

TT-CIS-XA-003



UNIVERSIDAD
NACIONAL
DE LOJA



Área de la Energía, las Industrias y los Recursos Naturales No Renovables

CARRERA DE INGENIERÍA EN SISTEMAS

“Arquitectura clúster de alto rendimiento utilizando herramientas de software libre”

Tesis previa a la obtención del título de Ingeniero en Sistemas.

Autores:

- *Leonardo Rafael Chuquiguanca Vicente.*
- *Edyson Javier Malla Bustamante.*

Director:

Ing. Gabriela Viñan Rueda, Mg. Sc.

LOJA – ECUADOR

2015

Certificación de director

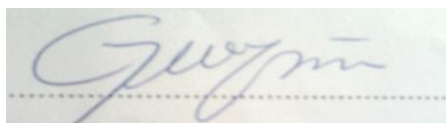
Ing. Gabriela Viñan Rueda, Mg. Sc.

**DIRECTORA DEL TRABAJO DE TITULACIÓN
DOCENTE DE LA CARRERA DE INGENIERÍA EN SISTEMAS DE LA UNIVERSIDAD
NACIONAL DE LOJA**

CERTIFICA:

Que los Egresados **Leonardo Rafael Chuquiguanca Vicente** y **Edyson Javier Malla Bustamante** desarrollaron cabalmente el trabajo de titulación denominado **“ARQUITECTURA CLÚSTER DE ALTO RENDIMIENTO UTILIZANDO HERRAMIENTAS DE SOFTWARE LIBRE”**, dicho proyecto cumple con los requisitos establecidos en las normas generales tanto en el aspecto de forma como de contenido bajo las sugerencias de mi dirección, supervisión y asesoramiento.

Loja, 05 de Enero de 2015.



Ing. Gabriela Viñan Rueda, Mg. Sc.

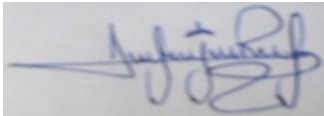
DIRECTORA DE TESIS

Autoría

Nosotros: **LEONARDO RAFAEL CHUQUIGUANCA VICENTE y EDYSON JAVIER MALLA BUSTAMANTE** declaramos ser autores del presente trabajo de tesis y eximimos expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos o acciones legales por el contenido de la misma.

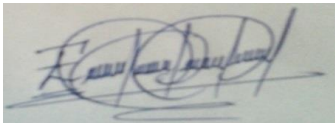
Adicionalmente acepto y autorizo a la Universidad Nacional de Loja, la publicación de nuestra tesis en el repositorio institucional – Biblioteca Virtual.

Autor: Leonardo Rafael Chuquiguanca Vicente

Firma: 

Cédula: 1104952708

Autor: Edyson Javier Malla Bustamante

Firma: 

Cédula: 1105035941

Fecha: 28 de Enero de 2015

CARTA DE AUTORIZACIÓN DE TESIS POR PARTE DE LOS AUTORES, PARA LA CONSULTA, REPRODUCCIÓN PARCIAL O TOTAL Y PUBLICACIÓN ELECTRÓNICA DEL TEXTO COMPLETO.

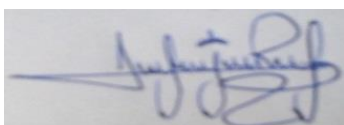
Nosotros: **LEONARDO RAFAEL CHUQUIGUANCA VICENTE y EDYSON JAVIER MALLA BUSTAMANTE**, declaramos ser autores de la tesis titulada: **ARQUITECTURA CLÚSTER DE ALTO RENDIMIENTO UTILIZANDO HERRAMIENTAS DE SOFTWARE LIBRE**, como requisito para optar al grado de: **INGENIERO EN SISTEMAS**; autorizo al Sistema Bibliotecario de la Universidad Nacional de Loja para que con fines académicos, muestre al mundo la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Digital Institucional:

Los usuarios puedan consultar el contenido de este trabajo en el RDI, en las redes de información del país y del exterior, con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia de la tesis que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, veinte y ocho días del mes de **enero** del dos mil quince.

Firma:



Autor: Leonardo Rafael Chuquiguanca Vicente

Cédula: 1104952708

Dirección: Loja, Diego Portales y Getulio Vargas

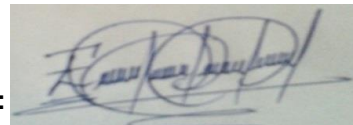
Correo electrónico:

lrchuquihuncav@unl.edu.ec

Teléfono: (07) 3033 759

Celular: 0992438768

Firma:



Autor: Edyson Javier Malla Bustamante

Cédula: 1105035941

Dirección: Loja, Av. Benjamín Carrión y Ontario esquina

Correo electrónico:

ejmallab@unl.edu.ec

Teléfono: (07) 2110 538

Celular: 0969196687

DATOS COMPLEMENTARIOS

Director de Tesis: Ing. Gabriela Viñan Rueda, Mg. Sc.

Tribunal de Grado: Ing. Alex Vinicio Padilla Encalada, Mgs.

Ing. Luis Roberto Jácome Galarza, Mg. Sc.

Ing. Lorena Elizabeth Conde Zhingre, Mg. Sc.

Agradecimiento

A la Universidad Nacional de Loja que nos acogió para formarnos como profesionales y a la Carrera de Ingeniería en Sistemas, porque en sus aulas, recibimos el conocimiento intelectual y humano de cada uno de los docentes, la misma que nos brindó la oportunidad y facilidad para llevar con éxito nuestro Trabajo de Titulación.

A nuestra directora de Tesis Ing. Gabriela Viñan Rueda, por su presencia incondicional, sus apreciados y relevantes aportes, críticas, comentarios y sugerencias durante el desarrollo de este proyecto.

A los Ing. Freddy Patricio Ajila Zaquinaula, Ing. René Guamán Quinche, por su apoyo al inicio del Trabajo de Titulación, por la revisión, sugerencias y críticas constructivas en lo que se refiere a la viabilidad del proyecto.

Dedicatoria

Dedico este trabajo de titulación principalmente a Dios, por derramar sus bendiciones sobre mí, darme salud bienestar y fuerza para seguir siempre adelante y no desfallecer en el intento.

A mis padres quienes me dieron la vida, educación, apoyo y consejos para poder llegar a esta instancia de mis estudios y una dedicación especialmente a mi madrecita Domitila ya que gracias a su esfuerzo y sacrificio he logrado salir adelante en todo este proceso, superando todas las adversidades y obstáculos que se me han presentado.

A mis hermanitas Marías, porque cada día llenan de alegría mi vida y por eso quiero decirles que el sacrificio y entrega vale la pena.

A mi compañero de tesis Edyson, porque gracias a su apoyo hemos podido sobresalir en cada una de las dificultades que se nos han presentado durante todo el proceso de desarrollo del trabajo de titulación logrando así alcanzar un objetivo en común.

Leonardo Rafael

Dedico este trabajo principalmente a Dios, por haberme dado la vida y permitirme haber llegado hasta este momento tan importante de mi formación profesional. A mis padres, Mercedes y Eduardo que con su demostración ejemplar me ha enseñado a no desfallecer ni rendirme ante nada y siempre perseverar a través de sus sabios consejos.

A mis hermanos Byron, Cristian, Elizabeth por su carisma y apoyo incondicional, a todos mis familiares por ser las personas que confiaron siempre en mis capacidades de superación. A mi compañero de proyecto Leonardo, que gracias a su esfuerzo se logró esta meta en nuestra formación profesional.

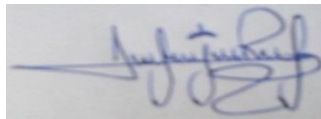
En especial a mi hijo Mateo Javier quien es y será siempre mi inspiración y fortaleza en mi vida profesional.

Edyson Javier

Cesión de derechos

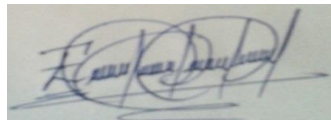
Nosotros: **Leonardo Rafael Chuquiguanca Vicente** y **Edyson Javier Malla Bustamante**, declaramos ser autores intelectuales del presente Trabajo de Titulación y autorizamos a la Universidad Nacional de Loja, al Área de Energía, las Industrias y los Recursos Naturales No Renovables y por ende a la Carrera de Ingeniería en Sistemas a hacer uso del mismo en lo que estime sea conveniente.

Para constancia firmamos a continuación.



Leonardo Rafael Chuquiguanca Vicente

CI: 1104952708



Edyson Javier Malla Bustamante

CI: 1105035941

a. Título

**“ARQUITECTURA CLÚSTER DE ALTO RENDIMIENTO UTILIZANDO
HERRAMIENTAS DE SOFTWARE LIBRE”**

b. Resumen

En los últimos años la ciencia y la tecnología han tenido grandes avances en lo que respecta a la computación de alto rendimiento, enmarcados en arquitecturas diseñadas especialmente para su funcionamiento y caracterizadas por su elevado coste, siendo inalcanzables por centros de investigación e Instituciones de Educación Superior.

La informática para investigaciones, usa potentes herramientas y métodos para procesar datos en investigaciones académicas avanzadas. Con un clúster construido por hardware convencional y herramientas de software libre, la Carrera de Ingeniería en Sistemas de la Universidad Nacional Loja, puede obtener una arquitectura con la velocidad y potencia de una supercomputadora, a una fracción del costo mínimo.

La finalidad de este proyecto es aportar e incentivar a la investigación académica y científica de la UNL, optimizando el tiempo de procesamiento en tareas como: simulación de procesos, aprendizaje de redes neuronales, renderizado de imagen y video, reconocimiento de patrones, análisis de grandes volúmenes de datos Big Data, cifrado de códigos, evaluación de algoritmos, etc. La implementación de la arquitectura clúster de alto rendimiento permite realizar millones de operaciones por segundo, logrando avances importantes en la indagación académica, en cualquier campo de la investigación dentro del ámbito universitario.

La implementación de la arquitectura clúster se la realizó con el sistema operativo GNU/Linux Debian "Wheezy" por su estabilidad y su funcionamiento como servidor de alto rendimiento, usado como estándar para el desarrollo del proyecto. Para el middleware se utilizó Mosix como gestor de procesos para obtener una imagen única de sistema orientada a la computación distribuida. Para la programación paralela se implementó MPICH como una de las aplicaciones más importantes del estándar Message Passing Interface. Para el proceso de renderizado se usó Blender, para análisis de Big Data se empleó el framework Apache Hadoop. Dentro de las configuraciones realizadas tenemos: el acceso remoto utilizando claves públicas a través del protocolo Secure Shell, la compartición del sistema de ficheros en red a través del protocolo Network File System, además del diseño de la topología de red con direccionamiento IP estático y la configuración del sistema de monitorización Ganglia Monitoring System.

Summary

In recent years science and technology have made great advances with respect to the high performance computing, framed in architectures specifically designed for their operation and characterized by its high cost, being unreachable by research centers and higher education institutions (IES).

Computing for researches use powerful tools and methods to process data in advanced academic researches. With a cluster built by conventional hardware and free software tools, the Systems Engineering Degree of National University Loja can obtain architecture with the speed and power of an expensive supercomputer at a fraction of the minimum cost.

The purpose of this project is to provide and encourage academic and scientific research, teachers and students from the CIS, obtaining good results in: simulation process, neural network learning, pictures processing, patterns recognition, database processing, processing of large volumes of Big Data information, data mining, decryption of codes, assessment of algorithms, etc. The implementation of a platform of higher yield allows made billions of operations per second, allowing significant advances into the academic inquiry in any field of research into the university area.

The implementation of the cluster was done with the GNU / Linux Debian "Wheezy" operating system, by its stability and efficiency as a server of high-performance used like a standard for the development of this project. For middleware we used Mosix in version 3.4.0.12 like a manager of process to get a unique image of the system oriented to distribute computing. MPICH was used in version 3.1.2 for parallel programming as one of the standard applications deployed Message Passing Interface (MPI). For the rendering process we used Blender, for the analysis of Big Data we used the Apache Hadoop framework. Among the configurations that we made they were: Remote access using public keys through Secure Shell protocol (SSH), sharing of network system through the Network File System protocol (NFS), also the design of network topology and the configuration of the motorization system Ganglia Monitoring System.

Índice de Contenidos

Índice General

Contenido

| | |
|---|------|
| Certificación de director | II |
| Autoría..... | III |
| CARTA DE AUTORIZACIÓN DE TESIS POR PARTE DE LOS AUTORES, PARA LA CONSULTA, REPRODUCCIÓN PARCIAL O TOTAL Y PUBLICACIÓN ELECTRÓNICA DEL TEXTO COMPLETO..... | IV |
| Agradecimiento..... | V |
| Dedicatoria | VI |
| Cesión de derechos..... | VIII |
| a. Título | IX |
| b. Resumen..... | X |
| Summary | XI |
| Índice de Contenidos | XII |
| Índice General..... | XII |
| Índice de Figuras..... | XVII |
| Índice de Tablas..... | XX |
| c. Introducción..... | 21 |
| d. Revisión de Literatura..... | 23 |
| 1. Introducción a los clúster | 23 |
| 1.1 ¿Qué es un clúster? | 23 |
| 1.1 Características de un clúster | 23 |
| 1.1.1 Clasificación de clúster..... | 24 |

| | |
|---|----|
| Clúster homogéneo: | 24 |
| Clúster heterogéneo: | 24 |
| 1.1.2 Componentes de clúster..... | 24 |
| 1.2.3 Tipos de clúster | 26 |
| 1.3 Clúster de alto rendimiento (High Performance Cluster)..... | 26 |
| 2. Cluster Operating System (Mosix) | 27 |
| 2.1 Definición | 27 |
| 2.2 Características principales de un clúster Mosix | 28 |
| 2.3 Requerimientos del sistema [10] | 28 |
| 3. Programación paralela..... | 29 |
| 3.1 Definición..... | 29 |
| 3.2 Importancia de la programación paralela en la computación de alto rendimiento | 29 |
| 4. Big data | 32 |
| 4.1. Definición | 32 |
| 4.2. Tecnologías para trabajar con Big Data | 32 |
| 4.3. Apache Hadoop | 33 |
| 4.3.1 Análisis de datos con Hadoop en modo clúster | 33 |
| 4.3.2. Módulos Hadoop | 33 |
| 4.3.3. Arquitectura Hadoop | 34 |
| 5. Blender | 35 |
| 5.1 Render distribuido de imagen y video sobre clúster computing | 35 |
| e. Materiales y Métodos..... | 36 |
| Metodología del Trabajo de Titulación..... | 37 |
| f. Resultados | 40 |
| 1. Fase 1:..... | 41 |

| | | |
|-------|---|----|
| 1.1 | Revisión bibliográfica y casos de éxito del funcionamiento de los clúster en centros de investigación y en instituciones de educación superior (IES)..... | 41 |
| 1.2 | Análisis de los recursos de los equipos informáticos con los que cuenta actualmente la institución..... | 45 |
| 1.2.1 | Análisis de la situación actual del laboratorio CIS-UNL..... | 45 |
| 1.2.2 | Recursos Hardware: | 48 |
| 1.2.2 | Recursos Software: | 48 |
| 1.2.3 | Interconexión de Red:..... | 49 |
| 1.3 | Seleccionar la plataforma de software libre que más se adapte a los requerimientos de los equipos informáticos existentes | 50 |
| 1.3.3 | Arquitectura clúster | 56 |
| 2. | Fase 2:..... | 58 |
| 2.1 | Diseño lógico de la arquitectura clúster en el laboratorio de la CIS | 58 |
| 2.1.1 | Arquitectura | 58 |
| 2.1.2 | Topología | 59 |
| 2.2 | Instalación y configuración del sistema operativo del clúster..... | 61 |
| 2.3 | Instalación y configuración de los servicios del clúster en el nodo maestro y los nodos esclavos | 61 |
| 3. | Fase 3:..... | 62 |
| 3.1 | Preparar el escenario para realizar las pruebas de funcionalidad | 62 |
| 3.2 | . Evaluar los resultados en base al escenario planteado mediante procesos que impliquen grandes recursos computacionales | 65 |
| 3.3 | Elaborar un artículo científico acorde a las normas IEEE | 71 |
| g. | Discusión..... | 81 |
| 1. | Desarrollo de la propuesta alternativa | 81 |
| 1.1 | Analizar las plataformas clúster para determinar cuál es la más óptima de acuerdo a los equipos y requerimientos existentes..... | 81 |

| | |
|--|-----|
| 1. Valoración técnica, económica, ambiental..... | 83 |
| h. Conclusiones..... | 87 |
| i. Recomendaciones..... | 88 |
| j. Bibliografía..... | 89 |
| Referencias Bibliográficas..... | 89 |
| k. Anexos..... | 93 |
| Anexo 1: Anteproyecto Trabajo de Titulación..... | 93 |
| A. Tema:..... | 98 |
| B. Problemática:..... | 98 |
| C. Justificación..... | 100 |
| D. Objetivos..... | 101 |
| E. Alcance..... | 101 |
| F. Metodología..... | 103 |
| G. Cronograma de actividades..... | 104 |
| H. Presupuesto y financiamiento..... | 105 |
| I. Bibliografía..... | 106 |
| J. Anexos..... | 108 |
| Anexo 2: Testeo de rendimiento con Stress-Test..... | 109 |
| 2.2. Prueba test distkeygen..... | 111 |
| Anexo 3: Cálculo del valor aproximado de π utilizando MPI..... | 114 |
| Anexo 4: Pruebas de compresión de música con Mosix..... | 118 |
| Anexo 5: Instalación y configuración de John the Ripper (JTR)..... | 122 |
| 5.1 Cifrado de códigos con John The Ripper (JTR) y MPI..... | 122 |
| 4.1 Instalación y configuración JTR [48]..... | 122 |
| 4.2 Ejecutando John the Ripper utilizando MPI..... | 125 |
| Anexo 6. Prueba de cifrado de códigos con MPICH-JTR..... | 126 |

| | |
|---|-----|
| Anexo 7: Renderización de imágenes y videos con blender..... | 129 |
| Anexo 8: Análisis de datos big data con hadoop | 131 |
| 8.1. Copiar el fichero en el HDFS..... | 131 |
| 8.2. Ejecución del programa desarrollado con librerías hadoop | 132 |
| 8.3. Resultado del análisis de datos..... | 132 |
| Anexo 9: Certificación Summary | 133 |
| Anexo 10: Certificación Revisión Literaria | 134 |
| Anexo 11: Licencia Creative Commons..... | 135 |

Índice de Figuras

| | |
|---|-----|
| Figura 1. Clúster de computadoras con hardware convencional. | 23 |
| Figura 2. Diagrama esquemático componentes de un clúster..... | 25 |
| Figura 3. Mapa conceptual sobre programación paralela..... | 31 |
| Figura 4. Arquitectura del proyecto Apache Hadoop [24]. | 34 |
| Figura 5. Distribución de los equipos en el laboratorio de la CIS-UNL. | 46 |
| Figura 6. Vista 3D laboratorio CIS-UNL. | 47 |
| Figura 7. Arquitectura clúster de alto rendimiento CIS-UNL..... | 58 |
| Figura 8. Diagrama de topología propuesto. | 59 |
| Figura 9. Tiempo de ejecución testeo de rendimiento con Stress-Test. | 65 |
| Figura 10. Tiempos de ejecución cálculo del valor aproximado de π utilizando MPI. ... | 66 |
| Figura 11. Tiempo de ejecución de pruebas de compresión de música con Mosix. | 67 |
| Figura 12. Tiempo de ejecución de cifrado de código utilizando John The Ripper. | 68 |
| Figura 13. Tiempo de ejecución de renderización de imágenes y videos con blender. | 69 |
| Figura 14. Tiempo de ejecución de análisis de datos (Big Data) con Hadoop. | 70 |
| Figura 15. Cronograma de actividades. | 104 |
| Figura 16. Mover stress-test al directorio /usr/local/. | 109 |
| Figura 17. Ingresamos al directorio /usr/local/. | 109 |
| Figura 18. Descompresión del archivo stress-test en el directorio /usr/local/. | 110 |
| Figura 19. Ingresamos al directorio omtest | 110 |
| Figura 20. Compilación de stress-test mosix..... | 110 |
| Figura 21. Compilación de stress-test finalizada con éxito..... | 111 |
| Figura 22. Iniciando stress-test con mosix. | 111 |
| Figura 23. Archivos de la aplicación test distkeygen. | 112 |
| Figura 24. Tiempo de ejecución de la aplicación distkeygen nodo maestro. | 112 |
| Figura 25. Tiempo de ejecución de la aplicación distkeygen recursos clúster..... | 113 |

| | |
|---|-----|
| Figura 26. Claves RSA generadas por la aplicación test distkeygen..... | 113 |
| Figura 27. Ejecución de MPICH para calcular el valor de pi desde el nodo maestro. | 114 |
| Figura 28. Resultado del tiempo de ejecución para el cálculo de valor de pi en el nodo maestro. | 114 |
| Figura 29. Archivo machinefile que contiene el número de procesadores de cada nodo. | 115 |
| Figura 30. Ejecución de MPICH para el cálculo del valor de pi. | 115 |
| Figura 31. Resultado de la ejecución del proyecto 1 utilizando 5 nodos. | 116 |
| Figura 32. Resultado de la ejecución del proyecto 1 utilizando 10 nodos. | 116 |
| Figura 33. Resultado de la ejecución del proyecto 1 utilizando 15 nodos. | 116 |
| Figura 34. Listado de canciones a comprimir con Mosix. | 118 |
| Figura 35. Tamaño de los ficheros antes de la compresión. | 118 |
| Figura 36. Script para la compresión de los ficheros .wav | 119 |
| Figura 37. Ejecución del script para la compresión de música con Mosix utilizando los recursos del nodo maestro. | 119 |
| Figura 38. Resultado de la aplicación utilizando los recursos del nodo maestro. | 120 |
| Figura 39. Ejecución del script para la compresión de música con Mosix. | 120 |
| Figura 40. Resultado de la aplicación utilizando los recursos de los 5 nodos de la arquitectura clúster. | 120 |
| Figura 41. Resultado de la aplicación utilizando los recursos de los 10 nodos de la arquitectura clúster. | 121 |
| Figura 42. Resultado de la aplicación utilizando los recursos de los 15 nodos de la arquitectura clúster. | 121 |
| Figura 43. Tamaño de los ficheros luego del proceso de compresión utilizando Mosix. | 121 |
| Figura 44. Descompresión del programa John the Ripper. | 123 |
| Figura 45. Cambio de directorio de instalación de John the Ripper. | 123 |
| Figura 46. Ingreso al directorio de John the Ripper..... | 123 |

| | |
|--|-----|
| Figura 47. Edición del fichero makefile para habilitar la configuración de John The Ripper..... | 124 |
| Figura 48. Compilación de John the Ripper utilizando librerías MPI..... | 124 |
| Figura 49. Prueba de testeo de John the Ripper..... | 125 |
| Figura 50. Ejecución del test de JTR con MPI. | 125 |
| Figura 51. Descifrando el fichero shadow con John The Ripper – MPI. | 125 |
| Figura 52. Tiempo aproximado de descifrado del password. | 126 |
| Figura 53. Descifrado de contraseña con JTR utilizando el nodo maestro..... | 127 |
| Figura 54. Descifrado de código con JTR, utilizando 5 nodos..... | 127 |
| Figura 55. Descifrado de contraseña con JTR, utilizando 10 nodos..... | 127 |
| Figura 56. Descifrado de código con JTR, utilizando 15 nodos..... | 128 |
| Figura 57. Balanceo de carga en el clúster al ejecutar MPICH+JTR..... | 128 |
| Figura 58. Proceso de renderización de imágenes y vídeos con blender..... | 129 |
| Figura 59. Proceso de renderizado de imágenes y video con blender utilizando los recursos del clúster..... | 130 |
| Figura 60. Proceso de renderizado finalizado. | 130 |
| Figura 61. Distribución de bloques de tamaño fijo en los nodos del clúster..... | 131 |
| Figura 62. Resultado del análisis de datos sobre la calidad del aire. | 132 |

Índice de Tablas

| | |
|---|-----|
| TABLA I. PARTICIONAMIENTO LÓGICO SISTEMAS OPERATIVOS LABORATORIO CIS-UNL..... | 48 |
| TABLA II. COMPARATIVA ENTRE SISTEMAS OPERATIVOS GNU/LINUX..... | 50 |
| TABLA III. COMPARATIVA ENTRE SISTEMAS MIDDLEWARE CLÚSTER..... | 52 |
| Tabla IV. COMPARATIVA ENTRE ARQUITECTURAS CLÚSTER..... | 56 |
| TABLA V. DIRECCIONAMIENTO IP CLÚSTER CIS-UNL..... | 60 |
| TABLA VI. RESULTADO DE LA EJECUCIÓN DE PRUEBAS DE TESTEO DE RENDIMIENTO CON STRESS-TEST. | 65 |
| TABLA VII. RESULTADO DE LA EJECUCIÓN DEL PROCESO DE CÁLCULO DEL VALOR APROXIMADO DE π UTILIZANDO MPI..... | 66 |
| TABLA VIII. RESULTADO DE LA EJECUCIÓN, PRUEBAS DE COMPRESIÓN DE MÚSICA CON MOSIX | 67 |
| TABLA IX. RESULTADO DE LA EJECUCIÓN DE CIFRADO DE CÓDIGO UTILIZANDO JOHN THE RIPPER. | 68 |
| TABLA X. RESULTADO DE LA EJECUCIÓN DE RENDERIZACIÓN DE IMÁGENES Y VIDEOS CON BLENDER. | 69 |
| TABLA XI. RESULTADO DE LA EJECUCIÓN DE ANÁLISIS DE DATOS (BIG DATA) CON HADOOP. | 70 |
| TABLA XII. RECURSOS HUMANOS..... | 84 |
| TABLA XIII. RECURSOS MATERIALES. | 84 |
| TABLA XIV. RECURSOS HARDWARE..... | 85 |
| TABLA XV. RECURSOS SOFTWARE. | 85 |
| TABLA XVI. RECURSOS TÉCNICOS Y TECNOLÓGICOS..... | 86 |
| TABLA XVII. APROXIMACIÓN DEL COSTO REAL DEL TT. | 86 |
| TABLA XVIII. PRESUPUESTO ESTIMADO TT..... | 105 |
| TABLA XIX. CÓDIGO FUENTE VALOR DE PI..... | 116 |

c. Introducción

Multitud de aplicaciones dentro de la investigación científica como: simulación de fenómenos físicos o meteorológicos, reconocimiento de patrones, procesamiento de consultas de base de datos, procesamiento de grandes volúmenes de información como Big Data, minería de datos, aprendizaje de redes neuronales entre otros, tienen una creciente demanda de potencia de cómputo.

En la actualidad, dicha demanda está cubierta en su mayoría por supercomputadores, con costes muy elevados inalcanzables para las Instituciones de Educación Superior (IES) y centros de investigación, cuyo fin es realizar investigaciones académicas y científicas.

Una de las soluciones más fiables para ejecutar cálculos científicos, es la implementación de un clúster de alto rendimiento formado por hardware convencional y herramientas de software libre, que unidos a una red de alta velocidad puede proporcionar un ahorro económico importante y resultados con un margen de error mínimo en tiempos relativamente bajos, de manera que los centros de investigación y las IES pueden obtener la velocidad y potencia de una costosa supercomputadora a una fracción de coste mínimo.

En relación a lo descrito el presente Trabajo de Titulación (TT) cuyo entorno es la implementación de un clúster de alto rendimiento utilizando hardware convencional y software libre, el mismo que se desarrolló en el laboratorio de la Carrera de Ingeniería en Sistemas (CIS) de la Universidad Nacional de Loja, con la finalidad de tener un centro de investigación científica que ayude a docentes y estudiantes de la CIS en la realización de futuras investigaciones.

El proyecto inicia con la revisión bibliográfica de casos de éxito de la implementación de clústers de alto rendimiento en centros de investigación y en IES, en el Ecuador y a nivel internacional, que sirvió como punto de partida para la elección de las herramientas de software libre a utilizar para la implementación del clúster y la arquitectura hardware del mismo; luego se realizó el análisis de los recursos informáticos hardware y software con los que cuenta actualmente el laboratorio de la CIS, finalmente en esta primera parte se efectuó la elección de las herramientas de software libre que se adapten a los recursos existentes en el laboratorio de la CIS.

Luego se desarrolló el diseño lógico de la arquitectura, además la instalación y configuración del sistema operativo Debian 7.5 “Wheezy”, el middleware Mosix en su versión 3.4.0.12 y los aplicativos relacionados a la programación paralela MPICH, para el proceso de renderizado se utilizó el software Blender, mientras que para el análisis de Big Data se empleó el framework Apache Hadoop. Para la evaluación del funcionamiento del clúster de alto rendimiento CLUSTER_CIS-UNL se realizaron pruebas de funcionamiento que involucran gran cantidad de recursos computacionales.

Los lineamientos establecidos por la Universidad Nacional de Loja y el Área de la Energía, las Industrias y los Recursos Naturales No Renovables rigen la estructura de éste Trabajo de Titulación, el cual tiene el siguiente orden: RESUMEN contiene un extracto del contenido general del Trabajo de Titulación, ÍNDICE describe la ubicación de los temas tratados con sus respectivas figuras, tablas indicando el número de página a la que pertenece, INTRODUCCIÓN relata lo relevante que es el tema y su aplicabilidad en la investigación científica, METODOLOGÍA se realiza una descripción de los principales materiales empleados y métodos de investigación tanto científicos, experimentales y técnicas investigativas, REVISIÓN DE LA LITERATURA, que comprende la sustentación teórica de las temáticas que ayudan a la comprensión del Trabajo de Titulación, RESULTADOS tiene como propósito la evaluación y el cumplimiento de los objetivos planteados, además de presentar la evaluación técnica, económica y ambiental de la investigación realizada, DISCUSIÓN aquí se explica el uso de los métodos y técnicas utilizadas, CONCLUSIONES detalla las ideas a las que llegamos, el proyecto finaliza con las RECOMENDACIONES, BIBLIOGRAFÍA y ANEXOS.

d. Revisión de Literatura

1. Introducción a los clúster

1.1 ¿Qué es un clúster?

Un clúster es un conjunto de nodos de bajo costo conectados entre sí a través de una red de comunicaciones de alta velocidad, que opera bajo software que actúa como un sistema único de administración, responsable de distribuir las cargas de trabajo entre los nodos, de forma automática y transparente al usuario como si se tratará de un único ordenador [1], [2], [3].

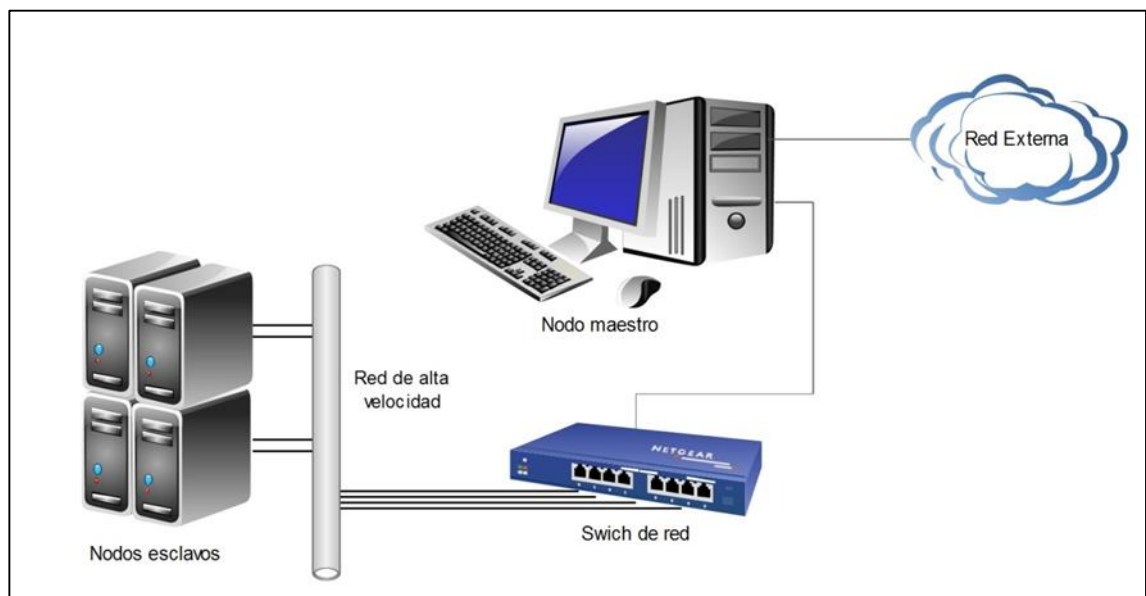


Figura 1. Clúster de computadoras con hardware convencional.

1.1 Características de un clúster

De manera lógica, cada nodo del clúster está formado hardware y software. El hardware está compuesto por el mainboard, procesador, memoria, interfaz de red, disco duro, periféricos de entrada-salida, entre otros elementos. En cuanto al software, el nivel bajo corresponde al sistema operativo, el nivel medio consiste en las librerías de paralelización y el nivel alto está representado por la aplicación que se desea ejecutar;

una aplicación se ejecuta en el nodo maestro, el sistema operativo y las librerías de paralelización se encargan de ejecutar copias de este programa en los nodos esclavos del clúster[2].

1.1.1 Clasificación de clúster

Existen dos tipos de clúster según la arquitectura de las computadoras que lo conforman [4], [5]:

Clúster homogéneo: son clúster en que todos los nodos tienen las mismas características de hardware y software. Son idénticos y por lo tanto la capacidad de procesamiento y rendimiento de cada nodo es la misma.

Clúster heterogéneo: al contrario de los clúster homogéneos, los nodos son completamente distintos en cuanto a hardware y software. Esto conlleva a que las posibilidades de expansión del clúster crezcan de forma exponencial, debido a que es más fácil conseguir ordenadores con características distintas que con características similares.

1.1.2 Componentes de clúster

Un clúster necesita componentes software y hardware para poder desempeñar sus funciones; entre los componentes están: nodos, tipo de almacenamiento, sistema operativo, conexiones de red, middleware; en la Fig. 2 (*Diagrama esquemático componentes de un clúster*), se ilustra los componentes que conforman una arquitectura clúster [4], [5].

- **Nodos:** pueden ser simples ordenadores, sistemas multi-procesador o estaciones de trabajo. El clúster puede estar conformado por nodos dedicados o por nodos no dedicados. En un clúster con nodos dedicados, los nodos no disponen de teclado, ratón, ni monitor y su uso esta exclusivamente dedicado a realizar tareas relacionadas con el clúster, mientras que en un clúster con nodos no dedicados, los nodos disponen de teclado, ratón y monitor y su uso no está exclusivamente dedicado a realizar tareas relacionadas con el clúster.
- **Almacenamiento:** el almacenamiento puede consistir en un sistema NAS (Network Attached Storage), un SAN (Storage Area Network), o almacenamiento interno en el servidor. El protocolo más comúnmente utilizado es NFS (Network

File System), que permite compartir el sistema de ficheros entre el servidor y los nodos esclavos.

