TINKU – IMPLEMENTACIÓN DE UN CLUSTER HETEROGÉNEO DE ALTO RENDIMIENTO, FUERTEMENTE ACOPLADO CON ROCKS EN LA UNIVERSIDAD DE NARIÑO

JORGE EDUARDO MORA ARELLANO
EVELYN CAROLINA VINUEZA MONTENEGRO

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE SISTEMAS
PROGRAMA DE INGENIERÍA DE SISTEMAS
SAN JUAN DE PASTO
2010

TINKU – IMPLEMENTACIÓN DE UN CLUSTER HETEROGÉNEO DE ALTO RENDIMIENTO, FUERTEMENTE ACOPLADO CON ROCKS EN LA UNIVERSIDAD DE NARIÑO

JORGE EDUARDO MORA ARELLANO
EVELYN CAROLINA VINUEZA MONTENEGRO

Trabajo de Grado presentado como requisito parcial para optar el título de Ingeniero de Sistemas

MANUEL BOLAÑOS GONZALEZ
I.S. Esp.
Director

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE SISTEMAS
PROGRAMA DE INGENIERÍA DE SISTEMAS
SAN JUAN DE PASTO
2010

NOTA DE ACEPTACIÓN

_____
_____
_____
_____
_____
_____

_____
Presidente del jurado

_____
Jurado

_____
Jurado

San Juan de Pasto, Octubre 29 de 2010

“Las ideas y conclusiones aportadas en el Trabajo de Grado son responsabilidad exclusiva del autor.”

Artículo 1º del Acuerdo Nº. 324 de octubre 11 de 1966, emanado del Honorable Consejo Directivo de la Universidad de Nariño.

DEDICATORIA

Es increíble que hace algunos años entramos a la universidad con muchas ganas de seguir adelante, de superarnos y convertirnos en profesionales de éxito. Hoy nuestro sueño de ser ingenieros es una realidad, es una meta más de la cual nos sentimos muy orgullosos.

Este proyecto de finalización de carrera y los años de estudio en la Universidad se los dedicamos a Dios, gracias a Él tuvimos la inspiración y las ganas de continuar ante cada obstáculo que se presentaba.

A nuestros padres y madres, gracias a sus esfuerzos pudimos estudiar y culminar una etapa más en nuestras vidas, gracias a sus consejos sabios y por estar siempre a nuestro lado.

Lo dedicamos a nuestras familias que nos han apoyado en todo lo que hemos hecho, quienes son el motor de nuestras ilusiones.

Lo dedicamos a nuestros profesores, quienes compartieron su conocimiento a lo largo de nuestra carrera universitaria, son las personas que siempre llevaremos en el corazón.

Dedicamos este proyecto a nuestros amigos, quienes siempre estuvieron a nuestro lado y nos aconsejaron en momentos en los cuales la vida no marcha bien, quienes contagian la alegría de vivir y seguir adelante.

Nuestro camino aún no termina, este es un gran paso donde dan fruto los años de estudio y esfuerzo, es el momento de desplegar las alas de nuestros sueños y hacer lo necesario para alcanzar el cielo.

AGRADECIMIENTOS

Queremos expresar nuestro agradecimiento a nuestros padres, sin ellos todo lo que hemos logrado no hubiera sido posible.

Al Doctor Jorge Zuluaga de la Universidad de Antioquia, por su inmensa colaboración en el área de renderización de imágenes.

Al Doctor José Hernández Palancar del Instituto CENATAV de La Habana Cuba y al ingeniero Andrés Calderón, gracias por su colaboración en el área de minería de datos.

A nuestro asesor, Especialista Manuel Bolaños González, muchas gracias por su colaboración a lo largo del desarrollo de este proyecto.

A nuestro compañero, Ingeniero Fabio Hurtado, al señor Alfredo Insuasty y a la doctora María Fernanda López, quienes colaboraron con el préstamo de equipos de cómputo, sin su colaboración este proyecto no hubiera podido ser culminado.

Por ahora, nos enorgullece saber que llevaremos durante nuestra vida esta gran disciplina de la mejor manera posible.

CONTENIDO

	Pág.
GLOSARIO	19
RESUMEN	22
ABSTRACT	23
INTRODUCCIÓN	24
1. MARCO TEÓRICO	28
1.1 TÉRMINOS TECNOLOGÍA CLUSTER	28
1.2 TÉRMINOS TRAZADO DE RAYOS	29
1.3 TÉRMINOS PROGRAMACIÓN PARALELA	31
1.4 TÉRMINOS MINERÍA DE DATOS	34
1.5 TÉRMINOS COMPLEMENTARIOS	36
1.6 CONCEPTO CLUSTER	37
1.7 TIPOS DE CLUSTER	37
1.7.1 Cluster de alta disponibilidad HA (High Availability)	38
1.7.2 Cluster de balanceo de carga LB (Load Balancing)	38
1.7.3 Cluster de alto rendimiento HP (High Performance)	38
1.8 APLICACIONES COMUNES DE LOS CLUSTER HP	38

1.8.1 Investigación en ciencias naturales. 39

1.8.2 Procesamiento de gráficos. 39

1.8.3 Procesamiento de datos 40

1.9 ARQUITECTURA DE CLUSTER 40

1.10 ACOPLAMIENTO DE UN CLUSTER 41

1.11 HOMOGENEIDAD DE UN CLUSTER 42

1.12 ARQUITECTURA DE COMPUTADORES 43

1.13 ARQUITECTURAS DE COMPUTACIÓN PARALELA 48

1.14 CATEGORÍAS DE CLUSTER 51

1.15 PROCESAMIENTO EN PARALELO 52

1.16 MODELO GENERAL DE COMPUTACIÓN EN PARALELO 53

1.17 COMPUTACIÓN SECUENCIAL 53

1.18 COMPUTACIÓN PARALELA 54

1.19 MODELO DE INTERACCIÓN ENTRE PROCESADORES 54

1.20 PROS Y CONTRAS DE LA PROGRAMACIÓN EN PARALELO 55

1.21 APROXIMACIÓN A PI UTILIZANDO MONTECARLO 56

1.22 ANTECEDENTES 57

2. DESCRIPCIÓN DE TINKU 60

2.1 GENERALIDADES 60

2.1.1 Arquitectura de Tinku 62

2.1.2 Distribución Rocks 62

2.1.3 Requisitos y características del Cluster Tinku 63

2.2 COMPONENTES BÁSICOS DE UN CLUSTER 64

2.3 SERVICIOS 65

2.4 HERRAMIENTAS UTILIZADAS EN EL CLUSTER TINKU 66

2.4.1 Ganglia 66

2.4.2 Sun Grid Engine (SGE) 67

2.4.3 Java 68

2.5 HERRAMIENTAS PARA DETERMINAR EL DESEMPEÑO DEL CLUSTER TINKU 69

2.5.1 MPJ Express 69

2.5.2 POV-Ray 70

2.5.3 Zebra-Core 72

2.5.4 WEKA-Parallel 72

2.6 DIAGRAMAS GENERALES UML DEL SISTEMA CLUSTER TINKU 73

3. DESARROLLO DEL CLUSTER TINKU 75

3.1 INSTALACIÓN Y CONFIGURACIÓN DE TINKU CON ROCKS 75

3.1.1 Instalación y configuración del frontend 75

3.1.2 Instalación nodos 91

3.1.3 Instalación de herramientas para renderizar imágenes en paralelo 95

3.1.3.1 POV-Ray 95

3.1.3.2 Instalación Zebra-Core 98

3.1.4 Instalación de herramienta para minería de datos en paralelo 100

3.1.4.1 Instalación WEKA-Parallel 100

3.1.5 Instalación de herramienta para programación en paralelo 105

3.1.5.1 Instalación MPJ Express 105

4. MANEJO BÁSICO DE ROCKS 108

4.1 INTRODUCCIÓN A LA PLATAFORMA 108

4.1.1 Encender y apagar el cluster 108

4.1.2 Conectarse a un nodo 108

4.1.3 Sistemas de archivos 109

4.1.4 Ejecución de comandos remotos 110

4.1.5 Comandos propios de Rocks 111

4.1.6 Monitoreo de los recursos principales 112

4.1.7 Manejo sistema gestor de colas Sun Grid Engine 113

4.2 ADMINISTRACIÓN Y MONITOREO 116

4.2.1 Sistemas de archivos locales 116

4.2.2 Servicio 411 116

4.2.3 Administración de usuarios 118

4.2.4 Instalación software adicional 119

4.2.5 Monitoreo con Ganglia 125

4.3 CONEXIONES DE RED DEL CLUSTER TINKU 126

4.3.1 Conexión del Cluster Tinku con Internet 126

4.3.2 Escritorio remoto al Cluster Tinku 127

4.3.2.1 Configuración VNCServer. 129

4.3.2.2 Instalación y configuración Hamachi 131

5. PRUEBAS Y RESULTADOS 134

5.1 PRUEBAS DE RENDERIZACION DE IMÁGENES CON ZEBRA-CORE 135

5.1.1 Renderización primer archivo 135

5.1.2 Renderización segundo archivo 137

5.2 EVALUACIÓN DE LAS PRUEBAS DE RENDERIZACIÓN 139

5.3 PRUEBAS CON TÉCNICA DE MINERÍA DE DATOS 139

5.3.1 Procesamiento 50000 registros 142

5.3.2 Procesamiento 100000 registros 143

5.3.3 Procesamiento 200000 registros 144

5.4 EVALUACIÓN PRUEBAS TÉCNICA DE MINERÍA DE DATOS 145

5.5 PRUEBAS ALGORITMO PARALELO 146

5.6 EVALUACION DE LAS PRUEBAS DEL ALGORITMO EN PARALELO 148

CONCLUSIONES 149

RECOMENDACIONES 150

REFERENCIAS 151

ANEXOS 156

LISTA DE TABLAS

	Pág.
Tabla 1. Partición por defecto	86
Tabla 2. Características equipos	134
Tabla 3. Escenas a procesar en el cluster	135
Tabla 4. Tiempos de ejecución Zebra-Core chess2.pov 1024x768	136
Tabla 5. Tiempos de ejecución Zebra-Core chess2.pov 2816x2112	136
Tabla 6. Tiempos de ejecución Zebra-Core chess2.pov 3456x2304	137
Tabla 7. Tiempos de ejecución en Zebra-Core de glasschess.pov 800x600	138
Tabla 8. Tiempos de ejecución en Zebra-Core de glasschess.pov 1024x768	138
Tabla 9. Tiempos de ejecución en Zebra-Core de glasschess.pov 2816x2112	138
Tabla 10. Conjuntos de datos	140
Tabla 11. Tiempos promedio ejecución algoritmo paralelo	147

LISTA DE FIGURAS

	Pág.
Figura 1. Arquitectura de un cluster	41
Figura 2. Modelo SISD	43
Figura 3. Modelo SIMD	44
Figura 4. Modelo MISD arreglo de procesadores	45
Figura 5. Arreglo sistólico	45
Figura 6. Modelo UMA sistema de memoria compartida	47
Figura 7. Modelo NUMA memoria distribuida	48
Figura 8. Arquitectura DMM sin compartir	49
Figura 9. Arquitectura de disco compartido	49
Figura 10. Cluster de SMPs	49
Figura 11. Máquinas de memoria compartida distribuida	50
Figura 12. Computación secuencial	53
Figura 13. Computación paralela	54
Figura 14. Aproximación PI mediante Montecarlo	56
Figura 15. Estructura cluster FING	58
Figura 16. Diagrama de casos de uso Cluster Tinku	73

Figura 17. Diagrama de clases Cluster Tinku 73

Figura 18. Diagrama de despliegue Cluster Tinku 74

Figura 19. Demonios de Ganglia 66

Figura 20. Interfaz Ganglia 67

Figura 21. Interfaz de SGE 68

Figura 22. Escena generada con POV-Ray 70

Figura 23. Topología Cluster Tinku 75

Figura 24. Pantalla inicial instalación Rocks 5.2 76

Figura 25. Configuración TCP/IP 77

Figura 26. Selección inicial de Roll 77

Figura 27. Selección Kernel Roll 78

Figura 28. Selección de Rolls – Kernel Roll agregado 78

Figura 29. Insertar Roll 79

Figura 30. Selección Core Roll 79

Figura 31. Selección de Rolls - Core Roll agregado 79

Figura 32. Selección de OS Roll Disk 1 80

Figura 33. Selección de Rolls – OS Roll Disk 1 agregado 80

Figura 34. Selección de OS Roll Disk 2 81

Figura 35. Selección de Rolls – Todos los Rolls agregados 81

Figura 36. Información del cluster 82

Figura 37. Configuración interfaz eth0 83

Figura 38. Configuración interfaz eth1 83

Figura 39. Configuración puerta de enlace y servidor DNS 84

Figura 40. Asignación contraseña root 84

Figura 41. Configuración zona horaria 85

Figura 42. Particionamiento del disco 85

Figura 43. Particionamiento manual 86

Figura 44. Instalación final de los paquetes 87

Figura 45. Instalación frontend Scientific Linux 87

Figura 46. Ajustes de fecha y hora 90

Figura 47. Propiedades de fecha y hora 90

Figura 48. Conexión Servidor NTP 91

Figura 49. Adición de un nodo 91

Figura 50. Nuevo nodo descubierto 92

Figura 51. Nodo reconocido esperando Kickstart 93

Figura 52. Nodo reconocido Kickstart enviado 93

Figura 53. Adición de nodos utilizando Avalanche Installer 94

Figura 54. Interfaz Inicial WEKA 101

Figura 55. WEKA Knowledge Explorer 102

Figura 56. Abrir repositorio de datos 103

Figura 57. Interfaz algoritmos de clasificación 103

Figura 58. Ajustes de WEKA-Parallel 104

Figura 59. Configurar validación cruzada en paralelo 104

Figura 60. Interfaz Ganglia 125

Figura 61. Archivo configuración VNC 130

Figura 62. Imagen prueba chess2.pov 136

Figura 63. Resultados renderización Zebra-Core chess2.pov 137

Figura 64. Imagen prueba glasschess.pov 137

Figura 65. Resultados renderización Zebra-Core glasschess.pov 139

Figura 66. Tiempos de entrega de resultados J.48 con 50000 142

Figura 67. Tiempos de entrega de resultados M5P con 50000 143

Figura 68. Tiempos de entrega de resultados J.48 con 100000 143

Figura 69. Tiempos de entrega de resultados M5P con 100000 144

Figura 70. Tiempos de entrega de resultados J.48 con 200000 145

Figura 71. Tiempos de entrega de resultados M5P con 200000 145

Figura 72. Tiempos promedio de ejecución del algoritmo paralelo 147

LISTA DE ANEXOS

Anexo A. Código fuente paralelo Aproximación Pi mediante Montecarlo 156

Anexo B. Lista de siglas 159

GLOSARIO

APLICACIÓN DISTRIBUIDA: aplicación con varios componentes que se ejecutan en entornos diferentes, conectados a través de una red. Las típicas aplicaciones distribuidas son de dos niveles (cliente-servidor), tres niveles (cliente-middleware-servidor) y multinivel.

COMANDO: es una orden dada a un programa de computación que actúa como intérprete, para realizar una tarea específica.

CLUSTER: se aplica a los conjuntos o conglomerados de computadores construidos mediante la utilización de componentes de hardware comunes y que se comportan como si fuesen un único computador.

DHCP (Dynamic Host Configuration Protocol o Protocolo de configuración de host dinámico): es un servicio que permite a los ordenadores asignar automáticamente determinados parámetros de configuración de red.

DNS (Domain Name System o Sistema de Nombre de Dominio): es un sistema de nomenclatura jerárquica para computadores, servicios o cualquier recurso conectado a Internet o a una red privada. Traduce nombres inteligibles para los humanos en identificadores binarios asociados con los equipos conectados a la red, esto con el propósito de poder localizar y direccionar de una forma más eficiente.

ETHERNET: estándar 802.3x de redes de computadores de área local con acceso al medio por contienda CSMA/CD. Define las características de cableado y señalización de nivel físico y los formatos de tramas de datos del nivel de enlace de datos del modelo OSI.

COMPUTACIÓN GRID: infraestructura que permite el acceso y procesamiento concurrente de un programa entre varios computadores independientes.

FOLDS: número de particiones en un conjunto de datos al cual se le aplica validación cruzada.

FRONTEND: es el equipo principal en un cluster, donde los usuarios se conectan al sistema, presentan tareas, compilan código, distribuyen procesos, entre otras funciones.

HTTP (Hypertext Transfer Protocol o Protocolo de Transferencia de Hipertexto): protocolo usado para acceder a la Web, el cual se encarga de procesar y responder las peticiones para visualizar una página web.
IP (Internet Protocol o Protocolo de Internet): es un protocolo no orientado a conexión utilizado para la comunicación de datos entre un origen y un destino a través de una red de paquetes conmut

MIDDLEWARE: software de conectividad que ofrece un conjunto de servicios con el fin de soportar aplicaciones distribuidas sobre plataformas heterogéneas. Funciona como una capa de abstracción de software distribuida, que se sitúa entre las capas de aplicación y el sistema operativo.

MULTICAST: protocolo que permite la comunicación de uno a muchos, es una de las características que debe estar presente en una red para que se considere avanzada, además permite optimizar el uso de la red.

NODO: Se refiere a un computador que contiene recursos específicos, tales como memoria, interfaces de red, uno o más procesadores, etc. Son las unidades esenciales de un cluster que realizan el trabajo de procesamiento u almacenamiento.

RENATA (Red Nacional Académica de Tecnología Avanzada): red de tecnología avanzada que conecta, comunica y propicia la colaboración entre la comunidad académica y científica de Colombia con la comunidad académica internacional y los centros de investigación más importantes del mundo.

RSA: sistema criptográfico de clave pública válido para cifrar como para firmar digitalmente. La seguridad de este algoritmo radica en el problema de la factorización de números enteros.

SHELL: intérprete de comandos por el cual un usuario se comunica con el sistema operativo.

SERVICIOS WEB: conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, utilizan los servicios web para intercambiar datos en redes de ordenadores como Internet.

SSH (Secure Shell): es un protocolo de red y un conjunto de estándares que permiten una conexión encriptada entre dos computadoras. SSH se utiliza para acceder a una máquina remota y ejecutar comandos en ésta. Sirve para proteger conexiones inseguras y transferir archivos de manera segura.

TCP (Transmission Control Protocol o Protocolo de Control de Transmisión): es uno de los protocolos fundamentales en Internet, que permite crear conexiones

entre computadores a través de las cuales se envía un flujo de datos. Garantiza que los datos sean entregados sin errores y en el mismo orden en que se transmitieron.

RESUMEN

En este documento se describe la implementación de un modelo de cluster heterogéneo de alto rendimiento en la Universidad de Nariño.

Este proyecto denominado Tinku tiene como finalidad unir la potencia de procesamiento de varios computadores para la solución rápida y eficiente de un determinado problema complejo.
Principalmente el cluster Tinku desarrolla funcionalidades sobre el procesamiento de imágenes y procesamiento de datos, que demostrarán las cualidades de este tipo de tecnología.

Con la implementación del clúster Tinku se alcanzan grandes avances académicos e investigativos, de este modo la Universidad de Nariño estará en el mismo nivel de otras universidades en Colombia que ya tienen implementados sus propios clusters.

Se ha elaborado el manual de usuario del cluster y video tutoriales para la instalación y configuración del cluster.

ABSTRACT

This document describes a heterogeneous, tightly coupled prototype cluster implementation at Universidad de Nariño.

This project has been denomined Tinku, is intended to unite the processing power of multiple computers for fast and efficient solution of a complex problem.
Mainly Tinku cluster develops many functions about image rendering and data processing; those will demonstrate the features of this kind of technology.

With the implementation of Tinku cluster will achieve greater academic and research progress, thus Universidad de Nariño will be at the same level like another universities in Colombia.

Related user manual and video tutorials have been made for installing and configuring the cluster.

INTRODUCCIÓN

En el entorno académico e investigativo los problemas computacionales que surgen son cada vez más complejos, por esto demandan gran capacidad de procesamiento de datos y por ende tiempo. Debido a esto es necesario que los resultados sean confiables, eficientes y se entreguen de manera oportuna, para satisfacer estas exigencias se requiere la adquisición de un supercomputador, el cual da soporte a los requerimientos de eficiencia, eficacia y confiabilidad a los problemas propuestos. Se presenta un inconveniente cuando se quiere adquirir una máquina con características especiales, el primero es el costo, es muy elevado y no todas las instituciones investigativas pueden obtener, el segundo problema se presenta en el mantenimiento preventivo y correctivo del equipo, resulta difícil encontrar un elemento dañado en una maquina de esta magnitud, si se llega a detectar el daño y se necesita reemplazar por un nuevo elemento, será una tarea dispendiosa y costosa la adquisición de éste. Una de las consecuencias que se generarían a causa de esto sería la suspensión temporal de las actividades que se venían realizando y las que estaban programadas.

Una alternativa factible y al alcance de la academia es la implementación de un cluster que ofrece las mismas y mejores prestaciones de un supercomputador. Un cluster brinda mayor capacidad de procesamiento de datos debido a que se puede aumentar la capacidad según se estime, los problemas que se pueden presentar en un cluster son de fácil e inmediata solución.

Observando las necesidades que se presentan en la Universidad de Nariño respecto a estos problemas de cálculo y procesamiento de datos para las diferentes investigaciones, por esto es importante llevar a cabo este proyecto denominado “Tinku – Implementación de un cluster heterogéneo de alto rendimiento, fuertemente acoplado con Rocks en la Universidad de Nariño”, Tinku es un vocablo quechua que significa: unión de cosas. El presente proyecto corresponde a la modalidad de Trabajo de Investigación aprobado por el Departamento de Sistemas de la Universidad de Nariño, bajo la línea de Sistemas y Manejo de Información enfocado en las siguientes áreas: Telemática, Sistemas Distribuidos, Sistemas Operativos, Base de Datos y Multimedia.

La realización de este proyecto se justifica en la demostración de la cantidad de funcionalidades que aportaría un clúster para el progreso y desarrollo de la Universidad de Nariño en el aspecto de desarrollo de investigaciones y proyectos de mejor calidad. Dando así un impulso para una posterior implementación permanente de un clúster una vez se determine su espacio y componentes adecuados que se facilitará por medio de la documentación generada por este proyecto. Además, con la implementación de un cluster se logran aprovechar los

diferentes recursos computacionales que tiene la Universidad de Nariño destinados a la investigación en nuevas áreas del conocimiento, evitando el gasto en máquinas costosas que poseen muchos inconvenientes.

La Universidad de Nariño como fuente de conocimiento del suroccidente colombiano y especialmente el Departamento de Sistemas demandan la implementación de tecnologías que ayuden a obtener resultados de manera confiable y eficiente, al mismo tiempo se desea establecer nuevas líneas de investigación en el programa acreditado.

El problema objeto de estudio en este proyecto es buscar una alternativa que ofrezca grandes ventajas en el procesamiento masivo de datos logrando una reducción de tiempo en la entrega de resultados de estos, que pueden aportar a proyectos de investigación y prácticas académicas obteniendo confiabilidad en la entrega de resultados.

Además, es muy importante estar al mismo nivel de otras universidades en el país que poseen proyectos de esta magnitud. A partir de este planteamiento es necesario formular la siguiente pregunta: ¿Cómo realizar procesamiento de alto rendimiento al alcance de la Universidad de Nariño, obteniendo eficiencia en los resultados?

Tinku, un modelo de cluster de alto rendimiento es la solución a lo anterior porque se compone de varios recursos computacionales que cuenta con herramientas software especializadas en alta capacidad de procesamiento.
La meta de un cluster de alto rendimiento es alcanzar el mayor desempeño en la velocidad de procesamiento de datos. Un cluster es una agrupación de computadores que trabajan en paralelo, dividiendo el trabajo en tareas más pequeñas las cuales se desarrollan al mismo tiempo disminuyendo el tiempo en obtener resultados.

Tinku es un modelo de cluster de alto desempeño fuertemente acoplado que otorga lo siguiente:
Ejerce un papel muy importante en el desarrollo de investigaciones académicas en las cuales surgen de manera regular proyectos que tienen que ver con cálculos complejos o con el procesamiento masivo de información.
Desempeña un papel importante en la creación de imágenes fotorealistas usando la técnica de trazado de rayos.
Otorga soporte a problemas de minería de datos enfatizando en la técnica de validación cruzada.
Brinda un nuevo espacio para desarrollar nuevas técnicas de programación como lo es la programación en paralelo.

El objetivo principal de este proyecto es implementar un modelo de cluster de alto rendimiento, heterogéneo y fuertemente acoplado, bajo software libre.

Con el fin de cumplir con lo mencionado se plantearon los siguientes objetivos específicos:

- Investigar los beneficios que trae la implementación de un cluster de alto rendimiento.

- Estudiar y comparar diferentes distribuciones de software especializado para cluster y herramientas para el correcto funcionamiento de un cluster.

- Estudiar y analizar diferentes herramientas que evidencien el alto rendimiento de un cluster.

- Implementar un modelo de cluster de alto rendimiento con las herramientas apropiadas para este tipo de tecnología.

- Realizar pruebas que demuestren el rendimiento del cluster y analizar los resultados comprobando así la eficiencia y eficacia que éste presta.

- Elaborar la respectiva documentación de la implementación del cluster que contará con video tutoriales y su correspondiente manual.

Tinku estará a la disposición de otras instituciones que tengan convenios con la Universidad de Nariño como el que se adquirió con RENATA (Red Nacional Académica de Tecnologías Avanzadas) con fines de mutua colaboración en el campo del cálculo intensivo y el procesamiento de datos.
Por último se quiere que este proyecto sea el motor que impulse nuevas investigaciones en diferentes áreas del conocimiento con el fin de obtener grandes beneficios como mejorar la calidad educativa en el departamento de Nariño.

Este documento se encuentra organizado de la siguiente manera: En el capítulo 1 se encuentra el marco teórico, seguido del capítulo que describe las características de Tinku, después se encuentra el capítulo que describe todo el desarrollo del proyecto, posteriormente se describe el manejo básico de Tinku, y finalmente se presentan los resultados de las pruebas.

1. MARCO TEÓRICO

1.1 TÉRMINOS TECNOLOGÍA CLUSTER

- **Tecnología cluster**

"Permite a las organizaciones incrementar su capacidad de procesamiento usando tecnología estándar, tanto en componentes de hardware como de software que pueden adquirirse a un costo relativamente bajo"[1].

- **Conexiones de red**

"Los nodos de un cluster pueden conectarse mediante una simple red Ethernet, o puede utilizar tecnologías especiales de alta velocidad como Fast Ethernet, Gigabit Ethernet, Myrinet, Infiniband, SCI"[2].

- **Cómputo científico**

"El desarrollo y traducción en términos de tecnología de cómputo para procesar modelos computacionales, cuya finalidad es resolver problemas complejos de ciencia e ingeniería"[3].

- **Sistema operativo (SO)**

"Es el software básico de una computadora que provee una interfaz entre el resto de programas del ordenador, los dispositivos hardware y el usuario. Las funciones básicas del Sistema Operativo son administrar los recursos de la máquina, coordinar el hardware y organizar archivos y directorios en dispositivos de almacenamiento"[4].

- **Middleware**

---

[1] CLUSTERFIE.EPN.EDU.EC. Clusters. [en línea]. Disponible en web: (http://clusterfie.epn.edu.ec/clusters/Definiciones/definiciones.html).

[2] Ibíd.

[3] CASTILLO, José et al. Implementación de un cluster OpenMosix para cómputo científico en el Instituto de Ingeniería. México, 2006. Universidad Nacional Autónoma de México. Facultad de Ingeniería.

[4] MASADELANTE.COM. Definición sistema operativo [en línea]. Disponible en web: (http://www.masadelante.com/faqs/sistema-operativo).

"Middleware es una clase de tecnología software diseñada para ayudar a manejar la complejidad y heterogeneidad inherentes a sistemas distribuidos. Se define como una capa de software por encima del sistema operativo, pero abajo del programa de aplicación que proporciona una abstracción de programación común a través de un sistema distribuido. Existen diversos tipos de middleware, como por ejemplo: Rocks, Open MOSIX, PelicanHPC, OSCAR, entre otros"[5].

- **Sistema de imagen única (SSI)**

"Es un tipo de cluster donde los usuarios o procesos ven el conjunto de máquinas como una unidad. Este es el modelo de referencia para muchos de los sistemas operativos que actualmente están en desarrollo para arquitecturas del tipo cluster"[6].

- **Computación paralela**

"La computación paralela como modelo informático, tiene el objetivo de solventar problemas de alta complejidad computacional mediante la unión de los recursos individuales de un conjunto de elementos de proceso"[7].
"En el sentido más simple, la computación paralela es el uso simultáneo de varios recursos de cómputo para resolver problemas:
Para ser ejecutado en múltiples CPU.
Un problema puede ser divido en partes discretas que pueden ser resueltas concurrentemente.
Cada parte es subdividida en una serie de instrucciones.
Las instrucciones de cada parte se ejecutan simultáneamente en diferentes CPU"[8].

- **Proceso**

"Un proceso es básicamente un programa en ejecución. Cada proceso tiene asociado un espacio de direcciones, es decir una lista de posiciones de memoria desde algún mínimo hasta algún máximo que el proceso puede leer y escribir"[9].

1.2 TÉRMINOS TRAZADO DE RAYOS

[5] BAKKEN, David. Middleware. [en línea]. Disponible en web: (http://eecs.wsu.edu/~bakken/middleware.pdf).
[6] PEROZO, Freddy et al. Cluster Mangosta, Implementación y evaluación. [en línea]. Disponible en web: (http://servicio.cid.uc.edu.ve/facyt/v1n1/1-1-2.pdf).
[7] GÁLVEZ, Yadisnel.Computación Paralela: Una solución para el presente y el futuro. [en línea]. Disponible en web: (http://www.uci.cu/files/investigaciones/vol3/Vol.%203,%20No.%203.pdf).
[8] BARNEY,Blaise. Introduction to Parallel Computing. [en línea]. Disponible en web: (https://computing.llnl.gov/tutorials/parallel_comp/).
[9] CASTILLO, Op. cit.

- **Renderización**

"Es el proceso de generar una imagen de una descripción de objetos tridimensionales por medio de un software. La descripción está en un lenguaje estrictamente definido o en una estructura de datos, y contiene formas geométricas, puntos de observación, texturas e información de luces"[10].

- **Trazado de rayos**

"Se basa en lanzar una serie de rayos a partir de una fuente de luz, y rastrearlos al momento en que son interceptados por algún objeto en cierta escena. Por lo tanto crea imágenes que son resultado de reflexiones y refracciones que dependen de las superficies de los cuerpos que se encuentren en el ambiente de la escena"[11].

- **Radiosidad**

"Permite calcular la cantidad de iluminación que recibe cualquier elemento de una escena de forma relativa al resto de elementos"[12].

- **Mapeado de fotones**

"Es una de las técnicas que ayudan a acelerar el cálculo de la iluminación global drásticamente. Consiste en dos pasos: disparo de fotones y recolección final. En el primer paso, un número específico de paquetes de luz llamado fotones son disparados desde una fuente de luz y rebotados por la escena. Siempre que un fotón choque con una superficie, el punto de intersección, la dirección entrante, y la energía del fotón son almacenadas en una caché llamada mapa de fotones.
El Mapeado de Fotones es sólo una aproximación de una luz en cualquier punto. Por lo tanto, necesitamos un paso más para conseguir resultados más precisos.
El segundo paso utiliza recolección final que calcula con mayor precisión la iluminación en un punto de muestreo"[13].

- **Geometría sólida constructiva (CSG)**

---

[10] WORDIQ. Computer rendering – Definition. [en línea]. Disponible en web: (http://www.wordiq.com/definition/Computer_rendering).
[11] CARRASCO, Jorge. Una herramienta para la síntesis de imágenes de alta resolución empleando Ray-Tracing. [en línea]. Disponible en web: (http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/carrasco_r_j/capitulo_2.html).
[12] ALONSO, David et al. Radiosidad. [en línea]. Disponible en web: (http://blackspiral.org/docs/radiosidad/guion.pdf).
[13] KRAYTRACING.COM. Photon mapping. [en línea]. Disponible en web: (http://www.kraytracing.com/wiki/Basics_of_global_illumination/es).

"Es una poderosa herramienta para combinar primitivas simples para crear objetos más complejos de cuatro maneras diferentes.
En la unión, dos o más formas se asocian para ser utilizadas como una sola pieza. En el caso de la intersección, sólo aparece el volumen común a todas las figuras. La diferencia, es análoga a la intersección, aunque no lo parezca, ya que en este caso las formas recortan su propio volumen de la que aparece en primer lugar. Y por último, aunque no sea la menos importante, la fusión, que es un tipo especial de unión en la que las superficies interiores a los objetos desaparecen, lo cual es especialmente útil en los objetos transparentes"[14].

- **Lenguaje de descripción de escena**

"El lenguaje de descripción de escena permite describir el mundo de forma legible y cómoda. Los archivos son creados en texto plano ASCII"[15].
Se define en base a un espacio de coordenadas x,y,z. Se puede definir una fuente luz, una cámara y una serie de objetos. "La forma en que trabajan los trazadores de rayos, es por imitación a la forma en que las imágenes llegan a una cámara o a la retina de un ojo, pero otras veces solo se busca un efecto parecido y se usan trucos de computación"[16].

1.3 TÉRMINOS PROGRAMACIÓN PARALELA

- **Ambientes de programación**

"Los ambientes de programación paralela permiten implementar algoritmos que hagan uso de recursos compartidos: CPU, memoria, datos y servicios"[17].

- **Tarea**

"Una sección lógica discreta de trabajo computacional. Una tarea es típicamente un programa o conjunto de programas de instrucciones que es ejecutada por un procesador"[18].

- **Paso de mensajes**

---

[14] POV-RAY.ORG. POV-Ray Documentation What is CSG?. [en línea]. Disponible en web: (http://www.povray.org/documentation/view/3.6.1/28/).
[15] POV-RAY.ORG POV-Ray Documentation Scene Description Language. [en línea]. Disponible en web: (http://www.povray.org/documentation/view/3.6.1/224/).
[16] CASTRO, Antonio. El trazador de rayos Povray (Tutorial, principios básicos I). [en línea]. Disponible en web: (http://www.ciberdroide.com/wordpress/el-trazador-de-rayos-povray-principios-basicos-i/).
[17] CLUSTERFIE.EPN.EDU.EC, Op. Cit.
[18] BARNEY, Op. Cit.

"El paso de mensajes es un modelo de interacción entre los procesadores que forman un sistema paralelo. A grandes rasgos un mensaje es un conjunto de datos definido mediante software por un procesador y enviado a través de una red de comunicaciones hacia otro procesador, el cual debe aceptar el mensaje y actuar según su contenido"[19].

- **Memoria compartida**

"Es una extensión directa del modelo convencional uniprocesador, en el que a cada procesador se le proporciona la abstracción de que existe una memoria única en la máquina. Una actualización a los datos compartidos es visible a todos los procesadores del sistema"[20]

- **Memoria distribuida**

"Cada procesador tiene una memoria privada a la que otros procesadores no tienen acceso directo. La única forma de comunicación entre los procesadores es explícitamente a través del paso de mensajes."[21].

