradigma computacional tradicional ha sido de computación serial, donde una tarea es dividida en una serie finita de instrucciones que son ejecutada de forma secuencial, donde una sola instrucción es ejecutada en un momento dado.

La computación paralela rompe el paradigma anterior, buscando que en un momento dado se puedan ejecutar varias instrucciones, utilizando múltiples procesadores y una entidad que orqueste los mismos.[4] [5]

2.3. Grafos

Los grafos son abstracciones matemáticas que son útiles a la hora de resolver muchos tipos de problemas en las ciencias de la computación.

Un grafo (G) es una dupla compuesta por dos elementos, el primero un conjunto de nodos o vértices (representado por la letra V de vértices) y el segundo un conjunto de aristas o arcos (representado por la letra E por Edges o arcos en ingles) , donde las aristas forman relaciones entre los diferentes vértices.[6]

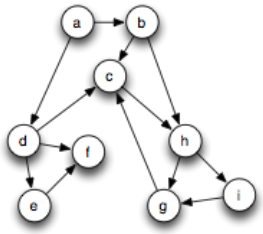

Figura 2.4: (tomada de [7])

Cuando los grafos son bidireccionales se les denomina no dirigidos y por el contrario si la dirección de cada relación entre los nodos es especifica se le denomina dirigidos. [8]

La aplicación de la teoría de grafos permite un estudio de relaciones entre sus nodos, utilizando sus propiedades para el análisis de la información que estos contienen a través de algoritmos que recorren los nodos de diferentes maneras.

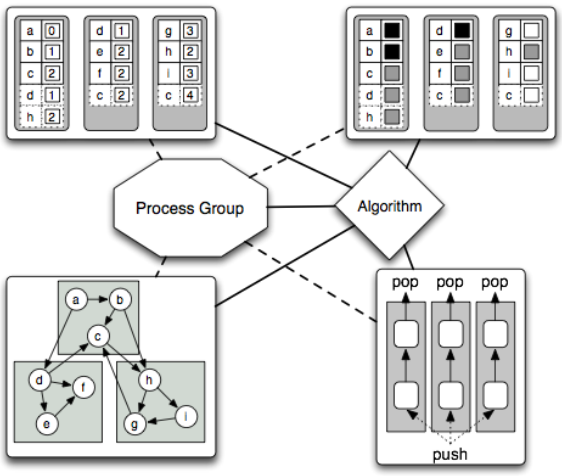

Figura 2.5: (tomada de [7])

Los grafos pueden ser representados de diferentes formas, una de esas formas es con el uso de matrices, optimo con grafos densos; donde el numero de arcos se aproxima o supera el numero de vértices al cuadrado:

$$|E| \approx |V|^2$$

También pueden ser representados mediante listas de adyacencia, lo cual facilita su representación para grafos distribuidos, donde un subconjunto de relaciones es conocida por quien procesa la información.

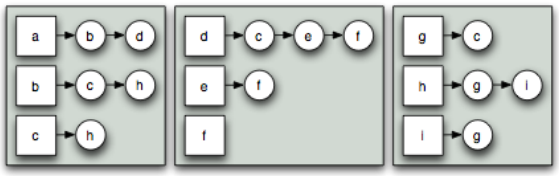

Figura 2.6: (tomada de [7])

Para este proyecto estaremos utilizando el algoritmo de búsqueda por amplitud que tiene como ejemplo la librería paralela para grafos de Boost, que hace uso del esquema distribuido.

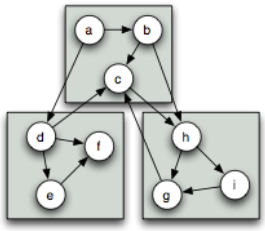

Figura 2.7: (tomada de [7])

2.4. C++

C++ es un lenguaje de programación de propósitos generales, que soporta abstracción de datos, es multiparadigma (programación estructurada, programación genérica[9] y programación orientada a objetos).[10]

C++ es reconocido por su buen uso de los recursos computacionales, por lo cual siendo un lenguaje altamente eficiente, muy veloz y potente, resulta optimo para la computación de alto desempeño. Las librerías de Boost hacen uso extensivo de la programación genérica y la metaprogramación de C++.

2.5. MPI

MPI es una interfaz de paso de mensajes individuales, ya sea punto a punto o de manera coordinada como grupo diseñado para permitir la comunicación entre varias unidades de computo, este surgió como un mecanismo para resolver la necesidad de permitir que múltiples procesos en paralelo trabajen de manera concurrente hacia la solución de un problema en particular. A diferencia de la comunicación en sistemas multihilos o de memoria compartida, MPI puede interactuar con diferentes sistemas operativos y arquitecturas subyacentes. [11] [12]

2.6. Boost

Boost son una serie de librerías de C++, implementadas y mejoradas por la comunidad cuya base es el buen funcionamiento con las librerías estándar de C++ (C++ STL) y la eventual estandarización de las mismas, mediante el mejoramiento progresivo de estas.[13]

Boost dentro de sus librerías tiene implementaciónes de programación genérica para grafos y adicionalmente una librería de algoritmos paralelos para grafos. [6]

2.7. Parallel Boost Graph Library

La PBGL es una librería de Boost para Grafos de Boost para computación paralela y distribuida. Esta ofrece algoritmos para computación en paralelo y distribuida, conservando las mismas interfaces que la librería secuencia para grafos de Boost.[13]

Capítulo 3

Ambiente de Pruebas

En este capitulo se presenta el procedimientos para la virtualización de un Ambiente de Pruebas que simula el funcionamiento de un centro de computación científica con la arquitectura de Apolo.

Este proceso es importante para familizarse con el ambiente y los procedimientos en un ambiente seguro, con la posilidad de interacturar con los componentes de manera segura sin afectar de ninguna manera un centro de computo real.

La siguiente es la descripción de los pasos a seguir para la creación de un ambiente de pruebas paralelo virtualizado que simula el clúster de Apolo.[1]

3.1. Requerimientos Mínimos

- Verifique que su computador puede virtualizar: Ingrese a la BIOS y verifique que esta activada la opción de Virtualización de Hardware.
- 3Gb de RAM (o mas) disponibles para virtualizar.[2]

[1] Estas instrucciones fueron probadas en un ambiente de Linux Fedora 18.
[2] 1024 RAM para cada Maquina Virtual

3.2. Configuración del Nodo Maestro

1. Descargue e Instale VirtualBox Manager junto con el correspondiente "Extension Pack" de la siguiente dirección: `https://www.virtualbox.org/wiki/Downloads` [3]

