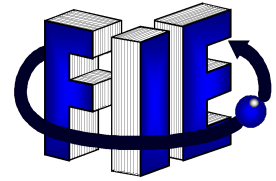




Universidad Michoacana
de San Nicolás de Hidalgo
Facultad de Ingeniería Eléctrica



IMPLEMENTACIÓN DE UN CLÚSTER DE ALTO RENDIMIENTO EN LA E.N.E.S. MORELIA

TESIS

Que para obtener el grado de
INGENIERO EN COMPUTACIÓN

presenta

Miguel Angel Arroyo Villalobos

Dr. Juan José Flores Romero

Director de Tesis

M.T.I. Froylan Hernández Rendón

Co-Director de Tesis

Morelia, Michoacán

septiembre de 2017

Agradecimientos

A Miguel Arroyo Villa por el apoyo moral y económico brindado y por aguantar mis ausencias cuando esperabas una ayuda de mi parte en tu labor diaria.

A Margarita Villalobos Ayala por el apoyo moral brindado, por estar siempre ahí para empujarme a dar lo mejor de mí y por las desmañanadas provocadas todo por no dejar solo a tu hijo en el desayuno antes de salir rumbo a clases.

A Victor Manuel Arroyo Villalobos por tomar un papel que no te correspondía y aguantar las molestias causadas en tu casa.

A Juan Carlos Arroyo Villalobos por el apoyo económico, por pagar caro mis primeros servicios técnicos para hacerme sentir útil y apoyado.

De una manera muy especial a María Guadalupe Sereno Segura por el amor, la paciencia, por soportar mis ausencias durante los proyectos, por la comprensión y el cariño que me dieron fuerzas para seguir adelante y alcanzar la meta.

A mis profesores Dr. Juan José Flores Romero, M.C.C. Froylán Hernández Rendón, Ing. Bertha Georgina Flores Diaz, M.C. José Francisco Rico Andrade, M.I. Moises García Villanueva y M.C. Luis Fernando Guzmán Nateras, por la dedicación y atención prestadas, por aguantar los desplantes de un alumno irreverente y exigente, por enseñarme que en la vida, lo que de verdad vale la pena, cuesta trabajo conseguir.

A mis compañeros Bryan Martínez, Ivan Valdovinos, Yessica Calderon, Gerardo Mauricio, Diego Gómez, Javier Caballero, Ana Rosa Leal, Felipe González, Aarón Ávila, Michel Martínez y Angelica Ayala, que me apoyaron con proyectos y tareas.

A los amigos Dino Gil, Waldir Mendoza, Ricardo Chagolla, por la hermosa experiencia de estar bajo su tutela durante el tiempo que presté mi servicio social, por la formación que ninguna escuela podía brindarme.

Al I.S.C. Raúl Ramírez Baeza, por ser el amigo guía, por la confianza depositada y las dudas aclaradas.

A mi abue Galdina Ayala, por tus oraciones y convertirte en el ángel que intercede por mi ante Dios para que tenga la fuerza de seguir adelante, que Dios te tenga en su santa gloria.

A mis padres Miguel Arroyo y Margarita Villalobos

A mis hermanos Juan Carlos y Victor Manuel

A mi corazón, Lupita Sereno.

¡Meta cumplida!

Contenido

Agradecimientos	III
Dedicatoria	V
Contenido	VII
Lista de Figuras	IX
Lista de Tablas	XI
Resumen	XIII
Abstract	XV
1. Introducción	1
1.1. Antecedentes	1
1.2. Planteamiento del Problema	4
1.3. Justificación	6
1.4. Objetivos de la Tesis	8
1.4.1. Objetivo general	8
1.4.2. Objetivos particulares	8
1.5. Descripción de los capítulos	9
2. Puesta en Marcha	11
2.1. Instalación Física	11
2.1.1. Espacio	12
2.1.2. Piso	13
2.1.3. Montaje	14
2.1.4. Instalación y Cableado	19
2.2. Instalaciones Eléctricas	20
2.3. Instalaciones de aire acondicionado	23
2.4. Conocimientos Generales	25
2.4.1. Definición	25
2.4.2. Propósito de un clúster	26
2.5. Sistemas Operativos hechos para clústers	28

2.5.1.	CentOS.	30
2.5.2.	Rocks Clusters.	31
2.5.3.	Sun Grid Engine (SGE).	32
2.5.4.	Fedora.	33
2.5.5.	OpenMosix.	37
2.5.6.	Solaris.	37
2.5.7.	OpenSolaris.	40
2.5.8.	Windows Server 2008.	42
2.5.9.	Resumen comparativo	45
2.6.	Instalaciones de Red	47
2.6.1.	Topología de Red	47
2.6.2.	Configuración del Front-end	47
2.6.3.	Configuración de los nodos	48
2.7.	Sistema Operativo	49
2.7.1.	Front-end	50
2.7.2.	Nodos de cómputo	61
2.7.3.	Algunos comandos útiles	65
2.8.	Conclusiones de capítulo	66
3.	Pruebas y Resultados	69
3.1.	Introducción	69
3.2.	Marco teórico	70
3.2.1.	Historia	71
3.2.2.	Características de Interfaz de Paso de Mensajes (por sus siglas en inglés <i>Message Passing Interface</i>) (MPI)	72
3.2.3.	Implementaciones MPI	73
3.2.4.	Algunos fundamentos de MPI	75
3.2.5.	Funciones básicas	79
3.3.	Pruebas	79
3.3.1.	Estructura del archivo holamundompi.c	79
3.4.	Presentación de resultados	82
4.	Conclusiones y Trabajos futuros	85
4.1.	Conclusiones Generales	85
4.2.	Trabajos Futuros	86
	Glosario	89
	Bibliografía	94

Lista de Figuras

2.1. Site	12
2.2. Piso	14
2.3. Comparación Rack vs Estante	15
2.4. Montaje en rack (frente)	16
2.5. Riel	18
2.6. Fijación del switch	18
2.7. Montaje en rack (atrás)	19
2.8. Enchufes	20
2.9. UPS	21
2.10. Aire acondicionado	24
2.11. Línea de tiempo de OpenSolaris	41
2.12. Topología de conexión.	47
2.13. Configuración del Sistema Básico de Entrada/Salida (por sus siglas en inglés B asic I nput/ O utput S ystem) (BIOS) en Front-end	51
2.14. Pantalla de inicio	52
2.15. Tipo de Instalación y selección de Rolls.	53
2.16. Selección de Roll para CentOS 6.8.	54
2.17. Información básica	56
2.18. Configuración de red	57
2.19. Etapa final de la configuración de la instalación	58
2.20. Primer inicio de sesión.	59
2.21. Reconfiguración de la Ubicación	60
2.22. Menú para la configuración de Idioma del Teclado.	61
2.23. Selección de Latin American	62
2.24. Configuración del BIOS en nodos de cómputo.	63
2.25. Pantalla de insert-ethers	64
2.26. Inserción de nuevos nodos	65

3.1. Salida del programa holamundompi 84

Lista de Tablas

1.1. Número de alumnos de licenciatura por año en la E scuela N acional de E ducación S uperior (ENES) Unidad Morelia.	4
2.1. Especificaciones técnicas del UPS Eaton [®] 93E.	22
2.2. Versiones de Fedora.	35
2.3. Versiones de Solaris.	38
2.4. Requisitos de hardware (hardware).	45
2.5. Tabla comparativa de sistemas operativos.	46
2.6. Configuración de interfaces del front-end.	48
2.7. Configuración de interfaces de los nodos	49

Resumen

Vivimos en una época en la que los avances tecnológicos se dan con gran velocidad gracias a la misma tecnología. De manera proporcional también crecen las necesidades de disponer de equipos tecnológicamente poderosos capaces de resolver problemas cada vez exponencialmente más complejos y que requieren mucho mayor poder de cómputo, muchos más recursos para ser solucionados en un tiempo óptimo.

Desafortunadamente, el desarrollo de dispositivos de cómputo más poderosos también tiene una gran limitante: la relación tamaño-portabilidad, es decir, los transistores que componen una unidad central de proceso, que es la encargada de realizar las tareas, no pueden ser más pequeños de lo que ya son como para poder colocar más de estos transistores en el mismo espacio, esto aunado a la cantidad de calor que disiparían y los materiales incapaces de soportar dichas temperaturas.

Si fuera posible desarrollar esos materiales y productos, el costo crece aún más rápido. Entonces ¿Cómo hacer para alcanzar el poder de cómputo que se requiere para dar solución a las tareas que la era nos exige? La solución se encuentra en la implementación y programación de clústers de computadoras capaces de compartir recursos, dividir tareas y aportar una solución a los problemas en tiempos record, y sobre todo, hacerlo con un costo considerablemente mucho más económico que el que implicaría construir una supercomputadora capaz de resolver nuestro problema.

En esta tesis se describen de manera resumida todas las consideraciones a tomar en cuenta para la implementación y programación de un clúster de alto rendimiento, desde la instalación física y de sistema hasta la ejecución de pruebas de paralelización.

Abstract

We live in an era in which technological advances occur with great speed thanks to the same technology. In a proportional way also the necessities grow to have technologically powerful equipment capable of solving problems increasingly complex and require much greater computation, many more resources to be solved in an optimal time.

Unfortunately, the development of more powerful computing devices has a great limitation: the size-portability relationship, ie a central processing unit, which is responsible for the tasks, can not be smaller than already is like to be able to place more of these transistors in the same space. This coupled with the amount of heat that dissipate and materials unable to withstand such temperatures.

If it were possible to develop these materials and products, the cost grows even more quick. So how do you achieve the computing power that is required to give solution to the tasks that the era requires us? The solution is found in the implementation and programming of clusters of computers capable of sharing resources, divide tasks and provide a solution to problems in record time, and above all, to do so at a much cheaper cost than would involve building a supercomputer capable of solving our problem.

In this thesis I'll to summarize all the considerations to be taken into account for the implementation and programming of a high-performance, from physical and system installation to performance testing parallelization.

Introducción

1.1. Antecedentes

La ENES Unidad Morelia, asentada en el campus de la Universidad Nacional Autónoma de México (UNAM) ubicado en Antigua Carretera a Pátzcuaro No.8701, Col. Ex Hacienda de San José de la Huerta de la ciudad de Morelia, Michoacán, México, inició sus actividades formalmente el 6 de agosto del año 2012, en respuesta a la necesidad de ampliar la oferta educativa de la máxima casa de estudios en la región centro-occidente con un enfoque multidisciplinario. En este campo se integran las humanidades y las artes, las ciencias sociales, las ciencias físico-matemáticas y las ciencias biológicas, químicas y de la salud.

La ENES Unidad Morelia es sede de ocho entidades:

- 2 Institutos:
 1. Instituto de Investigaciones en Ecosistemas y Sustentabilidad.
 2. Instituto de Radioastronomía y Astrofísica.
- 2 Centros:
 1. Centro de Investigaciones en Geografía Ambiental.
 2. Centro de Ciencias Matemáticas.

- 3 Unidades:

1. Unidad del Instituto de Geofísica.
2. Unidad del Instituto de Investigaciones en Materiales.
3. Unidad de Investigación sobre Representaciones Culturales y Sociales.

En enero de 2012 se inició la construcción de la institución en una extensión de 13.2 hectáreas en las que se desarrolló la infraestructura física con un total de 30,701 m^2 de construcción compuesta por 10 edificios de los cuales:

- 4 de docencia.
- 1 de investigación y postgrado.
- 1 de apoyo a la academia y gobierno.
- 1 de talleres de arte y diseño.
- 1 centro de información, cómputo e idiomas.
- 1 de servicios y administración.
- 1 anfiteatro.

La ENES Unidad Morelia ofrece actualmente 12 licenciaturas vinculadas a la investigación científica del más alto nivel.

1. Ciencias ambientales.
2. Geociencias.
3. Literatura Intercultural.
4. Geohistoria.
5. Historia del Arte.
6. Arte y Diseño.
7. Tecnologías para la Información en Ciencias.
8. Estudios Sociales y Gestión Local.
9. Ciencias de Materiales Sustentables.
10. Administración de Archivos y Gestión Documental.

11. Ecología
12. Música y Tecnología Artística.

Además, se encuentra en proceso de aprobación la licenciatura en Ciencias Agroforestales.

Asimismo, la escuela participa en 4 programas de postgrado de la UNAM:

1. Ciencias Biológicas.
2. Docencia para la Educación Media Superior (MADEMS).
3. Ciencias de la Sustentabilidad.
4. Ciencias de la Tierra.

Los 4 programas ofrecen el nivel de maestría y solo 3 el de doctorado.

La comunidad estudiantil de la ENES Unidad Morelia está conformada por alumnos de los programas de licenciatura y de postgrado, los estudiantes de educación continua y los que provienen del programa de movilidad estudiantil. Hasta el semestre 2017-I se contaba con una población estudiantil total de licenciatura de 882 alumnos, la Tabla 1.1 muestra la distribución del alumnado en las 12 licenciaturas desde el año 2012:

Tabla 1.1: Número de alumnos de licenciatura por año en la ENES Unidad Morelia.

Nombre de la licenciatura	2012	2013	2014	2015	2016
Ciencias ambientales	144	161	214	253	262
Geociencias	17	37	59	66	73
Literatura Intercultural	21	43	69	87	102
Geohistoria		15	25	41	56
Historia del Arte		19	39	81	75
Arte y Diseño		15	25	41	97
Tecnologías para la Información en Ciencias		11	20	26	46
Estudios Sociales y Gestión Local			24	37	61
Ciencias de Materiales Sustentables			17	40	54
Administración de Archivos y Gestión Documental				3	7
Ecología				9	36
Música y Tecnología Artística					13
Total de alumnos de licenciatura	182	313	522	695	882

El modelo educativo de la ENES Unidad Morelia contempla a una planta docente que realiza labores de investigación y está constituida por 2 investigadores titulares, 70 profesores de carrera, 34 técnicos académicos, 88 profesores de asignatura y 63 ayudantes de profesor, de ellos el 40 % pertenecen al **Sistema Nacional de Investigadores (SNI)**, lo que se ha traducido en 39 proyectos de investigación financiados en diversas áreas del conocimiento, y en donde participan los estudiantes de la escuela.

Para las labores de investigación y enseñanza la ENES Unidad Morelia es sede de los siguientes laboratorios:

- Laboratorio Nacional de Análisis y Síntesis Ecológica para la Conservación de los Recursos Genéticos.
- Laboratorio de Ciencias Geoespaciales
- Laboratorio de Ecofisiología y Genómica Funcional
- Laboratorio de Magnetismo Natural
- Laboratorio de Microscopía Electrónica
- Laboratorio de Sistemas de Información Geográfica.

1.2. Planteamiento del Problema

La ENES unidad Morelia requiere de la construcción de un sistema de cómputo que permita dar solución a tareas como el renderizado de imágenes. Debido a la necesidad de dar cumplimiento a estas tareas en periodos de tiempo más cortos, se requiere de un alto poder de procesamiento y de cómputo que no se puede cubrir con equipo convencional. Por tal motivo se ha planteado la construcción de un Clúster de Alto Rendimiento (por sus siglas en inglés **High Performance Cluster**) (HPC) que permita aprovechar el poder de cómputo de varios equipos de manera paralela. Esto acortaría los tiempos en la solución de dichas tareas y mejoraría la calidad y precisión en los trabajos realizados con el equipo.

Para tal motivo la ENES cuenta ya con equipo recientemente adquirido cuyas

características se detallan a continuación.

Dos nodos SUPERMICR[®] modelo SuperServer 6018R-MT mismos que se emplearán como nodos de procesamiento para un clúster de alto rendimiento con las siguientes especificaciones técnicas:

- Controlador de Disco Duro (por sus siglas en inglés **H**ard **D**isk **D**rive) (HDD) 3TeraBytes (TBs), 3.5" a 6.0Gb/s a 7200rpm con 64MegaBytes (MBs) Cache.
- 64GigaBytes (GBs) en Memoria de Acceso Aleatorio (por sus siglas en inglés **R**andom **A**cces **M**emory) (RAM) (4 × 16GB DDR4 RDIM ECC a 2400 MHz)
- Procesador Intel[®] Xeon[™] E5-2630v4 a 2.2GHz con 25MB en Cache
- 20 Nucleos con 40 Hilos
- 2 puertos RJ45 para Red de Área Local (por sus siglas en inglés **L**ocal **A**rea **N**etwork) (LAN) Intel[®] i210 GbE (1 PCI-E 3.0 x16, 1 PCI-E 2.0 x4)
- 1 puerto RJ45 para LAN Dedicada IPMI (Teclado-Monitor-Ratón (por sus siglas en inglés **K**eyboard **V**ideo **M**ouse) (KVM))
- Puertos I/O (1 × VGA, 1 × COM, 2 × USB 3.0 y 2 × USB 2.0)

Aunado al equipo anterior se cuenta también con un equipo de escritorio marca Lenovo con modelo de sistema 10A6A0RQLS mismo que se destinará como el front-end del clúster y el cual dispone de las siguientes características técnicas.:

- HDD de 1TB
- 8GB en RAM
- Procesador Intel[®] Core[™] i7-4790CPU x86_64 a 3.6GHz (8 CPUs)
- Intel[®] Ethernet Connection I217LM a 1Gbs
- Interfaz de Conexión de Red (por sus siglas en inglés **N**etwork **I**nterface **C**onnection) (NIC) FastEthernet Realtek RTL8139/810x a 100Mbs

así como también un switch NETGEAR[®] modelo ProSafe M7100-24X con las siguientes características:

- 24 puertos 10GBase-T (RJ45) con soporte para velocidades Fast Ethernet, Gigabit Ethernet y 10 Gigabit.
- 4 puertos SPF+ para links de fibra a 1G/10G y otras conexiones DAC.
- Enrutamiento estático IPv4 en paquetes de capa 2 y superiores con Lista de Control de Acceso (por sus siglas en inglés **A**ccess **C**ontrol **L**ist)s (ACLs) y Calidad en el Servicio (por sus siglas en inglés **Q**uality **o**f **S**ervice) (QoS) IPv4/IPv6.
- Tablas L2/L3 de clase empresarial con 32K para MAC, 6K para ARP/DNP, 1K para VLANs y 128 rutas estáticas L3.
- Dos fuentes de alimentación redundantes intercambiables sobre la marcha (una viene con el switch, la segunda se solicita por separado y es opcional).
- Dos bandejas de ventilador extraíbles y flujo de aire de refrigeración de adelante hacia atrás para una mejor compatibilidad con los patrones de flujo de aire del centro de datos.

