DISEÑO DE UN PROTOTIPO WORKLOAD PARA MEDIR LA EFICIENCIA DE E/S EN UN CLUSTER HPCC ORANGEFS

PRESENTACIÓN DE PROYECTO DE GRADO COMO REQUISITO PARA OPTAR AL TÍTULO DE INGENIERO DE SISTEMAS DE LA INSTITUCIÓN UNIVERSITARIA POLITÉCNICO GRANCOLOMBIANO

CARLOS A CHARRIS S, DIEGO F MEZA P

Marzo 2017

DISEÑO DE UN PROTOTIPO WORKLOAD PARA MEDIR LA EFICIENCIA DE E/S EN UN CLUSTER HPCC CON ORANGEFS

CARLOS ANDRES CHARRIS SANDOVAL

DIEGO FERNANDO MEZA PATACON

DIRECTOR
ALEXIS ROJAS CORDERO
Profesor de Planta

INSTITUCIÓN UNIVERSITARIA POLITÉCNICO GRANCOLOMBIANO FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
BOGOTÁ
2017

Certifico que he leído esta tesis y en mi opinión, es totalmente adecuada en calidad y alcance como una tesis para el grado de ingeniero de sistemas.

Certifico que he leído esta tesis y en mi opinión, es totalmente adecuada en calidad y alcance como una tesis para el grado de ingeniero de sistemas.

Agradecimientos

Inicialmente nos gustaría mostrar nuestro más sincero agradecimiento y gratitud a nuestro director de tesis, el profesor Alexis Rojas, por el tiempo que nos dedicó durante este semestre debido a que, sin su consejos y ayuda, no hubiera sido posible concluir este trabajo de investigación. Su constante perseverancia en el mundo de la investigación son las cualidades que son dignas de mencionar.

En momentos en donde no se lograba ver una luz en el camino y se pensó en rediseñar el alcance de los objetivos de investigación apareciste Elaine Quarles (Desarrolladora Software OrangeFs), con tu humildad, paciencia e infinita colaboración brindándonos las guías y orientaciones necesarias para volver a retomar ánimos y cumplir con las metas establecidas inicialmente, mil y mil gracias por esa ayuda incondicional.

Después de una ardua tarea de investigación se logró la el desarrollo y ejecución de estas tecnologías, esta investigación no hubiera sido posible sin el valioso aporte de los destacados autores y especialistas en esta materia. Motivo por el cual también los queremos hacer partícipes de este reconocimiento.

A nuestros amigos y todas aquellas personas que, de una forma u otra, lograron contribuir en que se pudiera llevar a cabo esta investigación

Por ultimo y de forma especial, a nuestras familias, por su comprensión y ánimo, sin el apoyo de ellos hubiera sido posible culminar este trabajo.

Resumen

Alan Turing demostró teóricamente, que diseñar una maquina secuencial y de propósito general era posible y esta podría realizar eficientemente cálculos como los que realizaba una máquina de propósito específico. Luego Von Neumann realizo su propuesta de un computador secuencial de propósito general donde usaba los principios de Turing en su método teórico. Este modelo ha sido base para un gran número de computadores secuenciales desde su presentación hasta nuestros días.

En búsqueda de un mayor rendimiento de las aplicaciones, se ha crecido el uso de la computación en paralelo en distintas áreas conforme a la categorización del Dr. flynn, una de ellas ha sido en el álgebra lineal en donde ha existido un mayor esfuerzo generando nuevos desarrollos de algoritmos en paralelo, uno de ellos ha sido la resolución de sistemas lineales por medio de métodos directos o iterativos.

A principios de los años 90, los computadores paralelos empiezan a evolucionar a un modelo arquitectónico estándar, esta evolución se da debido a la rápida evolución de los procesadores, esto es con base en la publicación de Gordon Moore (co-fundador de la compañía Intel en 1968) en donde menciona que la cantidad de transistores por centímetro cuadrado en un circuito integrado se va a duplicar en cada año y que este será el comportamiento durante las próximas dos décadas. La evolución de los supercomputadores disminuyó debido al alto costo que esto generaba y de ahí muchos fabricantes (Intel, IBM) empezaron a trabajar con los procesadores existentes y dejaron de diseñar nuevas máquinas; esto dio paso para dejar de construir computadores monoprocesador y se empezaran a enfocar en las estructuras de memoria distribuida. Una de las más grandes finalidades de esto, es el desarrollo de software escalable sin importar el tamaño del problema o el número de procesadores.