- **Sincronización**

"La coordinación de tareas paralelas en tiempo real, con frecuencia asociadas con las comunicaciones. A menudo implementadas por el establecimiento de un punto de sincronización dentro de una aplicación donde una tarea no puede continuar hasta que otra u otras tareas alcanzan el mismo punto"[22].

- **Granularidad**

"El término granularidad se usa como el mínimo componente del sistema que puede ser preparado para ejecutarse de manera paralela"[23].

"Granularidad gruesa: Relativamente grandes cantidades de trabajo computacional son hechas entre eventos de comunicación.
Granularidad fina: Relativamente pequeñas cantidades de trabajo computacional son hechas entre eventos de comunicación"[24].

---

[19] REYES, Vicente. Procesamiento Paralelo en Redes Linux Utilizando MPI. [en línea]. Disponible en web: (http://beta.redes-linux.com/manuales/cluster/mpi-spanish.pdf).
[20] TRIBALDOS, Miguel. Elefante, Memoria Distribuida. [en línea]. Disponible en web: (http://personales.alumno.upv.es/~mitriher/elefante/memoria.html).
[21] CATALÁN, Miquel. El manual para el clustering con openMosix. [en línea]. Disponible en web: (http://es.tldp.org/Manuales-LuCAS/doc-manual-openMosix-1.0/doc-manual-openMosix-1.0.pdf).
[22] BARNEY, Op. Cit.
[23] CATALÁN, Op. Cit.
[24] BARNEY, Op. Cit.

- **Aceleración o Speed-Up**

"Es una métrica importante de desempeño en computación paralela se refiere a que tan rápido es un algoritmo paralelo en comparación con su correspondiente algoritmo secuencial. Está definido por la siguiente fórmula:

$$S_p = T_s / T_p \qquad (1)$$

Donde p es el número de procesadores, Ts es el tiempo de ejecución del algoritmo secuencial, y Tp es el tiempo de ejecución del algoritmo paralelo con p procesadores"[25].

- **Eficiencia**

"La eficiencia mide la cantidad de potencia de procesamiento disponible que está siendo usada"[26].
"La eficiencia de un algoritmo paralelo ejecutado en p procesadores es la aceleración dividida por p"[27].

$$E = \frac{S_p}{p} \qquad (2)$$

"Lo usual es que la eficiencia vaya disminuyendo a medida que crece el número de procesadores. Al crecer el número de procesadores empiezan a aparecer fenómenos de congestión en diferentes componentes del equipo"[28].

- **Escalabilidad**

"Un sistema escalable permite al usuario ir incrementando el número de procesadores en su equipo a medida que van creciendo sus necesidades"[29].

- **Velocidad de transferencia de datos**

---

[25] XIAO, Mei et al. A Parallel Algorithm of Constructing Gene Regulatory Networks. [en línea]. Disponible en web: (http://www.aporc.org/LNOR/11/OSB2009F25.pdf).
[26] Ibíd.
[27] GARCÍA, Oscar et al. Computación Paralela. [en línea]. Disponible en web: (http://www.dynamics.unam.edu/DinamicaNoLineal/Proyectos/Supercomputo/ComputacionParalela.pdf).
[28] TREFFTZ, Christian. Procesamiento Paralelo en EAFIT. [en línea]. Disponible en web: (http://www1.eafit.edu.co/drupal/?q=node/549).
[29] Ibíd.

Se define “la velocidad de transmisión como el número de bits transmitidos por segundo.
Su unidad es el bps (bits por segundo). En general si el número de estados posibles de la línea de comunicación es n, a cada estado le corresponderán $\log_2 n$ bits de información”[30].

- **Latencia**

Se llama “latencia al mínimo tiempo empleado en transmitir un dato, incluyendo la información de control necesaria para enviarlo o recibirlo. La latencia es muy importante en procesamiento paralelo porque determina el tamaño granular, es decir, lo mínimo que debe tardar la ejecución de un segmento de código para que su ejecución en paralelo sea rentable, hablando en términos de rendimiento computacional.
Básicamente si un segmento de código se ejecuta en menos tiempo del que se emplea en transmitir su resultado (latencia), entonces será más rápido ejecutar dicho segmento de código de manera secuencial en el procesador que necesita dicho resultado; de este modo la ejecución secuencial evitaría la sobrecarga en la comunicación”[31].

1.4 TÉRMINOS MINERÍA DE DATOS

- **Base de datos**

“Es un conjunto de elementos de datos que se describen a sí mismos y a sus relaciones, que presentan una interfaz uniforme a los usuarios”[32].

- **Minería de datos**

“Es el proceso de extraer conocimiento útil y comprensible, previamente desconocido, desde grandes cantidades de datos almacenados en distintos formatos”[33].

- **Algoritmo de minería de datos**

“Es el mecanismo que crea un modelo de minería de datos. Para crear un modelo, un algoritmo analiza primero un conjunto de datos y luego busca patrones y

[30] REYES, Op. CIt.
[31] REYES, Op. Cit.
[32] JOHNSON, James. Bases de Datos Modelos, lenguajes, diseño. México: Oxford University Press, 2000.
[33] MARTÍNEZ, Javier. ¿Qué es la Minería de Datos?. [en línea]. Disponible en web: (http://www.hotinnovacion.es/uploads/EDMANS_DATAMINING_Que_es_la_Mineria_de_Datos.pdf)

tendencias específicos. El algoritmo utiliza los resultados de este análisis para definir los parámetros del modelo de minería de datos. A continuación, estos parámetros se aplican en todo el conjunto de datos para extraer patrones procesables y estadísticas detalladas"[34].

- **Asociación**

"Las reglas de asociación detectan eventos asociados que se ocultan en las bases de datos. Este tipo de análisis a menudo se utiliza para encontrar combinaciones populares de productos.
Las reglas de asociación generan un conjunto de pares A-B con una probabilidad n%"[35].

- **Clasificación**

"Se trata de obtener un modelo que permita asignar un caso de clase desconocida a una clase concreta (seleccionada de un conjunto predefinido de clases)"[36].

- **Agrupación (Clustering)**

"Es la detección de grupos de individuos. Se diferencia de la clasificación en el que no se conocen ni las clases ni su número (aprendizaje no supervisado), con lo que el objetivo es determinar grupos o racimos diferenciados del resto)"[37].

- **KDD (Knowledge Discovery in Databases)**

"Se ha definido como la extracción no trivial de información potencialmente útil a partir de un gran volumen de datos en el cual la información está implícita (aunque no se conoce previamente). Se trata de interpretar grandes cantidades de datos y encontrar relaciones o patrones"[38].

- **Validación cruzada**

---

[34] MICROSOFT TECHNET. Algoritmos de minería de datos (Analysis Services: Minería de Datos). [en línea]. Disponible en web: (http://technet.microsoft.com/es-es/library/ms175595.aspx).
[35] ORACLE. Oracle Data Mining, Reglas de Asociación. [en línea]. Disponible en web: (http://www.oracle.com/global/es/database/docs/oracle_data_mining.pdf).
[36] AGRAWAL, Rakesh. Data Mining & Knowledge Discovery in Databases (KDD). [en línea]. Disponible en: (http://elvex.ugr.es/etexts/spanish/kdd/KDD.html).
[37] HERNÁNDEZ, José. El Proceso de KDD. [en línea]. Disponible en web: (http://elvex.ugr.es/etexts/spanish/kdd/KDD.html).
[38] Ibíd.

"La validación cruzada le permite particionar un conjunto de datos en muchas secciones transversales de menor tamaño y crear varios modelos en dichas secciones para probar la validez del conjunto de datos completo"[39].

1.5 TÉRMINOS COMPLEMENTARIOS

- **UML**

"El Lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación.
UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real"[40].

- **Diagrama de clases**

"Un diagrama de clase muestra un conjunto de clases, interfaces, colaboraciones y sus relaciones. Este diagrama es el que más se encuentra en los sistemas de modelado orientado a objetos. Los diagramas de clase dirigen la visión de diseño estática de un sistema"[41].

- **Diagrama de caso de uso**

"Un diagrama de caso de uso muestra un conjunto de casos de uso y actores (un tipo especial de clase) y sus relaciones. Los diagramas de casos de uso dirigen la visión de caso de uso estática de un sistema. Estos diagramas son importantes a la hora de organizar y modelar los comportamientos de un sistema"[42].

- **Diagrama de despliegue**

"Un diagrama de despliegue muestra la configuración de los nodos que se procesan en tiempo de ejecución y los componentes que están dentro de ellos. Los diagramas de despliegue dirigen la visión de despliegue estática de una

---

[39] MICROSOFT TECHNET. Validar modelos de minería de datos (Analysis Services - Minería de datos). [en línea]. Disponible en web: (http://technet.microsoft.com/es-es/library/ms174493.aspx).

[40] POPKIN SOFTWARE AND SYSTEMS. Modelado de Sistemas con UML. [en línea]. Disponible en web: (http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf).

[41] OTERO, Mari. Introducción al UML. [en línea]. Disponible en web: (http://www.vc.ehu.es/jiwotvim/IngenieriaSoftware/Teoria/BloqueII/UML-1.pdf).

[42] Ibíd.

arquitectura. Estos diagramas se relacionan con los diagramas de componente en el sentido de que un nodo encierra, normalmente, uno o más componentes"[43].

1.6 CONCEPTO CLUSTER

El concepto de cluster está relacionado con la unión de computadores de escritorio que tengan hardware similar y que se comporten de manera que pareciera un solo computador con ayuda de un software especial. Están encaminados a resolver problemas de diferentes tipos de ciencias como lo son las ingenierías, ciencias naturales y económicas.

"La tecnología de clusters ha evolucionado en apoyo de actividades que van desde aplicaciones de supercómputo y software de misiones criticas, servidores web y comercio electrónico, hasta bases de datos de alto rendimiento, entre otros usos.
El cómputo con clusters surge como resultado de la convergencia de varias tendencias actuales que incluyen la disponibilidad de microprocesadores económicos de alto rendimiento y redes de alta velocidad, el desarrollo de herramientas de software para computo distribuido de alto rendimiento, así como la creciente necesidad de potencia computacional para aplicaciones que la requieran"[44].

Especificaciones de un cluster:

- ✓ Un cluster está creado por dos o más nodos.
- ✓ Los nodos está conectados entre sí por al menos un canal de comunicación.
- ✓ Los clusters necesitan de software especializado para realizar diferentes funciones[45].

1.7 TIPOS DE CLUSTER

Los cluster se pueden clasificar según las funciones que estos desempeñen, es por eso que se los clasifica de la siguiente forma.

[43] Ibíd.
[44] SIECAR. Cluster Computacional. [en línea]. Disponible en web: (http://siecar.upbbga.edu.co/areas.html).
[45] CASTILLO, Op. Cit.

- **Cluster de alta disponibilidad HA (High Availability).** Los clusters de alta disponibilidad son aquellos que ante fallos de hardware, son capaces de seguir funcionando. Se basan en redundancia; esto es, que poseen uno o más nodos que pueden realizar las tareas de otro de los nodos, en caso de que deje de funcionar. De esta manera serán capaces de continuar funcionando ante indisponibilidades de hardware[46].

- **Cluster de balanceo de carga LB (Load Balancing).** "Técnica muy utilizada para lograr que un conjunto de servidores de red compartan la carga de trabajo y con ello el tráfico de sus clientes. Este proceso de dividir la carga de trabajo entre los servidores reales permite obtener un mejor tiempo de acceso a las aplicaciones y con ellos tener una mejor confiabilidad del sistema. Además como es un conjunto de servidores el que atiende el trabajo, la falla de uno de ellos no ocasiona una falla total del sistema ya que las funciones de uno, las puede suplir el resto"[47].

- **Cluster de alto rendimiento HP (High Performance).** Este proyecto se centra principalmente en este tipo de cluster, el cual se comporta de manera muy compenetrada en la interacción de sus componentes hardware para poder obtener el mayor rendimiento en la resolución de tareas asignadas, se realiza una administración común para todos los nodos[48].

  Este tipo de cluster es utilizado en aplicaciones donde se requiere gran capacidad de procesamiento computacional, la cual soluciona problemas complejos mediante la utilización de técnicas necesarias para la paralelización de la aplicación, distribución de los datos a los nodos, la obtención y presentación de resultados finales[49].

1.8 APLICACIONES COMUNES DE LOS CLUSTER HP

En el mundo de hoy se presentan problemas de gran complejidad, debido a esto se requiere más potencial y mayor capacidad en la solución de este tipo de problemas, que se presentan en diferentes áreas académicas e investigativas, es

[46] MARTÍNEZ, Ignacio. Creación y Validación de un Cluster de Computación Científica Basado en Rocks. [en línea]. Disponible en web: (http://e-archivo.uc3m.es/bitstream/10016/5871/1/PFC_Ignacio_Martinez_Fernandez.pdf).
[47] CATALÁN, Op. Cit.
[48] ACOSTA, Ricardo et al. Implementación de un CLUSTER de alto rendimiento como herramienta para resolver problemas de cómputo científico. [en línea]. Disponible en web: (http://docente.ucol.mx/~mgarcia/cluster.pdf).
[49] BUSTOS, Andrés. Configuración de un cluster de alta disponibilidad y balanceo de carga en Linux para satisfacer gran demanda web y servicios de resolución de nombres.Ecuador. 2007. Escuela Politécnica Nacional.

allí donde la Tecnología Cluster ofrece una alternativa para la solución de problemas diversos. Se puede mencionar los siguientes campos en los que tiene cabida este tipo de tecnología:

1.8.1 Investigación en ciencias naturales. "Las moléculas de proteínas son cadenas largas y flexibles que pueden asumir un número virtualmente infinito de formas en 3D. En la naturaleza, cuando se las ubica en un solvente, rápidamente se doblan a sus estados de origen. Se cree que el plegamiento incorrecto lleva a una variedad de enfermedades como Alzheimer; por lo tanto, el estudio del plegamiento de proteínas es de fundamental importancia.

Una forma que los científicos tratan de entender el plegamiento de proteínas es mediante la simulación en computadores. En la naturaleza, el plegamiento ocurre rápidamente (aproximadamente una millonésima de segundo), pero es tan compleja que su simulación en un computador regular podría tomar décadas. Esta área de enfoque es pequeña en una industria con muchos más ámbitos, pero se necesita gran potencia computacional"[50].

1.8.2 Procesamiento de gráficos. El manejo de gráficos que posean alta resolución, siempre ha representado un desafío en cuanto a obtener un mayor rendimiento y escalabilidad sobre estos, ya que contienen muchos datos en su estructura.

Con la aparición de la tecnología cluster, surgió una técnica apropiada para resolver problemas que tengan que ver con la generación de imágenes de alta gama. Esta se basa principalmente en la fragmentación del archivo imagen que se quiera renderizar. Y cada fragmento es asignado a un nodo para que este lo renderize, y luego estas partes sean unidas en el nodo maestro. Para mostrar la totalidad de la imagen generada en un menor tiempo[51].

"La generación de imágenes fotorrealistas con características 3D, es hoy un problema muy común en sectores como el diseño gráfico, el diseño industrial, la publicidad y el cine. Basta reconocer como un ejemplo inmediato la popularidad que en años recientes han tenido películas animadas con simulaciones ultra realistas de personajes y escenas complejas, o el uso de la síntesis de imágenes de alta calidad y resolución para representar de forma realista sitios o eventos no existentes, difíciles de acceder, en negocios como diseño industrial, propiedad raíz, publicidad o cultura"[52].

[50] ADITYA, Narayan. High-performance Linux clustering, Part 1: Clustering fundamentals. [en línea]. Disponible en web: (http://www.ibm.com/developerworks/linux/library/l-cluster1/).
[51] Ibíd.
[52] BEDOYA, Diego et al. ZeBrA-Core: Una Herramienta para la Generación de Imágenes

1.8.3 Procesamiento de datos. La Minería de Datos y el Descubrimiento de Conocimiento abarca un campo interdisciplinario que unifica conceptos de la estadística, del aprendizaje automático, bases de datos y de la computación paralela y distribuida. Esta fusión se debe fundamentalmente al fenomenal crecimiento de los datos en todas las esferas de la actividad humana y a la necesidad económica y científica de extraer información valiosa de los datos recolectados, es por ello, que si en un principio el reto principal de la Minería de Datos era la extracción de conocimientos en bases de datos, hoy no es tan sólo eso, sino en general sobre volúmenes de datos muy grandes.

"Una de las cuestiones más importantes de la Minería de Datos y del Descubrimiento de Conocimiento es lograr que los algoritmos, las aplicaciones y los sistemas, sean escalables ante grandes volúmenes de datos, es decir trabajen con la misma eficiencia ante el aumento del tamaño y la dimensionalidad de los datos, pero como bien se plantea, alcanzar este resultado constituye un reto. Para lograr que un algoritmo de minería de datos sea escalable ante grandes volúmenes de datos, muchas aplicaciones y sistemas utilizan como tecnología, lo que se conoce por el término de computación de Alto Rendimiento (High Performance Computing) o computación paralela"[53].

Pero no sólo en esto son las aplicaciones en las que puede trabajar un cluster, existen muchas otras las cuales solo se mencionan pero tienen mucha relevancia en el mundo actual: simulación astrofísica, simulación del clima, el diseño de ingeniería, modelación financiera, simulación de carteras, y los efectos especiales en películas demandan gran uso de recursos computacionales. La demanda de incrementar potencia está aquí para quedarse[54].

1.9 ARQUITECTURA DE CLUSTER

Para la conformación de un cluster, este debe estar formado por nodos conectados a través de una red de comunicación.
La mayoría de los cluster están formados por computadores convencionales, que cuentan con su propia memoria.
La distribución de los componentes hardware de cluster puede estar sujeta a que todos los componentes se encuentren en una sola parte, esta forma sería a través

---

Fotorrealistas Usando POV-Ray en Ambientes de Computación Distribuida. [en línea]. Disponible en web: (http://www.saber.ula.ve/bitstream/123456789/16021/1/bedoya.pdf).

[53] HERNÁNDEZ, José. Estado Actual de la Aplicación de la Computación Paralela a la Minería de Datos. 2004. Cuba.

[54] ADITYA, Op. Cit.

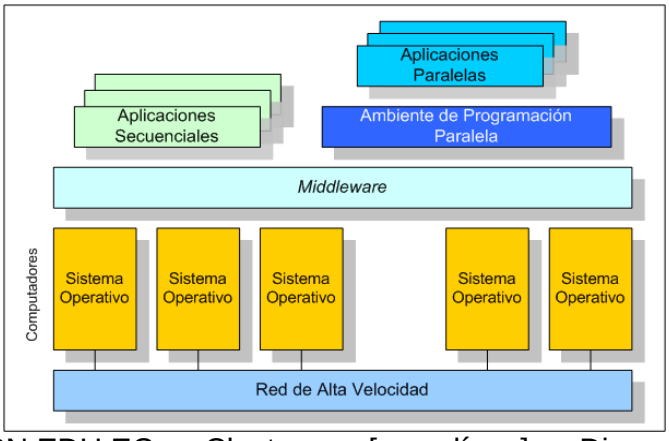
de un rack o pueden estar físicamente separados pero unidos a través de una red local.

Los elementos esenciales en un sistema cluster son los siguientes:

- Grupo de computadores.
- Canal de comunicación para conectar los diferentes nodos.
- Tarjetas de red que soporten tecnologías avanzadas por ejemplo: Fast Ethernet ó Myrinet.
- Protocolos y servicios de comunicación de alta velocidad.
- Middleware, compuesto de dos subniveles de software:
    - La imagen de sistema única (SSI: Single System Image) que ofrece a los usuarios un acceso unificado a todos los recursos del sistema.
    - Disponibilidad del sistema que permite servicios como puntos de chequeo, recuperación de fallos, soporte para tolerancia a fallos.
- Entornos y herramientas de programación paralela, compiladores paralelos, Java, PVM, MPI.[55]

En Figura 1, se muestra la forma de organización del software en un cluster.

Figura 1. Arquitectura de un cluster



CLUSTERFIE.EPN.EDU.EC. Clusters. [en línea]. Disponible en web: (http://clusterfie.epn.edu.ec/clusters/Definiciones/definiciones.html).

1.10 ACOPLAMIENTO DE UN CLUSTER

[55] SÁNCHEZ, Alberto et al. Tema 2: Arquitecturas de memoria compartida y multicore. [en línea]. Disponible en web: (http://dac.escet.urjc.es/docencia/LAAC/LAAC_Tema3.pdf).

A partir de la unificación que tenga el software con los nodos, se pude determinar si el cluster está más o menos acoplado.
"Se entiende por acoplamiento del software a la integración que tengan todos los elementos software que existan en cada nodo. Gran parte de la integración del sistema la produce la comunicación entre los nodos"[56].

- **Acoplamiento fuerte**. "El software que entra en este grupo es software cuyos elementos se interrelacionan mucho unos con otros y posibilitan la mayoría de las funcionalidades del cluster de manera altamente cooperativa. El caso de acoplamiento más fuerte que se puede dar es que solamente haya una imagen del kernel del sistema operativo, distribuida entre un conjunto de nodos que la compartirán.
  Si arranca o ejecuta una aplicación, esta vería un sistema homogéneo, por lo tanto los kernels tienen que conocer los recursos de otros nodos para presentarle al sistema local los recursos que encontraría si estuviera en otro nodo"[57]. Por supuesto se necesita un sistema de nombres único, manejado de forma distribuida o centralizada y un sistema de monitoreo de recursos.

- **Acoplamiento medio.** "A este grupo pertenece un software que no necesita un conocimiento tan exhaustivo de todos los recursos de otros nodos, pero que sigue usando el software de otros nodos para aplicaciones de muy bajo nivel"[58].
  Por ejemplo, en un cluster openMosix es obligatorio que todos los kernels sean de la misma versión. Por otro lado, no está tan acoplado como el caso anterior: no necesita un sistema de nombres común en todos los nodos, y su capacidad de dividir los procesos en una parte local y otra remota consigue que por un lado se necesite el software del otro nodo donde está la parte del fichero que falta en el nodo local y por otro que no se necesite un SSI para hacer otras tareas.

- **Acoplamiento débil.** "Generalmente se basan en aplicaciones construidas por bibliotecas preparadas para aplicaciones distribuidas.
  Es el caso de por ejemplo PVM, MPI o CORBA. Éstos por sí mismos no funcionan hay que dotarles de una estructura superior que utilice las capacidades del cluster para que éste funcione"[59].

1.11 HOMOGENEIDAD DE UN CLUSTER

[56] CATALÁN, Op. Cit.
[57] Ibíd.
[58] Ibíd.
[59] Ibíd.

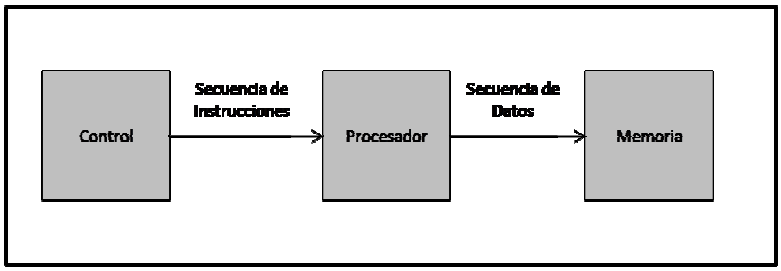
Cuando todos los computadores que componen el cluster poseen el mismo hardware e igual software son de tipo homogéneo, pero si estos computadores difieren en sus características se catalogan como heterogéneos[60].

1.12 ARQUITECTURA DE COMPUTADORES

- **Taxonomía de Flynn.** La taxonomía de Flynn es la manera clásica de organizar los computadores, de acuerdo al número de instrucciones y la secuencia de datos que utiliza para procesamiento. Las instrucciones y las secuencias de los datos son sencillas y múltiples[61].

- **SISD (Single Instruction Single Data).** Es un tipo de procesador que ejecuta las instrucciones secuencialmente, está diseñado para dar soporte al procesamiento secuencial clásico, basado en el intercambio de datos entre memoria y registros del procesador, y la realización de operaciones aritméticas en ellos[62]. Su estructura se observa en Figura 2.

Figura 2. Modelo SISD



CASTILLO, José et al. Implementación de un cluster OpenMosix para cómputo científico en el Instituto de Ingeniería. México, 2006. Universidad Nacional Autónoma de México. Facultad de Ingeniería.

- **SIMD (Single Instruction Multiple Data).** El modelo SIMD consiste de varias unidades de procesamiento bajo el control de una unidad de control.

[60] MARTÍNEZ, Op. Cit.
[61] CASTILLO, Op. Cit.
[62] FING.EDU.UY. Computación de Alta Performance Curso 2009 Arquitecturas Paralelas. [en línea]. Disponible en web: (http://www.fing.edu.uy/inco/cursos/hpc/material/clases/Clase2-2009.pdf).

Todos los procesadores reciben la misma instrucción de la unidad de control, con la diferencia que operan sobre diferentes conjuntos de datos. La unidad de memoria compartida contiene módulos especiales para que pueda comunicarse con todos los procesadores simultáneamente[63].

"En SIMD existe una única unidad de control que busca, decodifica y entrega una sola instrucción a un conjunto de unidades de proceso"[64].

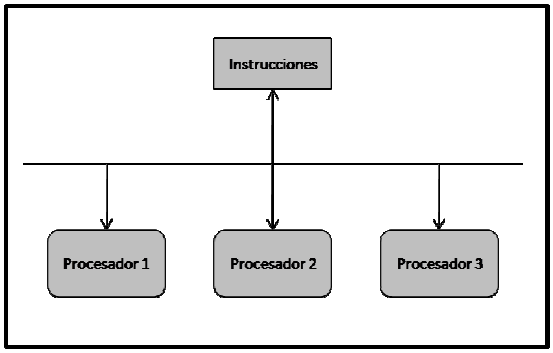
Este tipo de procesador cuenta con un reloj global para que los procesadores estén sincronizados, "es decir, en cada paso todos los procesadores ejecutan la misma instrucción, cada uno con diferentes datos, ejemplo: sumando dos matrices A + B = C, siendo A y B de orden 2 y teniendo 4 procesadores.

A11 + B11 = C11, A12 + B12 = C12
A21 + B21 = C21, A22 + B22 = C22

La misma instrucción se ejecuta en los 4 procesadores (sumando dos números) y los 4 ejecutan las instrucciones simultáneamente. Esto toma un paso en comparación con cuatro pasos en máquina secuencial"[65].

En Figura 3, se puede observar la estructura de una máquina SIMD.

Figura 3. Modelo SIMD



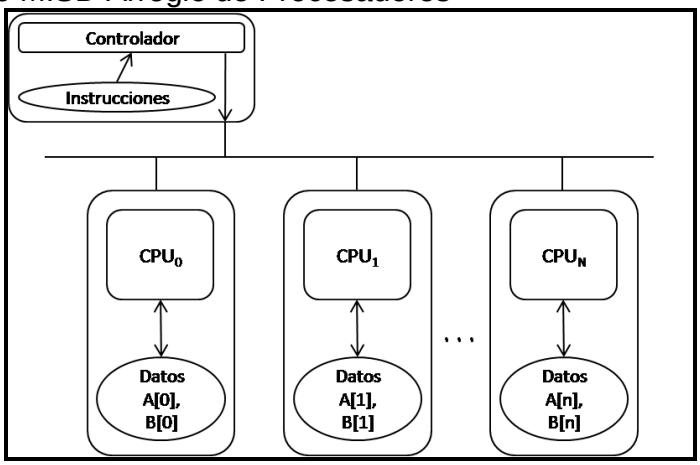
CASTILLO, José et al. Implementación de un cluster OpenMosix para cómputo científico en el Instituto de Ingeniería. México, 2006. Universidad Nacional Autónoma de México. Facultad de Ingeniería.

[63] PERALES, Víctor. Arquitectura Paralela. [en línea]. Disponible en web:(http://www.monografias.com/trabajos16/arquitectura-paralela/arquitectura-paralela.shtml).
[64] SÁNCHEZ, Op. Cit.
[65] CASTILLO, Op. Cit.

- **MISD (Multiple Instruction Single Data).** "Varias unidades funcionales ejecutan diferentes operaciones sobre el mismo conjunto de datos. Las arquitecturas de tipo pipeline (basada en filtros) pertenecen a esta clasificación aunque no puramente, debido a que pueden modificar los datos sobre los que operan. Los modelos MIMD y SIMD son más apropiados para la aplicación del paralelismo tanto a nivel de datos como de control"[66]. En Figura 4, se muestra el modelo MISD.

  Los arreglos sistólicos son un ejemplo de un conjunto de procesadores dispuestos de una manera regular (por lo general rectangular) donde los datos fluyen de forma sincronizada a través de los equipos vecinos[67]. Se puede observar claramente la organización de un arreglo sistólico en Figura 5.

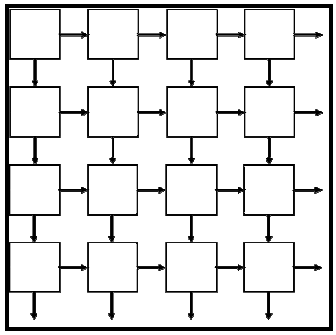
Figura 4. Modelo MISD Arreglo de Procesadores



CASTILLO, José et al. Implementación de un cluster OpenMosix para cómputo científico en el Instituto de Ingeniería. México, 2006. Universidad Nacional Autónoma de México. Facultad de Ingeniería.

Figura 5. Arreglo sistólico

[66] FING.EDU.UY, Op. Cit.
[67] MERY, Domingo. Arquitectura de Computadores Multiprocesador y Arquitecturas Alternativas. [en línea]. Disponible en web: (http://dac.escet.urjc.es/docencia/LAAC/LAAC_Tema2-A.pdf).



MERY, Domingo. Arquitectura de Computadores Multiprocesador y Arquitecturas Alternativas. [en línea]. Disponible en web: (http://dac.escet.urjc.es/docencia/LAAC/LAAC_Tema2-A.pdf).

- **MIMD (Multiple Instruction Multiple Data**). Es un tipo mejorado de SIMD, que contiene varios procesadores que pueden trabajar de forma independiente sobre diferentes datos, no cuenta con un reloj central, en otras palabras operan asincrónicamente[68].

  El hecho de trabajar sobre diferentes datos se puede decir que es una característica muy fuerte.
  Dentro de los computadores de tipo MIMD están:

  Memoria compartida

  - UMA (Uniform Memory Access)

  Memoria distribuida

  - NUMA (Non Uniform Memory Access)

- **UMA (Uniform Memory Access).** Estos sistemas se caracterizan porque todos los procesadores tienen acceso a la memoria principal, comparten un espacio de direccionamiento compartido. El acceso a memoria se hace mediante un solo canal. Es importante que los procesadores no tengan acceso al mismo tiempo a regiones de memoria de manera que ocurra algún error. En Figura 6, se puede observar la estructura del modelo UMA.

  Ventajas

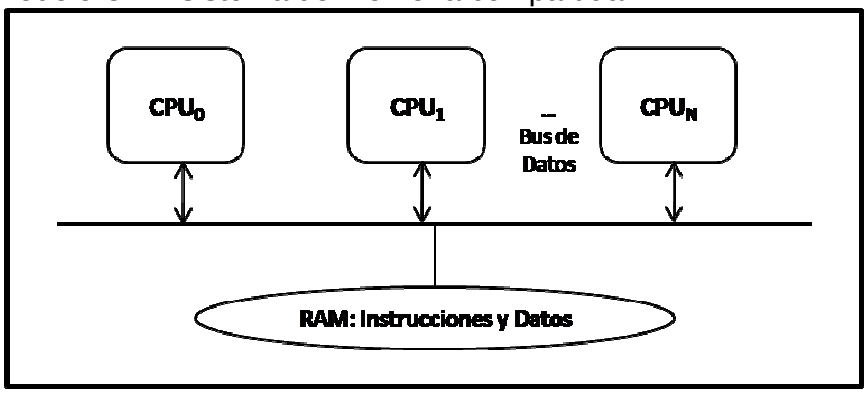
[68] FING.EDU.UY, Op. Cit.

- ✓ La programación es más sencilla en este tipo de sistemas en comparación con los sistemas de memoria distribuida[69].
- ✓ "Es una buena solución para el procesamiento transaccional"[70].

Desventajas

- ✓ Un fallo en memoria, afecta todo el sistema.
- ✓ Si se incrementan los procesadores puede llevar a cabo problemas en el acceso a memoria.
- ✓ Se limita la escalabilidad a un máximo de pocas decenas de procesadores[71].

Figura 6. Modelo UMA sistema de memoria compartida



CASTILLO, José et al. Implementación de un cluster OpenMosix para cómputo científico en el Instituto de Ingeniería. México, 2006. Universidad Nacional Autónoma de México. Facultad de Ingeniería.

- **NUMA (Non Uniform Memory Access).** En estos sistemas cada procesador cuenta con su propia memoria. Para intercambiar información lo hacen mediante el envío de mensajes, es decir, "si un procesador requiere los datos contenidos en la memoria de otro procesador, debería enviar un mensaje solicitándolos. A esta comunicación se le conoce como paso de mensajes"[72].
  El modelo NUMA se muestra en Figura 7.

  Ventajas

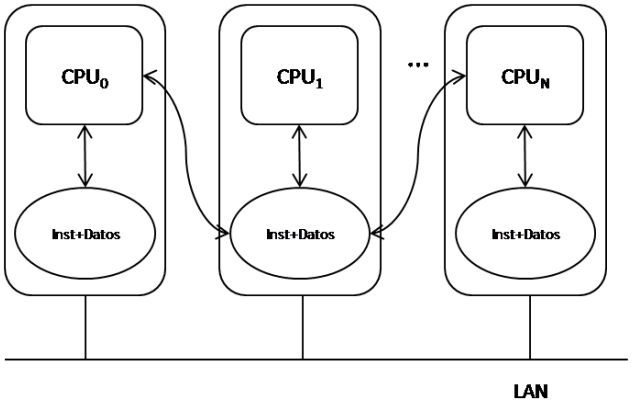
[69] CASTILLO, Op. Cit.
[70] FING.EDU.UY, Op. Cit.
[71] CASTILLO, Op. Cit.
[72] CASTILLO, Op. Cit.

- ✓ Es fácilmente escalable, se pueden ir aumentando procesadores según lo necesario.
- ✓ El fallo de un procesador no afecta a los demás.

Desventajas

- ✓ La comunicación puede convertirse en cuello de botella, esto depende en gran parte de la topología de red.
- ✓ El acceso a memoria remota es lento.
- ✓ La programación suele ser complicada.

Figura 7. Modelo NUMA memoria distribuida



CASTILLO, José et al. Implementación de un cluster OpenMosix para cómputo científico en el Instituto de Ingeniería. México, 2006. Universidad Nacional Autónoma de México. Facultad de Ingeniería.

1.13 ARQUITECTURAS DE COMPUTACIÓN PARALELA

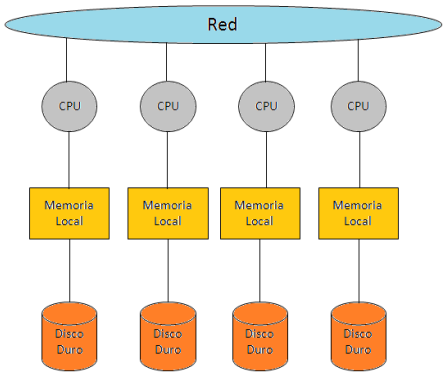
Se describen algunos tipos de computadores que se utilizan para el trabajo en computación paralela.