- **Sistema operativo:** es el software destinado que permite una gestión eficaz de los recursos. En este caso debe ser multiproceso y multiusuario.
- **Conexiones de red:** los nodos de un clúster pueden conectarse mediante una simple red Ethernet con placas comunes (adaptadores de red o NIC), o utilizar tecnologías especiales de alta velocidad como Fast Ethernet, Gigabit Ethernet, Myrinet, InfiniBand, etc.
- **Middleware:** software que generalmente actúa entre el sistema operativo y las aplicaciones con la finalidad de proveer una interfaz única de acceso al sistema, SSI (Single System Image), la cual genera la sensación al usuario de que utiliza un único ordenador potente, posee herramientas para la optimización y mantenimiento del sistema, migración de procesos, balanceo de carga, tolerancia a fallos, entre otras, además de escalabilidad para detectar automáticamente nuevos nodos conectados al clúster para proceder a su utilización.

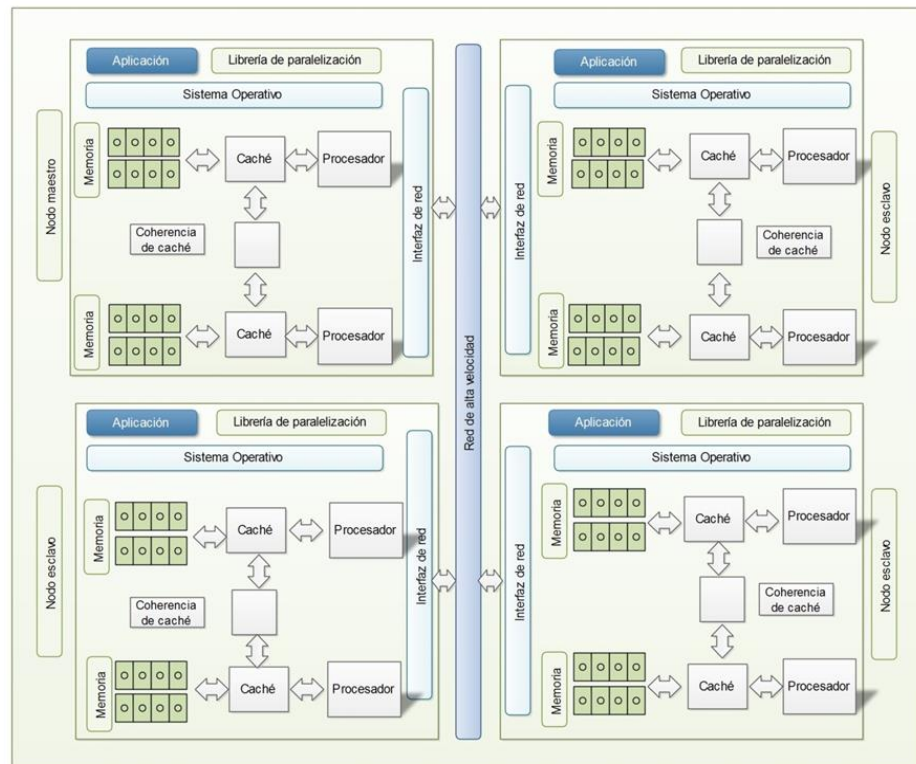


Figura 2. Diagrama esquemático componentes de un clúster.

1.2.3 Tipos de clúster

Los clúster dependiendo de su aplicabilidad pueden clasificarse de diferentes maneras. La clasificación más generalizada es la que se presenta a continuación [6], [7]:

- Alto rendimiento (HP, high performance)
- Alta disponibilidad (HA, high availability)
- Balanceo de carga (Load balancing)
- Alta confiabilidad (HR, high reliability).

En función de las características mencionadas y los tipos de clúster derivados de ellas, podemos encontrar distintas familias de clúster que utilizan una serie de herramientas por ejemplo:

Clúster beowulf: es un tipo de clúster construido bajo Linux con hardware convencional, conectado mediante una red local Ethernet, cuyo objetivo es conseguir alto rendimiento. Para su implementación se utiliza la computación paralela para las aplicaciones paralelizadas, es decir se utilizan librerías de paso de mensajes PVM (Máquina Virtual Paralela) y PMI (Interfaz de Paso de Mensajes) para que los procesos se ejecuten en múltiples procesadores siguiendo el paradigma cliente/servidor. Están destinados a centros de cálculo, cuyo objetivo principal es la computación de alto rendimiento.

Clúster alta disponibilidad con LVS (Linux Virtual Server): utilizados principalmente en granjas de servidores en donde se deben trabajar conjuntamente varias piezas. Consta de los servidores reales que prestan el servicio a los clientes para asegurar un servicio 24 x 7, balanceador de carga virtual es el encargado de repartir las peticiones entre los servidores reales y los datos.

Clúster SSI: todas las computadoras vinculadas dependen de un sistema operativo común, diseñado a tal efecto, significa que el usuario tiene una visión global de los recursos disponibles independientemente del nodo al que están ligados esos recursos.

1.3 Clúster de alto rendimiento (High Performance Cluster)

Son clúster destinados a ofrecer un alto rendimiento paralelizando las tareas. Las mismas son divididas y son enviadas a muchos nodos de proceso. Son utilizados principalmente con fines científicos. Generalmente estos clúster se implantan en sistemas bastantes cerrados y las aplicaciones ejecutadas están ligadas

específicamente con la arquitectura, intentando aprovechar al máximo los recursos disponibles [8].

En este tipo de clúster el objetivo principal es proporcionar altas prestaciones de capacidad de cómputo superior a las que pudiera ofrecer un ordenador común.

Se utiliza principalmente en aplicaciones que realizan cálculos intensivos: simulaciones científicas, renderización de gráficos, compilación de programas, comprensión de datos, descifrado de datos, rendimiento del sistema operativo, modelos meteorológicos, entre otras aplicaciones. Pretenden sustituir a grandes y costosas supercomputadoras [9].

Éstas son sólo algunas de las aplicaciones que se pueden funcionar actualmente en un clúster. Cada vez son más las grandes empresas que demandan este tipo de sistemas, por lo que todo lo relacionado con ellos está en primera línea de investigación.

Utiliza una red privada de alta velocidad. Donde los nodos están dedicados exclusivamente al clúster. Existe un nodo maestro que se encarga de acceder, compilar y manejar las aplicaciones a ejecutar. Normalmente es el único con salida al exterior.

2. Cluster Operating System (Mosix)

2.1 Definición

Mosix es un software que transforma un conjunto de computadoras conectadas en red bajo un sistema operativo GNU/Linux en un clúster. Mosix equilibra la carga automáticamente entre los diferentes nodos del clúster y los nodos pueden unirse o dejar el clúster sin interrumpir el servicio [8].

Mosix es una extensión del Kernel de Linux (Linux Kernel Patch) el cual está dirigido a la computación de alto rendimiento, esto permite ejecutar aplicaciones normales (no paralelizadas) en un clúster. Una de las principales utilidades de mosix es la migración de procesos, un proceso puede iniciar en un nodo y posteriormente migrar y ejecutarse de manera transparente en otros nodos [10].

El proceso de migración se realiza para optimizar el rendimiento global. La migración de un proceso se produce de forma automática y transparente en respuesta a la disponibilidad de los recursos. Este proceso de migración se repite según sea necesario

entre los distintos nodos que componen el clúster, incluso puede volver al nodo que inició el proceso si las condiciones lo permiten [10].

2.2 Características principales de un clúster Mosix

Las características de Mosix están destinadas a proporcionar a los usuarios y a las aplicaciones la impresión de que se ejecutan las tareas en un único ordenador (nodo) con varios procesadores [10], [11].

- Proporciona una imagen única del sistema (SSI).
 - Los usuarios pueden iniciar sesión en cualquier nodo y no necesitan saber dónde se están ejecutando sus aplicaciones.
 - No es necesario modificar las aplicaciones, ni enlazarlas a ninguna biblioteca en especial.
 - No es necesario copiar los archivos en los nodos remotos.
- Detección automática de los recursos y distribución de la carga de trabajo.
 - Balanceo automático entre los procesos que han migrado.
 - Migración de procesos desde los más lentos a los nodos más rápidos.
 - Migración de procesos que necesitan más memoria de la disponible en un nodo.
- Comunicación directa entre los procesos migrados.
- Proporciona un entorno de tiempo de ejecución seguro para los procesos huésped.
- Compatible con arquitecturas de x64 y x86.
- Incluye herramientas para la instalación y configuración de manera automática.
- Incluye monitoreo en línea.

En un clúster mosix se puede ejecutar las aplicaciones desde cualquier nodo pero normalmente se lo debe realizar desde un nodo principal (nodo maestro), en el cual se encuentran instaladas las diferentes herramientas de administración y monitoreo, para la correcta administración del clúster y así los usuarios se puedan logear y ejecutar las distintas aplicaciones de una manera más fácil.

2.3 Requerimientos del sistema [10]

- Soporta equipos con arquitectura de x64 y x86.
- Todos los núcleos de un nodo deben tener la misma velocidad.

- Todos los nodos del clúster deben estar conectados a una red que soporte los protocolos TCP/IP y UDP/IP.
- Cada nodo deber tener una dirección IP única en el rango 0.1.0.0 a 255.255.254.255 que sea accesible por el resto de nodos.
- Los puertos TCP/IP 250, 252, 253 y los puertos UDP/IP 249, 250, 253 deben ser reservados para Mosix y no deben ser utilizados por otras aplicaciones o bloqueados por el firewall.
- Mosix puede ser instalado en cualquier distribución GNU/Linux siempre y cuando el kernel de Linux permita la compilación con la versión de Mosix.

3. Programación paralela

3.1 Definición

“La programación paralela es la disciplina que se encarga del estudio de las notaciones que permiten especificar la ejecución concurrente de las acciones de un programa, así como de las técnicas para resolver los problemas inherentes a la ejecución concurrente que son básicamente comunicación y sincronización” [12].

La programación paralela o concurrente es aquella que tiene más de un proceso en marcha a la vez, y un programa en paralelo es el que se ejecuta en varios procesadores a la vez [13].

3.2 Importancia de la programación paralela en la computación de alto rendimiento

La necesidad de la programación paralela se origina por las limitaciones de las aplicaciones secuenciales; integrando varios procesadores para llevar a cabo sus funciones, la programación paralela permite resolver problemas que requieren de más memoria o de mayor velocidad de cómputo. También existen razones económicas, pues el precio de los ordenadores secuenciales no es proporcional a su capacidad computacional [14], mientras que la conexión de varios procesadores utilizando una red nos permite obtener un aumento de prestaciones prácticamente proporcional al número de procesadores con un coste adicional mínimo.

La computación paralela se utiliza para reducir el tiempo de resolución de problemas computacionales, o bien para resolver problemas que no cabrían en la memoria de un solo ordenador. Y para esto es necesario utilizar sistemas de altas prestaciones y algoritmos paralelos que utilicen estos sistemas de manera eficiente [15].

Su ventaja reside en todas las aplicaciones ya desarrolladas para portarlas sin esfuerzo a entornos reconfigurables y obtener aceleración inmediata [14].

Los problemas comunes que abordan la programación paralela son: problemas de alto coste, o problemas de no tan alto coste pero de gran dimensión, o problemas de tiempo real, en los que se necesita la respuesta en un tiempo máximo [16].

Así, la comunidad científica usa la computación paralela para resolver problemas que sin el paralelismo serían intratables, o que se pueden resolver con mayor precisión o en menor tiempo usando el paralelismo. Algunos campos que se benefician de la programación paralela son: predicciones y estudios meteorológicos, estudio del genoma humano, modelado de la biosfera, predicciones sísmicas, simulación de moléculas, entre otras [17].

Para el funcionamiento de la programación paralela en un clúster, se puede utilizar librerías OpenMPI, MPI u MPICH; para desarrollar versiones eficientes que distribuyan automáticamente los datos de la memoria y generar las comunicaciones necesarias para acceder a los datos en cada uno de los nodos del clúster.

La Fig. 3 (*Mapa conceptual sobre programación paralela*), explica los conceptos más importantes de la programación paralela [12], [13], [15], [18], [19].

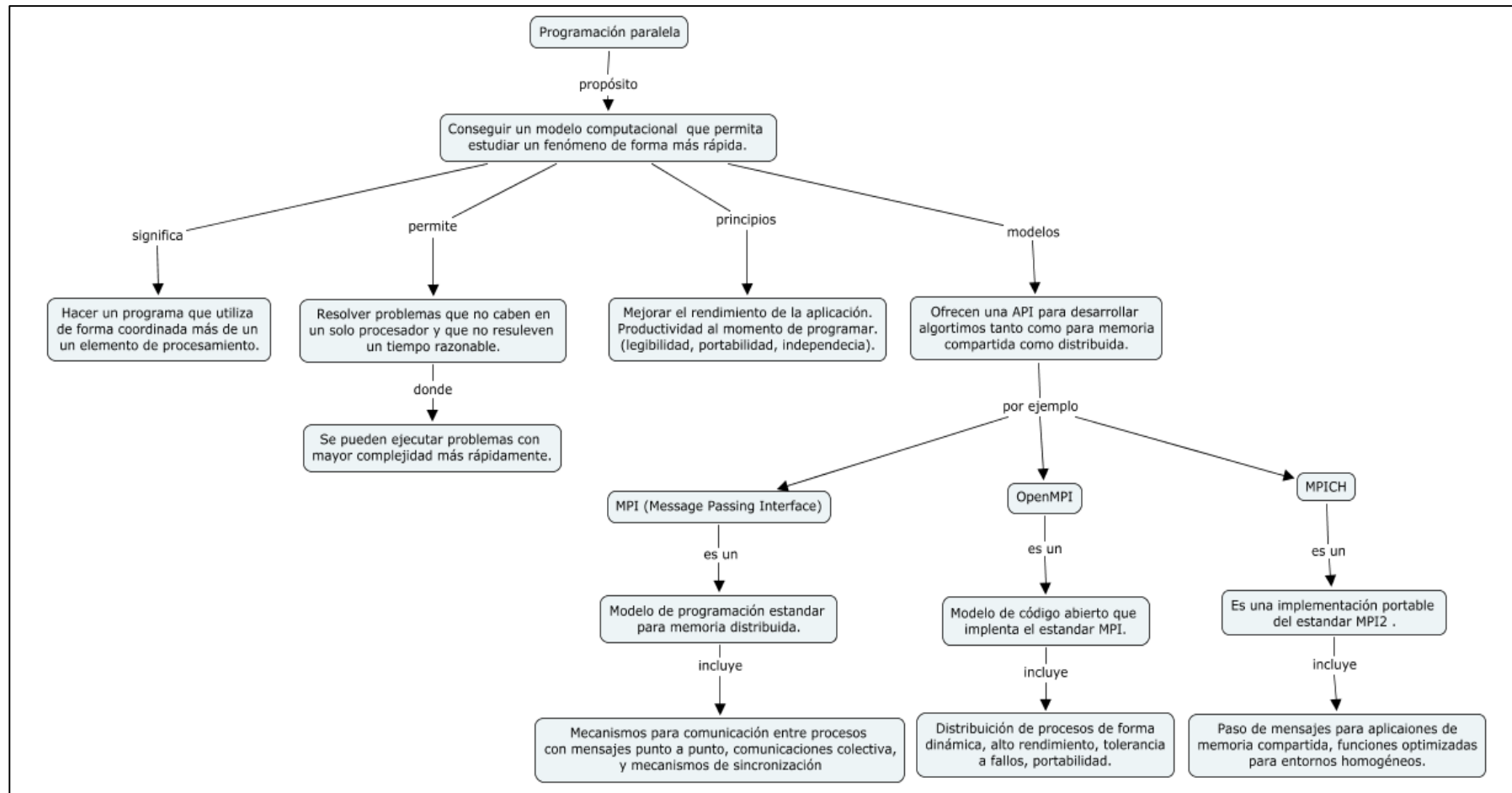


Figura 3. Mapa conceptual sobre programación paralela.

4. Big data

En los últimos años la manera en la que los usuarios interactúan con la tecnología ha cambiado de manera radical debido a la constante evolución de ésta. Revoluciones tecnológicas como la web 2.0, blogs, foros de opinión, redes sociales, vídeos, imágenes, transacciones de comercio electrónico, búsquedas en internet, comunicaciones 3G, 4G, GPS y dispositivos como los teléfonos inteligentes facilitan la conectividad y la generación de grandes cantidades de información que hasta hace muy poco eran impensables. Y no solo la sociedad de consumo ha avanzado, tecnológicamente campos como la ciencia, medicina o la economía también requieren cada vez más tratar con grandes cantidades de datos [20].

4.1. Definición

Big Data son datos que exceden la capacidad de procesamiento de los sistemas de bases de datos convencionales. La razón es que los datos son muy voluminosos, se mueven demasiado rápido o bien no se ajustan a ninguna de las estructuras de la arquitectura de la base de datos. Es por eso que cuando se define Big Data se habla de la definición de las 3Vs: Volumen, Velocidad y Variedad. Que describen las características más importantes de Big Data. De tal manera que, el concepto de Big Data aplica para toda aquella información que no puede ser procesada o analizada utilizando procesos o herramientas tradicionales. Sin embargo, Big Data no se refiere a alguna cantidad en específico, ya que es usualmente utilizado cuando se habla en términos de terabytes, petabytes y exabytes de datos [21], [22].

4.2. Tecnologías para trabajar con Big Data

Existe una gran variedad de tecnologías que nos permiten trabajar con grandes volúmenes de datos entre ellas tenemos; Hadoop, NoSQL, Column oriented DB, SQL Databases, etc.

A continuación se describe una de las tecnologías más utilizadas para el análisis de Big Data, como lo es Apache Hadoop, pero es importante señalar que no se trata de la única existente aunque a lo largo de los años, desde su desarrollo se ha popularizado su

implementación al grado de contar con grandes empresas especializadas en la realización de proyectos basados en dicha tecnología.

4.3. Apache Hadoop

Hadoop es un proyecto de Apache de código libre desarrollado en Java, en el que colaboran un gran número de empresas entre ellas: Facebook, Microsoft, Cloudera y Yahoo! que ha sido el mayor contribuyente al proyecto, su finalidad es englobar un conjunto de herramientas, aplicaciones y frameworks Java para el desarrollo de sistemas y utilidades de computación escalable y distribuida [20].

4.3.1 Análisis de datos con Hadoop en modo clúster

Hadoop se inspiró en el proyecto de Google File System (GFS) y en el paradigma de programación MapReduce, el cual consiste en dividir en dos tareas (mapper - reducer) para manipular los datos distribuidos a nodos de un clúster logrando un alto paralelismo en el procesamiento,

Apache Hadoop permite el desarrollo de aplicaciones para procesar grandes volúmenes de datos, distribuido a través de un modelo de programación sencillo. Está diseñado para ser escalable puesto que trabaja con almacenamiento y procesamiento local (pero distribuido), de manera que funciona tanto para clústeres de un solo nodo como para los que estén formados por miles. Otra característica importante de Hadoop es la detección de errores a nivel de aplicación, pudiendo gestionar los fallos en los distintos nodos y ofreciendo un buen nivel de tolerancia a errores [23].

4.3.2. Módulos Hadoop

El proyecto Hadoop está formado por cuatro módulos:

- **Hadoop Common:** Un conjunto de utilidades que dan soporte a los otros tres módulos.
- **Hadoop Distributed FileSystem (HDFS):** Sistema de ficheros distribuido que provee acceso a los datos que necesitan las aplicaciones.
- **Hadoop MapReduce:** Un modelo de programación distribuido para procesar grandes volúmenes de datos.
- **Hadoop YARN:** Un gestor de recursos del clúster y planificador de ejecución de las aplicaciones.

Aparte de estos módulos también hay otros proyectos que completan el ecosistema Hadoop para desarrollar soluciones basadas en Big Data.

4.3.3. Arquitectura Hadoop

Hadoop trabaja con una arquitectura Maestro-Esclavo con dos tipos de nodo: nodo máster (maestro) y nodos slaves (esclavos), lo cual es perfecto para montar una arquitectura clúster para el análisis de Big Data, en la Fig.4 (*Arquitectura del proyecto Apache hadoop*) se ilustra la arquitectura del proyecto Hadoop [24].

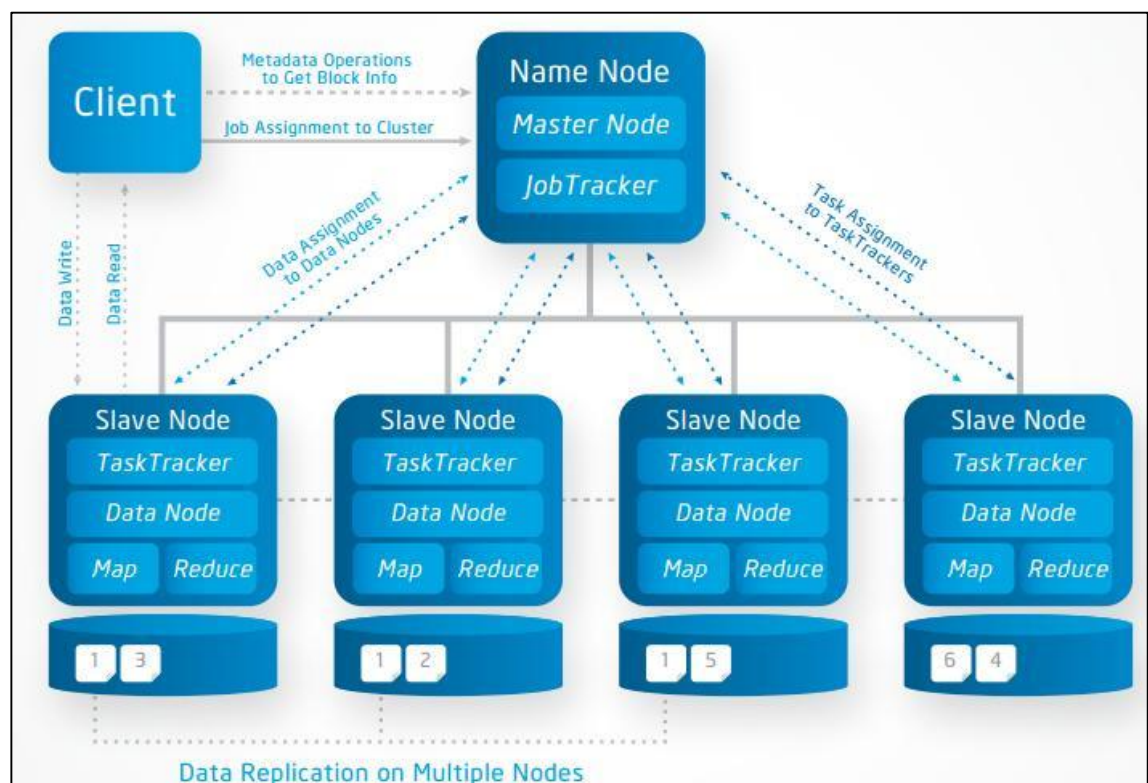


Figura 4. Arquitectura del proyecto Apache Hadoop [24].

5. Blender

Blender es un programa que integra una serie de herramientas para la creación de visualizaciones 3D, imágenes estáticas así como vídeos de alta calidad. Blender tiene incorporado un motor en tiempo real de renderizado que permite la creación de contenido interactivo que puede ser reproducido de manera independiente, además de ser un software multiplataforma y de licencia GPL [25].

5.1 Render distribuido de imagen y video sobre clúster computing

Rendering es un proceso de generación de uno o más imágenes digitales a partir de un modelo o de una colección de modelos, que se caracteriza como una escena virtual [26].

Por lo general una granja de render y un clúster de cálculo utilizan diferentes gestores de colas por lo que implica una distribución eficiente de los procesos entre los recursos existentes de una arquitectura clúster [26], [27] .

El proceso de render ya sea de visualizaciones 3D, o vídeos de alta calidad necesita de una gran cantidad de procesamiento, por lo que se necesitan computadores especialmente diseñados para este tipo de procesos. Con la ayuda de un entorno de renderizado distribuido dinámico, sin la necesidad de la separación física de los recursos, sino formando una granja de render permite acelerar los procesos de creación de contenido cinematográfico [27].

Por lo general una granja de render tiene un nodo maestro y un conjunto de nodos esclavos, donde el nodo maestro ejecuta el planificador de tareas que gestiona la asignación de recursos en los nodos esclavos. Algunas de las características del proceso de renderizado en una arquitectura clúster de computadores son la priorización los gestores de cola, la concesión de las licencias de software lo que implica la asignación dinámica de núcleos de la CPU [26].

e. Materiales y Métodos

Durante el proceso de desarrollo del Trabajo de Titulación, se recurre a diferentes técnicas de recolección de información, métodos científicos y procedimientos que la investigación científica ofrece y que son de mucha utilidad. Entre los métodos de investigación científica principales se manejaron tenemos:

- **Observación activa:**

Con este método se logró involucrar en el problema de investigación, sobre cómo la arquitectura clúster formada por herramientas de software libre y hardware convencional ayudan a la investigación académica-científica a alcanzar resultados más exactos en el menor tiempo posible ejecutando aplicaciones que involucren la utilización de grandes cantidades de recursos computacionales.

Debido a las distintas arquitecturas clúster y la variedad de software que existe se toma en cuenta distintos puntos de vista como: hardware, software con licencia GPL, estabilidad de las versiones del sistema operativo, el middleware, y las herramientas de programación paralela.

- **Estudio de casos:**

La utilización de este método permitió realizar una investigación basada en casos de éxito, sobre la implementación de una arquitectura clúster en centros de investigación superior, a nivel internacional y en el Ecuador, convirtiéndose en la base de la fundamentación teórica del proyecto. Identificando los beneficios en la resolución de problemas de investigaciones académicas-científicas en los diversos campos de aplicación.

- **Experimentación:**

Con la utilización de esta técnica se realizaron las diferentes pruebas garantizando el correcto funcionamiento de nuestra solución, en base a escenarios donde se evaluó cada una de las aplicaciones utilizando 1, 5 10 y 15 nodos, que componen la arquitectura clúster.

- **Técnicas de recolección de información:**

Permitió la recolección de información de las principales temáticas que comprenden la instalación y configuración de un clúster de alto rendimiento, constituyéndose en la base teórica del proyecto, ver (*Sección Revisión de Literatura*).

La revisión bibliográfica se realizó en bases de datos científicas, entre la documentación revisada tenemos: libros, revistas y artículos científicos, tesis doctorales, casos de éxito en centros de investigación e Instituciones de Educación Superior.

Metodología del Trabajo de Titulación

Durante el desarrollo del proyecto, se utilizó la metodología de resolución de problemas que se organiza en siete etapas descritas a continuación:

a. Identificación del problema.

Esta fase comprendió el estudio de la revisión bibliográfica y casos de éxito del funcionamiento de los clúster, en centros de investigación e instituciones de educación superior con la utilización de software libre y hardware convencional. Se analizó también la situación actual del laboratorio de cómputo de la Carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja (CIS-UNL) para identificar si existían procesos que requieran procesar grandes volúmenes de información a altas velocidades y no se disponía de un supercomputador.

b. Explicación del problema.

En el laboratorio de la CIS de la UNL no existían procesos que requieran grandes velocidades de procesamiento pero por la falta de un sistema computacional de alto rendimiento los docentes y estudiantes estaban privados de realizar proyectos académicos relacionados con el procesamiento de grandes volúmenes de información y se desconocía que con la utilización de hardware convencional y software libre se podían obtener grandes beneficios económicos, comparados con la adquisición de supercomputadores que realicen tareas dedicadas a la computación de alto rendimiento, ver (*Sección Resultados Fase 1: 1.1 Revisión bibliográfica y casos de éxito del funcionamiento de los clúster en centros de investigación y en Instituciones de Educación Superior (IES)*).

c. Idear estrategias alternativas de intervención.

Para idear alternativas que permitieron solucionar el problema mencionado se realizó el análisis de los recursos técnicos hardware, software y redes con las que cuenta el laboratorio de la CIS de la UNL, ver (*Sección Resultados Fase 1: 1.2 Análisis de los recursos de los equipos informáticos con los que cuenta actualmente la institución*), en los que se detalla cada una de las características de los apartados mencionados, siendo el punto de partida para la implementación de la arquitectura clúster.

d. Decidir la estrategia.

Una vez realizado el análisis de los recursos técnicos del laboratorio de la CIS de la UNL, se procedió a realizar la búsqueda de las herramientas de software libre, la elección del middleware, la elección de la arquitectura clúster, el diseño de la topología y el direccionamiento de red, ver (*Sección Resultados Fase 1: 1.3 Seleccionar la plataforma de software libre que más se adapte a los requerimientos de los equipos informáticos existentes*).

e. Diseño de la intervención.

En esta fase se determinó los tiempos de implementación del proyecto en el laboratorio de la CIS de la UNL, las actividades a cumplir en los plazos establecidos, ver (*Sección Anexo 1: Anteproyecto de Trabajo de Titulación apartado cronograma de actividades*), en las que se detalla cada una de las actividades desarrolladas en los tiempos indicados, donde cada actividad incluye subactividades, logrando con éxito la implementación de la arquitectura clúster.

f. Desarrollo de la intervención.

En esta fase se desarrolló la instalación del sistema operativo Debian 7.5 como sistema operativo base de la arquitectura clúster, la configuración de la red local, instalación y configuración del protocolo SSH (Secure Shell), instalación y configuración del protocolo NFS (Network File System), instalación y configuración de Mosix, instalación y configuración de MPICH como aplicación del estándar MPI, instalación y configuración de Blender, instalación y configuración de Apache Hadoop, instalación y configuración de Ganglia Monitoring System, ver (*Manual de instalación y configuración: Arquitectura clúster de alto rendimiento utilizando herramientas de software libre*).

g. Evaluación de los logros.

La evaluación de los logros se la realizó con la ejecución de seis proyectos, ver (*Sección Resultados Fase 3: 3.1 Preparar el escenario para realizar las pruebas de funcionalidad*), obteniendo mejores resultados en tiempos de ejecución en comparación a la utilización de un solo computador ver (*Sección Resultados Fase 3: 3.2 Evaluar los resultados en base al escenario planteado mediante procesos que impliquen grandes recursos computacionales*). Como último punto se elaboró un artículo académico sobre la investigación realizada denominado “Arquitectura clúster de alto rendimiento utilizando herramientas de software libre”.

f. Resultados

Durante el desarrollo del proyecto se llevaron a cabo fases y actividades esenciales para su desarrollo; en la primera fase se buscó y documentó sobre casos de éxito de la utilización de clúster de alto rendimiento en centros de investigación e Instituciones de Educación Superior a nivel internacional y en el Ecuador, luego se analizó los recursos hardware y software con los que cuenta actualmente el laboratorio de la CIS, sirviendo como punto de partida para la elección de las herramientas de software libre para la implementación del clúster de alto rendimiento.

En la segunda fase se realizó la selección de la arquitectura clúster, así como el diseño de la topología de red, entre las principales configuraciones realizadas están: instalación del sistema operativo base Debian 7.5 “Wheezy”, configuración de la red local, instalación y configuración del protocolo SSH, instalación y configuración del protocolo NFS, instalación y configuración del middleware Mosix en el kernel de Linux, instalación y configuración de MPICH como estándar para la programación paralela, instalación y configuración de Blender para el renderizado de imagen y video, instalación y configuración de Apache Hadoop para el análisis de datos, instalación y configuración de Ganglia Monitoring System como sistema de monitorización de la arquitectura clúster, ver (*Manual técnico de instalación y configuración: Arquitectura clúster de alto rendimiento utilizando herramientas de software libre*).

En la tercera y última fase del proyecto, se realizaron las pruebas de funcionalidad del clúster de alto rendimiento con la evaluación de seis proyectos entre los que tenemos: pruebas de rendimiento, evaluación de algoritmos, compresión de archivos de audio, cifrado de código, renderización de imágenes, análisis de datos, ver (*Sección Resultados Fase 3: 3.2 Evaluar los resultados en base al escenario planteado mediante procesos que impliquen grandes recursos computacionales*).

1. Fase 1:

1.1 Revisión bibliográfica y casos de éxito del funcionamiento de los clúster en centros de investigación y en instituciones de educación superior (IES)

a. Implementación de un clúster para la ejecución de un modelo de predicción climática, (Grupo de Ciencias de la Tierra y del Ambiente de la Dirección de Investigación (DIUC) de la Universidad de Cuenca, Ecuador [28])

El objetivo de la implementación del clúster dentro del Grupo CTA, es fomentar a la investigación científica en áreas que requieren alta capacidad computacional, optimizando la utilización de los recursos de hardware disponibles, para contar con una herramienta potente y capaz de solventar, en lo posible esas necesidades.

Se tiene como base el sistema operativo, GNU/Linux; el sistema de administración del clúster, cuyo objetivo es la interacción con el usuario y el manejo de ordenadores físicos independientes y conexiones de red de alta velocidad, permitiendo que los ordenadores funcionen como un sólo sistema de computación integrado; y las herramientas de programación que consisten de compiladores, librerías y software especial para el desarrollo y prueba de las aplicaciones.

El estándar utilizado en este proyecto para el desarrollo de software es PVM y MPI como sistemas de programación paralela, el middleware Mosix contiene lo necesario para ejecutar una aplicación que permitirán enlazar el entorno de programación paralela de la capa superior con compiladores paralelos MPI, PVM, GNU, JAVA, etc; software y herramientas de administración, que facilitan la gestión de archivos, balanceo de nodos, la imagen del sistema operativo, que ofrece a los usuarios el acceso unificado a los recursos del sistema.

Dentro de cada nodo enlazado a la red, se tiene una interfaz de red (GigaEthernet) y el software necesario para iniciar, mantener y finalizar la comunicación con el nodo maestro.

La comunicación entre nodos, se realiza a través de una red de alta velocidad GigaEthernet con ayuda de un switch.

Robustez, capacidad de cálculo y bajo costo de implementación son unas de las ventajas por las cuales la implementación de un clúster se ha convertido en una solución ampliamente utilizada dentro del campo de la investigación científica.

b. Diseño e implementación de un clúster de cómputo de alto rendimiento en el Centro de Investigación en matemáticas de la universidad de Guanajuato, México [2].

El clúster de alto rendimiento denominado “El Insurgente” fue diseñado y construido específicamente para aplicaciones de cómputo científico que requieren grandes volúmenes de datos en memoria RAM, programación distribuida con un híbrido de MPI y OpenMPI, usando redes de bajo costo.

Para poder resolver problemas de modelación numérica, por ejemplo de yacimientos petroleros, mecánica de fluidos, mecánica de sólidos, solución de ecuaciones diferenciales parciales, entre otros, es necesario disponer de la capacidad de cómputo ofrecida por un clúster de computadoras.

De manera lógica, cada nodo del clúster tiene una parte de hardware y otra de software. El hardware está compuesto por procesadores, memoria, interfaz de red y discos duros entre otros. En cuanto al software, el nivel bajo corresponde al sistema operativo, el medio consiste en las librerías de paralelización y el alto está representado por la aplicación que se desea ejecutar en el clúster.