Internet fue uno de los ambiciosos proyectos de red de DARPA (Defense Advanced Research Projects Agency). Fue utilizado solamente por pocos departamentos militares en los EEUU para el intercambio de la información. Posteriormente, este ambicioso proyecto fue aceptado y utilizado por muchos, para formar una red de computadoras a través de las instituciones de investigación y educativas que posteriormente se expandieron a otros dominios para formar la World Wide Web. Actualmente Internet se convirtió en una parte de nuestra vida de alta tecnología, algunos de nosotros no podemos imaginar a vivir sin ella. La estamos utilizando para realizar intercambio de información y navegación, el intercambio de correo y la publicación, el comercio electrónico y otros que lo hacen como una autopista de la información del mundo actual. Aceptamos que Internet ha cambiado la vida en muchos aspectos de las comunicaciones y otras utilidades

de la vida. Internet se ha convertido en uno de los miembros de la lista de productos del mundo de la informática.

Contenido

Tabla de contenido

LISTA DE IMÁGENES ..... **VIII**

LISTA DE TABLAS ..... **X**

GLOSARIO ..... **- 11 -**

1 INTRODUCCIÓN **1**

1.1 EVOLUCIÓN TECNOLÓGICA 6
1.2 LÍNEA DE TIEMPO HPC 8
1.3 TIPOS DE PROBLEMAS QUE SOLUCIONA 10

2 DISEÑO Y MONTAJE ESTRUCTURA **12**

2.1 **DISEÑO** 12
2.2 **PLANTEAMIENTO DEL PROBLEMA** 14
2.3 **OBJETIVOS** 14
2.3.1 Objetivo general *14*
2.3.2 Objetivos específicos *14*
2.4 **DELIMITACIÓN** 15
2.4.1 Tiempo *15*
2.4.2 Alcance *15*

3 MARCO TEÓRICO **16**

3.1 **¿QUÉ ES LUSTRE?** 16
3.1.1.1 Componentes de Lustre 17
3.1.1.2 Funcionamiento de Lustre 18
3.2 **¿QUÉ ES GANGLIA?** 19
3.2.1.1 Arquitectura de Ganglia 20
3.3 **¿QUÉ ES NMON?** 22
3.3.1.1 Características de Nmon 22
3.4 **¿QUÉ ES IOR?** 25
3.5 **¿QUÉ ES ORANGEFS?** 26
3.5.1.1 PVFS 26
3.5.1.2 PVSF2 28
3.5.1.3 ARQUITECTURA DEL SISTEMA DE FICHEROS PVFS2 28
3.5.1.4 ESTRUCTURA HARDWARE 28
3.5.1.5 **ARQUITECTURA SOFTWARE** 29

4 MONTAJE FÍSICO **31**

4.1 VISUALIZACIÓN DE LOS NODOS DESDE EL NODO MASTER 32
4.2 SHELL PARA MONTAJE LUSTRE 32

5 PROYECTOS MONTADOS EN EL CLÚSTER **33**

6 ANEXOS **42**

6.1 ENSAMBLAJE DEL CLUSTER 42
6.2 CONFIGURACIÓN DE LOS NODOS 53
6.2.1.1 Configuración del Nodo No. 1 53

6.2.1.2 Configuración del Nodo No. 2 54
6.2.1.3 Configuración del Nodo No. 3 55
6.2.1.4 Configuración del Nodo No. 4 56
6.2.1.5 Configuración del Nodo No. 5 57
6.2.1.6 Configuración del Nodo No. 6 58
6.3 CONFIGURACIÓN DEL NODO MASTER 59
Configuración del Nodo Master 59
BIBLIOGRAFÍA Y REFERENCIAS 40