- **DMM (Distributed Memory Arquitecture).**

  - ✓ Sin compartir.

    En Figura 8, se muestra la organización de una máquina con memoria distribuida que no comparte ningún elemento de hardware.

Figura 8. Arquitectura DMM sin compartir

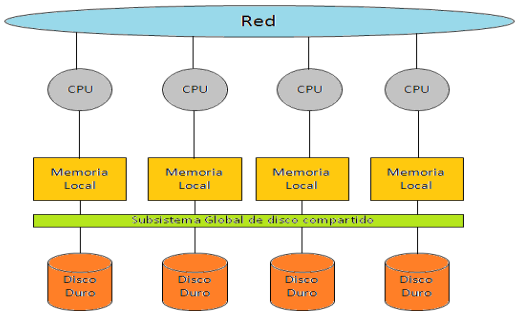


HERNÁNDEZ, José. Estado Actual de la Aplicación de la Computación Paralela a la Minería de Datos. 2004. Cuba.

- ✓ Disco compartido.

  En Figura 9, se muestra una arquitectura de máquina con memoria distribuida y disco compartido, mediante un sistema de archivos por red.

Figura 9. Arquitectura de disco compartido

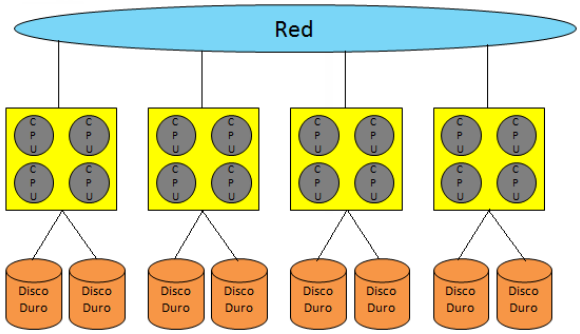


HERNÁNDEZ, José. Estado Actual de la Aplicación de la Computación Paralela a la Minería de Datos. 2004. Cuba.

- **Clusters de SMPs**

  En Figura 10, se muestra la disposición de los elementos hardware en un cluster de SMPs.

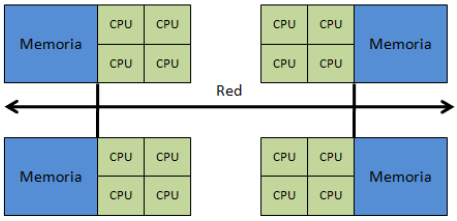
Figura 10. Cluster de SMPs



HERNÁNDEZ, José. Estado Actual de la Aplicación de la Computación Paralela a la Minería de Datos. 2004. Cuba.

- **Máquinas de memoria compartida distribuida.** "Los componentes de memoria compartida son usualmente maquinas SMP.
  El componente distribuido lo proporciona una interconexión de red entre las máquinas SMP"[73]. La organización de este tipo de máquinas se muestra en Figura 11.

Figura 11. Máquinas de memoria compartida distribuida



JORGE, Javier et al. Introducción al Clustering con MPI. [en línea]. Disponible en web:(http://www.efn.uncor.edu/escuelas/computacion/files/Introducci%C3%B3n%20al%20clustering%20con%20MPI.pdf).

- **Cluster de PC.** Un cluster es un conjunto de computadores interconectados mediante una red de alta velocidad los cuales se comportan como si fuesen una sola máquina con una capacidad de cómputo superior. Actualmente los clusters son muy importantes en aplicaciones científicas, de ingeniería, comerciales, simulaciones, etc.

[73] JORGE, Javier et al. Introducción al Clustering con MPI. [en línea]. Disponible en web: (http://www.efn.uncor.edu/escuelas/computacion/files/Introducci%C3%B3n%20al%20clustering%20con%20MPI.pdf).

"La tecnología de clusters ha evolucionado apoyándose en actividades que van desde aplicaciones de supercómputo y software de misiones críticas, servidores Web y comercio electrónico, hasta bases de datos de alto rendimiento. El uso de clusters surge gracias a la convergencia de varias tendencias actuales como la disponibilidad de microprocesadores económicos de alto rendimiento y redes de alta velocidad, la existencia de herramientas para cómputo distribuido de alto rendimiento, así como la creciente necesidad de potencia computacional para aplicaciones que la requieran"[74].

1.14 CATEGORÍAS DE CLUSTER

- **Beowulf.** Un cluster del tipo Beowulf posee múltiples computadores y es posible ir agregando otros a medida que se necesiten, se utiliza para entornos de computación paralela y distribuida. Estos sistemas son construidos con componentes de hardware y software de uso general, es decir, no contienen ningún tipo de hardware especializado. Los nodos son capaces de ejecutar el sistema operativo Linux, y software para el desarrollo de aplicaciones paralelas o distribuidas. En su forma general, un cluster Beowulf consiste de un nodo servidor y al menos uno o varios nodos de cómputo, interconectados a través de una LAN. El nodo servidor controla los nodos cliente y también es la puerta de entrada al cluster desde el mundo exterior.

  Los nodos del sistema se usan únicamente para cómputo. Los clusters Beowulf se comportan como una unidad. En la mayoría de los casos, los nodos cliente no poseen dispositivos de entrada como teclados, mouse o dispositivos de salida como el monitor, se accede a ellos desde el nodo servidor. Así, los nodos cliente, representan un componente que posee una unidad central de procesamiento para cómputo y cierta cantidad de memoria que se anexa al sistema. No existe necesidad de que los nodos clientes sean accedidos directamente por sistemas externos, ni que éstos accedan al exterior[75].

- **NOW/COW (Networks/Cluster Of Workstations).** Difieren físicamente de Beowulf, ya que son esencialmente PCs conectados a través de una red. En la mayoría de los casos COW son utilizados para cálculos en paralelo

[74] ITURRIAGA, Santiago. Proyecto Fenton - Cluster de Computadores de Alto Desempeño con Acceso Remoto. [en línea]. Disponible en web: (http://pgruso2007.googlecode.com/files/PGCCADAR-Informe.pdf).
[75] PEROZO, Op. Cit.

en la noche y los fines de semana cuando las personas no están realmente utilizando las estaciones de trabajo para su trabajo diario.
"Otro ejemplo de este tipo de redes es cuando las personas están utilizando su PC, pero el servidor utiliza una porción de su potencia de procesamiento global y agrega esto a través de la red, por lo tanto utiliza ciclos de ocio de CPU. En este caso, la programación de una NOW es un intento de cosecha de los ciclos no utilizados en cada estación de trabajo. La programación en este entorno requiere algoritmos que son extremadamente tolerantes a los problemas de balanceo de carga y latencia de las comunicaciones de gran tamaño. Cualquier programa que se ejecuta en una NOW se ejecutará al menos tan bien como en un clúster"[76].

1.15 PROCESAMIENTO EN PARALELO

El concepto de procesamiento en paralelo tiene que ver con la eficiencia a la hora de ejecutar cierto programa, segmentándolo en varios fragmentos, para que puedan ser trabajados por separado por un procesador distinto y ejecutarse de forma simultánea, de tal manera que se acelera la realización del programa

Por lo tanto, un programa que se ejecute en n procesadores, podría realizarse n veces más rápido que utilizando un solo procesador.

Aplicaciones con suficiente paralelismo como para hacer bueno el uso de múltiples procesadores. El problema radica en identificar las porciones de programa que puedan ser ejecutadas independiente y simultáneamente en procesadores separados; esto en realidad es complejo, ya que se encontrarán aplicaciones que podrían ser ejecutadas en paralelo y sin embargo se ralentizan al ser paralelizadas en un sistema particular. Por ejemplo, un programa que tarda 4 segundos en ser ejecutado en una sola máquina podría tardar 1 segundo de procesamiento en cada uno de los cuatro procesadores disponibles en una red, pero no se lograría nada si la coordinación entre dichos procesadores tardase más de 3 segundos.
Implementaciones de algoritmos que son paralelos (escritos para obtener las ventajas del procesamiento paralelo) o bien se espera paralelizar, codificando de nuevo al menos alguna de sus partes.

El procesamiento paralelo puede proporcionar el rendimiento de un supercomputador, aplicado a algunos programas que realizan complejas operaciones u operan en grandes bloques de datos. Y lo que es más, ello se puede lograr con hardware relativamente barato. Además es posible utilizar dichos

[76] MORRISON, Richard. Cluster Computing Architectures, Operating Systems, Parallel Processing & Programming Languages. [en línea]. Disponible en web: (www.ace.ual.es/~jaberme/docsppal/cluster/CLSTR_CMPTNG_THEORY.pdf).

sistemas paralelos para la realización de otros trabajos cuando no estén ocupados con una tarea en paralelo[77].

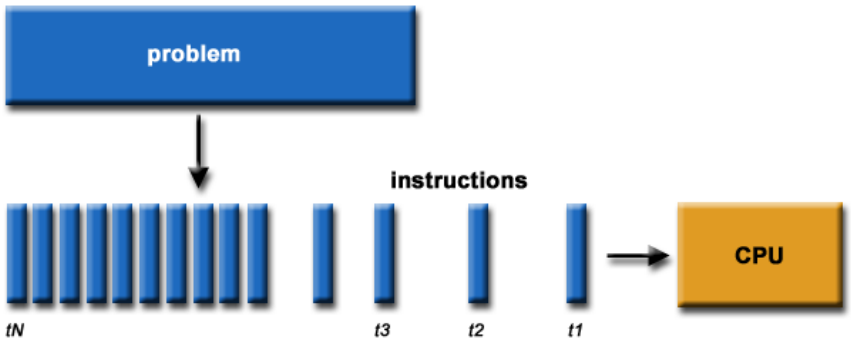
1.16 MODELO GENERAL DE COMPUTACIÓN EN PARALELO

La programación en paralelo tiene sus propias abstracciones que son primordiales para la construcción y el diseño de un algoritmo o una solución.
Los objetos básicos de ese modelo de abstracción son las tareas, los canales y los mensajes.
Todo programa o solución en paralelo puede considerarse como un conjunto de tareas que se ejecutan concurrentemente y que se conectan a través de canales por los que intercambian mensajes.
Un programa en paralelo puede consistir de un número variable de tareas y canales por los que circulan un número también variable de mensajes[78].

1.17 COMPUTACIÓN SECUENCIAL

La computación secuencial trabaja un problema en un conjunto de instrucciones que son ejecutadas de una en una por un procesador.[79] En Figura 12, se muestra la forma de trabajo de la computación serial.

Figura 12. Computación secuencial



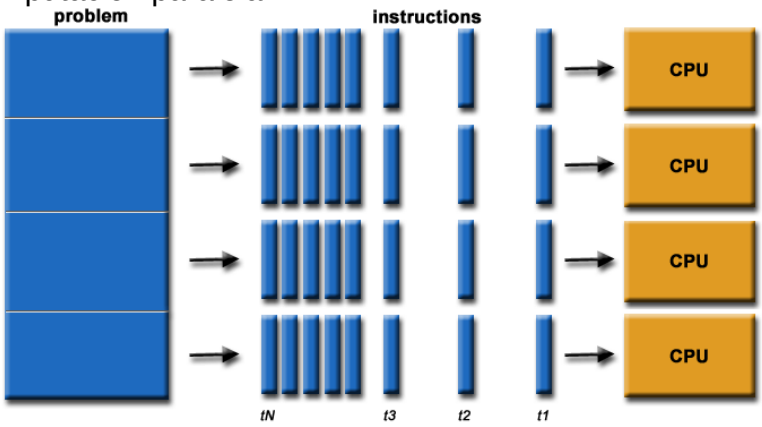
BARNEY,Blaise. Introduction to Parallel Computing. [en línea]. [28 junio de 2010]. Disponible en web: (https://computing.llnl.gov/tutorials/parallel_comp/).

[77] REYES, Op. Cit.
[78] ZULUAGA, Jorge. Introducción a la Programación Paralela - descomposición -. [en línea]. Disponible en web: (http://astronomia.udea.edu.co/sitios/jzuluaga/pages/distos-2009-1-usb.rs/files/distos-2009-1-usbwn3t/distos-presentacion-intro_prog_paralela-descomposicion.pdf).
[79] JORGE, Op. Cit.

1.18 COMPUTACIÓN PARALELA

La computación paralela toma un problema y lo divide en varios conjuntos de instrucciones que van a ser ejecutados por varios procesadores al mismo tiempo. En Figura 13, se muestra la forma de trabajo de la computación paralela.

Figura 13. Computación paralela



BARNEY,Blaise. Introduction to Parallel Computing. [en línea]. [28 junio de 2010]. Disponible en web: (https://computing.llnl.gov/tutorials/parallel_comp/).

1.19 MODELO DE INTERACCIÓN ENTRE PROCESADORES

- **Paso de Mensajes.** "El paso de mensajes es un modelo de interacción entre los procesadores que forman un sistema paralelo. A grandes rasgos un mensaje es un conjunto de datos definido mediante software por un procesador y enviado a través de una red de comunicaciones hacia otro procesador, el cual debe aceptar el mensaje y actuar según su contenido"[80]. Todos los nodos pueden ser origen o destino de mensajes en cualquier momento y todas pueden intercambiar datos entre sí. El enrutamiento de mensajes se le deja al sistema operativo y el número de saltos o puntos de intercomunicación por los que el mensaje debe pasar dependerán de la topología de la máquina en uso. Este sistema puede utilizarse alcanzando un rendimiento óptimo en multicomputadores y en multiprocesadores con memoria distribuida.
  Aunque la sobrecarga en la comunicación (latencia) en cada paso de mensaje puede ser alta normalmente hay pocas restricciones sobre la

[80] REYES, Op. Cit.

cantidad de información que pueden contener dichos mensajes. De este modo, si se cuenta con un ancho de banda amplio el paso de mensajes puede ser un método muy efectivo para transmitir grandes bloques de datos de entre los procesadores.

- **MPI (Message Passing Interface).** Es un estándar que define la sintaxis y la semántica de las funciones contenidas en una biblioteca de paso de mensajes, diseñada para ser usada en programas que exploten la existencia de múltiples procesadores. Puede implementarse tanto en sistemas que utilicen paso de mensajes, como sistemas de memoria compartida.
  La primera versión de MPI fue desarrollada en 1993-1994 por un grupo de investigadores al servicio de la industria, el gobierno y el sector académico. Debido al apoyo recibido MPI es relativamente el nuevo estándar para la programación de procesadores paralelos basado en el paso de mensajes[81].

  Como aplicaciones (implementaciones) de MPI se encuentran:

  - ✓ MPJ Express
  - ✓ LAM/MPI (Local Area Multicomputer/Message Passing Interface)
  - ✓ MPICH

1.20 PROS Y CONTRAS DE LA PROGRAMACIÓN EN PARALELO

Pros

- Disminución del tiempo de ejecución.
- Alternativa para resolver problemas de gran envergadura.
- Utilización apropiada de todos los recursos con que se cuenta así estos no se encuentren en la misma zona.
- Reducción en el costo de los recursos (muchos, muy baratos).
- Superar límites en memoria o capacidad de procesamiento.

Contras

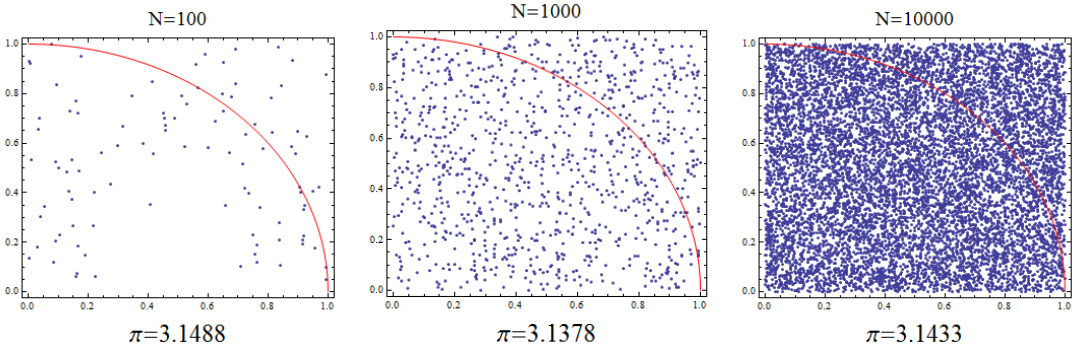
[81] Ibíd.

- Rediseñar los algoritmos, puede ser un gran dolor de cabeza.
- Difícil depuración y realización de pruebas.
- Hasta supeditado a los componentes de red con que se cuente, además del tráfico que este pueda generar[82].

1.21 APROXIMACIÓN A PI UTILIZANDO MONTECARLO

Montecarlo es un método para la solución de problemas utilizando estadística. Dada la probabilidad, que un evento sucederá en ciertas condiciones, un computador puede ser usado para generar esas condiciones repetidamente. "El número de veces que ocurre el evento dividido entre el número de veces que las condiciones son generadas debe ser aproximadamente igual a Pi"[83].
"Por el método de Montecarlo es posible estimar el área que tiene una línea cerrada haciendo muestreos uniformes de puntos en un cuadrado que la contenga. Si se divide la cantidad de puntos que caen en el área sobre la cantidad de puntos muestreados en total esta fracción es igual, aproximadamente, al área de que subtiende la línea cerrada entre el área del cuadrado en el que está inscrita dicha línea"[84]. En Figura 14, se muestra el método de aproximación a Pi con varios intentos.

Figura 14. Aproximación PI mediante Montecarlo



FUENTES, Maikel et al. Metodo Montecarlo. [en línea]. Disponible en web: (http://www.fisica.uh.cu/fisteo/resources/asignaturas/Probabilidades/mc.pdf).

[82] ZULUAGA, Op. Cit.
[83] ANDERSSON, Eve. Calculation of Pi Using the Monte Carlo Method. [en línea]. Disponible en web: (http://www.eveandersson.com/pi/monte-carlo-circle).
[84] FUENTES, Maikel et al. Metodo Montecarlo. [en línea]. Disponible en web: (http://www.fisica.uh.cu/fisteo/resources/asignaturas/Probabilidades/mc.pdf).

1.22 ANTECEDENTES

- **Cluster Universidad de Los Andes.** "El Departamento de Física de Los Andes es pionero en la configuración de clusters y grids en Colombia, nuevos paradigmas de computación distribuida que permiten que el poder de procesamiento de un computador se multiplique por el número de ellos conectados en red.

  Primero cluster luego Grid

  El de Los Andes, aunque pequeño con respecto a otros en el mundo, es hasta ahora el único Grid en Colombia y nació como consecuencia de una experiencia anterior: El clúster, un conjunto de computadores conectados por una red local, que suman sus capacidades de cómputo bajo un mismo sistema operativo y donde todos se encuentran en el mismo lugar. Esto, a diferencia del grid donde los sistemas pueden ser heterogéneos y donde las máquinas pueden estar en diferentes lugares del mundo.

  Los estudiantes de posgrado (Magíster y Doctorado) son, hasta el momento, los que más están aprovechando el clúster porque ejecutan cálculos de gran complejidad. En resumen, este es un súper computador que hoy no se puede conseguir en ninguna parte del mundo como un computador independiente con esta configuración y por sus características costaría muchísimo dinero"85.

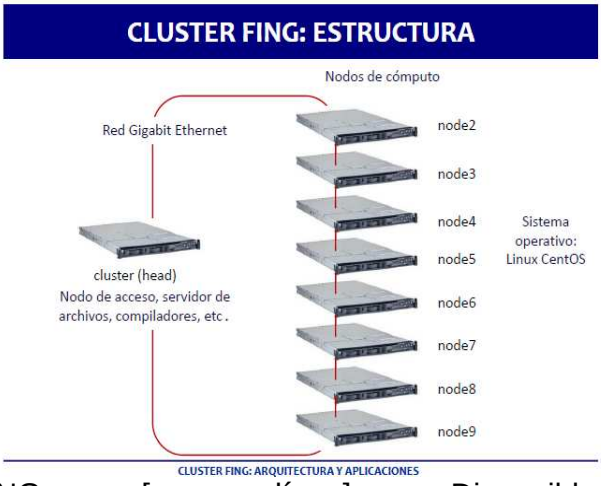
- **Cluster Universidad de Antioquia.** En la Universidad de Antioquia, la distribución Rocks se ha utilizado para el montaje de la mayoría los sistemas de cómputo intensivo, incluyendo su recién adquirido Centro de Simulación y Cálculo Avanzado. "Algunos de esos clusters (especialmente los operados por grupos de investigación particulares) son administrados por personal con una experiencia mínima en la administración de servidores Linux y mucho menos en la administración de sistemas de cómputo distribuido. Aún así esas plataformas pudieron ser instaladas y configuradas sin inconvenientes y son activamente utilizadas en la investigación en Ciencias e Ingeniería"[86].

[85] SOLANO, Franco. Clúster y Grid en el Departamento de Física En red la unión hace la fuerza. [en línea]. Disponible en web: (http://notauniandina.edu.co/html/nota6/agosto2006/pag14-17Clusters.pdf).

[86] ZULUAGA, Jorge. Rocks and Rolls: Sistemas de Computación de Alto Rendimiento de Fácil Despliegue. [en línea]. Disponible en web: (http://www.saber.ula.ve/bitstream/123456789/16036/1/jzuluaga.pdf).

- **Cluster FING (Uruguay).** "El cluster FING es una infraestructura de cómputo de alto desempeño perteneciente a la Facultad de Ingeniería. Su principal objetivo consiste en proveer soporte para la resolución de problemas complejos que demanden un gran poder de cómputo. El cluster FING fue adquirido con fondos del llamado de Fortalecimiento de Equipamientos para la Investigación de la Comisión Sectorial de Investigación Científica (2008)"[87]. En Figura 15 se puede observar la estructura del cluster FING.

Figura 15. Estructura cluster FING



CLUSTER FING. [en línea]. Disponible en web: http://www.fing.edu.uy/cluster/index.php

- **Cluster Mangosta (Venezuela).** "El Cluster Mangosta es una herramienta útil que permitirá a sus usuarios ejecutar sus aplicaciones paralelas y secuenciales de forma eficiente, ofreciendo gran capacidad de cómputo.
  Es una plataforma computacional altamente escalable y económica tanto para su mantenimiento como para su actualización, en relación con otras plataformas computacionales fabricadas por grandes empresas como SGI, IBM, SUN, HP, etc. dado que sus componentes de hardware pueden ser adquiridos en cualquier distribuidor de PCs y el software principal para su funcionamiento es completamente gratuito.
  Los nodos del Cluster Mangosta tienen la siguiente configuración básica:

  - ✓ Procesador de 2,4 GHz, 1Gbyte de memoria RAM, Disco Duro de 40GB@7200 RPM, 2 NICs Fast Ethernet.

[87] CLUSTER FING. [en línea]. Disponible en web: (http://www.fing.edu.uy/cluster/index.php).

- ✓ Sistema operativo RedHAT Linux 8.0, Kernel 2.4.20
- ✓ MPICH versión 1.2.5, versión portable de la librería de paso de mensajes MPI.

- ✓ OpenMosix 3. Extensiones para el Kernel de LINUX 2.4.20.

- ✓ Los nodos están interconectados a través de dos unidades Switch Fast Ethernet de 16 puertos, uno para cada canal"[88].

[88] PEROZO, Op. Cit.

2. DESCRIPCIÓN DE TINKU

2.1 GENERALIDADES

Se realizó un análisis sobre distribuciones de software libre para cluster de alto rendimiento donde se tuvo en cuenta las siguientes distribuciones candidatas, de las cuales se hace un balance según la funcionalidad de cada una de estas con el fin de elegir la más apropiada para el proyecto.

- OpenMosix. La distribución OpenMosix es un parche al núcleo del sistema Linux que otorga compatibilidad con la versión de Linux 2.4.x. OpenMosix es dependiente del kernel, además el proyecto fue cancelado. Aunque existe una versión para núcleo 2.6 pero se encuentra en fase beta. Presenta limitaciones funcionales al momento de migrar procesos.

- PelicanHPC. PelicanHPC es un live CD basado en Debian Lenny que proporciona implementaciones MPI. El inconveniente que se presentó fue incompatibilidad con herramientas de cálculo científico y procesamiento de imágenes. Además como es un live CD no soporta las exigencias de un clúster dedicado para alto rendimiento, porque cuando se apaga el cluster este automáticamente se borra, debido a que no se instala como un sistema operativo primario. La mayor debilidad de PelicanHPC es en el campo de la investigación, no hay la suficiente documentación que compruebe que en la vida real se puede implementar como un cluster de alto rendimiento estable a largo plazo.

Dentro de las opciones estudiadas se encontró una distribución de software que satisfacía las necesidades de este proyecto, es por este motivo se instaló en Tinku la versión de Rocks Chimichanga 5.2, este middleware ofrece características como son:

- Instalación práctica y eficiente.

- Contiene software especializado en monitoreo, control y uso para el cálculo en Ciencias e Ingeniería.

- Constante actualización.

- Existe documentación completa sobre el funcionamiento.

- Compatibilidad con herramientas de propósito específico.
- Alta tolerancia a fallos de un nodo.
- Es altamente escalable, debido a que se pueden agregar nodos sin afectar el desempeño del cluster.
- Facilidad de mantenimiento.
- Se presenta al usuario como una sola máquina.
- Nivel de exigencia en cuanto a conocimientos para la administración del clúster.
- Software completamente gratuito.
- Funciona en arquitecturas x86 y x86_64.

Tinku es un modelo de cluster heterogéneo de alto rendimiento, fuertemente acoplado con Rocks, para apoyar y fomentar investigaciones que requieran de gran capacidad de procesamiento, y a la vez eficiencia de tiempo en entrega de resultados.

En el cluster Tinku, el servidor, de aquí en adelante se denominará frontend, es el único equipo que tiene monitor, teclado y mouse. Es el primero en el cual se instaló el software Rocks, este proceso de instalación se explica más adelante en el capítulo 3 DESARROLLO DE TINKU. El frontend es el encargado de enviar tareas a los demás nodos, almacena los paquetes de instalación para la posterior adición de nodos, configura los nodos para sincronización, crea el sistema de archivos en red.

En la implementación del cluster Tinku se hizo énfasis para que su diseño sea de fácil administración y tenga la cualidad de una excelente escalabilidad para la posterior adición de nodos. De tal manera se consigue un mejor desempeño a la hora de ejecutar las aplicaciones.
La organización del cluster Tinku está determinada de modo que el nodo maestro destine una parte de su poder de cómputo a la administración, mientras que los nodos esclavos se dedican exclusivamente a la resolución de las tareas que necesitan potencia en el procesamiento.
La red desempeña un papel muy importante dentro del desarrollo del proyecto, se consideraron las redes tipo FastEthernet como las más convenientes por su bajo costo y facilidad de implementación.

Se instalaron las siguientes herramientas con las que se trabajaron para alto desempeño. Estas herramientas serán descritas en el capítulo 3 DESARROLLO DE TINKU.

- MPJ Express.
- WEKA-Parallel.
- POV-Ray.
- Zebra-Core.

2.1.1 Arquitectura de Tinku. Es un cluster en el cual la memoria se encuentra de forma distribuida, es decir cada nodo contiene su propia memoria, y se comunican mediante paso de mensajes. Además, ofrece dos formas de acceso a todos los archivos en cualquiera de los nodos debido a que utiliza NFS, el cual crea un sistema de archivos de la cuenta de usuario a través de la red, las modificaciones que se realizan se guardan en el servidor donde residen realmente. El servicio autofs, garantiza que los sistemas de archivos se carguen de forma automática en el momento que el usuario inicie sesión, también se encarga de desmontar el sistema de archivos cuando el usuario no lo está utilizando.

Tinku cumple con la función de escalabilidad, se pueden ir añadiendo más nodos según se requiera, si un procesador falla esto no afecta el desempeño de los demás nodos.

Según las arquitecturas de computación paralela, Tinku es un cluster de PC heterogéneo de alto rendimiento que trabaja con procesadores de 32 bits, el cual se desempeña mejor con aplicaciones que se basen en renderización de imágenes en POV-Ray y algoritmos de clasificación de minería de datos utilizando la herramienta WEKA-Parallel.

2.1.2 Distribución Rocks. Rocks se soporta en la distribución Red Hat Linux la cual es una de las de mayor relevancia en el mundo Linux debido a que cuenta con una gran cantidad de desarrolladores y usuarios que hace que esta distribución este en constante actualización de sus programas, además de poseer bastantes recursos de apoyo.

En un cluster Linux configurado con Rocks existen dos clases de máquinas: el frontend (servidor) donde se realizan las configuraciones, se crean las cuentas de usuario y donde se ejecutan los servicios principales del cluster; y los nodos de computo que son los equipos encargados de realizar efectivamente los trabajos de computo en la plataforma o que se pueden utilizar para almacenar información de manera distribuida.

Rocks utiliza un ingenioso sistema para la instalación sistemática en los nodos llamado Kickstart Lo que hace este sistema es construir un archivo de configuración para la instalación del nodo con los respectivos parámetros para su instalación y posterior configuración, el archivo Kickstart es trasmitido por el frontend por medio de su servidor web de donde el nodo descarga los paquetes para su instalación y configuración.
Además de que cumple la función de detección automática de los componentes hardware que se quieran adherir al cluster.

Rocks incorpora un sistema de monitoreo para la instalación de sus nodos llamado eKV(Ethernet Keyboard Video), el cual permite capturar la información del proceso de instalación que se esté presentado en el momento y de esta manera saber cómo se está desarrollando la instalación.

La facilidad y flexibilidad a la hora de sincronización de los nodos y actualización del sistema en el cluster, es una de sus fortalezas de Rocks ya que este maneja un modelo en el cual a sufrir una actualización de algún paquete el frontend. Este se encarga de distribuir la aplicación a los nodos y estos a su vez se reinstalan nuevamente con la aplicación incluida sin modificar contenidos que se encuentren almacenados en los nodos quedando estos compenetrados con el frontend para continuar con un mejor funcionamiento del cluster.

El sistema Rocks presenta un nuevo concepto llamado Roll. Los roll se pueden definir como una colección de programas informáticos destinados a una tarea específica, los cuales determinan la personalización que tendrá el cluster para afrontar las necesidades sobre las que el cluster va a trabajar.
De este forma sólo se agregan los Rolls que se van a utilizar evitando instalar otros paquetes que pueden ser innecesarios en el uso del sistema.

El sistema Rocks utilizado en este proyecto es la versión Chimichanga por lo que tiene la cualidad de hacer actualizaciones al sistema operativo evitando conflictos con el middleware Rocks.

Rocks proporciona sus propios comandos los cuales son básicos para el manejo y administración del cluster siendo de fácil comprensión para el usuario a la hora de ejecutarlos.

2.1.3 Requisitos y características del Cluster Tinku

Frontend / Servidor

- 3 Gb de RAM.
- 40 Gb de Disco Duro.

- Velocidad de procesador: 2.40 Ghz.
- Se recomiendan dos interfaces de red 10/100/1000.
- Unidad CD/DVD.

Nodos

- 1 Gb de RAM.
- 30 Gb de Disco Duro.
- Velocidad de procesador: 2.40 Ghz.
- Interfaz de red 10/100/1000
- Unidad de CD/DVD

Red

- Switch Ethernet 10/100/1000.
- Cable UTP categoría 6 ó 6a.

Hardware Adicional

- Monitor.
- Teclado.
- Mouse.
- Se recomienda una UPS para el servidor.

2.2 COMPONENTES BÁSICOS DE UN CLUSTER

Los siguientes componentes son parte esencial para lograr un buen desempeño y aprovechamiento de los recursos computacionales que hacen parte del Cluster Tinku y de cualquier cluster en general.

- **Administrador de recursos distribuidos.** Es un programa que se encarga de distribuir procesos a los nodos de manera controlada. Actualmente estos sistemas incluyen una interfaz gráfica para monitorear el estado de los procesos.
  Tinku utiliza Sun Grid Engine como administrador de colas que se lo emplea conjuntamente con el programa para el procesamiento de imágenes, el cual se explicará con más detalle en este mismo capítulo.

- **Monitor de recursos.** Esta aplicación suministrará al administrador del sistema información actualizada acerca del estado de todos los nodos del cluster.
  Además, el usuario podrá adecuar la información que quiera visualizar acerca de todos los parámetros que el monitor tenga a cargo.
  En este caso se utilizó el monitor de recursos Ganglia que se explicará con más detalle en este mismo capítulo.

- **Web del cluster.** Se puede decir que la web del cluster tiene la misión de escuchar las peticiones de los usuarios, las atiende o satisface. Su funcionamiento se basa en la búsqueda de una página o programa, enviando una respuesta sobre la búsqueda solicitada.
  Tinku trabaja con Apache como servidor web, descrito en el capítulo 4 MANEJO BÁSICO DE ROCKS.

- **Entornos de ejecución.** Comprende las bibliotecas de paso de mensajes, compiladores y variables de entorno sin estos elementos no es posible desarrollar programas paralelos tampoco es posible el correcto funcionamiento de los programas. En el proyecto se utilizaron los siguientes entornos de ejecución: GCC, ImageMagick, MPJExpress, entre otros.

2.3 SERVICIOS

- **HTTP.** Servicio vital tanto para obtener la información de la monitorización de los recursos y el estado de Tinku así como también para la transferencia de los paquetes en el servidor web en el frontend hacia los nodos.

- **DHCP.** Servicio esencial con el que cuenta el frontend para asignar IP fija a los nodos que se conectan al cluster.

- **SSH.** Es un protocolo necesario para hacer conexión desde el frontend hacia los nodos y de esta forma tener el control para administrar y ejecutar acciones en los equipos.

- **411 Secure Information System.** Siendo una implementación nativa de Rocks, este servicio permite que archivos de configuración y contraseñas utilizados para los servicios del cluster sean distribuidos por el frontend hacia sus respectivos nodos, garantizando que se mantengan sincronizados a lo largo de la operación del cluster.
  Además este servicio informa a los nodos, desde el frontend de la existencia de cambios en los archivos de configuración, haciendo que los

nodos hagan una petición al frontend para su envío y conservar la sincronización con este.

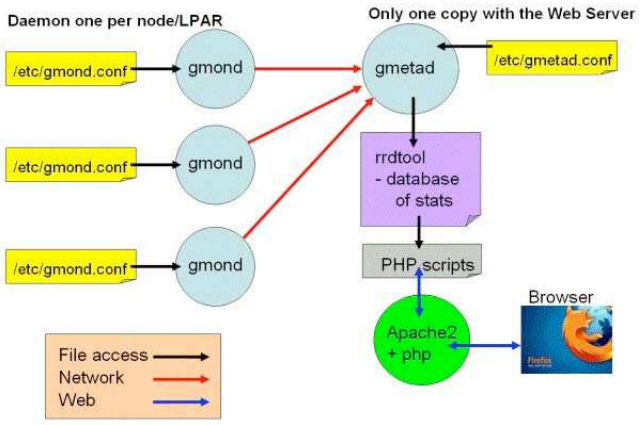
- **DNS.** Este servicio almacena el nombre asignado al frontend con su correspondiente dirección IP, y el nombre de los nodos con su respectiva IP.