Una aplicación o programa diseñado para ejecutarse en un clúster se ejecuta en el nodo maestro, el sistema operativo y la librería de paralelización se encargan de ejecutar copias de este programa en los nodos esclavos del clúster.

Se armaron dos clústers, uno con procesadores AMD Opteron en cada nodo y otro con procesadores Intel Xeon series E5500, ambos procesadores de 64 bits. Se eligieron memorias totalmente comerciales y fáciles de comprar en el mercado aunque de velocidad media. En la parte de redes se utilizaron switch Fast Ethernet con una velocidad de 1000 Mbps, los cuales son más económicos que los de fibra óptica. Se eligió Rocks Cluster System como middleware para la implementación del clúster esta distribución es fácil de instalar y configurar, además de tener la ventaja de ser software libre con licencia "open source".

Rocks Clusters System está basado en RedHat Enterprise Server 5, en cuanto a compiladores, se instaló GCC de GNU el cual también es software libre.

Software: El software instalado en cada uno de los nodos es el siguiente:

- Sistema operativo: Rocks Cluster Linux 5.3 (64 bits)
- Compiladores: GCC 4.5.2. Soporte para C, C++, Fortran y OpenMPI
- Librería de manejo de mensajes: Open-MPI 1.5.3
- Software de pre y post procesamiento: GiD 9.0.6

Hardware: El clúster está compuesto por 32 nodos con las siguientes características:

- **Nodo Maestro:** Procesador: AMD Quad Core Opteron 2350 HE (8 núcleos), Memoria: 12 GB, Disco Duros: SATA 250 GB, 1 TB. Ambos de 7200 revoluciones por minuto
- **16 Nodos Esclavos:** Procesador: 2 x AMD Quad Core Opteron 2350 HE (8 núcleos), Disco Duro: SATA 160 GB, Memoria: 12 GB con velocidad de 667Mhz.
- **15 Nodos Esclavos:** Procesador: Intel(R) Xeon(R) CPU E5502 (4 núcleos), Disco Duro: SATA 160 GB, Memoria: 16 GB con velocidad de 1333 MHz.

El hardware adicional es el siguiente:

Interconexión de Red.

- **Switch Linksys:** 48 puertos, 1Gbps, modelo SLM2048. Para administración interna.
- **Switch SMC:** 48 puertos. 1 Gbps (red interna). Modelo SMC8150L2. Capa 2.
- **Switch Linksys:** 24 puertos. 1 Gbps, modelo SRW2024 (red externa).
- Sistema de energía ininterrumpida (UPS) de 30 KVA, salida senoidal, tres salidas 120 v y entrada de 220 v. El equipo trabaja a 120 v.

Recursos totales del Clúster:

- Núcleos de unidades de procesamiento (cores): 216
- Capacidad en memoria: 455.3 GB
- Capacidad en disco: 5.7 TB

c. Clúster Mangosta: implementación y evaluación, Facultad de Ciencias y Tecnología FACyT Universidad de Carabobo, Venezuela [29].

La implementación del clúster de ordenadores para el departamento de computación de la FACyT como una plataforma computacional de alto rendimiento, usando como base hardware computadoras personales de bajo costo y software de código abierto como Linux.

Este trabajo es utilizado, en áreas como cálculo numérico, simulación y modelado numérico de problemas científicos, procesamiento de imágenes y datos en general.

El prototipo del clúster Mangosta instalado en el departamento de computación son clústers dedicados de alto rendimiento que provee la arquitectura beowulf.

Imagen única del sistema provista por las extensiones del Kernel de Linux que provee OpenMosix.

Se tomaron en cuenta dos escenarios típicos de uso para la plataforma.

- Aplicaciones de tipo secuencial, realizando tareas de cálculo paramétrico y que pueden ser distribuidas a lo largo de los nodos del clúster.
- Aplicaciones paralelas usando librerías de paso de mensaje, específicamente MPI.

Los 16 nodos del clúster Mangosta tienen la siguiente configuración básica:

- Procesador Intel P4 de 2,4 GHz, 1 Gb de memoria RAM, Disco Duro de 40GB 7200 RPM, 2 NICs Fast Ethernet.
- Sistema Operativo RedHat Linux 8.0, Kernel 2.4.20
- MPICH versión 1.2.5, versión portable de la librería de paso de mensajes MPI.
- OpenMosix 3. Extensiones para el Kernel de LINUX 2.4.20.

Los nodos están interconectados a través de dos unidades switch Fast Ethernet de 16 puertos, uno para cada canal.

Características comunes de los casos de éxito investigados.

Entre las características del hardware utilizado para la implementación de la arquitectura clúster están: procesadores Intel Pentium 4, Xeon, Corei3, AMD, Quad Core, Opteron, memorias RAM de 1 a 16 Gb, espacio en disco de 40 a 250 Gb, middleware para proporcionar una imagen única del sistema como Mosix, Rocks Clúster System,

OpenMosix, herramientas para programación paralela encontramos PVM, MPI, MPICH, entre los estándares más utilizados.

La arquitectura clúster es utilizada principalmente en áreas como el cálculo numérico, la simulación y modelado de problemas científicos, procesamiento de imágenes y datos en general.

1.2 Análisis de los recursos de los equipos informáticos con los que cuenta actualmente la institución

1.2.1 Análisis de la situación actual del laboratorio CIS-UNL

El laboratorio de la CIS de la UNL tiene un área total de 46.64 m², las especificaciones técnicas de las dimensiones del laboratorio se detallan en la Fig. 4 (*Distribución de los equipos en el laboratorio de la CIS UNL*), la ubicación física de los equipos se ilustra en la Fig.5 (*Vista 3D laboratorio CIS UNL*).

El laboratorio está conformado por 15 ordenadores con características similares, en cuanto a hardware y software se refiere, ver (*Sección Resultados Fase 1: 1.2 Análisis de los recursos de los equipos informáticos con los que cuenta actualmente la institución apartado Recursos Hardware, Recursos Software*).

El laboratorio no cuenta con una topología de red cableada, ni un direccionamiento IP estático; los equipos están interconectados a través de la red inalámbrica mediante un Router Cisco LINKSYS EA4500, el cual utiliza direccionamiento IP dinámico (DHCP). Además cuenta con un switch Cisco Catalyst serie 2960 de 24 puertos que soportan velocidades de 10/100/100 Mbps, pero actualmente no está en funcionamiento ver (*Sección Resultados Fase 1: 1.2 Análisis de los recursos de los equipos informáticos con los que cuenta actualmente la institución apartado Recursos de interconexión de red*).

Por lo que se ha determinado que los recursos informáticos y la infraestructura con la que cuenta el laboratorio de la CIS-UNL son idóneos para la implementación del clúster de alto rendimiento.

1.2.2 Ubicación física de los equipos en el laboratorio de la carrera de Ingeniería en Sistemas.

El laboratorio de la CIS está conformado por 15 ordenadores, en la Fig. 5 (*Distribución de los equipos en el laboratorio de la CIS-UNL*) podemos observar las especificaciones técnicas de la distribución de los equipos en el laboratorio, la vista panorámica del mismo se observa en la Fig. 6 (*Vista 3D laboratorio CIS-UNL*). El área total del laboratorio es de 46.64 m².



Figura 5. Distribución de los equipos en el laboratorio de la CIS-UNL.

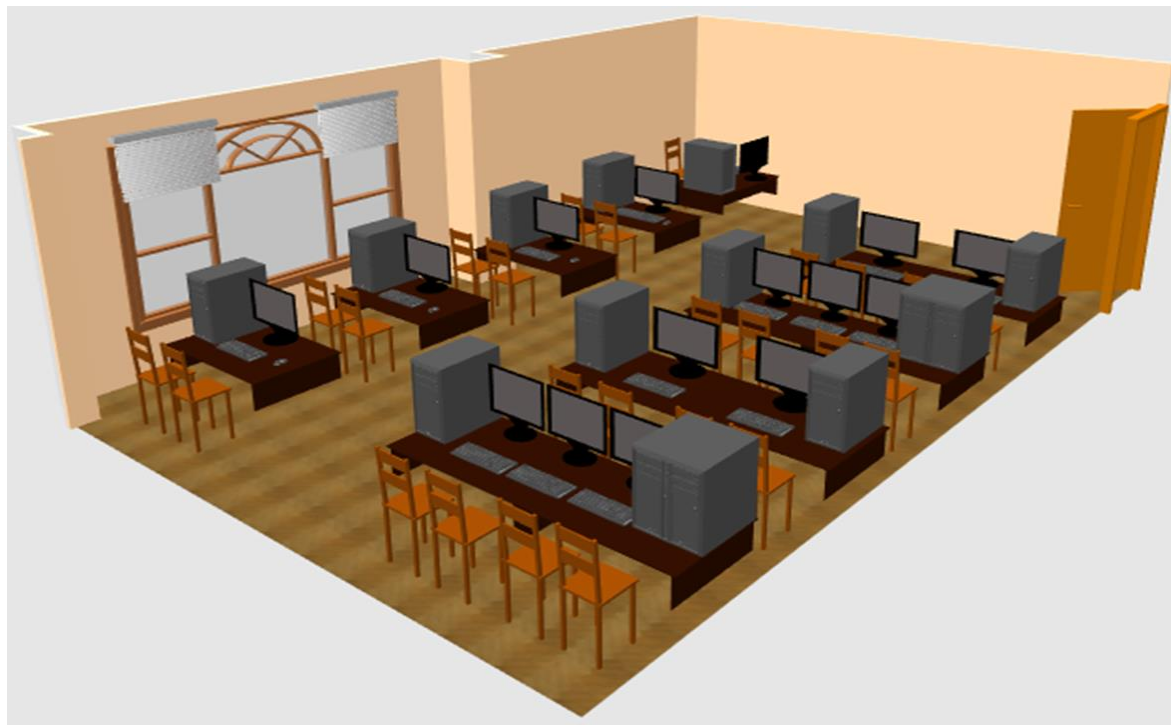


Figura 6. Vista 3D laboratorio CIS-UNL.

1.2.2 Recursos Hardware:

El laboratorio está formado por 15 computadoras con características homogéneas que se detallan a continuación:

Procesador Intel Corei7-2600, 8 procesadores lógicos (4 núcleos por procesador) de 3.64 GHz, Memoria RAM de 4 GB de 1333 MHz, Disco Duro: SATA de 500 (500.107.862.016 bytes) GB de 7200 rpm.

Sistema de energía ininterrumpida: cada ordenador del laboratorio cuenta su propio UPS TRIPP LITE AVR550U que proporciona salida de 550 VA 120 V, 4 tomacorrientes soportados por la batería, 4 tomacorrientes únicamente con protección contra sobretensiones.

1.2.2 Recursos Software:

El software instalado en cada uno de los ordenadores del laboratorio de la CIS se detallan en la Tabla I (*Particionamiento lógico sistemas operativos laboratorio cis-unl*).

TABLA I. PARTICIONAMIENTO LÓGICO SISTEMAS OPERATIVOS LABORATORIO CIS-UNL.

| Sistema Operativo | Particiones del disco | Tipo | Capacidad |
|---------------------------------------|-----------------------|-----------|-----------|
| Reservado para el sistema (Windows 8) | /dev/sda1 | ntfs | 367 MB |
| Windows 8 (64 bits) | /dev/sda2 | ntfs | 161 GB |
| Ubuntu 13.04 (64 bits) | /dev/sda3 | Ext4 | 86 GB |
| Centos 6.3 (64 bits) | /dev/sda4 | Extendida | 153 GB |
| | /dev/sda5 | Swap | 4 GB |
| | /dev/sda6 | Ext4 | 10 GB |
| Partición libre | ----- | ----- | 85 GB |
| Total | ----- | ----- | 500 |

1.2.3 Interconexión de Red:

Tarjetas de Red.

- **Wireless:** D-Link DWA525 PCI Wireless Adapter 802.11.

Basado en la tecnología Wireless N con una sola antena, el adaptador DWA-525 alcanza velocidades inalámbricas de hasta 150 Mbps y cobertura extendida sin puntos muertos.

- **PCI:** INTEL 82579LM Gigabit Network Connection.

Cantidad de Puertos: 1.

Velocidad de Datos por puerto: 10/100/1000 Mbps.

Rango de temperatura: 0° C a 85° C.

Tipo de interfaz del sistema: PCI Express.

- **Router: Cisco Linksys EA4500.**

Especificaciones Técnicas:

- Doble banda 900N (2,4 GHz y 5 GHz).
- Velocidad de transmisión de 450 Mbps.
- Matriz de antenas MIMO 3x3.
- 4 Puertos Gigabit Ethernet.
- 1 puerto USB con almacenamiento compartido y servidor virtual.

Switch: Cisco Catalyst serie 2960.

Especificaciones técnicas:

- Switch capa 2.
- Fuente de alimentación fija con una fuente de alimentación externa redundante.
- Son switches Gigabit Ethernet (10/100/1000) apilables de configuración fija que ofrecen conectividad de red para grandes y medianas empresas.
- Los conmutadores Cisco Catalyst 2960 tienen una banda de envío de 8.8 Gbps.

Los switches Cisco Catalyst serie 2960-X consumen hasta un 80% menos de energía, al contar con las mejores funciones de administración energética [30].

1.3 Seleccionar la plataforma de software libre que más se adapte a los requerimientos de los equipos informáticos existentes

La elección de la tecnología a utilizar para la implementación de la arquitectura clúster en el laboratorio de la CIS, se realizó acorde a los requerimientos de hardware existentes, ver (*Sección Resultados Fase 1: 1.2 2 Análisis de los recursos de los equipos informáticos con los que cuenta actualmente la institución*).

Para la elección del sistema operativo base del clúster se tomaron algunos puntos en consideración como: tipo de licencia, sistema de archivos, arquitecturas soportadas, última versión estable, versión del kernel, requisitos mínimos de memoria RAM, requisitos mínimos de Disco Duro; entre los sistemas operativos analizados tenemos CentOS, OpenSuse, Ubuntu, Debian; las características soportadas por cada uno de los sistemas operativos se detallan en la Tabla II. (*Comparativa entre sistemas operativos GNU/LINUX*).

1.3.1 Sistema Operativo [31].

TABLA II. COMPARATIVA ENTRE SISTEMAS OPERATIVOS GNU/LINUX.

| Características | Sistema Operativo | | | |
|---------------------------------|--------------------------|---------------------------------------|---------------------------------------|---|
| | CentOS | OpenSuse | Ubuntu | Debian |
| Licencia | GPL | GPL | GPL | GPL |
| Sistema de Archivos | XFS | Btrfs, ext3, ext4, JFS, ReiserFS, XFS | Btrfs, ext3, ext4, JFS, ReiserFS, XFS | Btrfs, ext3, ext4, JFS, ReiserFS, XFS |
| Arquitecturas soportadas | i386, x86_64 | i586, x86_64 | amd64, armhf, i686, powerpc | amd64, armel, i386, ia64, mips, mipsel, powerpc, s390x, sparc |
| Última Versión Estable | CentOS 7.0 2014/07/07 | 13.1 2013/11/19 | 14.04 LTS trusty 2014/04/17 | 7.6 Wheezy 2014/07/12 |
| Versión del Kernel | 3.10 | 3.11.6 | 3.13 | 3.2.41 |
| Instalación | Modo Grafico | Modo Grafico | Modo Grafico | Modo Grafico |

| Gestor de paquetes | RPM | RPM | DEB | DEB |
|--------------------|-----------------|---------------|--------|---------------|
| Basado en | Fedora, Red Hat | Independiente | Debian | Independiente |
| Memoria RAM mínima | 2 GB | 2 GB | 512 MB | 512 MB |
| Disco duro mínimo | 20 GB | 5 GB | 5 GB | 5 GB |

Una vez analizadas las características de los sistemas operativos, se llegó a la conclusión de que el sistema operativo base a utilizar en la implementación del clúster es Debian en su versión 7.5 denominada “Wheezy”, ya que cuenta con soporte constante por parte de la comunidad linuxera para la distribución, además cuenta con actualizaciones y eliminación de dependencias de otras librerías al momento de instalar paquetes, soporta múltiples arquitecturas como alpha, amd64, armel, hppa, i386, ia64, mips, mipsel, powerpc, s390, sparc. También funciona en los kernels GNU Hurd y FreeBSD además de Linux [32].

Debian ofrece estabilidad, es rápido y ligero en memoria, los controladores para la mayoría del hardware son desarrollados por usuarios GNU/Linux, ofrece software de seguridad que permite enviar correo entre usuarios preservando su privacidad [33].

Entre algunas de las instituciones educativas más prestigiosas del mundo que usan Debian citamos: Departamento de Informática de la Universidad de Zurich, Zurich, Suiza; Universidad de Saarland, Saarbrücken, Alemania; Ciencias de la Computación, Universidad de Brown, Providence, RI, EEUU; en las que se utiliza software de código abierto en su infraestructura de servidores, entre los servicios prestados están: DHCP, LDAP, SMTP, IMAP, Webmail, listas de correo, Samba, almacenamiento redundante, copias de seguridad, repositorios de Subversión, Drupal, Jommla, MediaWiki, Tomcat, PostgreSQL, con lo que se ofrece que los servicios centrales de tecnologías de información estén completos [34].

1.3.2 Sistema Middleware

En la Tabla III (*Comparativa entre sistemas middleware clúster*) se realizó una síntesis de los puntos más relevantes de las características de los sistemas middleware. Entre los sistemas middleware analizados están OpenMosix, Mosix, Condor, OpenSSI, SSI, Kerrighed, Rocks Cluster, Oscar.

TABLA III. COMPARATIVA ENTRE SISTEMAS MIDDLEWARE CLÚSTER.

| Middleware | Descripción | Componentes | Soporte |
|------------------|--|---|--|
| OpenMosix | <ul style="list-style-type: none"> • Actúa como un sistema multiprocesador. • Balanceo de carga. • Migración entre los procesos. • No migran los procesos multi-hilo. • El script de inicio <i>/etc/init.d/openmosix</i>. | <ul style="list-style-type: none"> • Parche para el kernel Linux. • Herramientas para línea de comandos y para entorno gráfico. | <ul style="list-style-type: none"> • El Proyecto OpenMosix ha cerrado oficialmente el 1 de marzo de 2008 <p>URL: [openmosix.sourceforge.net]</p> |
| Mosix | <ul style="list-style-type: none"> • Sistema operativo que proporciona una imagen única del sistema. | <ul style="list-style-type: none"> • Un entorno no virtualizado requiere parchar el kernel de | <ul style="list-style-type: none"> • El proyecto Mosix actualmente está brindando soporte continuo a sus versiones por lo |

| | | | |
|----------------------|---|---|--|
| | <ul style="list-style-type: none"> • Detección automática de recursos. • Soporta aplicaciones secuenciales y paralelas. • Asignación dinámica de recursos. • Equilibrio de carga. | <p>Linux y proporciona un mejor rendimiento.</p> | <p>que la última versión fue lanzada el 31 de marzo del 2014 3.4.0.12, que además es compatible con el kernel de Linux 3.12</p> <p>URL: www.mosix.org</p> |
| <p>Condor</p> | <ul style="list-style-type: none"> • Utiliza eficazmente la potencia de cálculo de las estaciones conectadas en la red. • Migran sus tareas a distintas máquinas que estén disponibles. • Condor no exige la utilización de archivos compartidos (NFS, AFS), permite migrar los distintos trabajos por distintos dominios administrativos. | <ul style="list-style-type: none"> • Checkpoint y migración. • Sistema de llamadas remotas. • No es necesario modificar el código fuente de las aplicaciones. • Condor permite computación Grid. • Sistema Operativo: Unix y Windows. • Interfaz de usuario: Consola/Terminal | <ul style="list-style-type: none"> • Estado del desarrollo: estable y en desarrollo. • Versión estable 22 de Mayo del 2014. HTCondor 8.1.6 <p>URL: http://research.cs.wisc.edu/htcondor</p> |

| | | | |
|-------------------------|---|--|---|
| <p>OpenSSI</p> | <ul style="list-style-type: none"> • Alta disponibilidad y escalabilidad. • Capacidad de administración. • Sistema es capaz de migrar una aplicación multihebra, pero no es capaz de migrar hebras individualmente. | <ul style="list-style-type: none"> • No es necesario utilizar MPI o PVM para beneficiarse de paralelismo. • Migración de Procesos. • Nivelación de carga. | <ul style="list-style-type: none"> • OpenSSI 1.9.6. x86 x64 versión para Debian Lenny x86_64 (Linux-2.6.12). • Fecha de lanzamiento 18 De febrero del 2010. <p>URL: [http://openssi.org/]</p> |
| <p>Kerrighed</p> | <ul style="list-style-type: none"> • Kerrighed es un sistema operativo de imágenes de sistema único para Clúster. • Kerrighed ofrece la vista de una máquina SMP única en la parte superior de un clúster de PC estándar. • Script de inicio <i>/etc/init.d/kerrighed.</i> | <ul style="list-style-type: none"> • Apoyo a la agrupación de memoria compartida amplia. • Puntos de control de proceso transparente. • Alta disponibilidad para las aplicaciones de usuario. • No tiene herramientas de usuario gráficas. | <ul style="list-style-type: none"> • La última versión de Kerrighed es la 3.0.0 . • Fue lanzada el 14 de junio de 2010. Está basado en Linux 2.6.30 . <p>URL: [kerrighed.org]</p> |

| | | | |
|---------------------|--|--|--|
| <p>Rocks</p> | <ul style="list-style-type: none"> • Construido bajo Red Hat, soporta las arquitecturas x86, x86_64 y IA_64. • Soporta los procesadores AMD Athlon, Opteron, Itanium. • Una de las distribuciones más utilizadas en el ámbito de los clúster por su facilidad de instalación e incorporación de nuevos nodos. | <p>Basado inicialmente en Linux pero las últimas versiones están basadas en CentOS, las instalaciones pueden ser personalizadas utilizando CD's especiales, que extienden el sistema integrando automáticamente los mecanismos de gestión y empaquetamiento usados por el software base.</p> | <ul style="list-style-type: none"> • Última versión estable 6.1. Fue lanzada el 29 de noviembre del 2012. • Basado en Linux con licencia GPL. <p>URL: [http://www.rocksclusters.org]</p> |
| <p>OSCAR</p> | <ul style="list-style-type: none"> • Incluye una arquitectura robusta y extensible lista para iniciar la producción. • Crea imágenes de disco personalizadas para la prestación de los nodos con todo el software necesario para su funcionamiento. | <ul style="list-style-type: none"> • Contiene aplicaciones como cúmulos Web de balanceo de carga y paquetes de clustering de alta disponibilidad. • Utilizado principalmente en la computación científica mediante una interfaz de paso de mensajes. | <ul style="list-style-type: none"> • Última versión estable 6.1.1. Fue lanzada el 31 de mayo del 2011. • Basado en Linux. <p>URL: [http://svn.oscar.openclustergroup.org]</p> |

Una vez finalizado el análisis de las características de los sistemas middleware se llegó a determinar que Mosix es el más adecuado para la implementación del clúster de alto rendimiento, es descentralizado, es decir cualquiera de los nodos puede ser nodo maestro, además permite agregar y quitar nodos en cualquier momento, brinda soporte constante a cada una de las versiones, la última versión estable salió el 31 de marzo del 2014.

Permite paralelizar aplicaciones que no han sido desarrolladas con ese fin, y dichas aplicaciones ejecuten más de un proceso a la vez.

Entre las aplicaciones que pueden utilizar las características de Mosix tenemos: aplicaciones científicas e ingenieriles, procesos paralelos, además ofrece herramientas gráficas que facilitan su administración.

Una de las principales funciones es la migración de procesos, un proceso que se ejecuta en un nodo puede migrar y ejecutarse en otros nodos y volver a migrar si es necesario incluso volver al nodo que lo inició, todo dependiendo de las condiciones del programa en ejecución [8].

Las migraciones de los procesos ocurren de forma automática y transparente en respuesta a la disponibilidad de los recursos.

1.3.3. Arquitectura clúster

En la Tabla IV (*Comparativa entre arquitecturas clúster*), se detalla las características analizadas para la elección de la arquitectura clúster basada en: tipos de nodo, arquitecturas, red de comunicaciones, sistema operativo y herramientas de programación paralela.

Tabla IV. COMPARATIVA ENTRE ARQUITECTURAS CLÚSTER

| Tipo Arquitectura | Nodos | Red de Comunicación | Sistema Operativo | Herramientas de programación |
|---------------------|------------------------|---------------------------|-----------------------|-------------------------------|
| Beowulf | PC`s | Switch Ethernet TCP/IP | GNU/Linux | MPI/PVM Sockets and HPF |
| Berkeley NOW | PC`s basado en Solaris | Red Myrinet. | Solaris, GNU/LINUX | AM, PVM, MPI, HPF, Split-C |

| | | | | |
|-------------------|--|--------------------------------|-----------------|--|
| HPVM | PC`s | Mirynet with Fast Messanges | Microsoft NT | Java frontend, FM, Sockets, Global Arrays, SHMEM, and MPI |
| Solaris MC | Solaris-based PC`s y Workstation | Solaris- supported | Solaris | C++ and Corba |

Finalizado el análisis de las características de las distintas arquitecturas clúster, se determinó que la arquitectura clúster de tipo beowulf es la más óptima ya que cumple con características requeridas para la implementación del proyecto, al estar formada por PC´s convencionales, sistemas operativos libres y redes de alta velocidad y una variedad de herramientas de programación paralela.

2. Fase 2:

2.1 Diseño lógico de la arquitectura clúster en el laboratorio de la CIS

2.1.1 Arquitectura



Figura 7. Arquitectura clúster de alto rendimiento CIS-UNL.

En la Fig. 7 (*Arquitectura clúster de alto rendimiento CIS-UNL*) se visualiza la arquitectura del clúster, la misma que cuenta con las siguientes capas:

- **Capa de aplicación**, permite la ejecución aplicaciones secuenciales, aplicaciones paralelas (*desarrolladas específicamente para la ejecución en el clúster*) y ambiente de programación paralela (*permite implementación de algoritmos que hacen uso de recursos compartidos: CPU, memoria, y servicios*).
- **Capa de middleware**, es el software que generalmente actúa entre el sistema operativo y las aplicaciones con la finalidad de proveer:
 - **Librerías MPI**, enlaza a los compiladores con el ambiente de programación paralela que se encuentra en la capa de aplicación.
 - **Herramientas del clúster**, permiten su correcto funcionamiento: migración de procesos, checkpoint-restart (detener y restaurar uno o

- varios procesos, migrarlos a otro nodo) balanceo de carga entre los nodos, tolerancia a fallos, etc.
- **Imagen única del sistema SSI**, da al usuario la sensación de hacer uso de un único computador
- **Sistema operativo**, autoriza a los usuarios acceso unificado a los recursos del sistema.
- **Red de alta velocidad**, permite la comunicación entre los distintos nodos del clúster con tecnología de alta velocidad(Gigabit)

2.1.2 Topología

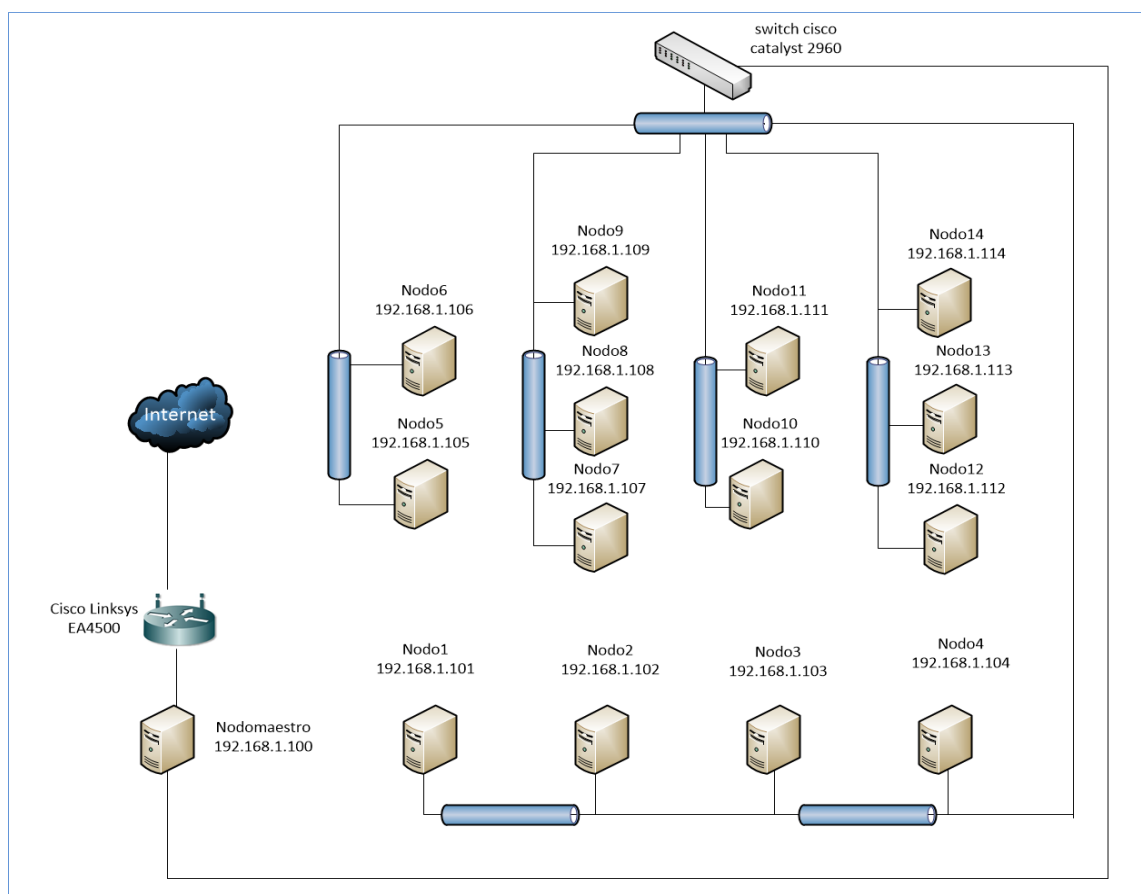


Figura 8. Diagrama de topología propuesto.

En la Fig. 8 (*Diagrama de topología propuesto*) se visualiza la topología utilizada. En esta topología no es necesario que los nodos esclavos tengan conexión a internet, por tal razón estos nodos pertenecen a una red privada, para poder hacer uso de este

servicio, el nodo maestro cuenta con dos tarjetas de red, una para conectarse a la red LAN en la cual están enlazados todos los nodos que componen el clúster y la segunda para poder tener conexión a Internet.

La topología que se utilizó es de tipo estrella que entre las principales ventajas se destacan: si el cable de un computador se daña o se rompe las demás computadoras conectadas al switch siguen funcionando, agregar un nodo es muy fácil lo único que hay que hacer es conectarla al switch, se tiene una mejor organización, a cada uno de los switch se los ubicó en el centro del laboratorio como se observa en la Fig. 8 (*Diagrama de topología propuesto*), entre las desventajas encontradas están que es más cara de realizar por la cantidad de cable categoría 5E que se utilizó, si el switch deja de funcionar ninguna de las computadoras tendrá conexión a la red, el número de nodos del clúster no puede exceder la cantidad de puertos disponibles en el switch.

Se implementó un direccionamiento IP estático para los ordenadores en el laboratorio, el direccionamiento se realizaba mediante DHCP, lo que dificultaba las configuraciones de cada uno de los nodos. La Tabla V (*Direccionamiento IP clúster CIS-UNL*), detalla el rango de direcciones IP estáticas utilizadas, desde la dirección [192.168.1.100] hasta [192.168.1.114].

TABLA V. DIRECCIONAMIENTO IP CLÚSTER CIS-UNL.

| Dirección de Red | Máscara de Subred | Gateway | Nodo |
|------------------|-------------------|-------------|--------------|
| 192.168.1.100 | 255.255.255.0 | 192.168.1.1 | Nodo Maestro |
| 192.168.1.101 | 255.255.255.0 | 192.168.1.1 | Nodo1 |
| 192.168.1.102 | 255.255.255.0 | 192.168.1.1 | Nodo2 |
| 192.168.1.103 | 255.255.255.0 | 192.168.1.1 | Nodo3 |
| 192.168.1.104 | 255.255.255.0 | 192.168.1.1 | Nodo4 |
| 192.168.1.105 | 255.255.255.0 | 192.168.1.1 | Nodo5 |
| 192.168.1.106 | 255.255.255.0 | 192.168.1.1 | Nodo6 |
| 192.168.1.107 | 255.255.255.0 | 192.168.1.1 | Nodo7 |
| 192.168.1.108 | 255.255.255.0 | 192.168.1.1 | Nodo8 |
| 192.168.1.109 | 255.255.255.0 | 192.168.1.1 | Nodo9 |

| | | | |
|---------------|---------------|-------------|--------|
| 192.168.1.110 | 255.255.255.0 | 192.168.1.1 | Nodo10 |
| 192.168.1.111 | 255.255.255.0 | 192.168.1.1 | Nodo11 |
| 192.168.1.112 | 255.255.255.0 | 192.168.1.1 | Nodo12 |
| 192.168.1.113 | 255.255.255.0 | 192.168.1.1 | Nodo13 |
| 192.168.1.114 | 255.255.255.0 | 192.168.1.1 | Nodo14 |

2.2 Instalación y configuración del sistema operativo del clúster

Los criterios analizados para la elección del sistema operativo base del clúster se detallan en, ver (*Sección Resultados Fase 1: 1.2 Análisis de los recursos de los equipos informáticos con los que cuenta actualmente la institución apartado Sistema operativo*), luego de la selección del sistema operativo Debian 7.5 “Wheezy” [32], se realizó la instalación y configuración en cada uno de los nodos que conforman la arquitectura.

2.3 Instalación y configuración de los servicios del clúster en el nodo maestro y los nodos esclavos

La implementación del clúster de alto rendimiento “CLUSTER_CIS-UNL” comprendió algunas actividades, ver (*Manual técnico de instalación y configuración del clúster de alto rendimiento utilizando de herramientas de software libre*), que se detallan a continuación:

- Configuración de la red local.
- Asignación de los nodos al archivo host.
- Instalación y configuración del protocolo Secure Shell (SSH).
- Instalación y configuración del protocolo Network File System (NFS).
- Instalación y configuración del middleware Mosix en el kernel de Linux.
- Instalación y configuración de MPICH.
- Instalación y configuración del framework Apache Hadoop.
- Instalación y configuración de Ganglia Monitoring System.

3. Fase 3:

3.1 Preparar el escenario para realizar las pruebas de funcionalidad

Proyecto 1: Testeo de rendimiento con Stress-Test.

Stress-Test es un programa utilizado para evaluar un clúster Mosix. Realiza evaluaciones a las aplicaciones y al kernel, para testear la estabilidad, la migración de procesos, balanceo de carga entre otros [35].

Durante la ejecución del test el clúster se verá sobrecargado, por lo que se debe cerrar cualquier aplicación antes de iniciar el proceso del test. Al finalizar la evaluación del clúster se genera un informe detallado acerca de los componentes que han sido evaluados [36].