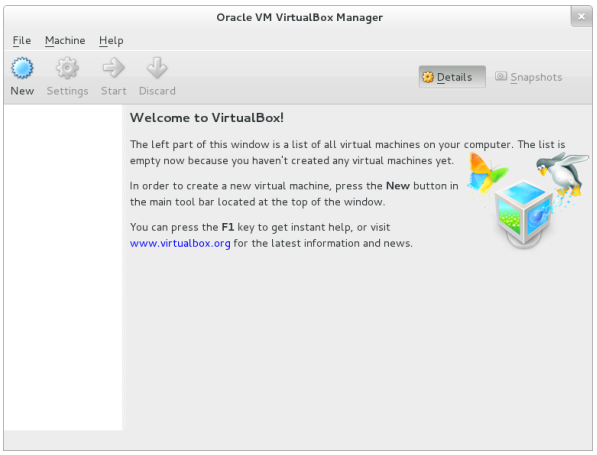

Figura 3.1: Ventana de Virtualbox después de la instalación

2. Descargar la imagen del Master del siguiente enlace: `http://goo.gl/8eTJOr`

3. Una vez instalado VirtualBox en su computador, proceda a instalar el Extensión Pack:

[3]Tanto la versión de VirtualBox como la versión del "Extension Pack" deben coincidir

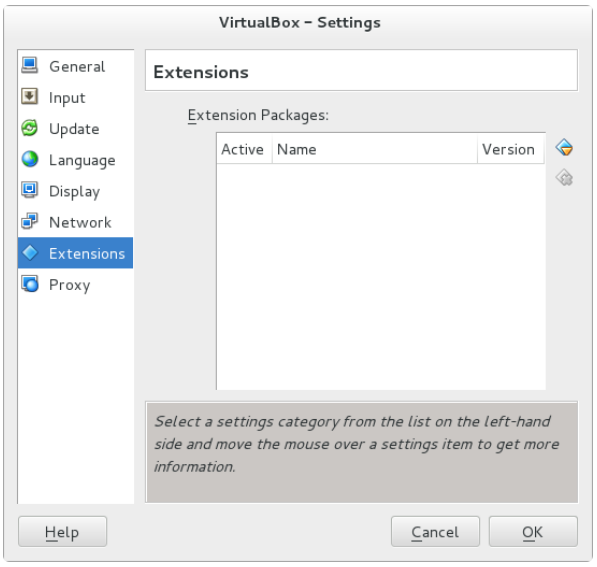

Figura 3.2: VirtualBox antes de instalar el "Extension Pack"

- En VirtualBox acceda al menú: Archivo → Preferencias → Sección Extensiones → Proceda a instalar el "Extension Pack".

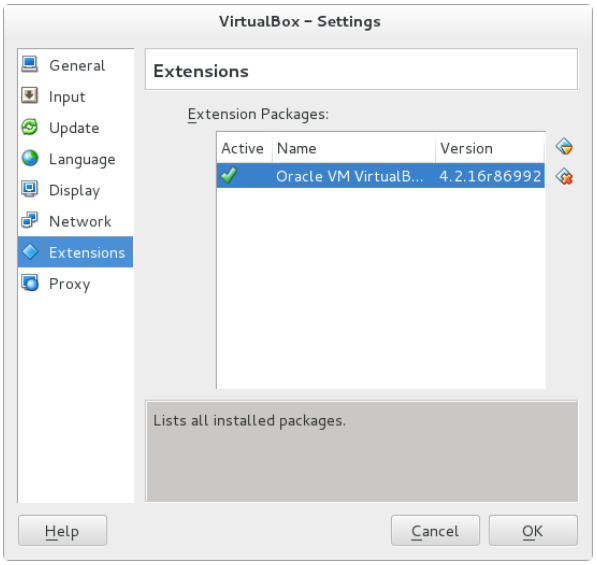

Figura 3.3: Despues de Instalar el "Extension Pack"

- Vaya a la sección Red → Adicione una red Sólo Anfitrión. Asegúrese en caso de tener o crear otra red "Sólo anfitrión" (Host Only) que todos los nodos compartan la misma red.

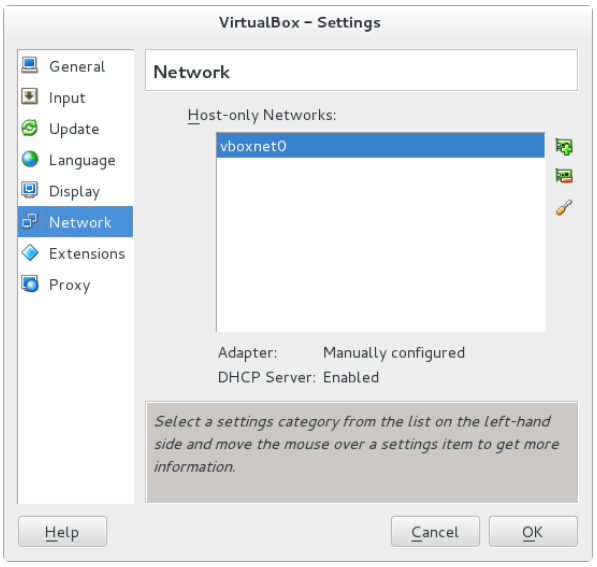

Figura 3.4: Red Sólo Anfitrion “vboxnet0”

- Haga clic en aceptar para finalizar la operación.

4. En VirtualBox, importe desde el Menú → Importar Appliance. Deje la configuración por defecto y **no** reinicialice la dirección MAC.

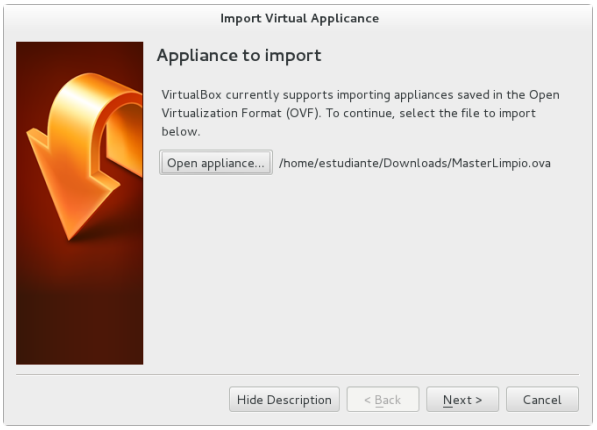

Figura 3.5: Importar Maquina Virtual

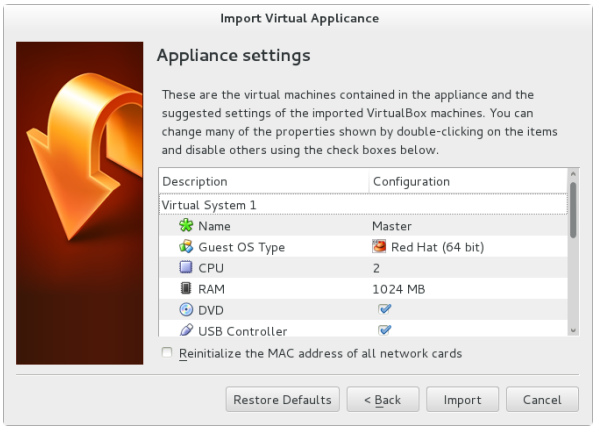

Figura 3.6: Opciones de configuración de la Maquina Virtual

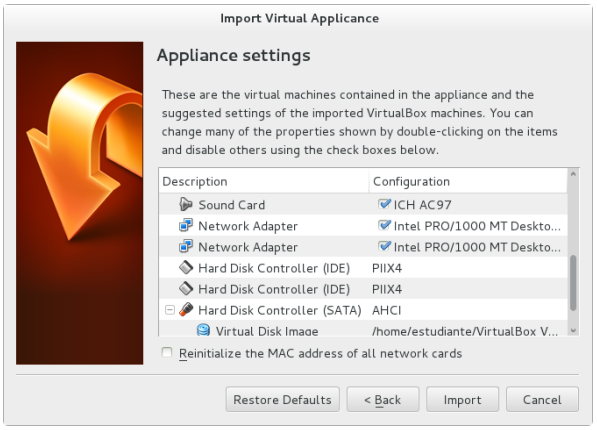

Figura 3.7: Más opciones de configuración de la Maquina Virtual

5. Seleccione la máquina virtual llamada “Master” y vaya a la configuración y revise la siguiente configuración en esta:

   - En la sección Sistema, pestaña Procesador, debe tener dos procesadores y habilitado PAE/NX.