- **Autofs.** Asegura que el usuario tenga acceso directo a sus archivos aún si se encuentra conectado a uno de los nodos de computo además realiza el montaje de los sistemas de archivos de forma automática y ocurra al momento de acceso del usuario, este servicio también desmonta el sistema de archivos cuando el usuario deja de utilizarlo.

2.4 HERRAMIENTAS UTILIZADAS EN EL CLUSTER TINKU

2.4.1 Ganglia. Ganglia fue desarrollada en la Universidad de Berkeley por la división de ciencias. Ganglia utiliza un protocolo multicast y un árbol de conexiones punto a punto entre los nodos del cluster. Utiliza XML para la representación de los datos, XDR para transmitir datos y RRDTool para almacenar en una base de datos circular y generar los gráficos. En Figura 16, se muestra el funcionamiento de Ganglia.

Figura 16. Demonios de Ganglia



Griffiths, Nigel. Ganglia How To. [en línea]. Disponible en: (http://www.ibm.com/developerworks/wikis/display/WikiPtype/ganglia).

Demonios:

- gmond. Es el demonio encargado de la monitorización de todo el cluster. Es un servicio que se instala en todas las máquinas que se quiere monitorizar, gmond utiliza el XDR para recopilar información del estado y enviarla en XML vía TCP.

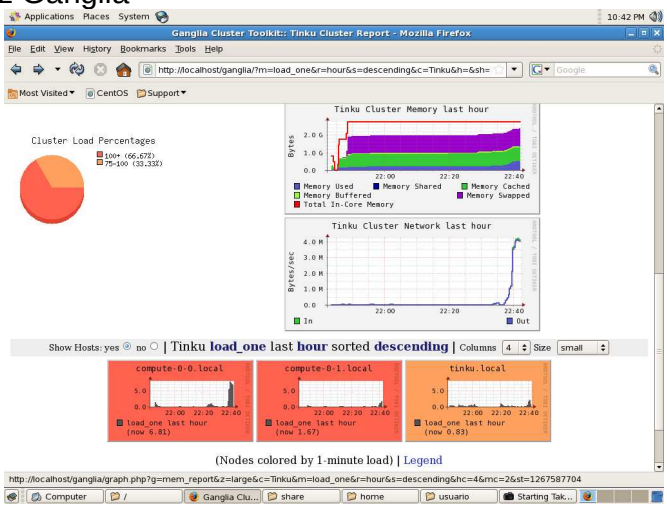
- gmetad. Su función es recoger la información servida por otros gmetad y gmond y almacenar su estado en bases de datos de tipo RRD. gmetad incluye una función para obtener información histórica de un conjunto de máquinas.

- gmetric. Es una aplicación tipo línea de comandos en la que se puede ingresar métricas a gusto del administrador sobre nodos que son monitorizados por Ganglia.

- gstat. Es una aplicación que interactúa con Ganglia directamente y obtiene información sobre los estados guardados.

- web. Presenta al usuario una interfaz gráfica con la información guardada por gmetad utilizando PHP (ver Figura 17).

Figura 17. Interfaz Ganglia



El funcionamiento de esta herramienta, se explica en el capítulo 4 MANEJO BÁSICO DE ROCKS.

2.4.2 Sun Grid Engine (SGE). "En el año 2000 Sun adquirió Gridware, empresa especializada en sistemas de gestión de recursos de computación para ofrecer una versión free de Gridware para Solaris y Linux a la que llamó Sun Grid Engine.

Conocido anteriormente como CODINE (COmputing in DIstributed Networked Environments) o GRD (Global Resource Director), es un sistema de colas de código abierto"[89].

SGE se encarga de aceptar, poner en lista de espera y enviar las tareas a los nodos que hacen parte del cluster. También se hace cargo de otorgar los recursos distribuidos, como procesadores, memoria, espacio en disco y licencias de software.
Controla los usuarios, limitando el número máximo de trabajos por usuario, grupo y proyecto.
SGE actúa como la parte fundamental de un cluster de alto rendimiento. El demonio Sun Grid Master vigila los recursos de la red para permitir control completo y la utilización óptima los mismos. El funcionamiento de esta herramienta, se explicará en el capítulo 4 MANEJO BÁSICO DE ROCKS.

Figura 18. Interfaz de SGE

2.4.3 Java. Java es un lenguaje de programación de alto nivel y orientado a objetos cuya sintaxis es muy similar a la de los lenguajes C o C++. Creado por la empresa Sun Microsystems, de la mano de James Gosling. El objetivo original era construir una plataforma de desarrollo de aplicaciones para controlar electrodomésticos y pequeños equipos electrónicos, como televisores, vídeos, teléfonos celulares, etc. Posteriormente, se desechó el propósito inicial y se utilizó como lenguaje para Internet, debido a que facilita la introducción de nuevas funcionalidades en las páginas Web. Una característica importante es que permite crear las páginas mucho más dinámicas de lo que era posible hasta ese momento, sin la necesidad de establecer una conexión con el servidor[90].

En este proyecto se escogió a Java como lenguaje de programación en paralelo porque existe una herramienta especialmente diseñada para este tipo de entornos, aparte que se tiene un buen conocimiento del lenguaje.

[89] MARTÍNEZ, Op. Cit.
[90] ANGELFIRE.COM. Generalidades de Java. [en línea]. Disponible en web: (http://www.angelfire.com/electronic/econetcol/generalidadesjava.htm).

- **Apache Tomcat.** Es el servidor web que tiene soporte para servlets y Java Server Pages. En Tinku, principalmente se encarga visualizar el estado del cluster con Ganglia, envía los archivos de instalación a los nodos, entre otras funciones.

2.5 HERRAMIENTAS PARA DETERMINAR EL DESEMPEÑO DEL CLUSTER TINKU

2.5.1 MPJ Express. Es una biblioteca de paso de mensajes para crear y ejecutar aplicaciones paralelas en lenguaje Java en clusters de cómputo y redes de computadores.
MPJ Express está diseñado para equipos con memoria distribuida como los clusters, también ejecuta aplicaciones paralelas en equipos de escritorio o portátiles que contienen memoria compartida o procesadores multicore.
MPJ Express fue desarrollado por Aamir Shafi, bajo la supervisión de Bryan Carpenter y Mark Baker. Como interfaz de comunicaciones utiliza Fast Ethernet y Myrinet.

MPJ Express puede ser configurado de dos maneras:
Configuración Multicore: Esta configuración es válida para procesadores multicore o máquinas de memoria compartida.
Configuración Cluster: Esta configuración es usada para ejecutar aplicaciones paralelas Java en máquinas de memoria distribuida como cluster o NOW (Network Of Workstations).

Como MPJ Express es una implementación del estándar MPI, el cual tiene 129 funciones, aquellas que se utilizaron en el desarrollo del algoritmo en paralelo son:

- **MPI_Init().** Esta función permite inicializar en el sistema la biblioteca MPI para que pueda ser empleada.

- **MPI_Finalize().** Termina la ejecución del entorno MPI y libera los recursos utilizados.

- **MPI.COMM_WORLD.Size().** Retorna el número de procesos de un comunicador.

- **MPI.COMM_WORLD.Rank().** Devuelve el identificador de proceso dentro del comunicador especificado.

- **MPI.COMM_WORLD.Send(void *mensaje , int contador, MPI_Datatype tipodedato, int destino, int etiqueta).** Envía un mensaje a un proceso determinado.

Los parámetros de la función utilizada en el algoritmo:

mensaje: Es la dirección del arreglo o buffer que se va a enviar.
contador: es el número de elementos que se va a enviar, debe ser igual o mayor a cero.
tipodedato: tipo de dato de los elementos a enviar.
destino: Identificador del proceso que envía.
etiqueta: Particiona el espacio de los mensajes.

- **MPI.COMM_WORLD.MPI_Recv(void *mensaje , int contador, MPI_Datatype tipodedato, int origen):** Recibe un mensaje de un proceso.
  mensaje: Es la dirección del arreglo o buffer que se va a recibir.
  contador: es el número a elementos que se va a recibir, debe ser igual o mayor a cero.
  tipodedato: tipo de dato de los elementos a recibir.
  origen: Identificador del proceso que está recibiendo.

- **MPI.COMM_WORLD.Barrier():** Sirve para sincronización global entre los procesadores del comunicador. Esta función se bloquea hasta que todos los procesos del comunicador la ejecutan.

2.5.2 POV-Ray. Crea imágenes tridimensionales utilizando la técnica de trazado de rayos. Para crear una imagen de este tipo, POV-Ray lee un archivo de texto plano que contiene la información que describe los objetos, la iluminación y las vistas en una escena.
El trazado de rayos es un proceso lento que genera imágenes de muy alta calidad y resolución con efectos realistas, sombreados, perspectivas y texturas que se utilizan en los medios audiovisuales.
POV-Ray incluye ejemplos de escenas que se pueden modificar y empezar a crear las propias imágenes. También contiene una gran biblioteca de formas predefinidas y materiales que se pueden importar en la escena que se está trabajando.

Figura 19. Escena generada con POV-Ray

TRAN, Gilles. Sci-fi cars and buildings (POV-Ray). [en línea]. Disponible en web: (http://www.oyonale.com/modeles.php?lang=en&page=37).

Características

- Incluye bibliotecas de archivos de escenas.
- Lenguaje de descripción de escenas en archivos de texto plano.
- Variedad de fuentes luminosas dirigidas, cilíndricas y extendidas.
- Utiliza radiosidad para conseguir iluminación realista.
- Permite crear efectos como nubes, polvo, fuego, vapor, lluvia, niebla y arcoíris.
- Primitivas para figuras básicas, como esferas, cajas, cilindros, conos, triángulos y planos.
- Primitivas para figuras avanzadas.
- Las primitivas se pueden combinar para crear nuevas figuras complejas usando Geometría Constructiva Sólida (CSG).
- Permite crear nuevas texturas o usar las predefinidas.
- Combina texturas usando capas de texturas semitransparentes.

2.5.3 Zebra-Core. Es una herramienta creada en la Universidad de Antioquia por Carolina Escobar, Diego Bedoya y Jorge Zuluaga, se encarga de la división y distribución de imágenes tridimensionales generadas mediante técnicas de trazado de rayos con POV-Ray sobre plataformas tipo cluster. Está desarrollada en Perl y utiliza línea de comandos. El proceso inicia con la división de la imagen en partes iguales, por cada fracción de la imagen se crean instancias de POV-Ray que se ejecutan en paralelo. Las imágenes resultantes se ubican en el frontend y se unen utilizando un módulo de ImageMagick que muestra al final la imagen completa. Zebra–Core se encarga de los procesos de división, aglomeración, asignación, generación y fusión final de la imagen. Mediante la utilización de esta herramienta en Tinku se pudo demostrar la capacidad para reducir el tiempo de generación de imágenes tridimensionales de alta resolución, con alta eficiencia y un costo bajo de uso de red.
En este proyecto se trabajó con escenas de ejemplo que vienen en POV-Ray y algunas descargadas de la web. Las pruebas que se realizaron en Tinku en comparación con un solo computador evidenciaron resultados favorables a la computación de alto rendimiento y están descritos en el capítulo 5 PRUEBAS Y RESULTADOS.

Métodos:

Strips. Divide la imagen en un número de franjas especificado, pueden ir en sentido vertical u horizontal. En este método se asigna un nombre al trabajo a realizar.

Procs. En este proceso se garantiza que la renderización sea válida según la disponibilidad de recursos. También se generan los scripts de las franjas.

Launch. Durante este proceso se asignan los scripts generados en la fase anterior a cada uno de los procesadores disponibles, haciendo uso de SGE.

Build. Su trabajo es fusionar en una sola imagen las franjas que se generan en la fase anterior. Es importante que los archivos se encuentren en un directorio que tenga permisos de lectura y escritura.

2.5.4 WEKA-Parallel. WEKA es una herramienta de aprendizaje automático y minería de datos, desarrollada en Java por la universidad de Waikato (Nueva Zelanda), distribuido bajo licencia GPL.
Una versión de WEKA especialmente diseñada para ambientes paralelos, llamada WEKA-Parallel fue utilizada en el proyecto, esta herramienta contiene un componente especial para fragmentar un conjunto de datos en subconjuntos para que de este modo se pueda trabajar en estos de forma independiente y así lograr trabajar en forma paralela, a esto se le conoce con el nombre de validación cruzada.

WEKA-Parallel maneja el mismo entorno gráfico que su antecesor WEKA, provee algunas modificaciones de configuración para trabajar de forma paralela, de tal modo que su manejo no tiene mayor cambio.
WEKA-Parallel forma el complemento perfecto para comprobar el comportamiento de la computación de alto rendimiento en técnicas de minería de datos.
En este proyecto se trabajó tanto para determinar la computación de alto rendimiento y el funcionamiento de WEKA-Parallel con repositorios de datos en formato ARFF en los que se midió la rapidez con la cual se entregaron los resultados por parte del cluster comparados con la misma herramienta WEKA pero en forma secuencial en un solo equipo. Se muestran los resultados de manera detallada en el capítulo 5 PRUEBAS Y RESULTADOS.

2.6 DIAGRAMAS GENERALES UML DEL SISTEMA CLUSTER TINKU

Figura 20. Diagrama de casos de uso Cluster Tinku

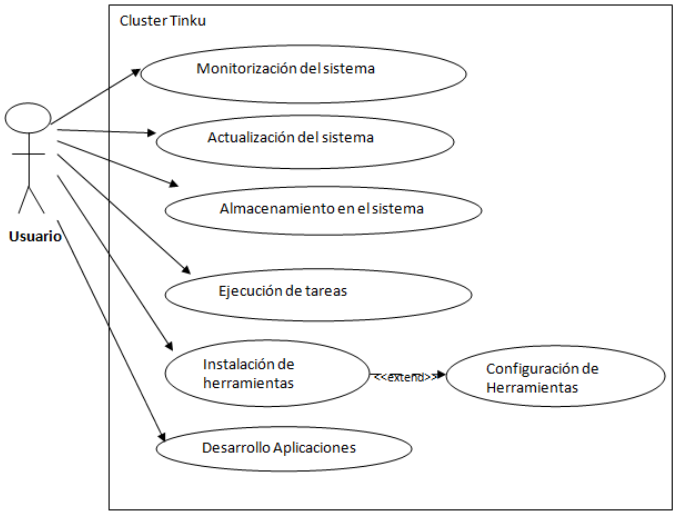


Figura 21. Diagrama de clases Cluster Tinku

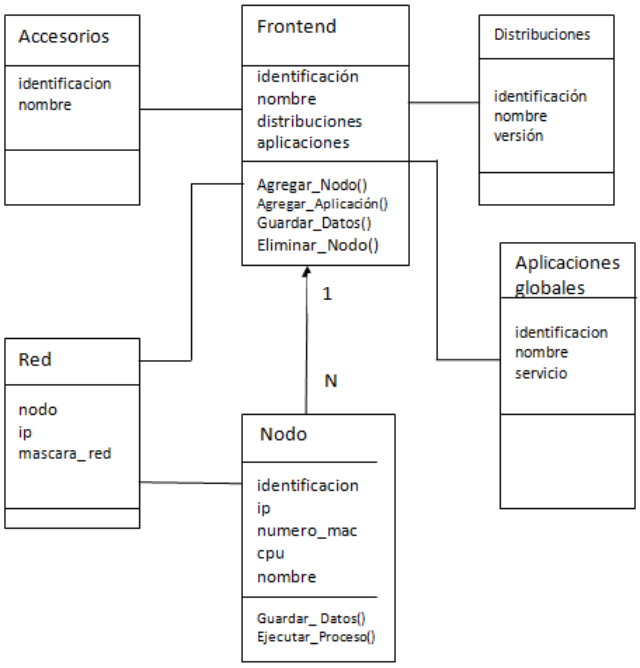
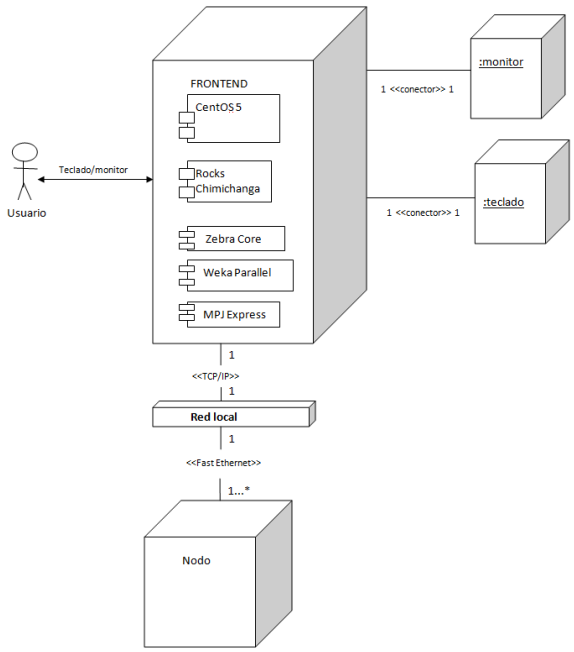


Figura 22. Diagrama de despliegue Cluster Tinku



3. DESARROLLO DEL CLUSTER TINKU

3.1 INSTALACIÓN Y CONFIGURACIÓN DE TINKU CON ROCKS

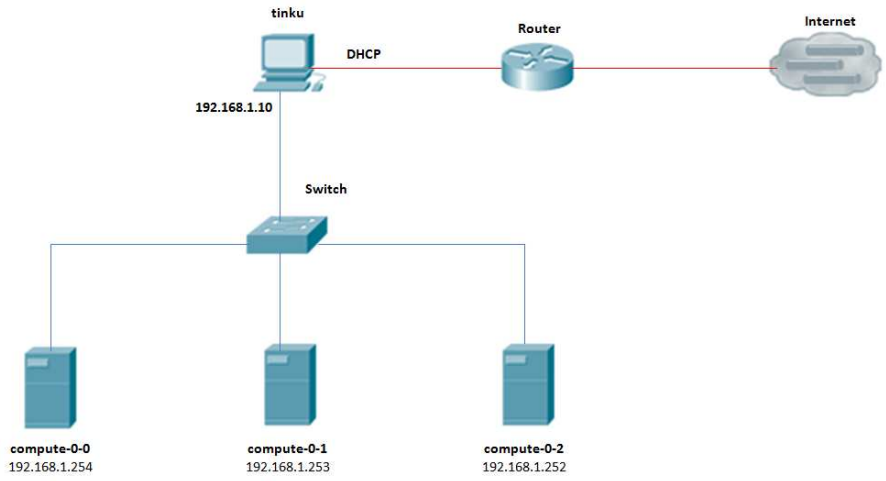
3.1.1 Instalación y configuración del frontend. En el presente capítulo se exponen los pasos necesarios para instalar Rocks Chimichanga en el frontend y en los nodos. Se tiene que disponer de los siguientes paquetes de software:

- Kernel Roll.
- Core Roll.
- OS Roll Disk 1.
- OS Roll Disk 2.

Cualquier otro roll que considere necesario.

El Kernel Roll es el encargado del arranque del sistema, el Core Roll contiene las herramientas para computación de alto rendimiento y los OS Roll Disk contienen el sistema operativo CentOS 5, si se desea instalar otro sistema operativo diferente a CentOS tiene que ser una distribución basada en RHEL.
Todos los componentes software vienen en formato .iso, se pueden descargar gratuitamente de www.rocksclusters.org.

Figura 23. Topología Cluster Tinku



Es recomendable que el equipo que vaya a ser servidor tenga dos interfaces de red, una para la conexión con el cluster y la otra para la red externa, esto se hace para tener una mejor organización de la estructura del cluster. Además, el cluster debe estar en un cuarto con buena ventilación para disipar el calor que genera el cluster que en algunos casos puede llegar a dañar los equipos. La estructura del cluster se observa en Figura 23.

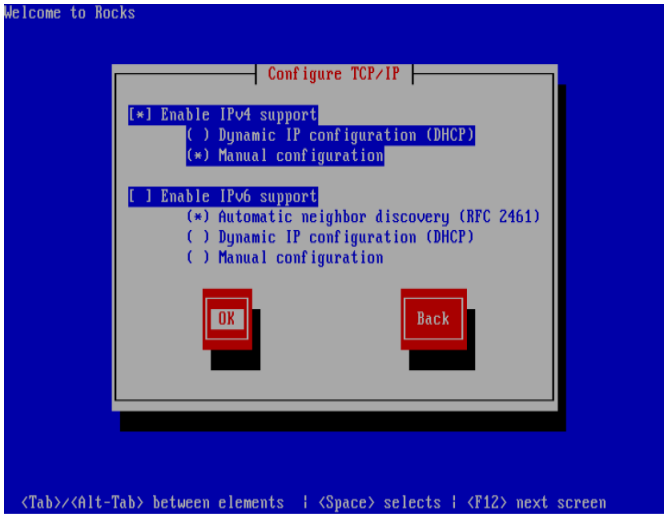
En el caso que se cuente con dos interfaces de red la eth0 será la que se comunique con los nodos y la eth1 se comunicará con el mundo exterior.
El arranque del frontend se debe configurar de la siguiente manera: primero el arranque por unidad CD/DVD, y segundo el Disco Duro. Una vez hecho esto se procede a insertar el disco Kernel Roll. Al realizar lo anteriormente descrito aparece una pantalla como muestra Figura 24:

Figura 24. Pantalla inicial instalación Rocks 5.2

Para determinar que este equipo va a funcionar como frontend, se debe escribir la palabra build y presionar Enter para continuar con la instalación.
En caso de omitir el paso anterior el programa de instalación entenderá que se va a instalar un nodo, si esto llega a suceder reinicie el equipo.

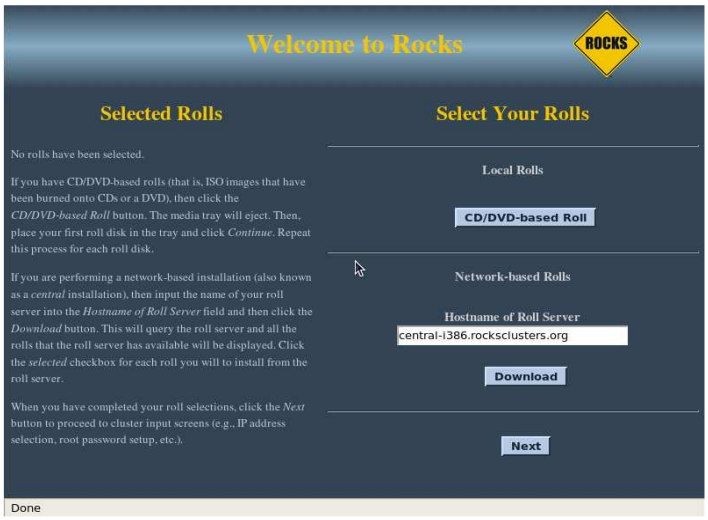
A continuación aparece una pantalla igual a Figura 25.

Figura 25. Configuración TCP/IP



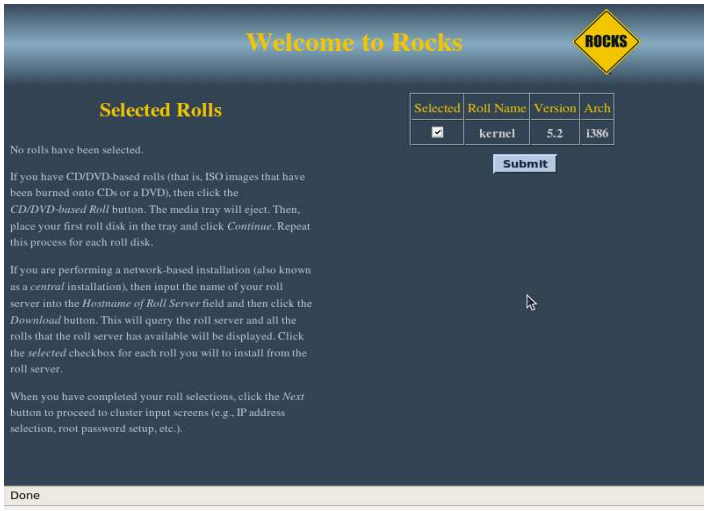
Donde se habilita la dirección IP versión 4 y se deshabilita la dirección IP versión 6, y se presiona OK.
Posteriormente muestra una pantalla como se muestra en Figura 26, que indica si los Rolls se van a instalar desde CD/DVD o directamente desde Internet, si selecciona la instalación desde la red es necesario contar con un ancho de banda lo suficientemente grande para evitar posibles complicaciones.

Figura 26. Selección inicial de Roll



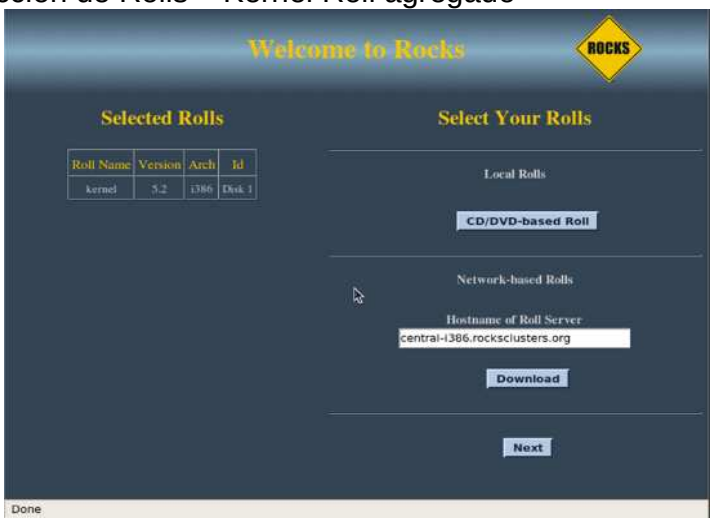
En el proyecto los Rolls fueron descargados en CD, se da clic en el botón CD/DVD – based Roll y aparece el primer Roll a instalar como se muestra en Figura 27.

Figura 27. Selección Kernel Roll



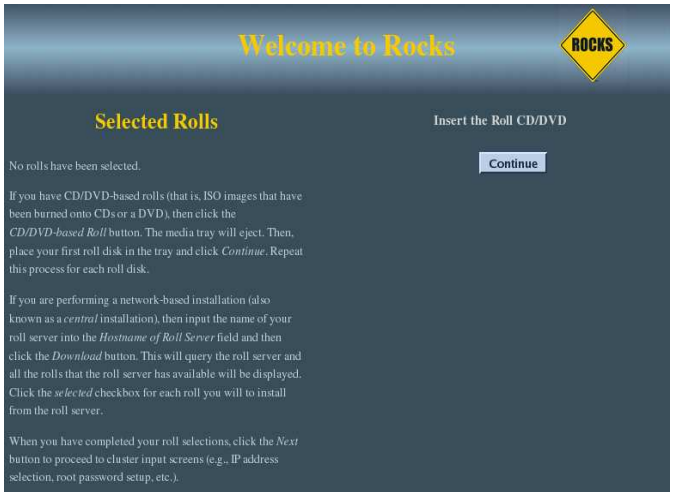
Como en la bandeja de CD se encuentra el Kernel Roll, se activa esa casilla de verificación y se da clic en el botón Submit, para continuar con la instalación de los demás Rolls.
Una vez hecho esto aparece la pantalla que se muestra en Figura 28.

Figura 28. Selección de Rolls – Kernel Roll agregado



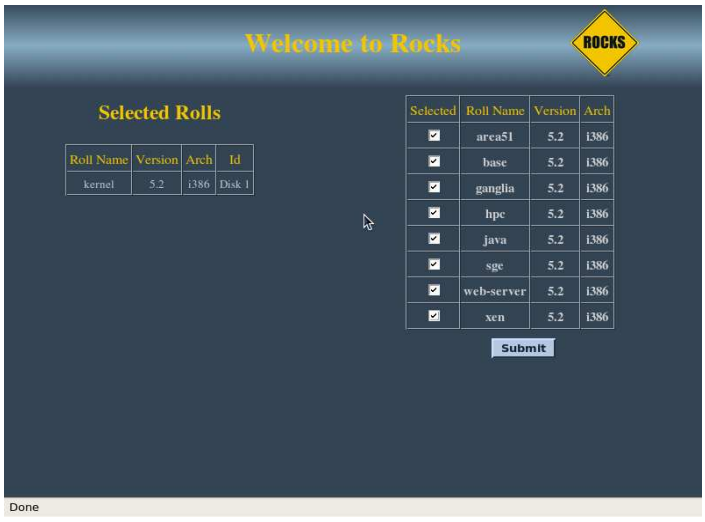
Aquí se observa en la parte izquierda que el roll kernel se ha agregado satisfactoriamente. Ahora se presiona el botón CD/DVD – based Roll.

Figura 29. Insertar Roll



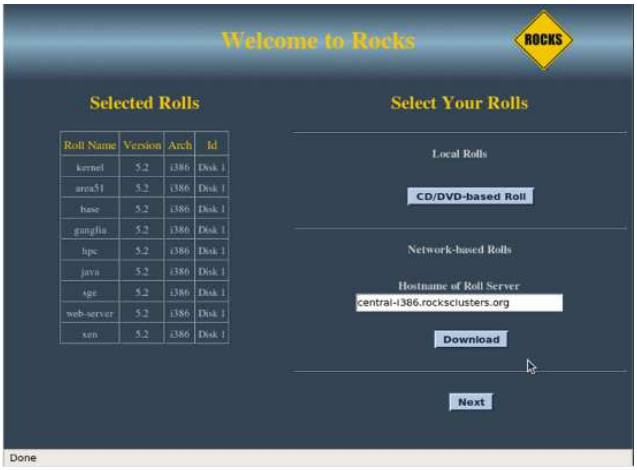
En Figura 29, se realiza el cambio de CD, en este caso se introduce el Core Roll y se presiona el botón Continue.

Figura 30. Selección Core Roll



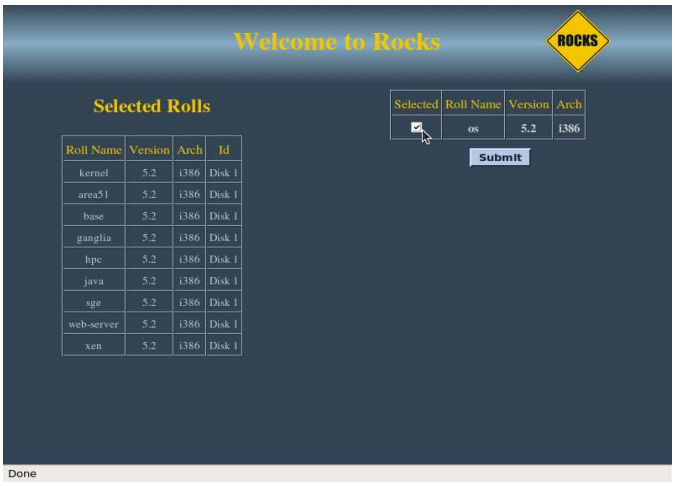
El Core Roll es especial, en el se encuentran varios Rolls debido a esto es necesario seleccionar los Rolls para determinar el tipo de configuración que va a tener el cluster. Una vez hecho esto se da clic en el botón Submit.

Figura 31. Selección de Rolls - Core Roll agregado



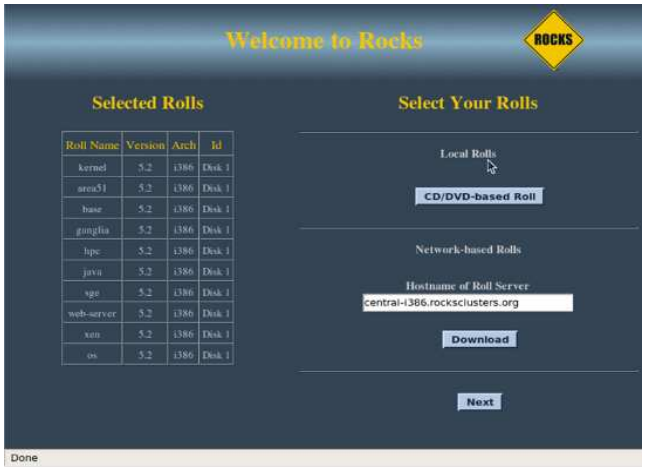
En Figura 31, se puede observar que la lista de la izquierda muestra los Rolls que se han agregado. Se oprime el botón CD/DVD based Roll, y aparece Figura 26, se cambia el CD por el OS Roll Disk 1 o la distribución RHEL 5 de su preferencia y se presiona Continue.
Una vez realizado lo anterior aparece lo siguiente:

Figura 32. Selección de OS Roll Disk 1



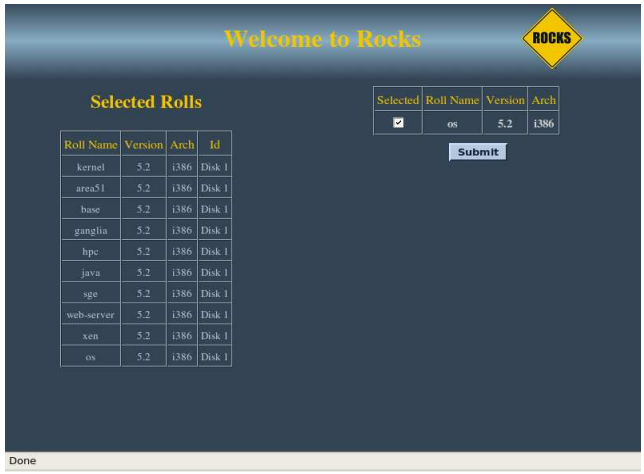
Se selecciona el Roll disponible y se oprime el botón Submit. Una vez realizado lo anterior aparece una pantalla como muestra Figura 33.

Figura 33. Selección de Rolls – OS Roll Disk 1 agregado



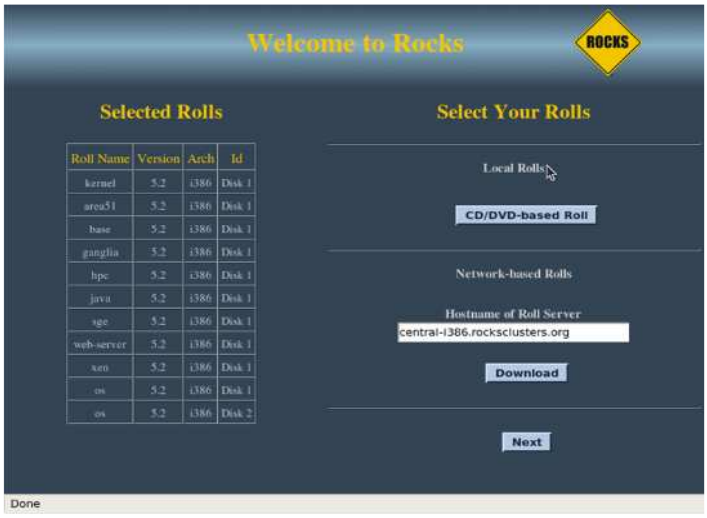
Se oprime el botón CD/DVD based Roll, y aparece la pantalla (ver Figura 29), se cambia el CD por el OS Roll Disk 2 o la distribución RHEL 5 de su preferencia y se presiona Continue.
Aparece una pantalla idéntica a la que se muestra en Figura 34.