Se dispone, además, de un espacio adecuado para la implementación del clúster así como el sistema de comunicación adecuado para la administración del clúster, por tal motivo el presente proyecto abarca exclusivamente las siguientes tareas:

1. Colocación del equipo en el rack previamente asignado para tal fin.
2. Instalación de un **S**istema **O**perativo (SO), basado en Linux, que le permita al equipo disponible, ofrecer las prestaciones propias de un HPC.
3. Instalación de las librerías de cálculo necesarias.
4. Puesta en marcha y prueba de funcionamiento.

1.3. Justificación

El avance en la tecnología de investigación hace que cada vez surjan aplicaciones que necesiten de recursos de computación más elevados. Una de las formas de enfrentar tal necesidad es unificando equipos para obtener otro de mucha mayor capacidad el cual ofrece las siguientes ventajas:

1. El trabajo en conjunto de los elementos que forman parte del clúster permite dar solución a tareas complejas de una manera mucho más eficiente.
2. Los sistemas son poco propensos a fallas debido a que al distribuirse la carga de trabajo en varias máquinas la eventual falla de una de ellas no afecta a las demás y el sistema se mantiene en funcionamiento como un todo.
3. Capacidad de crecimiento. Se pueden añadir procesadores al sistema incrementando la potencia según las necesidades. Debido a que el crecimiento de un clúster es modular se pueden incluir rápida y fácilmente nuevos recursos sin que afecten los actuales ni perder la inversión previa.
4. No requieren de proyectos costosos y de largo plazo para su implementación por lo que es posible responder rápida y eficazmente a los avances tecnológicos de hoy día.

La tecnología avanza exponencialmente y esto hace que cada día los usuarios de equipos de cómputo, desde estudiantes hasta profesionales, utilicen software que requiere de un alto poder de procesamiento como lo son el software (software) para aplicaciones científicas, de diseño e investigación.

Por otro lado, la evolución y versatilidad que ha alcanzado el SO Linux, al ser abierto y libre, ha contribuido de manera muy importante al desarrollo de muchas tecnologías nuevas, entre ellas la de clúster.

La idea de hacer un clúster, surge como resultado de la convergencia de varias tendencias que incluyen la disponibilidad de equipos de cómputo más económicos, redes de alta velocidad, pero sobre todo aplicaciones que requieren mayor poder de procesamiento y el desarrollo de herramientas de software para cómputo distribuido de alto rendimiento.

Debido al rápido crecimiento que ha tenido en los últimos años la ENES unidad Morelia en cuanto a la demanda de servicios tanto educativos como de investigación en las diversas carreras que la institución oferta, ha sido necesaria la implementación de un HPC que permita dar solución eficiente en tiempo y forma a la demanda

de trabajos mismos que tanto su personal académico como la población estudiantil requieren para llevar a buen fin el proceso de investigación-enseñanza-aprendizaje.

Con tal finalidad la institución cuenta ya con el equipo técnico (hardware) necesario para la implementación del mismo, restando la instalación tanto de hardware como de software y librerías necesarias para la configuración, puesta en marcha y pruebas de funcionamiento del mismo.

1.4. Objetivos de la Tesis

1.4.1. Objetivo general

El objetivo principal de este proyecto consiste en la instalación y configuración de un HPC para la solución de problemas complejos mediante programas estocásticos que emplean modelos no determinísticos. Para ello se hará uso de equipo previamente adquirido cuyas especificaciones técnicas se detallaron en la Sección 1.2.

Otro de nuestros objetivos planteados en este proyecto es la implementación de herramientas (scripts) que permitan la instalación, administración y mantenimiento de los sistemas del clúster, de una forma automática y desatendida.

1.4.2. Objetivos particulares

- Instalación física de los nodos y hardware de red necesario.
- Instalación del SO que alojará el clúster.
- Configuración de la red interna del clúster para la comunicación entre los nodos.
- Instalación y configuración del software y librerías necesarias para el funcionamiento del clúster.
- Corridas y pruebas para verificar el funcionamiento correcto del clúster.

1.5. Descripción de los capítulos

Esta tesis está dividida en 4 capítulos, un glosario y bibliografía.

Capítulo 1. Introducción: Presenta un panorama general previo a la implementación del proyecto. En los antecedentes se describe algo del entorno en la institución, como surgió, las demás instituciones que la conforman así como la población que se verá beneficiada con el proyecto. Se hace el planteamiento y la justificación del problema que se pretende resolver con el proyecto que aborda esta tesis así como los objetivos que se pretenden alcanzar y se delimita el trabajo enlistando de manera concisa las actividades que el proyecto comprende.

Capítulo 2. Puesta en Marcha: En este capítulo se describen las consideraciones pertinentes que se deben tomar en la implementación de un clúster; la preparación del entorno y espacios físicos donde se montara el equipo, las necesidades de ventilación y aire acondicionado así como de instalaciones eléctricas. También se incluyen características y requerimientos de diversos sistemas operativos diseñados para clústers de alto rendimiento, esto con la finalidad de compararlos, proponer e implementar el que mejor se adapte a las necesidades del proyecto.

Capítulo 3. Pruebas y Resultados: Expone una síntesis básica en relación al estándar MPI en la programación paralela. Si bien el tema de MPI es muy extenso, tanto como para dedicarle libros completos, en este capítulo se aborda apenas lo básico que permita la elaboración de un programa capaz de evaluar el funcionamiento correcto del clúster, mismo que se presenta junto con el análisis de la información obtenida tras las pruebas de funcionamiento.

Capítulo 4. Conclusiones y Trabajos futuros: En él se describen las experiencias y reflexiones adquiridas durante la realización del proyecto.

Glosario: Se describen de manera general los conceptos más relevantes empleados en esta tesis.

Bibliografía: Compuesta principalmente por sitios web y manuales técnicos.

Capítulo 2

Puesta en Marcha

En este capítulo se describen las consideraciones esenciales que se deben tomar en la planificación del sitio donde localizaremos el clúster; la preparación del entorno y espacios físicos donde se montara el equipo, las necesidades de ventilación y aire acondicionado así como de instalaciones eléctricas. También se incluyen características y requerimientos de diversos sistemas operativos diseñados para clústers de alto rendimiento, esto con la finalidad de compararlos, proponer e implementar el que mejor se adapte a las necesidades del proyecto.

2.1. Instalación Física

Definir un lugar apropiado para montar el clúster es un tema medular en la implementación del mismo. A diferencia de una computadora personal, un clúster requiere de mucho más equipo, mismo que requiere más espacio físico y un soporte más firme para recibir el peso total del equipo que también disipa una cantidad de calor mucho mayor por lo que requiere de sistemas de enfriamiento especializados.

La cantidad de espacio requerida para el clúster depende directamente del tipo de clúster que planeemos construir, la cantidad y tipos de nodos que vayamos a implementar, si se trata de nodos montables en rack o nodos towercase.

Preparar el entorno en el que nuestro clúster estará operando es tan importante como los componentes que usemos para construirlo. Debemos poner atención al espacio, piso, aire acondicionado e instalación eléctrica, lo que ayudará a la estabilidad de nuestro clúster y al aprovechamiento integral de las capacidades del mismo.

2.1.1. Espacio

En la construcción de cualquier tipo de clúster, no solo necesitamos encontrar un lugar apropiado, sino también asegurarnos de atender otros detalles importantes en relación con la instalación eléctrica y el aire acondicionado.

Actualmente la institución cuenta ya con un espacio de $20m^2$ y mobiliario apropiado para la instalación física del clúster. En la Figura 2.1 se puede apreciar el espacio adecuado en el cual está dispuesto el rack en un entorno data center idóneo para la concentración de grandes cantidades de dispositivos de hardware.



Figura 2.1: Site

Las consideraciones más importantes que hay que tener en mente en relación al espacio son:

- Asegurar que haya espacio suficiente para desplazarnos alrededor del clúster

para cuando se requiera dar servicio al mismo o en la instalación del cableado. Esto nos permitirá trabajar de manera segura sin afectar el sistema.

- Un espacio muy reducido entre los equipos impedirá una ventilación adecuada, lo cual puede desencadenar un sobrecalentamiento de los equipos, reduciendo considerablemente el tiempo de vida de los mismos.
- Asegurarnos de tener un cableado adecuado en el cuarto, que los cables no estén demasiado estirados, trozados o expuestos a filos que puedan trozarlos.
- De ser posible, instalar los nodos y equipo de red en racks o estantes. Esto nos permitirá tener fácil acceso a componentes y lugares para extender el cableado de manera ordenada. Si instalamos en racks debemos asegurarnos de usar los estabilizadores del mismo, evitando así que eventualmente puedan llegar a caer encima al momento de jalar un nodo para algún servicio que se requiera.

2.1.2. **Piso**

La construcción de la base o piso es muy importante desde el punto de vista de seguridad y usabilidad.

Las computadoras personales pueden no ser muy pesadas individualmente, 10 equipos pueden pesar aproximadamente 227 kgs. pero 100 nodos pueden llegar a pesar 2 toneladas de ahí la importancia de consultar con el administrador del edificio o el arquitecto y asegurarnos que el piso pueda soportar esas cargas. Esto es especialmente importante si concentramos todo ese peso en una area menor como es el caso de un montaje en rack.

Si alojamos un gran número de nodos, debemos considerar la instalación de un piso elevado (Fig. 2.2(a)) como lo es en el caso de un data center. Esto permitirá hacer el tendido de cableado por debajo del piso (Fig. 2.2(b)), reduciendo así el riesgo de tropiezos y mejora la apariencia de nuestra instalación; además permite un mejor flujo de aire desde el sistema de enfriamiento hasta los nodos lo que ayudará a que estos últimos se mantengan fríos y prolonguemos así el tiempo de vida del hardware.

(a) *Piso Elevado*(b) *Cableado bajo piso*

Figura 2.2: Piso

2.1.3. Montaje

La colocación de los equipos que componen el clúster se puede realizar de dos formas, una es en un rack especialmente diseñado para tal fin y la segunda es en estantería general. Para decidir cual de las formas de apilado se empleará es necesario identificar las características físicas de los equipos que conformarán el clúster, ya sea que lo tengamos ya en existencia o se vaya a adquirir equipo nuevo.

Existen diferencias entre estas formas de apilado, estas diferencias impactan tanto en el presupuesto para la construcción del clúster como en la planificación de los espacios y distribución de los equipos. A continuación se detallan más las ventajas y desventajas tanto del uso de racks como de estantes.

Racks vs. Estante

La ventaja de usar un bastidor vertical o rack es la facilidad con que podemos llegar a un sistema si necesitamos repararlo. Si apilamos nodos montados sobre una mesa, tendremos que alterar (y probablemente apagar) todo el clúster para quitar el nodo que necesita nuestra atención, en lugar de simplemente deslizarlo fuera del rack sobre un conjunto de rieles.

Las cajas de montaje en bastidor se miden en un esquema llamado Unidades de Bastidor (por sus siglas en inglés **Rack Units**) (RU) el cual, cuando se usa con un número este indica el número de unidades de rack de altura. Cada unidad de rack tiene un espacio vertical de 1.75 pulgadas (4.4 cms.) en un bastidor estándar de 19 pulgadas (47.5 cms.) de ancho y una altura de 80 pulgadas (2 metros).

Con un sistema de torre, se necesita un lugar por separado para localizar la pantalla y el teclado del nodo maestro. Para sistemas montados en rack, hay kits de piezas, disponibles para colocar un teclado y una pantalla plana en una ranura de 1U. Esto permite crear un clúster completamente contenido en un rack. La Figura 2.3 nos da una idea de la diferencia de espacio que puede llegar a ocupar un clúster dependiendo de el tipo de montaje que se implemente.

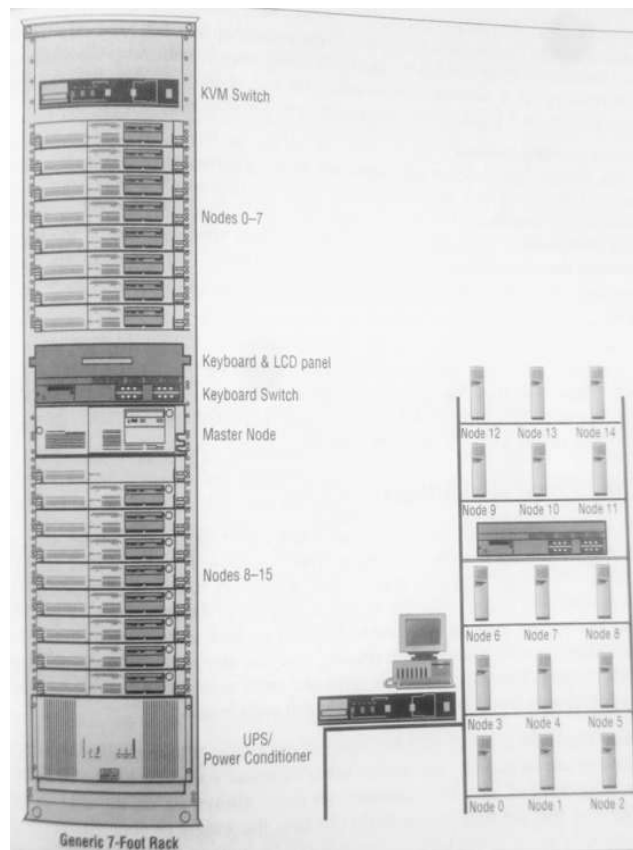


Figura 2.3: Comparación Rack vs Estante

El montaje en rack es lo ideal, ya que optimiza la capacidad de servicio y el espacio de una manera que no es posible hacerlo con nodos towercase.

En la Figura 2.4 se puede observar el montaje de nuestro clúster en el rack correspondiente. Debido a que para el front-end se emplea un equipo tipo torre, y para evitar colocar peso extra e innecesario en los nodos de cómputo, el front-end se dejó sobre piso y sobre de él, en las unidades del rack, se colocaron el switch y los dos nodos de cómputo haciendo uso de las unidades 8, 10 y 12 del rack. Con la finalidad de permitir un flujo de aire apropiado entre cada uno de los componentes del clúster, se dejaron libres las unidades 7, 9 y 11 del rack.



Figura 2.4: Montaje en rack (frente)

Diseño de un sistema basado en bastidor

Usando cajas de montaje en rack de 2U, podemos poner hasta 20 nodos de clúster en un gabinete de 80 pulgadas de alto, con los restantes 13 pulgadas de espacio disponible para el equipo de red que unirá los nodos. El único cuidado importante que debemos tener es cómo tender el cableado, o si debemos dividir el hardware de

red en varios gabinetes también. Esto puede aumentar considerablemente el costo, pero también puede darnos capacidad adicional para una expansión posterior.

Además de los nodos y la conexión en red, hay otros dispositivos que podríamos considerar incluir en nuestros racks de clúster:

Secuenciador o controladores de potencia: Es un dispositivo que enciende la alimentación a diferentes salidas en una secuencia retardada. Son útiles cuando necesitamos encender un gran número de máquinas pues aseguran que todos nuestros sistemas y unidades de disco no se enciendan exactamente al mismo tiempo. Si tuviéramos que encender todos nuestros sistemas a la vez, la demanda de corriente de arranque puede superar la suministrada por la instalación eléctrica, esto se debe a que el consumo de corriente de dispositivos como unidades de disco puede llegar a ser diez veces mayor que el promedio de funcionamiento de un sistema único.

Ventiladores internos para ayudar en el movimiento del aire: Los equipos montados en el rack tendrán ventiladores que lleven el aire desde la parte frontal del gabinete a la parte posterior del mismo para ayudar a evitar que el sistema se sobrecaliente. Además de estos ventiladores, también podemos conectar otros en la parte superior de la mayoría de los gabinetes de 19 pulgadas para forzar la extracción de aire frío de la parte inferior del gabinete.

Barras deslizantes para cajas de ordenador: Las cajas suelen estar diseñadas para montarse permanentemente en bastidores de 19 pulgadas. Esto funciona bien para casos como hubs/routers o switches KVM que rara vez tienen que ser desmontados para su mantenimiento. Para los nodos de cálculo, es útil agregar rieles de deslizamiento a los lados del gabinete y tener rieles coincidentes en el interior del bastidor que permitan meter y sacar los nodos individualmente. Esto nos permitirá abrir un nodo a la vez sin molestar los demás para trabajar en el nodo de nuestro interés (Fig. 2.5).

Como se puede observar en la Figura 2.4, las dimensiones del rack impiden que el switch alcance ambos extremos del rack por lo que se hace necesario el empleo de reductores especiales que permitan un mejor ajuste de los tornillos que fijan el dispositivo al rack (Fig. 2.6).



Figura 2.5: Riel



(a) *Tornillo* (b) *Reductor ros-* (c) *Coloca-*
cado *ción*

Figura 2.6: Fijación del switch

Barras estabilizadoras: Si tenemos rieles deslizantes, también debemos tener barras estabilizadoras en el fondo de nuestro rack. Estas barras se deslizan y proporcionan un mejor soporte para el estante cuando los gabinetes se extraen del rack. Esta es una característica de seguridad que puede evitar que el rack caiga sobre nosotros mientras trabajamos o instalamos sistemas.

2.1.4. **Instalación y Cableado**

Una vez que tengamos todos nuestros sistemas juntos, la único que resta antes de iniciar la instalación es cablear todo (Fig. 2.7).



Figura 2.7: Montaje en rack (atrás)

Mantener el equipo de red cerca del centro del rack nos permitirá el uso de cables más cortos y mucho más manejables.

Si estamos utilizando más de una interfaz de red por nodo, es una buena idea etiquetar los cables o utilizar cables de red con códigos de colores para asegurar que podemos identificar fácilmente segmentos de red diferentes.

También es recomendable crear un archivo que describa la configuración de hardware y software de nuestro clúster. Esto puede ser una herramienta útil al diagnosticar problemas del sistema o al planificar mejoras en nuestro clúster.