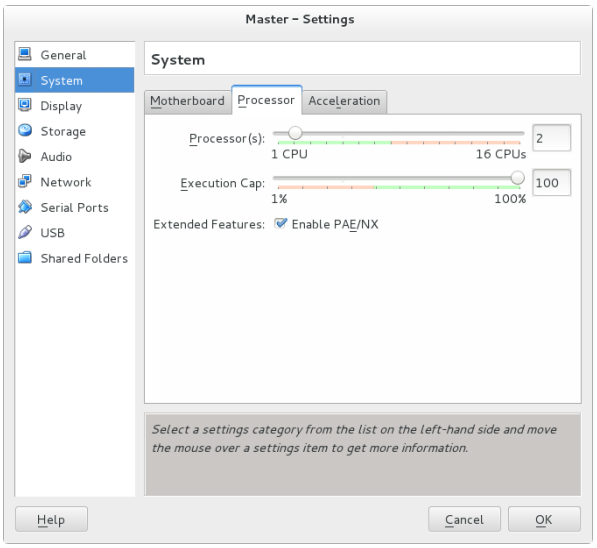

Figura 3.8: Procesadores e Inicio por PAE/NX

- En la sección Red, pestaña Adaptador 1, deberá estar configurado como NAT.

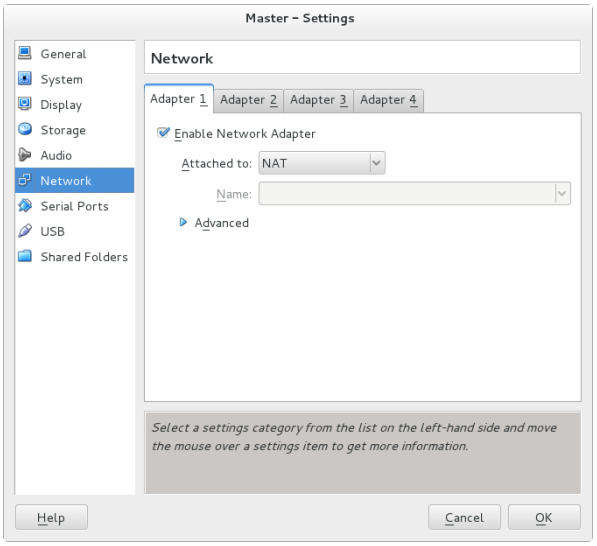

Figura 3.9: Configuración Red Adaptador 1: NAT

- En la pestaña Adaptador 2 deberá estar en Adaptador Sólo–Anfitrión y el nombre deberá ser `vboxnet0`[4].

[4]Tenga en cuenta que este adaptador se llama `vboxnet0` en Linux, en Windows tendrá otro nombre,

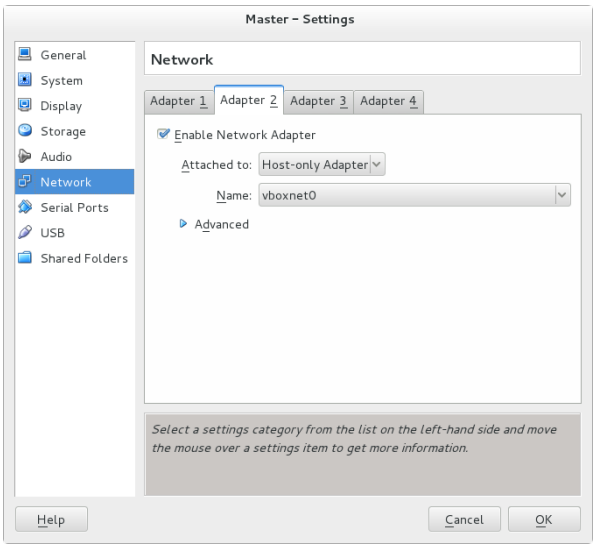

Figura 3.10: Red Host Only (Sólo Anfitrión)

- Acepte todos los cambios.

6. Seleccione la máquina Master e iníciela.

7. Una vez iniciada la máquina virtual del Master proceda a crear una nueva máquina virtual como Nodo Trabajador a partir de los siguientes pasos:

   - Haga clic en Crear Nueva Máquina.
   - El nombre será `compute-0-0`.
   - El tipo será Linux.
   - La versión será Red Hat de 64 bits.

lo más importante es que sea la misma interfaz de red de Sólo Anfitrión, ya que sino dará lugar al problema de reasignación de interfaces de red dentro del nodo Master, en otras palabras, asignará las interfaces eth2 y eth3 en vez de asignar eth0 y eth1 a la NAT y a la Sólo Anfitrión respectivamente.

Figura 3.11: Opciones para el compute-0-0

- Clic en siguiente.
- Asigne 1024 Megabytes de Memoria RAM[5].
- Cree el disco duro con 30 gigas de espacio, recuerde que esto se asignará dinámicamente. El tipo de disco duró será VDI y dinámicamente asignado.

Figura 3.12: Crear un Nuevo Disco Virtual (VHD)

[5]La cantidad de memoria RAM asignada será proporcional a la que tenga el sistema en el cual están las máquinas virtuales. Igual con el disco duro y los procesadores, sin embargo, para efectos de ver la paralelización en memoria compartida se recomienda tener 2 procesadores por máquina

Figura 3.13: Tipo de disco a crear

Figura 3.14: Opciones para Disco Dinámico o Estático

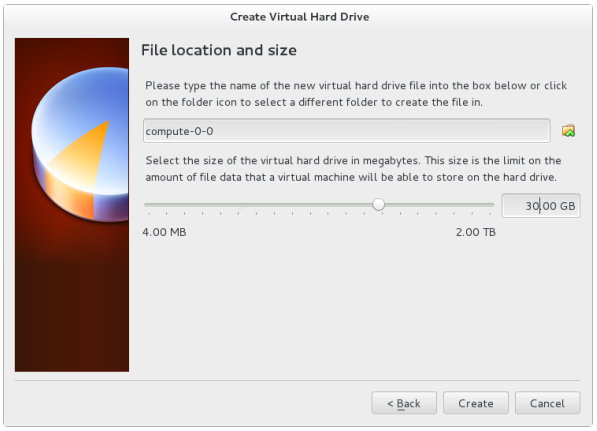

Figura 3.15: Tamaño del Disco

8. Una vez creado el nodo trabajador se procede a configurarlo a partir de los siguientes pasos:

   - Señalar la máquina `compute-0-0` y dar clic en Configurar.
   - En la pestaña Sistema, asegúrese de que tenga la cantidad de memoria RAM correcta
   - Deshabilite el `floppy disk`
   - Habilite la red como dipositivo de `booteo` y además súbalo como primer dispositivo, sólo deberá quedar la tarjeta de red como primera opción y como segunda el disco duro de la máquina virtual, de esta manera nos aseguramos que esta máquina virtual pueda instalarse automáticamente de manera desatendida, por esta razón el clúster es escalable.