Figura 34. Selección de OS Roll Disk 2



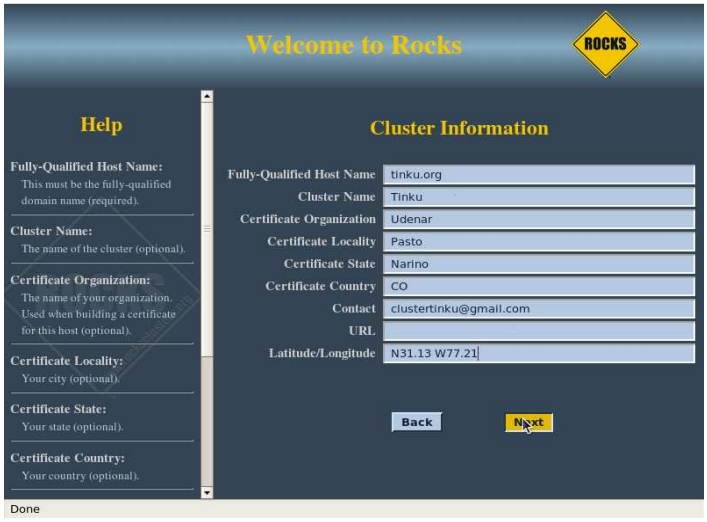
Se selecciona el Roll disponible y se oprime el botón Submit.
Una vez realizado lo anterior aparece una pantalla (ver Figura 35), que muestra todos los Rolls que se han agregado para el cluster de alto rendimiento.

Figura 35. Selección de Rolls – Todos los Rolls agregados



A continuación se oprime el botón Next.

Figura 36. Información del cluster



En Figura 36, se muestra un formulario, en el cual se ingresa información concerniente al cluster.

El campo Fully-Qualified Host Name es obligatorio, es el nombre con el cual se conocerá en la red externa.

Cluster Name: Es el nombre que manejará internamente el cluster, de este modo sirve como identificador para las herramientas que maneja el cluster.

Certificate Organization: Es la entidad propietaria a la que corresponde el cluster. En los campos siguientes se escribe la ciudad, el estado y el país en abreviatura donde se encuentra el cluster.

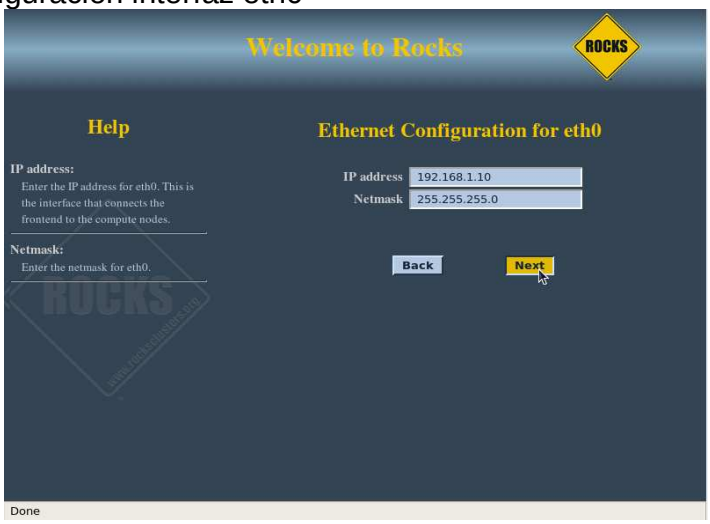
Contact: Un correo electrónico válido.

URL: Página web del cluster o de la organización propietaria.

Latitude/Longitude: La ubicación geográfica donde está el cluster.

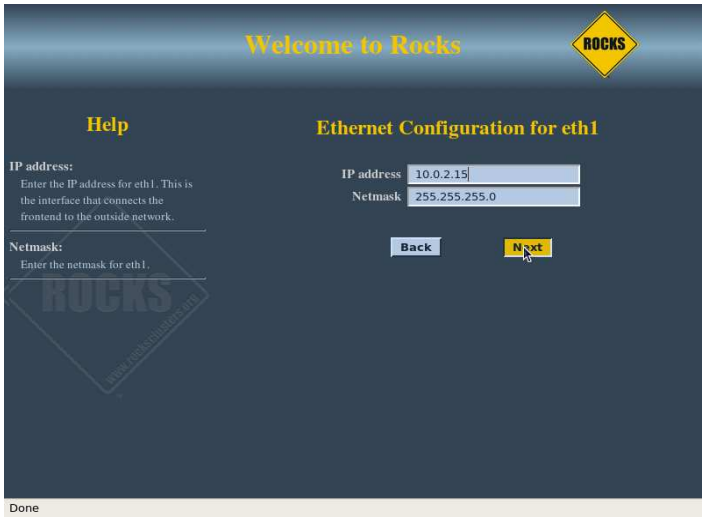
Una vez realizado esto se oprime el botón Next.
Aparece una pantalla como se muestra en Figura 37.

Figura 37. Configuración interfaz eth0



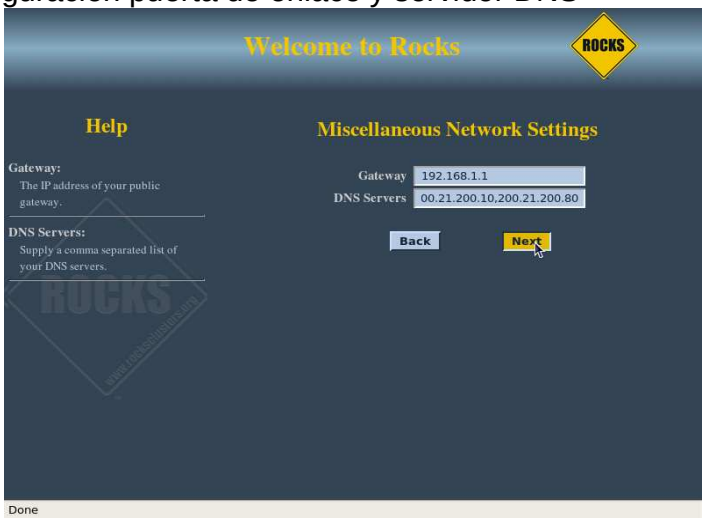
En IP address se escribe la dirección IP del frontend, en este caso se utilizó la IP privada con 192.168.*.*
En Netmask se escribe la máscara de red, en este caso fue 255.255.255.0
A continuación se oprime Next.

Figura 38. Configuración Interfaz eth1



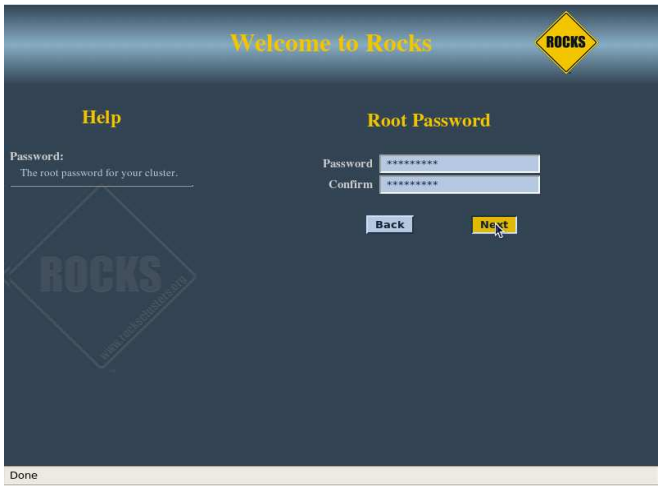
En Figura 38, se configura la interfaz eth1. Si se dispone de un servidor DHCP, estos campos no se modifican. Se oprime el botón Next.

Figura 39. Configuración puerta de enlace y servidor DNS



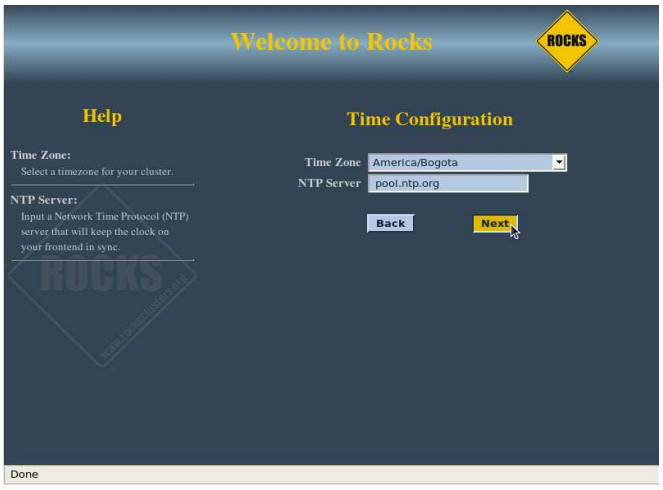
En Figura 39, se configura la puerta de enlace y los servidores DNS, en caso de que tenga, en este proyecto la puerta de enlace es: 10.1.1.1 y los servidores DNS se dejaron por defecto.
Se da clic en el botón Next.
Una vez realizado esto aparece una pantalla la cual se muestra en la Figura 40.

Figura 40. Asignación contraseña root



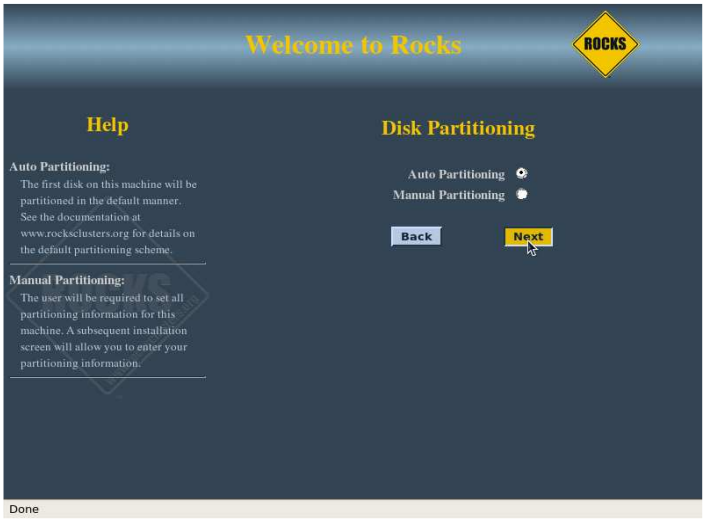
En esta pantalla se establece la contraseña de administrador (root). Se oprime el botón Next.
Aparece la siguiente pantalla la cual se muestra en Figura 41, en la que se va a configurar la zona horaria.

Figura 41. Configuración zona horaria



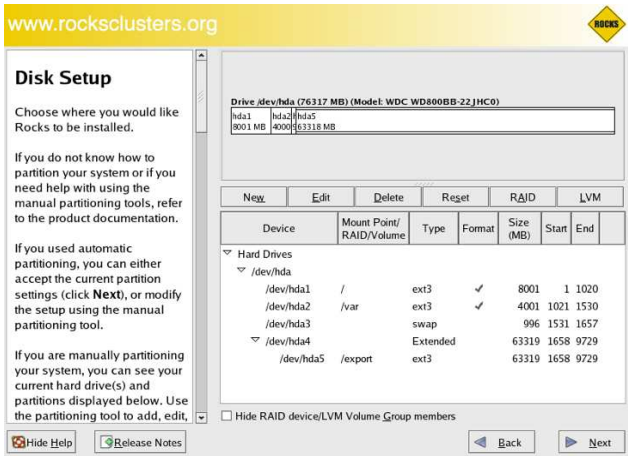
En este caso seleccionamos America/Bogota como zona horaria y el servidor NTP por defecto. Se da clic en Next.

Figura 42. Particionamiento del Disco



En Figura 42, aparece la forma en que se va a particionar el disco, se elige la primera opción que particiona los discos automáticamente, Rocks elimina la información guardada en ese disco, si se selecciona la segunda opción aparece una pantalla como se muestra en Figura 43.

Figura 43. Particionamiento manual



Si escoge de manera manual, se debe reservar un espacio en disco como lo explica la Tabla 1.

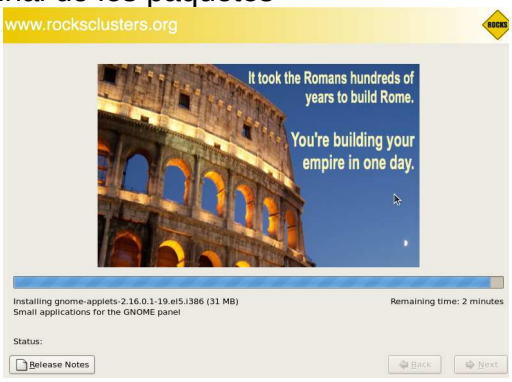
Tabla 1. Partición por defecto

Partición	Tamaño
/	16 Gb

/var	4 Gb
Swap	1 Gb
/export (enlace simbólico a /state/partition1)	resto del disco

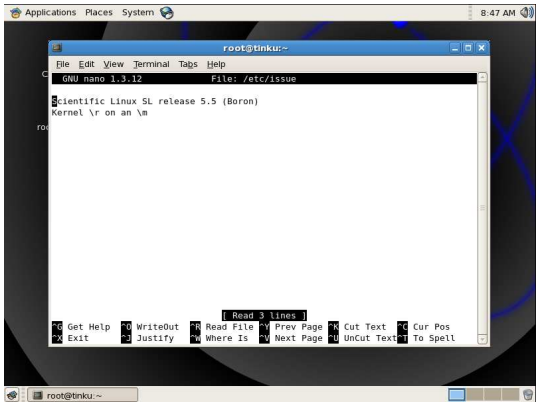
Rocks monta los directorios que van a ser compartidos por red en /export.
Por último se insertan los discos, según el orden que los vaya pidiendo e instalará los paquetes como se muestra en Figura 44.

Figura 44. Instalación final de los paquetes



Como alternativa se realizó la instalación del cluster a manera de prueba con la distribución Scientific Linux 5 y no presentó ningún inconveniente. En Figura 45, se muestra el frontend con Scientific Linux como sistema operativo.

Figura 45. Instalación frontend Scientific Linux



- **Configuración teclado.** Una vez instalado el frontend, es necesario ajustar el teclado para usuarios con teclado latinoamericano o español, se requiere cargar el mapa correspondiente:

  # loadkeys /lib/kbd/keymaps/i386/qwerty/es.map.gz

  Para garantizar que la configuración de teclado siempre se guarde cada vez que se encienda el frontend, se debe agregar esta línea de comando al archivo rc.local.

  # echo loadkeys /lib/kbd/keymaps/i386/qwerty/es.map.gz >> /etc/rc.local

- **Forzar servicio Ganglia.** Al momento de instalar Rocks, el servicio Ganglia muestra que se inicia correctamente pero al momento de monitorear los equipos, indica que los demonios no han sido iniciados, para tener la seguridad que Ganglia se va a ejecutar en el instante de carga del sistema se agregan las siguientes líneas al archivo /etc/rc.local
  …
  service gmond restart
  service gmetad restart

  Una vez realizado esto, se guardan los cambios.

- **Creación de cuentas de usuario.** Antes de instalar los nodos se recomienda crear las cuentas de usuario.

  # useradd usuario

  Una vez creada la cuenta se asigna como directorio home de la nueva cuenta la ruta /export/home/usuario. Para que el usuario encuentre su directorio home en otros nodos, este debe ser cambiado a /home.

  # usermod –d /home/usuario usuario

  Se asigna una contraseña a usuario.

  # passwd usuario

  Se procede a configurar la cuenta para que todos los nodos se conecten de manera transparente en el cluster. Para esto se configura el archivo autofs para que el directorio home se monte automáticamente en los nodos.

- **Configuración de autofs.** Se agrega la siguiente línea al archivo /etc/auto.home

usuario tinku.local:/export/home/usuario

- **Sincronización con 411.** Los cambios anteriormente hechos deben sincronizarse en todo el cluster.

  # make –C /var/411 force
  # service autofs reload
  # rocks run host service autofs reload
  # rocks sync users

  Una vez realizado esto es muy importante revisar si el usuario puede encontrar su directorio home.

- **Modificación de cuentas de usuario.** Es posible cambiar el grupo al cual pertenece el usuario o la contraseña.

  # usermod –g <gid> usuario

  Donde <gid> es el nuevo identificador de grupo al que se desea que pertenezca el usuario.Se requiere hacer una sincronización, para ello se utiliza:

  # service 411 commit

  El sistema 411 realiza una sincronización automática cada hora.

- **Eliminación de cuentas de usuario.**

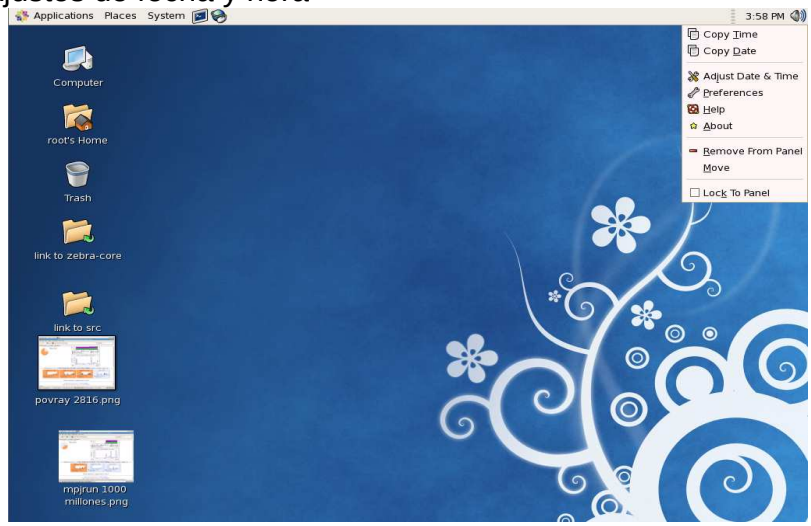
  # userdel usuario
  # umount /home/usuario
  # rocks run host “umount /home/usuario”
  # rm –Rf /export/home/usuario
  # make –C /var/411 force
  # service autofs reload
  # rocks run host “service autofs reload”
  # rocks sync users

- **Estructura directorio /export/apps.** Para instalar paquetes en el cluster es necesario crear una estructura de archivos.
  # mkdir –p etc man bin sbin src share

- **Ajustar la fecha y hora mediante NTP.** Antes de adicionar los nodos es importante sincronizar el frontend con un servidor NTP, que se encarga que los nodos del cluster tengan la misma hora.
  Debe estar conectado a Internet para realizar este procedimiento.

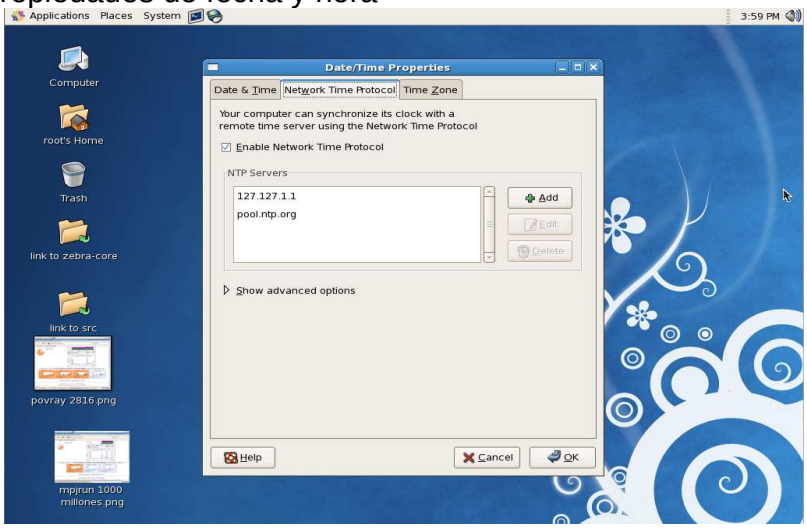
Como root, en modo gráfico debe dar clic derecho en el reloj que se encuentra en la parte superior derecha de la pantalla, se despliega un submenú y se da clic en Adjust Date & Time, como se muestra en Figura 46.

Figura 46. Ajustes de fecha y hora



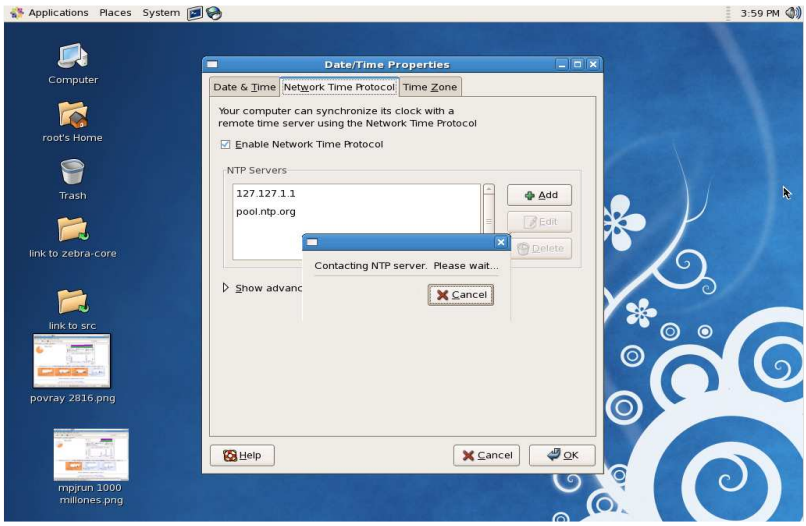
Se despliega una ventana como muestra Figura 47.

Figura 47. Propiedades de fecha y hora



Se puede observar que la ventana tiene tres pestañas, se da clic en la pestaña marcada con Network Time Protocol, a continuación se selecciona el segundo servidor NTP, y se da clic en el botón OK. Aparece una ventana que indica la conexión con el servidor NTP, como se muestra en Figura 48.

Figura 48. Conexión servidor NTP



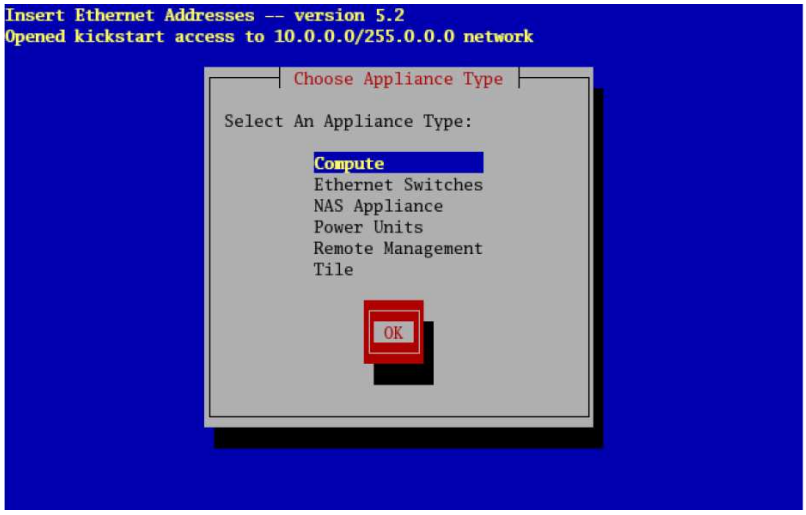
Luego de la instalación del frontend, se procedió a instalar los nodos para el procesamiento paralelo, el frontend manda los archivos de instalación por medio de la red a cada uno de los nodos.

3.1.2 Instalación nodos. Se procede con la ejecución de un programa que configura automáticamente los nodos.
Es importante mencionar los ajustes que se hicieron en la BIOS de cada nodo se configuró en el siguiente orden de arranque: CD, Arranque en red y Disco duro.
Una vez los nodos estén conectados a la red interna se ejecuta la siguiente línea desde el frontend:

insert-ethers

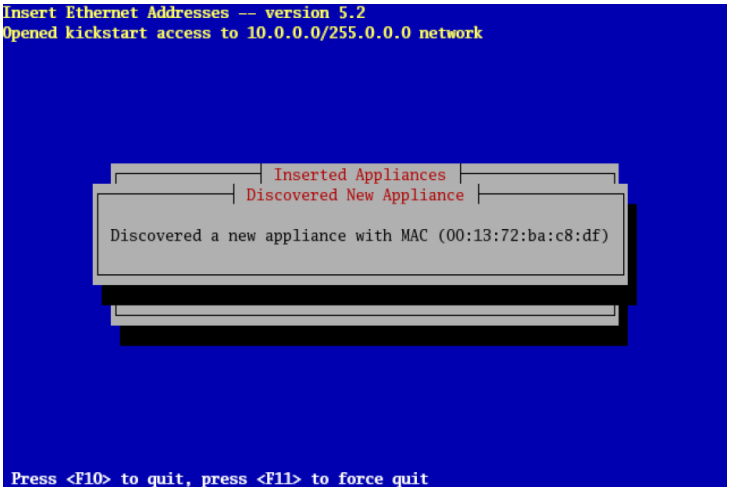
Este comando muestra una pantalla como se muestra en Figura 49:

Figura 49. Adición de un Nodo



Si es un switch manejado, preferiblemente deshablitar el servicio DHCP para que el frontend sea quien asigne las direcciones IP a los nodos de una manera ordenada.
Una vez determinado el nodo a instalar, se enciende el equipo con el CD Kernel Roll. Mientras en el frontend se selecciona la opción Compute, se espera hasta que el frontend reciba la MAC del nodo que se conectó como se muestra en Figura 50.

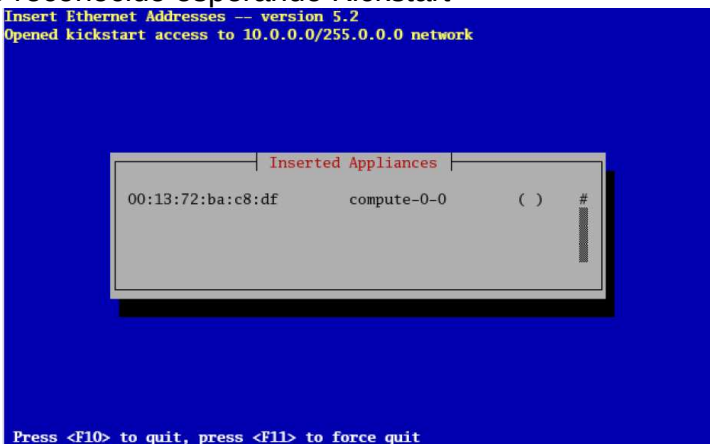
Figura 50. Nuevo nodo descubierto



Una vez detecte un nuevo equipo (ver Figura 50), el nodo recibe un nombre y la IP que manejará el frontend, este asignará secuencialmente los nombres de la siguiente manera, dado que es el primer nodo que se agrega al cluster este recibe el nombre de compute-0-0 a medida que se vayan ingresando los nodos el

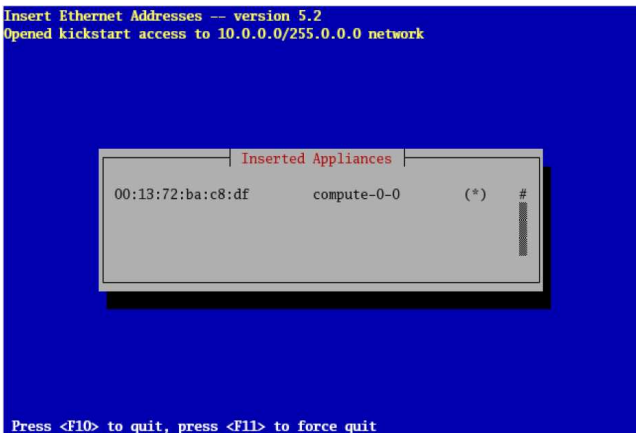
segundo digito del nombre se incrementará. De acuerdo como se haya configurado la red, el último digito de la IP irá decreciendo.

Figura 51. Nodo reconocido esperando Kickstart



Además, el nodo quedará registrado en la base de datos MySQL que posee el frontend.
A continuación el nodo solicitará un archivo Kickstart al frontend (ver Figura 51), si la respuesta es satisfactoria aparece un asterisco al lado del nombre, como muestra Figura 52.

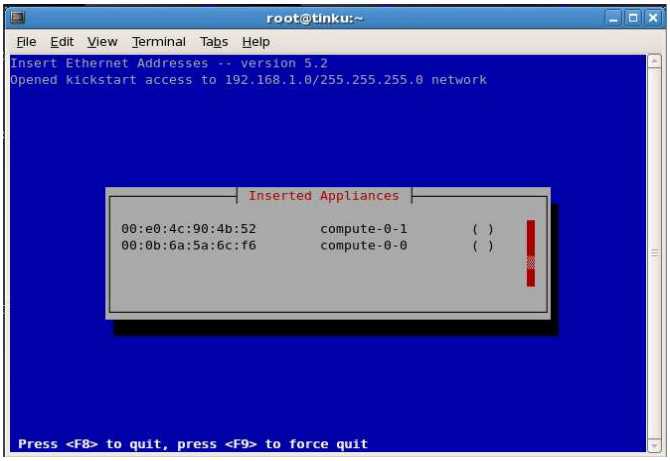
Figura 52. Nodo reconocido Kickstart enviado



Si no es posible agregar el nodo, en lugar de (*) aparece un mensaje de error.
Para simplificar la instalación en varios nodos al mismo tiempo se deben hacer varias copias del Kernel Roll.
Rocks cuenta con Avalanche Installer, el cual permitirá una instalación en masa.

Una vez realizado esto se agregan los nodos con el mismo procedimiento que se mencionó anteriormente, cuando se agregan los nodos aparece un mensaje en pantalla como se muestra en Figura 53.

Figura 53. Adición de nodos utilizando Avalanche Installer



Si se desea monitorear la instalación del nodo, en modo gráfico a través de línea de comandos se escribe la siguiente línea:

rocks-console compute-0-0

Una vez instalado el nodo, dependiendo de la versión, se presiona F8 para quitar la pantalla de insert-ethers.

- **Recomendaciones en caso de fallos.** Si el nodo no fue instalado correctamente se procede a reiniciar los siguientes servicios:

  # service httpd restart
  # /etc/rc.d/init.d/mysqld restart
  # service autofs restart

Para comprobar el correcto funcionamiento del cluster se procede a realizar los siguientes pasos desde el frontend:

Comprobar el funcionamiento del comando rocks run host, este comando permite enviar una orden a todo el cluster a o un sub conjunto.
Verificar la conexión SSH de un usuario sea transparente

ssh usuario@compute-0-0

Si no es posible conectarse realice lo siguiente:

- Compruebe que el usuario existe.
- Verifique que la contraseña sea correcta.
- Si no encuentra el directorio home, revise el archivo /etc/auto.home y realice lo siguiente:

  # make –C /var/411 force
  # service autofs reload
  # rocks sync users

La instalación debe ser cuidadosamente vigilada, en caso de dejar un CD en la unidad, este entrará en modo cíclico y volvería a instalar el Roll ocasionando error en el proceso anteriormente descrito.

3.1.3 Instalación de herramientas para renderizar imágenes en paralelo

3.1.3.1 POV-Ray. El paquete de instalación de POV-Ray viene comprimido en un archivo de tipo tar.gz, disponible en http://www.povray.org/download/.
Permite configurar en detalle los archivos binarios y los directorios en los cuales se almacenarán los archivos de instalación.
Primero se copia el paquete archivo .tar.gz en /share/apps/src, mediante el siguiente comando:

#cp povray-3.6.tar.gz /share/apps/src

Se procede a desempaquetar y descomprimir el archivo:

cd /export/apps/src
tar zxvf povray-3.6.tar.gz

Después de haber descomprimido, se dirige al nuevo directorio que se ha creado.

cd povray-3.6.1

Se preparan los archivos para la compilación de POV-Ray mediante el siguiente comando:

./configure COMPILED_BY ”Jorge Mora y Evelyn Vinueza <clustertinku@gmail.com>” --prefix=/share/apps

La opción --prefix=/share/apps, permite el acceso al programa a todos los nodos.

Adicionalmente, si desea saber que otras opciones de instalación tiene POV-Ray ejecute:

./configure –help

Una vez seleccionado el directorio destino, se procede a compilar las fuentes:

make

También es posible utilizar make check, que al final de todo el proceso realiza una prueba de renderización:

make check

Si no aparece ningún mensaje de error en la compilación se procede con la instalación.

make install

Es necesario modificar el archivo /etc/profile, para agregar al PATH el nuevo programa.

…
export PATH=$PATH:/share/apps/bin
…

Ahora se debe modificar el archivo /var/411/Files.mk para que la línea anterior se agregue a todos los nodos del cluster.

...
FILES = $(AUTOMOUNT) \
/etc/passwd \
/etc/shadow \
/etc/group \
/etc/services \
/etc/rpc \
/etc/profile
...

Se guardan los cambios y se ejecuta:

```
# make –C /var/411 clean
# make –C /var/411
```

Ahora con la ejecución del siguiente comando se mandan los nuevos archivos a los nodos.

```
#rocks run host “411get”
```

Es necesario reiniciar el cluster para que los cambios hagan efecto.
En algunos casos es importante modificar el archivo de configuración de POV-Ray, este se guarda en el sistema de archivos en red, se puede acceder a él mediante el siguiente comando:

```
# cd /share/apps/etc/povray/3.6/povray.conf

…
[File I/O Security]
none            ; all read write operations on files are allowed
;read-only      ; uses the read+write directories for writing (see below)
;restricted     ; uses_only_ ”read” and “read+write” directories for file I/O
…
```

Para probar el correcto funcionamiento de POV-Ray en el cluster se realiza lo siguiente:

Iniciar sesión como un usuario diferente a root.

Se copia en el directorio home alguna escena que viene incluida en POV-Ray.

```
$ cp /share/apps/src/povray-3.6.1/scenes/advanced/sunsethf.pov /home/usuario
```

Ejecutar POV-Ray en el frontend, los parámetros de POV-Ray que se utilizaron: +D muestra la imagen que se está renderizando, +W indica el ancho de la imagen, +H indica la altura de la imagen y +I la ruta de la escena que se va a renderizar.

```
$ povray +D +W640 +H480 +I sunsethf.pov
```

Ejecutar POV-Ray en un nodo.

```
$ ssh compute-0-0 /share/apps/bin/povray +D +W640 +H480 +I sunsethf.pov
```

En la ejecución del anterior comando se debe indicar la ruta del binario, cuando se utiliza SSH las variables de entorno no funcionan.

3.1.3.2 Instalación Zebra-Core. A continuación se va a explicar la instalación de la herramienta complemento a POV-Ray que permite renderizar imágenes fotorrealistas en clusters.
El paquete de software fue proporcionado por los autores, no es posible descargarlo de ninguna web, sin permiso. Asegúrese que POV-Ray esté instalado correctamente.

Una vez adquirido el archivo, se copia a /share/apps/src

cp zebra-core.tar.gz /share/apps/src

Se procede a descomprimir y desempaquetar con el siguiente comando:

tar zxvf zebra-core.tar.gz

Una vez realizado lo anterior se accede a la carpeta que se ha creado

#cd ./zebra-core

El archivo zebra-core.config explica que variables de entorno se deben ajustar. En el archivo /etc/profile se exportan las siguientes variables de entorno:

…
export PREFIX=/share/apps
export TMPDIR=/tmp
export POVEXE=$PREFIX/bin
export POVINC=$PREFIX/share/povray-3.6/include
export SCRATCHDIR=$PWD/scratch

Es necesario guardar cambios y ejecutar los siguientes comandos para que las variables de entorno se agreguen al archivo /etc/profile en todos los nodos.

make –C /var/411 clean
make –C /var/411
rocks run host “411get”

Verificar si los comandos anteriores hicieron efecto. Si no ocurrió ningún cambio, reinicie todo el cluster.