En el Anexo 2 (*Testeo de rendimiento con Stress-Test*), se describe el proceso de instalación y configuración, en el programa Stress-Test corren un total de 7 aplicaciones para evaluar el rendimiento del clúster, en esta prueba utilizaremos la aplicación **distkeygen**, en el Anexo 2 (*Testeo de rendimiento con Stress-Test apartado Prueba test distkeygen*), se detalla el proceso de ejecución, en la Tabla VI (*Resultado de la ejecución de pruebas de testeo de rendimiento con stress-test*), se detallan los resultados del tiempo de ejecución de la aplicación.

Proyecto 2: Cálculo del valor aproximado de π utilizando MPI.

El proyecto evalúa el valor aproximado de pi, utilizando librerías de paso de mensajes MPI, en el Anexo 3 (*Cálculo del valor aproximado de π utilizando MPI*), se detalla las configuraciones realizadas, los resultados obtenidos de la ejecución de la aplicación utilizando 1, 5, 10 y 15 nodos respectivamente.

El código utilizado se detalla en el Anexo 3 (*Cálculo del valor aproximado de π utilizando MPI apartado Tabla XVIII Código fuente valor de pi*), el mismo pertenecen a los archivos de ejemplo de MPICH 3.1.2.

Los resultados obtenidos se detallan en la Tabla VII (*Resultado de la ejecución del proceso de cálculo del valor aproximado de π utilizando mpi*).

Proyecto 3: Pruebas de compresión de música con Mosix.

En el proyecto se realiza la compresión de 32 ficheros de audio que inicialmente están en un formato .wav con un tamaño total de 5.2 Gb, a un formato **.flac**. Para ello se utilizó la librería **flac**, ésta librería permite la compresión de ficheros en formato Open Source.

En el Anexo 4 (*Pruebas de compresión de música con Mosix*), se detalla el script utilizado para el proceso de compresión. Para la ejecución del script utilizando solo el nodo maestro hacemos uso del comando **time sh jnormal.sh**; al momento de utilizar 5, 10 y 15 nodos utilizamos el comando **time sh jmosix.sh**, dicho script permite realizar la migración automática de los procesos en el clúster.

Los resultados obtenidos se detallan en la Tabla VIII (*Resultado de la ejecución, pruebas de compresión de música con Mosix*)

Luego del proceso de compresión el tamaño total de los ficheros es de 1.9 Gb, ver Fig. 43 (*Tamaño de los archivos luego del proceso de compresión utilizando Mosix*).

Proyecto 4: Cifrado de códigos con John The Ripper (JTR) y MPI.

La versión utilizada de JTR es 1.7.9-jumbo-7, la cual implementa librerías MPI (ver *Anexo 4: Instalación y configuración de John The Ripper*); JTR es un programa de criptografía que aplica ataques de diccionario o fuerza bruta para descifrar contraseñas.

Las pruebas se realizaron sobre uno de los ficheros más importantes de Linux como lo es **shadow**, este fichero almacena información cifrada de las contraseñas de cada una de las cuentas de usuario del sistema operativo.

Las contraseñas decodificadas son de longitud variable y consta de caracteres alfanuméricos.

Los resultados obtenidos se detallan en la Tabla IX (*Resultado de la ejecución de cifrado de código utilizando John the Ripper*).

Proyecto 5: Renderización de imágenes y videos con Blender.

Blender es un software informático multiplataforma, que puede usarse en las diferentes arquitecturas dedicadas al modelado, animación y creación de gráficos tridimensionales e interactivos [37].

El proceso de renderizado de imágenes o videos se realiza mediante el cálculo de iluminación partiendo de un modelo 3D, proceso realizado por animadores o productores visuales.

En el Anexo 7 (*Renderización de imágenes y videos con blender*), se detalla el proceso de renderización utilizando uno y con distintos número de nodos, los resultados obtenidos se detallan en la Tabla X (*Resultado de la ejecución de renderización de imágenes y videos con blender*).

Proyecto 6: Análisis de datos (Big Data) con Apache Hadoop.

Para realizar esta prueba hemos hecho uso de los Open Data o datos abiertos, que están a disposición de forma libre sobre temas como: economía, mediciones medioambientales, movilidad, seguridad, servicios, deporte, cultura, etc. Para esta prueba utilizaremos un dataset sobre medición de la calidad del aire en España en las estaciones de Castilla y León, en el que se encuentra información recopilada desde el año 1997 hasta el año 2013, el mismo que se encuentra disponible en el enlace <http://goo.gl/jZO6OT>. El fichero de extensión .csv, fue modificado a nuestros requerimientos con lo que se logró obtener un archivo que contiene más de 16 millones de líneas y un peso mayor a 950 Mb. El código fuente desarrollado para el análisis del dataset utilizando el framework Apache Hadoop se encuentra disponible de forma libre en <http://goo.gl/Jy6gVr>.

Los resultados obtenidos de la ejecución del proyecto 6 se detallan en la Tabla XI (*Resultado de la ejecución del proyecto 6 con diferente número de nodos y cores*).

3.2. Evaluar los resultados en base al escenario planteado mediante procesos que impliquen grandes recursos computacionales

Proyecto 1: Testeo de rendimiento con Stress-Test.

TABLA VI. RESULTADO DE LA EJECUCIÓN DE PRUEBAS DE TESTEO DE RENDIMIENTO CON STRESS-TEST.

| Número de Nodos | Número de procesadores | Tiempo (min) |
|-----------------|------------------------|-------------------|
| 1 | 8 | 18 min 27.825 seg |
| 5 | 40 | 11 min 42.526 seg |
| 10 | 80 | 9 min 39.128 seg |
| 15 | 120 | 3 min 29.970 seg |

En la Fig. 9 (*Tiempo de ejecución testeo de rendimiento con Stress-Test*) se ilustra de manera gráfica los resultados obtenidos al ejecutar la aplicación en los distintos nodos.

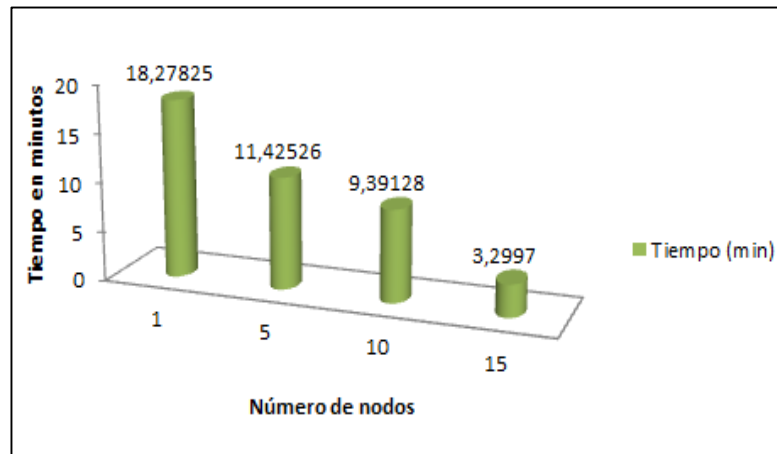


Figura 9. Tiempo de ejecución testeo de rendimiento con Stress-Test.

Interpretación de los resultados.

De la prueba realizada se deduce que utilizando un solo nodo, el tiempo de ejecución es de 18.27825 minutos, mientras que al utilizar los 15 nodos de la arquitectura el tiempo es de 3.2997 minutos, obteniendo una disminución de 14.9785 minutos de diferencia, lo que se interpreta como una eficiencia de aproximadamente 85%.

Proyecto 2: Cálculo del valor aproximado de π utilizando librerías MPI.

TABLA VII. RESULTADO DE LA EJECUCIÓN DEL PROCESO DE CÁLCULO DEL VALOR APROXIMADO DE π UTILIZANDO MPI.

| Número de Nodos | Número de procesadores | Tiempo (seg) |
|------------------------|-------------------------------|---------------------|
| 1 | 8 | 0.959801 |
| 5 | 40 | 0.321487 |
| 10 | 80 | 0.228983 |
| 15 | 120 | 0.136832 |

En la Fig. 10 (*Tiempos de ejecución cálculo del valor aproximado de π utilizando MPI*) se ilustra de manera gráfica los resultados obtenidos al ejecutar la aplicación en los distintos nodos.

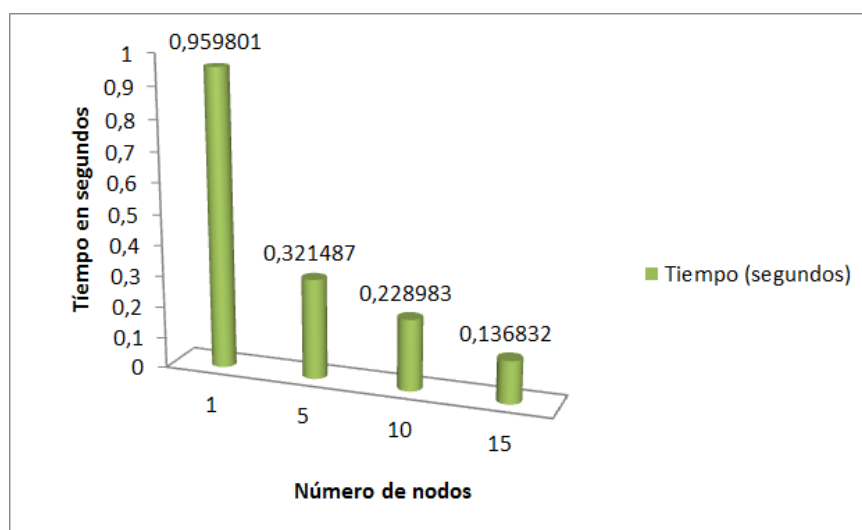


Figura 10. Tiempos de ejecución cálculo del valor aproximado de π utilizando MPI.

Interpretación de los resultados.

De la prueba realizada se deduce que utilizando un solo nodo, el tiempo de ejecución es de 0.959801 segundos, mientras que al utilizar los 15 nodos de la arquitectura el tiempo es de 0.136832 segundos, obteniendo una disminución de 0.822969 segundos de diferencia, lo que se interpreta como una eficiencia de aproximadamente 88%.

Proyecto 3: Pruebas de compresión de música con Mosix.

TABLA VIII. RESULTADO DE LA EJECUCIÓN, PRUEBAS DE COMPRESIÓN DE MÚSICA CON MOSIX

| Número de Nodos | Número de procesadores | Tiempo (min) |
|-----------------|------------------------|----------------|
| 1 | 8 | 22 min 16.065s |
| 5 | 40 | 11 min 23.602s |
| 10 | 80 | 09 min 34.572s |
| 15 | 120 | 07 min 45.562s |

En la Fig. 10 (*Tiempo de ejecución de pruebas de compresión de música con Mosix*) se ilustra de manera gráfica los resultados obtenidos al ejecutar la aplicación en los distintos nodos.

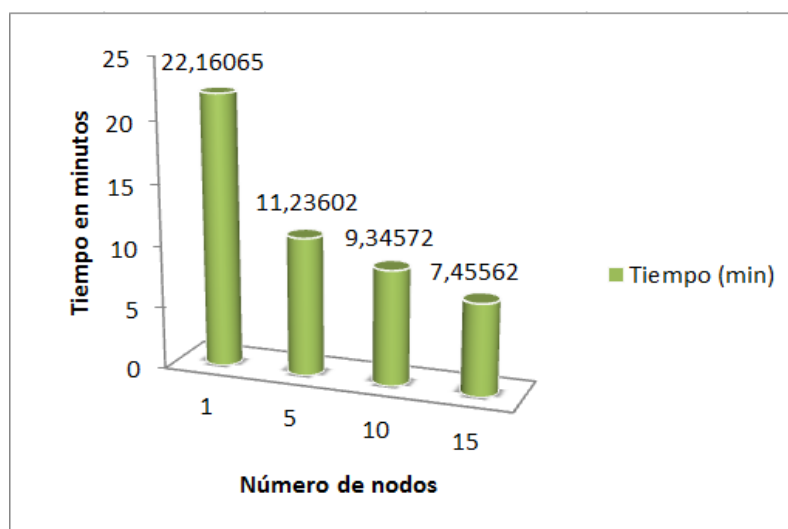


Figura 11. Tiempo de ejecución de pruebas de compresión de música con Mosix.

Interpretación de los resultados.

De la prueba realizada con el middleware Mosix para la compresión de los ficheros de audio se deduce que utilizando un solo nodo, el tiempo de ejecución es de 22.16065 minutos, mientras que al utilizar los 15 nodos de la arquitectura el tiempo es de 7.45562 minutos, obteniendo una disminución de 14.70503 de diferencia, lo que se interpreta como una eficiencia de aproximadamente 75%.

Proyecto 4: Cifrado de códigos con John The Ripper (JTR) y MPI.

TABLA IX. RESULTADO DE LA EJECUCIÓN DE CIFRADO DE CÓDIGO UTILIZANDO JOHN THE RIPPER.

| Número de nodos | Número de procesadores | Tiempo (hh:min:seg) |
|-----------------|------------------------|---------------------|
| 1 | 8 | 09:48:34 |
| 5 | 40 | 07:21:22 |
| 10 | 80 | 03:06:57 |
| 15 | 120 | 00:27:24 |

En la Fig. 12 (*Tiempo de ejecución de cifrado de código utilizando John The Ripper*) se ilustra de manera gráfica los resultados obtenidos al ejecutar la aplicación en los distintos nodos.

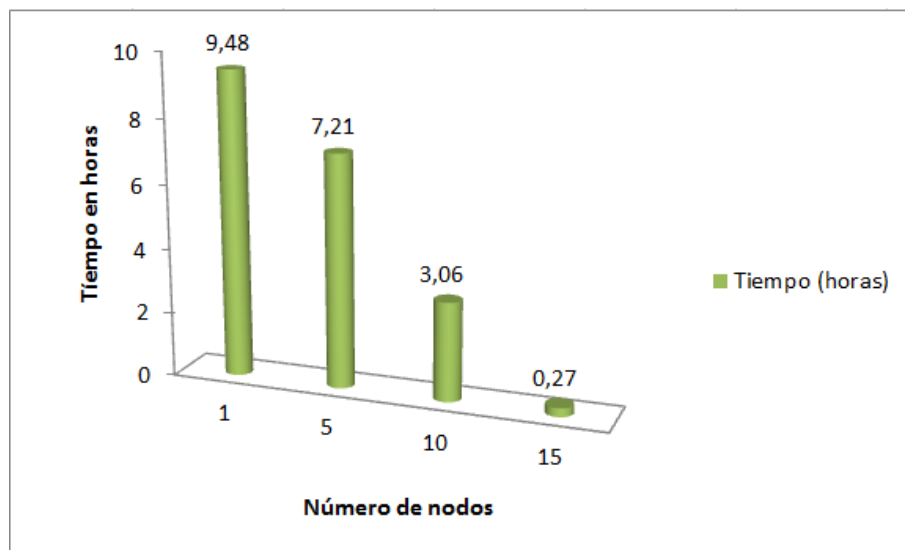


Figura 12. Tiempo de ejecución de cifrado de código utilizando John The Ripper.

Interpretación de los resultados.

De la prueba realizada para descifrar el fichero shadow se deduce que utilizando un solo nodo, el tiempo de ejecución es de 9.48 horas, mientras que al utilizar los 15 nodos de la arquitectura el tiempo es de 27 minutos, obteniendo una disminución de 9.21 horas de diferencia, lo que se interpreta como una eficiencia de aproximadamente 97%.

Proyecto 5: Renderización de imágenes y videos con Blender.

TABLA X. RESULTADO DE LA EJECUCIÓN DE RENDERIZACIÓN DE IMÁGENES Y VIDEOS CON BLENDER.

| Número de Nodos | Número de procesadores | Tiempo (min) |
|-----------------|------------------------|---------------|
| 1 | 8 | 15 min 40 seg |
| 5 | 40 | 09 min 13 seg |
| 10 | 80 | 04 min 06 seg |
| 15 | 120 | 01 min 31 seg |

En la Fig. 13 (*Tiempo de ejecución de renderización de imágenes y videos con blender*) se ilustra de manera gráfica los resultados obtenidos al ejecutar la aplicación en los distintos nodos.

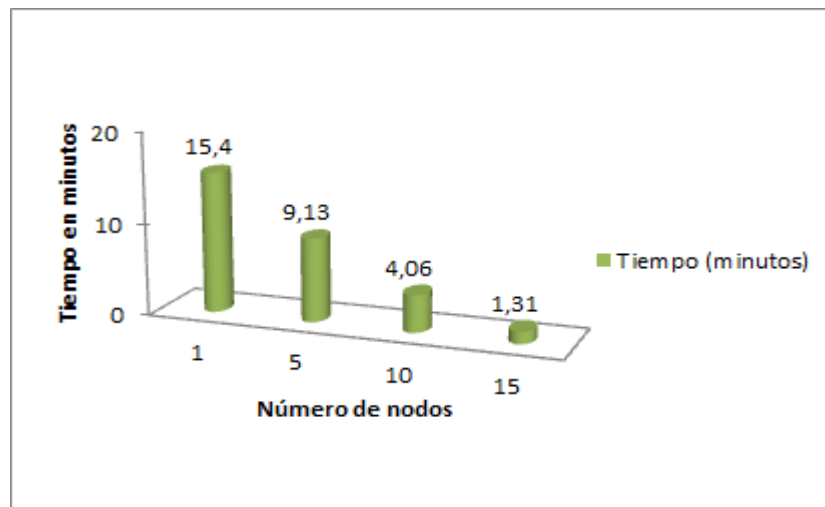


Figura 13. Tiempo de ejecución de renderización de imágenes y videos con blender.

Interpretación de los resultados.

De la prueba realizada haciendo uso de blender para el renderizado, se deduce que utilizando un solo nodo, el tiempo de ejecución es de 15.4 minutos, mientras que al utilizar los 15 nodos de la arquitectura el tiempo es de 1.31 minutos, obteniendo una disminución de 14.09 de diferencia, lo que se interpreta como una eficiencia de aproximadamente 92%.

Proyecto 6: Análisis de datos (Big Data) con Apache Hadoop.

TABLA XI. RESULTADO DE LA EJECUCIÓN DE ANÁLISIS DE DATOS (BIG DATA) CON HADOOP.

| Número de Nodos | Número de procesadores | Tiempo (seg) |
|-----------------|------------------------|--------------|
| 1 | 8 | 92 seg |
| 5 | 40 | 78 seg |
| 10 | 80 | 54 seg |
| 15 | 120 | 32 seg |

En la Fig. 14 (*Tiempo de ejecución de análisis de datos (Big Data) con Hadoop*) se ilustra de manera gráfica los resultados obtenidos al ejecutar la aplicación en los distintos nodos.

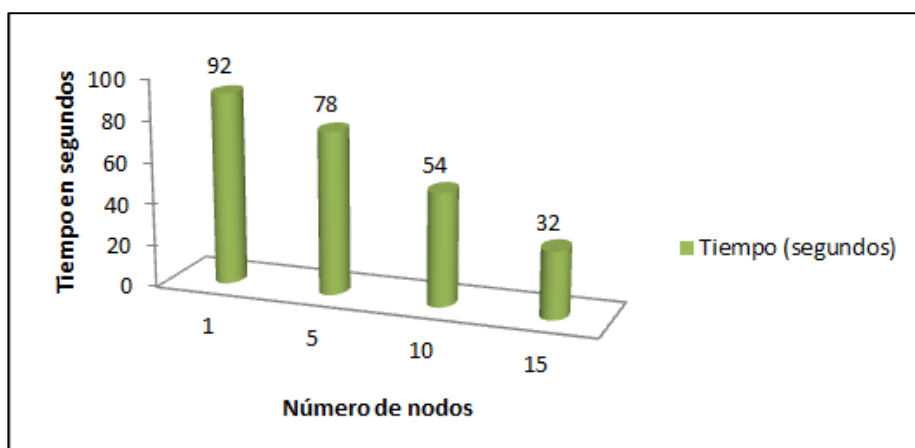


Figura 14. Tiempo de ejecución de análisis de datos (Big Data) con Hadoop.

Interpretación de los resultados.

Utilizando el framework Apache Hadoop para el análisis de datos (Big Data), se deduce que utilizando un solo nodo, el tiempo de ejecución es de 1 minuto con 32 seg, mientras que al utilizar los 15 nodos de la arquitectura el tiempo es de 32 segundos, obteniendo una disminución de 60 segundos de diferencia, lo que se interpreta como una eficiencia de aproximadamente 74% de aceleración de los procesos relacionados al análisis de datos.

3.3 Elaborar un artículo científico acorde a las normas IEEE

Como apoyo a la comunidad académica - científica se elaboró el artículo titulado “**Arquitectura clúster de alto rendimiento utilizando herramientas de software libre**”, que contiene principalmente temas como: resumen, introducción, estado del arte, metodología, implementación, resultados y conclusiones del proyecto. Los lineamientos que rigen el artículo son acordes a las normas IEEE.

Arquitectura clúster de alto rendimiento utilizando herramientas de software libre

Cluster architecture of high performance using free software tools

L. Chuquiguanca and E. Malla.

Carrera de Ingeniería en Sistemas, Universidad Nacional de Loja, Ciudad Universitaria
Guillermo Falconí Espinosa “La Argelia” Loja, Ecuador
{lchuquihuancav, ejmallab}@unl.edu.ec

Resumen— El artículo presenta la implementación de una arquitectura clúster que convierte un laboratorio inicialmente concebido para la enseñanza de la comunidad estudiantil, en una sala de altas prestaciones mediante el uso de herramientas de software libre como: MPICH, Mosix, Blender, Hadoop, Ganglia Monitoring System. Con el clúster se obtiene un rendimiento similar al de un supercomputador, optimizando el tiempo de procesamiento en: simulación de procesos, reconocimiento de patrones, renderizado de imagen y video, análisis de grandes volúmenes de datos Big Data, cifrado de códigos, evaluación de Algoritmos, etc. Además, con esta arquitectura se puede procesar millones de operaciones por segundo en los procesos antes mencionados. Este importante avance facilita la investigación académica en otros campos donde se requiera procesar grandes volúmenes de información y obtener resultados en un corto tiempo lo cual permitirá una adecuada toma de decisiones.

Palabras clave- Clúster, Middleware, Mosix, MPICH, Beowulf.

Abstract— The paper presents the implementation of a cluster architecture that initially becomes a laboratory designed to teach the student community, in a high performance room through the use of free software tools such as: MPICH, Mosix, Blender, Hadoop, Ganglia Monitoring System. With the cluster we obtained a similar performance like a supercomputer optimizing the processing time in: simulation process, pattern recognition, image and video rendering, analysis of big data volumes, encryption codes, assessment of algorithms, etc. In addition, with this architecture can be processed millions of operations per second in the above-mentioned processes. This advance facilitates academic research in other fields which require processing large volumes of information and results in a short time which will allow an adequate decision making.

Keywords— Cluster, Middleware, Mosix, MPICH, Beowulf.

I. INTRODUCCIÓN

Multitud de aplicaciones dentro de la investigación científica requieren de una gran demanda de potencia de cómputo que solo pueden ser cubiertas por supercomputadores, que por su alto costo no pueden ser adquiridas por Instituciones de Educación Superior y centros de investigación. Una de las soluciones más fiables comparadas a la adquisición de supercomputadores es la implementación de un clúster de alto rendimiento, formado por hardware convencional y herramientas de software libre que unidos a una red de alta velocidad [1], [2], [3] ofrecen ventajas significativas en términos económicos y de escalabilidad comparadas al utilizar un único ordenador convencional, proporcionando así resultados con un margen de error mínimo en tiempos relativamente bajos [4], [5].

La implementación de la arquitectura clúster se realizó en un laboratorio de la Carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja, formado por 15 computadores personales con características homogéneas; las principales configuraciones que se realizaron son: instalación y configuración del sistema operativo Debian “Wheezy”, configuración e instalación del middleware Mosix [6], [7] en el kernel de Linux para obtener una imagen única de sistema orientada a la computación distribuida, para la programación paralela se utilizó MPICH [8], como una de las aplicaciones más implementadas del estándar Message Passing Interface (MPI). Para el proceso de renderizado se usó el software Blender [9]; para el análisis de datos (Big Data) [10], se implementó el framework Apache Hadoop [11]. Además se configuró el acceso remoto utilizando claves públicas a través del protocolo Secure Shell (SSH), para la configuración y compartición del sistema de ficheros en red se utilizó el

protocolo Network File System (NFS) que trabaja a nivel de capa de aplicación según el modelo OSI, se realizó el diseño de la topología de red y la configuración del sistema de monitorización con Ganglia Monitoring System [12].

La estructura del artículo es la siguiente: la Sección II presenta como estado del arte, material bibliográfico sintetizado referente a arquitecturas clúster y la computación de alto rendimiento. La Sección III detalla la metodología de resolución de problemas utilizada para el desarrollo del proyecto. La Sección IV describe el proceso de implementación de la arquitectura clúster. En la Sección V se presentan los resultados obtenidos en diversas pruebas utilizando herramientas de software libre. Finalmente, la Sección VI se establece las conclusiones a las que se ha llegado al término del proyecto.

II. ESTADO DEL ARTE

A. Arquitectura clúster de alto rendimiento

Un clúster es un conjunto de nodos de bajo costo conectados entre sí a través de una red de comunicaciones de alta velocidad, que operan bajo software que actúa como un sistema único de administración, responsable de distribuir las cargas de trabajo entre los nodos, de forma automática y transparente al usuario como si se tratara de un único ordenador [3], [4], [5].

De manera lógica, cada nodo del clúster está formado por hardware y software. El hardware está compuesto por las partes de un ordenador convencional. En cuanto al software, el nivel bajo corresponde al sistema operativo, el nivel medio consiste en las librerías de paralelización y el nivel alto está representado por la aplicación que se desea ejecutar; una aplicación se ejecuta en el nodo maestro, el sistema operativo y las librerías de paralelización se encargan de ejecutar copias de este programa en los nodos esclavos del clúster [6], [7], [8]. De las distintas arquitecturas disponibles, se ha elegido la arquitectura clúster de tipo beowulf, como se ilustra en la Fig. 1, la misma que está compuesta por hardware convencional y herramientas de software libre.

Un clúster de alto rendimiento es utilizado principalmente con fines académico-científicos, su objetivo principal es proporcionar altas prestaciones de capacidad de cómputo superior a los que pudiera ofrecer un ordenador común. Este tipo de arquitecturas son una alternativa a la utilización de grandes y costosas supercomputadoras [9], [10].

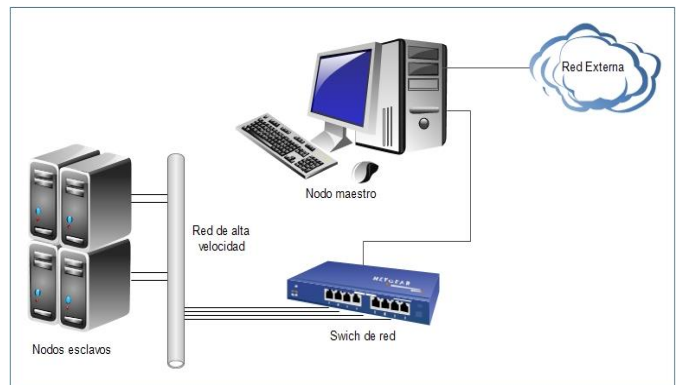


Fig 1. Clúster de computadoras formado por hardware convencional

B. Clasificación de los clústers

Existen dos tipos de clúster según la arquitectura de las computadoras que lo conforman:

Clúster homogéneo: Todos los nodos tienen las mismas características técnicas de hardware y software. Son idénticos y por lo tanto la capacidad de procesamiento y rendimiento de cada nodo es la misma.

Clúster heterogéneo: Al contrario de los clúster homogéneos, los nodos son completamente distintos en cuanto a hardware y software.

Esto conlleva a que las posibilidades de expansión del clúster crezcan de forma exponencial, debido a que es más fácil conseguir ordenadores con características distintas que con características similares.

C. Importancia de la computación de alto rendimiento

La programación paralela se origina por las limitaciones de las aplicaciones secuenciales; integrando varios procesadores para llevar a cabo sus funciones, la programación paralela permite resolver problemas que requieren más memoria o mayor velocidad de cómputo. También existen razones económicas, pues el precio de los ordenadores secuenciales no es proporcional a su capacidad computacional, mientras que la conexión de varios procesadores utilizando una red nos permite obtener un aumento de prestaciones prácticamente proporcional al número de procesadores con un coste adicional mínimo [13].

Además con el uso del clúster se logra reducir el tiempo de resolución de problemas computacionales, o bien resolver problemas que no cabrían en la memoria de un solo procesador secuencial. Y para esto es necesario utilizar sistemas de altas prestaciones y algoritmos paralelos que utilicen estos sistemas de manera eficiente [14].

III. METODOLOGÍA

Durante el desarrollo de esta investigación, se utilizó la metodología de resolución de problemas que se organiza en siete etapas descritas a continuación:

- 1. Identificación del problema.** Esta fase comprendió el estudio de la revisión bibliográfica y casos de éxito del funcionamiento de los clúster, en centros de investigación e instituciones de educación superior con la utilización de software libre y hardware convencional. Se analizó también la situación actual del laboratorio de cómputo de la Escuela de Ingeniería en Sistemas de la Universidad Nacional de Loja (CIS-UNL) para identificar si existían procesos que requieran procesar grandes volúmenes de información a altas velocidades y no se disponía de un supercomputador.
- 2. Explicación del problema.** En el laboratorio de la CIS de la UNL no existían procesos que requieran grandes velocidades de procesamiento pero por la falta de un sistema computacional de alto rendimiento los docentes y estudiantes estaban privados de realizar proyectos académicos relacionados con el procesamiento de grandes volúmenes de información y se desconocía que con la utilización de hardware convencional y software libre se podían obtener grandes beneficios económicos, comparados con la adquisición de supercomputadores que realicen tareas dedicadas a la computación de alto rendimiento.
- 3. Idear estrategias alternativas de intervención.** Para idear alternativas que permitieron solucionar el problema mencionado se realizó el análisis de los recursos técnicos de hardware, software y redes con las que cuenta el laboratorio de la CIS de la UNL, siendo el punto de partida para la implementación de la arquitectura clúster.
- 4. Decidir la estrategia.** Una vez realizado el análisis de la situación actual del laboratorio de la CIS de la UNL, se procedió a realizar la búsqueda de las herramientas de software libre, la elección del middleware, la

elección de la arquitectura clúster, el diseño de la topología y el direccionamiento de red.

- 5. Diseño de la intervención.** En esta fase se determinó los tiempos de implementación del proyecto para el laboratorio CIS de la UNL y las actividades a cumplir en los plazos establecidos, logrando con éxito la culminación del mismo.
- 6. Desarrollo de la intervención.** En esta fase se estableció la instalación y configuración de cada uno de los aplicativos de software libre mencionados anteriormente.
- 7. Evaluación de los logros.** La evaluación de los logros obtenidos se la realizó aplicando pruebas de procesamiento a distintos proyectos, logrando evidenciar tiempos mínimos de ejecución al utilizar los recursos de la arquitectura clúster.

IV. RESULTADOS

A. Recursos técnicos laboratorio

El laboratorio de la CIS-UNL cuenta con los siguientes recursos técnicos en cuanto a hardware y redes se refiere.

- **Recursos Hardware**

El laboratorio cuenta con 15 computadoras, con características homogéneas:

Hardware: Procesador Intel Corei7-2600 de 3.64 GHz, Memoria RAM de 4 GB de 1333 MHz, Disco Duro: SATA de 500 GB de 7200 rpm.

Sistema de energía ininterrumpida: Cada computador cuenta con su propio UPS TRIPP LITE AVR550U que proporciona salida de 550 VA o 120 V.

- **Recursos de Red**

Tarjeta PCI Adapter 802.11: Cantidad de Puertos: 1, soporta velocidades de 10/100/1000 Mbps, tipo de interfaz del sistema: PCI Express.

Router: CISCO LINKSYS EA4500: Doble banda 900N (2,4 GHz y 5 GHz), velocidad de transmisión de 450 – 450 Mbps, 4 Puertos Gigabit Ethernet.

Switch: CISCO Catalyst serie 2960: Switch de capa 2, 48 puertos, Gigabit Ethernet (10/100/1000) Mbps.

B. Diseño lógico de la arquitectura clúster

- **Arquitectura**

En la Fig. 2 se visualiza la arquitectura clúster utilizada, la misma que cuenta con las siguientes capas:

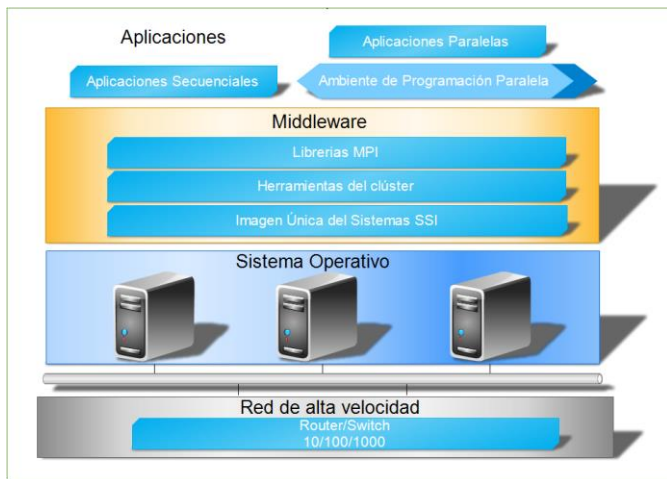


Fig 2. Diseño lógico arquitectura clúster de alto rendimiento CIS-UNL

Capa de aplicación: Permite la ejecución de aplicaciones secuenciales, aplicaciones paralelas (desarrolladas específicamente para la ejecución en el clúster) y ambiente de programación paralela (permite implementación de algoritmos que hacen uso de recursos compartidos: CPU, memoria, y servicios).

Capa de middleware: Es el software que generalmente actúa entre el sistema operativo y las aplicaciones con la finalidad de proveer:

- **Librerías MPI:** Enlaza los compiladores con el ambiente de programación paralela que se encuentra en la capa de aplicación.
- **Herramientas del clúster:** permiten su correcto funcionamiento: migración de procesos, checkpoint-restart (detener y restaurar uno o varios procesos, migrarlos a otro nodo) balanceo de carga entre los nodos, tolerancia a fallos, etc.
- **Imagen única del sistema SSI:** Da al usuario la sensación de hacer uso de un único computador.

Sistema operativo: Autoriza a los usuarios acceso unificado a los recursos del sistema.

Red de alta velocidad: Permite la comunicación entre los distintos nodos del clúster con tecnología de alta velocidad (Gigabit).

- **Topología**

En la Fig. 3, se visualiza la topología implementada, que es de tipo estrella. En esta topología no es necesario que los nodos esclavos tengan conexión a internet, por tal razón estos nodos pertenecen a una red privada, para poder hacer uso de este servicio, el nodo maestro cuenta con dos tarjetas de red, una para conectarse a la red LAN en la cual están enlazados todos los nodos que componen el clúster y la segunda tarjeta para tener conexión a Internet.