Si implementamos nuestro clúster en un entorno data center, sería una buena idea incluir la información de configuración del clúster en algo llamado “libro de ejecución”. Un libro de ejecución incluye toda la documentación que se utiliza como guía de operaciones diarias para un sistema dado. Esta guía de operaciones debe especificar toda

la información que necesitaría el personal de operaciones para mantener el clúster en funcionamiento, como realizar copias de seguridad o comprobar el estado de varios dispositivos de hardware, etc. Un buen libro de ejecución también incluye una descripción completa del hardware y software del sistema para que, si hay problemas, la información sobre el sistema esté fácilmente disponible.

2.2. Instalaciones Eléctricas

Debido a que un clúster grande requiere mucho consumo de energía, se recomienda la contratación de personal altamente capacitado en el área eléctrica para ajustar los requerimientos de energía para todos los componentes de nuestro clúster y otros equipos, como equipos de red, luces, UPS, aire acondicionado, etc.

Todos los nodos, hardware de red y otros componentes de nuestro clúster deben hacer uso de enchufes con conexión a tierra de tres cables como lo muestran las imágenes de las Figuras 2.8(c) y 2.8(b). Además tener fuentes de alimentación con fusibles o interruptores termomagnéticos como los mostrados en la Figura 2.8(a).



(a) *Interruptores termomagnéticos*

(b) *Tierra física*

(c) *Enchufe*

Figura 2.8: Enchufes

La mayoría de las pólizas de seguro no cubrirán daños causados por sobrecar-

ga deliberada o por instalaciones que no cumplan con los requerimientos mínimos de seguridad. También es conveniente invertir en un **Sistema de Alimentación Ininterrumpida** (SAI) (Fig. 2.9) para garantizar que nuestro clúster se pueda apagar de forma adecuada en caso de un fallo de alimentación. Podemos perder datos si la alimentación se apaga sin previo aviso, ya que los nodos de nuestro clúster no podrán escribir sus búfers en los discos de forma segura. Los fallos abruptos de energía también pueden causar que los cabezales de lectura/escritura de los discos duros afecten la superficie del disco causando daños físicos que destruyen el dispositivo debiendo entonces reemplazar la unidad de disco y también perderemos cualquier información que estuviera en el disco en el momento del bloqueo.



Figura 2.9: UPS

La mayoría de los SAI también funcionan como sistemas de acondicionamiento de energía. Una de las causas más comunes de falla del equipo es como resultado de picos de energía u otras fluctuaciones que pueden destruir las placas del sistema, unidades de disco y otros componentes. Las sobretensiones también dañan los componentes de modo que no fallan completamente, pero presentan fallas intermitentes difíciles de

localizar.

Afortunadamente, para fines prácticos no es un tema del cual debemos preocuparnos en el presente caso puesto que el site cuenta ya con un SAI protegido por un UPS Eaton[®] modelo 93E de 40 kVA cullas especificaciones técnicas se detallan en la Talba 2.1.

Tabla 2.1: Especificaciones técnicas del UPS Eaton[®] 93E.

Entrada	
Voltaje	208/220 V_{ac}
Rango de Frecuencia	50/60 Hz
Corriente	128/121~185/181 Amps. de A.C. para el rectificador 111/105~167/157 Amps. de A.C. para Bypass 165~248 Amps. de D.C. de la Bateria Externa
Voltaje de la Bateria	216 V_{dc}
Carga de la batería	40 A
Salida	
Regulación de voltaje	$\pm 1\%$ (10% al 100% de carga)
Voltaje nominal	208 V_{ac} y 220 V_{ac}
Corriente	111/105~167/157 Amps. de A.C. para carga crítica
Dimensiones	
Alto	136 cms.
Ancho	60 cms.
Fondo	96 cms.
Peso	680 kgs.

2.3. Instalaciones de aire acondicionado

El tema del control de temperatura es importante independientemente del tamaño del clúster. Si vamos a tener un número considerable de nodos, requeriremos de un sistema de aire acondicionado de nivel comercial para mantener fresco el lugar. Junto al agua, el calor es el enemigo número uno de la electrónica. Cuanto más caliente sea el entorno, más corto será el tiempo de vida de los componentes. Desafortunadamente, los efectos del sobrecalentamiento no aparecen de inmediato, los daños surgen inicialmente en los pequeños componentes, de manera casi imperceptible, uno tras otro; después el daño es mayor, afectando a un subsistema tras otro repercutiendo en un costo que excede por mucho el de un sistema de aire acondicionado apropiado.

La mayoría de las salas de ordenadores se mantienen entre 18° C - 21° C. Debemos asegurarnos que la sala en la que colocamos nuestro clúster puede mantener el ambiente en/por debajo de este rango de temperatura. Instalar nuestras máquinas en racks o tener un buen espacio entre sistemas individuales ayudará a asegurar que no haya puntos calientes o “islas de calor” que puedan causar un sobrecalentamiento del equipo.

Para tener una idea muy aproximada sobre el calor que puede generar el clúster se puede aplicar la Ecuación 2.1, usando información sobre el consumo de energía del equipo que planeamos usar:

$$\text{GeneracionDeCalor}_{BTU} = \text{PotenciaTotal}_{Volts} * 3.412 \quad (2.1)$$

Para ello tendríamos que recopilar toda la información de alimentación sobre cada dispositivo ubicado en la sala con nuestro clúster. Esto incluye información sobre el consumo de energía de:

- Cada nodo maestro/esclavo.
- Cada dispositivo de red, router y switch.
- Monitores.

- Luces en la habitación.
- Cualquier otro dispositivo.

Esta información sobre el consumo de energía se puede encontrar cerca del enchufe de alimentación en la mayoría de los dispositivos. Algunos dispositivos no indican explícitamente el consumo máximo de energía del dispositivo, pero es muy fácil de calcular a partir de la información de la etiqueta y haciendo uso de la ecuación 2.2:

$$P = |I| * |V| \quad (2.2)$$

Además deberíamos tener en cuenta que no hemos considerado en lo anterior otros tipos de objetos generadores de calor como la iluminación de la habitación, las paredes de las tuberías de calefacción, los pisos, el calor entrante a través de las paredes de otras habitaciones, la luz del sol entrando a través de cualquier ventana, etc. Como podemos ver, enfriar un clúster es un tema bastante complejo para tratar.

Este es otro aspecto a desestimar en nuestro caso puesto que la habitación que alojará nuestro clúster cuenta ya con una unidad evaporadora de la marca York[®] modelo UE-12 (Fig. 2.10) que cuenta con una potencia eléctrica de 88W y una potencia térmica de 24 Btu/h y capaz de mantener una temperatura ambiente dentro del site según las necesidades del mismo como se muestra en la Figura 2.10(a).

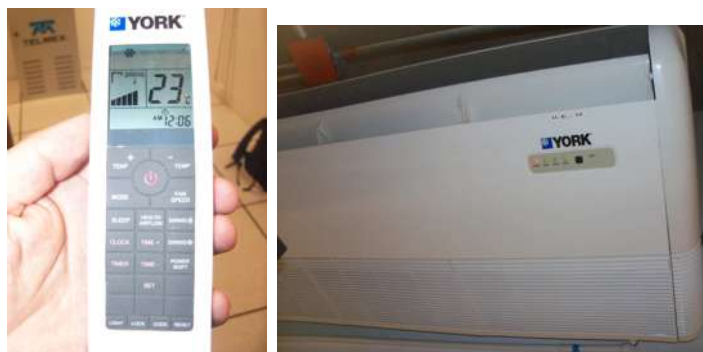
(a) *Control*(b) *Modelo*

Figura 2.10: Aire acondicionado

2.4. Conocimientos Generales

Existe una gran variedad de definiciones de clúster, algunas muy similares y otras un poco más profundas, sin embargo, tratando de estructurar una definición sencilla que explique de manera general el concepto de un clúster tenemos la siguiente:

2.4.1. Definición

Definición 2.4.1 *Clúster: Conjunto o conglomerado de computadoras unidas entre sí, normalmente por una red de alta velocidad; este conjunto se comporta como un solo computador.[Spector, 2000]*

La tecnología de clústers ha evolucionado en apoyo de actividades que van desde aplicaciones de supercómputo y software de misiones críticas, servidores web y comercio electrónico, hasta bases de datos de alto rendimiento, entre otros usos [Rial, 2012]. Los clústers son usualmente empleados para mejorar el rendimiento y/o la disponibilidad por encima de la que es provista por un solo equipo de cómputo.

Tal vez nos preguntamos ¿Por qué la molestia de diseñar y construir un clúster cuando existen en el mercado actual supercomputadoras capaces de resolver problemas de gran complejidad?. La razón es mucho muy sencilla, diseñar y contruir un clúster es considerablemente mucho más económico que adquirir una de esas supercomputadoras.

La construcción de los ordenadores del clúster es más fácil y económica debido a su flexibilidad: pueden tener todos la misma configuración de hardware y SO (clúster homogéneo), diferente rendimiento pero con arquitecturas y sistemas operativos similares (clúster semi-homogéneo), o tener diferente hardware y SO (clúster heterogéneo), este último es la forma más económica de implementar un clúster.

Para que un clúster funcione como tal, no basta sólo con conectar entre sí los ordenadores, sino que es necesario proveer un sistema de manejo del clúster, el cual se

encargue de interactuar con el usuario y los procesos que corren en él para optimizar el funcionamiento.

Los primeros clústers fueron diseñados e implementados por Don Becker y sus colegas de la NASA a raíz de la necesidad de realizar, con un presupuesto muy reducido, análisis complejos del gran conjunto de datos que las misiones espaciales generaban [Spector, 2000].

2.4.2. Propósito de un clúster

Se espera que el clúster presente características de Alto Rendimiento, Alta Disponibilidad, Balanceo de Carga y Escalabilidad.

HPC

Un HPC está diseñado para maximizar y aprovechar la capacidad de cálculo de todos los ordenadores de manera paralela. Los motivos que impulsan la implementación de este tipo de clústers son:

- El tamaño y complejidad de la tarea a resolver.
- El costo y la viabilidad en la construcción de la máquina necesaria para resolver dicha tarea.

HAC

Un Clúster de Alta Disponibilidad (por sus siglas en inglés **H**igh **A**vailability **C**luster) (HAC) se caracteriza por mantener una serie de servicios, generalmente web, compartidos entre sus nodos y que éstos últimos se estén monitoreando constantemente entre sí para garantizar la disponibilidad ininterrumpida de los servicios montados en el clúster.

Balanceo de Carga

Se refiere a la técnica empleada para compartir el trabajo entre varios procesos, ordenadores, discos u otros recursos. El balanceo de carga se mantiene gracias a un algoritmo que divide de manera equitativa el trabajo, reduciendo en gran medida los llamados cuellos de botella.

Un clúster con balanceo de carga se compone por uno o más ordenadores front-end encargados de repartir las peticiones de servicio que recibe el clúster a los ordenadores que forman el back-end.

Sistemas Gestores de Colas

Los sistemas de gestión de colas planifican la ejecución de los procesos y gestionan los recursos que se requieren para dicha ejecución minimizando así costes y maximizando rendimiento. La forma básica en que un sistema de colas trabaja se puede resumir en la siguiente secuencia de pasos:

1. El usuario envía el proceso.
2. El gestor de recursos registra el proceso.
3. Tan pronto como los recursos solicitados se encuentren disponibles, el gestor de colas pone en ejecución el proceso que, según su planificador, tenga mayor prioridad.
4. Se puede consultar el estado de los procesos.
5. Se puede eliminar un proceso.
6. El gestor de colas es configurable.

Definición 2.4.2 *Solver*: Término genérico que se refiere a una pieza de software, posiblemente en la forma de un programa de ordenador independiente, o de una librería, que “resuelve” un problema matemático (teoremas, ecuaciones lineales y no lineales, árboles de expansión, algoritmos de búsqueda...)[Rial, 2012].

Un solver toma como entrada la descripción del problema de una manera específica, y calcula su solución, la cual da como salida. En un solver, se hace hincapié en la creación de un programa o librería que pueda ser fácilmente aplicada a otros problemas similares.

2.5. Sistemas Operativos hechos para clústers

Definición 2.5.1 *Sistema Operativo: Software principal o conjunto de programas de un sistema informático que gestiona los recursos de hardware y provee servicios a los programas de aplicación de software, ejecutándose en modo privilegiado respecto de los restantes (aunque puede ser que parte de él se ejecute en espacio de usuario).*[Wikipedia, 2017b]

Aunque el hardware está disponible para construir estos sistemas distribuidos, hacer que el sistema sea utilizable desde el punto de vista de los operadores sigue siendo un problema. El problema radica en que *los sistemas operativos de computadora no se escalan de ninguna manera utilizable a la arquitectura agrupada de un clúster*. En esta sección se plantea una visión general de varios sistemas operativos diseñados específicamente para HPC.

Al igual que cualquier otro SO, un SO de clúster debe proporcionar una interfaz fácil de usar entre el usuario, las aplicaciones y el hardware del clúster.

En un SO distribuido, es imprescindible tener una Imágen Única del Sistema (por sus siglas en inglés **Single System Image**) (SSI) de tal manera que los usuarios puedan interactuar con el sistema como si se tratara de una sola computadora. También es necesario implementar características de tolerancia a fallos o recuperación de errores para que cuando un nodo falle o se elimine del clúster, el resto de los procesos que se ejecutan en el clúster puedan seguir funcionando. El SO del clúster debe ser escalable y debe propiciar que el sistema esté altamente disponible.

La implementación de un SSI implica hacer que el clúster parezca ser una sola computadora para los usuarios y para la red a la que está conectado. El SSI debe implementar un solo punto de entrada al clúster, un único punto de control, una red virtual, un solo espacio de memoria, una sola gestión de trabajos y una sola interfaz de usuario.[Clarke and Lee,]

En las últimas dos décadas, los investigadores han construido muchos prototipos y sistemas de explotación de clúster de calidad de producción. A continuación se enlistan y resumen brevemente las características y especificaciones mas relevantes de varios de estos sistemas:

- GNU/Linux:
 1. CentOS.
 2. Rocks Clusters.
 3. SGE.
 4. Fedora.
 5. OpenMosix.

- UNIX:
 1. Solaris.
 2. OpenSolaris.

- Windows:
 1. NT.
 2. 2000 Server.
 3. 2003 Server.
 4. 2008 Server.

2.5.1. CentOS.

SO de código abierto, basado en la distribución Red Hat Enterprise Linux, cuyo objetivo es ofrecer al usuario un software “de clase empresarial” gratuito. Se define como robusto, estable, fácil de utilizar. Desde la versión 5, cada lanzamiento recibe soporte durante 10 años por lo que la actual versión 7 recibirá actualizaciones de seguridad hasta el 30 de junio del año 2024.

CentOS usa *Yellow dog Updater Modified*: Herramienta de gestión de paquetes para sistemas Linux basados en RPM (YUM) como paquete de gestión de las actualizaciones. Incluye los dos tipos de escritorios conocidos (Gnome y KDE) que pueden ser instalados por separado o juntos.

CentOS está disponible de forma gratuita para su descarga y en versiones de 64 bits, desde la página web del proyecto dónde también se encuentran notas completas del lanzamiento.

Requerimientos.

- RAM 64MB a 1GB.
- Espacio en Disco 1 a 40GB.
- Arquitecturas x86 y x86_64.
- Escritorios: Gnome 3.8 y KDE 4.10.
- Kernel 3.10.0.
- Capacidad máxima de archivos: 500TB.
- Basado en RedHat.
- Soporte para tarjetas de 40G Ethernet.

Systemd es el reemplazo de *initd* como demonio para iniciar servicios, procesos y recursos del sistema. CentOS 7 incluye PCP (Performance Co-Pilot), un conjunto de frameworks y servicios en tiempo real para supervisar y monitorizar el rendimiento del sistema.

Imágenes disponibles de CentOS 7.

- *CentOS-7.0-1406-x86_64-DVD.iso*: Contiene todos los paquetes que se pueden instalar empleando el instalador.
- *CentOS-7.0-1406-x86_64-Everything.iso*: Imagen de unos 7 GB que contiene todos los paquetes de CentOS 7. Requiere un DVD de doble capa o una memoria USB de 8 GB.
- *CentOS-7.0-1406-x86_64-GnomeLive.iso* y *CentOS-7.0-1406-x86_64-KdeLive.iso*: Imágenes live con los entornos de escritorio Gnome (1.1 GB) y KDE (1.2 GB), respectivamente; usadas para probar, y en su caso instalar, CentOS 7. Estas imágenes no contienen todos los paquetes disponibles de la distribución.
- *CentOS-7.0-1406-x86_64-livecd.iso*: Imagen live adaptada para que se pueda grabar en CD-ROM. Ofrece Gnome como entorno de escritorio y no trae la suite ofimática LibreOffice.
- *CentOS-7.0-1406-x86_64-NetInstall.iso*: Imagen reducida (362 MB), con los recursos mínimos necesarios para realizar una instalación a través de red.

2.5.2. Rocks Clusters.

Rocks es una distribución opensource de Linux para clústers de computadoras de alto rendimiento. Las versiones más modernas de Rocks están basadas en CentOS que es un clon a nivel binario de la distribución **Red Hat Enterprise Linux** (RHEL) compilado por voluntarios a partir del código fuente liberado por Red Hat por lo que los comandos son típicos de esa distribución. Rocks cuenta con un instalador Anaconda modificado, que simplifica la instalación “en masa” en muchos equipos. Rocks incluye herramientas tales como MPI que no forman parte de CentOS pero son los componentes integrales que hacen un grupo de ordenadores en un clúster.

Es una de las distribuciones más empleadas en el ámbito de clústers, por su facilidad de instalación e incorporación de nuevos nodos. Otra de sus grandes facilidades

es que incorpora gran cantidad de paquetes de software (*Rolls*) que permiten la gestión, mantenimiento, monitorización y uso efectivo de la estructura del clúster. Entre estos rolls, aparte de los que permiten y garantizan el buen funcionamiento del clúster, existen varios que se usan en áreas específicas como la medicina u otros campos científicos.

Siendo Rocks una de las distribuciones actuales más moderna y preparada para la gestión de clústers, es también un software siempre ampliable y compatible con cualquier software para Linux que mantenga compatibilidad con CentOS.

Requerimientos.

- 20 GB en HDD.
- RAM 1 GB.