Lista de Imágenes

Ilustración 1 Clúster GSCRATCH/PROJECT Recuperado de https://www.nersc.gov/ 1
Ilustración 2 Vista Esquemática del SuperMUC Recuperado de https://www.lrz.de (Leibniz Supercomputing Center) 2
Ilustración 3 Ejemplo curvas de Speedup. Adaptado de https://nex.nasa.gov 3
Ilustración 4 Tabla de Eficiencia. Adaptado de https://nex.nasa.gov 3
Ilustración 5 Supercomputador 10-Petaflow. Recuperado de http://www.extremetech.com 4
Ilustración 6 Grilla Computacional 4
Ilustración 7 Grilla Computacional de Clúster Distribuidos. Recuperado de: http://adarshpatil.com..... 5
Ilustración 8 Diseño de una arquitectura HPC. Recuperado de: http://runge.math.smu.edu..... 9
Ilustración 9 Problemas que Soluciona HPC. Recuperado de: https://www.fing.edu.uy..... 10
Ilustración 10 Los datos a través del tiempo, evolución de los computadores paralelos. Recuperado de: (top500, 2006)..... 11
Ilustración 11 Diseño y estructura en Virtual Box. Recuperado de: el Autor 12
Ilustración 12 Diseño y estructura física Institución Universitaria Politécnico Grancolombiano. Recuperado de: el Autor 13
Ilustración 13 Esquema componentes de Lustre. Adaptada de: http://www.systerminal.com 17
Ilustración 14 Funcionamiento de Lustre. Adaptada de: http://www.systerminal.com 18
Ilustración 15 Arquitectura Ganglia. Adaptado de: https://www.novell.com 21
Ilustración 16 Ganglia corriendo en un clúster Scalable Grid Engine HPC en Amazon EC2. Recuperado de: https://en.wikipedia.org 22
Ilustración 17 Inicio Nmon. recuperado de: https://devops.profitbricks.com/ 23
Ilustración 18 Comprobar las estadísticas de la red pulsando n. Recuperado de: https://devops.profitbricks.com 24
Ilustración 19 Comprueba la CPU por procesador pulsando c. Recuperado de: https://devops.profitbricks.com/ 24
Ilustración 20 El diseño del benchmark IOR para el tipo de archivo compartido. Los bloques se almacenan en archivos separados para el modo de operación de 1 archivo por procesador. Recuperado de: https://cug.org..... 25
Ilustración 21 Configuración SAN. Recuperado de: https://www.uaeh.edu.mx 27
Ilustración 22 Beneficio del almacenamiento de cada nodo del clúster como un sistema virtual de archivos. Recuperado de: https://www.uaeh.edu.mx 27
Ilustración 23 Arquitectura de PVFS2. Recuperado de: https://www.uaeh.edu.mx 28
Ilustración 24 Arquitectura software de PVFS2. Recuperado de: https://www.uaeh.edu.mx 30
Ilustración 25 Diseño del Clúster 31
Ilustración 26 vista de los nodos desde el nodo Master 32
Ilustración 27 Chasis Clúster..... 42
Ilustración 28 Montaje del primer nodo 42
Ilustración 29 Finalización montaje nodos frontal..... 43
Ilustración 30 Finalización montaje nodos trasera 43
Ilustración 31 Montaje nodo Master frontal 44
Ilustración 32 Montaje nodo Master trasero..... 44

Ilustración 33 Montaje de los Switch 45
Ilustración 34 Cableado de los Switch con los nodos 45
Ilustración 35 Peinado de los cables Parte 1 46
Ilustración 36 Peinado de los cables Parte 2 46
Ilustración 37 Conexión cables de poder Switch 47
Ilustración 38 Encendida y prueba de la corriente en todos los nodos 47
Ilustración 39 Inicio Proceso de Instalación 48
Ilustración 40 BIOS del nodo 48
Ilustración 41 Formateo del nodo parte 1 49
Ilustración 42 Formateo del nodo parte 2 49
Ilustración 43 Formateo del nodo parte 3 49
Ilustración 44 Formateo del nodo parte 4 50
Ilustración 45 Formateo del nodo parte 5 50
Ilustración 46 Formateo del nodo parte 6 50
Ilustración 47 Formateo del nodo parte 7 51
Ilustración 48 Formateo del nodo parte 8 51
Ilustración 49 Formateo del nodo parte 9 51
Ilustración 50 Formateo del nodo parte 10 52
Ilustración 51 Formateo del nodo parte 11 52
Ilustración 52 Configuración nodo 1 53
Ilustración 53 Configuración nodo 1 53
Ilustración 54 Configuración nodo 2 54
Ilustración 55 Configuración nodo 2 54
Ilustración 56 Configuración nodo 3 55
Ilustración 57 Configuración nodo 3 55
Ilustración 58 Configuración nodo 4 56
Ilustración 59 Configuración nodo 4 56
Ilustración 60 Configuración nodo 5 57
Ilustración 61 Configuración nodo 5 57
Ilustración 62 Configuración nodo 6 58
Ilustración 63 Configuración nodo 6 58
Ilustración 64 Configuración nodo Master 59