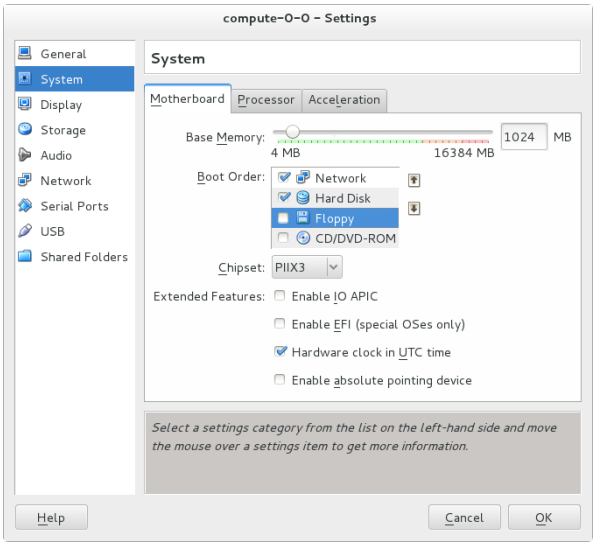

Figura 3.16: Opciones del Nodo

   - En la pestaña Procesador asegúrese de que hay dos procesadores y está habilitado PAE/NX.

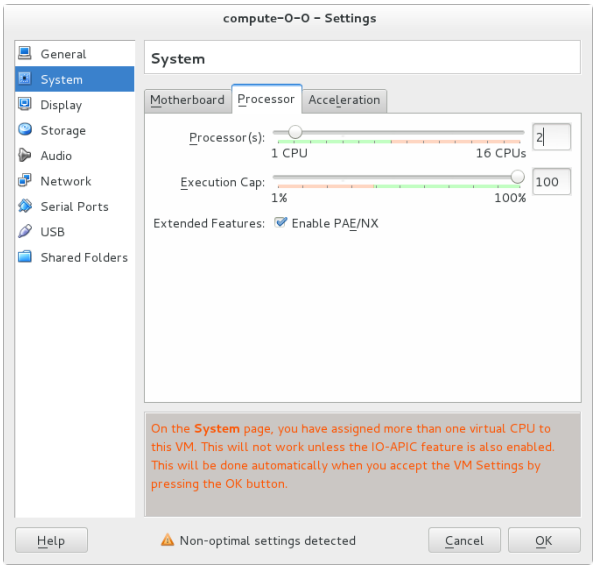

Figura 3.17: Opciones Procesador para el Nodo

- En la sección Red deberá configurar sólo la primera interfaz de red y deberá estar configurada como Sólo–Anfitrión y deberá tener `vboxnet0`.

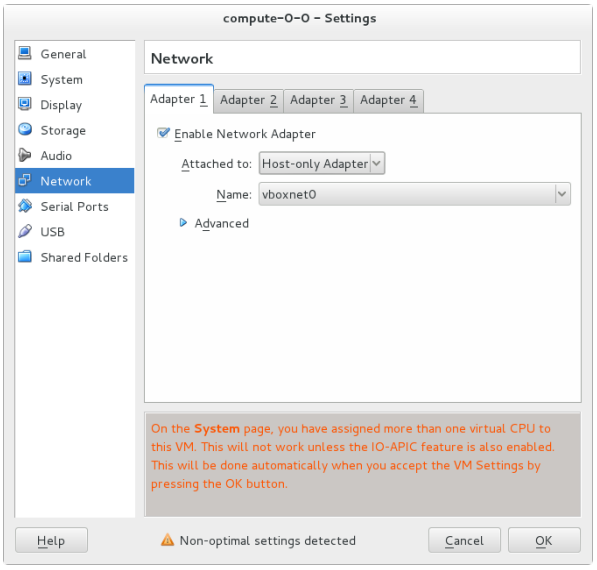

Figura 3.18: Opciones de Red del Nodo

- Acepte los cambios.

9. Repita los pasos para crear otro nodo si lo considera necesario para crear el `compute-0-1`, siempre y cuando la computadora que usted tiene soporte una tercera máquina virtual ejecutandose al tiempo que el Nodo Master y el `compute-0-0`

10. Ingrese en el Nodo Master como usuario `root` y la contraseña es `apolito123!`

11. Abra una consola (La combinación de teclas Alt+F2 permite ejecutar una aplicación escribiendo su nombre)

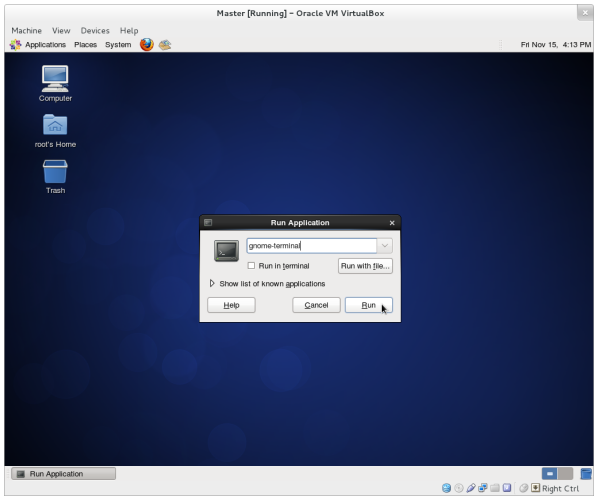

Figura 3.19: Abrir Terminal

12. Para agregar las interfaces de los nodos ejecute el siguiente comando:

```
$ insert-ethers
```

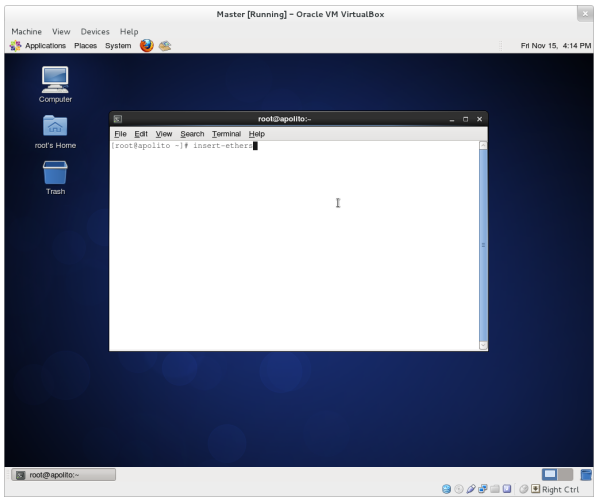

Figura 3.20: Captura de los nodos por parte del Master

13. Escoja `Compute`. Aparecerá una interfaz de consola mostrándole los nodos agregados en la medida que se les vaya asignando IP por medio de DHCP y una imagen de Linux para instalar con PXE.