2.5.3. SGE.

SGE (actualmente conocido como Oracle Grid Engine), es software de código abierto desarrollado por Sun Microsystems, cuya función principal es la gestión de un sistema manejador de recursos computacionales en ambientes heterogéneos, de modo que se utilicen dichos recursos de la manera más eficiente posible.

Todas las versiones posteriores a la 6.2u6 son para uso comercial, siendo gratuitas únicamente durante un periodo de prueba de 90 días a partir de su adquisición.

SGE se usa normalmente en aquellos lugares que disponen de un gran número de ordenadores o de HPC. En estas estructuras, se encarga de funciones como la aceptación, programación, envío y gestión de la ejecución remota y distribuida de un gran número de tareas ya sean secuenciales, paralelas o interactivas. Además existe la posibilidad de gestionar y programar la reserva de recursos distribuidos en toda la estructura. Es un sistema altamente configurable, sencillo de administrar y de utilizar. Trabaja con *shellscripts* que definen los requisitos de los trabajos de los usuarios,

pero además puede tratar directamente con binarios e incluso ejecutar trabajos interactivos.

Políticas de planificación.

SGE tiene tres políticas de planificación a la hora de establecer un orden y distribuir los trabajos. Estas políticas pueden combinarse para implementar nuevas políticas de uso de un clúster:

- Política basada en tickets de usuario: Un ticket es una marca o puntuación asignada al usuario, cuántos más tickets tenga un usuario mayor será su prioridad.
- Política de urgencia: Ésta se define por alguno de los siguientes criterios:
 - *Deadline*. Calcula la prioridad del proceso en base al tiempo total de ejecución que le resta.
 - *Tiempo de espera*. Le asigna una prioridad en base al tiempo que lleva el proceso en espera de ser ejecutado.
 - *Demanda de recursos*. La prioridad se define en base a la cantidad de recursos que el proceso demanda para ser ejecutado. Generalmente se ejecutan primero los procesos que demandan menos recursos.
- Política personalizada: permite asignar una prioridad a los trabajos, con unos valores entre -1023 y 1024.

2.5.4. Fedora.

Es una distribución Linux para propósitos generales basada en **Red-Hat Package Manager**: Herramienta de administración de paquetes para GNU/Linux (RPM), que se caracteriza por ser un sistema estable. Cuenta con el respaldo y la promoción de Red Hat. Las principales ediciones de Fedora son Workstation, Server y Cloud, esta última es desarrollada para máquinas virtuales.

Repositorios.

Existen cuatro repositorios mantenidos por el Proyecto Fedora desde donde se puede obtener el software:

1. El repositorio **fedora** almacena todo el software con el que se desarrolla una versión (desde que se separa de la versión en desarrollo). Cuando una versión es liberada, el repositorio fedora se mantiene sin cambios. Es representado por el archivo *fedora.repo*.
2. El repositorio **updates** almacena todas las actualizaciones de software del repositorio fedora, principalmente en la versión estable. Es representado por el archivo *fedora-updates.repo*.
3. El repositorio **updates-testing** almacena todos los paquetes candidatos a incluirse en el repositorio updates. Es representado por el archivo *fedora-updates-testing.repo*.
4. El repositorio **rawhide** corresponde a los paquetes de la versión rawhide, la cual mantiene el modelo de desarrollo rolling-release. Es representado por el archivo *fedora-rawhide.repo*.

Además de los oficiales, existen varios repositorios de terceros. Los más importantes son RPM Fusion y Livna, que proveen tanto software libre como privativo.

Seguridad.

Security-Enhanced Linux (SELinux): Módulo de seguridad para el kernel Linux. Se destaca entre las características de seguridad de Fedora, pues implementa gran variedad de políticas de seguridad, incluyendo Control de Acceso Obligatorio (por sus siglas en inglés **M**andatory **A**ccess **C**ontrol) (MAC).

Lanzamientos.

Las versiones de Fedora se publican cada 6 meses aproximadamente, con un tiempo de soporte que termina un mes después del lanzamiento de la versión subsiguiente, (aprox. 13 meses). La última versión es Fedora 24, publicada el 21 de junio de 2016; en la Tabla 2.2 se muestra un listado de las últimas 4 versiones de Fedora con su fecha de lanzamiento indicando si la versión tiene soporte hasta el momento en que se escribió esta tesis.

Tabla 2.2: Versiones de Fedora.

Versión	Lanzamiento	Nota
21	26-may-2015	sin soporte
22	26-may-2015	con soporte
23	3-nov-2015	con soporte
24	21-jun-2016	versión actual
25	15-nov-2016	en desarrollo

Distribución.

El Proyecto Fedora se distribuye en muchas formas diferentes:

1. Fedora DVD. un DVD con todos los paquetes disponibles.
2. Live CD. Imágenes de CD o DVD que también pueden ser instalados en unidades USB.
3. Imagen de CD o USB. Usado para ser instalado sobre **HiperText Transfer Protocol**: Protocolo de Transferencia de Hipertexto (HTTP), Protocolo de Transferencia de Archivos (por sus siglas en inglés **File Transfer Protocol**) (FTP) o Sistema de Archivos de Red (por sus siglas en inglés **Network File System**) (NFS).

4. Imagen de rescate en CD o USB. Usado si alguna parte del sistema ha fallado y requiere ser reparado. También permite instalaciones desde Internet.

También se distribuyen variantes personalizadas de Fedora, (Fedora spins), construidas de un set de paquetes de software específico y tienen una combinación de software para satisfacer las necesidades de un usuario final determinado. Los Fedora spins son desarrollados por diferentes grupos especiales de Fedora.

DNF es el gestor de paquetes por defecto de Fedora desde la versión 22, es un fork de YUM. Repositorios extra pueden ser agregados al sistema y de esta forma pueden ser instalados los paquetes que no estén disponibles en Fedora. En las primeras 6 versiones había tres repositorios principales:

- **Fedora Core:** Conteníá todos los paquetes básicos que eran requeridos por el SO, así como otros que eran distribuidos con los CD o DVD de la instalación.
- **Fedora Extras:** Repositorio secundario que estaba incluido en Fedora Core 3, era mantenido por la comunidad y no estaba incluido en los discos de instalación, contiene los paquetes más utilizados o demandados.
- **Updates:** En el cual se encuentran las actualizaciones periódicas.

Desde Fedora 7, los repositorios Core y Extras han sido fusionados. Junto con los repositorios fundamentales indicados con anterioridad, algunos de los repositorios más utilizados son *Atrpms*, *Livna*, *FreshRPM*, *Dag*, y *Dries*. Existe la posibilidad de incompatibilidades entre repositorios, especialmente entre *Livna* y *Atrpm*, debido a que emplean diferentes opciones de compilación y por ello las dependencias pueden llegar a ser distintas.

Requerimientos.

- Procesador a 400 MHz.
- RAM 1 GB.
- HDD 10 GB.

2.5.5. OpenMosix.

Es un sistema de clúster para Linux. Esto permite que no se tengan que reprogramar las aplicaciones para que aprovechen el clúster. OpenMosix implementa un algoritmo balanceador que permite repartir de forma óptima la carga.

Actualmente el desarrollo de openMosix está parado. Sigue funcionando bien para el kernel 2.4 pero no completamente así para el kernel 2.6. Por su estabilidad y robustez aún hay personas que lo emplean, y siguen instalándolo. El único problema real es con las máquinas que tienen hardware no soportado por el kernel 2.4.

Características.

- No hay necesidad de programar las aplicaciones para aprovechar el clúster.
- Implementa un algoritmo balanceador que reparte de forma óptima la carga si está bien calibrado el clúster.
- Proyecto cerrado desde el 1 de marzo de 2008.

2.5.6. Solaris.

Es un SO de tipo UNIX desarrollado desde 1992 inicialmente por Sun Microsystems y actualmente propiedad de Oracle Corporation, está certificado oficialmente como versión de UNIX.

Arquitecturas compatibles.

Solaris usa una base de código común para las arquitecturas que soporta: *Scalable Processor Architecture* (SPARC) y x86 (incluyendo AMD64/EM64T).

Solaris tiene una reputación de ser muy adecuado para el Multiprocesamiento Simétrico (por sus siglas en inglés *Simetric MultiProcessing*) (SMP), soportando un gran número de Unidad Central de Procesamiento (por sus siglas en inglés *Central Process Unit*)s (CPUs). También ha incluido soporte para aplicaciones de 64 bits

SPARC desde Solaris 7. Sun dejó de ofrecer estaciones de trabajo basadas en arquitectura SPARC, reemplazándolas por algunos modelos basados en x86 y AMD64.

Entornos de escritorio.

El primer entorno de escritorio para Solaris fue OpenWindows. Fue reemplazado por CDE en la versión Solaris 2.5. El escritorio Java Desktop System, basado en Gnome, se incluye por defecto desde la versión Solaris 10.

Versiones.

La Tabla 2.3 enlista, de manera descendente, las versiones de Solaris que fueron liberadas hasta 2010, Solaris 7 ya no se distribuye pero aún está soportada. Las versiones anteriores no están soportadas.

Tabla 2.3: Versiones de Solaris.

Versión de Solaris	Versión de SunOS	Publicación
Solaris 11	SunOs 5.11	9-Nov-11
Solaris 10	SunOs 5.10	31-ene-05
Solaris 9	SunOS 5.9	28-May-02 (SPARC), 10-ene-03 (x86)
Solaris 8	SunOS 5.8	Feb-00
Solaris 7	SunOS 5.7	Nov-98
Solaris 2.6	SunOS 5.6	Jul-97
Solaris 2.5.1	SunOS 5.5.1	May-96
Solaris 2.5	SunOS 5.5	Nov-95
Solaris 2.4	SunOS 5.4	Nov-94
Solaris 2.3	SunOS 5.3	Nov-93
Solaris 2.2	SunOS 5.2	May-93
Solaris 2.1	SunOS 5.1	Dic-92 (SPARC), May-93 (x86)
Solaris 2.0	SunOS 5.0	Jun-92

Cada versión se basa en una instantánea (*snapshot*) de este tren de desarrollo, tomada cerca del momento de su liberación, que es después mantenida como un proyecto derivado. Desde 2003 una instantánea del tren de desarrollo se hace disponible para su descarga cada mes, permitiendo a cualquiera probar las nuevas características y probar la calidad y estabilidad del sistema a medida que progresa hacia la liberación de la siguiente versión oficial.

Tecnología Preventiva de Auto-Recuperación (por sus siglas en inglés *Predictive Self-Healing*) (PSH). Con esta tecnología se reducen los riesgos y aumenta la disponibilidad del equipo, permite tomar medidas (diagnosticar, aislar, y recuperar los fallos existentes en los dispositivos de E/S o zonas en la memoria) para reducir daños por futuros peligros que puedan causar el caos en los sistemas y como resultado reducir los tiempos de caída. Algunas de las ventajas de las características de PSH son:

- Disponibilidad de servicio y sistema mejorado a través de un diagnóstico y aislamiento de los componentes defectuosos.
- Diagnóstico automático y reinicio de componentes de hardware y software en milésimas de segundo.
- Administración simplificada de los servicios.

ZFS. Nuevo sistema de archivos dinámico de Solaris. Primer sistema de archivos de 128 bits, ofrece una capacidad 16,000 millones de veces superior a la de los sistemas de 32 o 64 bits.

Características de Solaris.

- Adecuado para multiprocesamiento simétrico.
- Escritorios: CDE (v 2.5) y Java Desktop System basado en Gnome (v 10).
- Licencia libre bajo el nombre OpenSolaris hasta 2010.

- RAM 256~512 MB.
- HDD 2 GB.
- Arquitecturas x86 y x86_64 a 120 MHz.

2.5.7. OpenSolaris.

El código fuente de Solaris fue liberado bajo la licencia Licencia Común de Desarrollo y Distribución (por sus siglas en inglés *Common Development and Distribution Licence*) (CDDL) como un proyecto de software libre bajo el nombre OpenSolaris. En agosto de 2010, Oracle decidió interrumpir la publicación y distribución de OpenSolaris, así como su modelo de desarrollo, basado en la disponibilidad de versiones de desarrollo compiladas cada dos semanas y versiones estables cada seis meses. Sin embargo, los términos de su licencia libre no han sido modificados, por lo que el código fuente afectado por ella será publicado cuando Oracle publique nuevas versiones de Solaris.

A raíz del cierre del repositorio de OpenSolaris, un grupo de ex-desarrolladores de OpenSolaris decidió hacer una bifurcación del código y ahora el desarrollo del núcleo del Sistema Operativo (o lo que hubiera sido OpenSolaris) continúa con un nuevo proyecto de la comunidad llamada *Ilumos* que es básicamente el código fuente de OpenSolaris, pero reemplazando los componentes privativos que quedaban por código libre, y basado en el desarrollo continuo en forma de comunidad.

Versiones.

En la Figura 2.11 se puede observar la evolución de las versiones de OpenSolaris. En ella se muestran con azul el periodo de disponibilidad general mientras que la sección marcada con amarillo expone la fase posterior al fin de la versión y finalmente en rojo nos indica el espectro para el periodo de tiempo final para el servicio de soporte de la versión.

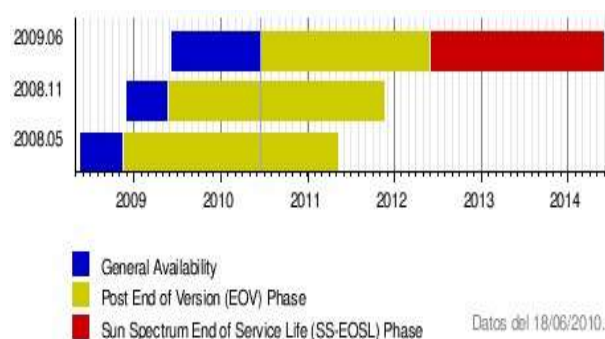


Figura 2.11: Línea de tiempo de OpenSolaris

Distribuciones.

Al igual que sucede en el ámbito de Linux, existen varias distribuciones basadas en OpenSolaris, algunas de ellas respaldadas por ingenieros de Sun y otras totalmente independientes:

Belenix: Desarrollada por ingenieros de India Engineering Centre de Sun. Segunda distribución basada en OpenSolaris. Ahora trata de resituarse para cubrir el lugar del escritorio KDE (Indiana trae Gnome).

Nexenta OS: Proyecto independiente de Sun, basada en Debian/Ubuntu y en el sistema de paquetes *apt*. Primera distribución que combina las librerías *gcc* y las herramientas GNU con el kernel SunOS de OpenSolaris.

OpenSolaris release: Live CD. Orientada a escritorio pero fácilmente adaptable a servidores.

marTux: Live CD/DVD, primera distribución para SPARC.

Polaris: Versión experimental para PowerPC.

SchilliX: Live CD orientado a Gestión.

Características.

- Versión Libre de Solaris

- Los términos de su licencia libre no han sido modificados, por lo que el código fuente afectado por ella será publicado cuando Oracle publique nuevas versiones de Solaris
- Un grupo de ex-desarrolladores de OpenSolaris hizo una bifurcación del código y ahora el desarrollo de lo que hubiera sido OpenSolaris continúa con un nuevo proyecto de la comunidad llamada Illumos

2.5.8. Windows Server 2008.

Sucesor de Windows Server 2003, se basa en el núcleo Windows NT 6.1. Entre las mejoras, se destacan nuevas funcionalidades para el *Active Directory*, nuevas prestaciones de virtualización y administración de sistemas y soporte para más de 256 procesadores. Hay siete ediciones diferentes:

1. *Foundation*.
2. *Standard*.
3. *Enterprise*.
4. *Datacenter*.
5. *Web Server*.
6. *HPC Server*.
7. Para Procesadores *Itanium*.

Características de Windows Server 2008.

- Nuevo proceso de reparación de sistemas NTFS.
- Creación de sesiones de usuario en paralelo.
- Sistema de archivos SMB2: acceso a servidores multimedia 30 a 40 veces más rápido.
- Address Space Load Randomization (ASLR): protección contra malware en la carga de controladores en memoria.
- Windows Hardware Error Architecture (WHEA): protocolo mejorado y estandarizado de reporte de errores.
- Virtualización de Windows Server: mejoras en el rendimiento de la virtualización.

- PowerShell: consola con soporte Interfaz Gráfica de Usuario (por sus siglas en inglés *Graphic User Interface*) (GUI) para administración.

Requerimientos.

- Procesador 1.4 GHz
- RAM 512 MB
- HDD 10 GB

Ediciones.

La mayoría de las ediciones de Windows Server 2008 están disponibles en x86_64 y x86. Será el último SO para servidores disponible en 32 bits. Está disponible en las siguientes ediciones:

1. Standard Edition (x86 y x86_64).
2. Todas las Ediciones (x86_64).
3. Enterprise Edition (x86 y x86_64).
4. Datacenter Edition (x86 y x86_64).
5. R2 Standard Edition (x86_64).
6. R2 Todas las Ediciones (x86_64).
7. R2 Enterprise Edition (x86_64).
8. R2 Datacenter Edition (x86_64).
9. Windows HPC Server 2008.
10. Windows Web Server 2008 R2 (x86_64).
11. Windows Storage Server 2008 (x86 y x86_64).
12. Windows Small Business Server 2008 (x86_64) pequeñas empresas.
13. Windows Essential Business Server 2008 (x86_64) medianas empresas.
14. Windows Server 2008 para sistemas basados en Itanium.
15. R2 Foundation Server.

Actualmente las número 1, 3, 4, 5 (con y sin 1), 7 (con y sin 1), 8 (con y sin 1) y 10 (con y sin 1) están disponibles gratuitamente para estudiantes a través del programa *Microsoft DreamSpark*, al renovarse la licencia.

SPs.

Microsoft lanza ocasionalmente service packs para su familia de sistemas operativos Windows para proteger al usuario frente a nuevas vulnerabilidades detectadas y también añadir nuevas características para que el equipo este en óptimas condiciones.

SP2. Anunciado el 24 de octubre de 2008, este SP contiene los mismos cambios y mejoras que el equivalente Windows Vista SP2, así como la versión final de Hyper-V (1.0) y mejoras que le permiten una reducción del 10% en el uso de energía.

Requisitos de hardware.

La Tabla 2.4 muestra los principales requerimientos de hardware necesarios para la instalación de Windows Server 2008.