Lista de Tablas

Tabla 1 Escala de unidades de medida en HPC en bytes. Fuente: los autores. 9
Tabla 2 Tiempos de ejecución del desarrollo de la tesis. Fuente: los autores..... 15

Glosario

Las palabras que componen este glosario que ayudara a entender y comprender mejor la información de este documento fue consultada en la página web de Wikipedia (/www.wikipedia.org)

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

Flop: Operaciones de punto flotante.

Flops/s: Operaciones de punto flotante por cada segundo.

Bytes: para el tamaño de datos de memoria (una variable flotante de doble precisión ocupa 8 bytes).

Rack: Es estructura metálica destinado a almacenar uno o varios equipos electrónicos, informáticos y/o de comunicación.

Router: dispositivo que nos permite podernos conectar a un nivel de red.

Servidor: es una aplicación que se encuentra ejecutándose la cual puede atender las solicitudes de un cliente y retornar una respuesta.

Switch: es el dispositivo de interconexión de equipos que se encuentran operando en la capa de enlace de datos.

Clúster: Son conjuntos o grupos de computadores unidos normalmente por una red de alta velocidad y los cuales se comportan como una única computadora.

Nodo: son computadores básicos o sistemas multiprocesador

NAS (Network Attached Storage), Es una tecnología de almacenamiento que se dedica a compartir la capacidad de almacenamiento de un computador con otros computadores o clientes a por medio de una red.

Almacenamiento: Consiste en una Network Attached Storage, o almacenamiento interno en el servidor. Es protocolo que mayormente se usa es él Network File System.

Sistema operativo: es un programa o conjunto de programas de computadora que tienen como fin de permitir gestionar eficazmente y segura de los recursos.

Middleware: es una capa de software que facilita el proceso de compartir recursos y ejecutar un proceso en sistemas remotos entrelazados, sin importar las plataformas y por lo general actúa entre las aplicaciones y el sistema operativo.

Rocks Clúster: Distribución de Linux que es usada para clústers de computadores de alto rendimiento.

Lustre: es un sistema de archivos distribuido, que normalmente se usa en clústers a gran escala.

NFS: Network File System, es un protocolo de nivel de aplicación. Se usa en sistemas de archivos distribuido dentro de un entorno de red local.

IP (Internet Protocol): es un número que identifica a una interfaz en red de un dispositivo que utilice el protocolo IP.

Mascara de Red: es una combinación de bits que sirve para delimitar el ámbito de una red de ordenadores.

Gateway (puerta de enlace):es el dispositivo que actúa de interfaz de conexión entre aparatos o dispositivos, y también posibilita compartir recursos entre dos o más computadoras.

LAN (Red de Área Local): es una red de computadores que tiene como alcance área reducida a una casa, un departamento, un edificio, o una casa.

Ganglia: Es una herramienta de supervisión escalable y distribuida para sistemas informáticos de alto rendimiento, clústers y redes.

CentOS: Sistema operativo, se opera de manera similar, y como fin es brindar al usuario un software de "tipo empresarial" el cual es gratuito.

CIFS (Common Internet File System): Es un protocolo que nos permite usar archivos compartido los cuales se basan en otras utilidades de red y en Windows .

NFS (Network File System): es un protocolo que permite acceso remoto a un sistema de archivos a través de la red.

FTP (File Transfer Protocol): Protocolo de transferencia de archivos entre sistemas que están conectados a una red la cual se basa en arquitectura cliente-servidor.

TFTP (Trivial file transfer Protocol): Protocolo de transferencia sencillo e idéntico a FTP. TFTP a por lo general se usa para enviar archivos pequeños entre computadores dentro de una red.

GPFS (General Parallel File System): Sistema de archivos distribuido de alto rendimiento que da un acceso simultaneo de alta velocidad a aplicaciones que se están ejecutando en varios nodos de un clúster brindando una visión de una unidad de almacenamiento compartida entre ellos.