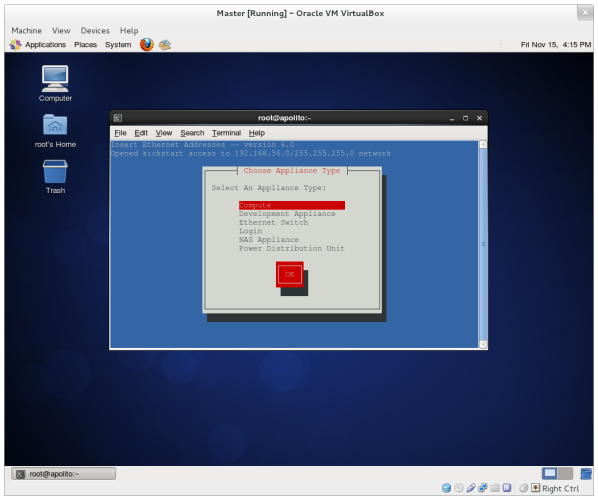

Figura 3.21: inserción de los Nodos tipo “Compute”

14. Encienda de uno en uno los Nodos trabajadores que haya creado, empezando por el `compute-0-0` y así sucesivamente. Observará que aparece en la interfaz `insert-ethers` la dirección MAC del Nodo Trabajador, se le asignará el

nombre compute-0-0 y si aparece un símbolo de asterisco es porque recibió exitosamente la imagen para la instalación de Linux.

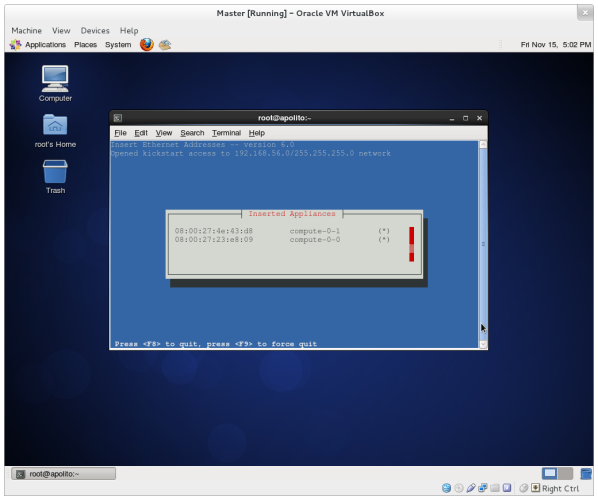

Figura 3.22: Nodos ya reconocidos por el master

15. Una vez de que los nodos se termine de instalar automáticamente salga de la interfaz de `insert-ethers` en el Master con la tecla F8.

16. Ingrese al directorio especificado con el siguiente comando:

    ```
    $ cd /export/apps/installers
    ```

17. Descargue el instalador del comando `htop` con la siguiente instrucción:

    ```
    $ wget http://goo.gl/TDWExw
    ```

18. Instale `htop` en el nodo Master con la siguiente instrucción:

    ```
    $ rpm -ivh htop*.rpm}
    ```

19. Ahora instale masivamente `htop` en el resto del clúster con el siguiente comando:

    ```
    $  rocks run host ``rpm -ivh /share/apps/installers/htop*.rpm''}
    ```

20. El cluster está completo.

3.3. Configuración de los Nodos Esclavos

1. Una vez que se tiene el nodo Master funcionando y por lo menos un nodo trabajador como el `compute-0-0` se procede a realizar los siguientes pasos de configuración:

   - Adicione un usuario sin privilegios con los siguientes comandos[6]:

     - Para crear un usuario sin privilegios:

       ```
       $ adduser smonsalve
       ```

     - Para cambiar la contraseña del usuario:

       ```
       $ passwd smonsalve
       ```

     - Para sincronizar el usuario creado en todo el cluster, este usuarió estará creado tanto en el nodo master como en los nodos trabajadores.

       ```
       $ rocks sync users
       ```

   - Para cambiar del usuario `root` al recient creado usuario sin privilegios:

     ```
     $ su - smonsalve
     ```

   - Se le harán algunas preguntas de contraseñas, puede llenarlas o déjelas vacías y continuar presionando la tecla `enter` varias veces:

3.4. Apagar el Clúster

Cada vez que se vaya a apagar el ambiente de Pruebas es necesario realizarlo con el siguiente procedimiento para evitar dañar la configuración de los nodos.

- Desde una consola en el master ejecutar el siguiente comando que le indicara los respectivos nodos que se apaguen de manera adecuada.

[6]En adelante el usuario de ejemplo será smonsalve, pero usted podrá asignar el nombre de usuario que desee

```
$ rocks run host poweroff
```

- Desde la misma consola para apagar el master:

  ```
  $ poweroff
  ```

3.5. Posibles Problemas

- Los nodos de trabajo no inician Correctamente: Es posible que estos hayan sido apagados de manera incorrecta.
  - Remueva los nodos con el siguiente comando:

    ```
    $ insert-ethers --remove compute-0-0
    $ insert-ethers --remove compute-0-1
    ```

  - Apagargregue de nuevo los nodos:

    ```
    $ insert-ehters
    ```

Capítulo 4

Instalación

Este procedimiento de instalación esta basado en la instalación de Repast HPC[14] que hace uso de Boost y MPI.

Para el proceso de instalación del software necesario, estando en el nodo maestro con el usario root procedemos a realizar las siguientes instrucciones:

- Ir a la carpeta de instalación compartida en el cluster:

  ```
  $ cd /share/Apps/installers/
  ```

- Se debe descargar InstaladorTesis.tar.gz de la dirección:

  ```
  $ wget   http://goo.gl/kSmFQY
  ```

- Se procede a descomprimirlo en la carpeta compartida del cluster:

  ```
  $ tar xvf InstaladorTesis.tar.gz
  ```

  En este archivo se encuentran los siguientes archivos:

  - Boost 1.54
  - mpich 2 Version 1.4.1
  - Scripts de instalación

4.1. Instalación MPI

Para instalar mpich nos movemos a la carpeta de instalación:

```
$ cd /INSTALLATION
```

y procedemos a ejecutar el script de instalación con mpich como argumento de la siguiente manera:

```
$ ./install.sh mpich
```

el cual ejecutara la siguiente secuencia de instrucciónes:

- Define una variable como el directorio base donde se encuentran los archivos.
- Verfica que no se haya corrido anteriormente el scprit o instalado mpich.
- Verfica que se encuentre el archivo *.tar.gz para instalar mpich.
- Descomprime el archivo.
- Se cambia a la carpeta descomprimida.
- Configura con los argumentos de instalación.
- Hace make y make install.
- Exporta el path en el que quedo instalado.