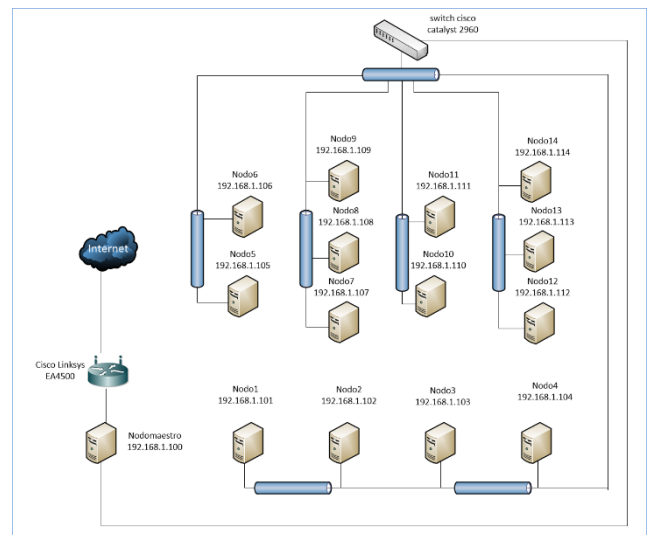


Fig 3. Diagrama de topología de la arquitectura clúster CIS-UNL

La Tabla I, detalla las direcciones IP asignadas de manera estática a cada uno de los nodos que componen el clúster, en un rango de direcciones de [192.168.1.100] hasta [192.168.1.114].

TABLA I. DIRECCIONAMIENTO IP CLÚSTER CIS-UNL

| Dirección de red | Máscara de Subred | Gateway | Nodo |
|------------------|-------------------|-------------|---------|
| 192.168.1.100 | 255.255.255.0 | 192.168.1.1 | Maestro |
| 192.168.1.101 | 255.255.255.0 | 192.168.1.1 | Nodo1 |
| 192.168.1.102 | 255.255.255.0 | 192.168.1.1 | Nodo2 |
| 192.168.1.103 | 255.255.255.0 | 192.168.1.1 | Nodo3 |
| 192.168.1.104 | 255.255.255.0 | 192.168.1.1 | Nodo4 |
| 192.168.1.105 | 255.255.255.0 | 192.168.1.1 | Nodo5 |
| 192.168.1.106 | 255.255.255.0 | 192.168.1.1 | Nodo6 |
| 192.168.1.107 | 255.255.255.0 | 192.168.1.1 | Nodo7 |

| | | | |
|---------------|---------------|-------------|--------|
| 192.168.1.108 | 255.255.255.0 | 192.168.1.1 | Nodo8 |
| 192.168.1.109 | 255.255.255.0 | 192.168.1.1 | Nodo9 |
| 192.168.1.110 | 255.255.255.0 | 192.168.1.1 | Nodo10 |
| 192.168.1.111 | 255.255.255.0 | 192.168.1.1 | Nodo11 |
| 192.168.1.112 | 255.255.255.0 | 192.168.1.1 | Nodo12 |
| 192.168.1.113 | 255.255.255.0 | 192.168.1.1 | Nodo13 |
| 192.168.1.114 | 255.255.255.0 | 192.168.1.1 | Nodo14 |

C. Herramientas de software libre

El software empleado en la arquitectura clúster se detalla a continuación:

- Sistema operativo: Debian 7.5 “Wheezy” (64 bits)
- Middleware Mosix 3.4.0.12
- Compiladores: GCC 4.5.2. Soporte para C, C++
- MPICH 3.1.2, como implementación del estándar Message Passing Interface (MPI)
- John The Ripper 1.7.9-jumbo-7
- Blender 2.72b
- Apache Hadoop 2.5.1
- Ganglia Monitoring System 3.3.8

D. Recursos totales de la arquitectura clúster

Una vez finalizada la implementación, se logró obtener una arquitectura clúster de alto rendimiento con la siguiente capacidad de procesamiento:

TABLA II. RECURSOS TOTALES DE LA ARQUITECTURA CLÚSTER

| Recursos | Descripción |
|--------------------------|-------------|
| Capacidad en memoria RAM | 60 GB |
| Número de procesadores | 120 |
| Capacidad en disco duro | 7.32 TB |

V. PRUEBAS

Se realizaron diversas pruebas en la arquitectura clúster, para determinar la capacidad de procesamiento alcanzado, ejecutando aplicaciones que requieren alta capacidad computacional. Para realizar las pruebas a los siguientes proyectos se utilizó 1, 5, 10 y 15 nodos respectivamente.

A. Proyecto 1: Cálculo del valor aproximado de pi utilizando librerías MPI

El proyecto evalúa el algoritmo utilizado para calcular el valor de pi, utilizando librerías de paso de mensajes MPI.

TABLA III. RESULTADOS CÁLCULO DEL VALOR APROXIMADO DE PI UTILIZANDO MPI

| Número de nodos | Número de procesadores | Tiempo de ejecución (seg) |
|-----------------|------------------------|---------------------------|
| 1 | 8 | 0.959801 |
| 5 | 40 | 0.321487 |
| 10 | 80 | 0.228983 |
| 15 | 120 | 0.136832 |

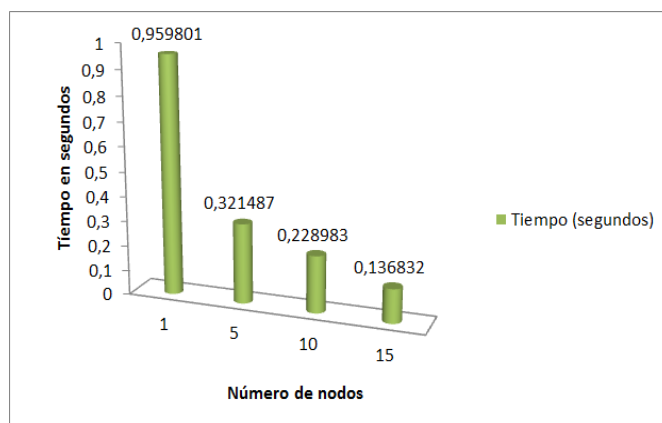


Fig 4. Tiempos de ejecución cálculo del valor aproximado de pi.

De la prueba realizada se deduce que utilizando un solo nodo, el tiempo de ejecución es de 0.959801 segundos, mientras que al utilizar los 15 nodos de la arquitectura el tiempo es de 0.136832 segundos, como se observa en la Tabla. III, obteniendo una disminución de 0.822969 segundos de tiempo de procesamiento, lo que se interpreta como una eficiencia de aproximadamente 88%, como se ilustra en la Fig. 4.

B. Proyecto 2: Cifrado de códigos con John The Ripper (JTR) y MPI.

Las pruebas se realizaron sobre uno de los ficheros más importantes de Linux como lo es shadow, este fichero almacena información cifrada de las contraseñas de cada una de las cuentas de usuario del sistema operativo. La contraseña utilizada tiene una longitud de 8 símbolos, y consta de 62 combinaciones de caracteres (26 letras del abecedario mayúsculas + 26 letras del abecedario minúsculas + 10 dígitos), por lo que se realizaran cerca de 218 trillones de posibles combinaciones.

TABLA IV. RESULTADOS CIFRADO DE CÓDIGOS CON JOHN THE RIPPER (JTR) Y LIBRERÍAS MPI

| Número de nodos | Número de procesadores | Tiempo de ejecución (horas) |
|-----------------|------------------------|-----------------------------|
| 1 | 8 | 09:48:34 |
| 5 | 40 | 07:21:22 |
| 10 | 80 | 03:06:57 |
| 15 | 120 | 00:27:24 |

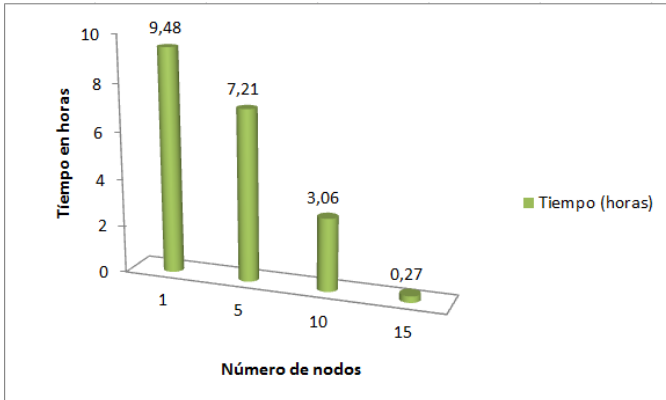


Fig 5. Tiempo de ejecución de cifrado de código utilizando John The Ripper

El tiempo de ejecución al descifrar el fichero shadow, en un solo nodo es de 9.48 horas, mientras que al utilizar los 15 nodos de la arquitectura el tiempo de la decodificación es de 27 minutos, como se detalla en la Tabla IV obteniendo una disminución de 9.21 horas de tiempo de procesamiento, lo que se interpreta como una eficiencia de aproximadamente 97%, ver Fig. 5. En la Fig. 6, se observa el balanceo de carga al ejecutar el proceso de cifrado de código utilizando John The Ripper y librerías MPI en los 15 nodos de la arquitectura clúster.

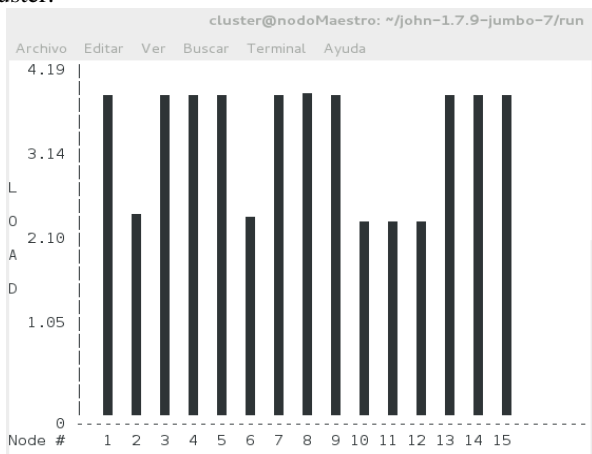


Fig 6. Balanceo de carga en el clúster al ejecutar MPICH+JTR

C. Proyecto 3: Renderización de imagen y video con Blender

Este proyecto trata sobre el armado final de un cubo de rubik que está formado por 115 frames, una vez terminado el renderizado se obtiene como producto final, un video con una duración aproximada de 11 segundos y un tamaño total de 18.5 MB.

TABLA V. RESULTADOS RENDERIZACIÓN DE IMÁGENES Y VIDEOS CON BLENDER

| Número de nodos | Número de procesadores | Tiempo de ejecución (min) |
|-----------------|------------------------|---------------------------|
| 1 | 8 | 15:40 |
| 5 | 40 | 09:13 |
| 10 | 80 | 04:06 |
| 15 | 120 | 01:31 |

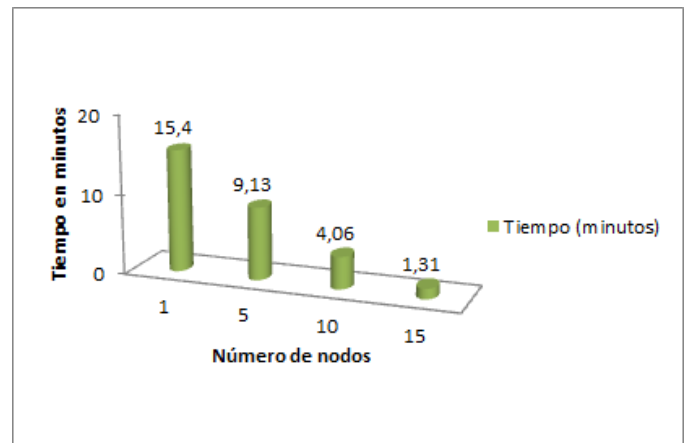


Fig 7. Tiempo de ejecución de renderización de imágenes y videos con Blender.

El tiempo de ejecución utilizando Blender para renderizar el proyecto en un solo nodo es de 15.4 minutos, en cambio utilizando los 15 nodos de la arquitectura el tiempo de renderizado es de 1.31 minutos, ver Tabla V, obteniendo una disminución de 14.09 minutos de tiempo de procesamiento, alcanzando un rendimiento de aproximadamente 92%, como se observa en la Fig. 7.

D. Proyecto 4: Análisis de datos (Big Data) con Apache Hadoop

Para esta prueba se utilizó un dataset sobre la medición de la calidad del aire en España en las estaciones de Castilla y León, en el que se encuentra información recopilada desde el año 1997 hasta el año 2013, el mismo que se encuentra disponible en el siguiente enlace <http://goo.gl/jZO6OT>. El fichero de extensión .csv, fue modificado a nuestros requerimientos con

lo que se logró obtener un archivo que contiene más de 16 millones de líneas y un peso mayor a 950 Mb.

TABLA VI. RESULTADOS DE ANÁLISIS DE DATOS (BIG DATA) CON HADOOP.

| Número de nodos | Número de procesadores | Tiempo de ejecución (seg) |
|-----------------|------------------------|---------------------------|
| 1 | 8 | 92 |
| 5 | 40 | 78 |
| 10 | 80 | 54 |
| 15 | 120 | 32 |

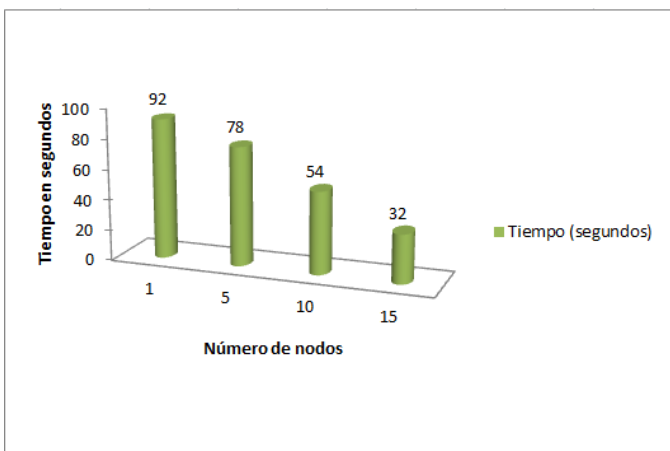


Fig 8. Tiempo de ejecución de análisis de datos (Big Data) con Hadoop

Utilizando el framework Apache Hadoop para el análisis de datos (Big Data), se deduce que utilizando un solo nodo, el tiempo de ejecución es de 1 minuto con 32 seg, mientras que al utilizar los 15 nodos de la arquitectura el tiempo es de 32 segundos de tiempo de procesamiento, como se detalla en la Tabla VI obteniendo una disminución de 60 segundos de diferencia, lo que se interpreta como una eficiencia de aproximadamente 74% de aceleración de los procesos relacionados al análisis de datos, ver Fig. 8.

E. Proyecto 5: Pruebas de compresión de música con MOSIX

En el proyecto realiza la compresión de 32 canciones en formato .wav a un formato de archivos de audio .flac, haciendo uso de Mosix para la migración automática de los procesos en cada uno de los nodos del clúster. El tamaño de los archivos antes de realizar la compresión es de 5.2 Gb, luego de realizar el proceso el tamaño es de 1.9 Gb.

TABLA VII. RESULTADO DE LA EJECUCIÓN, PRUEBAS DE COMPRESIÓN

| Número de nodos | Número de procesadores | Tiempo de ejecución (min) |
|-----------------|------------------------|---------------------------|
| 1 | 8 | 22:16 |
| 5 | 40 | 11:23 |
| 10 | 80 | 09:34 |
| 15 | 120 | 07:45 |

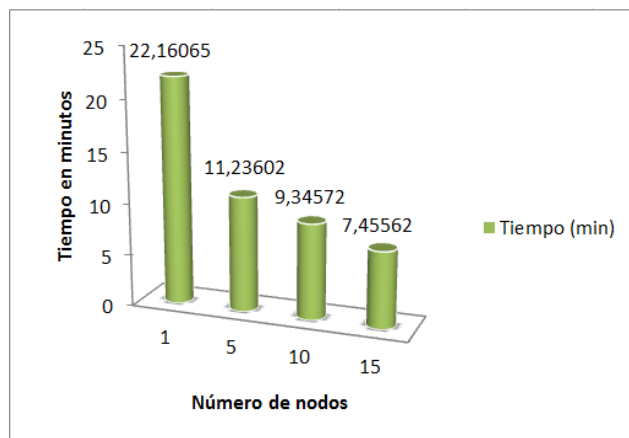


Fig 9. Tiempo de ejecución de pruebas de compresión de música con Mosix

De la prueba realizada con el middleware mosix para la compresión de los ficheros de audio se deduce que utilizando un solo nodo, el tiempo de ejecución es de 22.16065 minutos, mientras que al utilizar los 15 nodos de la arquitectura el tiempo es de 7.45562 minutos, ver Tabla VII, obteniendo una disminución de 14.70503 tiempo de procesamiento, lo que se interpreta como una eficiencia de aproximadamente 75%, ver Fig. 9.

F. Monitorización de la arquitectura clúster

Gracias al sistema de monitoreo Ganglia Monitoring System, tenemos registradas las actividades de cada uno de los nodos de la arquitectura clúster, como se observa en la Fig. 10 permitiendo recolectar métricas como: ocupación de los procesadores, uso de memoria, espacio en disco, etc [15].

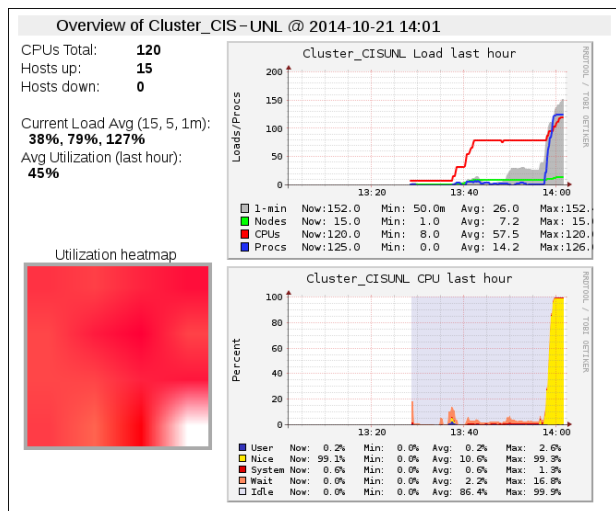


Fig 10. Monitorización arquitectura clúster de alto rendimiento CIS-UNL

VI. CONCLUSIONES Y TRABAJOS FUTUROS

De acuerdo a las especificaciones técnicas de cada uno de los equipos informáticos con los que cuenta el laboratorio de la CIS, la elección de la arquitectura clúster tipo beowulf es la más óptima ya que se implementó utilizando hardware convencional y herramientas de software libre, con lo que se obtiene un ahorro generalizado en costes de administración, mantenimiento y monitorización, comparadas con la adquisición de un supercomputador.

Al realizar la ejecución de varios tipos de algoritmos complejos como: cifrado de código, renderización de imágenes, análisis de datos (Big Data), compresión de ficheros de audio, la arquitectura clúster alcanzó un rendimiento del 85.2%, en relación a la capacidad de procesamiento de un solo computador convencional.

En la arquitectura clúster se puede ejecutar una amplia gama de aplicaciones de cómputo científico, para la evaluación de métodos numéricos y técnicas de optimización, solucionando problemas principalmente de ciencia e ingeniería.

La arquitectura clúster de alto rendimiento está diseñada para proporcionar capacidad de procesamiento de grandes volúmenes de datos, lo que permite realizar investigaciones en campos como: minería de datos, Big Data, sistemas de gestión de base de datos, evaluación de estructuras de datos, evaluación de algoritmos, programación paralela, reconocimiento de patrones, entre otros.

VII. AGRADECIMIENTOS

Agradecemos a la Universidad Nacional de Loja, el Área de la Energía, las Industrias y los Recursos Naturales no Renovables, y la Carrera de Ingeniería en Sistemas por brindarnos el apoyo para la culminación exitosa del proyecto.

REFERENCIAS

- [1] G. Cáceres, "Estrategia de implementación de un clúster de alta disponibilidad de N nodos sobre linux usando software libre", 2012.
- [2] L. M. Santos Jaimes, S. Peñaloza, and E. R. Cruz Cruz, "Cluster implementation of a prototype for the resolution of a particular problem", Journal Article, vol. 1 2010.
- [3] N. Pérez Otero, S. Méndez, C. V. Ayusa, M. I. Aucapiña, and V. J. Lopez, "Aplicaciones del cómputo de altas prestaciones", in XI Workshop of Investigadores en Ciencias de la Computación, 2013.
- [4] D. Zhao, K. Qiao, and I. "Raicu, HyCache+: Towards Scalable High-Performance Caching Middleware for Parallel File Systems", in IEEE/ACM CCGrid, 2014.
- [5] J. d. J. R. Quezada, S. B. Rionda, J. M. V. Félix, and I. A. M. Torres, "Diseño e implementación de un clúster de cómputo de alto rendimiento", Acta Universitaria, vol. 21, pp. 24-33, 2011.
- [6] A. Barack and A. Shiloh, "The MOSIX Cluster Operating System for Distributed Computing on Linux Clusters, Multi-Clusters and Clouds", 2014.
- [7] M. C. O. System, "Administrator's, User's and Programmer's Guides and Manuals", July 2014.
- [8] R. Latham and A. J. Pe, "MPICH Installer's Guide", Mathematics and Computer Science Division Argonne National Laboratory, 2014.
- [9] Blender, "Blender is a free and open source 3D animation suite" [Online]. Available: <http://www.blender.org/>. [Accessed: 05-Nov-2014].
- [10] R. Serrat Morros, "Big Data: análisis de herramientas y soluciones", 2013.
- [11] Apache, "Hadoop - Apache Hadoop 2.5.1." [Online]. Available: <http://hadoop.apache.org/docs/r2.5.1/index.html>. [Accessed: 13-Nov-2014].
- [12] G. M. System, "Ganglia Monitoring System" [Online]. Available: <http://ganglia.sourceforge.net/>. [Accessed: 04-Dec-2014].
- [13] B. Otero, R. Astudillo, and Z. Castillo, "Un esquema paralelo para el cálculo del pseudoespectro de matrices de gran magnitud," Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería, 2014.
- [14] A. Sheharyar and O. Bouhali, "A Framework for Creating a Distributed Rendering Environment on the Compute Clusters", arXiv preprint arXiv:1401.0608, 2014
- [15] R. Bhatnagar and J. Patel, "Performance Analysis of A Grid Monitoring System-Ganglia", International Journal of Emerging Technology and Advanced Engineering, vol. 3, pp. 362-365, 2013.



Leonardo Chuquiguanca, Egresado de la Carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja, conocedor de software libre, administración de servidores, redes, telecomunicaciones, seguridad de información. Provincia de Loja, Ciudad Loja, Ecuador, 2015.



Edyson Malla, Egresado de la Carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja, conocedor de software libre, redes, telecomunicaciones, análisis y diseño de sistemas. Provincia de Loja, Ciudad Loja, Ecuador, 2015

g. Discusión

1. Desarrollo de la propuesta alternativa

El desarrollo de la propuesta alternativa se basa en la realización y cumplimiento de los objetivos planteados.

1.1 Analizar las plataformas clúster para determinar cuál es la más óptima de acuerdo a los equipos y requerimientos existentes.

Para el cumplimiento de este objetivo, se realizó la búsqueda, selección y análisis de casos de éxito en los cuatro últimos años sobre la implementación de un clúster de alto rendimiento utilizando herramientas de software libre; la búsqueda se efectuó en universidades del Ecuador, a nivel internacional y en centros de investigación científica, ver (*Sección Resultados Fase 1: 1.1 Revisión bibliográfica y casos de éxito del funcionamiento de los clúster en centros de investigación y en Instituciones de Educación Superior (IES)*).

Además se desarrolló una descripción detallada de los equipos con los que cuenta el laboratorio de la CIS de la UNL, en lo que se refiere a las características hardware y software, el mismo proceso de análisis se realizó con los equipos de networking con los que cuenta el laboratorio, ver (*Sección Resultados fase 1: 1.2 Análisis de los recursos de los equipos informáticos con los que cuenta actualmente la institución*).

Una vez concluido el análisis de los casos de éxito y la infraestructura tecnológica del laboratorio de la CIS de la UNL, se elaboró un análisis exhaustivo para la elección de las herramientas de software libre que más se adapten a los requerimientos existentes, los puntos clave para la elección correcta de las herramientas son: arquitectura soportada, última versión estable, requisitos mínimos de hardware y software, soporte, ver (*Sección Resultados Fase 1: 1.3 Seleccionar la plataforma de software libre que más se adapte a los requerimientos de los equipos informáticos existentes*).

1.2 Implementar una plataforma clúster para integrar varias computadoras independientes.

Dentro de este apartado se realizó el diseño de topología de red propuesto para la implementación de la arquitectura clúster, así como la configuración con un direccionamiento IP estático en el laboratorio, con la finalidad de sustituir el direccionamiento dinámico DHCP, para evitar reconfiguraciones posteriores, ver (*Sección Resultados Fase 2: 2.1 Diseño lógico de la arquitectura clúster en el laboratorio de la CIS*).

La implementación del clúster se la llevó a cabo con el sistema operativo GNU/Linux Debian "Wheezy" por su estabilidad y su funcionamiento como servidor de alto rendimiento, usado como estándar para el desarrollo del proyecto. Para el middleware se utilizó Mosix como gestor de procesos para obtener una imagen única de sistema orientada a la computación distribuida. Para la programación paralela se empleó MPICH como una de las aplicaciones más implementadas del estándar Message Passing Interface. Para el proceso de renderizado se hizo uso del software Blender, para análisis de Big Data se empleó el framework Apache Hadoop. Dentro de las configuraciones que se realizaron están: el acceso remoto utilizando claves públicas a través del protocolo SSH, la compartición del sistema de ficheros en red a través del protocolo NFS, además la instalación y configuración del sistema de monitorización Ganglia Monitoring System, ver (*Manual técnico de instalación y configuración del clúster de alto rendimiento utilizando de herramientas de software libre*).

1.3 Evaluar la funcionalidad del clúster a través de procesos con grandes cantidades de procesamiento computacional.

La evaluación de los logros obtenidos se la realizó con seis proyectos considerando la cantidad de procesamiento y el número de nodos utilizados, ver (*Sección Fase 3: Resultados*), con la aplicación de testeo de rendimiento Stress-Test se obtuvo una eficiencia del 85%, en el proyecto sobre el cálculo del valor aproximado de π utilizando librerías MPI se logró un 88% de eficiencia, en el proyecto sobre pruebas de compresión de música con Mosix se obtuvo una eficiencia de 75%, en el proyecto cifrado de códigos con John The Ripper (JTR) y librerías MPI se alcanzó un 97% de efectividad, en el proyecto de renderización de imágenes y videos con Blender se

logró un 92% de efectividad, en el proyecto de análisis de datos (Big Data) con Apache Hadoop se alcanzó un 74% de eficiencia.

De los proyectos realizados en la arquitectura clúster se deduce que el porcentaje de eficiencia corresponde a un 85.16%, de acuerdo a las aplicaciones ejecutadas.

1. Valoración técnica, económica, ambiental

El Trabajo de Titulación denominado “Arquitectura clúster de alto rendimiento utilizando herramientas de software libre”, se considera desde el punto de vista técnico como un trabajo factible y necesario dentro de la Carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja, puesto que servirá como alternativa para las futuras investigaciones académicas-científicas que se desarrollen en la CIS, además de servir como apoyo a las distintas asignaturas que se imparten en la misma.

Gracias a la arquitectura clúster, se podrá ejecutar aplicaciones obteniendo resultados en el menor tiempo posible. De este modo se ahorrará tiempo y dinero ya que la arquitectura implementada está formada por hardware convencional y herramientas de software libre.

La valoración económica del proyecto tiene su base en que el desarrollo del mismo se ajusta a los intereses de la institución, puesto que el software utilizado tiene licencia GPL, mientras que el hardware utilizado son los equipos tecnológicos con los que cuenta el laboratorio de la CIS.

El talento humano que participó en el desarrollo del Trabajo de Titulación, está conformado principalmente por los investigadores quienes fuimos los encargados de desarrollar a cabalidad el proyecto, el docente de la materia de anteproyecto de tesis quien fue el guía para la elaboración del anteproyecto y la memoria final, y el director del trabajo de titulación. La Tabla XII (*Recursos Humanos*) detalla el tiempo y costo asignado a los investigadores, docente de anteproyecto de tesis, y director del proyecto, responsables de la culminación exitosa del mismo.

TABLA XII. RECURSOS HUMANOS.

| DESCRIPCIÓN | CANTIDAD | COSTO/HORA | HORAS | COSTE TOTAL |
|-------------------------------|----------|------------|-------|-------------|
| Investigadores | 2 | \$ 10.00 | 400 | \$ 4000.00 |
| Docente Anteproyecto de Tesis | 1 | \$ 00.00 | 64 | \$ 00.00 |
| Director de Tesis | 1 | \$ 00.00 | 200 | \$ 00.00 |
| TOTAL | | | | \$ 4000.00 |

La Tabla XIII (*Recursos Materiales*) presenta una descripción detallada de los recursos materiales que fueron necesarios para la presentación de los avances y el informe final del proyecto.

TABLA XIII. RECURSOS MATERIALES.

| DESCRIPCIÓN | CANTIDAD | V. UNITARIO | V. TOTAL |
|--------------------|----------|-------------|-----------|
| Resma de papel A4 | 1 | \$ 6.00 | \$ 6.00 |
| Cartuchos de tinta | 3 | \$ 20.00 | \$ 60.00 |
| Empastado | 3 | \$ 10.00 | \$ 30.00 |
| Anillados | 3 | \$ 2.50 | \$ 7.50 |
| CD's | 3 | \$ 1.00 | \$ 3.00 |
| TOTAL | | | \$ 106.50 |

La Tabla XIV (*Recursos Hardware*) detalla los recursos hardware utilizados en el desarrollo del TT; que comprende dos computadores portátiles usados para la implementación virtual del Clúster de alto rendimiento, para luego implementarlo en el laboratorio de la CIS, la redacción de los avances e informe final del mismo.

TABLA XIV. RECURSOS HARDWARE.

| DESCRIPCIÓN | COSTO | DEPRECIACIÓN (3 años) | T. UTILIZACIÓN (horas) | TOTAL |
|--------------------------|-----------|--------------------------|---------------------------|-----------|
| Portátil HP Pavilion dv6 | \$ 850.00 | \$ 161 | 400 | \$ 214.67 |
| Portátil DELL INSPIRON | \$ 900.00 | \$ 214 | 400 | \$ 285.33 |
| Impresora | \$ 80.00 | \$ 16 | 400 | \$ 21.23 |
| TOTAL | | | | \$ 521.33 |

La Tabla XV (*Recursos Software*) muestra las herramientas de software libre utilizadas en la implementación del proyecto, por tratarse de herramientas con licencia GPL.

TABLA XV. RECURSOS SOFTWARE.

| DESCRIPCIÓN | VALOR |
|------------------|----------|
| GNU/Linux Debian | \$ 00.00 |
| Mosix | \$ 00.00 |
| Mpich | \$ 00.00 |
| Blender | \$ 00.00 |
| Apache Hadoop | \$ 00.00 |
| Ganglia | \$ 00.00 |
| TOTAL | \$ 00.00 |

En la Tabla XVI (*Recursos técnicos y tecnológicos*) se aprecia la suma parcial de los recursos técnicos y tecnológicos, usados luego en la suma total de recursos usados.

TABLA XVI. RECURSOS TÉCNICOS Y TECNOLÓGICOS.

| DESCRIPCIÓN | VALOR TOTAL |
|-------------------|------------------|
| Recursos hardware | \$ 521.33 |
| Recursos software | \$ 00.00 |
| SUBTOTAL | \$ 521.33 |

La Tabla XVII (*Aproximación del costo real del TT*), ilustra la suma total de todos los recursos: humanos, materiales, técnicos y tecnológicos usados en el TT, que nos brinda una aproximación real del coste del proyecto.

TABLA XVII. APROXIMACIÓN DEL COSTO REAL DEL TT.

| DESCRIPCIÓN | VALOR TOTAL |
|----------------------------------|-------------------|
| Recursos humanos | \$ 4000.00 |
| Recursos materiales | \$ 106.50 |
| Recursos técnicos y tecnológicos | \$ 521.33 |
| Subtotal | \$ 4627.83 |
| Imprevistos (10%) | \$ 426.78 |
| TOTAL | \$ 5090.61 |

El Trabajo de Titulación se considera factible en el aspecto ambiental puesto que el clúster de alto rendimiento está formado por un cableado estructurado por lo que el impacto que genera en el ambiente es mínimo.

h. Conclusiones

- De acuerdo a las especificaciones técnicas de los equipos informáticos con los que cuenta el laboratorio de la CIS y una vez realizado el análisis de los casos de éxito, la elección de la arquitectura clúster tipo beowulf es la más óptima ya que fue implementada utilizando hardware convencional y herramientas de software libre.
- Para integrar los computadores independientes a la arquitectura clúster, se realizó el diseño de la topología de red tipo estrella con un direccionamiento IP estático, la configuración de protocolos de comunicación remota sin autenticación, la compartición de ficheros en red, y la utilización de aplicativos de licencia GPL que permiten obtener el funcionamiento similar al de un supercomputador.
- Utilizando librerías de paso de mensajes (MPI) y el software John the Ripper para el descifrado de contraseñas, la arquitectura clúster alcanza una eficiencia de 97% en rendimiento, en comparación con los recursos computacionales de un solo nodo.
- De los proyectos ejecutados en la arquitectura clúster como las pruebas de rendimiento, evaluación de algoritmos, compresión de archivos, cifrado de código, renderización de imágenes, análisis de datos (Big Data), se obtuvo un porcentaje promedio de eficiencia correspondiente al 85.16%, haciendo uso de los recursos disponibles de cada uno de los nodos, comparados al rendimiento de un computador convencional.
- Con la implementación de la arquitectura clúster, la CIS podrá dar soluciones potentes, eficientes y económicas a los entornos de investigación, mejorando la capacidad de procesamiento de las aplicaciones, en un tiempo relativamente bajo, ideales para las necesidades educativas en investigaciones actuales.

i. Recomendaciones

- Renovar periódicamente los repositorios de la distribución Debian en cada uno de los nodos de la arquitectura clúster, para mantener actualizados los paquetes y/o programas con el fin de para evitar bugs, problemas de seguridad y errores en ejecución, lo que permite mejorar el rendimiento al ejecutar las diferentes aplicaciones en la arquitectura clúster.
- Para tener mayor control sobre el paralelismo en aplicaciones que trabajen con threads y sean ejecutadas en una plataforma clúster, se requiere que el paralelismo sea implementado en su totalidad por el programador, ya sea utilizando librerías sobre memoria compartida (OpenMP) o utilizando librerías de paso de mensajes (MPI).
- La arquitectura clúster está diseñada para procesar grandes volúmenes de datos, por lo que se recomienda realizar investigaciones en campos como minería de datos, Big Data, sistemas de gestión de base de datos, evaluación de estructuras de datos, evaluación de algoritmos, programación paralela, reconocimiento de patrones, entre otras.
- Recomendamos la creación de una VLAN exclusiva para la utilización de la arquitectura clúster, además de segmentar la red con la finalidad de aumentar el rendimiento, tomando en cuenta que existe una única topología, un mismo protocolo de comunicación y un solo entorno de trabajo.
- Al tratarse de una arquitectura clúster de alto rendimiento que hace uso intensivo de las CPU's, recomendamos disponer de un sistema de ventilación ya que el exceso de calor afecta de manera negativa el rendimiento de los nodos y acorta su vida útil.

j. Bibliografía

Referencias Bibliográficas

- [1] I. Verona Ríos, "Instalación y configuración de un cluster de alto rendimiento," 2010.
- [2] J. d. J. R. Quezada, S. B. Rionda, J. M. V. Félix, and I. A. M. Torres, "Diseño e implementación de un clúster de cómputo de alto rendimiento," *Acta Universitaria*, vol. 21, pp. 24-33, 2011.
- [3] M. T. Ortega, "Adaptación a grid computing del módulo de cálculo de disponibilidad de satélites de uns sistema de navegación."
- [4] G. Molero Escobar, "Clúster de alto rendimiento en un cloud: ejemplo de aplicación en criptoanálisis de funciones HASH," 2012.
- [5] F. J. Fernández-Baldomero and M. Anguita, "Diseño de un cluster de computadores como actividad para Arquitectura de Computadores," *Suma*, vol. 10, p. 11.0.
- [6] G. Cáceres, "Estrategia de implementación de un clúster de alta disponibilidad de N nodos sobre linux usando software libre," 2012.
- [7] I. Luna and D. Haidee, "Implementación de un cluster de alta disponibilidad de bases de datos para la GEDGAPA," 2012.
- [8] J. M. Rodríguez Del Amo, "Clustering en Linux," 2013.
- [9] E. Montes de Oca, L. C. De Giusti, A. E. De Giusti, and M. Naiouf, "Comparación del uso de GPU y cluster de multicore en problemas con alta demanda computacional," in *XVIII Congreso Argentino de Ciencias de la Computación*, 2012.
- [10] M. C. O. System, "Administrator's, User's and Programmer's Guides and Manuals," July 2014.
- [11] A. Barack and A. Shiloh, "The MOSIX Cluster Operating System for Distributed Computing on Linux Clusters, Multi-Clusters and Clouds," 2014.
- [12] J. T. Palma, C. Garrido, F. Sánchez, and A. Quesada, "Programación Concurrente," *Thomson*, 2003.