NERSC (The National Energy Research Scientific Computing Center): Es un computador de alto desempeño (supercomputador) este contiene varios clústeres de supercomputadores, el más grande es Cori, que fue seleccionado en el quinto lugar de la lista TOP500 de los supercomputadores más rápidos del mundo en noviembre de 2016 y se encuentra en Berkeley, California.

Capítulo 1

1 Introducción

Podríamos establecer que el procesamiento en paralelo es usado desde los primeros computadores, en los años 50, IBM adiciona un hardware encargado del procesamiento de números decimales (co-procesador) e inicia el proyecto para crear un "supercomputador". Con este proyecto se quería lograr una maquina con capacidad computacional, 100 veces mayor a la de cualquier maquina existente. Durante el transcurso del mismo año, el proyecto Livermore Automatic Research Computer comienza a crear otro "supercomputador" para el Lawrence Livermore National Laboratory. Estos dos proyectos tomaran un tiempo de tres años para ser construidos, los cuales fueron conocidos como STRETCH y LARC (Domingo Gimenéz, Mantas, & Vidal, 2008)

En paralelo a estas iniciativas, muchas otras producen computadores con las arquitecturas más variadas y con diferentes tipos de software.

GSCRATCH y **PROJECT** son dos sistemas de archivos en The National Energy Research Scientific Computing Center que uno puede tener acceso en la mayoría de los sistemas computacionales. Ambos se basan en el sistema de archivos IBM GPFS (General Parallel File System) y tienen múltiples racks de servidores dedicados y arreglos de discos.

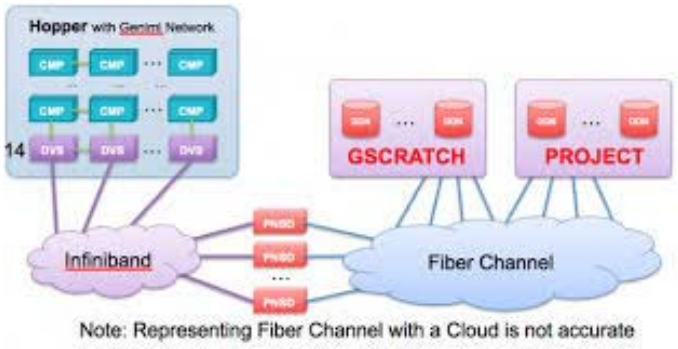


Ilustración 1 Clúster GSCRATCH/PROJECT Recuperado de https://www.nersc.gov/

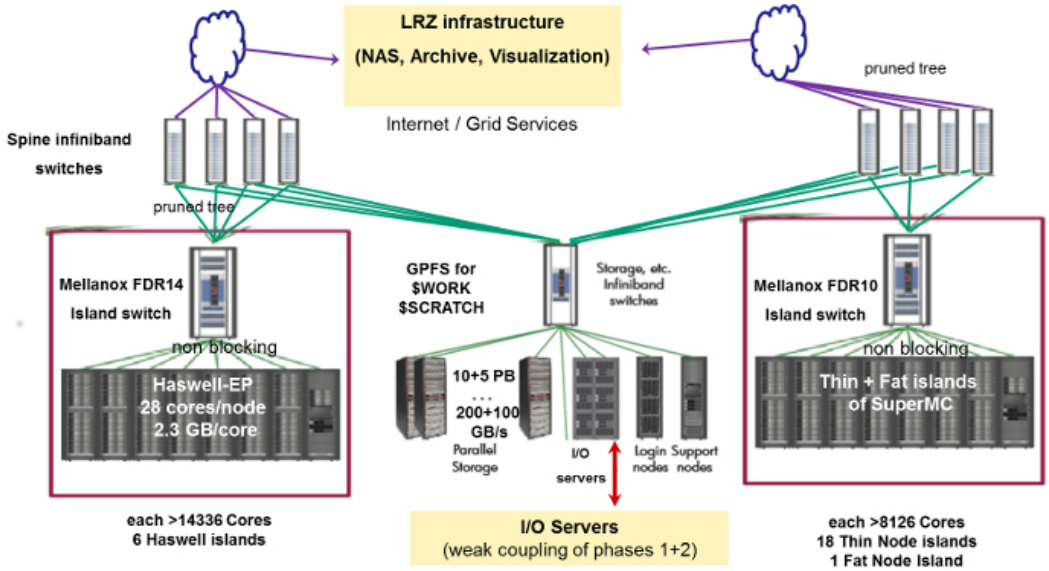


Ilustración 2 Vista Esquemática del SuperMUC Recuperado de https://www.lrz.de (Leibniz Supercomputing Center)