Listing 4.1: Script de Instalación de Mpich

```
#!/bin/bash

cd ..
BASE_DIR=$PWD/ext/
cd INSTALLATION

MPI_COMPILER_INVOCATION=$BASE_DIR/MPICH/bin/mpicxx

if [[ $1 == *mpich* ]]
then
  if [ -e $BASE_DIR/MPICH ]
  then
    echo "A directory named $BASE_DIR/MPICH already exists; you
       must delete it before it can be rebuilt."
    exit
  fi
  if [ ! -e mpich2-1.4.1p1.tar.gz ]
  then
    echo "MPICH tar file (mpich2-1.4.1p1.tar.gz) not found; you
       must download this and put it in this directory to
       continue."
    exit
  fi
  tar -xvf mpich2-1.4.1p1.tar.gz
  cd mpich2-1.4.1p1
  ./configure --prefix=$BASE_DIR/MPICH --disable-f77 --disable-fc
  make
  make install
  cd ..
  export PATH=$BASE_DIR/MPICH/bin/:$PATH
fi
```

4.2. Instalación de Boost

Para instalar boost nos movemos a la carpeta de instalación:

```
$ cd /INSTALLATION
```

y procedemos a ejecutar el script de instalación con boost como argumento de la siguiente manera:

```
$ ./install.sh boost
```

el cual ejecutara la siguiente secuencia de instrucciónes:

- Define una variable como el directorio base donde se encuentran los archivos.
- Define una variable con la dirección que tiene mpich en el sistema.
- Verfica que no se haya instalado o corrido anteriormente boost.
- Verfica que se encuentre el archivo *.tar.gz para instalar boost.
- Descomprime el archivo.
- Se cambia a la carpeta descomprimida.
- Copia los archivos descomprimidos a la carpeta de boost.
- Configura con los argumentos de instalación y las banderas apropiadas (librerias a compilar).
- Corre el script de compilación propio de Boost.[1]

[1]La libreria que hace uso de Grafos en paralelo necesita de las siguientes librerias compiladas: serialization, mpi, graph, graph parallel, regex.

Listing 4.2: Script de Instalación de Boost

```
#!/bin/bash

cd ..
BASE_DIR=$PWD/ext/
cd INSTALLATION

MPI_COMPILER_INVOCATION=$BASE_DIR/MPICH/bin/mpicxx


if [[ $1 == *boost* ]]
then
if [ -e $BASE_DIR/Boost ]
  then
    echo "A directory named $BASE_DIR/Boost already exists; you
       must delete it before it can be rebuilt."
    exit
  fi
  if [ ! -e boost_1_54_0.tar.gz ]
  then
    echo "Boost tar file (boost_1_54_0.tar.gz) not found; you must
        download this and put it in this directory to continue."
    exit
  fi
  tar -xvf boost_1_54_0.tar.gz
  mkdir -p $BASE_DIR/Boost/Boost_1.54/include
  cp -r ./boost_1_54_0/boost $BASE_DIR/Boost/Boost_1.54/include
  cd boost_1_54_0
  ./bootstrap.sh --prefix=$BASE_DIR/Boost/Boost_1.54/ --with-
      libraries=chrono,context,coroutine,date_time,exception,
      filesystem,graph,graph_parallel,iostreams,locale,log,math,
      mpi,program_options,python,random,regex,serialization,
      signals,system,test,thread,timer,wave
  echo "using mpi : $MPI_COMPILER_INVOCATION ;" >>./tools/build/v2
      /user-config.jam
  ./b2 --layout=tagged link=static variant=release threading=multi
       runtime-link=static stage install
```

```
29   cd ..
30 fi
```

De esta manera se completa el proceso de instalación.

Capítulo 5

Ejecución

Contando ya con el ambiente necesario para la ejecución del código a paralelizar se procede a realizar los siguientes pasos:

1. Desde la maquina virtual del master abrir una consola.
2. Se procede a loggearse como el usuario sin privilegios que se creo (smonsalve) a través de ssh:

   ```
   $ ssh smonsalve@192.168.56.101
   ```

3. Descargar el Código Fuente a ejecutar:

   ```
   $ wget http://goo.gl/xBhYJ0
   ```

Listing 5.1: Busqueda por amplitud

```
1 // Copyright (C) 2004-2008 The Trustees of Indiana University.
2
3 // Use, modification and distribution is subject to the Boost
      Software
4 // License, Version 1.0. (See accompanying file LICENSE_1_0.txt or
       copy at
5 // http://www.boost.org/LICENSE_1_0.txt)
6
7 //  Authors: Douglas Gregor
8 //           Andrew Lumsdaine
9
10 // Example usage of breadth_first_search algorithm
11
12 // Enable PBGL interfaces to BGL algorithms
13 #include <boost/graph/use_mpi.hpp>
14
15 // Communicate via MPI
16 #include <boost/graph/distributed/mpi_process_group.hpp>
17
18 // Breadth-first search algorithm
19 #include <boost/graph/breadth_first_search.hpp>
20
21 // Distributed adjacency list
22 #include <boost/graph/distributed/adjacency_list.hpp>
23
24 // METIS Input
25 #include <boost/graph/metis.hpp>
26
27 // Graphviz Output
28 #include <boost/graph/distributed/graphviz.hpp>
29
30 // For choose_min_reducer
31 #include <boost/graph/distributed/distributed_graph_utility.hpp>
32
33 // Standard Library includes
34 #include <fstream>
```

```
35 #include <string>
36
37 #ifdef BOOST_NO_EXCEPTIONS
38 void
39 boost::throw_exception(std::exception const& ex)
40 {
41     std::cout << ex.what() << std::endl;
42     abort();
43 }
44 #endif
45
46 using namespace boost;
47 using boost::graph::distributed::mpi_process_group;
48
49 /* An undirected graph with distance values stored on the vertices
       . */
50 typedef adjacency_list<vecS, distributedS<mpi_process_group, vecS
       >, undirectedS,
51                        /*Vertex properties=*/property<
                              vertex_distance_t, std::size_t> >
52   Graph;
53
54 int main(int argc, char* argv[])
55 {
56   boost::mpi::environment env(argc,argv);
57
58   // Parse command-line options
59   const char* filename = "weighted_graph.gr";
60   if (argc > 1) filename = argv[1];
61
62   // Open the METIS input file
63   std::ifstream in(filename);
64   graph::metis_reader reader(in);
65
66   // Load the graph using the default distribution
67   Graph g(reader.begin(), reader.end(),
68           reader.num_vertices());
```

```
69
70  // Get vertex 0 in the graph
71  graph_traits<Graph>::vertex_descriptor start = vertex(0, g);
72
73  // Compute BFS levels from vertex 0
74  property_map<Graph, vertex_distance_t>::type distance =
75    get(vertex_distance, g);
76
77  // Initialize distances to infinity and set reduction operation
       to 'min'
78  BGL_FORALL_VERTICES(v, g, Graph) {
79    put(distance, v, (std::numeric_limits<std::size_t>::max)());
80  }
81  distance.set_reduce(boost::graph::distributed::
       choose_min_reducer<std::size_t>());
82
83  put(distance, start, 0);
84  breadth_first_search
85    (g, start,
86     visitor(make_bfs_visitor(record_distances(distance,
           on_tree_edge()))));
87
88  // Output a Graphviz DOT file
89  std::string outfile;
90
91  if (argc > 2)
92    outfile = argv[2];
93  else {
94    outfile = filename;
95    size_t i = outfile.rfind('.');
96    if (i != std::string::npos)
97      outfile.erase(outfile.begin() + i, outfile.end());
98    outfile += "-bfs.dot";
99  }
100
101 if (process_id(process_group(g)) == 0) {
102   sleep(10);
```

```
103     std::cout << "Writing GraphViz output to " << outfile << "...
            ";
104     std::cout.flush();
105   }
106   write_graphviz(outfile, g,
107                  make_label_writer(distance));
108   if (process_id(process_group(g)) == 0)
109     std::cout << "Done." << std::endl;
110
111   sleep(5);
112   return 0;
113 }
```