- [13] F. Almeida, D. Giménez, J. M. Mantas, and A. M. Vidal, *Introducción a la programación paralela*: Paraninfo Cengage Learning, 2008.
- [14] B. Otero, R. Astudillo, and Z. Castillo, "Un esquema paralelo para el cálculo del pseudoespectro de matrices de gran magnitud," *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería*, 2014.
- [15] F. Leibovich, S. Gallo, A. E. De Giusti, L. C. De Giusti, F. Chichizola, and M. Naiouf, "Comparación de paradigmas de programación paralela en cluster de multicores: pasaje de mensajes e híbrido," in *XVII Congreso Argentino de Ciencias de la Computación*, 2011.
- [16] E. Rucci, "Comparación de modelos de sincronización en programación paralela sobre cluster de multicores," Universidad Nacional de La Plata, 2011.
- [17] D. Zhao, K. Qiao, and I. Raicu, "HyCache+: Towards Scalable High-Performance Caching Middleware for Parallel File Systems," in *IEEE/ACM CCGrid*, 2014.
- [18] J. Balladini, E. Grosclaude, M. Hanzich, R. Suppi, D. Rexachs del Rosario, and E. Luque Fadón, "Incidencia de los modelos de programación paralela y escalado de frecuencia de CPUs en el consumo energético de los sistemas de HPC," in *XVI Congreso Argentino de Ciencias de la Computación*, 2010.
- [19] D. Jiménez-González, "Introducción a las arquitecturas paralelas."
- [20] R. Serrat Morros, "Big Data: análisis de herramientas y soluciones," 2013.
- [21] J. S. Piñacol, "Big Data," *Universidad Politécnica de Catalunya - España*, 2013.
- [22] I. B. Machines, "Comment lines: Nine things that make the Liberty profile so fast, easy, and smart to use," *developerWorks*, 08 May 2013.
- [23] A. Hadoop, "Welcome to Apache Hadoop," 2014.
- [24] SerendipityMobile, "Hadoop ¿Qué arquitectura utiliza Hadoop?," 2014.
- [25] P. S. Pérez, *Manual de modelado y animación con Blender*. Universidad de Alicante, 2011.
- [26] A. Sheharyar and O. Bouhali, "A Framework for Creating a Distributed Rendering Environment on the Compute Clusters," *arXiv preprint arXiv:1401.0608*, 2014.
- [27] P. I. G. Varas and M. A. P. Enferriz, "Generación de entornos 3D a partir de mapas digitales," 2014.

- [28] R. M. Gualán, A. O. Vázquez, and O. F. Vega, "Una primera aproximación a la implementación de un clúster para la ejecución de un modelo de predicción climática," 2012.
- [29] F. J. P. Rondón and E. Valencia, "CLUSTER MANGOSTA: Implementación y evaluación," *Faraute de Ciencias y tecnología*, 2012.
- [30] C. Systems, "Switches Cisco Catalyst serie 2960-X," 2013.
- [31] D. Watch, "Put the funk into computing," 2014.
- [32] Raphaël Hertzog and R. Mas, "The Debian Administrator's Handbook," 2013.
- [33] Debian and t. u. o. system, "Razones para usar Debian," 2014.
- [34] Debian.org, "Quienes usan Debian," 2014.
- [35] K. Buytaert, "The openmosix howto," *The Linux Documentation Project*, 2002.
- [36] J. D. Sloan, *High performance Linux clusters with OSCAR, Rocks, openMosix, and MPI*: Rodopi, 2009.
- [37] A. Fernández Rodríguez and J. Valdaracete Peinado, "Introducción a Blender," 2012.
- [38] L. M. Santos Jaimes, S. Peñaloza, and E. R. Cruz Cruz, "Cluster implementation of a prototype for the resolution of a particular problem," *Journal Article*, vol. 1 2010.
- [39] Ari J, Morales J, Marck F, and S. G., "Implementación de un servidor clúster de alta disponibilidad bajo herramientas de código abierto en la UJAP," *Tesis, Universidad José Antonio Páez, Venezuela*, 2012.
- [40] N. Pérez Otero, S. Méndez, C. V. Ayusa, M. I. Aucapiña, and V. J. Lopez, "Aplicaciones del cómputo de altas prestaciones," in *XI Workshop de Investigadores en Ciencias de la Computación*, 2009.
- [41] R. Moreno Galdón, "Computación de altas prestaciones aplicada al cálculo de variaciones en genómica," 2013.
- [42] F. G. Tinetti, A. E. De Giusti, F. Romero, D. Encinas, and F. E. Frati, "Procesamiento paralelo de aplicaciones numéricas de alto rendimiento," in *XII Workshop de Investigadores en Ciencias de la Computación*, 2010.

-
- [43] N. Sadashiv and S. M. D. Kumar, "Cluster, grid and cloud computing: A detailed comparison," in *Computer Science & Education (ICCSE), 2011 6th International Conference on*, 2011, pp. 477-482.
- [44] Rocha J. et al, "Diseño e implementación de un clúster de cómputo de alto rendimiento," *Universidad de Guanajuato, México, Acta Universitaria*, vol. vol. 21, núm. 3, pp. pp. 24-33, diciembre, 2011.
- [45] Dawson W and Q. G., "El proyecto fin de carrera en Ingeniería Informática: Una guía para el estudiante," *Madrid - España, Pearson Educación*, 2002.
- [46] Kendall K and K. J, "Análisis y diseño de sistemas," *Pearson Educación*, 2005.
- [47] C. W. OpenWall, "Parallel and distributed processing with John the Ripper," 2014.
- [48] Louwrentius, "Parallel / Distributed Password Cracking With John the Ripper and MPI," 2011.

k. Anexos

Anexo 1: Anteproyecto Trabajo de Titulación



UNIVERSIDAD
NACIONAL
DE LOJA



Área de la Energía, las Industrias y los Recursos Naturales No Renovables

CARRERA DE INGENIERÍA EN SISTEMAS

“Arquitectura clúster de alto rendimiento utilizando herramientas de software libre”

ANTEPROYECTO DE TRABAJO DE TITULACIÓN

Autores:

- *Leonardo Rafael Chuquiguanca Vicente.*
- *Edyson Javier Malla Bustamante.*

Asesor:

- *Luis Antonio Chamba Eras, Mg. Sc.*

LOJA – ECUADOR
2014

Contenido

| | |
|---------------------------------------|-----|
| Índice de Contenidos | 95 |
| Índice General..... | 95 |
| Índice de Figuras..... | 96 |
| Índice de Tablas..... | 97 |
| A. Tema: | 98 |
| B. Problemática:..... | 98 |
| 1. Situación problemática | 98 |
| 2. Problema de Investigación. | 100 |
| C. Justificación | 100 |
| D. Objetivos | 101 |
| 1. Objetivo General | 101 |
| 2. Objetivos Específicos | 101 |
| E. Alcance..... | 102 |
| Fase 1: | 102 |
| Fase 2: | 102 |
| Fase 3: | 103 |
| F. Metodología..... | 103 |
| G. Cronograma..... | 104 |
| H. Presupuesto y financiamiento. | 105 |
| 1. Presupuesto | 105 |
| 2. Financiamiento | 105 |
| I. Bibliografía | 106 |
| J. Anexos | 108 |
| Anexo 1: Licencia..... | 108 |

Índice de Figuras

Figura 1. Cronograma de actividades.....104

Índice de Tablas

| | |
|--|-----|
| TABLA I. PRESUPUESTO ESTIMADO TRABAJO DE TITULACIÓN..... | 105 |
|--|-----|

A. Tema:

“Arquitectura clúster de alto rendimiento utilizando herramientas de software libre”

B. Problemática:

1. Situación problemática.

La constante reducción en el costo del hardware ha promovido la masificación de la arquitectura clúster, la que posee una característica especial, que pueden ser construidas con hardware convencional lo cual reduce en gran medida los costos ocasionados al comprar piezas exclusivas [1], [2].

En el clima actual de presupuestos reducidos y fondos limitados para investigaciones académicas y científicas, las inversiones en computación deben resultar rentables en cuanto a escalabilidad y rendimiento. Después de todo, el éxito de cualquier programa de investigación se mide por su capacidad para acceder a la información de manera oportuna [2].

Actualmente, la supercomputación nos permite simular la mayoría de los sistemas informáticos que procesan grandes cantidades de información con un margen de error inapreciable, al realizar cálculos muy fiables en todos los campos: automoción, física, aeronáutica, química, informática, etc., conocidos como sistemas de alto rendimiento [3], [4], [5].

Los sistemas de alto rendimiento hacen referencia a una rama de la computación aplicada que se centra fundamentalmente en la solución de problemas que hacen un uso intensivo del cálculo, compuestos por grandes sistemas, especializados y de elevado coste, que se pueden encontrar principalmente en centros de investigación [6].

La informática para investigaciones, denominada computación de alto rendimiento usa potentes herramientas y procesos de computación para generar datos en investigaciones académicas avanzadas. Con un clúster de computación de alto rendimiento, los centros de investigación pueden obtener la velocidad y potencia de una

costosa supercomputadora a una fracción del costo y con menos riesgo de sufrir tiempos de inactividad prolongados [7].

Algunas de las aplicaciones de los clúster de alto rendimiento son: simulaciones, modelización, optimización discreta, análisis molecular, búsquedas en árboles, aprendizaje en redes neuronales, tratamiento de imágenes, reconocimiento de patrones, procesamiento de consultas en Base de Datos, procesamiento de grandes volúmenes de información como Big Data y minería de datos, entre otras [8], [9].

Para un centro de investigación de tamaño medio una ventaja obvia de usar clústers de PC's frente a otros súper-ordenadores paralelos convencionales es el precio. Estos ordenadores se producen a muy baja escala y necesitan además de un desarrollo de hardware muy especializado [10].

La Carrera de Ingeniería en Sistemas (CIS) de la Universidad Nacional de Loja cuenta un laboratorio, conformado por 15 computadores personales con las siguientes características: procesador Core i7 de 4 núcleos con de 3.4 GHz de velocidad , 4 Gb de memoria RAM; teniendo en cuenta estas características se puede concluir que el laboratorio de la CIS cuenta con equipos que tienen características técnicas limitadas para atender de forma independiente los siguientes procesos, que forman parte de las investigaciones académicas y científicas:

- ✓ Tratamiento y análisis de grandes cantidades de datos (Big Data).
- ✓ Procesamiento de consultas en base de datos.
- ✓ Software de simulación.
- ✓ Reconocimiento de imágenes.
- ✓ Reconocimiento de patrones (Data Mining).
- ✓ Cifrado y descifrado de códigos (criptografía).
- ✓ Limitaciones en el análisis del resultado de algoritmos.
- ✓ Limitaciones en los resultados de experimento*s que implican gran cantidad de recursos hardware.

1. Problema de Investigación.

¿La implementación de una arquitectura clúster de alto rendimiento con los recursos técnicos con los que cuenta el laboratorio de la Carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja, servirá como alternativa para la ejecución de procesos que requieren de gran capacidad computacional, para lograr mejores resultados en investigaciones académicas?

C. Justificación

En la actualidad, muchos proyectos de investigación de Universidades exigen soluciones informáticas para investigación de alto rendimiento. La capacidad de una solución informática para realizar miles de millones de operaciones por segundo permite lograr avances importantes en la investigación académica [11], [12]. Desafortunadamente, la computación de alto rendimiento siempre requirió supercomputadoras costosas y propietarias que pueden ser difíciles y caras de instalar, mantener y usar, por tal razón resulta necesaria la implementación de mecanismos de procesamiento avanzado de datos, que ayuden al mejor desempeño de las actividades dentro de la carrera, elevando así también el nivel de aprendizaje e investigación en los estudiantes.

Actualmente, la mayoría de los Clúster están formados por componentes de hardware comunes, herramientas de software libre y tecnología estándar. Cabe destacar que equipos para estos fines, con estas capacidades disponibles en el mercado, diseñadas e implementadas por las grandes casas de hardware, tienen un costo exorbitante y que solo pueden ser costeados por entes gubernamentales o por empresas con un gran capital, en respuesta a ello una combinación de hardware y software permite un importante ahorro a nivel económico, ya que al ser un hardware y software de uso común no debe suponer un coste muy importante, y las herramientas de software libre permiten su uso sin coste de licencia [13], [14].

El objetivo del presente Trabajo de Titulación es servir como apoyo a la investigación académica en la CIS de la UNL, con la implementación de un clúster de alto rendimiento la cual permitirá dar una solución potente y económica para mejorar los entornos de investigación, así como también mejorar la capacidad de obtener resultados más exactos y en el menor tiempo posible al procesar grandes cantidades de datos.

Es por eso, que con la realización del presente Trabajo de Titulación se pueden obtener resultados que cumplan con las expectativas propuestas, a un costo notablemente por debajo del sugerido por las grandes empresas de hardware, y con equipos convencionales de cómputo, pudiéndose utilizar de manera opcional equipos actualmente en uso, reutilizando tecnología.

Por lo tanto la implementación y puesta en marcha del presente Trabajo de Titulación se justifica; académica, tecnológica y económicamente, permitiéndole a la Universidad Nacional de Loja y de manera directa a la Carrera de Ingeniería en Sistemas ser más competitiva en el ámbito educativo, pudiendo obtener ventajas significativas entre precio y rendimiento, y soluciones de administración muy flexibles ideales para las necesidades de los entornos educativos en investigaciones actuales.

D. Objetivos.

1. Objetivo General.

Implementar una arquitectura clúster de alto rendimiento en el laboratorio de la Carrera de Ingeniería en Sistemas usando software libre.

2. Objetivos Específicos.

- ✓ Analizar las plataformas clúster para determinar cuál es la más óptima de acuerdo a los equipos y requerimientos existentes.
- ✓ Implementar una plataforma clúster para integrar varias computadoras independientes.
- ✓ Evaluar la funcionalidad del clúster a través de procesos con grandes cantidades de procesamiento computacional

E. Alcance

La finalidad del presente Trabajo de Titulación (TT) es implementar una arquitectura clúster de alto rendimiento, que sirva como apoyo a la investigación científica y académica de la CIS de la UNL.

El tiempo estimado para el desarrollo 400 horas, para lo cual se ha determinado las fases y actividades estimadas que permitirán el cumplimiento de los objetivos planteados.

El escenario en el cual se llevarán a cabo la evaluación del presente proyecto, será el laboratorio de la Carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja, haciendo uso de los recursos hardware de la misma.

Sin embargo el presente proyecto servirá como punto de partida para futuras implementaciones o mejoras en el mismo, y emprender acciones como apoyo a los estudiantes en su formación académica y docentes investigadores.

a. Fase 1:

Analizar las plataformas de clúster para determinar cuál se adapta mejor de acuerdo a los equipos y requerimientos existentes.

- ✓ Revisión bibliográfica y casos de éxito del funcionamiento de los clúster en Centros de Investigación y en la Educación Superior.
- ✓ Analizar los recursos de los equipos informáticos con los que cuenta actualmente la institución.
- ✓ Seleccionar la plataforma de software libre que más se adapten a los requerimientos de los equipos informáticos existentes.

b. Fase 2:

Implementar una plataforma clúster para integrar varias computadoras independientes.

- ✓ Diseño lógico de la arquitectura clúster en el laboratorio de la CIS.
- ✓ Instalación y configuración del sistema operativo del clúster.
- ✓ Instalación y configuración de los servicios del clúster en nodo maestro y nodos esclavos.

c. Fase 3:

Evaluar la funcionalidad del clúster a través de procesos con grandes cantidades de procesamiento computacional.

- ✓ Preparar el escenario para realizar las pruebas de funcionalidad.

- ✓ Evaluar los resultados en base al escenario planteado mediante procesos que impliquen grandes cantidades de recursos computacionales.
- ✓ Elaborar un artículo científico acorde a las normas IEEE.

F. Metodología

Para el desarrollo del Trabajo de Titulación (TT) se utilizará distintos métodos de investigación y técnicas de recolección de información bibliográfica, para poder cumplir con los objetivos planteados [45], [46].

Observación Activa.

Este método nos permitirá involucrarnos directamente con el objeto a investigar para poder tener información más detallada de los problemas existentes y las posibles alternativas de solución que se pueden dar

Estudio de casos.

Este método nos permitirá realizar una exploración e investigación en profundidad de problemas específicos, basados en experiencias ya vividas y contadas y casos de éxito que funcionen y den un aval de que los resultados pueden llegar a ser satisfactorios.

Experimentación.

Técnica mediante la cual se podrá realizar las diferentes pruebas que garantice el correcto funcionamiento de nuestra solución, en base a un escenario donde se ponga a prueba cada una de las aplicaciones y servicios que se puedan implementar.

Técnica de recolección bibliográfica.

Esta técnica nos permitirá extraer la información necesaria para poder sustentar la base teórica del trabajo de titulación, mediante consultas en: fuentes bibliográficas confiables, casos de éxito, artículos científicos, revistas indexadas; entre otras.

G. Cronograma de actividades.














| Id | Mod de tarea | Nombre de tarea | Comienzo | Fin | Texto1 |
|----|---|---|---------------------|---------------------|------------------|
| 1 |  | Implementar una arquitectura clúster de alto rendimiento en el laboratorio de la Carrera de Ingeniería en Sistemas usando software libre | lun 28/04/14 | vie 28/11/14 | 400 horas |
| 2 |  | FASE 1: | lun 28/04/14 | lun 19/05/14 | 90 horas |
| 3 |  | Revisión bibliográfica y casos de éxito del funcionamiento de los clúster en Centros de Investigación y en la Educación Superior. | lun 28/04/14 | lun 05/05/14 | 30 horas |
| 4 |  | Analizar los recursos de los equipos informáticos con los que cuenta actualmente la institución. | mar 06/05/14 | mar 13/05/14 | 30 horas |
| 5 |  | Seleccionar la plataforma de software libre que más se adapten a los requerimientos de los equipos informáticos existentes. | mié 14/05/14 | lun 19/05/14 | 30 horas |
| 6 |  | FASE 2: | mar 20/05/14 | mar 15/07/14 | 170 horas |
| 7 |  | Diseño lógico de la arquitectura clúster en el laboratorio de la CIS. | mar 20/05/14 | lun 02/06/14 | 40 horas |
| 8 |  | Instalación y configuración del sistema operativo del clúster. | mar 03/06/14 | vie 27/06/14 | 60 horas |
| 9 |  | Instalación y configuración de los servicios del clúster en nodo maestro y nodos esclavos. | vie 27/06/14 | vie 11/07/14 | 70 horas |
| 10 |  | FASE 3: | mar 14/10/14 | vie 28/11/14 | 140 horas |
| 11 |  | Preparar el escenario para realizar las pruebas de funcionalidad. | mar 14/10/14 | mié 22/10/14 | 40 horas |
| 12 |  | Evaluar los resultados en base al escenario planteado mediante procesos que impliquen grandes cantidades de recursos computacionales | jue 23/10/14 | vie 14/11/14 | 70 horas |
| 13 |  | Elaborar un artículo científico acorde a las normas IEEE. | mié 12/11/14 | vie 28/11/14 | 30 horas |

Figura 15. Cronograma de actividades.

H. Presupuesto y financiamiento.

1. Presupuesto

TABLA XVIII. PRESUPUESTO ESTIMADO TT.

| Recurso | Subtotal (\$) |
|------------------|----------------|
| Equipos | 800.00 |
| Recursos Humanos | 4000.00 |
| Capacitación | 800.00 |
| Movilización | 400.00 |
| Talleres | 200.00 |
| Servicios varios | 200.00 |
| Total | 6400.00 |

2. Financiamiento

El financiamiento total del presente Trabajo de Titulación será asumido en su totalidad por los responsables del mismo.

I. Bibliografía

Referencias

- [1] G. Cáceres, "Estrategia de implementación de un clúster de alta disponibilidad de N nodos sobre linux usando software libre," 2012.
- [2] L. M. Santos Jaimes, S. Peñaloza, and E. R. Cruz Cruz, "Cluster implementation of a prototype for the resolution of a particular problem," *Journal Article*, vol. 1 2010.
- [3] I. Verona Ríos, "Instalación y configuración de un cluster de alto rendimiento," 2010.
- [4] I. Luna and D. Haidee, "Implementación de un cluster de alta disponibilidad de bases de datos para la GEDGAPA," 2012.
- [5] Ari J, Morales J, Marck F, and S. G., "Implementación de un servidor clúster de alta disponibilidad bajo herramientas de código abierto en la UJAP," *Tesis, Universidad José Antonio Páez, Venezuela*, 2012.
- [6] N. Pérez Otero, S. Méndez, C. V. Ayusa, M. I. Aucapiña, and V. J. Lopez, "Aplicaciones del cómputo de altas prestaciones," in *XI Workshop de Investigadores en Ciencias de la Computación*, 2009.
- [7] R. Moreno Galdón, "Computación de altas prestaciones aplicada al cálculo de variaciones en genómica," 2013.
- [8] E. Montes de Oca, L. C. De Giusti, A. E. De Giusti, and M. Naiouf, "Comparación del uso de GPU y cluster de multicore en problemas con alta demanda computacional," in *XVIII Congreso Argentino de Ciencias de la Computación*, 2012.
- [9] F. G. Tinetti, A. E. De Giusti, F. Romero, D. Encinas, and F. E. Frati, "Procesamiento paralelo de aplicaciones numéricas de alto rendimiento," in *XII Workshop de Investigadores en Ciencias de la Computación*, 2010.
- [10] N. Sadashiv and S. M. D. Kumar, "Cluster, grid and cloud computing: A detailed comparison," in *Computer Science & Education (ICCSE), 2011 6th International Conference on*, 2011, pp. 477-482.
- [11] Rocha J. et al, "Diseño e implementación de un clúster de cómputo de alto rendimiento," *Universidad de Guanajuato, México, Acta Universitaria*, vol. vol. 21, núm. 3, pp. pp. 24-33, diciembre, 2011.

-
- [12] Ari J. Morales J. Marck F. Schmiedeler G, “Implementación de un servidor clúster de alta disponibilidad”, Tesis Universidad José Antonio Páez, Venezuela 2011.
- [13] Dawson W. Quetglás G. “El proyecto fin de carrera en Ingeniería Informática: Una guía para el estudiante”, Madrid - España, Pearson Educación, 2002.
- [14] DELL, Educación Superior, “El valor de la computación de alto rendimiento”, [en línea]. Disponible en: [<http://www.dell.com/learn/mx/es/mxbiz1/hied-the-value-of-high-performance-computing>]
- [15] Kendall K. Kendall J. “Análisis y diseño de sistemas”. Pearson educación, 2005.

J. Anexos

1. Licencia



Anteproyecto de Trabajo de Titulación by Leonardo Rafael Chuquiguanca Vicente and Edyson Javier Malla Bustamante is licensed under a [Creative Commons Reconocimiento-NoComercial 4.0 internacional License](#).

Anexo 2: Testeo de rendimiento con Stress-Test

La aplicación Stress-Test está diseñado para evaluar el rendimiento de un clúster Mosix, la misma realizará muchas evaluaciones a las aplicaciones y al kernel para testear la estabilidad y migración de procesos.

Con la ejecución del test el clúster se verá sobrecargado, al finalizar la ejecución se generará un reporte detallado acerca de cada componente que ha sido evaluado.

2.1. Instalación y configuración.

Como primer paso cambiamos la ubicación del fichero **omtest-0.4.1.tar.gz**, hacia el directorio **/usr/local**, como se ilustra en la Fig. 16 (*Mover stress-test al directorio /usr/local*).

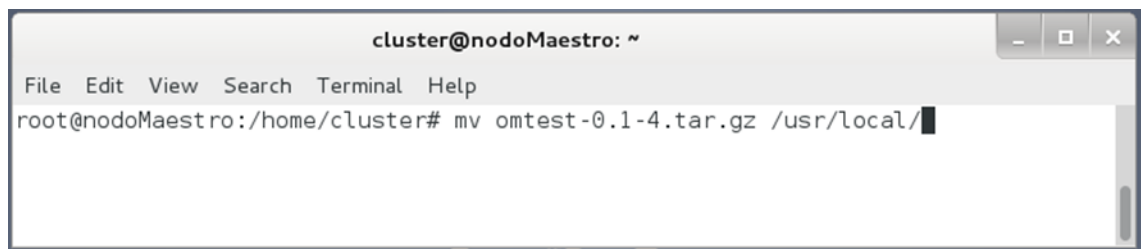


Figura 16. Mover stress-test al directorio /usr/local/.

Luego ingresamos al directorio **/usr/local**, donde se encuentra el archivo comprimido **stress-test**, tecleando en una terminal el comando; **cd /usr/local** como se ilustra en la fig. 17 (*Ingresamos al directorio /usr/local*)

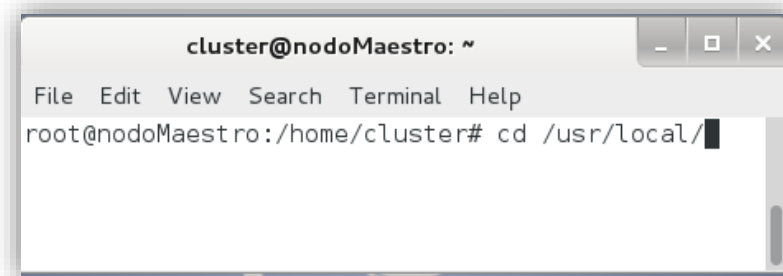
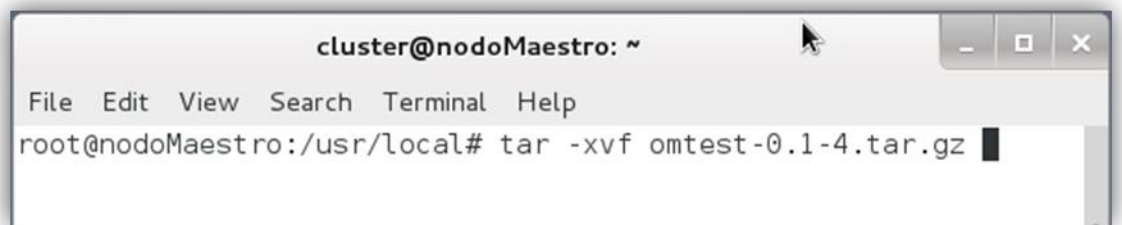


Figura 17. Ingresamos al directorio /usr/local/.

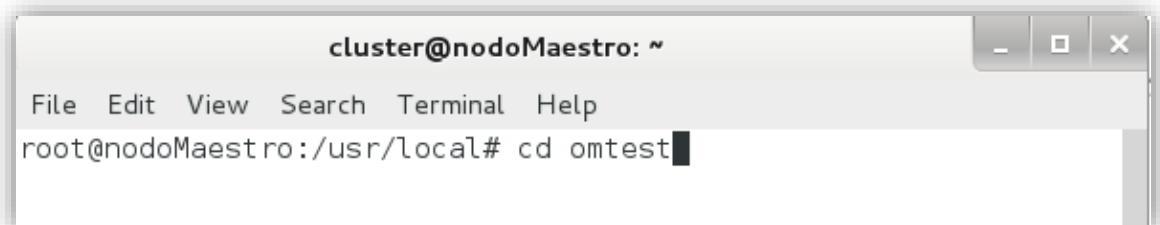
Descomprimos el archivo como usuario **root**, ejecutando el comando **tar -xvf omtest-0.1-4.tar.gz**, como se observa en la Fig. 18 (*Descompresión del archivo tar -xvf omtest-0.1-4.tar.gz*)



```
cluster@nodoMaestro: ~  
File Edit View Search Terminal Help  
root@nodoMaestro:/usr/local# tar -xvf omtest-0.1-4.tar.gz
```

Figura 18. Descompresión del archivo stress-test en el directorio /usr/local/.

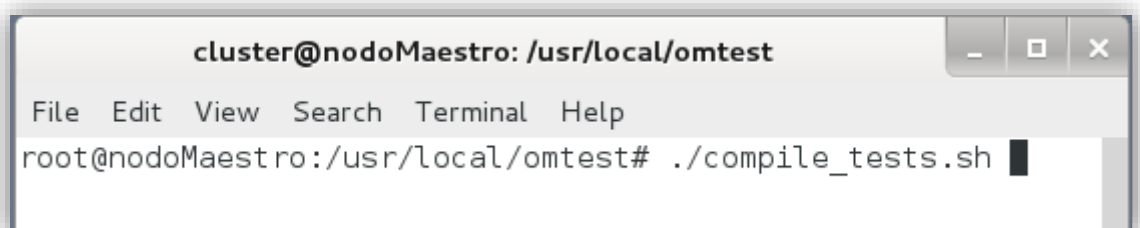
Ingresamos al directorio **omtest** luego de la descompresión del archivo .tar.gz, tecleando el comando **cd omtest**, como se observa en la Fig. 19 (*Ingresamos al directorio omtest*)



```
cluster@nodoMaestro: ~  
File Edit View Search Terminal Help  
root@nodoMaestro:/usr/local# cd omtest
```

Figura 19. Ingresamos al directorio omtest

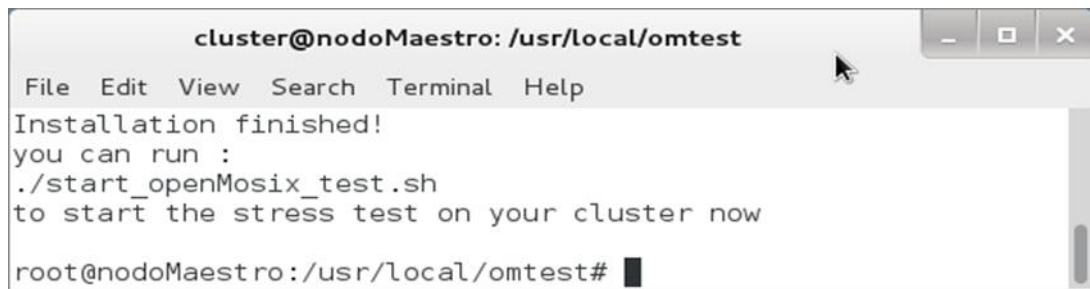
El siguiente paso es compilar la aplicación, esto lo hacemos como usuario **root** tecleando el siguiente comando en la terminal **./compile-test.sh**, como se observa en la Fig. 20 (*Compilación de stress-test mosix*)



```
cluster@nodoMaestro: /usr/local/omtest  
File Edit View Search Terminal Help  
root@nodoMaestro:/usr/local/omtest# ./compile_tests.sh
```

Figura 20. Compilación de stress-test mosix.

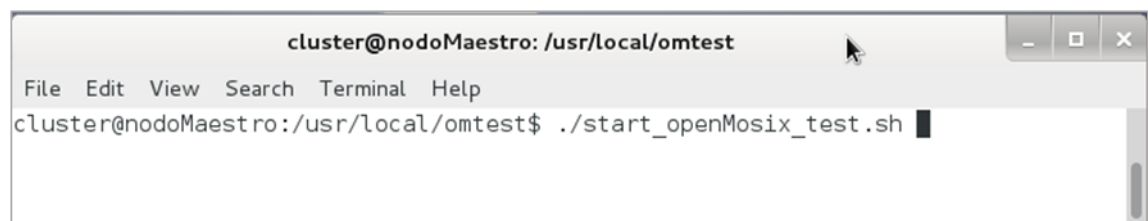
Luego de la compilación observamos el mensaje de que el proceso ha sido realizado con éxito como se ilustra en la Fig. 21 (*Compilación de stress-test finalizada con éxito*)



```
cluster@nodoMaestro: /usr/local/omtest
File Edit View Search Terminal Help
Installation finished!
you can run :
./start_openMosix_test.sh
to start the stress test on your cluster now
root@nodoMaestro:/usr/local/omtest#
```

Figura 21. Compilación de stress-test finalizada con éxito.

Luego ejecutamos el script de iniciación del stress-test de mosix con el comando `./start_openMosix_test.sh` como se observa en la Fig. 22 (*Iniciando stress-test con mosix*)



```
cluster@nodoMaestro: /usr/local/omtest
File Edit View Search Terminal Help
cluster@nodoMaestro:/usr/local/omtest$ ./start_openMosix_test.sh
```

Figura 22. Iniciando stress-test con mosix.