El funcionamiento de Zebra-Core se realiza mediante línea de comandos.
Para empezar a utilizar Zebra-Core, se abre el directorio donde anteriormente se copiaron los archivos.

$ cd /share/apps/src/zebra-core

El primer archivo que se debe ejecutar para empezar a utilizar el programa es:
$ ./zebra-core-strips [opciones] <archivo_pov> <ancho> <alto <numero_de_franjas>

En [opciones]:
-h: Ayuda
-v ó –z: Es la orientación que se le quiere dar al particionamiento de la imagen.
-s: Es un archivo con las coordenadas de las franjas.
-N: Es el nombre con el cual se identifica el trabajo que se va a iniciar, en caso de omitirlo, este se asigna de manera automática.

Parámetros
<archivo_pov>: Es la escena en POV-Ray que se quiere renderizar, debe estar almacenada en /share/apps/src/zebra-core/scenes.
<ancho> <alto>: Indican la resolución de la imagen.
<numero_de_franjas>: Indica el número de las franjas de la imagen.

Se procede con la generación de scripts para el gestor de colas, con el comando:

$ ./zebra-core-procs [opciones] <nombre_trabajo>

En [opciones]:
-h: Ayuda.
-n: Indica el número de procesadores a utilizar.
-d: Ejecuta el proceso de forma distribuida.
-s: Ejecuta el proceso de forma secuencial.
-c: Ejecuta de forma concurrente en cada procesador.

Parámetros
<nombre_trabajo>: Indica el nombre del trabajo, al cual se le van a generar los scripts.

Una vez se tienen los script generados, es necesario cambiar los permisos al directorio outputs, mediante el siguiente comando:
$ chmod 755 runs/<nombre_trabajo>/outputs

En caso de no hacer lo anterior, los scripts no se ejecutarán y ocasionarían un error.
Una vez se otorguen permisos al directorio se procede a ejecutar el siguiente comando:

$ ./zebra-core-launch [opciones] <nombre_trabajo>

En [opciones]:
-h: Ayuda.

-n: Indica el número de procesadores a utilizar.
-d: Ejecuta el proceso de forma distribuida.
-s: Ejecuta el proceso de forma secuencial.
-c: Ejecuta de forma concurrente en cada procesador.

Parámetros
<nombre_trabajo>: Indica el nombre del trabajo, cuyos scripts se ejecutarán en el sistema gestor de colas SGE.

Por último, se procede a unir las franjas de la imagen con el siguiente comando:

$./zebra-core-build [opciones] <nombre_trabajo>

En [opciones]:
-h: Ayuda
-s: Archivo con las coordenadas de las franjas.
-v ó –z: Es la orientación que se asignó a la imagen.
Parámetros
<nombre_trabajo>: Indica el nombre del trabajo en el cual se van a unir las franjas.

La imagen final se almacenará en la variable de entorno $SCRATCHDIR/<nombre_trabajo>.

3.1.4 Instalación de herramienta para minería de datos en paralelo

3.1.4.1 Instalación WEKA-Parallel. Primero se adquiere de http://sourceforge.net/projects/weka-parallel el archivo de tipo .jar, el cual se copia este archivo al directorio /share/apps/src

cp weka-3-2-3-parallel.jar /share/apps/src

Se descomprime el archivo con el siguiente comando:

jar -xvf weka-3-2-3-parallel.jar

Se crea un nuevo directorio cuyo nombre es weka-p en /usr/local

mkdir /usr/local/weka-p

Del archivo que se descomprimió anteriormente, se crea un nuevo directorio que contiene varios archivos, de los cuales se copia weka.jar en el directorio /usr/local/weka-p.

cp weka.jar /usr/local/weka-p

En los nodos se debe copiar el directorio /usr/local/weka-p de la siguiente manera:

scp –rp /usr/local/weka-p root@ <nombre_nodo>:/usr/local

Donde <nombre_nodo> es el nombre que Rocks asignó al nodo, el alias o la IP.

Ahora en cada máquina que conforme el cluster se abre el directorio /usr/local/weka-p.

cd /usr/local/weka-p

Para los nodos se hace mediante SSH

ssh root@compute-0-0
cd /usr/local/weka-p

Se ejecuta el DistributedServer mediante la siguiente instrucción:

java –cp weka.jar weka.core.DistributedServer xxxx

Donde xxxx es el número del puerto disponible para Weka-Parallel. El comando anterior se debe ejecutar en el servidor y en cada uno de los nodos.

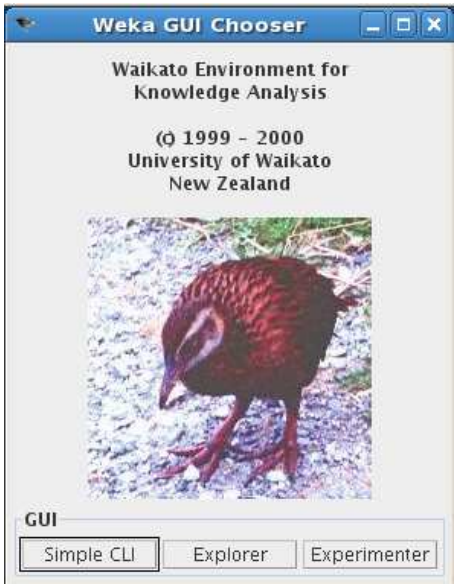
En el frontend, se abre una terminal en modo gráfico, se dirige a la ruta /usr/local/weka-p mediante la siguiente instrucción:

cd /usr/local/weka-p

Se ejecuta Weka-Parallel en modo gráfico (ver Figura 54).

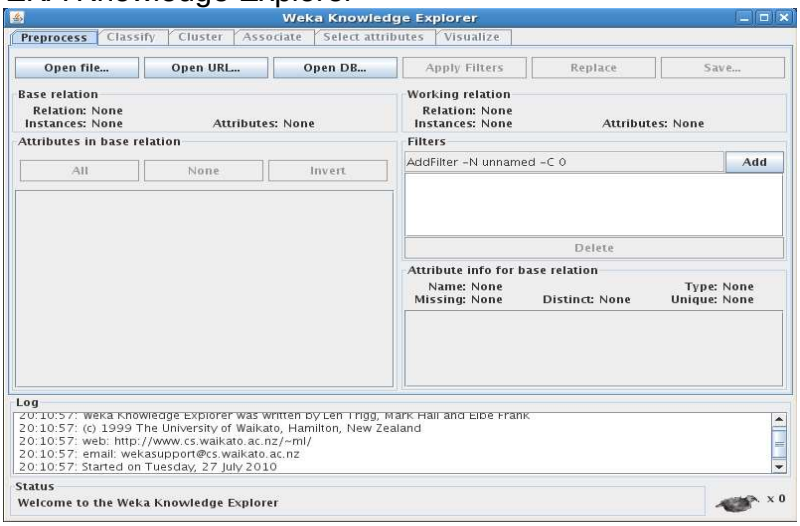
java –cp weka.jar weka.gui.GUIChooser

Figura 54. Interfaz Inicial WEKA



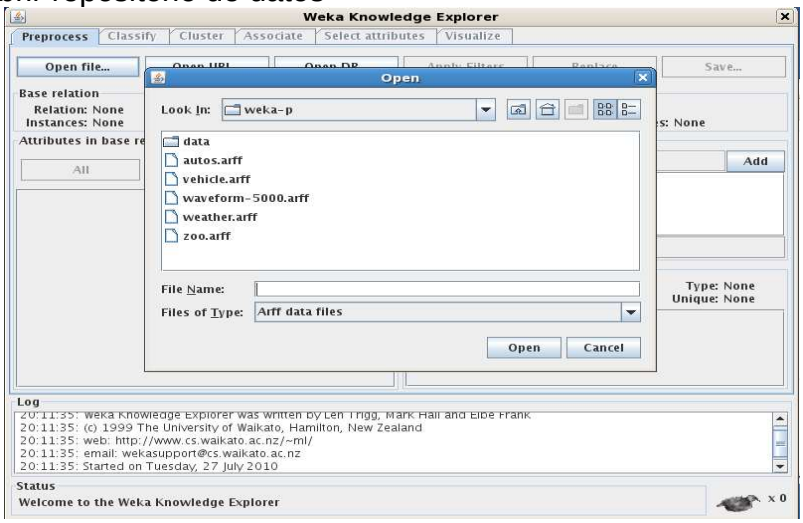
Se oprime el botón Explorer y aparece una interfaz como la que se muestra en Figura 55.

Figura 55. WEKA Knowledge Explorer



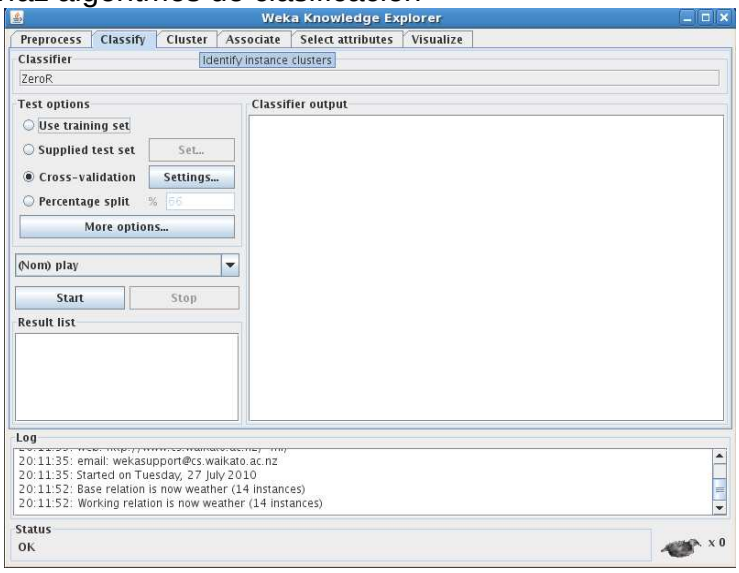
Esta interfaz es la principal de Weka, aquí se cargan los archivos o bases de datos a las cuales se les van a aplicar las técnicas de minería de datos.
Se da clic en el botón Open file… y aparece una ventana como se muestra en la Figura 56.
Esta ventana contiene un filtro que solo muestra los archivos de tipo ARFF que son los que maneja Weka.

Figura 56. Abrir repositorio de datos



Una vez abierto el repositorio, nótese que las pestañas superiores de Figura 55 se han habilitado, para configurar el puerto y los equipos que van a hacer validación cruzada, se hace clic en la pestaña Classify.

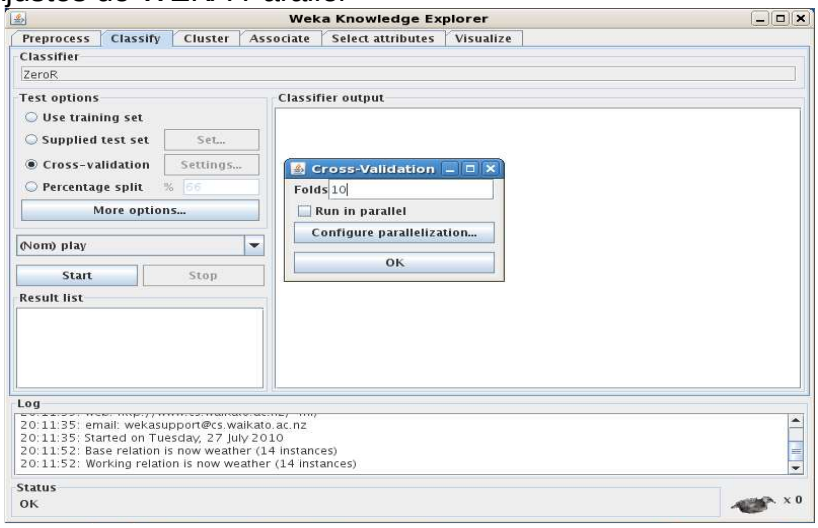
Figura 57. Interfaz algoritmos de clasificación



En esta sección se hacen los ajustes necesarios para que Weka funcione en paralelo con algoritmos de validación cruzada.

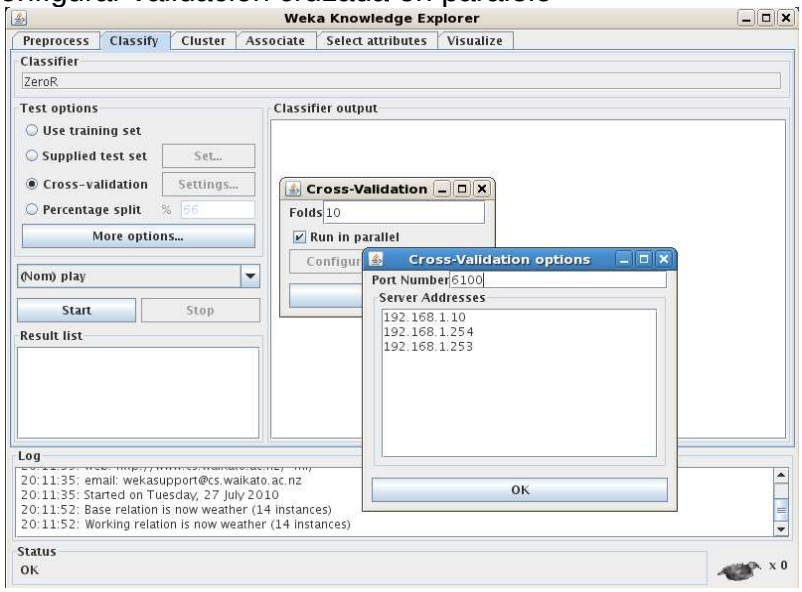
Se puede observar en Figura 57 que en la parte izquierda de la pantalla hay un botón marcado como Settings…, se oprime esté botón y abre una ventana como la que aparece en Figura 58.

Figura 58. Ajustes de WEKA-Parallel



Ahora se activa la casilla de verificación marcada como Run in parallel y se oprime en Configure parallelization y se abre otra ventana como se muestra en Figura 59.

Figura 59. Configurar validación cruzada en paralelo



En Port Number, se escribe un número de puerto disponible para el DistributedServer. En Server Addresses se escribe por cada línea las direcciones IP de los nodos.

3.1.5 Instalación de herramienta para programación en paralelo

3.1.5.1 Instalación MPJ Express. Antes de la instalación de MPJ Express, se debe asegurar que tiene instalado Java 1.5 o superior.
De http://sourceforge.net/projects/weka-parallel/ se descarga el programa en formato .tar.gz.

Se copia este paquete a /share/apps/src

#cp mpj-v0_35.tar.gz /share/apps/src

Se descomprime y desempaqueta mediante el siguiente comando:

tar zxvf mpj-v0_35.tar.gz

Para mayor comodidad se copia en /usr/local el directorio que se ha creado después de la descompresión.

cp –Rf mpj-v0_35 /usr/local/mpj

Se cambian el propietario, el grupo y los permisos del directorio con los siguientes comandos:

chown –Rf usuario /usr/local/mpj
chgrp –Rf usuario /usr/local/mpj
chmod -Rf 755 /usr/local/mpj

Para continuar con la instalación en los nodos, se copia el directorio /usr/local/mpj en la ubicación donde se encuentra en el frontend.

#scp –rp /usr/local/mpj root@<nombre_nodo>/usr/local

Donde <nombre_nodo> es el nombre que Rocks asignó al nodo, el alias o la IP. Se procede a cambiar el propietario, el grupo y los permisos del directorio que se copió en el nodo.
Se agregan las variables de entorno en el archivo /etc/profile

…
export MPJ_HOME=/usr/local/mpj

```
export MPJ_DAEMON=$MPJ_HOME/bin/mpjdaemon_linux_x86_32
export PATH=$PATH:$MPJ_HOME/bin
…
```

Se guardan los cambios y se ejecuta:

```
# make –C /var/411 clean
# make –C /var/411
```

Ahora con la ejecución del siguiente comando se mandan los nuevos archivos a los nodos.

```
# rocks run host “411get”
```

Una vez realizado esto se inicia sesión como un usuario diferente a root

En el directorio home de este usuario se crea un directorio llamado mpj-user

```
$ cd /home/usuario
$ mkdir mpj-user
```

Posterior a los comandos anteriores se crea un archivo en el editor de texto de preferencia, llamado machines. En él se escriben en cada línea las direcciones IP de los nodos donde se quiere ejecutar MPJ Express.

Por ejemplo:

```
192.168.1.10
192.168.1.254
192.168.1.253
```

Para iniciar el demonio en los nodos se ejecuta:

```
# mpjboot machines
```

Este comando hace SSH a cada máquina que está listada en el archivo que anteriormente se creó.

Si el demonio no se ejecuta en todos los nodos realice lo siguiente:

```
$ ssh <nombre_nodo>
$ MPJ_HOME/bin/mpjdaemon_linux_<arquitectura_maquina> start
```

Donde <nombre_nodo> es la dirección IP o el nombre que tiene el nodo.
<arquitectura_maquina> es la clase de arquitectura del nodo.

De otra forma puede ejecutar la siguiente línea de comando:

$ MPJ_DAEMON start

Hasta este punto se tiene la configuración de la herramieta MPJ Express, para comprobar el funcionamiento de MPJ Express, se compila el siguiente archivo.

HelloWorld.java

```
import mpi.*;
public class HelloWorld{
public static void main(String args[]) throws Exception {
        MPI.Init(args); int me = MPI.COMM_WORLD.Rank();
        int size = MPI.COMM_WORLD.Size();
        System.out.println("Hi from <"+me+">");
        MPI.Finalize();
        }
}
```

Para ejecutar un programa paralelo se realiza lo siguiente:
Primero se debe compilar la clase HelloWorld.java con el compilador básico de Java, especificando las bibliotecas de MPJ Express.

$ javac -cp .:$MPJ_HOME/lib/mpj.jar HelloWorld.java

Si no se presenta ningún error se procede a ejecutar el siguiente comando:

$ mpjrun.sh -np 2 –dev niodev HelloWorld

Donde –np 2, es el número de procesos que se quieren utilizar.
-dev niodev, indica que se está utilizando la configuración cluster, es decir que va a utilizar la red.

Para detener el demonio en todos los nodos se ejecuta el siguiente comando:

mpjhalt machines

Para detener el demonio debe ser usuario root.

4. MANEJO BÁSICO DE ROCKS

Este capítulo se presenta como una guía al usuario del cluster de alto rendimiento para que conozca los aspectos más importantes del funcionamiento del cluster como también de las herramientas con las que cuenta la plataforma Rocks.
Se hace referencia a las aplicaciones de monitoreo y control, de igual forma se mencionan los comandos básicos para una buena administración del sistema.
El usuario debe conocer los comandos básicos de Linux como manejo de archivos y directorios además de tareas básicas para la comunicación.
Este manual ha sido ordenado en capítulos que explican detalladamente las funciones más relevantes de la plataforma Rocks, el fin de este documento es orientar de una manera más cómoda al usuario cuando haga uso de este tipo de tecnologías. Además cuenta con material de ayuda audiovisual para mayor comprensión.

4.1 INTRODUCCIÓN A LA PLATAFORMA

4.1.1 Encender y apagar el cluster. Principalmente se debe asegurar que los equipos estén correctamente conectados a la alimentación eléctrica, una vez hecho esto se procede a encender el frontend, seguido de este se van enciendo los nodos. Posteriormente inicia sesión en el frontend.
Para apagar el cluster es muy importante que primero apague todos los nodos, debe estar como usuario root.

```
# rocks run host “poweroff”
```

Después de haber realizado esto puede proceder a apagar el frontend, en este caso no importa el usuario con el que se encuentre iniciado sesión.

```
# poweroff
```

4.1.2 Conectarse a un nodo. Antes de conectarse a un nodo verifique que los equipos envíen una respuesta, utilizando en comando ping.

```
$ ping 192.168.1.10
$ ping compute-0-0.local
```

Puede también hacer un ping para todos los nodos del cluster y observar cuales están activos.

$ ping –b 192.168.0.0

Para conectarse a los nodos se inicia sesión con el servicio SSH desde una terminal de tres maneras diferentes

$ ssh <nombre_nodo>
$ ssh <usuario>@<dir_IP>
$ ssh <dir_IP>

Donde <nombre_nodo> es el nombre o alias que Rocks asigna al nodo al momento de la instalación.
<usuario> es el nombre del usuario que se encuentra registrado en el frontend.
<dir_IP> es la dirección IP del nodo a conectarse.
La primera vez que se utiliza este servicio, el sistema genera y guarda las llaves RSA por eso se demora un poco, en sesiones posteriores esto no sucede.

El nombre del nodo que Rocks asigna al nodo tiene el prefijo compute-x-x, las x se reemplazan por los números de cada nodo los cuales pertenecen al dominio .local. Rocks asigna unos sobrenombres a los nodos, estos aliases tienen el prefijo c-x-x.

El listado de todos los nodos se guarda en /etc/hosts.

```
#/etc/hosts
127.0.0.1       localhost.localdomain   localhost
192.168.1.10    tinku.local             tinku   #frontend
192.168.1.254   compute-0-0.local       compute-0-0
192.168.1.253   compute-0-1.local       compute-0-1
192.168.1.10    tinku.org
```

4.1.3 Sistemas de archivos. Cada usuario que está registrado tiene asignado su directorio home. Este tipo de directorio es: /home/<nombre_usuario>. En un cluster es importante que los archivos se puedan acceder fácilmente cuando se haga una conexión remota.
En Rocks existen dos maneras que proporcionan acceso a los archivos sin importar el nodo en el cual está conectado. Por un lado está el NFS (Network File System), el cual carga el sistema de archivos del directorio home del usuario a través de toda la red en el nodo en el cual se conecta, cualquier cambio efectuado se verá reflejado en el frontend y en los nodos. Este sistema realmente se guarda en el frontend.

El servicio autofs garantiza que los sistemas de archivos se carguen de forma automática y que suceda en el momento en el que el usuario inicie sesión, también se encarga de desmontar los archivos cuando el usuario cierra sesión.
Para verificar el sistema de archivos realice lo siguiente:

Cree un archivo vacio en el frontend.

$ touch archivo-frontend

Conéctese a uno de los nodos y verifique que el archivo que creó anteriormente se encuentra.

$ ssh compute-0-0
$ ls

Ahora cree otro archivo vacío desde el nodo y salga de SSH.

$ touch archivo-nodo
$ exit

Desde el frontend revise que pueda acceder al archivo que creó en el nodo.

4.1.4 Ejecución de comandos remotos. A veces se requiere ejecutar un comando simple en un nodo sin acceder de manera remota a ese nodo. El comando SSH permite realizar este tipo de acciones, por ejemplo para ejecutar un comando en el nodo 2, en este caso w que muestra los usuarios conectados.

$ ssh compute-0-2 w

Cuando se ejecutan los comandos de esta manera, el directorio base del comando es el mismo directorio home del usuario sin importar que el comando sea ejecutado desde otro directorio.
Para comprobar esto se realiza el siguiente ejemplo:

$ mkdir dir
$ cd dir
$ ssh compute-0-0 pwd

El comando pwd muestra el directorio base sobre el cual se ejecutan los comandos.

4.1.5 Comandos propios de Rocks. Rocks incluye sus propios comandos para la administración del cluster. Se van a explicar los más importantes. Tenga en cuenta un usuario diferente a root puede ejecutar ciertos comandos.

- **rocks run host**

$ rocks run host “<comando>”
Este comando concatena otro comando de manera secuencial en todos los nodos activos. Si quiere que se ejecute en determinados nodos ejecute la siguiente instrucción:
$ rocks run host compute-0-0 compute-0-1 “<comando>”

- **insert-ethers**

insert-ethers
Este comando adiciona un nuevo nodo a la base de datos MySQL del cluster, detectándolo de manera automática.

- **add roll**

add roll <clean=bool> <roll>
Este comando adiciona un roll en una imagen ISO, en el directorio donde el frontend almacena el resto de rolls.
Argumentos:
<clean=bool>: Si es verdadero borra los archivos de cualquier roll existente que tenga el mismo nombre, versión o arquitectura antes de copiar el nuevo roll en el disco.

rocks add roll clean=1 kernel*iso
Esta línea adiciona el Kernel Roll a un directorio local. En este caso antes que el roll sea agregado el Kernel Roll instalado anteriormente será borrado.

- **rocks remove roll**

rocks remove roll < roll>
Remueve un roll de la base de datos y del sistema de archivos.

- **rocks enable roll**

rocks enable roll <roll> <arch=string> <version=string>
Habilita el roll disponible. El roll obligatoriamente debe estar copiado en el frontend.
Argumentos:
<roll>: debe ser el nombre base del roll.
<arch=string>: es la arquitectura para la cual se va a habilitar el roll.
<versión=string>: el número de la versión del roll que se quiere habilitar, si no se proporciona este argumento, todas las versiones disponibles quedarán habilitadas.

- **tentakel**

tentakel <comando>

Ejecuta un comando en paralelo en todos los nodos.

- **rocks-console**

rocks-console <nodo>
Realiza una conexión remota a un nodo que se está instalando.

- **rocks sync users**

rocks sync users
Actualiza los archivos que se refieren al usuario los cuales hayan sido modificados, además reinicia el servicio autofs.

- **rocks report distro**

$ rocks report distro
Muestra el directorio donde se encuentran los archivos de la distribución actual del cluster.

- **rocks report version**

$ rocks report version
Muestra la versión de Rocks que está utilizando.

- **rocks list host**

$ rocks list host
Muestra una lista de todos los nodos agregados, incluyendo el frontend.

4.1.6 Monitoreo de los recursos principales. Como recursos primordiales se tiene el procesador, la memoria RAM y el disco duro, siempre es bueno revisarlos para asegurarse del buen funcionamiento del cluster.

- Procesador. Es el recurso más importante para la revisión es el procesador, mediante un comando propio de Linux que muestra la carga promedio, si este supera el numero de procesadores entonces el procesador estará muy ocupado.

  $ uptime
  $ rocks run host “uptime”

- Memoria RAM. Se puede observar la cantidad de memoria total y el espacio de memoria libre.

  $ grep –A 1 /proc/meminfo
  $ rocks run host “grep –A 1 /proc/meminfo”

- Disco Duro. Para observar las particiones fijas y temporales, el tamaño, la cantidad de disco que ha sido utilizada, la cantidad de disco disponible, el porcentaje de uso, y el directorio en el cual se encuentra cargada.
  $ df –h
  $ rocks run host “df -h”

4.1.7 Manejo sistema gestor de colas Sun Grid Engine. Este programa es una aplicación que se encarga de trabajos que se ejecutan de manera desatendida, conocidos popularmente como procesos por lotes. Sun Grid Engine, en adelante SGE provee una interfaz gráfica de fácil uso.
Es muy importante contar con este tipo de herramientas porque la ejecución de algún proceso no controlada puede llegar a desbalancear el cluster, ocasionando un mal funcionamiento.

La ejecución de un proceso en SGE se realiza en cuatro pasos que se explicarán detalladamente.

- **Preparación del script.** Cada trabajo que se va a enviar a SGE debe tener un archivo de tipo Shell script, el archivo contiene los datos esenciales para la ejecución del proceso.

  ejemplo.sh
  #!/bin/bash
  #Script de lanzamiento
  #Opciones:
  #$-S /bin/bash
  #$-N ejemplo
  #$-cwd
  date
  sleep 10
  date

  Para indicar comentarios se utiliza #.
  Las líneas que comienzan con #$ son las opciones de lanzamiento de SGE.

  -S: indica que shell se va a utilizar, en este caso el shell escogido es bash.
  -N: mediante esta opción el usuario le asigna un nombre al proceso debe utilizarse caracteres alfanuméricos sin espacios.
  -cwd: con esta opción el proceso se ejecutará desde el directorio actual y no el directorio home que está por defecto.
  En el ejemplo el programa muestra la fecha actual del sistema, hace una pausa de diez segundos, nuevamente muestra la fecha.
  Para ver las opciones que acepta SGE ejecute:

$ qsub man

- **Envío del proceso a la cola.** Para enviar el trabajo anterior a la cola de trabajos se ejecuta la siguiente instrucción.

  $ qsub ejemplo.sh

  Your job 200 ("ejemplo") has been submitted

  Para obligar al gestor a ejecutar el trabajo una vez enviado, se ejecuta el siguiente comando:

  $ qsub –now y ejemplo.sh

  SGE asigna un número de trabajo secuencial que sirve para monitorear el proceso.

- **Monitoreo del proceso.** Para verificar que el proceso está en la cola de ejecución se utiliza el comando:

  $ qstat

job-ID prior name user state submit/start at queue slots ja-task-ID
---
200 0.55500 ejemplo usuario r 11/02/2010 09:01:33 all.q@compute-0-0.local 1

La salida de este comando muestra el identificador asignado al proceso, la prioridad, el nombre del proceso, el usuario, el estado del trabajo (r: ejecutándose, q: en cola, w: esperando, e: errores), la fecha de envío y la hora de ejecución, la cola en la cual se está ejecutando, y el identificador de proceso en un arreglo de procesos.

Para ver todas las colas de ejecución presentes en el cluster se ejecuta:

$ qstat –f

queuename qtype used/tot. load_avg arch states
---
all.q@compute-0-0.local BIP 0/1 0.00 lx26-x86
---
all.q@compute-0-1.local BIP 0/1 0.00 lx26-x86
---
all.q@compute-0-2.local BIP 0/1 0.00 lx26-x86

La información desplegada por este comando incluye el nombre de la cola, que por defecto tiene la forma all.q@<nodo>, el tipo de cola (B:Batch, I:interactive, P:parallel), el número de procesadores usados y disponibles, la carga del procesador en los últimos cinco minutos, la arquitectura del nodo y el estado del nodo, este indica cuando tiene un estado anormal.

Para suprimir un proceso de la cola de trabajos.

$ qdel 200

usuario has registered the job 200 for deletion

- **Recolección de los resultados.** Para revisar los resultados de la ejecución del proceso, se puede obtener información detallada sobre la tarea mediante el comando:

  $ qacct –j 200

qname	all.q
hostname	compute-0-0.local
group	usuario
owner	usuario
project	NONE
department	defaultdepartment
jobname	ejercicio
jobnumber	200
taskid	undefined
account	sge
priority	0
qsub_time	Thu Feb 11 09:01:33 2010
start_time	Thu Feb 11 09:02:07 2010
end_time	Thu Feb 11 09:03:51 2010
granted_pe	NONE
slots	1
failed	0
exit_status	0
ru_wallclock	126
ru_utime	121
...	
cpu	126
mem	93.296
io	0.000
iow	0.000
maxvmem	768.391M

SGE genera archivos de tipo .ox y .ex, donde la x indica el número de proceso asignado por el gestor. El primer tipo de archivo muestra la salida en pantalla, mientras el segundo muestra los errores que se produjeron.
Para visualizar la interfaz gráfica de SGE, en una terminal ejecute la siguiente instrucción:

$ qmon

4.2 ADMINISTRACIÓN Y MONITOREO

4.2.1 Sistemas de archivos locales. De los sistemas de archivos que se encuentran en el cluster se mencionan los siguientes directorios:

- /export/home. Este directorio almacena los directorios home de todos los usuarios del sistema.

- /export/home/install. Este directorio contiene la todos los paquetes de instalación, archivos especiales de configuración que Rocks utiliza para ejecutar la instalación en los nodos.

- /export. Este directorio en realidad es un enlace simbólico a la partición.

- /state/partition1. En el guarda todos los sistemas de archivos que se cargan por red.

Para cargar los directorios contenidos en /export, es necesario configurar el servicio autofs, los archivos de configuración de este residen en /etc de todos los equipos incluyendo el frontend.
Cuando se realizan cambios en los archivos de autofs, es necesario reiniciar el servicio de la siguiente manera:

service autofs reload
rocks run host “service autofs reload”

4.2.2 Servicio 411. El servicio 411 Secure Information System permite que los archivos de configuración básicos para el funcionamiento de la plataforma se puedan acceder desde cualquier equipo en el cluster actualizando los cambios que se hacen en los archivos de configuración.

411 utiliza servicios web para publicar en el frontend de forma encriptada los archivos de configuración que van a ser compartidos y poder descargarlos en cada nodo.
Los archivos que son compartidos se encuentran en el archivo /var/411/Files.mk

```
...
AUTOMOUNT = #(wildcard /etc/auto.*)
# These files all take a "#" comment character.
# If you alter this list, you must do a 'make clean; make'.
FILES = #(AUTOMOUNT) \
/etc/passwd \
/etc/shadow \
/etc/group \
/etc/services \
/etc/rpc
# FILES += /my/file
...
```

El servicio 411 realiza de forma automática la sincronización de los archivos de configuración en el cluster. En algunas situaciones es necesario forzar la sincronización después de que se ha hecho un cambio en los archivos de configuración (creación de un usuario, modificación de la configuración del servicio autofs, creación de un nuevo grupo, etc). La sincronización se puede realizar de tres maneras diferentes:

- Mediante service.

  ```
  # service 411 commit
  ```

- Actualizando todos los archivos que se han modificado últimamente.

  ```
  # make –C /var/411 force
  # make –C /var/411
  ```

- Utilizando 411 en todos los nodos.

  ```
  # rocks run host “411get”
  ```

  Este servicio es útil para detectar si los archivos no se están transmitiendo de manera correcta, en caso de que el comando no esté funcionando se obtiene la siguiente salida:

```
compute-0-0:
Error: Could not reach a master server. Masters: [http://10.1.1.1/411.d/ (-1)]
compute-0-1:
```

Error: Could not reach a master server. Masters: [http://10.1.1.1/411.d/ (-1)]

4.2.3 Administración de usuarios. Antes de instalar los nodos se recomienda crear las cuentas de usuario.

useradd usuario

Una vez creada la cuenta se asigna como directorio home de la nueva cuenta la ruta /export/home/usuario. Para que el usuario encuentre su directorio home en otros nodos, este debe ser cambiado a /home.

usermod –d /home/usuario usuario

Se asigna una contraseña a usuario.
passwd usuario

Se procede a configurar la cuenta para que todos los nodos se conecten de manera transparente en el cluster. Para esto se configura el archivo autofs para que el directorio home se monte automáticamente en los nodos.

- **Configuración de Autofs.** Se agrega la siguiente línea al archivo /etc/auto.home

  usuario tinku.local:/export/home/usuario

- **Sincronización Servicio 411.** Los cambios anteriormente hechos deben sincronizarse en todo el cluster.

  # make –C /var/411 force
  # service autofs reload
  # rocks run host service autofs reload
  # rocks sync users

Es muy importante revisar si el usuario puede encontrar su directorio home.

- **Modificación cuentas de usuario.** Es posible cambiar el grupo al cual pertenece el usuario o la contraseña.

  # usermod –g <gid> usuario

  Donde <gid> es el nuevo identificador de grupo al que se desea que pertenezca el usuario.

  Se requiere hacer una sincronización, para ello se utiliza:

service 411 commit

El sistema 411 realiza una sincronización automática cada hora.

- **Eliminación cuentas de usuario.**
  # userdel usuario
  # umount /home/usuario
  # rocks run host “umount /home/usuario”
  # rm –Rf /export/home/usuario
  # make –C /var/411 force
  # service autofs reload
  # rocks run host “service autofs reload”
  # rocks sync users

4.2.4 Instalación software adicional

En Rocks existen tres maneras para agregar nuevo software.