4. Para compilar este código es necesario:

```
$ mpicxx -I/share/apps/InstaBoost/ext/Boost/Boost_1.54/include/ \
 -c breadth_first_search.cpp -o breadth_first_search.o
```

5. Lo que arroja un archivo objeto antes de enlazar dinamicamente con las librerías necesarias, para enlazar:

```
$ mpicxx -L/share/apps/InstaladorTesis/ext/Boost/Boost_1.54/lib/ \
 -o breadth_first_search.s breadth_first_search.o \
 -lboost_mpi-mt-s -lboost_serialization-mt-s  \
 -lboost_system-mt-s -lboost_filesystem-mt-s  \
 -lboost_graph-mt-s -lboost_graph_parallel-mt-s
```

   donde breadth_first_search.s es el archivo binario a ejecutar.

6. También se deben descargar los respectivos datos de entrada:

```
$ wget http://goo.gl/yoOYRt
```

Listing 5.2: Lista de Cores para Ejecutar

```
1 5 9 1
2 3 1
3 2 2 4 1 5 2
4 2 7 4 3
```

```
5 5 1
6 1 1 2 1
```

7. Es también necesario contar con un archivo que le indique al proceso en que nodos y que cores se va ejecutar el archivo en nuestro caso, un archivo llamado nodos.txt:

Listing 5.3: Lista de Cores para Ejecutar

```
1 compute-0-0
2 compute-0-0
3 compute-0-1
4 compute-0-1
```

8. para ejecutar finalmente con MPI:

```
$ mpirun -n 4 -machinefile nodes.txt \
> breadth_first_search < weighted_graph.gr
```

9. Debido a las relaciones de confianza entre el master y los nodos, desde el master podemos pasar por ssh a los nodos a través del siguiente comando:

```
$ ssh compute-0-0
```

10. Desde allí se podrá monitorear la actividad del master y los nodos, para lo cual se puede utilizar el programa "htop", el cual se encarga de mostrar los procesos, las CPUs y los cores disponibles dentro del nodo.

11. Desde cada uno de los nodos se abre "htop" para monitorear que se esté procesando en todos los nodos y en cada core, en este caso, dos nodos con dos cores cada uno:

```
$ htop
```

```
$ ssh compute-0-1
```

```
$ htop
```

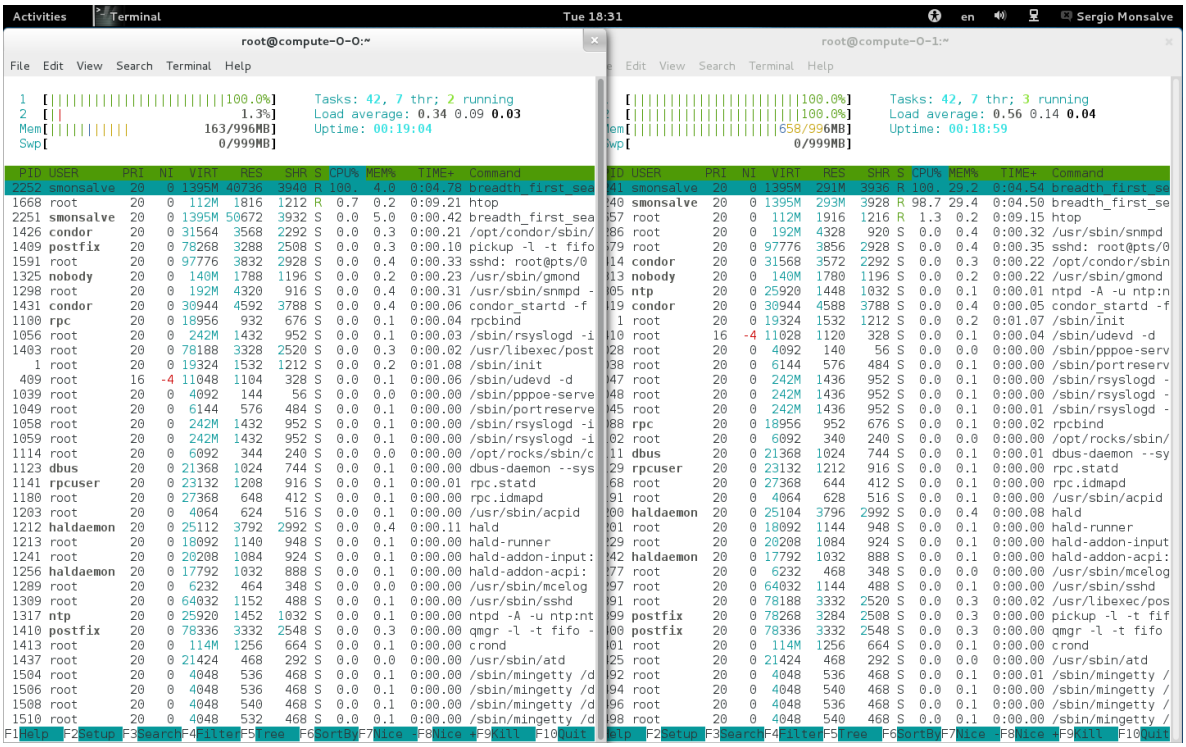

Figura 5.1:

12. Para salir de “htop” oprimir la letra q.

13. Para volver al Master se escribe el comando “exit”.

    ```
    $ exit
    ```

14. Donde se podrán ver los datos de Salida en el mismo directorio donde se ejecuto el programa desde el master. Si se verifica el contenido del directorio debe aparecer el archivo de salida generado, con el nombre: “weighted_graph-bfs.dot”

    ```
    (master)$ ls
    ```

Listing 5.4: Salida Generada por el programa

```
graph g {
  subgraph cluster_0 {
    label="p0";
    n0[label=0];
    n1[label=2];
  }

  n0 -- n2;
  n1 -- n1;
  n1 -- n3;
  n1 -- n4;

  subgraph cluster_1 {
    label="p1";
    n2[label=1];
  }

  n2 -- n1;
  n2 -- n3;

  subgraph cluster_2 {
    label="p2";
    n3[label=2];
  }

  n3 -- n4;

  subgraph cluster_3 {
    label="p3";
    n4[label=1];
  }

  n4 -- n0;
  n4 -- n1;
}
```

15. De esta manera se puede observar que el programa “breadth_first_search.s” se ejecuto de la manera deseada, produciendo como resultado el respectivo archivo de salida “weighted_graph-bfs.dot”, resultado de un proceso paralelizado, utilizando las librerias para uso de grafos en paralelo de Boost.