2.2. Prueba test distkeygen

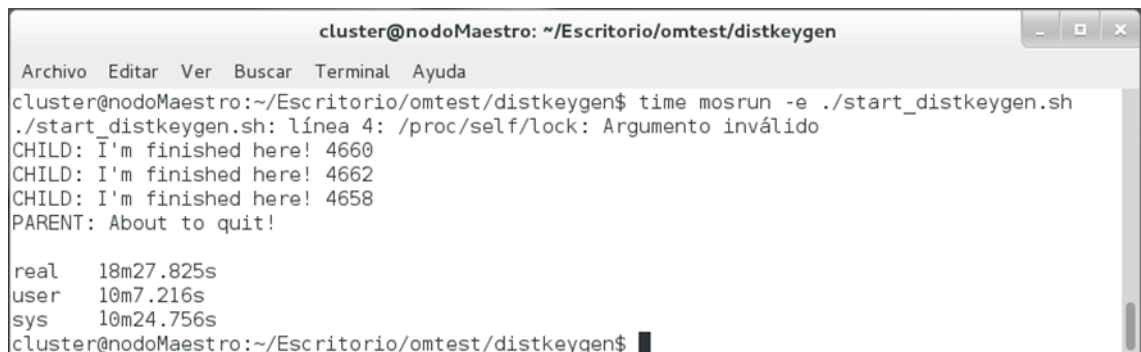
Esta aplicación es usada para generar 222760 pares de claves RSA con una longitud de 1024 bits. La aplicación se distribuye en tantos procesos como procesadores haya en el clúster. Dentro de la aplicación encontramos algunos archivos entre los que están el `cleanup.sh` que sirve para limpiar las claves RSA generadas, este archivo hace uso de `cleanup1.sh`, `cleanup2.sh` respectivamente. Encontramos el código fuente en el archivo `distkeygen.c`, el compilador de la aplicación `compile.sh`, así como el ejecutable de la misma `start_distkeygen.sh` como se ilustra en la Fig. 23 (*Archivos de la aplicación test distkeygen*)



```
cluster@nodoMaestro: /usr/local/omtest/distkeygen
File Edit View Search Terminal Help
cluster@nodoMaestro: /usr/local/omtest/distkeygen$ ls
cleanup.sh  cleanup2.sh  distkeygen  start_distkeygen.sh
cleanup1.sh compile.sh  distkeygen.c
cluster@nodoMaestro: /usr/local/omtest/distkeygen$
```

Figura 23. Archivos de la aplicación test distkeygen.

El tiempo de la ejecución de la aplicación **distkeygen** utilizando los recursos del nodo maestro es de 18 minutos con 27.852 segundos, como se detalla en la Fig. 24 (*Tiempo de ejecución de la aplicación distkeygen nodo maestro*), creando un total de 222760 pares de claves RSA (PrivateKey, PublicKey) como se observa en la Fig. 26 (*Claves RSA generadas por la aplicación test distkeygen*)



```
cluster@nodoMaestro: ~/Escritorio/omtest/distkeygen
Archivo Editar Ver Buscar Terminal Ayuda
cluster@nodoMaestro:~/Escritorio/omtest/distkeygen$ time mosrun -e ./start_distkeygen.sh
./start_distkeygen.sh: línea 4: /proc/self/lock: Argumento inválido
CHILD: I'm finished here! 4660
CHILD: I'm finished here! 4662
CHILD: I'm finished here! 4658
PARENT: About to quit!

real    18m27.825s
user    10m7.216s
sys     10m24.756s
cluster@nodoMaestro:~/Escritorio/omtest/distkeygen$
```

Figura 24. Tiempo de ejecución de la aplicación distkeygen nodo maestro.

El tiempo de la ejecución de la aplicación **distkeygen** utilizando todos los recursos de la arquitectura clúster es de 3 minutos con 59.970 segundos, como se detalla en la Fig. 25 (*Tiempo de ejecución de la aplicación distkeygen recursos clúster*). Obteniendo una diferencia de 14 minutos con 27.825 segundos respecto a los recursos de un solo nodo.


```
cluster@nodoMaestro: ~/Escritorio/omtest/distkeygen
Archivo Editar Ver Buscar Terminal Ayuda
cluster@nodoMaestro:~/Escritorio/omtest/distkeygen$ time mosrun -e ./start_distkeygen.sh
./start_distkeygen.sh: línea 4: /proc/self/lock: Argumento inválido
CHILD: I'm finished here! 11960
CHILD: I'm finished here! 11962
CHILD: I'm finished here! 11964
PARENT: About to quit!

real    3m59.970s
user    0m5.824s
sys     0m12.616s
cluster@nodoMaestro:~/Escritorio/omtest/distkeygen$
```

Figura 25. Tiempo de ejecución de la aplicación distkeygen recursos clúster.

En la Fig. 26 (*Claves RSA generadas por la aplicación test distkeygen*), se detalla el número total de claves RSA (PrivateKey, PublicKey) en un total de 222760 ficheros con la ejecución de la aplicación **distkeygen**.

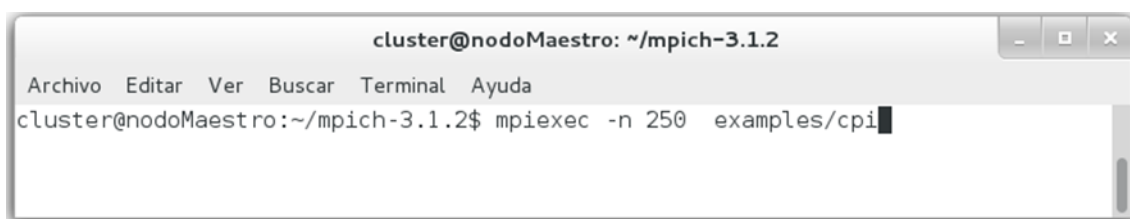
```
cluster@nodoMaestro: /usr/local/omtest/distkeygen
Archivo Editar Ver Buscar Terminal Ayuda
04574-PrivateKey 135881-PrivateKey 222758-PrivateKey cleanup1.sh
04574-PublicKey 135881-PublicKey 222758-PublicKey cleanup2.sh
04575-PrivateKey 135882-PrivateKey 222759-PrivateKey cleanup.sh
04575-PublicKey 135882-PublicKey 222759-PublicKey compile.sh
04576-PrivateKey 135883-PrivateKey 22275-PublicKey distkeygen
04576-PublicKey 135883-PublicKey 22275-PublicKey distkeygen.c
04577-PrivateKey 135884-PrivateKey 222760-PrivateKey machinefile
04577-PublicKey 135884-PublicKey 222760-PublicKey start_distkeygen.sh
cluster@nodoMaestro: /usr/local/omtest/distkeygen$
```

Figura 26. Claves RSA generadas por la aplicación test distkeygen.

Anexo 3: Cálculo del valor aproximado de π utilizando MPI

Para determinar el valor aproximado de π , se utilizó el código de la Tabla XVIII (*Código fuente valor de pi*), esta aplicación hace uso del estándar MPI.

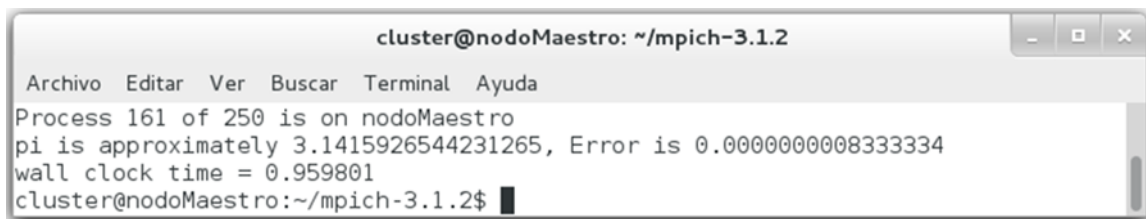
El proceso de ejecución se ilustra en la Fig. 27 (*Ejecución de MPICH para calcular el valor de pi utilizando los recursos del nodo maestro*), donde **mpiexec** es el ejecutable del estándar MPI, **-n** es el número de procesadores a utilizar seguido de la ruta de la aplicación a ejecutar **examples/cpi**.



```
cluster@nodoMaestro: ~/mpich-3.1.2
Archivo Editar Ver Buscar Terminal Ayuda
cluster@nodoMaestro:~/mpich-3.1.2$ mpiexec -n 250 examples/cpi
```

Figura 27. Ejecución de MPICH para calcular el valor de pi desde el nodo maestro.

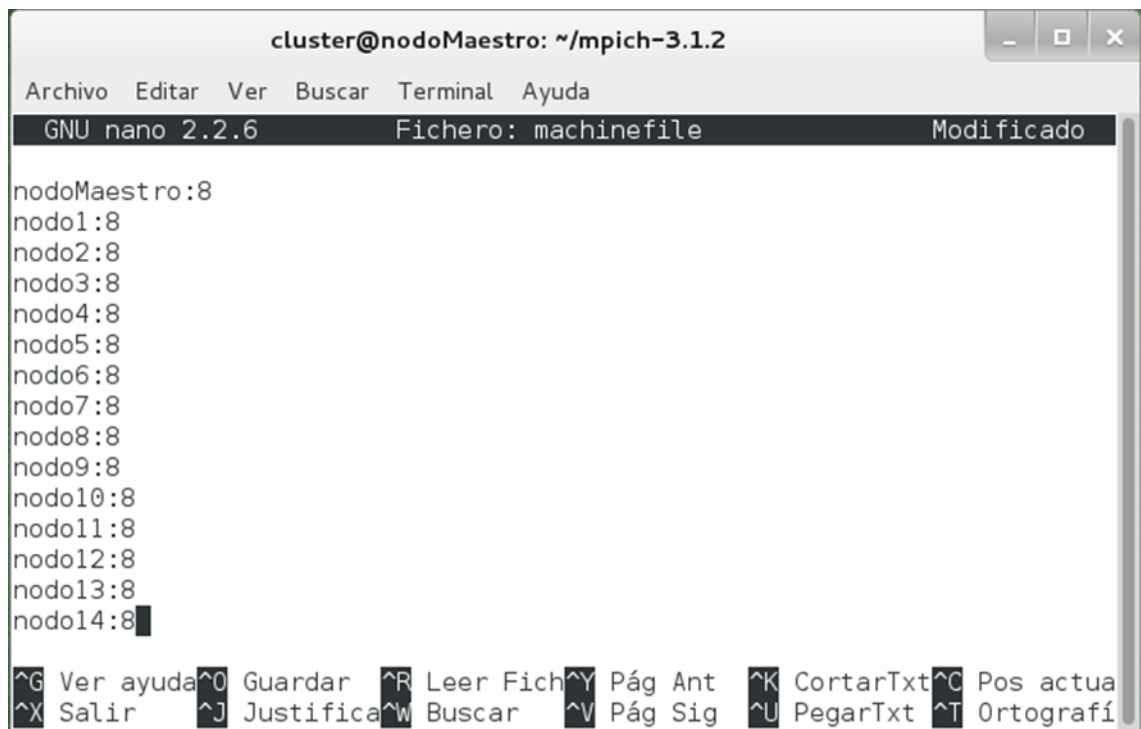
El tiempo que demora en ejecutarse la aplicación es de 0.959801 segundos, utilizando los recursos del nodo maestro, el proceso se detalla en la Fig. 28 (*Resultado del tiempo de ejecución para el cálculo del valor de pi en el nodo maestro*)



```
cluster@nodoMaestro: ~/mpich-3.1.2
Archivo Editar Ver Buscar Terminal Ayuda
Process 161 of 250 is on nodoMaestro
pi is approximately 3.1415926544231265, Error is 0.0000000008333334
wall clock time = 0.959801
cluster@nodoMaestro:~/mpich-3.1.2$
```

Figura 28. Resultado del tiempo de ejecución para el cálculo de valor de pi en el nodo maestro.

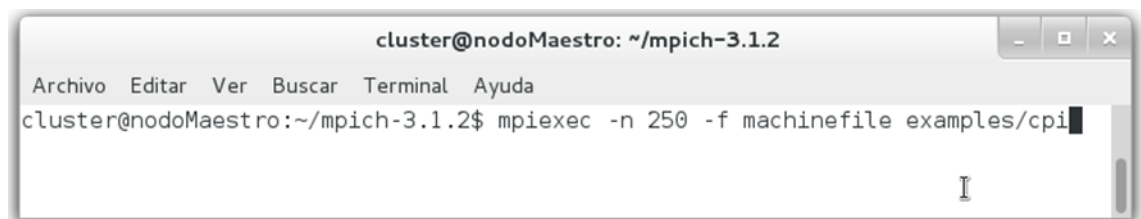
Para realizar las pruebas utilizando 5, 10 y 15 nodos hacemos uso del fichero **machinefile** ver Fig. 29 (*Archivo machinefile que contiene el número de procesadores de cada nodo*).



```
cluster@nodoMaestro: ~/mpich-3.1.2
GNU nano 2.2.6 Fichero: machinefile Modificado
nodoMaestro:8
nodo1:8
nodo2:8
nodo3:8
nodo4:8
nodo5:8
nodo6:8
nodo7:8
nodo8:8
nodo9:8
nodo10:8
nodo11:8
nodo12:8
nodo13:8
nodo14:8
```

Figura 29. Archivo machinefile que contiene el número de procesadores de cada nodo.

Para la evaluación con 5 nodos se hace uso del fichero **machinefile**, ver Fig. 30 (*Ejecución de MPICH para el cálculo del valor de pi*). El mismo proceso se realiza utilizando 10 y 15 nodos respectivamente.



```
cluster@nodoMaestro: ~/mpich-3.1.2
cluster@nodoMaestro:~/mpich-3.1.2$ mpiexec -n 250 -f machinefile examples/cpi
```

Figura 30. Ejecución de MPICH para el cálculo del valor de pi.

El resultado de la ejecución de la aplicación utilizando 5 nodos es de 0.0321487 segundos, el resultado se ilustra en la Fig. 31 (*Resultado de la ejecución del proyecto 1 utilizando 5 nodos*).

```
cluster@nodoMaestro: ~/mpich-3.1.2
Archivo Editar Ver Buscar Terminal Ayuda
pi is approximately 3.1415926544231265, Error is 0.000000008333334
wall clock time = 0.321487
cluster@nodoMaestro:~/mpich-3.1.2$
```

Figura 31. Resultado de la ejecución del proyecto 1 utilizando 5 nodos.

El resultado de la ejecución de la aplicación utilizando 10 nodos es de 0.228983 segundos, ver Fig. 32 (*Resultado de la ejecución del proyecto 1 utilizando 10 nodos*).

```
cluster@nodoMaestro: ~/mpich-3.1.2
Archivo Editar Ver Buscar Terminal Ayuda
pi is approximately 3.1415926544231265, Error is 0.000000008333334
wall clock time = 0.228983
cluster@nodoMaestro:~/mpich-3.1.2$
```

Figura 32. Resultado de la ejecución del proyecto 1 utilizando 10 nodos.

Finalmente utilizando 15 nodos, el total de la arquitectura clúster es de 0.136832 segundos, ver Fig. 33 (*Resultado de la ejecución del proyecto 1 utilizando 15 nodos*).

```
cluster@nodoMaestro: ~/mpich-3.1.2
Archivo Editar Ver Buscar Terminal Ayuda
Process 152 of 250 is on nodo4
pi is approximately 3.1415926544231270, Error is 0.000000008333338
wall clock time = 0.136832
cluster@nodoMaestro:~/mpich-3.1.2$
```

Figura 33. Resultado de la ejecución del proyecto 1 utilizando 15 nodos.

En la tabla XIX (Código fuente valor de pi), se detalla el código fuente utilizado de la aplicación para calcular el valor de pi, el mismo que pertenece a los archivos de ejemplo de mpich-3.1.2

TABLA XIX. CÓDIGO FUENTE VALOR DE PI

```
/* Mode: C; c-basic-offset:4 ; indent-tabs-mode:nil ;
 * (C) 2001 by Argonne National Laboratory.
 * See COPYRIGHT in top-level directory. */

#include "mpi.h"
#include <stdio.h>
```

```
#include <math.h>

double f(double);
double f(double a){
    return (4.0 / (1.0 + a*a));
}

int main (int argc, char *argv[]){
    int n, myid, numprocs, i;
    double PI25DT = 3.141592653589793238462643;
    double mypi, pi, h, sum, x;
    double startwtime = 0.0, endwtime;
    int namelen;
    char processor_name[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    MPI_Get_processor_name(processor_name, &namelen);

    fprintf(stdout, "Process %d of %d is on %s\n", myid, numprocs, processor_name);
    fflush(stdout);

    n = 10000; /* default # of rectangles */
    if (myid == 0)
        startwtime = MPI_Wtime();

    MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);

    h = 1.0 / (double) n;
    sum = 0.0;
    /* A slightly better approach starts from large i and works back */
    for (i = myid + 1; i <= n; i += numprocs) {
        x = h * ((double) i - 0.5);
        sum += f(x);
    }
    mypi = h * sum;

    MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

    if (myid == 0) {
        endwtime = MPI_Wtime();
        printf("pi is approximately %.16f, Error is %.16f\n",
            pi, fabs(pi - PI25DT));
        printf("wall clock time = %f\n", endwtime - startwtime);
        fflush(stdout);
    }

    MPI_Finalize();
    return 0;
}
```

Anexo 4: Pruebas de compresión de música con Mosix

Para la evaluación de Mosix se realizó la compresión de 32 ficheros de audio .wav como se observa en la Fig. 34 (*Listado de canciones a comprimir con Mosix*), las mismas que se encuentran ubicadas en el directorio música del nodo maestro, el tamaño total de estos ficheros es de 5.2 Gb ver Fig. 35 (*Tamaño de los ficheros antes de la compresión*).

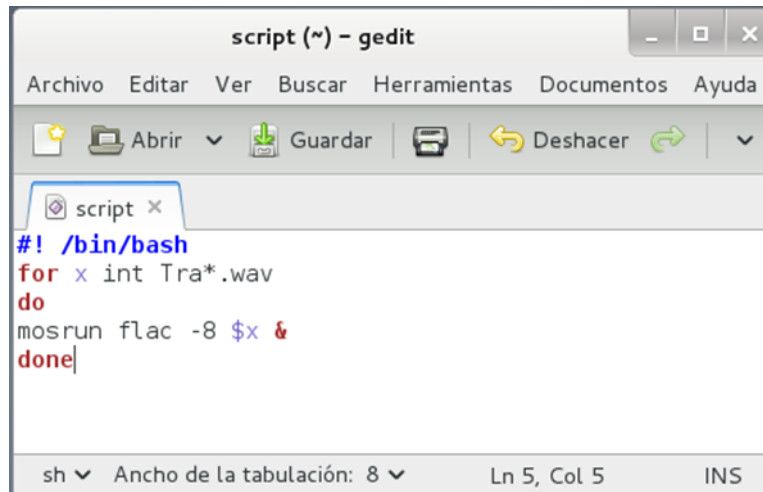
```
cluster@nodoMaestro: ~/Música
Archivo Editar Ver Buscar Terminal Ayuda
cluster@nodoMaestro:~/Música$ ls -l
total 5100256
-rwxrwxrwx 1 cluster cluster 58 ago 11 15:13 jmosix.sh
-rwxrwxrwx 1 cluster cluster 49 ago 11 10:03 jnormal.sh
-rw-r--r-- 1 cluster cluster 145673644 oct 17 13:26 Track10.wav
-rw----- 1 cluster cluster 167988684 ago 11 11:14 Track11.wav
-rw----- 1 cluster cluster 118177324 ago 11 11:25 Track12.wav
-rw-r--r-- 1 cluster cluster 133646732 oct 17 13:29 Track13.wav
-rw----- 1 cluster cluster 120137324 ago 11 11:50 Track14.wav
-rw----- 1 cluster cluster 172116652 ago 11 12:06 Track15.wav
-rw----- 1 cluster cluster 316345356 oct 1 16:02 Track16.wav
-rw----- 1 cluster cluster 172116652 ago 11 12:06 Track17.wav
-rw----- 1 cluster cluster 316345356 oct 1 16:02 Track18.wav
-rw----- 1 cluster cluster 120137324 ago 11 11:50 Track19.wav
-rw----- 1 cluster cluster 131535692 ago 11 10:42 Track1.wav
-rw-r--r-- 1 cluster cluster 133646732 oct 17 14:23 Track20.wav
-rw----- 1 cluster cluster 118177324 ago 11 11:25 Track21.wav
-rw-r--r-- 1 cluster cluster 167988684 oct 17 14:24 Track22.wav
-rw-r--r-- 1 cluster cluster 145673644 oct 17 13:26 Track23.wav
-rw----- 1 cluster cluster 131535692 ago 11 10:42 Track24.wav
-rw-r--r-- 1 cluster cluster 316345356 oct 17 14:26 Track25.wav
-rw----- 1 cluster cluster 133646732 ago 11 11:38 Track26.wav
-rw----- 1 cluster cluster 118177324 ago 11 11:25 Track27.wav
-rw-r--r-- 1 cluster cluster 167988684 oct 17 14:27 Track28.wav
-rw----- 1 cluster cluster 120137324 ago 11 11:50 Track29.wav
-rw----- 1 cluster cluster 145673644 ago 11 10:56 Track2.wav
-rw----- 1 cluster cluster 172116652 ago 11 12:06 Track30.wav
-rw----- 1 cluster cluster 131535692 ago 11 10:42 Track31.wav
-rw----- 1 cluster cluster 145673644 ago 11 10:56 Track32.wav
-rw----- 1 cluster cluster 167988684 ago 11 11:14 Track3.wav
-rw----- 1 cluster cluster 118177324 ago 11 11:25 Track4.wav
-rw----- 1 cluster cluster 133646732 ago 11 11:38 Track5.wav
-rw----- 1 cluster cluster 120137324 ago 11 11:50 Track6.wav
-rw----- 1 cluster cluster 172116652 ago 11 12:06 Track7.wav
-rw----- 1 cluster cluster 316345356 oct 1 16:02 Track8.wav
-rw----- 1 cluster cluster 131535692 ago 11 10:42 Track9.wav
cluster@nodoMaestro:~/Música$
```

Figura 34. Listado de canciones a comprimir con Mosix.

Nombres: Track32.wav, Track31.wav, Track30.wav, Track29.wav, Track28....
Tipo: sonido WAV (audio/x-wav)
Contenido: 32 elementos, 5,2 GB en total
Lugar: /home/cluster/Música
Volumen: desconocido

Figura 35. Tamaño de los ficheros antes de la compresión.

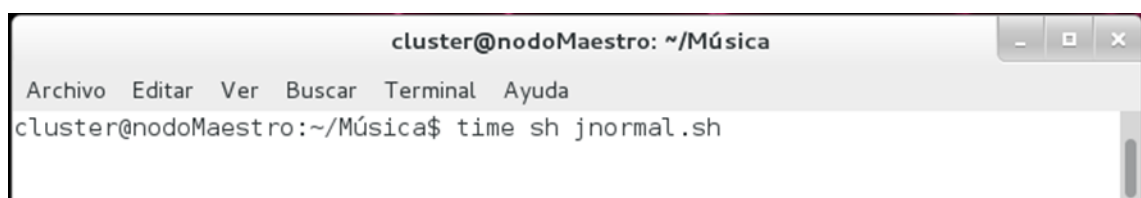
La compresión de los ficheros se realizó ejecutando el script de la Fig. 36 (*Script para la compresión de los ficheros .wav*).



```
script (*) - gedit
Archivo  Editar  Ver  Buscar  Herramientas  Documentos  Ayuda
Abrir  Guardar  Deshacer
script x
#!/bin/bash
for x in Tra*.wav
do
mosrun flac -8 $x &
done
sh  Ancho de la tabulación: 8  Ln 5, Col 5  INS
```

Figura 36. Script para la compresión de los ficheros .wav

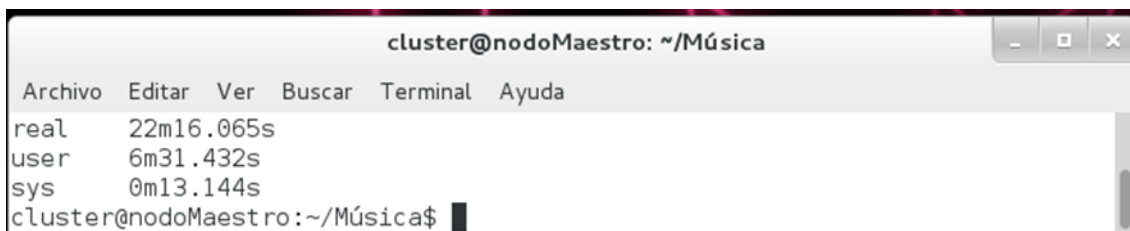
Para ejecutar la aplicación utilizamos el comando **\$ time sh jnormal.sh** como se observa en la Fig. 37 (*Ejecución del script para la compresión de música con Mosix utilizando los recursos del nodo maestro*). En este primer caso utilizaremos solo los recursos del nodo maestro, es decir utilizaremos un único nodo.



```
cluster@nodoMaestro: ~/Música
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
cluster@nodoMaestro:~/Música$ time sh jnormal.sh
```

Figura 37. Ejecución del script para la compresión de música con Mosix utilizando los recursos del nodo maestro.

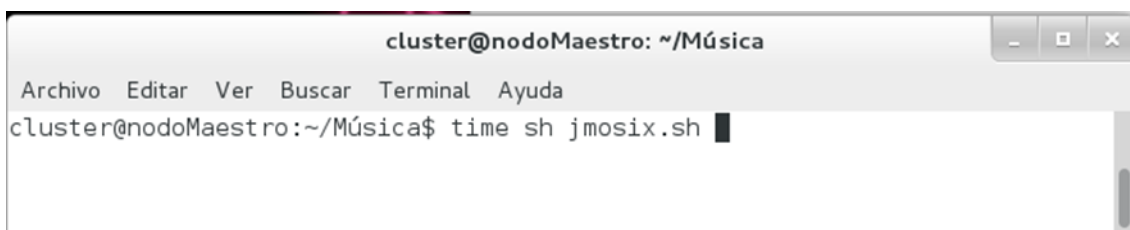
El tiempo de ejecución de la aplicación es de 22 minutos con 16.065 segundos, como se puede observar en la Fig. 38 (*Resultado de la aplicación utilizando los recursos del nodo maestro*).



```
cluster@nodoMaestro: ~/Música
Archivo Editar Ver Buscar Terminal Ayuda
real    22m16.065s
user    6m31.432s
sys     0m13.144s
cluster@nodoMaestro:~/Música$
```

Figura 38. Resultado de la aplicación utilizando los recursos del nodo maestro.

Para hacer uso de las características de migración automática de los procesos con Mosix utilizamos el comando **\$ time sh jmosix.sh** como se ilustra en la Fig. 39 (*Ejecución del script para la compresión de música con Mosix*). En este caso utilizaremos un total de 5 nodos de la arquitectura clúster.



```
cluster@nodoMaestro: ~/Música
Archivo Editar Ver Buscar Terminal Ayuda
cluster@nodoMaestro:~/Música$ time sh jmosix.sh
```

Figura 39. Ejecución del script para la compresión de música con Mosix.

El tiempo de ejecución de la aplicación utilizando 5 nodos es de 11 minutos con 23.602 segundos, como se puede observar en la Fig. 40 (*Resultado de la aplicación utilizando los recursos de 5 nodos de la arquitectura clúster*).



```
cluster@nodomaestro: ~/Música
Archivo Editar Ver Buscar Terminal Ayuda
cluster@nodomaestro:~/Música$ real 11m23.602
```

Figura 40. Resultado de la aplicación utilizando los recursos de los 5 nodos de la arquitectura clúster.

Utilizando 10 nodos del clúster se obtiene un tiempo de ejecución de 9 minutos con 34.572 segundos como se observa en la Fig. 41 (*Resultado de la aplicación utilizando los recursos de 10 nodos de la arquitectura clúster*).

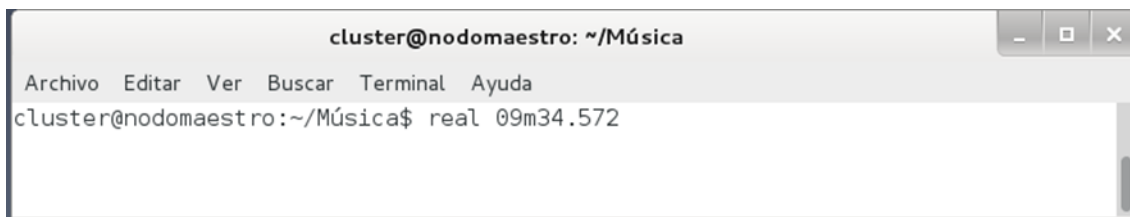


Figura 41. Resultado de la aplicación utilizando los recursos de los 10 nodos de la arquitectura clúster.

Finalmente utilizando los 15 nodos, arquitectura total del clúster, se obtiene un tiempo de ejecución de 7 minutos con 45.562 segundos como se observa en la Fig. 42 (*Resultado de la aplicación utilizando los recursos de los 15 nodos de la arquitectura clúster*).

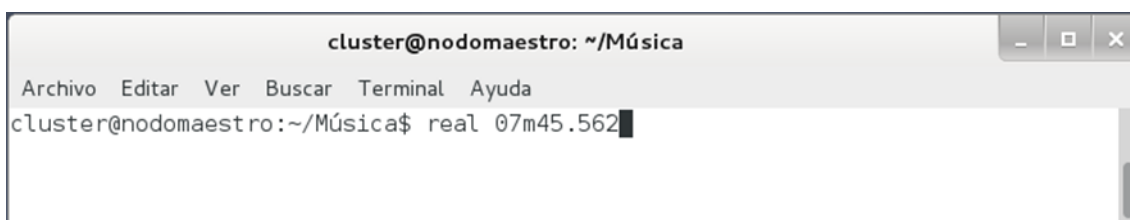


Figura 42. Resultado de la aplicación utilizando los recursos de los 15 nodos de la arquitectura clúster.

El tamaño total de los ficheros luego del proceso de compresión de las canciones utilizando Mosix es de 1.9 Gb, como se observa en la Fig. 43 (*Tamaño de los ficheros luego del proceso de compresión utilizando Mosix*).

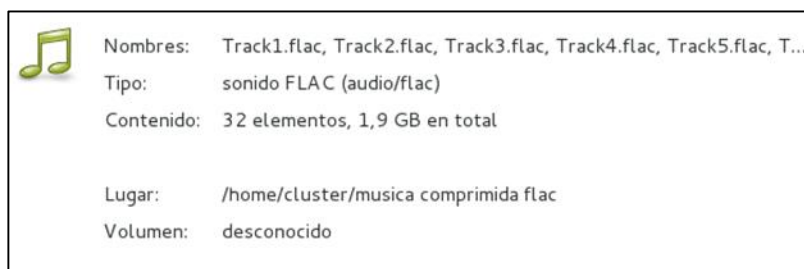


Figura 43. Tamaño de los ficheros luego del proceso de compresión utilizando Mosix.

Anexo 5: Instalación y configuración de John the Ripper (JTR)

5.1 Cifrado de códigos con John The Ripper (JTR) y MPI

Una de las preguntas más comunes que nos hacemos en el campo de la computación es “¿Podemos hacer uso de múltiples procesadores o núcleos para mejorar la velocidad de procesamiento?”, en el cifrado de códigos no es algo diferente. En este caso se utilizó las librerías MPI conjuntamente con el programa de cifrado de códigos John the Ripper. Originalmente John the Ripper era solamente utilizado en un solo procesador para realizar ataques de fuerza bruta o de diccionario [47].

Sin embargo John the Ripper desde la versión 1.7.6+ viene integrado paralelismo para poder ejecutarse en sistemas multiprocesamiento mediante directivas OpenMP. A partir de la versión 1.7.9-jumbo-7, John the Ripper es compatible con librerías MPI para trabajar en un entorno clúster, para lo cual no es necesario instalar ningún parche adicional, simplemente debemos editar el fichero **Makefile** (ver Fig. 47. Edición del fichero *makefile* para habilitar la configuración de John The Ripper).

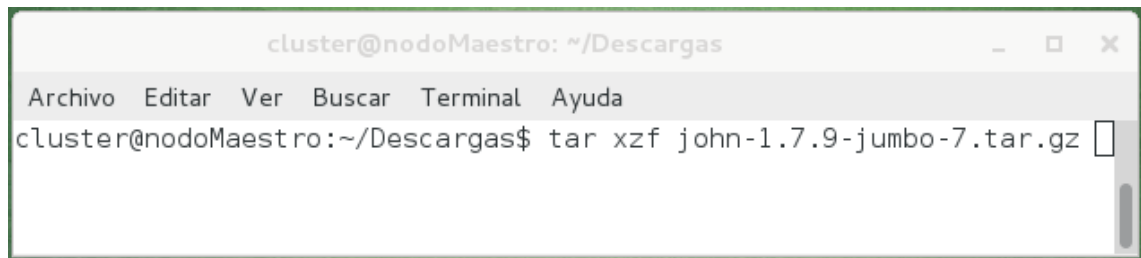
4.1 Instalación y configuración JTR [48]

Antes de iniciar con el proceso de instalación y configuración de JTR debemos de tener instalado y configurado correctamente MPICH, ver (*Manual Técnico Clúster de alto rendimiento apartado Instalación y configuración MPICH*), e instalar la librería de OpenSS que es necesaria para compilar JTR. Para instalar la librería de OpenSSL debemos de ejecutar en la terminal el siguiente comando:

- **# apt-get install libssl-dev**

Posteriormente procedemos a realizar la descarga de JTR en su versión 1.7.9-jumbo-7, con la cual vamos a trabajar en el presente proyecto, esto lo hacemos desde la web oficial de John the Ripper <http://www.openwall.com/john/>.

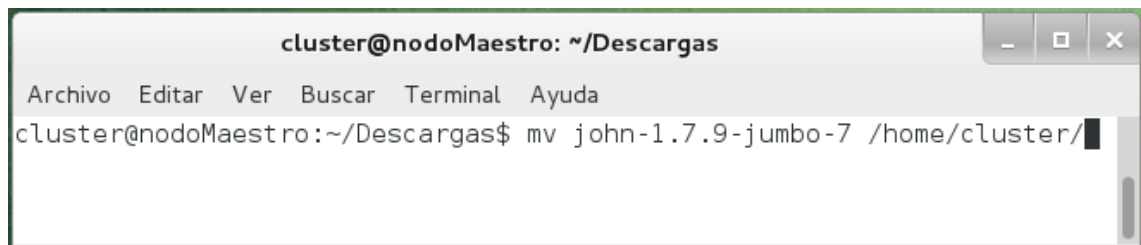
Una vez descargado el programa procedemos a descomprimirlo tecleando en la terminal el siguiente comando **\$ tar xzf john-1.7.9-jumbo-7.tar.gz**, ver Fig. 44 (*Descompresión del programa John the Ripper*)



```
cluster@nodoMaestro: ~/Descargas
Archivo Editar Ver Buscar Terminal Ayuda
cluster@nodoMaestro:~/Descargas$ tar xzf john-1.7.9-jumbo-7.tar.gz
```

Figura 44. Descompresión del programa John the Ripper.