Tabla 2.4: Requisitos de hardware.

	Mínimos	Recomendados
Procesador	1GHz(x86) / 1.4GHz(x64)	2GHz +
RAM	512MB (podría limitarse el rendimiento y algunas características)	2GB RAM + Máx. (32 bits): 4GB RAM (<i>Standard Edition</i>), 64GB (<i>Enterprise Edition</i> y <i>Datacenter Edition</i>) Máx. (64 bits): 32GB (<i>Standard Edition</i>), 2TB (<i>Enterprise Edition</i> , <i>Datacenter Edition</i> y para sistemas basados en <i>Itanium</i>)
Tarjeta gráfica	Super VGA (800x600)	Super VGA (800x600) o resolución mayor
Espacio libre en HDD	10GB	40GB o más. Los equipos con más de 16GiB de RAM requerirán más espacio en disco para archivos de paginación y volcado.
Unidades	DVD-ROM	DVD-ROM o mejor

2.5.9. Resumen comparativo

La Tabla 2.5 desgloza de manera conjunta las especificaciones más relevantes de las últimas versiones de los sistemas operativos descritos anteriormente. Se enfatizan primeramente el tipo de licencia (si se trata de software libre o si la licencia tiene un costo) y el periodo o tiempo límite en que la distribución tiene soporte, así como los requerimientos mínimos de RAM y espacio en HDD y finalmente la versión del kernel implementada. Con tal información la tabla nos permite tener una visión más

generalizada respecto a las distribuciones analizadas, esto facilita la identificación del SO que mejor se adapte a nuestras necesidades de clústering.

Tabla 2.5: Tabla comparativa de sistemas operativos.

Distro	Licencia	Soporte	Arq.	RAM	HDD	Kernel	Gestor actualizaciones
CentOS7	opensource	2024	x86 x86_64	64MB - 1GB	1 - 40GB	3.10.0	YUM
Rocks	opensource	—	x86 x86_64	512MB	16GB	2.6	YUM
SGE	opensource	—	—	—	—	2.4	—
Fedora	opensource	13 me- ses	400 MHz	1GB	19GB	—	YUM
OpenMosix	opensource	—	x86 x86_64	—	—	2.4	—
Solaris	comercial	—	x86 x86_64 a 120 MHz	512MB	2GB	—	—
OpenSolaris	opensource	—	x86 x86_64 a 120 MHz	512MB	2GB	—	—
Windows Server 2008	comercial	—	x86_64 a 1.4 GHz	512MB	10GB	—	—

2.6. Instalaciones de Red

2.6.1. Topología de Red

La topología de conexión de los equipos mostrada en la Figura 2.12 es realmente muy sencilla. El front-end se conecta al switch y éste a su vez a los nodos de cómputo todo mediante cable Par Tensado No blindado (por sus siglas en inglés *Unshielded Twisted Pair*) (UTP) cat. 6; la interfaz del front-end que asignamos a esta conexión es la *eth0* mientras que la *eth1* se conecta al switch por el cual el clúster se comunicará con la totalidad de la red interna de la ENES permitiendo el acceso a nuestro clúster de manera remota siempre dentro de la red interna de la institución.

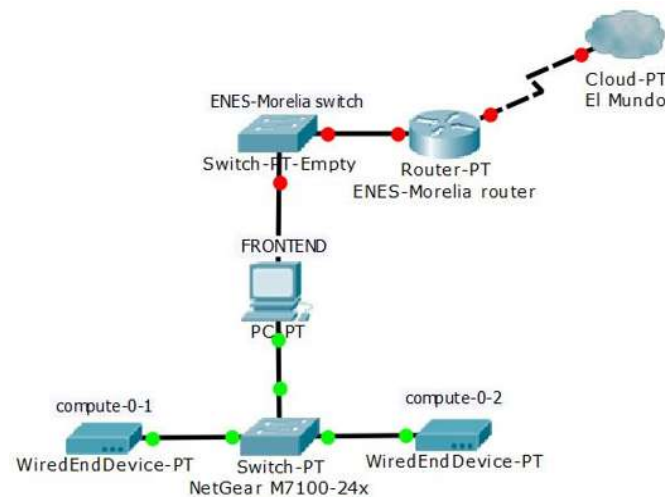


Figura 2.12: Topología de conexión.

2.6.2. Configuración del Front-end

Es realmente importante hacer un buen direccionamiento, acompañado del diagrama que muestre la topología de conexión y guardar en el archivo escrito los datos de tal direccionamiento, dado que el clúster que estamos implementando es aún pequeño en relación al número de nodos que emplea, hacer este direccionamiento es realmente

sencillo y nos facilitará el escalamiento de nuestro clúster en caso de ser necesario. Para la configuración de las interfaces del front-end tenemos los datos mostrados en la Tabla 2.6.

Tabla 2.6: Configuración de interfaces del front-end.

Interfaz <i>eth0</i>	
MAC	D8:CB:8A:42:17:12
IP	10.1.1.1
NETMASK	255.255.0.0
NETWORK	10.1.0.0
BROADCAST	10.1.255.255
Interfaz <i>eth1</i>	
MAC	00:E0:29:50:9D:60
IP	10.100.0.3
NETMASK	255.255.252.0
GATEWAY	10.100.0.1
NETWORK	10.100.0.0
BROADCAST	10.100.3.255
DNS1	132.248.10.2
DNS2	8.8.8.8

2.6.3. Configuración de los nodos

El direccionamiento de los nodos de cómputo lo hace el sistema de manera automática al momento de hacer la instalación de los mismos, iniciando por la última dirección ip usable para el primer nodo instalado, la penúltima para el segundo, la antepenúltima para el tercero y así sucesivamente, de tal manera que el direccionamiento queda con los datos que se muestran en la Tabla 2.7.

Tabla 2.7: Configuración de interfaces de los nodos

Primer nodo (compute-0-1)		
Interfaz <i>eth0</i>	MAC	0C:C4:7A:C3:B0:E2
	IP	10.1.255.253
	NETMASK	255.255.0.0
	GATEWAY	10.1.1.1
	NETWORK	10.1.0.0
	BROADCAST	10.1.255.255
Interfaz <i>eth1</i> (no usada)	MAC	0C:C4:7A:C3:B0:E3
Segundo nodo (compute-0-2)		
Interfaz <i>eth0</i>	MAC	0C:C4:7A:C3:B1:10
	IP	10.1.255.252
	NETMASK	255.255.0.0
	GATEWAY	10.1.1.1
	NETWORK	10.1.0.0
	BROADCAST	10.1.255.255
Interfaz <i>eth1</i> (no usada)	MAC	0C:C4:7A:C3:B1:11

Los datos mostrados en las Tablas 2.6 y 2.7 dependerán tanto de la configuración de la red a la que conectemos nuestro clúster (interfaz *eth1* del front-end) como también de los dispositivos que se estén empleando (direcciones MAC de las interfaces). Estos datos no necesariamente serán configurables de forma manual puesto que, como se muestra en la Sección 2.5, las implicaciones de implementar un SSI imprescindible en sistemas distribuidos impiden que el usuario configure la información de red en cada nodo, lo hace el sistema a partir de los datos insertados durante la instalación en el front-end. Con esto se reducen los errores en la configuración por el factor humano.

2.7. Sistema Operativo

Con la información presentada en la Sección 2.5 y particularmente en la Tabla 2.5 se escogió Rocks Clusters, en su versión 6.2, como la distribución que mejor se adaptó

a las necesidades de nuestro proyecto por el tiempo de soporte, tipo de licencia y requerimientos del sistema.

2.7.1. Front-end

Antes de proceder con la instalación del front-end, es necesario asegurarse de que las conexiones de red de los equipos junto con la configuración del BIOS de cada uno sean las correctas.

Rocks asigna automáticamente la interfaz *eth0* a la red interna del clúster y *eth1* a aquella conectada a la red externa. En caso de que éste sea incapaz de determinar cual es la red externa debido a la falta de conexión a internet, las interfaces pueden ser asignadas de manera incorrecta. Dado que Rocks no permite volverlas a asignar una vez finalizada la instalación, sería necesario reinstalarlo para poder solucionar el problema.[Redondo, 2015]

Configuración del BIOS

Definición 2.7.1 *Entorno de Ejecución de Prearranque (por sus siglas en inglés Preboot eXecution Environment) (PXE): Es un entorno para arrancar e instalar el SO en computadoras a través de una red, de manera independiente de los dispositivos de almacenamiento de datos disponibles (como discos duros) o de los sistemas operativos instalados.[Wikipedia, 2015]*

Previo a iniciar la instalación del SO es recomendable configurar el sistema de arranque del equipo en el BIOS para forzar el arranque desde la unidad de CD-ROM. Para ello requerimos presionar la tecla “F1” al momento de arrancar el sistema; una vez dentro del BIOS accedemos a la pestaña “Startup” y reacomodamos el orden de arranque para evitar que arranque desde el disco duro (Fig. 2.13(a)) (opcional). Otro aspecto que conviene habilitar es el arranque PXE, para ello seleccionamos “Network Setup” de la pestaña “Devices” y ahí verificamos que la sección “On board Ethernet

Controller” esté habilitada y que la sección “Boot Agent” sea PXE (Fig. 2.13(b)).

Debido a que los procesadores de nuestros nodos no son de doble núcleo “real” es recomendable deshabilitar la tecnología “*Hyper-Threading*” que aprovecha el procesamiento multi-hilo simulando que es de doble núcleo. Para deshabilitarla lo podemos hacer en la sección “CPU Setup” de la pestaña “Advanced” (Fig. 2.13(c)).

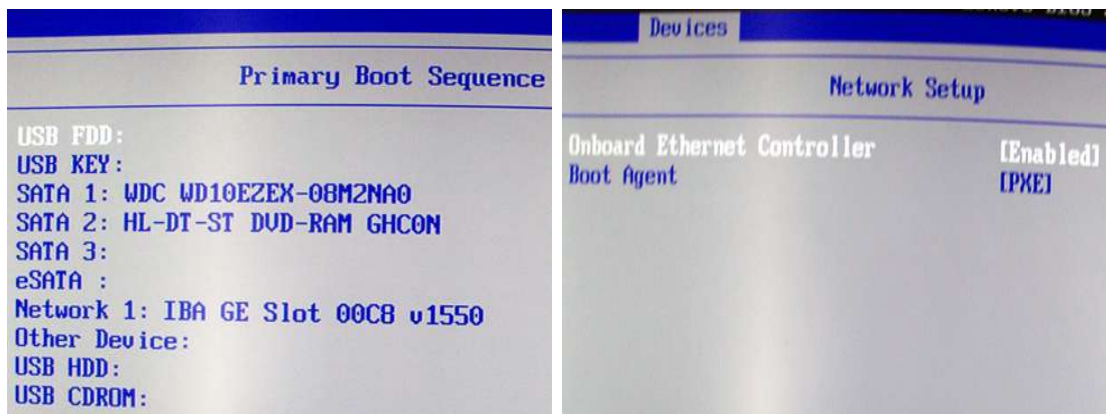
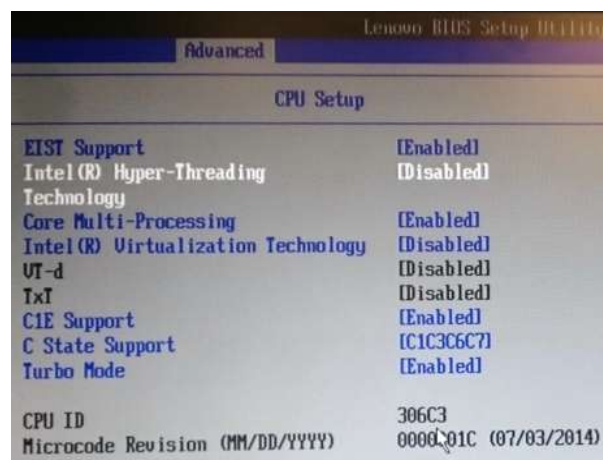
(a) *Secuencia de Arranque*(b) *Arranque PXE*(c) *Hyper-Threading*

Figura 2.13: Configuración del BIOS en Front-end

Instalación

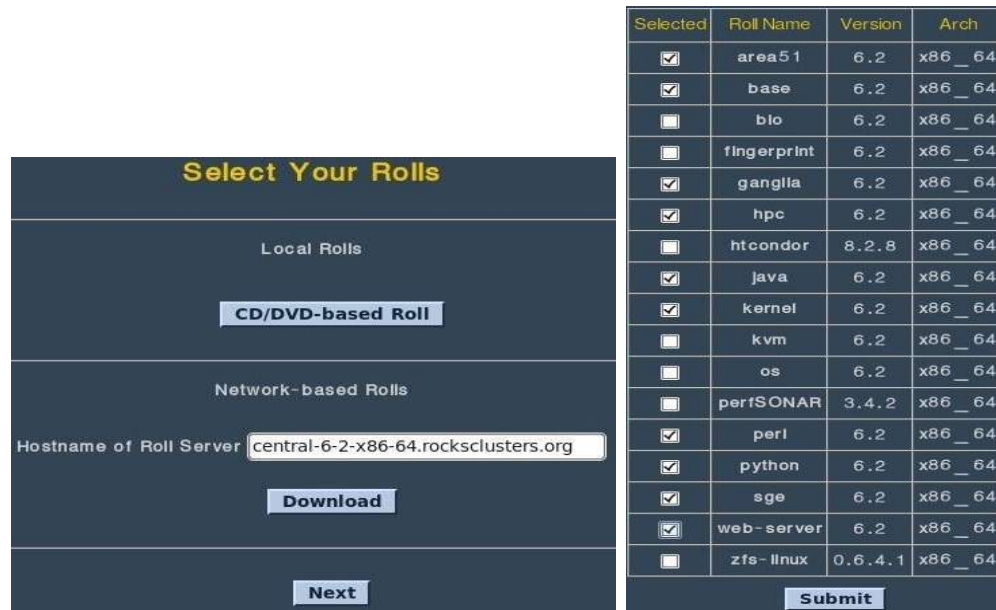
En caso de no haber reorganizado el orden de arranque desde el BIOS también es posible forzar el arranque desde el CD-ROM al momento de arrancar el sistema presionando la tecla “F12” para que se muestre el menú de arranque y seleccionar el arranque desde el CD. Habiendo echo esto y teniendo el disco de instalación en la bandeja, podremos ver la pantalla de arranque (Fig. 2.14), como nos lo indica, tecleamos el comando “build” para iniciar la instalación del front-end.



Figura 2.14: Pantalla de inicio

Después de finalizar el proceso de carga en memoria, Rocks mostrará un formulario en entorno gráfico (Fig. 2.15(a)), mediante el cual podremos configurar el front-end. La primera pantalla del formulario presenta como queremos proceder a realizar la instalación, dado que contamos con el DVD completo, seleccionamos “CD/DVD based rolls”. A continuación seleccionamos de la lista de Rolls aquellos que deseamos instalar en el clúster (are51, base, ganglia, hpc, java, kernel, perl, python, sge, web-server) y presionamos “Submit” (Fig: 2.15(b)). Aunque aparentemente lo anterior nos regrese a la pantalla del tipo de selección podemos notar que, a diferencia del principio, ahora se muestra del lado izquierdo un listado con los Rolls seleccionados (Fig. 2.15(c)).

En este punto no hemos seleccionado ningún roll de SO, para ello haremos uso de



(a) Tipo de instalación

(b) Selección de Rolls

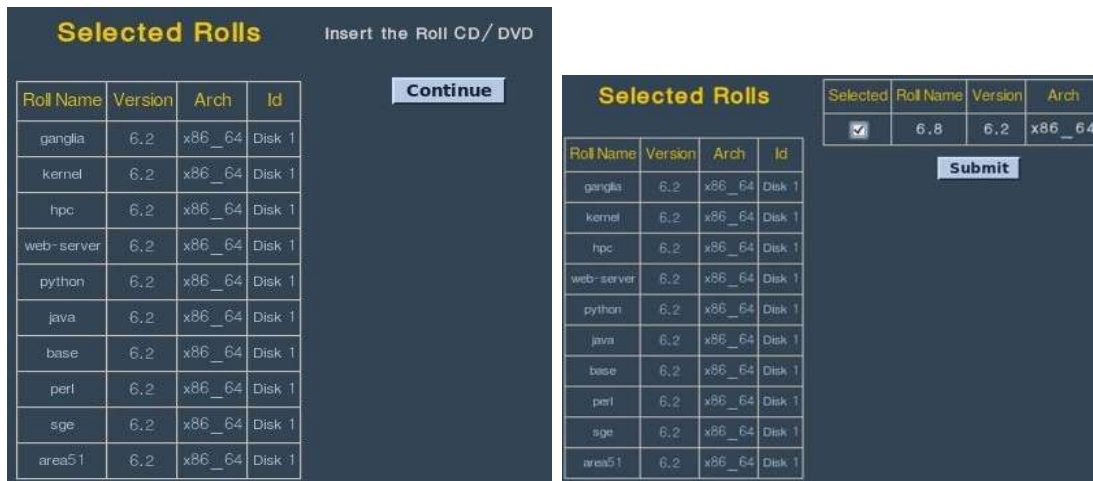


(c) Lista de Rolls

Figura 2.15: Tipo de Instalación y selección de Rolls.

los dos DVDs de CentOS 6.8. Primeramente con el disco 1 hacemos click en el botón “CD/DVD based rolls” que nos muestra la pantalla de la Figura 2.16(a), colocamos

el disco en la bandeja y presionamos “Continue”. Seleccionamos el único roll mostrado y presionamos “Submit” (Fig. 2.16(b)), repetimos para el disco 2 y finalmente presionamos “Next” en la pantalla de la Figura 2.16(c).



(a) Insertar CD

(b) Roll 6.8 d1



(c) Lista de Rolls

Figura 2.16: Selección de Roll para CentOS 6.8.