- **Instalación de paquetes de fuentes:** Para un mejor entendimiento se explicará la forma de instalar POV-Ray un programa comprimido en un archivo de tipo tar.gz, disponible en http://www.povray.org/download/.
  Permite configurar en detalle los archivos binarios y los directorios en los cuales se almacenarán los archivos de instalación.

  Primero se copia el archivo en /share/apps/src, mediante el siguiente comando:

  #cp povray-3.6.tar.gz /share/apps/src

  Se procede a descomprimir y desempaquetar el archivo:

  # cd /export/apps/src
  # tar zxvf povray-3.6.tar.gz

  Después de haber realizado lo anterior, abre el directorio que se ha creado.

  # cd povray-3.6.1

  Se preparan los archivos para la compilación de POV-Ray mediante el siguiente comando:

./configure COMPILED_BY ”Jorge Mora y Evelyn Vinueza <clustertinku@gmail.com>” --prefix=/share/apps

La opción --prefix=/share/apps, permite el acceso al programa a todos los nodos.

Adicionalmente, si desea saber que otras opciones de instalación tiene POV-Ray ejecute:

./configure –help

Una vez seleccionado el directorio destino, se procede a compilar las fuentes:

make

También es posible utilizar make check, que al final de todo el proceso realiza una prueba de renderización.

make check

Si no aparece ningún mensaje de error en la compilación se procede con la instalación.

make install

Es necesario modificar el archivo /etc/profile, para agregar al PATH el nuevo programa.

…
export PATH=$PATH:/share/apps/bin
…

Ahora se debe modificar el archivo /var/411/Files.mk para que la línea anterior se agregue a todos los nodos del cluster.

...
FILES = $(AUTOMOUNT) \
/etc/passwd \
/etc/shadow \
/etc/group \
/etc/services \
/etc/rpc \
/etc/profile
...

Se guardan los cambios y se ejecuta:

make –C /var/411 clean
make –C /var/411

Ahora con la ejecución del siguiente comando se mandan los nuevos archivos a los nodos.

#rocks run host “411get”

Es necesario reiniciar el cluster para que los cambios hagan efecto.
En algunos casos es importante modificar el archivo de configuración de POV-Ray, este se guarda en el sistema de archivos por red, se puede acceder a él mediante el siguiente comando:

cd /share/apps/etc/povray/3.6/povray.conf
…
[File I/O Security]
none ; all read write operations on files are allowed
;read-only ; uses the read+write directories for writing (see below)
;restricted ; uses_only_ ”read” and “read+write” directories for file I/O
…

Para comprobar el correcto funcionamiento de POV-Ray en el cluster se realiza lo siguiente:

Iniciar sesión como un usuario diferente a root.

Copiar en el directorio home alguna escena que viene incluida en POV-Ray.

$ cp /share/apps/src/povray-3.6.1/scenes/advanced/sunsethf.pov /home/usuario

Ejecutar POV-Ray en el frontend. Los parámetros de POV-Ray que se especifican son: el ancho, alto de la imagen resultante y la escena que se va a renderizar.

$ povray +D +W640 +H480 +I sunsethf.pov

Ejecutar POV-Ray en un nodo utilizando SSH

$ ssh compute-0-0 /share/apps/bin/povray +D +W640 +H480 +I sunsethf.pov

En este comando se debe indicar la ruta del binario, cuando se utiliza SSH las variables de entorno no funcionan.

- **Instalación de RPM:** La instalación de binarios es la opción más cómoda para la instalación de nuevo software en el cluster. Aunque no es necesariamente la mejor. El RPM tiene configurado los directorios donde se instalarán los archivos que se incluyen con el paquete, para una plataforma como un cluster no puede ser la mejor opción.

  Para explicar el proceso de instalación se utiliza el paquete de graficación gnuplot el cual es un graficador científico.
  Antes de instalar un paquete, debe cerciorarse que no se encuentre instalado, o que no se incluya en el sistema operativo.

  # rpm – gnuplot
  Package gnuplot is not installed

  En caso contrario, el comando anterior muestra toda la información correspondiente al paquete.

  Para instalar el paquete se realiza de la siguiente manera:

  # rpm –Uvh gnuplot-3.7.3-2.i386.rpm
  Preparing... ### [100%]
  1:gnuplot ### [100%]

  El proceso anterior se realiza únicamente para instalar el paquete en el frontend.

  Para instalar el paquete en el resto de nodos hay dos formas.

- **Instalación distribuida.** Se copia el archivo rpm en /export/apps/src

  # cp –Rf gnuplot-3.7.3-2.i386.rpm /export/apps/src

  Se comprueba que el archivo es visible desde todos los nodos:

  # rocks run host “ls – l /share/apps/src”

  Una vez que se comprueba que el archivo es visible para todos los equipos, se procede con la instalación:

  # rocks run host “rpm –Uvh gnuplot-3.7.3-2.i386.rpm”

compute-0-0:
Preparing... ##
gnuplot ##
compute-0-1:
Preparing... ##
gnuplot ##
compute-0-2:
Preparing... ##
gnuplot

Una vez realizado lo anterior se verifica la ruta del binario en todos los nodos.

rocks run host “which gnuplot”
compute-0-0:
/usr/bin/gnuplot
compute-0-1:
/usr/bin/gnuplot
…

- **Instalación incluyendo el paquete.** Para un cluster lo ideal es que una vez se instalen los nodos, el paquete también se instale.
  Rocks tiene un directorio especial, en el cual se almacenan aportes a la distribución, se encuentra en /export/rocks/install/contrib/<versión>/<arch>.
  En el ejemplo se copia al subdirectorio RPMS de la ruta anterior el archivo que se quiere incluir.

  # cp –Rf gnuplot-3.7.3-2.i386.rpm
  /export/rocks/install/contrib/5.2/i386/RPMS

  Ahora se procede a configurar la distribución para que incluya el nuevo paquete, y sacar una nueva versión.

  El archivo /export/rocks/install/site-profiles/5.2/nodes/skeleton.xml es una plantilla para redactar un nuevo archivo de configuración que se llamará extend-compute.xml. Antes de modificar el original, se recomienda hacer una copia con los siguientes comandos:

  # cd /export/rocks/install/site-profiles/5.2/nodes
  # cp skeleton.xml extend-compute.xml

  Abre el archivo extend-compute.xml en su editor favorito, y aparece lo siguiente:
  ...
  <changelog>

```
</changelog>
<main>
<!-- kickstart 'main' commands go here, e.g., partitioning info -->
</main>
<!-- There may be as many packages as needed here. Just make sure you only uncomment as many package lines as you need. Any empty <package></package> tags are going to confuse rocks and kill the installation procedure
-->
<!-- <package> insert your 1st package name here and uncomment the line</package> -->
<!-- <package> insert your 2nd package name here and uncomment the line</package> -->
<!-- <package> insert your 3rd package name here and uncomment the line</package> -->
...
```

Se modifican las líneas donde se incluyen los paquetes que quieran ser parte de la distribución, el archivo debe quedar de la siguiente manera:

```
...
<changelog>
</changelog>
<main>
<!-- kickstart 'main' commands go here, e.g., partitioning info -->
</main>
<package>gnuplot</package>
...
```

Una vez escrito el archivo se procede a reconstruir la distribución utilizando un comando propio de Rocks.

```
# rocks-dist dist
Cleaning distribution
Resolving versions (base files)
including "kernel" (4.2.1,i386) roll...
including "area51" (4.2.1,i386) roll...
including "java" (4.2.1,i386) roll...
including "condor" (4.2.1,i386) roll...
including "web-server" (4.2.1,i386) roll...
including "base" (4.2.1,i386) roll...
including "grid" (4.2.1,i386) roll...
including "sge" (4.2.1,i386) roll...
including "hpc" (4.2.1,i386) roll...
including "ganglia" (4.2.1,i386) roll...
```

including "os" (4.2.1,i386) roll...
Including critical RPMS
...

Lo que hace el comando anterior es construir la estructura del directorio /export/rocks/install/rocks-dist, contiene enlaces simbólicos a los paquetes que se van a instalar en los nodos.
Para probar que la nueva versión funciona se procede a reinstalar un nodo, de la siguiente manera:

Se borra mediante SSH el archivo /.rocks-release

ssh compute-0-0 rm –Rf /.rocks-release

Esto hace que la próxima vez que arranque el nodo se active el modo de reinstalación.
Se inicia la reinstalación, para verificar si el archivo kickstart fue copiado exitosamente.

ssh compute-0-0 /boot/kickstart/cluster-kickstart

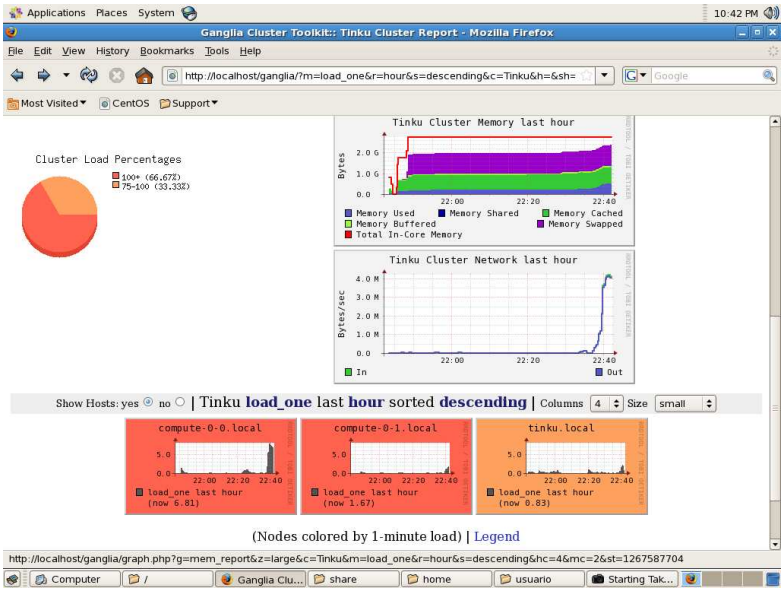
En caso de errores se debe revisar el archivo extend-compute.xml y reconstruir nuevamente la distribución en caso de cambios significativos a este archivo.

4.2.5 Monitoreo con Ganglia. Rocks cuenta con una herramienta muy potente para observar el estado del cluster mediante una interfaz web.
Para acceder a Ganglia, se abre un navegador, en este caso Firefox y en la parte de dirección se digita:

http://localhost/ganglia

El manejo de Ganglia es muy intuitivo, indica la carga actual de cada procesador, el uso de la red, la cantidad de memoria ocupada.
Por cada nodo muestra un reporte detallado del estado y gráficas de la actividad reciente.
Ganglia muestra en su página principal (ver Figura 60) el uso de los nodos de acuerdo a unos colores establecidos que miden el estado de ocupación de cada uno es presentado también en la página principal de Ganglia.

Figura 60. Interfaz Ganglia



4.3 CONEXIONES DE RED DEL CLUSTER TINKU

En el siguiente apartado se aborda las respectivas configuraciones de red, que se realizaron en el Cluster Tinku

4.3.1 Conexión del Cluster Tinku con Internet. Para que el Tinku tuviera acceso al mundo externo e interactuara con este, se hizo necesario realizar una configuración extra, modificando archivos del sistema del frontend. Para que de esta manera el cluster cuente con este importante servicio.

El acceso web del cluster está deshabilitado por razones de seguridad, para permitir el acceso al cluster se modifica el archivo /etc/sysconfig/iptables

Este es el archivo original:

...
http and https is allowed for all nodes on the public subnet
-A INPUT -m state --state NEW -p tcp --dport https --source XXX.XXX.XXX.0/255.255.255.0 -j ACCEPT
-A INPUT -m state --state NEW -p tcp --dport www --source XXX.XXX.XXX.0/255.255.255.0 -j ACCEPT
... other firewall directives ...

Debe quedar de la siguiente manera:

...
http and https is allowed for all nodes on the public subnet
-A INPUT -m state --state NEW -p tcp --dport https -j ACCEPT
-A INPUT -m state --state NEW -p tcp --dport www -j ACCEPT
... other firewall directives ...

Después se reinicia el firewall del sistema
service iptables restart

Como usuario root abre una terminal y se dirige al directorio /etc/sysconfig/network-scripts, con el siguiente comando:

cd /etc/sysconfig/network-scripts

En este directorio, se puede observar que están los archivos de configuración de las interfaces de red, se selecciona el archivo con los ajustes del dispositivo por el cual se va a conectar a Internet.

El archivo debe quedar de la siguiente manera:

DEVICE=eth0
HWADDR=00:0b:6a:5a:6c:f6
IPADDR=192.168.1.10
NETMASK=255.255.255.0
GATEWAY=10.1.1.1
BOOTPROT=static
ONBOOT=yes
MTU=1500

Se agrega la línea con la dirección IP de la puerta de enlace.
Posteriormente se reinicia el servicio de red con el siguiente comando:

service network restart

Ahora se obtiene una dirección IP para Internet mediante un servidor DHCP, con el siguiente comando:

dhclient <interfaz>

Donde <interfaz> es el dispositivo que se va a utilizar para la conexión a Internet.

4.3.2 Escritorio remoto al Cluster Tinku. Consiste en otorgar permisos a un usuario con el fin de trabajar en un equipo por medio de su escritorio gráfico desde otro equipo ubicado en otro lugar.

En el proyecto se consideró importante que el cluster de alto rendimiento Tinku tuviera la capacidad de proporcionar acceso remoto para que los usuarios lo puedan controlar y monitorizar.

De esta forma se tuvo que adicionar software apropiado para realizar este procedimiento, las herramientas escogidas para lograr esto fueron Hamachi la cual se puede descargar de: http://files.hamachi.cc/linux/hamachi-0.9.9.9-20-lnx.tar.gz y UltraVNC win32 Viewer 1.2.8, está disponible en el sitio: //www.uvnc.com/download/1082/.

Se describe el funcionamiento de cada una de estas herramientas con las cuales se logró acceso remoto al Cluster Tinku.

- **Hamachi.** Es una aplicación gratuita con fines no comerciales desarrollada para Windows como para Linux, que sirve para configurar redes privadas virtuales más conocidas como VPN de modo que crea enlaces directos entre computadoras. Es decir, construye una conexión por medio de Internet, simulando una red de área local formada por computadoras que se encuentran físicamente separados.

- **VNC:** Es un programa con licencia GNU, que maneja el modelo cliente-servidor, de manera que este permite controlar remotamente al equipo que se determine como servidor a través del equipo determinado como cliente. VNC tiene la cualidad de permitir el acceso al escritorio remoto sin importar los sistemas operativos que estén instalados en los equipos, tanto como el cliente y el servidor.Se debe clasificar a VNC en dos componentes debido a sus funcionalidades.

    - VNCServer: Es el programa que se debe ejecutar en el equipo servidor. En la mayoría de los sistemas operativos Linux, VNC Server se encuentra preinstalado, se explicará con detalle el proceso de configuración.

    - VNCViewer: Es el programa que se debe ejecutar en el cliente para poder visualizar el entorno grafico de equipo servidor que se quiera controlar. En el proyecto se utilizo como visor UltraVNC win32 Viewer 1.2.8.

    En general, se crea una VPN (Virtual Private Network) entre el cluster Tinku que tiene como sistema operativo Linux y un equipo personal con cualquier sistema operativo que tenga configuradas las herramientas anteriores, con el propósito de que un usuario acceda y maneje el cluster desde cualquier lugar del mundo.

Debido a que la instalación de Hamachi y UltraVNC Viewer es un proceso trivial e irrelevante en el sistema operativo Windows, además se procederá a explicar con detalle la configuración de Hamachi y VNC Server en el cluster Tinku.
De esta forma se debe seguir la siguiente secuencia de procesos para lograr el acceso remoto al cluster Tinku.

4.3.2.1 Configuración VNCServer. En el frontend se requieren abrir terminales que se indican por un número, por defecto es la número 13 como la del usuario root.

Debe iniciar sesión como root.

Después se escribe el siguiente comando:

vncserver :<número>

Por ejemplo:
vncserver :13

La salida de este comando es la siguiente:
You will require a password to access your desktops.
Password: *****
Verify: *****
New 'localhost.localdomain:13 (root)' desktop is localhost.localdomain:13
Creating default startup script /root/.vnc/xstartup
Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/localhost.localdomain:13.log

Se verifica que el servidor tenga los puertos abiertos con el siguiente comando:
netstat –natp

El comando genera lo siguiente:

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 127.0.0.1:2208 0.0.0.0:* LISTEN 2119/hpiod
tcp 0 0 0.0.0.0:5807 0.0.0.0:* LISTEN 3558/Xvnc
tcp 0 0 0.0.0.0:111 0.0.0.0:* LISTEN 1919/portmap
tcp 0 0 0.0.0.0:848 0.0.0.0:* LISTEN 1938/rpc.statd
tcp 0 0 0.0.0.0:5907 0.0.0.0:* LISTEN 3558/Xvnc
tcp 0 0 0.0.0.0:5813 0.0.0.0:* LISTEN 3516/Xvnc
tcp 0 0 0.0.0.0:6007 0.0.0.0:* LISTEN 3558/Xvnc

```
tcp 0 0 0.0.0.0:5913 0.0.0.0:* LISTEN 3516/Xvnc
tcp 0 0 127.0.0.1:25 0.0.0.0:* LISTEN 2161/sendmail: acce
tcp 0 0 0.0.0.0:6013 0.0.0.0:* LISTEN 3516/Xvnc
tcp 0 0 :::80 :::* LISTEN 2189/httpd
tcp 0 0 :::22 :::* LISTEN 2143/sshd
tcp 0 0 :::6007 :::* LISTEN 3558/Xvnc
tcp 0 0 ::1:631 :::* LISTEN 2134/cupsd
tcp 0 0 :::443 :::* LISTEN 2189/httpd
tcp 0 0 :::6013 :::* LISTEN 3516/Xvnc
```

Los puertos que están abiertos son los que inician con 58XX o 59XX, las XX representan las terminales que se habilitaron.

En el archivo /root/.vnc/startup aparece lo siguiente:

Figura 61. Archivo configuración VNC

```
#!/bin/sh

# Uncomment the following two lines for normal desktop:
# unset SESSION_MANAGER
# exec /etc/X11/xinit/xinitrc

[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
xsetroot -solid grey
vncconfig -iconic &
xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
twm &
```

Quitar la marca de comentario

Cambiar esta línea por exec gnome-session &

Guarde los cambios.

Para configurar el número de terminales que se quiere abrir y el usuario, se modifica el archivo /etc/sysconfig/vncservers, quitando la marca de comentario.

…
#VNCSERVERS=”20:myusername”
…

Para cambiar la contraseña se ejecuta:

vncpasswd /root/.vnc/passwd

El programa esperará a que escriba la nueva contraseña y la confirme.

Para eliminar una terminal:

vncserver -kill :<número>

4.3.2.2 Instalación y configuración Hamachi. Se descarga el archivo comprimido de: http://www.uvnc.com/download/1082/, se copia el archivo en /usr/local, es un archivo de tipo tar.gz, para descomprimir y desempaquetar este archivo se utiliza el comando:

tar zxvf hamachi-0.9.9.9-20-lnx.tar.gz

Posterior a esto se crea un nuevo directorio al cual se le cambia el nombre para mayor comodidad:

rename hamachi hamachi-0.9.9.9-20-lnx

Hamachi es una herramienta que se ejecuta por línea de comandos.
Se abre el nuevo directorio y se ejecutan las siguientes instrucciones

make install
cd tuncfg
./tuncfg

Cada vez que se quiera utilizar Hamachi, es necesario iniciar el demonio que se encuentra en el directorio /usr/local/hamachi/tuncfg/tuncfg.

Por cada usuario que se quiera crear una red privada virtual se debe ejecutar el siguiente comando (en este caso root), para crear las llaves de tipo RSA, este comando solo se ejecuta una única vez:

hamachi-init

La salida del comando es la siguiente:

Initializing Hamachi Configuration (/root/.hamachi). Please wait

generating 2048-bit RSA keypair .. ok
making /root/.hamachi directory .. ok
saving /root/.hamachi/client.pub .. ok
saving /root/.hamachi/client.pri .. ok
saving /root/.hamachi/state .. ok

Authentication information has been created. Hamachi can now started with ‘hamachi start’ command and then brought online with ‘hamachi login’.

Para inicializar el demonio se ejecuta:

hamachi start

Para ubicar el demonio en línea y crear una cuenta:

hamachi login

Para unirse a una red existente:

hamachi join <nombre_red> <contraseña>

Para crear una propia red:

hamachi create <nombre_red> <contraseña>

Por defecto el nickname viene como anonymous, si se quiere cambiar el nickname para ser identificado en la red:

hamachi set-nick <nuevo_nick>

Para indicar al cliente que se encuentra conectado:

#hamachi go-online <nombre_red>

Para indicar al cliente que no se encuentra conectado:

hamachi go-offline <nombre_red>

Para ver una lista de todas las redes y sus miembros:
hamachi list

Para eliminar miembros no deseados de la red:

hamachi evict <miembro>

Para desconectarse de la red:

hamachi logout

Para detener el demonio se ejecuta el comando:

#hamachi stop

Para eliminar una red:

delete <nombre_red>

Para mostrar el estado del demonio se ejecuta:

hamachi

5. PRUEBAS Y RESULTADOS

Las pruebas tienen como finalidad evidenciar el desempeño y rendimiento del cluster, las cuales se ejecutaron sobre el cluster completo y gradualmente se fueron disminuyendo el número de nodos, hasta que un único nodo sea el que realice la misma prueba. Todo esto con el fin de determinar la productividad a partir del incremento en los nodos que conforman el cluster. Las pruebas realizadas en un solo nodo, se llevaron a cabo en el frontend, éstas se ejecutaron de forma secuencial.
Las pruebas fueron realizadas con el objetivo de establecer el rendimiento del cluster frente a situaciones en las que se logre demostrar su potencia de procesamiento en paralelo, minimizando así el tiempo en la ejecución de las mismas.

La primera prueba consiste en generar seis imágenes de alta calidad y gran resolución a partir de dos archivos .pov, utilizando herramientas como POV-Ray y Zebra-Core previamente configuradas para realizar esta actividad. El proceso de configuración se describe en el capítulo 3 DESARROLLO DE TINKU.

La segunda prueba consistió en realizar técnicas de minería de datos con la herramienta WEKA-Parallel. La descripción de su configuración para su funcionamiento en paralelo se describió en el capítulo 3 DESARROLLO DE TINKU.
Los repositorios de datos se pueden descargar de http://archive.ics.uci.edu/ml/machine-learning-databases/poker/. Los cuales van hacer procesados por la herramienta WEKA-Parallel.

La última prueba concierne a todo lo relacionado con el paradigma de la programación en paralelo, a través de resolución de un determinado problema mediante una biblioteca de funciones para programar en paralelo.
Al final de cada tipo de prueba, se analizó la eficiencia en el tiempo de entrega de los resultados obtenidos, que fueron procesados por el Cluster Tinku con la totalidad de sus nodos, así como en la variación de los mismos.

En Tabla 2, se muestran las especificaciones técnicas de los equipos utilizados en este proyecto:

Tabla 2. Características equipos

Equipo	Velocidad procesador	Tipo procesador	Memoria RAM	Disco Duro
Nodo 1/ frontend	2.3 GHz	Quad Core	2 GB	250 GB
Nodo 2	2.1 GHz	Dual Core	2 GB	40 GB
Nodo 3	2.7 GHz	i7	2 GB	40 GB

5.1 PRUEBAS DE RENDERIZACION DE IMÁGENES CON ZEBRA-CORE

El objetivo de esta prueba es evidenciar el rendimiento del cluster en cuanto al tiempo de resolución a partir de dos archivos de escena, procedentes de una carpeta de escenas predefinidas de POV-Ray, que contienen archivos de extensión .pov, el cual a su vez contiene las características en formato texto de la escena que deseamos renderizar. Posteriormente este archivo es procesado con tres distintas resoluciones obteniendo por lo tanto un total de seis imágenes.

El proceso consiste en ampliar la resolución de la imagen a procesar, con el objetivo de aumentar la dificultad en el proceso de renderización en el cluster, creando un mayor esfuerzo para éste, a la hora de trabajar, con lo que se podrá analizar mejor su comportamiento, además que al aumentar la resolución implica que el archivo generado incluye más detalles de mejor calidad en la imagen resultante.

La carpeta contenida en POV-Ray donde se encuentran múltiples escenas a procesar es Advanced.
En Tabla 3, se muestra una descripción de los archivos a procesar:

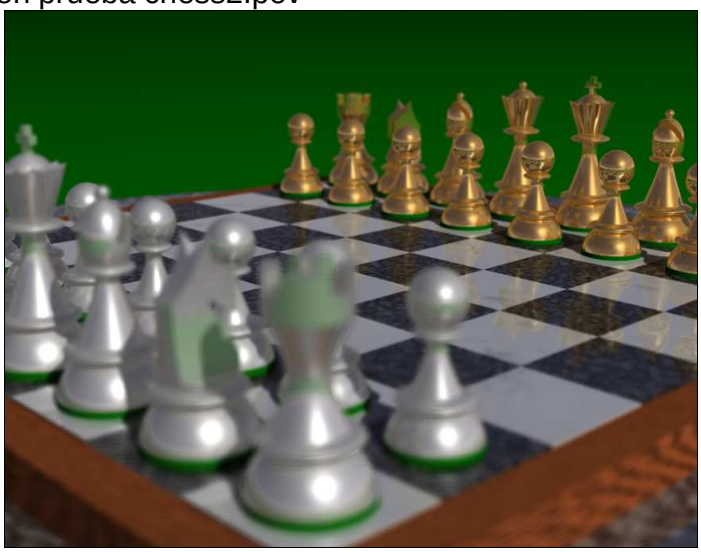
Tabla 3. Escenas a procesar en el cluster

Escenas a procesar en el cluster	
Nombre archivo	Tamaño Kb
chess2.pov	14.1
glasschess.pov	4.4

5.1.1 Renderización primer archivo. El primer archivo a procesar es chess2.pov. Las resoluciones a las que fue sujeto este archivo fueron las siguientes: en

primera instancia fue de 1024x768 pixeles, luego 2816x2112 y por último 3456x2304 pixeles.
La imagen que se obtuvo se muestra en Figura 62.

Figura 62. Imagen prueba chess2.pov



POV-Ray. [en línea]. Disponible en web: http://www.povray.org/download/.

Los resultados que se obtuvieron a medida que se aumentaba el número de nodos para la generación de la imagen se muestran en tabla 4, tabla 5 y tabla 6. La gráfica que muestra el tiempo versus el número de procesadores se muestra en Figura 63.

Tabla 4. Tiempos de ejecución Zebra-Core chess2.pov 1024x768

Procesadores	Tiempo (segundos)	% Eficiencia
1	421,5	-
2	238,21	86,84
3	175,71	70,85

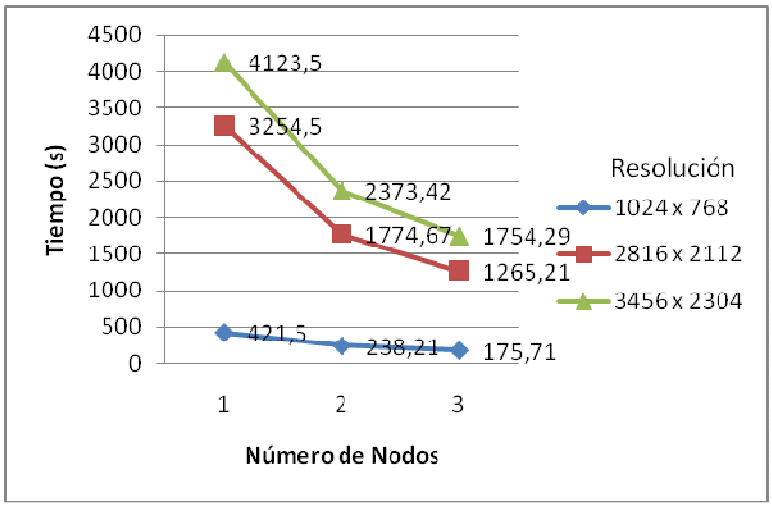
Tabla 5. Tiempos de ejecución Zebra-Core chess2.pov 2816x2112

Nodos	Tiempo (segundos)	% Eficiencia
1	3254,5	-
2	1774,67	86,87
3	1265,21	73,41

Tabla 6. Tiempos de ejecución Zebra-Core chess2.pov 3456x2304

Nodos	Tiempo (segundos)	% Eficiencia
1	4123,5	-
2	2373,42	88,21
3	1754,29	71,21

Figura 63. Resultados renderización Zebra-Core chess2.pov



5.1.2 Renderización segundo archivo. En siguiente archivo a procesar es la escena llamada glasschess.pov, el tamaño de la imagen cambió desde los 800x600 pixeles, después se aumentó a 1024x768 pixeles, hasta 2816x2112 pixeles.

La imagen que se obtuvo se muestra en Figura 64.

Figura 64. Imagen prueba glasschess.pov

POV-Ray. [en línea]. Disponible en web: http://www.povray.org/download/.

Los resultados que se obtuvieron a medida que se aumentaban los procesadores para la generación de la imagen se muestran en tabla 7, tabla 8 y tabla 9. El tiempo de renderización se muestra en Figura 65.

Tabla 7. Tiempos de ejecución en Zebra-Core de glasschess.pov 800x600

Nodos	Tiempo (segundos)	% Eficiencia
1	154	-
2	98,54	78,77
3	66,54	65,06

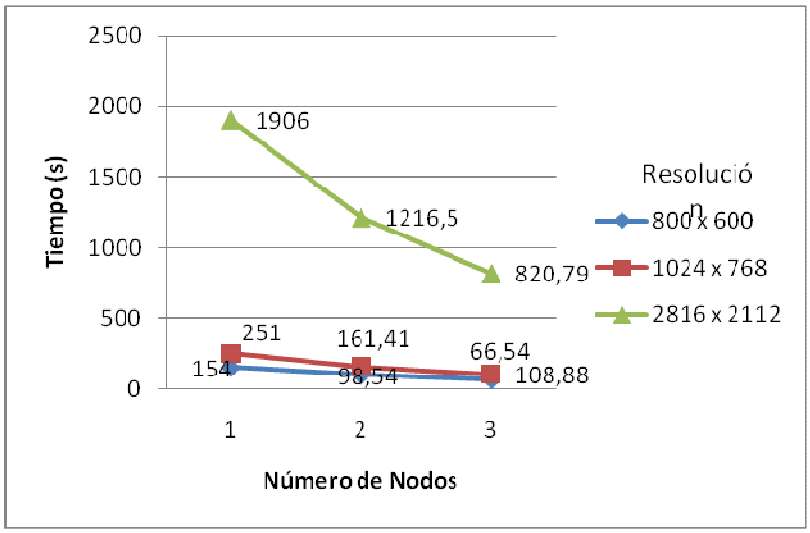
Tabla 8. Tiempos de ejecución en Zebra-Core de glasschess.pov 1024x768

Nodos	Tiempo (segundos)	% Eficiencia
1	251	-
2	161,41	78,68
3	108,88	69,77

Tabla 9. Tiempos de ejecución en Zebra-Core de glasschess.pov 2816x2112

Nodos	Tiempo (segundos)	% Eficiencia
1	1906	-
2	1216,5	78,53
3	820,79	64,92

Figura 65. Resultados renderización Zebra-Core glasschess.pov



5.2 EVALUACIÓN DE LAS PRUEBAS DE RENDERIZACIÓN

Una vez realizado el análisis de los resultados obtenidos a partir de la renderización de las imágenes, se puede concluir lo siguiente:

A medida que se agregan nodos al cluster en el proceso de renderizado, el tiempo de procesamiento de la imagen fotorrealista se reduce de una manera considerable.

A una mayor resolución de la imagen, se hace más evidente la reducción de los tiempos de procesamiento, de tal manera que se puede observar un mejor comportamiento del cluster ante resolución de problemas mayores.

En general, el rendimiento del cluster es óptimo para la resolución de este tipo problemas, partiendo de una adecuada configuración de herramientas, para lograr aprovechar el potencial de Tinku.

5.3 PRUEBAS CON TÉCNICA DE MINERÍA DE DATOS

El objetivo de esta prueba es demostrar la reducción del tiempo de procesamiento de datos independiente de la cantidad de registros a procesar y comprobar de

esta manera el rendimiento del cluster a través de un conjunto de repositorios que van a ser procesados por la herramienta WEKA-Parallel con la técnica de minería de datos tipo clasificación; validación cruzada la cual, permite particionar el conjunto de datos en subconjuntos (folds) y de esta forma poder trabajar en paralelo.

Se ejecutó WEKA-Parallel con los parámetros con los que vienen configurados por defecto, el número 10 folds hace referencia al intervalo de registros que se toma para validar el modelo, siendo 10 el número más utilizado para validación cruzada debido a que produce una estimación razonable de rendimiento de la prueba.

Se utilizaron dos algoritmos de clasificación distintos.
Los algoritmos utilizados en este proceso son:

- **J48.** Este algoritmo está basado en la implementación del algoritmo C4.5 revisión 8, es de tipo árboles de decisión, son los más populares dentro de esta clasificación, trabaja con tipo de datos nominales.
  El algoritmo puede detenerse antes de alcanzar las hojas en cada subárbol, a esto se denomina poda.

- **M5P.** Este algoritmo crea árboles de modelos de regresión. Combina un árbol de decisión convencional con la posibilidad de funciones de regresión lineal en los nodos. Además no se puede actualizar de forma incremental, debe generarse el modelo con los nuevos datos.

Se trabajaron con tres repositorios con diferente número de instancias, el primero fue de 50000 registros, el segundo de 100000 registros y el último de 200000 registros. Todo esto con el fin de aumentar el grado de dificultad de procesamiento de datos en el cluster, y poder observar desde otra perspectiva el comportamiento y el rendimiento de los nodos y el cluster en general, en cuanto al tiempo de entrega de resultados.
En Tabla 10, se muestra una descripción de los conjunto de datos a procesar.

Tabla 10. Conjuntos de datos

Conjunto de datos Clasificación		
Nombre del registro	Numero de registros	Tamaño MB
poker-hand-testing50.arff	50000	1.2
poker-hand-testing100.arff	100000	2.3
poker-hand-testing200.arff	200000	4.7

Cada registro es un ejemplo de una mano de póker en la cual se juegan cinco cartas de una baraja estándar de 52.
Cada carta se describe utilizando dos atributos (pinta y categoría), para un total de 10 atributos predictivos y un atributo CLASS el cual define el tipo de juego.

Información atributos:

S1: Pinta de Carta #1
Tipo: Nominal
Valores:(1-4) que representan Corazones, Espadas, Diamantes y Tréboles.

C1: Categoría de Carta #1
Tipo: Numérico
Valores: (1-13) que representan As, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K.

S2: Pinta de Carta #2
Tipo: Nominal
Valores:(1-4) que representan Corazones, Espadas, Diamantes y Tréboles.

C2: Categoría de Carta #2
Tipo: Numérico
Valores: (1-13) que representan As, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K.

S3: Pinta de Carta #3
Tipo: Nominal
Valores:(1-4) que representan Corazones, Espadas, Diamantes y Tréboles.