Capítulo 6

Conclusiones

- Con este proyecto se realizó una serie de procedimientos para simular un Ambiente de Pruebas que replica la arquitectura del centro de computación científica Apolo. Para trabajar en este ambiente de pruebas fue necesario aplicar los conocimientos adquiridos en materias como sistemas operativos, arquitectura de computadores, lenguajes de Programación, telemática, estructuras de datos y algoritmos, entre otras.
- La computación paralela es un campo que en nuestro contexto tiene un gran mercado para su aplicación y este trabajo permite una introducción al tema, el cual se espera sea aprovechado por otras personas para su aplicación.
- Las librerías de Boost son reconocidas por su alta calidad, las cuales brindan elementos para el usuario final que facilitan y potencian el trabajo realizado, permitiendo un análisis complejo de gran cantidad de datos, mediante diferentes herramientas, agilizando la implementación de tales proyectos con una alta calidad.
- Las librerías de Boost mediante su implementación basada en la programación genérica permite hacer uso de sus librerías de una manera fácil y transparente para el programador, permitiendo de esta manera una codificación clara y elegante, lo cual repercute en el tiempo de desarrollo y su legibilidad.

- MPI proporciona un encapsulamiento para las librerías de Boost que permite el funcionamiento del sistema, de tal manera que es transparente para el usuario la comunicación entre toda la arquitectura.

Capítulo 7

Trabajo Futuro

Adicional a las posibilidades de uso de este proyecto planteadas en la justificación y en las posibles aplicaciones, se desea continuar con este proyecto en aplicación a la solución de problemas reales, dentro de las posibilidades para su aplicación se destacan:

- Realizar un análisis de un grafo con gran cantidad de datos.
- Realizar un estudio del rendimiento de un programa con una implementación de Boost en Apolo.
- Agregar una sección para la instalación de un manejador de colas en el ambiente de pruebas, para el control de simulaciones de los usuarios con sus respectivas instrucciones de uso.

Capítulo 8

Glosario

HPC: (High Performance Computing) Computación de alto desempeño.

HTC: (High Throughput Computing.

Data Center: Centro de Computo.

PBGL: (Parallel Boost Graph Library) Libreria Paralela de Grafos de Boost.

MPI: (Message Passing Interface) Interfaz de paso de Mensajes.

Script: Conjunto de Instrucciones para la realización de tareas básicas.

Master: Nodo de mayor jerararquia dentro de un clúster.

Nodo: Donde se tienen varios Cores.

CPU: Unidad Central de Procesamiento.

Core: Nucleo de Procesamiento.

STL: (C++ Standar Library) Liberia Estandar de C++.

Aspectos legales

Boost Software License[15] - Versión 1.0 - 17 de agosto 2003

Se concede permiso, de forma gratuita, a cualquier persona u organización la obtención de una copia del software y la documentación adjunta cubierta por esta licencia (el "Software") para usar, reproducir, exhibir, distribuir, ejecutar y transmitir el Software, y para preparar trabajos derivados de la Software, y para permitir a terceras partes a las que el Software se suministra a hacerlo, todo ello sujeto a lo siguiente:

Los avisos de copyright en el Software y toda esta declaración, incluyendo la concesión de licencia más arriba, esta restricción y el siguiente descargo de responsabilidad, debe incluirse en todas las copias del Software, en todo o en parte, y todos los trabajos derivados del Software, a menos que tales copias o derivado obras son únicamente en forma de código objeto ejecutable por máquina que genera un procesador de lenguaje fuente.

EL SOFTWARE SE PROPORCIONA "TAL CUAL", SIN GARANTÍA DE NINGÚN TIPO, EXPRESA O IMPLÍCITA, INCLUYENDO PERO NO LIMITADO A LAS GARANTÍAS DE COMERCIALIZACIÓN, IDONEIDAD PARA UN PROPÓSITO PARTICULAR, TÍTULO Y NO INFRACCIÓN. EN NINGÚN CASO LOS PROPIETARIOS DEL COPYRIGHT O CUALQUIER PERSONA DISTRIBUIR EL SOFTWARE SE HACE RESPONSABLE POR CUALQUIER DAÑO O CUALQUIER OTRA RESPONSABILIDAD, YA SEA POR CONTRATO, AGRAVIO O DE OTRA MANERA, SURJA DE O EN CONEXION CON EL SOFTWARE O EL USO U OTRO OPERACIONES EN EL SOFTWARE.

Bibliografía

[1] "Hp proliant dl140 quickspecs." http://h18004.www1.hp.com/products/quickspecs/12509_na/12509_na.html.

[2] "Hp proliant bl460c quickspecs." http://h18004.www1.hp.com/products/quickspecs/12518_na/12518_na.html.

[3] "Dell poweredge 1950." http://www.dell.com/support/drivers/us/en/04/Product/poweredge-1950.

[4] "Introduction to Parallel Computing." https://computing.llnl.gov/tutorials/parallel_comp/.

[5] "Software Carpentry: Lessons (Version 4.0)." http://software-carpentry.org/v4/.

[6] J. G. Siek, L.-Q. Lee, and A. Lumsdaine, *Boost Graph Library: User Guide and Reference Manual, The*. Pearson Education, 2001.

[7] "An Overview of the Parallel Boost Graph Library - 1.55.0." http://www.boost.org/doc/libs/1_55_0/libs/graph_parallel/doc/html/overview.html.

[8] A. V. Aho *et al.*, *Data structures and algorithms*. Pearson Education India, 1983.

[9] A. Alexandrescu, *Modern C++ Design: Generic Programming and Design Patterns Applied*. C++ in-depth series, Addison-Wesley, 2001.

[10] B. Stroustrup, *The C++ Programming Language*. Pearson Education, 2013.

[11] G. E. Karniadakis and R. M. K. Ii, "in C ++ and MPI A seamless approach to parallel algorithms and their implementation,"

[12] "Boost MPI - 1.55.0." http://www.boost.org/doc/libs/1_55_0/doc/html/mpi.html.

[13] "Boost c++ libraries." http://www.boost.org/.

[14] "Repast for high performance computing." http://repast.sourceforge.net/repast_hpc.php.

[15] "Licencia de Boost." http://www.boost.org/LICENSE_1_0.txt.

Apéndice A

Repositorio del código

- Para una copia del código utilizado dirigirse a: `https://github.com/smonsalve/tesis.git`
- Para una copia de la maquina virtual utilizada: `http://goo.gl/8eTJOr`
- Codigo de C++ que hace uso de Boost (PBGL) y MPI: `http://goo.gl/xBhYJ0`
- Datos de Entrada para el grafo: `http://goo.gl/yoOYRt`
- Script de Instalación de Boost: `http://goo.gl/unJA9d`
- Script de Instalación de MPI: `http://goo.gl/OBBVf7`