Los rolls que se instalaron en el clúster de este caso son:

- **ganglia:** Sistema de monitorización del clúster de manera remota mediante el explorador web.
- **hpc:** Proporciona aplicaciones para ejecutar trabajos en paralelo mediante librerías como MPI (OpenMPI, MPICH, MPICH2), Máquina Virtual Paralela (por sus siglas en inglés *Parallel Virtual Machine*) (PVM).
- **kernel:** Distribución de kernels booteables.
- **web-server:** Servidor Web.
- **python:** Python 2.7 y Python 3.x.
- **java:** Java 6.2.
- **base:** Roll base del sistema.
- **perl:** Soporte para la nueva versión de Perl.
- **sgc:** Instala y configura el planificador de tareas Sun Grid Engine (v GE2011).
- **area51:** Servicios y utilidades relacionadas con la seguridad del sistema.
- **6.8:** S.O. CentOS 6.8 discos 1 y 2.

Tras haber seleccionado los Rolls que instalaremos, la siguiente pantalla (Fig. 2.17) muestra el formulario en el que se indicó la información básica del clúster la cual comprende:

Fully-Qualified Host Name: Nombre de dominio del clúster, como se conocerá en la red externa (obligatorio).

Cluster Name: Nombre del clúster, utilizado para identificarlo dentro de las herramientas del mismo como Ganglia (opcional).

Certificate Organization, Locally, State and Country: Organización, localidad, estado y país donde se encuentra instalado el clúster (opcional).

Contact: Dirección de correo electrónico del administrador del clúster (opcional).

URL: Dirección URL de la empresa o dependencia a la cual pertenece el clúster (opcional).

Latitude/Longitude: Coordenadas Geoespaciales donde se localiza el clúster (op-

cional).

Cluster Information	
Fully-Qualified Host Name	hpc.enesmorelia.unam.mx
Cluster Name	Rocks-HPC
Certificate Organization	ENESMorelia-UNAM
Certificate Locality	Morelia
Certificate State	Michoacan
Certificate Country	MX
Contact	fhernandez@enesmorelia.unam.mx
URL	http://www.enesmorelia.unam.mx/
Latitude/Longitude	N19.65 W101 22

Back Next

Figura 2.17: Información básica

En la Figura 2.17 podemos observar también los datos que se emplearon para la configuración básica de nuestro cluster.

El siguiente paso a llevar a cabo es la configuración de la dirección IP y máscara de subred (Netmask) de la interfaz “*eth1*” de la red interna del clúster (Fig. 2.18(a)), así como la interfaz “*eth0*” de la red pública o mediante la cual nos conectaremos al clúster (Fig. 2.18(b)). Se procede de igual manera con la puerta de enlace (Gateway) y los servidores DNS para la red pública (Fig. 2.18(c)).

(a) *Config. de red privada*(b) *Config. de red pública*(c) *Gateway y DNS*

Figura 2.18: Configuración de red

Finalmente, los últimos tres pasos en la configuración de la instalación consisten en establecer la contraseña de root (Fig. 2.19(a)); posteriormente se establece la zona horaria que en este caso particular es “America/Mexico_City” (Fig. 2.19(b)). Por último seleccionamos el tipo de particionado que deseamos hacer en el disco, altamente recomendado que sea “Auto Partitioning” (Fig. 2.19(c)). Al dar click en el último botón “Next” el sistema se instalará por completo en nuestro front-end y posteriormente se reiniciará para poder acceder a él y continuar con las configuraciones post-instalación así como la instalación de los nodos.

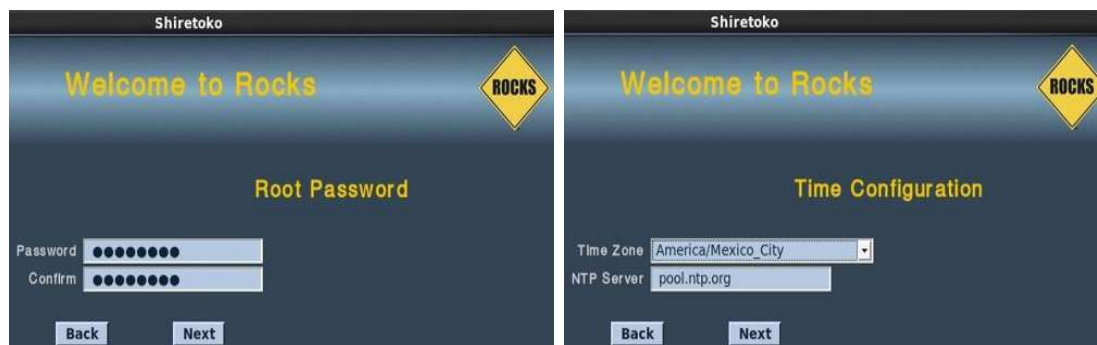
(a) *Contraseña de administrador*(b) *Zona horaria*(c) *Particionado de disco*

Figura 2.19: Etapa final de la configuración de la instalación

Es en este punto donde el sistema iniciará la descarga de los Rolls y la instalación de los mismos, para lo cual se estarán viendo mensajes solicitando los rolldisks; colocamos el disco correspondiente a cada caso.

Post-Instalación

Cuando el sistema termina la instalación se reinicia de manera automática. En este momento se procede a realizar configuraciones post-instalación como lo son la localización de la zona horaria y la configuración del idioma de teclado.

En el primer inicio de sesión debemos acceder como “root” para lo cual seleccionamos “Other” e introducimos los datos para el usuario “root” (Fig: 2.20).

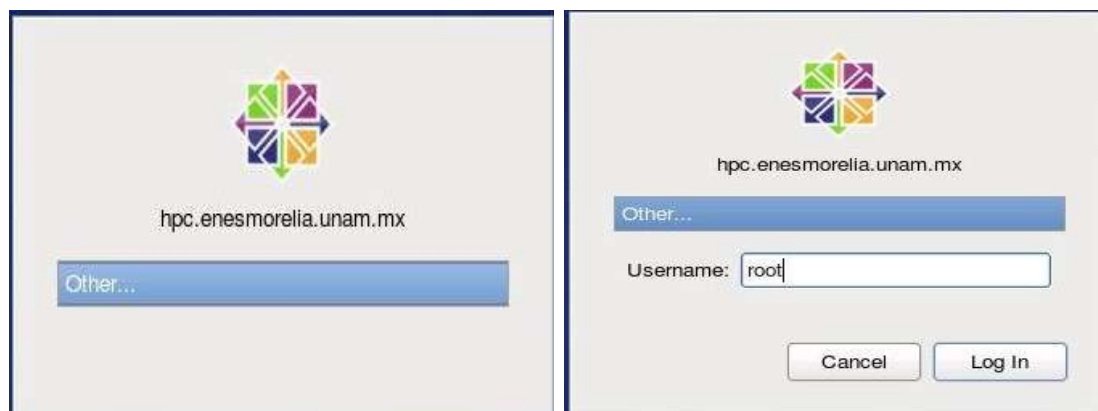
(a) *Otro usuario*(b) *Usuario root*(c) *Password de root*

Figura 2.20: Primer inicio de sesión.

Pese a haber configurado una zona horaria durante la instalación del sistema operativo, se pudo observar, en la barra de menú del escritorio al acceder por primera vez al sistema, que la configuración de zona y hora no es la correcta pues el sistema muestra “Boston” como zona horaria; con la finalidad de tener configuraciones correctas podemos cambiar esta configuración en el botón “Edit” lo que nos dará acceso a la ventana “Clock Preferences”, seleccionamos la pestaña “Locations”, introducimos la nueva ubicación en “Location Name” y aceptamos los cambios (Fig: 2.21).



(a) Menú

(b) Ubicación



(c) Datos de la ubicación

Figura 2.21: Reconfiguración de la Ubicación

Es importante hacer notar que el idioma de teclado configurado por defecto en el sistema es inglés, lo que nos puede causar problemas al momento de introducir contraseñas u otros comandos al sistema. Para garantizar que la configuración del teclado se mantenga, una muy buena solución, es cargar la distribución deseada del

teclado en el archivo `/etc/rc.local` el cual le indica al sistema las configuraciones de arranque. Para ello abrimos una terminal y ejecutamos el siguiente comando:

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

Ahora procedemos a cambiar la configuración de idioma para ello vamos al menú System → Preferences → Keyboard (Fig. 2.22(a)) y en la ventana “Keyboard Preferences” vamos a la pestaña “Layouts” y hacemos click en el botón “Add” (Fig. 2.22(b)). En la ventana emergente seleccionamos “Mexico” en la sección “Country” y damos “Add” (Fig. 2.23), de regreso en la ventana “Keyboard Preferences” seleccionamos el roundbottom de “Spanish (Latin American)” y podemos cerrar. Si así lo preferimos, podemos eliminar el idioma “English (US)”.

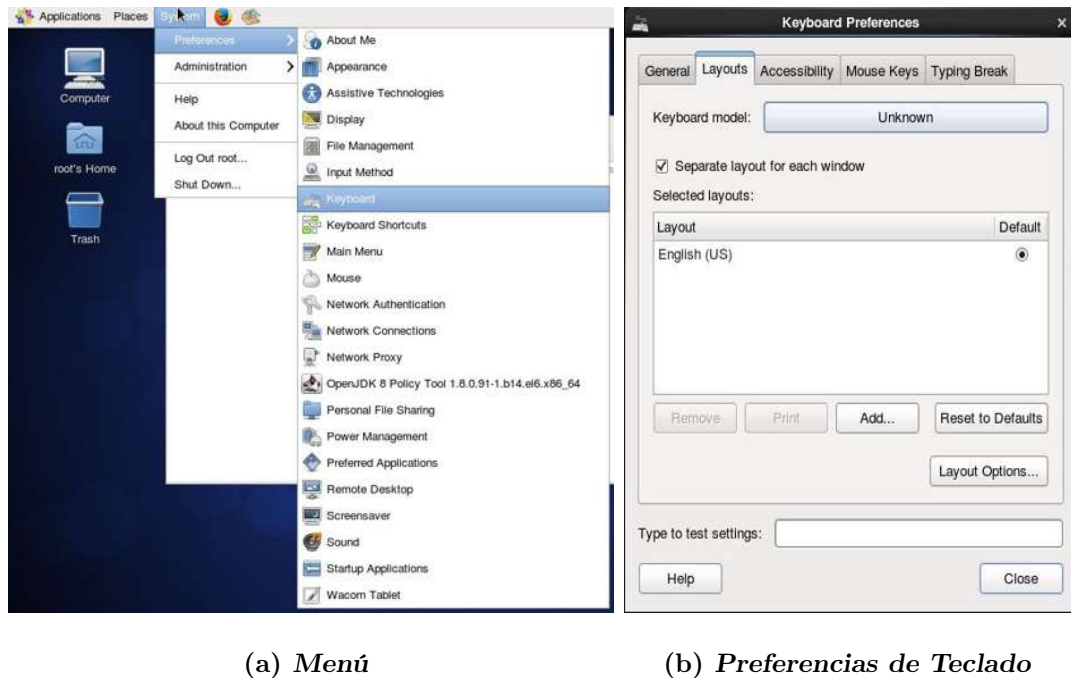
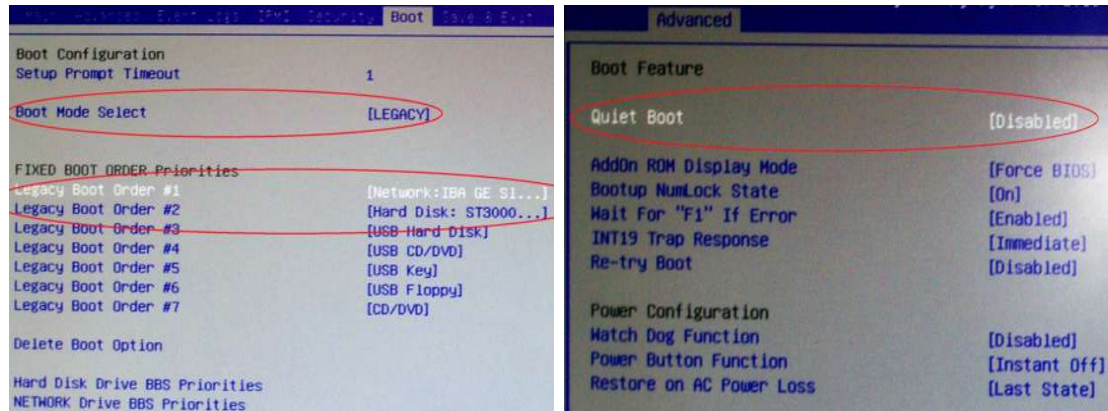


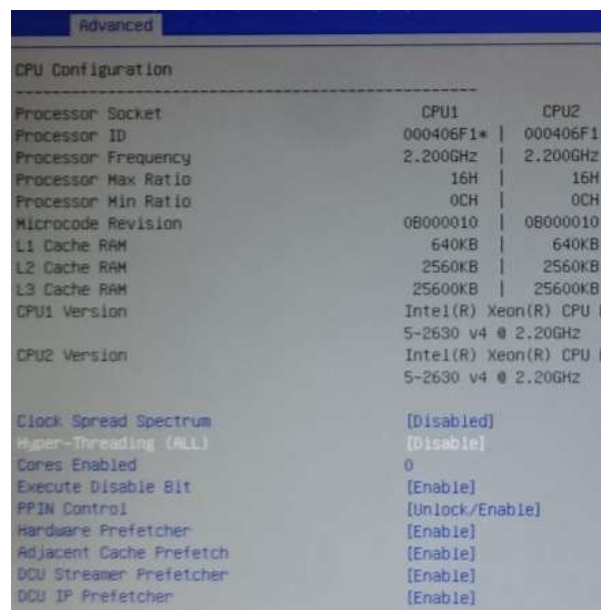
Figura 2.22: Menú para la configuración de Idioma del Teclado.

la penstaña “Advanced” (Fig: 2.24(c)).



(a) *Arranque PXE*

(b) *Quiet Boot*



(c) *HyperThreading*

Figura 2.24: Configuración del BIOS en nodos de cómputo.

Habiendo realizado los cambios previamente descritos, guardamos la configuración y apagamos los nodos; es importante resaltar que el sistema se instalará en los nodos mediante una instrucción en el front-end pues es este último el encargado de dar de alta en el sistema los nodos, instalarles el SO y reiniciarlos como elementos del clúster.

Instalación

Habiendo preparado los nodos de cómputo para el arranque desde la red, y estando estos apagados, procedemos a la instalación de los mismos dentro del clúster. Dado que el front-end se encarga de configurar los nodos dentro de la red interna del cluster, el front-end debe correr un servicio dhcp. Por esta razón verificamos que el servicio esté ejecutándose corriendo el comando: `[root@hpc ~]#service dhcpd status` a lo que el sistema debe respondernos `dhcpd (pid xxxx)is running...`; si, por el contrario, el mensaje es `dhcpd (pid xxxx)is stopped...` lo podemos reiniciar con el comando:

```
[root@hpc ~]#service dhcpd restart
```

En este punto procederemos a la parte medular de la instalación de los nodos mediante el comando:

```
[root@hpc ~]#insert-ethers
```

Este comando nos desplegará la ventana de la Figura 2.25. Ya que todos los nodos que se instalarán serán de cálculo seleccionamos “Compute”. Enseguida encendemos los nodos uno a uno. El sistema mostrará la pantalla de la Figura 2.26 que muestra la dirección MAC de los nuevos nodos, el nombre asignado al host y una columna con asterisco (*) que indica cuando el sistema ha sido instalado en el nodo.

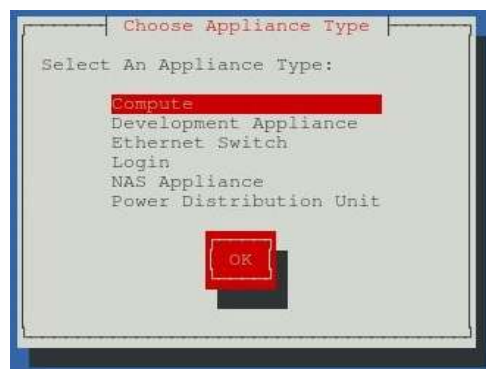


Figura 2.25: Pantalla de insert-ethers

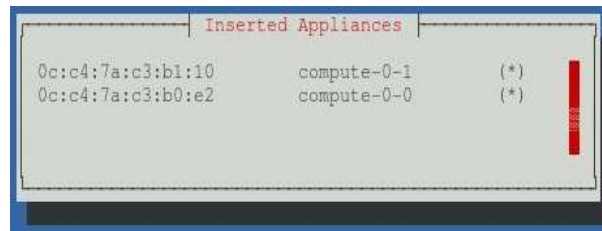


Figura 2.26: Inserción de nuevos nodos

El nombramiento de los nodos obedece a una estructura compuesta por el tipo de nodo (compute); el número de rack al que pertenece y el número (rank) que ocupa el equipo dentro del rack. Se recomienda esperar a que cada nuevo equipo encendido esté marcado con este asterisco antes de proceder a encender el siguiente. Así nos aseguramos de que cada nodo añadido posea el nombre y numeración deseada.

2.7.3. Algunos comandos útiles

Listar los nodos ya instalados

```
[root@hpc ~]#rocks list host
```

Listar la información de red de todos los nodos

```
[root@hpc ~]#rocks list host interface
```

Listar las direcciones MAC de los nodos

```
[root@hpc ~]#rocks run host <nombre_nodo> ifconfig | grep HWaddr
```

Eliminar un nodo de la lista

```
[root@hpc ~]#rocks remove host <nombre_nodo>
```

Reinstalar un nodo

```
[root@hpc ~]#rocks set host boot <nombre_nodo> action=install
```

```
[root@hpc ~]#ssh <nombre_nodo> ‘‘shutdown -r now’’
```

Acceder a un nodo

```
[root@hpc ~]#ssh <nombre_nodo>
```

Ejecutar un comando de manera remota a un nodo

```
[root@hpc ~]#ssh <nombre_nodo> <comando>
```

Listar los rolls instalados

```
[root@hpc ~]#rocks list roll
```

Sincronizar la configuración del clúster en todos los nodos

```
[root@hpc ~]#rocks sync config
```

Para agregar un usuario hay que realizar los siguientes pasos:

1. **Crear una cuenta**

```
[root@hpc ~]#useradd <nombre_usuario>
```

2. **Asignación de contraseña**

```
[root@hpc ~]#passwd <nombre_usuario>
```

3. **Sincronización de la cuenta con el resto de los nodos**

```
[root@hpc ~]#rocks sync users
```

2.8. Conclusiones de capítulo

En una primera opción se procuró instalar el SO CentOS 7 al ser el que cuenta con un mayor tiempo de soporte y ser de licencia libre, lamentablemente no se encontró documentación relacionada con la implementación de HPC. En la búsqueda de información se encontró que existe una distribución para HPC que se monta sobre CentOS

7, ésta es StackIQ, sin embargo se tuvieron demasiados conflictos al intentar instalarlo en un equipo portátil que inicialmente se había planeado ocupar como front-end. Se estimó que el problema radicó en permisos para acceder a los repositorios del sistema por políticas de seguridad en la red interna de la institución. Por tal motivo y tras varios intentos fallidos de concretar la instalación se optó por cambiar la decisión e implementar el clúster sobre la distribución Rocks en su versión más actual 6.2 que se monta sobre una plataforma CentOS 6.6.