C3: Categoría de Carta #3
Tipo: Numérico
Valores: (1-13) que representan As, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K.

S4: Pinta de Carta #4
Tipo: Nominal
Valores:(1-4) que representan Corazones, Espadas, Diamantes y Tréboles.

C4: Categoría de Carta #4
Tipo: Numérico
Valores: (1-13) que representan As, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K.

S5: Pinta de Carta #5
Tipo: Nominal
Valores:(1-4) que representan Corazones, Espadas, Diamantes y Tréboles.

C5: Categoría de Carta #5
Tipo: Numérico

Valores: (1-13) que representan As, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K.

CLASS: Poker Hand
Tipo: Nominal
Valores: (0-9)
0: No es una mano de póker reconocida
1: Un par
2: Dos pares
3: Trío
4: Escalera
5: Pinta
6: Full house
7: Four of a kind
8: Escalera con pinta
9: Escalera Real

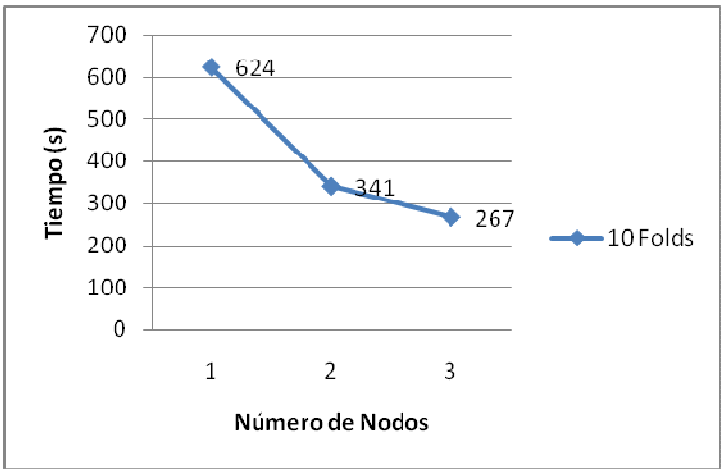
Los atributos utilizados en las pruebas fueron S1 para el algoritmo J48, el cual es nominal, y el atributo C1 que es numérico para M5P.

5.3.1 Procesamiento 50000 registros

Los resultados para J48 se muestran en Figura 66, y del algoritmo M5P se muestran en Figura 67.

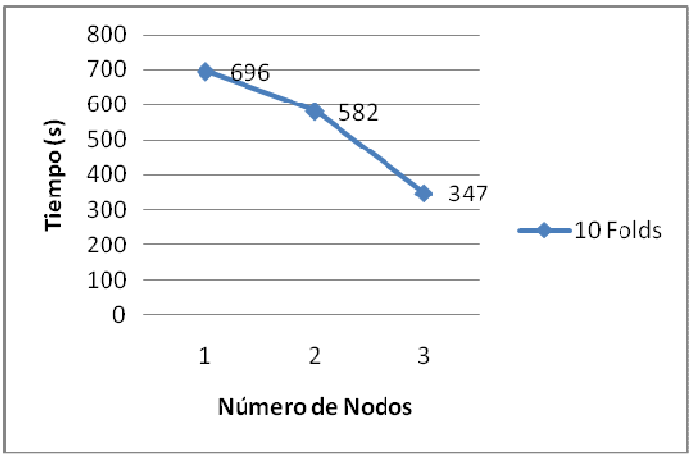
- Algoritmo J48

Figura 66. Tiempos de entrega de resultados J48 con 50000



- Algoritmo M5P

Figura 67. Tiempos de entrega de resultados M5P con 50000

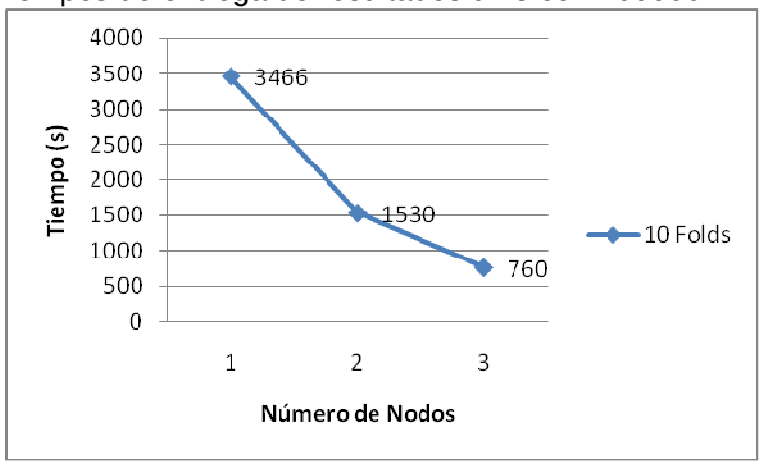


5.3.2 Procesamiento 100000 registros

Los resultados para J48 se muestran en Figura 68, y del algoritmo M5P se muestran en Figura 69.

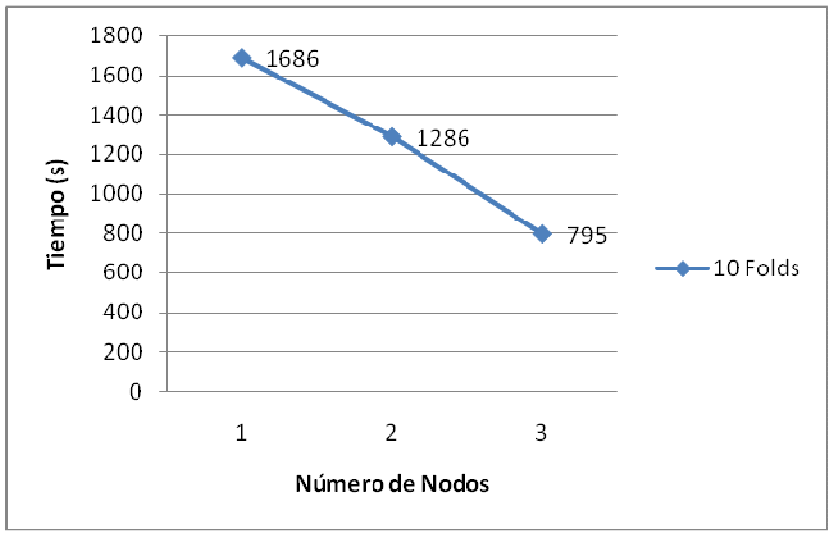
- Algoritmo J.48

Figura 68. Tiempos de entrega de resultados J.48 con 100000



- Algoritmo M5P

Figura 69. Tiempos de entrega de resultados M5P con 100000

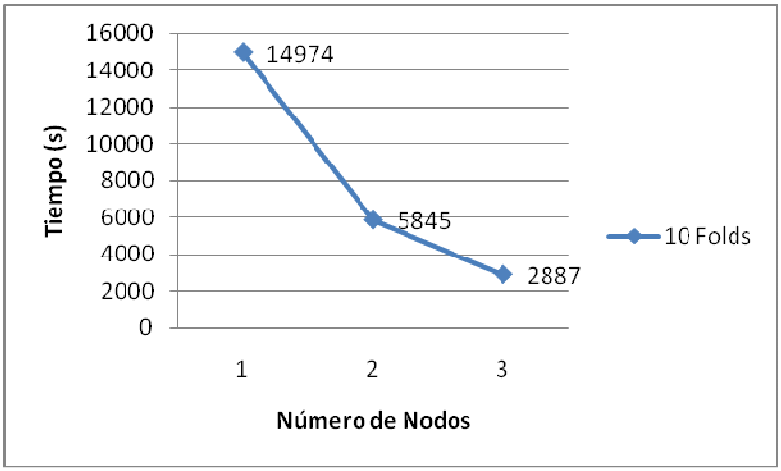


5.3.3 Procesamiento 200000 registros

Los resultados para J48 se muestran en Figura 70, y del algoritmo M5P se muestran en Figura 71.

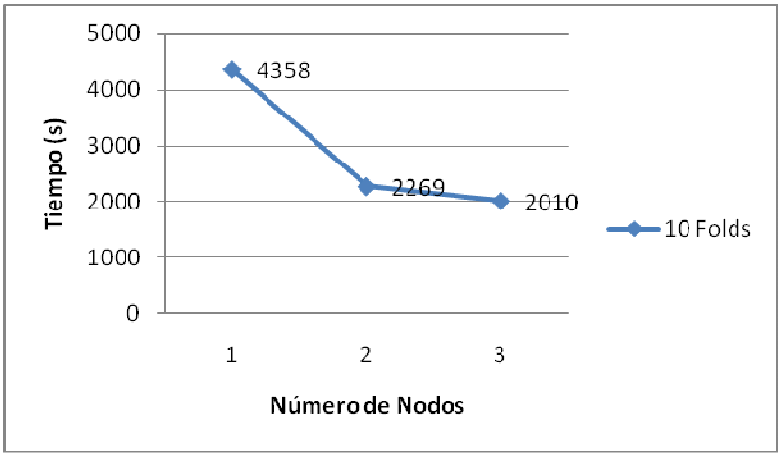
- Algoritmo J.48

Figura 70. Tiempos de entrega de resultados J.48 con 200000



- Algoritmo M5P

Figura 71. Tiempos de entrega de resultados M5P con 200000



5.4 EVALUACIÓN PRUEBAS TÉCNICA DE MINERÍA DE DATOS

Una vez realizado el análisis de los resultados obtenidos a partir del procesamiento de los repositorios con técnica de minería de datos tipo clasificación, se puede concluir lo siguiente:

A medida que se cuente con mayor numero de nodos para la solución de este tipo de tareas, el tiempo de entrega de los resultados disminuye de manera considerable.

Para este tipo de experimentos es conveniente instalar el frontend en un equipo con características avanzadas, especialmente con memoria bastante grande para poder cargar los datos, debido a que éste debe primero debe elaborar el modelo y después distribuye el trabajo a los demás procesadores.

Los resultados generados por los algoritmos J48 y M5P a partir del procesamiento de 50000, 100000, y 200000 registros, se obtienen las mismas reglas y el árbol de decisión no varía, por lo tanto, se puede afirmar que los resultados son confiables.

5.5 PRUEBAS ALGORITMO PARALELO

El paradigma de computación en paralelo consiste en dividir un problema en varias partes para que estas sean resueltas de forma independiente, y luego integrar sus resultados para obtener un mejor rendimiento, y como todos los algoritmos no son paralelizables, en este proyecto se investigaron situaciones a los cuales se podía aplicar dicho modelo, por esta razón se escogió el algoritmo de la aproximación Pi mediante Montecarlo (ver Anexo A).

Por medio de la implementación de este algoritmo, se pretendió determinar de otra forma, el rendimiento de cluster Tinku frente a este tipo de situaciones.
Como plataforma de desarrollo para este ambiente se utilizo MPJ Express, el cual su configuración está descrita en el capítulo 4 DESARROLLO DE TINKU.

El programa consiste en una aproximación del número Pi mediante probabilidades, se realiza un experimento aleatorio un determinado número de veces, entre más veces se repita el programa, el resultado se acerca más al número PI.

El algoritmo funciona de la siguiente manera:

- Inscribe un circulo de radio uno en un cuadrado de lado dos.
- Escoge un punto al azar y determina si el punto cae dentro o fuera del círculo.
- La probabilidad de que el punto esté dentro del círculo equivale a π/4.
- Cuadruplica la frecuencia en que los puntos caen dentro del círculo y se obtiene la aproximación a π.

Este algoritmo ha sido implementado únicamente con fines académicos para experimentar con la programación en paralelo y comprobar la importancia del alto desempeño.

Para compilar la aplicación se ejecuta:

$ javac –cp .:$MPJ_HOME/lib/mpj.jar Pi.java

Para ejecutar la aplicación se ejecuta la siguiente línea:

$ mpjrun.sh –np 2 –dev niodev Pi 1000000

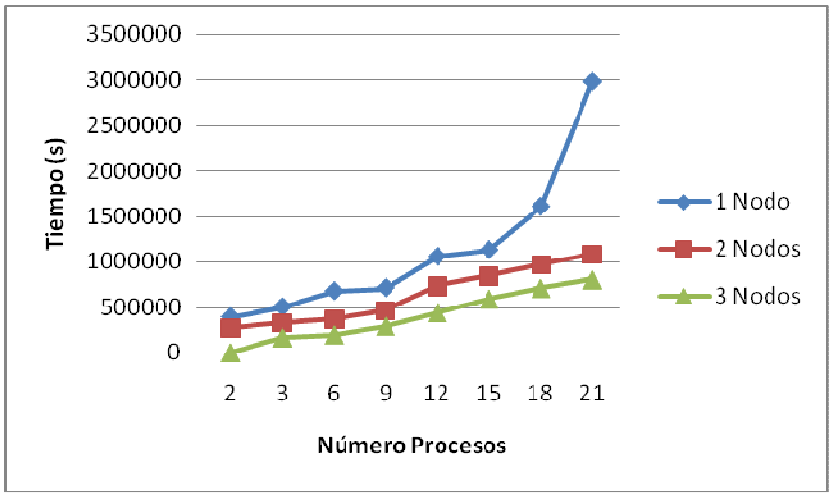
Donde –np 2 varía de acuerdo al número de subprocesos que se quiere se generen en el programa.
En la prueba a partir del número de subprocesos 3 (-np 3), este aumenta en múltiplos de 3, con el fin de que la carga de los subprocesos sea distribuida de manera equitativa entre los nodos.

El número 1000000 hace referencia a la cantidad de intentos que va a realizar el programa para calcular Pi.
Para obtener una mayor veracidad en la entrega de los resultados en cuanto al tiempo, se decidió realizar 5 ejecuciones del algoritmo, por cada nodo de acuerdo al número de procesos.
El tiempo promedio que se demora en entregar el resultado se muestra en microsegundos (ver Tabla 11).
Se presentan los resultados de la ejecución del algoritmo paralelo de la aproximación a Pi (ver Figura 72).

Tabla 11. Tiempos promedio ejecución algoritmo paralelo

	Número de Nodos		
Número Procesos	1	2	3
2	391827	272897	-
3	501833	325323	167326
6	676032	363461	196562
9	713026	464664	293054
12	1058683	732398	443006
15	1130059	843006	590529
18	1609582	975582	710555
21	2982659	1081411	805091

Figura 72. Tiempos promedio de ejecución del algoritmo paralelo



5.6 EVALUACION DE LAS PRUEBAS DEL ALGORITMO EN PARALELO

Como conclusión de ésta prueba se puede afirmar que los mejores tiempos se lograron cuando el número de nodos es igual al número de subprocesos. El tiempo aumenta cada vez que se aumentan los procesos, debido a esta razón este programa es muy útil para ejecutarse en muchos nodos. Si el número de subprocesos es menor que la cantidad de nodos a utilizar, no es conveniente porque quedan nodos sin utilizar.

CONCLUSIONES

El proyecto Tinku es la primera experiencia de configuración de un cluster heterogéneo en la Universidad de Nariño, realizado por los estudiantes de Ingeniería de Sistemas pertenecientes al grupo de investigación GRIAS en la línea de investigación de computación Grid y paralela.

Mediante el presente proyecto de investigación se implementó un prototipo de cluster de alto rendimiento v denominado Tinku así como su respectiva documentación en la Universidad de Nariño, para apoyar actividades e investigaciones complejas que requieran del servicio de esta tecnología que se encuentra al alcance de la academia.

Las pruebas realizadas con el cluster Tinku demostraron que con una configuración adecuada y con las herramientas necesarias funciona correctamente, se logra un mayor rendimiento y eficiencia en el procesamiento de datos.

La Universidad de Nariño, el programa de Ingeniería de Sistemas, el grupo de investigación GRIAS, y el laboratorio de computación Grid y paralela cuenta con un prototipo de cluster denominado Tinku que puede ser implementado formalmente en este laboratorio para el procesamiento de grandes cantidades de datos necesarias en las investigaciones que se realicen.

Personalmente este proyecto permitió a los estudiantes investigadores aplicar los conocimientos adquiridos en la carrera e iniciarse en el proceso investigativo.

RECOMENDACIONES

Reconocer la importancia académica e investigativa que ha representado la implementación del cluster en diferentes universidades de Colombia, se recomienda construir un cluster de alto rendimiento, ya que de esta manera la Universidad de Nariño queda al mismo nivel que otras instituciones educativas.

Utilizar el cluster Tinku en la implementación de un nuevo paradigma de estudio como lo es la computación en paralelo, en el currículo del programa de Ingeniería de Sistemas de la Universidad de Nariño.

Generar en la comunidad de la Universidad de Nariño el sentido de pertenencia e interés hacia el cluster, para así lograr un mejor aprovechamiento y conservación del mismo.

Adquirir hardware de red de altas prestaciones como tarjetas Gigabit Ethernet o switches de mayor capacidad, para alcanzar un óptimo desempeño en el funcionamiento del cluster.

Considerar que un cluster es un recurso informático valioso es muy importante determinar la ubicación definitiva, se recomienda que sea en el Centro de Datos de la Universidad de Nariño, porque cuenta con las instalaciones adecuadas para equipos de computación.

Abrir otras líneas de investigación en el programa de Ingeniería de Sistemas, en caso de una implementación real de un cluster, enfocadas hacia la programación en paralelo, para lograr un mejor aprovechamiento por parte de la comunidad estudiantil del cluster.

REFERENCIAS

ACOSTA, Ricardo et al. Implementación de un CLUSTER de alto rendimiento como herramienta para resolver problemas de cómputo científico.[en línea]. Disponible en web: (http://docente.ucol.mx/~mgarcia/cluster.pdf).

ADITYA, Narayan. High-performance Linux clustering, Part 1: Clustering fundamentals. [en línea]. Disponible en web: (http://www.ibm.com/developerworks/linux/library/l-cluster1/).

AGRAWAL, Rakesh. Data Mining & Knowledge Discovery in Databases (KDD). [en línea]. Disponible en: (http://elvex.ugr.es/etexts/spanish/kdd/KDD.html).

ALONSO, David et al. Radiosidad.[en línea]. Disponible en: (http://blackspiral.org/docs/radiosidad/guion.pdf).

ANDERSSON, Eve. Calculation of Pi Using the Monte Carlo Method. [en línea]. Disponible en web: (http://www.eveandersson.com/pi/monte-carlo-circle).

ANGELFIRE.COM. Generalidades de Java. [en línea]. Disponible en web: (http://www.angelfire.com/electronic/econetcol/generalidadesjava.htm).

BAKKEN, David. Middleware [en línea]. Disponible en web: (http://eecs.wsu.edu/~bakken/middleware.pdf).

BARNEY,Blaise. Introduction to Parallel Computing. [en línea]. Disponible en web: (https://computing.llnl.gov/tutorials/parallel_comp/).

BEDOYA, Diego et al. ZeBrA-Core: Una Herramienta para la Generación de Imágenes Fotorrealistas Usando POV-Ray en Ambientes de Computación Distribuida. [en línea]. Disponible en web: (http://www.saber.ula.ve/bitstream/123456789/16021/1/bedoya.pdf).

BUSTOS, Andrés. Configuración de un cluster de alta disponibilidad y balanceo de carga en Linux para satisfacer gran demanda web y servicios de resolución de nombres.Ecuador. 2007.Escuela Politécnica Nacional.

CARRASCO, Jorge. Una Herramienta para la Síntesis de Imágenes de Alta Resolución empleando Ray Tracing (RT).[en línea]. Disponible en web: (http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/carrasco_r_j/capitulo_2.html).

CASTILLO, José et al. Implementación de un cluster OpenMosix para cómputo científico en el Instituto de Ingeniería. México, 2006. Universidad Nacional Autónoma de México. Facultad de Ingeniería.

CASTRO, Antonio. El trazador de rayos Povray (Tutorial, principios básicos I). [en línea]. Disponible en web: (http://www.ciberdroide.com/wordpress/el-trazador-de-rayos-povray-principios-basicos-i/).

CATALÁN, Miquel. El manual para el clustering con openMosix. [en línea]. Disponible en web: (http://es.tldp.org/Manuales-LuCAS/doc-manual-openMosix-1.0/doc-manual-openMosix-1.0.pdf).

CLUSTER FING. [en línea]. Disponible en web: (http://www.fing.edu.uy/cluster/index.php).

CLUSTERFIE.EPN.EDU.EC. Clusters. [en línea]. Disponible en web: (http://clusterfie.epn.edu.ec/clusters/Definiciones/definiciones.html).

FING.EDU.UY. Computación de Alta Performance Curso 2009 Arquitecturas Paralelas. [en línea]. Disponible en web: (http://www.fing.edu.uy/inco/cursos/hpc/material/clases/Clase2-2009.pdf).

FUENTES, Maikel et al. Metodo Montecarlo. [en línea]. Disponible en web: (http://www.fisica.uh.cu/fisteo/resources/asignaturas/Probabilidades/mc.pdf).

GÁLVEZ, Yadisnel.Computación Paralela: Una solución para el presente y el futuro. [en línea]. Disponible en web: (http://www.uci.cu/files/investigaciones/vol3/Vol.%203,%20No.%203.pdf).

GARCÍA, Oscar et al. Computación Paralela. [en línea]. Disponible en web: (http://www.dynamics.unam.edu/DinamicaNoLineal/Proyectos/Supercomputo/ComputacionParalela.pdf).

GRIFFITHS, Nigel. Ganglia How To. [en línea]. Disponible en:(http://www.ibm.com/developerworks/wikis/display/WikiPtype/ganglia).

HERNÁNDEZ, José. El Proceso de KDD. [en línea]. Disponible en web: (http://elvex.ugr.es/etexts/spanish/kdd/KDD.html).

HERNÁNDEZ, José. Estado Actual de la Aplicación de la Computación Paralela a la Minería de Datos. 2004. Cuba.

ITURRIAGA, Santiago. Proyecto Fenton - Cluster de Computadores de Alto Desempeño con Acceso Remoto. [en línea]. Disponible en web: (http://pgruso2007.googlecode.com/files/PGCCADAR-Informe.pdf).

JOHNSON, James. Bases de Datos Modelos, lenguajes, diseño. México: Oxford University Press, 2000.

JORGE, Javier et al. Introducción al Clustering con MPI. [en línea]. Disponible en web: (http://www.efn.uncor.edu/escuelas/computacion/files/Introducci%C3%B3n%20al%20clustering%20con%20MPI.pdf).

KRAYTRACING.COM. Photon mapping. [en línea]. Disponible en web: http://www.kraytracing.com/wiki/Basics_of_global_illumination/es

MARTÍNEZ, Ignacio. Creación y Validación de un Cluster de Computación Científica Basado en Rocks. [en línea]. Disponible en web: http://e-archivo.uc3m.es/bitstream/10016/5871/1/PFC_Ignacio_Martinez_Fernandez.pdf

MARTÍNEZ, Javier. ¿Qué es la Minería de Datos?. [en línea]. Disponible en web: (http://www.hotinnovacion.es/uploads/EDMANS_DATAMINING_Que_es_la_Mineria_de_Datos.pdf).

MASADELANTE.COM. Definición Sistema Operativo http://www.masadelante.com/faqs/sistema-operativo

MERY, Domingo. Arquitectura de Computadores Multiprocesador y Arquitecturas Alternativas. [en línea]. Disponible en web: (http://dac.escet.urjc.es/docencia/LAAC/LAAC_Tema2-A.pdf).

MICROSOFT TECHNET. Algoritmos de minería de datos (Analysis Services: Minería de Datos). [en línea]. Disponible en web: (http://technet.microsoft.com/es-es/library/ms175595.aspx).

MICROSOFT TECHNET. Validar modelos de minería de datos (Analysis Services - Minería de datos). [en línea]. Disponible en web: (http://technet.microsoft.com/es-es/library/ms174493.aspx).

MORRISON, Richard. Cluster Computing Architectures, Operating Systems, Parallel Processing & Programming Languages. [en línea]. Disponible en web: (www.ace.ual.es/~jaberme/docsppal/cluster/CLSTR_CMPTNG_THEORY.pdf).

ORACLE. Oracle Data Mining, Reglas de Asociación. [en línea]. Disponible en web: (http://www.oracle.com/global/es/database/docs/oracle_data_mining.pdf).

OTERO, Mari. Introducción al UML. [en línea]. Disponible en web: (http://www.vc.ehu.es/jiwotvim/IngenieriaSoftware/Teoria/BloqueII/UML-1.pdf).

PERALES, Víctor. Arquitectura Paralela. [en línea]. Disponible en web: (http://www.monografias.com/trabajos16/arquitectura-paralela/arquitectura-paralela.shtml).

PEROZO, Freddy et al. Cluster Mangosta, Implementación y evaluación. [en línea]. Disponible en web: http://servicio.cid.uc.edu.ve/facyt/v1n1/1-1-2.pdf

POPKIN SOFTWARE AND SYSTEMS. Modelado de Sistemas con UML. [en línea]. Disponible en web: (http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf).

POV-Ray. [en línea]. Disponible en web: http://www.povray.org/download/.

POV-RAY.ORG POV-Ray Documentation Scene Description Language. [en línea]. Disponible en web: http://www.povray.org/documentation/view/3.6.1/224/

POV-RAY.ORG. POV-Ray Documentation What is CSG?. [en línea]. Disponible en web: (http://www.povray.org/documentation/view/3.6.1/28/).

REYES, Vicente. Procesamiento Paralelo en Redes Linux Utilizando MPI. [en línea]. Disponible en web: (http://beta.redes-linux.com/manuales/cluster/mpi-spanish.pdf).

SÁNCHEZ, Alberto et al. Tema 2: Arquitecturas de memoria compartida y multicore. [en línea]. Disponible en web: (http://dac.escet.urjc.es/docencia/LAAC/LAAC_Tema3.pdf).

SIECAR. Cluster Computacional. [en línea]. Disponible en web: (http://siecar.upbbga.edu.co/areas.html).

SOLANO, Franco. Clúster y Grid en el Departamento de Física En red la unión hace la fuerza. [en línea]. Disponible en web: (http://notauniandina.edu.co/html/nota6/agosto2006/pag14-17Clusters.pdf).

TRAN, Gilles. Sci-fi cars and buildings (POV-Ray). [en línea]. Disponible en web: (http://www.oyonale.com/modeles.php?lang=en&page=37).

TREFFTZ, Christian. Procesamiento Paralelo en EAFIT. [en línea]. Disponible en web: (http://www1.eafit.edu.co/drupal/?q=node/549).

TRIBALDOS, Miguel. Elefante, Memoria Distribuida. [en línea]. Disponible en web: (http://personales.alumno.upv.es/~mitriher/elefante/memoria.html).

WORDIQ. Computer rendering - Definition. [en línea]. Disponible en web(http://www.wordiq.com/definition/Computer_rendering).

XIAO, Mei et al. A Parallel Algorithm of Constructing Gene Regulatory Networks. [en línea]. Disponible en web: (http://www.aporc.org/LNOR/11/OSB2009F25.pdf).

ZULUAGA, Jorge. Rocks and Rolls: Sistemas de Computación de Alto Rendimiento de Fácil Despliegue. [en línea]. Disponible en web: (http://www.saber.ula.ve/bitstream/123456789/16036/1/jzuluaga.pdf).

ZULUAGA, Jorge.Introducción a la Programación Paralela - descomposición -. [en línea]. Disponible en web: (http://astronomia.udea.edu.co/sitios/jzuluaga/pages/distos-2009-1-usb.rs/files/distos-2009-1-usbwn3t/distos-presentacion-intro_prog_paralela-descomposicion.pdf).

ANEXOS

Anexo A. Código fuente paralelo Aproximación Pi mediante Montecarlo

```
//javac -cp  .:$MPJ_HOME/lib/mpj.jar Pi.java
//mpjrun.sh -np 2 Pi 100000000


import java.io.*;
import java.text.NumberFormat;
import java.lang.Math;
import mpi.*;
public class Pi {
  public static void main(String[] args) throws Exception {
      System.out.println("Aproximación      Pi      mediante
Montecarlo");
      Pi p = new Pi(args);
  }
  public Pi(String[] args) throws Exception {
    int id_p, n_proc;
    Status estado;

    MPI.Init(args);
    id_p = MPI.COMM_WORLD.Rank(); //identificador del proceso
    n_proc = MPI.COMM_WORLD.Size(); //numero de procesos

    if(id_p == 0){
     System.out.println("Codigo del cliente");
     long tInicio = 0;
     long tFinal = 0;
        double pi;
        double[] _pi = new double[1];
     tInicio = System.nanoTime();
        int intentos = Integer.parseInt(args[3]);
     System.out.println("Intentos: "+intentos);
     int[] _t = new int[1];
     _t[0] = intentos;
     MPI.COMM_WORLD.Send(_t, 0, 1, MPI.INT, 1, 10);
     estado = MPI.COMM_WORLD.Recv(_pi, 0 , 1, MPI.DOUBLE, 1,
20);
     tFinal = System.nanoTime();
```

```
     System.out.println( "Valor de Pi: "+_pi[0] );
     System.out.println( ((tFinal - tInicio) / 1000) );
     System.out.println("Tiempo = " + ((tFinal - tInicio) /
1000) + " micros");
     }
    else if(id_p == 1){
        System.out.println("Codigo del master");
     int[] _t = new int[1];
     estado = MPI.COMM_WORLD.Recv(_t, 0 , 1, MPI.INT, 0, 10);
     int veces = _t[0] / (n_proc - 1);
     System.out.println("Veces: "+veces);
     if(n_proc > 2){
            int[] _veces = new int[1];
            _veces[0] = veces;
            for(int i = 2; i < n_proc; i++){
                MPI.COMM_WORLD.Send(_veces, 0, 1, MPI.INT, i,
(i+1)*10);
            }
        }
     int j = 0;
     int aciertos = 0;
     while (j < veces){
            if (f_acierta())
            {
                aciertos++;
            }
            j++;
        }
        if(n_proc > 2){
            int[] res_parcial = new int[1];
            for(int i = 2; i < n_proc; i++){
                estado = MPI.COMM_WORLD.Recv(res_parcial, 0,
1, MPI.INT, i, (2*i+1)*10);
                aciertos +=res_parcial[0];
            }
        }
        double pi = 4.0 * ((double) aciertos) / ((double)
_t[0]);
     double[] _pi = new double[1];
        _pi[0] = pi;
        MPI.COMM_WORLD.Send(_pi, 0, 1, MPI.DOUBLE, 0, 20);
     }
        else{
            System.out.println("Codigo del trabajador");
            int[] _veces = new int[1];
```

```
            estado  =  MPI.COMM_WORLD.Recv(_veces,  0  ,  1,
MPI.INT, 1, (id_p + 1)*10);
          int j = 0;
          int aciertos = 0;
          while (j < _veces[0])
          {
               if (f_acierta())
               {
                    aciertos++;
               }

               j++;
          }
          int[] res_parcial = new int[1];
          res_parcial[0] = aciertos;
          MPI.COMM_WORLD.Send(res_parcial, 0, 1, MPI.INT, 1,
(2*id_p+1)*10);
        }
     MPI.COMM_WORLD.Barrier();
        MPI.Finalize();
   if(id_p == 0)
     System.out.println("Prueba Pi completa");
  }
  public boolean f_acierta(){
      double x = Math.random() * 2.0 - 1.0;
      double y = Math.random() * 2.0 - 1.0;
      return (x * x + y * y <= 1.0);
  }
}
```

Anexo B. Lista de siglas

ARFF: *Attribute - Relation File Format.*
ASCII: *American Standard Code for Information Interchange* (Código Americano Estándar para el Intercambio de Información).
CENTOS: *Comunity Enterprise Operating System.*
CODINE: *Computing in Distributed Networked Environments.*
CORBA: *Common Object Request Broker Architecture.*
COW: *Cluster Of Workstations.*
CPU: *Central Processing Unit.*
CSMA/CD: *Carrier Sense Multiple Access with Collision Detection* (Acceso Múltiple por Detección de Portadora con Detección de Colisiones)
CSG: *Constructive Solid Geometry* (Geometría Sólida Constructiva).
DHCP: *Dynamic Host Configuration Protocol* (Protocolo de Configuración Dinámica de Host).
DMM: *Distributed Memory Architecture* (Arquitectura de Memoria Distribuida).
DNS: *Domain Name System* (Sistema de Nombres de Dominio).
eKV: *Ethernet Keyboard Video.*
GCC: *GNU Compiler Collection* (Colección de Compiladores GNU).
GPL: *General Public License.*
GRD: *Global Resource Director*.
HA: *High Availability* (Alta Disponibilidad).
HP: *High Performance* (Alto Desempeño).
HPC: *High Performance Computing* (Computación de Alto Desempeño).
HTTP: *Hiper Text Transfer Protocol* (Protocolo de Transferencia de Hipertexto).
INFN: Instituto Nacional de Física Nuclear.
IP: *Internet Protocol* (Protocolo de Internet).
KDD: *Knowledge Discovery Databases* (Descubrimiento del Conocimiento en Bases de Datos).
LAN: *Local Area Network* (Red de Área Local)
LB: *Load Balancing* (Carga Balanceada).
MAC: *Media Access Control* (Control de Acceso al Medio).
MIMD: *Multiple Instruction Multiple Data* (Múltiples Instrucciones Multiples Datos).
MISD: *Multiple Instruction Single Data* (Múltiples Instrucciones Un Solo Dato).
MPI: *Message Passing Interface* (Interfaz de Paso de Mensajes).
MPP: *Massively Parallel Processor* (Procesador Masivamente Paralelo).
NIC: *Network Interface Card* (Tarjeta de Interfaz de Red).
NOW: *Network Of Workstations.*
NTP: *Network Time Protocol.*
NUMA: *Non Uniform Memory Access* (Acceso No Uniforme a Memoria).
OSI: *Open System Interconnection* (Interconexión de Sistemas Abiertos)
P2P: *Peer-to-Peer.*
PC: *Personal Computer* (Computador Personal).

POV-Ray: *Persistence Of Vision Raytracer.*
PVM: *Parallel Virtual Machine.*
RENATA: *Red Nacional Académica de Tecnologías Avanzadas.*
RHEL: *Red Hat Enterprise Linux.*
RRDTOOL: *Round Robin Database TOOL.*
RSA: *Rivest Shamir Adelman.*
SGE: *Sun Grid Engine.*
SIMD: *Single Instruction Multiple Data* (Una Instrucción Múltiple Flujo de Datos).
SISD: *Single Instruction Single Data* (Una Instrucción Un Flujo de Datos).
SMP: *Symetric Multi-Processing* (Multiprocesamiento Simétrico).
SO: *Sistema Operativo.*
SSH: *Secure Shell.*
SSI: *System Single Image* (Sistema de Imagen Única).
TCP: *Transmision Control Protocol* (Protocolo de Control de Transmisión).
UMA: *Uniform Memory Access* (Acceso Uniforme a Memoria).
UML: *Unified Modeling Language* (Lenguaje Unificado de Modelado)
UPS: *Uninterrumpible Power Supply.*
URL: *Uniform Resource Locator* (Localizador Uniforme de Recursos).
VNC: *Virtual Network Computing.*
VPN: *Virtual Private Network* (Red Privada Virtual).
WEKA: *Waikato Environment for Knowledge Analysis.*
XDR: *eXternal Data Representation.*
XML: *eXtensible Markup Language* (Lenguaje de Marcas Extensible).
WWW: *World Wide Web.*