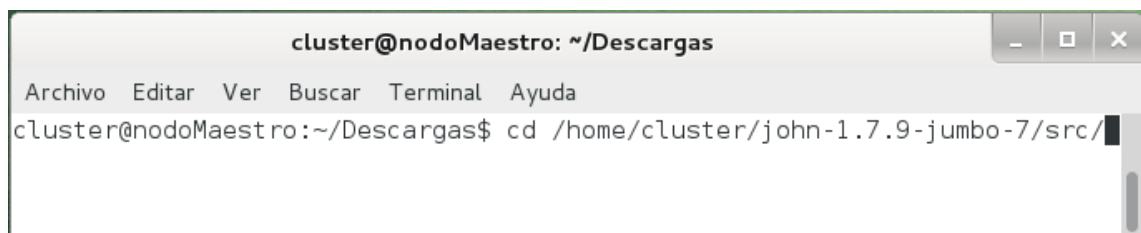
Luego de haber descomprimido el archivo lo movemos al directorio personal, tecleando en la terminal el comando **\$ mv john-1.7.9-jumbo-7 /home/cluster/** ver Fig. 45 (*Cambio de directorio de instalación de John the Ripper*)



```
cluster@nodoMaestro: ~/Descargas
Archivo Editar Ver Buscar Terminal Ayuda
cluster@nodoMaestro:~/Descargas$ mv john-1.7.9-jumbo-7 /home/cluster/
```

Figura 45. Cambio de directorio de instalación de John the Ripper.

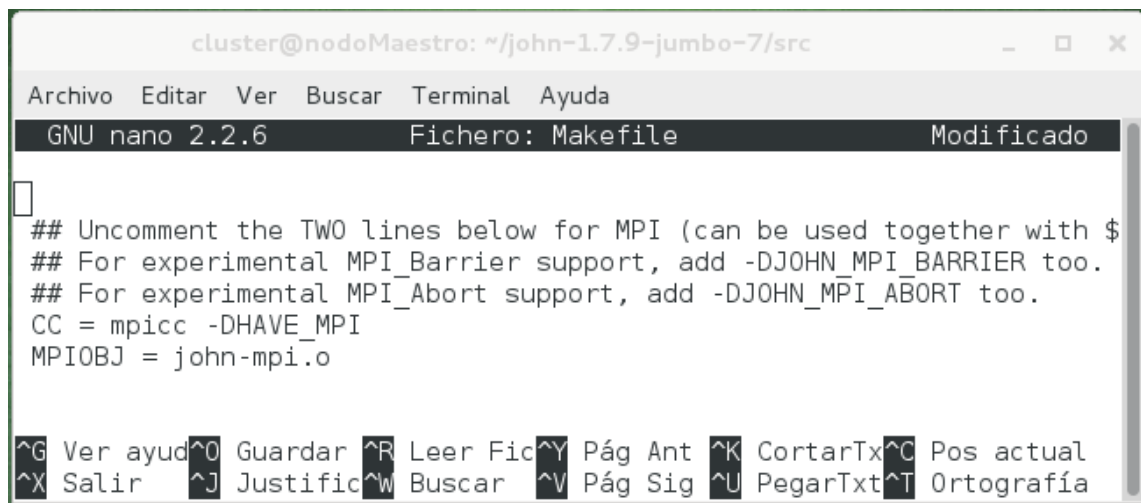
Luego ingresamos al directorio **john-1.7.9-jumbo-7/src/** ejecutando en la terminal el siguiente comando **\$ cd /home/cluster/john-1.7.9-jumbo-7/src/**, ver Fig. 46 (*Ingreso al directorio de John de Ripper*)



```
cluster@nodoMaestro: ~/Descargas
Archivo Editar Ver Buscar Terminal Ayuda
cluster@nodoMaestro:~/Descargas$ cd /home/cluster/john-1.7.9-jumbo-7/src/
```

Figura 46. Ingreso al directorio de John the Ripper.

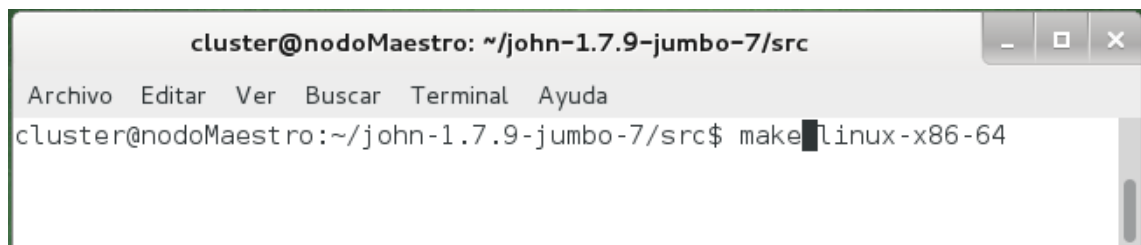
Posterior a este paso, editamos el fichero **Makefile**, en el cual debemos descomentar las dos líneas de la sección MPI, como se observa en la Fig. 47 (*Edición del fichero makefile para habilitar la configuración de John The Ripper*).



```
cluster@nodoMaestro: ~/john-1.7.9-jumbo-7/src
GNU nano 2.2.6 Fichero: Makefile Modificado
## Uncomment the TWO lines below for MPI (can be used together with $
## For experimental MPI_Barrier support, add -DJOHN_MPI_BARRIER too.
## For experimental MPI_Abort support, add -DJOHN_MPI_ABORT too.
CC = mpicc -DHAVE_MPI
MPIOBJ = john-mpi.o
^G Ver ayud^O Guardar ^R Leer Fic^Y Pág Ant ^K CortarTx^C Pos actual
^X Salir ^J Justific^W Buscar ^V Pág Sig ^U PegarTxt^T Ortografía
```

Figura 47. Edición del fichero makefile para habilitar la configuración de John The Ripper.

Compilamos JTR utilizando librerías MPI, para lo cual ejecutamos en la terminal el comando **\$ make linux-x86-64**, donde linux-x86-64 es la arquitectura del procesador en nuestro caso de 64 bits, ver Fig. 48 (*Compilación de John the Ripper utilizando librerías MPI*).



```
cluster@nodoMaestro: ~/john-1.7.9-jumbo-7/src
cluster@nodoMaestro:~/john-1.7.9-jumbo-7/src$ make linux-x86-64
```

Figura 48. Compilación de John the Ripper utilizando librerías MPI.

Una vez compilado hacemos una prueba de testeo de JTR, para ello accedemos al directorio **/run**, ejecutando en la terminal el comando **\$ cd ./run/** y posteriormente ejecutamos **\$/john --test**, ver Fig. 49 (*Prueba de testeo John the Ripper*)

```
cluster@nodoMaestro: ~/john-1.7.9-jumbo-7/run
Archivo Editar Ver Buscar Terminal Ayuda
cluster@nodoMaestro:~/john-1.7.9-jumbo-7/src$ cd ../run/
cluster@nodoMaestro:~/john-1.7.9-jumbo-7/run$ ./john --test
```

Figura 49. Prueba de testeo de John the Ripper.

4.2 Ejecutando John the Ripper utilizando MPI

La Fig. 50 (*Ejecución del test de JTR con MPI*), ilustra la prueba de testeo de John The Ripper con librerías MPI, haciendo uso del argumento **mpiexec** de MPICH, donde **-n** indica el número de procesadores ejecutándose simultáneamente.

```
cluster@nodoMaestro: ~/john-1.7.9-jumbo-7/run
Archivo Editar Ver Buscar Terminal Ayuda
cluster@nodoMaestro:~/john-1.7.9-jumbo-7/run$ mpiexec -n 2 ./john --test
Benchmarking: Traditional DES [128/128 BS SSE2-16]... (2xMPI) DONE
Many salts:      1440K c/s real, 1500K c/s virtual
Only one salt:   1422K c/s real, 1451K c/s virtual
```

Figura 50. Ejecución del test de JTR con MPI.

La Fig. 51 (*Descifrando el fichero shadow con John The Ripper-MPI*) ilustra el proceso de como descriptar de manera correcta el fichero **shadow** con JTR – MPI haciendo uso de 120 procesadores, del archivo **machinefile** ver Fig. 29 (*Archivo machinefile que contiene el número de procesadores de cada nodo*).

```
cluster@nodoMaestro: ~/mpich-3.1.2
Archivo Editar Ver Buscar Terminal Ayuda
cluster@nodoMaestro:~/mpich-3.1.2$ mpiexec -n 120 machinefile ../john-1.7.9-jumbo-7/run/john ../Escritorio/shadow
```

Figura 51. Descifrando el fichero shadow con John The Ripper – MPI.

Anexo 6. Prueba de cifrado de códigos con MPICH-JTR

El sitio web <https://howsecureismypassword.net/> se caracteriza por comprobar la fortaleza de las contraseñas, el sitio nos indica que nuestra clave tiene una longitud de 8 símbolos, y que consta de 62 combinaciones de caracteres (26 letras del abecedario mayúsculas + 26 letras del abecedario minúsculas + 10 dígitos) por lo que se realizaran 8^{62} posibles combinaciones (cerca de 218 trillones), por lo tanto; nuestro password puede ser descifrada en aproximadamente 20 horas en un computador que realice 3 billones de cálculo por segundo utilizando un solo procesador, como se observa en la Fig. 52 (*Tiempo aproximado de descifrado del password*).

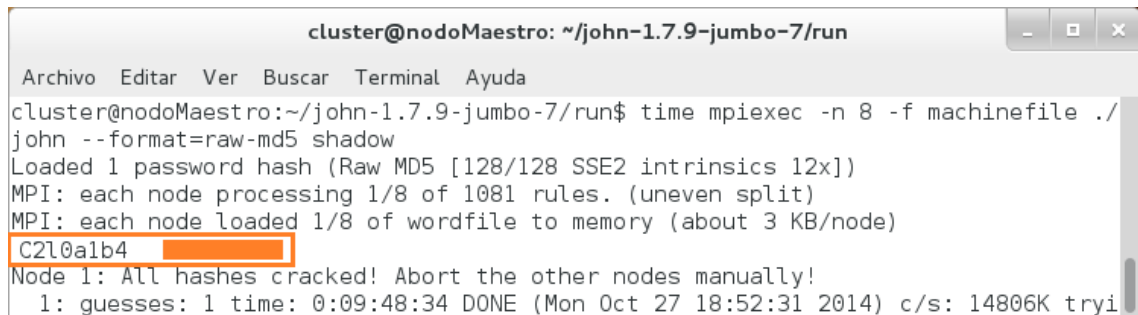


Figura 52. Tiempo aproximado de descifrado del password.

Para realizar las pruebas haciendo uso de todos los procesadores con los que cuenta cada uno de los nodos utilizaremos el archivo machinefile como observamos en la Fig. 29 (*Archivo machinefile que contiene el número de procesadores de cada nodo*).

En la Fig. 53 (*Descifrado de contraseña con JTR utilizando el nodo maestro*) podemos observar el proceso de ejecución de JTR en el nodo maestro, a la vez también podemos

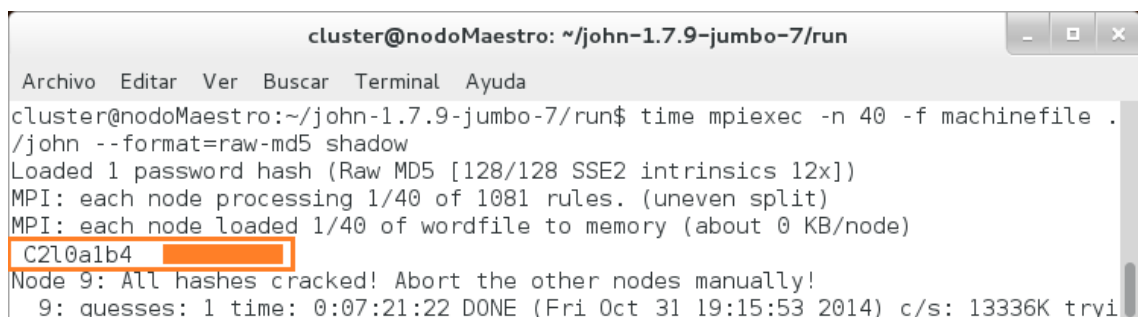
observar la clave descifrada, haciendo uso de los 8 procesador con los que cuenta el nodo maestro. El tiempo de ejecución es de 09:48:34 (9 horas con 48 minutos y 34 seg), como se ilustra en la Fig. 53.



```
cluster@nodoMaestro: ~/john-1.7.9-jumbo-7/run
Archivo Editar Ver Buscar Terminal Ayuda
cluster@nodoMaestro:~/john-1.7.9-jumbo-7/run$ time mpiexec -n 8 -f machinefile ./
john --format=raw-md5 shadow
Loaded 1 password hash (Raw MD5 [128/128 SSE2 intrinsics 12x])
MPI: each node processing 1/8 of 1081 rules. (uneven split)
MPI: each node loaded 1/8 of wordfile to memory (about 3 KB/node)
C2l0a1b4
Node 1: All hashes cracked! Abort the other nodes manually!
1: guesses: 1 time: 0:09:48:34 DONE (Mon Oct 27 18:52:31 2014) c/s: 14806K tryi
```

Figura 53. Descifrado de contraseña con JTR utilizando el nodo maestro.

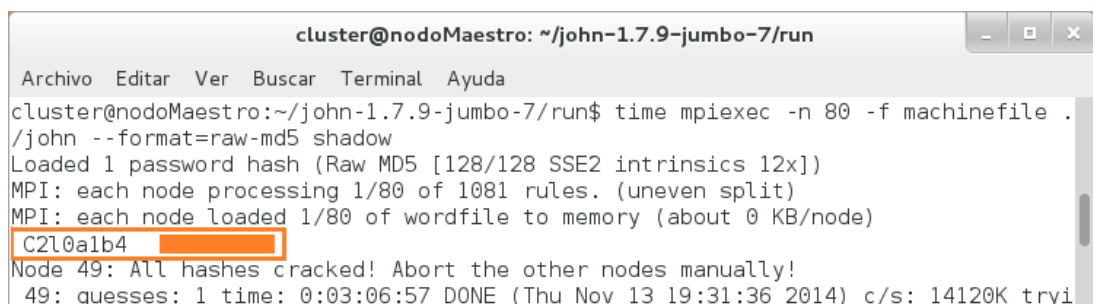
El resultado de desencriptar la contraseña haciendo uso de 5 nodos y un total de 40 procesadores es de 0:07:21:22 (7 horas con 21 minutos y 22 seg), como se ilustra en la Fig. 54 (Descifrado de código con JTR, utilizando 5 nodos).



```
cluster@nodoMaestro: ~/john-1.7.9-jumbo-7/run
Archivo Editar Ver Buscar Terminal Ayuda
cluster@nodoMaestro:~/john-1.7.9-jumbo-7/run$ time mpiexec -n 40 -f machinefile .
/john --format=raw-md5 shadow
Loaded 1 password hash (Raw MD5 [128/128 SSE2 intrinsics 12x])
MPI: each node processing 1/40 of 1081 rules. (uneven split)
MPI: each node loaded 1/40 of wordfile to memory (about 0 KB/node)
C2l0a1b4
Node 9: All hashes cracked! Abort the other nodes manually!
9: guesses: 1 time: 0:07:21:22 DONE (Fri Oct 31 19:15:53 2014) c/s: 13336K tryi
```

Figura 54. Descifrado de código con JTR, utilizando 5 nodos.

El resultado de desencriptar la contraseña haciendo uso de 10 nodos y un total de 80 procesadores es de 0:03:06:57 (tres horas con 06 minutos y 57 seg), como se ilustra en la Fig. 55 (Descifrado de contraseña con JTR, utilizando 10 nodos).



```
cluster@nodoMaestro: ~/john-1.7.9-jumbo-7/run
Archivo Editar Ver Buscar Terminal Ayuda
cluster@nodoMaestro:~/john-1.7.9-jumbo-7/run$ time mpiexec -n 80 -f machinefile .
/john --format=raw-md5 shadow
Loaded 1 password hash (Raw MD5 [128/128 SSE2 intrinsics 12x])
MPI: each node processing 1/80 of 1081 rules. (uneven split)
MPI: each node loaded 1/80 of wordfile to memory (about 0 KB/node)
C2l0a1b4
Node 49: All hashes cracked! Abort the other nodes manually!
49: guesses: 1 time: 0:03:06:57 DONE (Thu Nov 13 19:31:36 2014) c/s: 14120K tryi
```

Figura 55. Descifrado de contraseña con JTR, utilizando 10 nodos.

El resultado de descifrar la contraseña haciendo uso de toda la arquitectura clúster, 15 nodos y un total de 120 procesadores es de 0:00:27:24 (27 minutos y 24 seg), como se ilustra en la Fig. 56 (*Descifrado de código con JTR, utilizando 15 nodos*).

```
cluster@nodoMaestro: ~/john-1.7.9-jumbo-7/run
Archivo Editar Ver Buscar Terminal Ayuda
cluster@nodoMaestro:~/john-1.7.9-jumbo-7/run$ time mpiexec -n 120 -f machinefile ./
john --format=raw-md5 shadow
Loaded 1 password hash (Raw MD5 [128/128 SSE2 intrinsics 12x])
MPI: each node processing 1/120 of 1081 rules. (uneven split)
MPI: each node loaded 1/120 of wordfile to memory (about 0 KB/node)
C2l0a1b4
Node 9: All hashes cracked! Abort the other nodes manually!
 9: guesses: 1 time: 0:00:27:24 DONE (Thu Nov 13 10:31:36 2014) c/s: 15295K try
```

Figura 56. Descifrado de código con JTR, utilizando 15 nodos.

En la Fig. 57 (*Balaceo de carga en el clúster al ejecutar MPICH+JTR*), observamos el balanceo de carga entre los distintos nodos que componen el clúster.

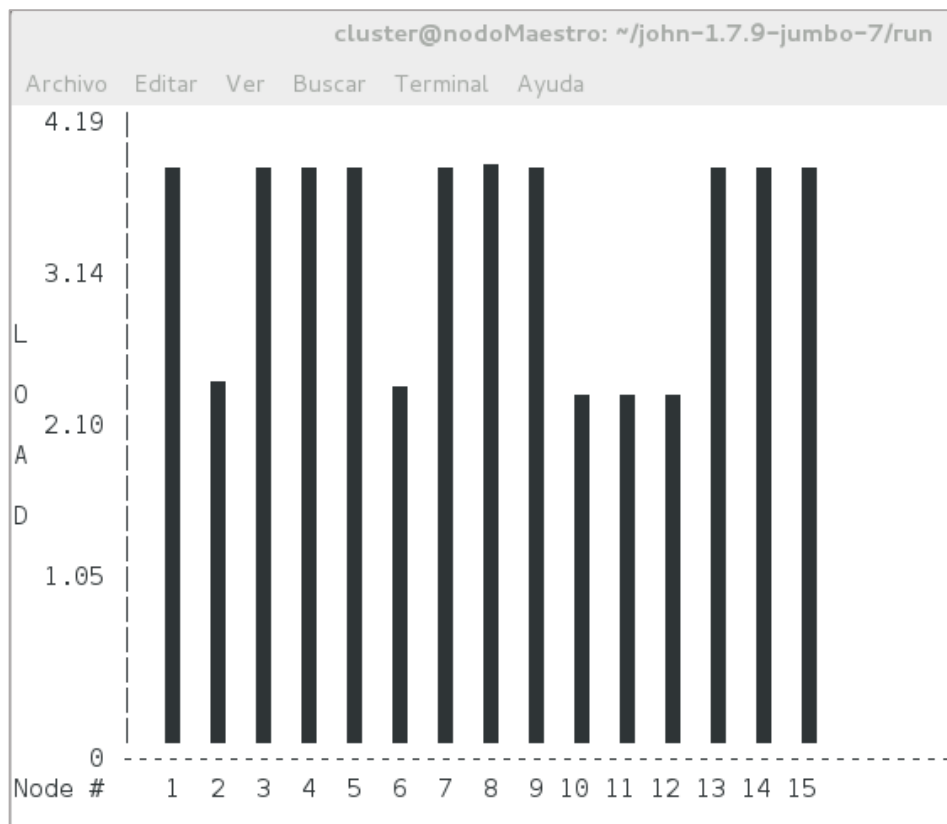


Figura 57. Balanceo de carga en el clúster al ejecutar MPICH+JTR.

Anexo 7: Renderización de imágenes y videos con blender

En la Fig. 58 (*Proceso de renderización de imágenes y vídeos con blender*), se muestra la interfaz gráfica del software blender que permitirá el proceso de renderizado de imágenes y vídeos, para realizar este proceso utilizando un nodo, seleccionamos la opción **Blender Render**, en cambio para realizar el proceso de renderización en mas de un nodo seleccionamos la opción **Network Render**, esta opción se activará un panel de opciones en la parte derecha de la interfaz donde seleccionaremos el nodo master, cliente y esclavos.

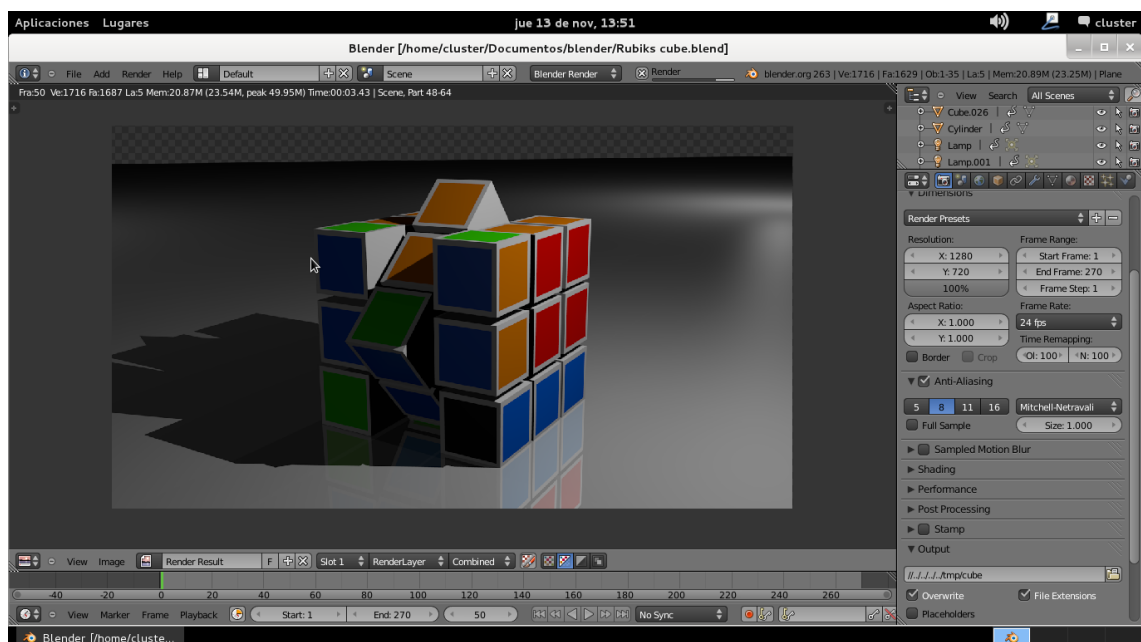


Figura 58. Proceso de renderización de imágenes y vídeos con blender.

En la Fig. 59 (*Proceso de renderizado de imágenes y video con blender utilizando los recursos del clúster*), el proceso de renderización de cada uno de los nodos del clúster. El tiempo de renderizado utilizando un solo nodo con la animación del cubo de rublik es de 15 minutos con 40 segundos, mientras que con la utilización de los recursos de la arquitectura clúster el tiempo de render es de 1 minuto con 31 segundos como se observa en la Fig. 59 (*Proceso de renderizado de imágenes y video con blender utilizando los recursos del clúster*).

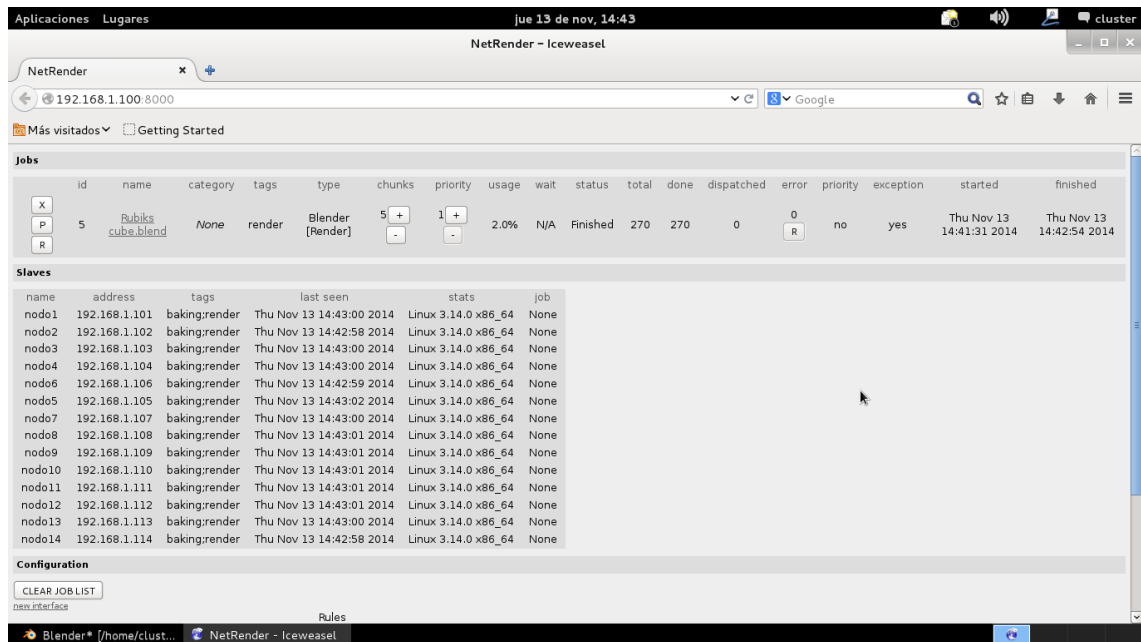


Figura 59. Proceso de renderizado de imágenes y video con blender utilizando los recursos del clúster.

En la Fig. 60 (*Proceso de renderizado finalizado*), observamos el cubo de rubik armado, luego de haber terminado el proceso de render en la arquitectura clúster.

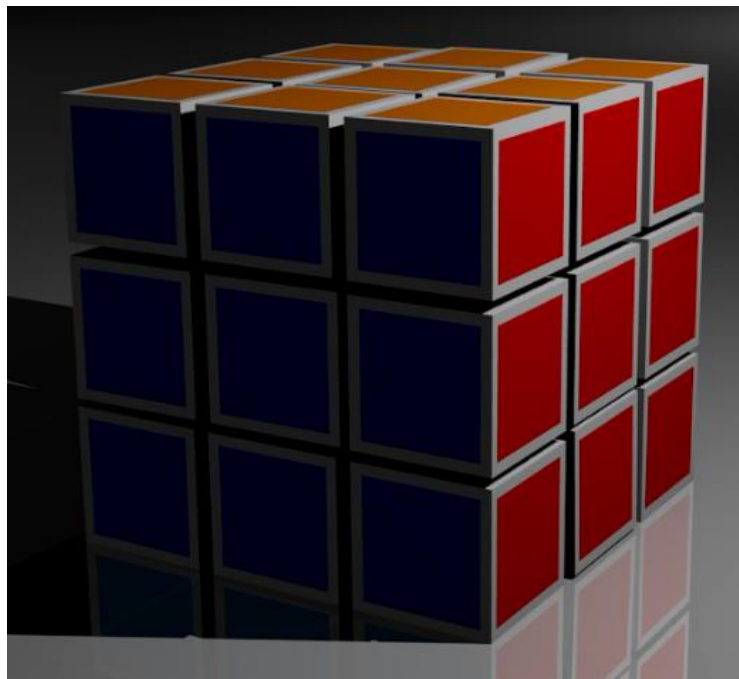


Figura 60. Proceso de renderizado finalizado.

Anexo 8: Análisis de datos big data con hadoop

Una vez levantado los servicios de hadoop, la forma de realizar el análisis de los datos es mediante los siguientes parámetros.

- `$ hadoop jar FICHERO_JAR.jar ClaseMain [parámetros]`

Primeramente ingresamos al directorio donde tenemos la aplicación desarrollado con la utilización de librerías apache hadoop; en nuestro caso se encuentra en el directorio */home/cluster/workspace/mapreduce-basico-master*, una vez que hemos ingresado realizamos los siguientes pasos.

8.1. Copiar el fichero en el HDFS

Lo primero que debemos hacer es crear una carpeta en el HDFS y colocar nuestro fichero dentro de dicho directorio. El objetivo de HDFS es dividir el fichero en bloques de tamaño fijo y distribuirlo en los distintos nodos del clúster como se observa en la Fig. 61 (*Distribución de bloques de tamaño fijo en los nodos del clúster*), esto lo podemos hacer con el siguiente comando:

- `$ hdfs dfs -mkdir /clima`
- `$ hdfs dfs -put /input/calidad_del_aire_cyl_1997_2013.csv /clima`

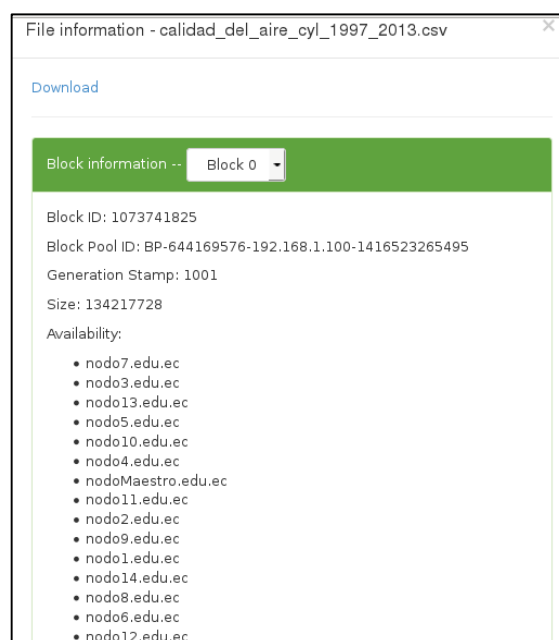


Figura 61. Distribución de bloques de tamaño fijo en los nodos del clúster.

8.2. Ejecución del programa desarrollado con librerías hadoop

Una vez realizado el proceso anterior con éxito, ya podemos hacer el análisis de los datos con hadoop ejecutando el siguiente comando.

- `$ hadoop jar target/mapreduce-basico-0.0.1-SMAPSHOT.jar com.autentia.tutoriales.AirQualityManager /clima/calidad_del_aire_cyl_1997_2013.csv /clima/output`

Donde:

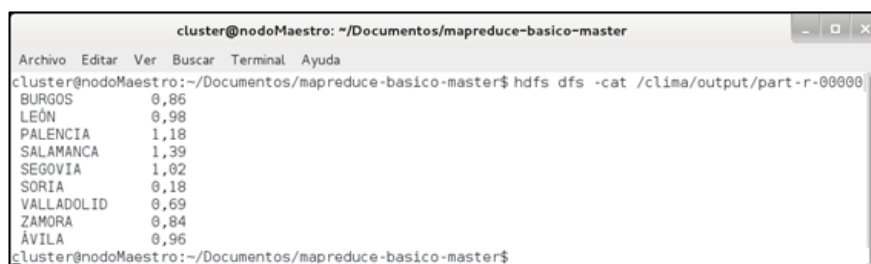
- **hadoop:** ejecuta los binarios de hadoop
- **jar:** ejecuta los ficheros .jar
- **target/mapreduce-basico-0.0.1-SMAPSHOT.jar:** aplicación .jar desarrollada para el análisis de los datos.
- **com.autentia.tutoriales.AirQualityManager:** clase main del fichero .jar que vamos a ejecutar
- **/clima/calidad_del_aire_cyl_1997_2013.csv:** fichero .csv el cual vamos a analizar
- **/clima/output:** directorio donde almacenaremos los resultados del análisis realizado

8.3. Resultado del análisis de datos

Para poder ver el resultado del análisis realizado los podemos hacer con el siguiente comando:

- `$ hdfs dfs -cat /clima/output/part-r-000000`

En la Fig. 62 (*Resultado del análisis de datos sobre la calidad del aire*), se ilustran los resultados obtenidos del análisis utilizando el framework Apache Hadoop.



```
cluster@nodoMaestro: ~/Documentos/mapreduce-basico-master
Archivo Editar Ver Buscar Terminal Ayuda
cluster@nodoMaestro:~/Documentos/mapreduce-basico-master$ hdfs dfs -cat /clima/output/part-r-000000
BURGOS 0,86
LEÓN 0,98
PALENCIA 1,18
SALAMANCA 1,39
SEGOVIA 1,02
SORIA 0,18
VALLADOLID 0,69
ZAMORA 0,84
ÁVILA 0,96
cluster@nodoMaestro:~/Documentos/mapreduce-basico-master$
```

Figura 62. Resultado del análisis de datos sobre la calidad del aire.

Anexo 9: Certificación Summary

Loja, 05 de Diciembre de 2014

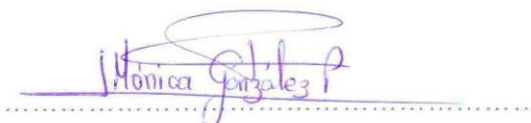
Lic. Mónica Cristina González

**ENGLISH LANGUAGE TEACHER
ESCUELA DE EDUCACIÓN GENERAL BÁSICA FISCOMISIONAL "MATER DEI"**

CERTIFICA

Que los egresados: Leonardo Rafael Chuquiguanca Vicente y Edyson Javier Malla Bustamante, autores del trabajo titulado **"ARQUITECTURA CLÚSTER DE ALTO RENDIMIENTO UTILIZANDO HERRAMIENTAS DE SOFTWARE LIBRE"**, en la traducción del resumen han cumplido con todas las normas y reglas gramaticales del idioma inglés, las cuales han sido revisadas minuciosamente para dar cumplimiento con la sección summary.

Es todo cuanto puedo decir en honor a la verdad, pudiendo los interesados hacer uso del presente en lo que estimen conveniente.



Mónica Cristina González
**LICENCIADA EN CIENCIAS DE LA EDUCACIÓN
ESPECIALIDAD INGLÉS**

Anexo 10: Certificación Revisión Literaria

Loja, 04 de Enero del 2015

Lic. Mercy Suárez Yanza

DOCENTE DE LENGUA Y LITERATURA
COLEGIO DE BACHILLERATO "HERNAN GALLARDO MOSCOSO"

CERTIFICA:

Que los egresados: Leonardo Rafael Chuquiguanca Vicente y Edison Javier Malla Bustamante, autores del trabajo titulado: "ARQUITECTURA CLÚSTER DE ALTO RENDIMIENTO UTILIZANDO HERRAMIENTAS DE SOFTWARE LIBRE" que en la redacción e interpretación del contenido de su trabajo, han cumplido con todas las normas y reglas ortográficas, las mismas que han sido revisadas minuciosamente.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo los interesados hacer uso del presente en lo que estimen conveniente.



Mercy Suárez Yanza

LICENCIADA EN CIENCIAS DE LA EDUCACIÓN
ESPECIALIDAD LENGUA Y LITERATURA

Anexo 11: Licencia Creative Commons



Arquitectura clúster de alto rendimiento utilizando herramientas de Software Libre by Leonardo Rafael Chuquiguanca Vicente and Edyson Javier Malla Bustamante is licensed under a [Creative Commons Reconocimiento-NoComercial 4.0 internacional License](https://creativecommons.org/licenses/by-nc/4.0/).