Inicialmente se pretendió usar un equipo portátil como front-end para destinar la interfaz inalámbrica para la red externa del clúster dejando así la interfaz *eth0* para la red interna. Esto causó conflictos en la conexión del clúster con el resto de la red de la ENES. Con la intención de solucionar este problema se modificaron los archivos de configuración de las interfaces cambiando el tipo de interfaz de *BRIDGE* a *ETHERNET*. Con lo anterior se logró establecer conexión con el front-end pero deshabilitó el servicio Protocolo de Configuración Dinámica de Host (por sus siglas en inglés **D**inamic **H**ost **C**onfiguration **P**rotocol) (DHCP) que se corre en el front-end y que es necesario para la instalación de los nodos.

Esta serie de complicaciones nos llevaron a desestimar el uso del equipo portátil como front-end utilizando en cambio el equipo desktop que actualmente está en uso con tal fin.

Habiendo realizado una primera instalación exitosa del SO en el clúster, el administrador del clúster requirió implementar un algoritmo de renderizado implementando las librerías de Python *pip*, *lxml*, *sklearn*, *scipy*, *pandas*, *mpi4py* y *opencv* en sus versiones más actuales mismas que no se encontraron instaladas en la distribución del clúster y que solo se lograron agregar a la distribución con la reinstalación del sistema, pero ahora sobre una plataforma CentOS 6.8 cuyo proceso de instalación se ha mostrado ya en la sección 2.7.

Pruebas y Resultados

3.1. Introducción

El multiprocesamiento es la conjunción de software y hardware, ambos adaptados para la ejecución, optimización y administración de programas los cuales tienen subrutinas que pueden ser ejecutadas de manera simultánea sin que ello implique poner en riesgo la integridad de los resultados finales.

El multiprocesamiento surgió como una alternativa para optimizar la ejecución de tareas que consumían grandes recursos de cómputo y a la vez gran cantidad de tiempo.

Las necesidades de cómputo de numerosas aplicaciones obligan a desarrollar software eficiente y seguro para plataformas multiprocesador. Además, el auge de los procesadores multinúcleo y de las redes de computadoras ha aumentado la difusión del procesamiento paralelo. No obstante, para utilizar los sistemas paralelos y/o distribuidos de forma eficiente es necesaria la programación paralela.

En este capítulo describiremos algunas herramientas de desarrollo en paralelo empleando la librería MPI. El Estándar MPI es una librería estándar de paso de mensajes basada en el consenso del foro MPI, el cual cuenta con la participación de más de 40 organizaciones, incluyendo vendedores, investigadores, desarrolladores y

usuarios. La finalidad de MPI es establecer un estandar portable, eficiente y flexible para el paso de mensajes que sea ampliamente usado en la escritura de programas de paso de mensajes. Las ventajas del desarrollo de software usando MPI son muy similares a las del diseño de portabilidad, eficiencia y flexibilidad. MPI no es un estandar IEEE o ISO, pero se ha convertido en el estandar industrial para la escritura de programas con paso de mensajes en plataformas HPC.

El objetivo de este capítulo es proporcionar una idea muy básica sobre cómo desarrollar y correr programas en paralelo de acuerdo al estandar MPI. Iniciaremos con una descripción del estandar MPI, un poco de historia e información básica, lenguajes de programación en los que se ha implementado el estandar, clasificación y descripción de algunas de las principales rutinas y funciones de MPI encargadas tanto de Comunicaciones Point-to-Point como de Comunicación Colectiva. Finalmente se presenta un caso de aplicación sencillo que nos permita evaluar si el clúster está paralelizando las tareas adecuadamente.

3.2. Marco teórico

Definición 3.2.1 *MPI: 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.*[Wikipedia, 2017a]

El paso de mensajes es una técnica empleada en programación concurrente para aportar sincronización entre procesos y permitir la exclusión mutua. Su principal característica es que no precisa de memoria compartida, por lo que es muy importante en la programación de sistemas distribuidos. Los elementos principales que intervienen en el paso de mensajes son el proceso que envía, el que recibe y el mensaje. Dependiendo de si el proceso que envía el mensaje espera o no a que el mensaje sea recibido, se puede hablar de paso de mensajes síncrono o asíncrono. En el paso de mensajes asíncrono, el proceso que envía, no espera a que el mensaje sea recibido,

y continúa su ejecución, siendo posible que vuelva a generar un nuevo mensaje y a enviarlo antes de que se haya recibido el anterior. Por este motivo se suelen emplear buzones, en los que se almacenan los mensajes a espera de que un proceso los reciba. Generalmente, empleando este sistema, el proceso que envía mensajes sólo se bloquea o para, cuando finaliza su ejecución, o si el buzón está lleno. En el paso de mensajes síncrono, el proceso que envía el mensaje espera a que un proceso lo reciba para continuar su ejecución. Por esto se suele llamar a esta técnica encuentro, o rendezvous. Dentro del paso de mensajes síncrono se engloba la llamada a procedimiento remoto, muy popular en las arquitecturas cliente/servidor.

MPI es un protocolo de comunicación entre computadoras. Es el estándar para la comunicación entre los nodos que ejecutan un programa en un sistema de memoria distribuida. Las implementaciones en MPI consisten en un conjunto de bibliotecas de rutinas que pueden ser utilizadas en programas escritos en los lenguajes de programación *C*, *C++*, *Fortran* y *Ada*. La ventaja de MPI sobre otras bibliotecas de paso de mensajes, es que los programas que utilizan la biblioteca son portables (dado que MPI ha sido implementado para casi toda arquitectura de memoria distribuida), y rápidos, (porque cada implementación de la biblioteca ha sido optimizada para el hardware en la cual se ejecuta).

3.2.1. Historia

Para el diseño de MPI se tomaron en cuenta las características más atractivas para el paso de mensajes en los sistemas existentes por lo que en el esfuerzo por estandarizar MPI se involucró a cerca de 60 personas de 40 organizaciones diferentes principalmente de E.U. y Europa.

El proceso de estandarización comenzó en el taller de estándares para el paso de mensajes en un ambiente con memoria distribuida, patrocinado por el Centro de Investigación en Computación Paralela en Williamsburg, Virginia, Estados Unidos (abril 29-30 de 1992). Se llegó a una propuesta preliminar conocida como MPI1,

enfocada principalmente en comunicaciones punto a punto sin incluir rutinas para comunicación colectiva y no presentaba tareas seguras.

En un ambiente de comunicación con memoria distribuida y rutinas de paso de mensajes de nivel bajo, los beneficios de la estandarización son muy notorios. La principal ventaja al establecer un estándar para el paso de mensajes es la portabilidad y el ser fácil de utilizar. El estándar final fue presentado en la conferencia de Supercómputo en noviembre de 1993, constituyéndose así el foro para el MPI. MPI-1 apareció en 1994, el estándar MPI-2 fue lanzado en 1997.

3.2.2. Características de MPI

Estandarización: MPI es compatible con prácticamente todas las plataformas HPC.

Portabilidad : Hay poca o ninguna necesidad de modificar el código fuente al migrar la aplicación a una plataforma diferente que soporte y/o sea compatible con el estándar MPI.

Oportunidades de Rendimiento: Las implementaciones de proveedores deberían ser capaces de aprovechar las características de hardware nativo para optimizar el rendimiento. Cualquier implementación es libre de desarrollar algoritmos optimizados.

Amplia funcionalidad: Existen más de 430 rutinas definidas en MPI-3, que incluye la mayoría de las de MPI-2 y MPI-1.

Disponibilidad: Hay una variedad de implementaciones disponibles como *mpich* y *LAM-MPI*, tanto de proveedores como de dominio público.

Definición 3.2.2 *Un Programa, Múltiples Datos (por sus siglas en inglés Single Program, Multiple Data) (SPMD). Técnica empleada para lograr paralelismo; considerado una subcategoría de Múltiples Instrucciones, Múltiples Datos (por sus siglas en inglés Multiple Instruction, Multiple Data) (MIMD). Las tareas son separadas y ejecutadas simultáneamente en múltiples procesadores con diferentes entradas para*

obtener los resultados con mayor rapidez. Estilo más común de programación paralela.

Definición 3.2.3 *MIMD: Técnica empleada para lograr paralelismo. Las máquinas que usan MIMD tienen un número de procesadores que funcionan de manera asíncrona e independiente. En cualquier momento, cualquier procesador puede ejecutar diferentes instrucciones sobre distintos datos.*

La arquitectura MIMD puede utilizarse en una amplia gama de aplicaciones como el diseño asistido, simulación, modelado y en interruptores. Las computadoras MIMD pueden categorizarse por tener memoria compartida o distribuida, clasificación que se basa en cómo el procesador accede a la memoria. La memoria compartida de las máquinas puede estar basada en buses, extensiones, o de tipo jerárquico. Las máquinas con memoria distribuida pueden tener esquemas de interconexión en hipercubo o malla.

Siguiendo el modelo SPMD, el usuario escribirá su aplicación como un proceso secuencial del que se lanzarán varias instancias que cooperan entre sí.

3.2.3. Implementaciones MPI

La mayoría de las implementaciones de MPI se realizan en una combinación de *C*, *C++* y lenguaje ensamblador, *C++* y *Fortran*. Además de la corriente principal de MPI para programación de alto rendimiento, MPI se ha utilizado con *Python*, *Perl* y *Java*.

Python

Hay al menos cinco intentos conocidos de aplicar MPI para *Python*: *mpi4py*, *PyPar*, *PyMPI*, *MYMPI*, y en el submódulo *ScientificPython MPI*.

PyMPI es notable porque se trata de una variante de python intérprete multinodo de la aplicación del propio intérprete, más que el código lo que se ejecuta es el

intérprete. Implementa la mayor parte de la especificación MPI y trabaja automáticamente con el código compilado que necesita para hacer llamadas MPI.

PyPar, *MYMPI*, y el módulo *ScientificPython* están diseñados para trabajar como un típico módulo realizando una declaración de importación. Estos hacen el trabajo del codificador para decidir cuándo y dónde pertenece la llamada a `MPI_Init`.

Java

Aunque Java no dispone de una lista de funciones MPI vinculantes, ha habido varios intentos de puente de Java y MPI, con diferentes grados de éxito y compatibilidad.

Uno de los primeros intentos fue Bryan Carpenter's `mpiJava`, básicamente una colección de envoltorios JNI de la biblioteca local MPI de *C*, el resultado es un híbrido con limitada portabilidad, y que debe recompilarse con la biblioteca específica de la biblioteca MPI que se utiliza. Este proyecto original definió el *mpiJava API* que es una API MPI para *Java* equivalente a la de *C++* y que posteriormente otros proyectos *Java MPI* han utilizado.

Una alternativa, aunque menos utilizada, es la API MPJ, diseñada para ser más orientada a objetos y más cerca a los convenios de codificación de Sun Microsystems.

Algunas de las partes más difíciles de cualquier aplicación para *Java MPI* surgen de las limitaciones del propio lenguaje de programación y sus peculiaridades, como la falta de los punteros explícitos y espacio de las direcciones de memoria lineales para sus objetos, que hacen ineficientes las transferencias de arrays multi-dimensionales y objetos complejos.

La cuestión clave en la actualidad no es debatir si es mejor la JNI con respecto al Java puro, sino proporcionar un mecanismo flexible de los programas de intercambiar los protocolos de comunicación.

El objetivo de este proyecto es proporcionar un sistema de mensajería de referencia de Java basada en el estándar MPI. La aplicación sigue una arquitectura basada en

la idea de los controladores de dispositivo.

.NET

Windows Compute Cluster Server utiliza el Microsoft Messaging Passing Interface v2 (MS-MPI) para la comunicación entre los nodos de procesamiento sobre el clúster de red. La interfaz de programación de aplicaciones consta de más de 160 funciones. MS-MPI fue diseñado, con algunas excepciones, debido a consideraciones de seguridad, para abarcar el conjunto completo de funciones que MPI-2 aplicado en MPICH2.

Hay también una aplicación completa *.NET* de MPI: *Pure Mpi.NET*. La API orientada a objetos es potente y fácil de utilizar para la programación paralela.

Pure Mpi.NET ha sido desarrollada sobre la última tecnología *.NET*, incluido en la **Windows Communication Foundation** (WCF). Esto le permite declarativamente especificar la configuración de llamada, la configuración de entorno final y las necesidades de rendimiento.

3.2.4. Algunos fundamentos de MPI

Con MPI el número de procesos requeridos se asigna antes de la ejecución del programa, y no se crean procesos adicionales mientras la aplicación se ejecuta. A cada proceso se le asigna una variable *rank*, que identifica a cada proceso, en un rango de 0 a $p - 1$, donde p es el número total de procesos. El control de la ejecución del programa se realiza mediante la variable *rank* la cual permite determinar que proceso ejecuta determinada porción de código.

Definición 3.2.4 *Comunicador: Colección de procesos, los cuales pueden enviar mensajes el uno al otro.*

El comunicador básico se denomina `MPI_COMM_WORLD` y se define mediante una macro del lenguaje *C* el cual agrupa todos los procesos activos durante la ejecución

de una aplicación. Las llamadas de MPI se dividen en cuatro clases:

1. **Utilizadas para inicializar, administrar y finalizar comunicaciones:** Permiten inicializar la biblioteca de paso de mensajes, identificar el número de procesos (*size*) y el rango de los procesos (*rank*).
2. **Utilizadas para transferir datos entre un par de procesos:** Incluyen operaciones de comunicación punto a punto, para diferentes actividades de envío-recepción.
3. **Para transferir datos entre varios procesos:** Conocidas como operaciones grupales, que proveen operaciones de comunicaciones entre grupos de procesos.
4. **Utilizadas para crear tipos de datos definidos por el usuario:** Proveen flexibilidad en la construcción de estructuras de datos complejos.

Llamadas utilizadas para inicializar, administrar y finalizar comunicaciones

MPI dispone de cuatro funciones primordiales que se utilizan en todo programa con MPI:

MPI_Init: Permite inicializar una sesión MPI. Esta función debe ser utilizada antes de llamar a cualquier otra función de MPI.

MPI_Finalize: Permite terminar una sesión MPI. Esta función debe ser la última llamada a MPI que un programa realice. Permite liberar la memoria usada por MPI.

MPI_Comm_size: Permite determinar el número total de procesos que pertenecen a un comunicador.

MPI_Comm_rank: Permite determinar el *rank* del proceso actual.

Llamadas utilizadas para transferir datos entre dos procesos

La transferencia de datos entre dos procesos se consigue mediante las llamadas `MPI_Send` y `MPI_Recv`. Estas devuelven un código que indica su éxito o fracaso:

`MPI_Send`: Envía información desde un proceso a otro.

`MPI_Recv`: Recibe información desde otro proceso.

Ambas funciones son bloqueantes (comunicación *síncrona*), es decir que el proceso que realiza la llamada se bloquea hasta que la operación de comunicación se complete. Las versiones no bloqueantes (comunicación *asíncrona*) son `MPI_Isend` y `MPI_Irecv`, respectivamente. Estas llamadas inician la operación de transferencia pero su finalización debe ser realizada de forma explícita mediante llamadas como `MPI_Test` y `MPI_Wait`. `MPI_Wait` es una llamada bloqueante y retorna cuando la operación de envío o recepción se completa. `MPI_Test` permite verificar si la operación de envío o recepción ha finalizado, esta función primero chequea el estado de la operación de envío o recepción y luego retorna.

Llamadas utilizadas para transferir datos entre varios procesos

MPI posee llamadas para comunicaciones grupales que incluyen operaciones tipo difusión (*broadcast*), recolección (*gather*), distribución (*scatter*) y reducción. Algunas de las funciones que permiten realizar estas comunicaciones son:

`MPI_Barrier`: Realiza operaciones de sincronización. En estas operaciones no hay intercambio de información. Se emplea para finalizar una etapa del programa, asegurándose de que todos los procesos han terminado antes de dar comienzo a la siguiente.

`MPI_Bcast`: Permite a un proceso enviar una copia de sus datos a otros procesos dentro de un grupo definido por un comunicador.

`MPI_Scatter`: Establece una operación de distribución, en la cual un dato o arreglo de datos se distribuye en diferentes procesos.

MPI_Gather: Establece una operación de recolección, en la cual los datos son recolectados en un sólo proceso.

MPI_Reduce: Permite que el proceso raíz recolecte datos desde otros procesos en un grupo, y los combine en un solo ítem de datos. Por ejemplo, se podría utilizar una operación reducción, para calcular la suma de los elementos de un arreglo que se distribuyó en algunos procesos.

Llamadas utilizadas para crear tipos de datos definidos por el usuario

Para definir nuevos tipos de datos se puede utilizar la llamada `MPI_Type_struct` para crear un nuevo tipo o se puede utilizar la llamada `MPI_Pack` para empaquetar los datos.

Estructura de un mensaje en MPI

- **Cuerpo:** Contiene los datos que serán enviados. Está conformado por 3 piezas de información:
 - **Buffer:** Localidad de memoria dónde se encuentran los datos de salida o se almacenan los datos de entrada.
 - **Tipo de Dato:** indica el tipo de los datos que se envían, puede ser un primitivo como un `int` o construido a través de datos primitivos como en el caso de estructuras en *C*.
 - **Count:** Número de secuencia que, junto al tipo de datos, permiten al usuario agrupar ítems de datos de un mismo tipo en un solo mensaje.
- **Envoltura:** Típicamente contiene:
 - **Fuente:** Identifica el proceso fuente.
 - **Destino:** Identifica el proceso receptor.
 - **Comunicador:** Especifica el grupo de procesos a los que pertenecen la fuente y destino.

- **Etiqueta:** Permite clasificar el mensaje, es un entero definido por el usuario que puede ser utilizado para distinguir los mensajes que recibe el proceso.

MPI estandariza los tipos de datos primitivos, evitando que el programador se preocupe de las diferencias que existen entre ellos, cuando se encuentran en distintas plataformas.

3.2.5. Funciones básicas

Existe un gran número de instrucciones y rutinas de la librería MPI que podemos consultar en la API en [SPI, 2017].

3.3. Pruebas

Es importante probar el funcionamiento del clúster, verificar que el sistema esté paralelizando las tareas de manera adecuada de modo que todos los nodos trabajen en la solución del problema. Con tal finalidad creamos un programa en *C* bastante sencillo, en el que cada nodo imprime un mensaje, haciendo uso de las funciones del estándar MPI que ya hemos analizado en la Sección 3.2.4.

Dicho programa debe ejecutarse desde una cuenta de usuario que no sea la de root por lo que es importante crear dicha cuenta con ayuda de los comandos listados en la Sección 2.7.3.

3.3.1. Estructura del archivo `holamundompi.c`

Primeramente no debemos olvidar incluir los archivos de cabecera que serán necesarios para el funcionamiento del programa

mpi.h - Proporciona las subrutinas del estándar MPI necesario en la paralelización del problema.

stio.h - Subrutinas de entrada/salida.

string.h - Subrutinas para el manejo de cadenas (sprintf, strcat).

Se definen las variables que emplearemos en el programa de manera local, por lo que se declaran dentro de la función main. Estas variables son:

idstr - Guarda la cadena que contiene el identificador de todo procesador esclavo.

buff - Buffer de comunicación.

hostname - Nombre del host en el que radica el procesador en turno. Éste se captura con la función `gethostname`.

numprocs - Número total de procesos que se ejecutarán. Este valor es pasado como parametro al archivo ejecutable en la línea de comandos y se almacena en la variable mediante la función `MPI_Comm_size` justo despues de inicializado el entorno MPI.

myid - Identificador del procesador en turno.

i - Entero que sirve como contador de ciclos.

stat - Variable de tipo `MPI_Status` que almacena un código referete al estado de la comunicación. [SPI, 2017]

El caso inicial se presenta con el identificador de proceso 0, que se define como el proceso maestro. Primeramente manda un mensaje al usuario indicando el total de procesos e identificándose como el proceso maestro. Posteriormente manda un saludo con `MPI_Send` a todos y cada uno de los procesos esclavos (1er ciclo). Finalmente espera, recibe e imprime la respuesta proveniente de los procesos esclavo (2o ciclo).

El trabajo de los procesos esclavos se presenta en la excepción del caso anterior (instrucción `else`). Es aquí dónde cada proceso esclavo recibe el saludo del proceso maestro y estructura el mensaje con el cual responde, llena el buffer de comunicación y envía el mensaje al proceso maestro.

Finalmente se cierra el entorno MPI con la instrucción `MPI_Finalize`. A continuación se muestra el contenido del código fuente:

holamundompi.c

```
1 #include <mpi.h>
2 #include <stdio.h>
3 #include <string.h>
4
5 #define BUFSIZE 128
6 #define HOSTNAMELENGTH 50
7
8 int main(int argc, char *argv[]){
9     char idstr[32];
10    char buff[BUFSIZE];
11    char hostname[HOSTNAMELENGTH];
12    int numprocs;
13    int myid;
14    int i;
15    MPI_Status stat;
16    MPI_Init(&argc,&argv);
17    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);//No. de procesos
    en este COMM_WORLD
18    MPI_Comm_rank(MPI_COMM_WORLD,&myid);//ID del proceso
19    if(myid == 0){//PROCESO MAESTRO
20        gethostname(hostname, HOSTNAMELENGTH);
21        printf("Tenemos %d procesadores, yo soy el proceso
    maestro: %d en maquina: %s\n",numprocs,myid,hostname);
22        for(i=1; i<numprocs; i++){
23            sprintf(buff, "F: Hola procesador %d!\n", i); //Llena
    el buffer con un saludo
24            MPI_Send(buff, BUFSIZE, MPI_CHAR, i, 0,
    MPI_COMM_WORLD); //envia el saludo a todos los
    procesos
25        }
26        for(i=1;i<numprocs;i++){
27            MPI_Recv(buff, BUFSIZE, MPI_CHAR, i, 0,
    MPI_COMM_WORLD, &stat); //recive de los rank 1
    hasta n-1
28            printf("%s\n", buff); //imprime el mensaje recibido
    de los procesos esclavo
29        }
30    }else{//PROCESOS ESCLAVO
31        gethostname(hostname, HOSTNAMELENGTH);
32        MPI_Recv(buff, BUFSIZE, MPI_CHAR, 0, 0, MPI_COMM_WORLD,
    &stat);//Recive del rank 0 el saludo
33        sprintf(idstr, "C: Procesador %d en maquina: %s ",
    myid,hostname);
34        strcat(buff, idstr);
35        strcat(buff, "reportandose");
36        MPI_Send(buff, BUFSIZE, MPI_CHAR, 0, 0,
    MPI_COMM_WORLD);//Envia al rank 0 la respuesta
37    }
38    MPI_Finalize();
39    return 0;
40 }
```

3.4. Presentación de resultados

Para compilar nuestro programa llamado `holamundompi.c` ejecutamos:

```
[miangel@hpc ~]#mpicc -o holamundompi holamundompi.c
```

En caso de que requiramos agregar funciones matemáticas agregamos la librería con `mpicc -lm -o <ejecutable> <fuente>`.

La línea de comando en la terminal para la ejecución del trabajo tiene la estructura:

`mpirun -np <num_procs> ./<ejecutable>`. Ejemplo:

```
[miangel@hpc ~]#mpirun -np 100 ./holamundompi
Tenemos 4 procesadores, yo soy el proceso maestro: 0 en
  maquina: hpc.enesmorelia.unam.mx
F: Hola procesador 1!
C: Procesador 1 en maquina: hpc.enesmorelia.unam.mx
  reportandose
F: Hola procesador 2!
C: Procesador 2 en maquina: hpc.enesmorelia.unam.mx
  reportandose
  .
  .
  .
```

Observamos que no se están usando los nodos del clúster ya que la máquina `hpc.enesmorelia.unam.mx` es el front-end o nodo maestro, para usar los nodos del clúster se requiere crear un archivo de `hosts` con datos relativos a los nodos del clúster mismos que podemos obtener utilizando el comando `rocks list host`. Con los datos obtenidos (nombre del nodo y CPUs) creamos nuestro archivo llamado “*nodes*”:

```
hpc slots=4 max_slots=9999
compute-0-1 slots=20 max_slots=9999
compute-0-2 slots=20 max_slots=9999
```

con `slots` indicamos el número de CPUs de cada nodo y con `max_slots` el máximo número de procesos que se podrán correr, Este último parámetro se utiliza para

simular un número máximo de CPUs mayor al real y puede omitirse. Ejecutamos el mismo programa modificando la estructura de la instrucción de esta manera:

```
mpirun -hostfile <hostfile_name> -np <num_procs> ./<ejecutable>
```

El parámetro `-hostfile` nos permite incluir el archivo *nodes* del cual se toman las cadenas de texto que permiten hacer la búsqueda e identificación de los nodos que forman parte del clúster. Ahora los procesos se ejecutan en las máquinas *compute-0-1* y *compute-0-2* que son los nodos de procesamiento pudiendo incluir también, como en este caso, el front-end. Si no se quiere usar el archivo *nodes*, se puede especificar la lista de nodos de procesamiento directamente en la línea de ejecución:

```
[miangel@hpc ~]#mpirun -host <hostname1>, <hostname2>, ..., <hostnameN> -np  
<numprocs> ./<ejecutable>
```

Habiendo realizado pruebas de exhaustividad incluyendo el front-end en el archivo *nodes* e incrementando considerablemente el parámetro *num_procs* en el comando `mpirun` con la finalidad de verificar el número máximo de procesos que podrían paralelizarse en el clúster (544 procesos). En la Figura 3.1 se muestra un resumen de la ejecución del programa.

```
[miangel@hpc ~]#mpirun -hostfile nodes -np 544 ./holamundompi
Tenemos 44 procesadores, yo soy el proceso maestro: 0 en
  maquina: hpc.enesmorelia.unam.mx
F: Hola procesador 1!
C: Procesador 1 en maquina: hpc.enesmorelia.unam.mx reportandose
.
.
F: Hola procesador 3!
C: Procesador 3 en maquina: hpc.enesmorelia.unam.mx reportandose
F: Hola procesador 4!
C: Procesador 4 en maquina: compute-0-1 reportandose
.
.
F: Hola procesador 23!
C: Procesador 23 en maquina: compute-0-1 reportandose
F: Hola procesador 24!
C: Procesador 24 en maquina: compute-0-2 reportandose
.
.
F: Hola procesador 43!
C: Procesador 43 en maquina: compute-0-2 reportandose
F: Hola procesador 44!
C: Procesador 44 en maquina: hpc.enesmorelia.unam.mx
  reportandose
.
.
F: Hola procesador 47!
C: Procesador 47 en maquina: hpc.enesmorelia.unam.mx
  reportandose
F: Hola procesador 48!
C: Procesador 48 en maquina: compute-0-1 reportandose
.
.
F: Hola procesador 67!
C: Procesador 67 en maquina: compute-0-1 reportandose
F: Hola procesador 68!
C: Procesador 68 en maquina: compute-0-2 reportandose
.
.
F: Hola procesador 543!
C: Procesador 543 en maquina: compute-0-1 reportandose
[miangel@hpc ~]
```

Figura 3.1: Salida del programa holamundompi

Conclusiones y Trabajos futuros

4.1. Conclusiones Generales

Hoy en día en la región así como en el país existe un muy amplio universo de oportunidades de desarrollo en el campo de la administración de sistemas UNIX en especial en sistemas de cómputo avanzado. Tanto dependencias como instituciones y en general empresas que así lo requieren, se han venido dando avances significativos. Se requiere hardware más sofisticado que sea capaz de dar solución a los problemas que se enfrentan día a día en el área de cómputo.

El área de la investigación no se queda atrás, las universidades se han mantenido al día con la adquisición de equipos cada vez más sofisticados tanto para el servicio a su matrícula como también los destinados al cómputo de alto rendimiento necesarios en el procesamiento de grandes bancos de información y ejecución de enormes cantidades de tareas.

Este panorama ha venido demandando capital humano mejor capacitado en el área de la administración de sistemas UNIX en particular de aquellos enfocados al alto rendimiento como es el caso del clúster en el cuál este trabajo de tesis fue enfocado.

La implementación y administración de un clúster de alto rendimiento y en general de sistemas UNIX es un tema que aún no es muy tomado en cuenta en los progra-

mas de estudio que se han enfocado más en la administración de la información, el desarrollo de aplicaciones, automatización de servicios y la administración de redes y servicios de comunicación.

Aunque existe material escrito y publicado en relación a la implementación e instalación de sistemas de alto rendimiento, es muy cierto también que, con el rápido avance de las tecnologías de este tipo, la acción de llevar a la práctica dichos manuales y más aún posteriormente mantener una buena administración de estos sistemas se vuelven tareas con más preguntas que respuestas; se convierte en una lucha constante por entender el funcionamiento del sistema y la búsqueda de información que suele darse mayormente en foros de discusión.

En lo personal, haber desarrollado este proyecto me dejó gratas experiencias, pues demandó de mi un nivel de razonamiento y análisis de la información que difícilmente había encontrado en los proyectos escolares durante mi estancia en la facultad. Si bien hago mención de la falta de capacitación respecto a la administración de sistemas UNIX, también cabe mencionar que los conocimientos adquiridos han sido una base importante para poder dar solución a los problemas como el planteado inicialmente en esta tesis.

4.2. Trabajos Futuros

Las pruebas realizadas al clúster solamente muestran una paralelización de procesos, para tener una mejor idea de las ventajas del sistema así como del rendimiento del mismo es necesario realizar otro tipo de pruebas con algoritmos mas complejos como pueden ser algoritmos de ordenamiento o algoritmos geneticos y probarlos con cantidades grandes de datos midiendo el tiempo de respuesta del cluster y comparandolo con los tiempos de un sistema secuencial.

Al momento de instalar los nodos de cómputo se observó que el sistema inserta como *compute-0-0* un nodo inexistente con la dirección MAC 0C:C4:7A:C3:AC:FA.

En un principio se pensó que algún dispositivo, en particular el switch por ser administrable, estaba siendo visto por el DHCP y esa era la razón de que se insertara ese nodo, por tal motivo se reintentó la instalación del sistema haciendo uso de un hub para la conexión de los equipos. Desafortunadamente el problema persistió, aunado a eso podemos ver que la dirección MAC es de la misma clase a las que poseen los nodos de cómputo sin coincidir ésta con ninguna de sus interfaces *eth0* y *eth1*.

Al no poder corregir en un corto plazo el problema, se opta por advertir al administrador del clúster de la existencia de ese nodo mismo que no puede ser utilizado en la ejecución de jobs.

Otro conflicto que se encontró es que al tratar de acceder a la url 10.100.0.3/ganglia para la monitorización del clúster con ganglia, la página no siempre carga, o cuando carga se observan los campos de *CPUs Total*, *Hosts up* y *Hosts down* con un valor de 0 lo que es indicativo de que ganglia no está reconociendo los nodos de cómputo.

Encontrar solución a los problemas descritos con anterioridad y probar de una manera mas exhaustiva el rendimiento del clúster son proyectos que se plantean para futuros trabajos a desarrollar con el objetivo de mejorar el funcionamiento del clúster.

Glosario

A

algoritmo

Secuencia prescrita de instrucciones bien definidas, ordenadas y finitas para realizar una tarea mediante pasos sucesivos que no generen dudas.. 27, 37

B

back-end

Nodos esclavo del clúster encargados de realizar el trabajo y atender las peticiones. 27

C

CDE

siglas en inglés de *Common Desktop Environment*. Es el entorno de escritorio gráfico para UNIX. 38, 39

cola

Contenedor para una categoría de trabajos que pueden ser asignados para ser ejecutados en un único host.. 27

D**deadline**

Tiempo limite para la conclusión de una tarea. 33

DNF

Dandified YUM: Gestor de paquetes de software que instala, actualiza y elimina los paquetes en las distribuciones Linux basadas en RPM. 36

F**framework**

Esquema para el desarrollo y la implementación de una aplicación. 30

front-end

Nodo principal del clúster mediante el cual se puede acceder y administrar éste último. 5, 16, 27, 47–52, 57, 62–64, 67, 82, 83

G**Gnome**

Entorno de escritorio para SOs GNU/Linux, UNIX y derivados UNIX como, BSD o Solaris; compuesto enteramente de software libre.. 30, 31, 38, 39

H**hilo**

Secuencia de tareas encadenadas muy pequeña que puede ser ejecutada por un sistema operativo.. 51

host

Dispositivo conectado a una red.. 64, 80, 82, 87, 89

J**job**

Unidad de trabajo que es definida por un programa ejecutable.. 87

K**KDE**

Entorno de escritorio para sistemas UNIX. 30, 31

kernel

Responsable de facilitar a los distintos programas acceso seguro al hardware.
30, 37, 41, 45, 46

L**live CD**

SO almacenado en un CD o DVD, que puede ejecutarse directamente en una computadora.. 35, 41

N**nodo**

Cada uno de los dispositivos interconectados en una misma red. 5, 8, 11, 13–17, 19–21, 23, 26, 28, 31, 47–49, 51, 57, 61–66, 71, 82, 86, 87, 89, 90

NTFS

New Technology File System es el sistema de archivos de Windows NT. 42

P

Perl

Lenguaje de programación diseñado por Larry Wall en 1987.. 55

PowerPC

nombre original de la arquitectura de computadoras de tipo RISC, que fue desarrollada por IBM, Motorola, y Apple.. 41

R**roll**

Paquete de software adicional al SO. 52–55, 58

S**SP**

Acronimo de *Service Pack*. Es un conjunto de programas informáticos que consisten en un grupo de actualizaciones que corrigen y mejoran aplicaciones y sistemas operativos.. 44

switch

Dispositivo para la interconexión de equipos en una misma red.. 5, 6, 16–18, 23, 47, 87

U**UNIX**

Sistema operativo portable, multitarea y multiusuario; desarrollado en 1969, por un grupo de empleados de los laboratorios Bell de AT& T, entre los que figuran Dennis Ritchie, Ken Thompson y Douglas McIlroy.. 29, 37, 85, 86, 89–91

Z

ZFS

Sistema de archivos y volúmenes desarrollado por Sun Microsystems para su sistema operativo Solaris.. 39

Bibliografía

- [Clarke and Lee,] Clarke, R. W. and Lee, B. T. B. Cluster operating systems. *School of Computer Science and Software Engineering Monash University Caulfield Campus, Melbourne, Australia*.
- [Redondo, 2015] Redondo, B. C. (2015). *Instalación, configuración y manejo de un Cluster Rocks*. Universidad de Valladolid.
- [Rial, 2012] Rial, S. M. (2012). *Instalación y configuración de un cluster de computación*. PhD thesis.
- [Spector, 2000] Spector, D. H. (2000). *Building LINUX CLUSTERS*. O'Reilly & Associates, Inc., 1 edition.
- [SPI, 2017] SPI (2017). Appi mpi. Fecha de consulta: julio 11, 2017 desde <https://www.open-mpi.org/doc/v2.1/>.
- [Wikipedia, 2015] Wikipedia (2015). Preboot execution environment. Fecha de consulta: julio 12, 2017 desde https://es.wikipedia.org/w/index.php?title=Preboot_Execution_Environment&oldid=87037255.
- [Wikipedia, 2017a] Wikipedia (2017a). Interfaz de paso de mensajes. Fecha de consulta: junio 24, 2017 desde https://es.wikipedia.org/w/index.php?title=Interfaz_de_Paso_de_Mensajes&oldid=99859645.

[Wikipedia, 2017b] Wikipedia (2017b). Sistema operativo. Fecha de consulta: julio 13, 2017 desde https://es.wikipedia.org/w/index.php?title=Sistema_operativo&oldid=99818722.