SuperMUC Phase 1 y Phase 2 sólo se suelen acoplar a través de los sistemas de archivos GPFS y NAS, usados tanto por la Fase 1 como por la Fase 2. No es posible ejecutar en un solo trabajo entre Fase1 y Fase2. Las clases de planificación y empleo de Phase1 y Phase2 son diferentes. Sin embargo, Phase1 y Phase2 comparten el mismo entorno de programación. (Supercomputing, 2011)

Las principales características para la generación de máquinas paralelas son: disminuir el tiempo de ejecución de una aplicación, poder resolver problemas mucho más complejos, de grandes tamaños y poder realizar la ejecución simultanea de tareas.

Durante los últimos 10 años, la computación avanza hacia la presencia continua de la computación paralela, esto se da por que las redes de interconexión han avanzado y continúan haciéndolo significativamente en razón de la velocidad de comunicación y ancho de banda.

El procesamiento en paralelo es la manera en que se permite la conexión de muchos dispositivos de cómputo y que estos trabajen de manera simultánea para resolver una actividad o problema. Por tradición, el paralelismo es utilizado en centros de supercomputación para la solución de problemas de gran tamaño, pero en los últimos años se ha extendido por la difusión de los procesadores con varios núcleos; podemos ver su medida de desempeño mediante la siguiente formula (NASA, 2012):

$$E = \frac{S}{t} = \frac{(\frac{T_{serial}}{T_{parallel}})}{t} = \frac{T_{serial}}{t\ x\ T_{parallel}}$$

Velocidad (Speedup) **S** es una medida que se obtiene de: $S = \frac{T_{serial}}{T_{parallel}}$

En procesadores **p** la velocidad perfecta seria: $S = p$

Generalmente $S < p$ debido a los consumos generales.

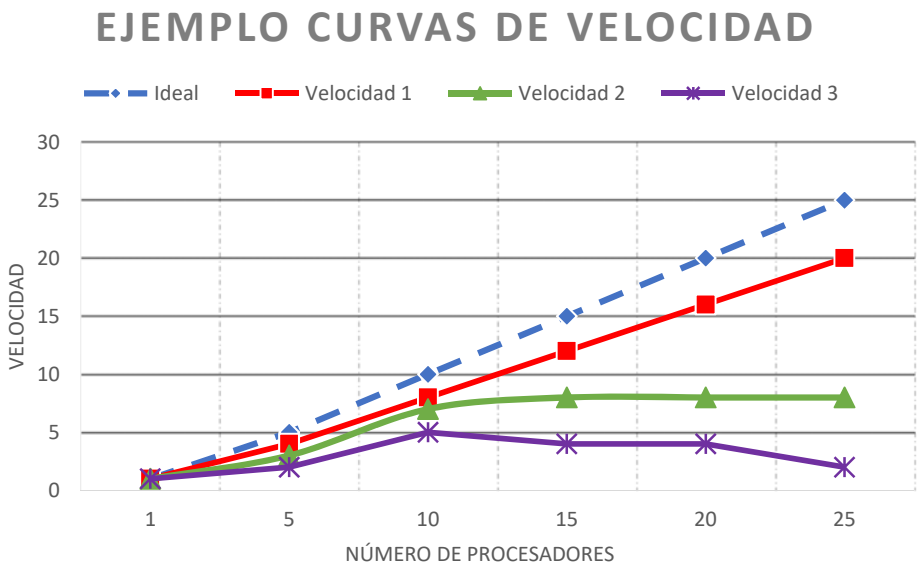


Ilustración 3 Ejemplo curvas de Speedup. Adaptado de https://nex.nasa.gov

La eficiencia la podemos calcular mediante la siguiente formula: $E = \frac{S}{p} = \frac{T_{serial}}{p \; x \; T_{parallel}}$

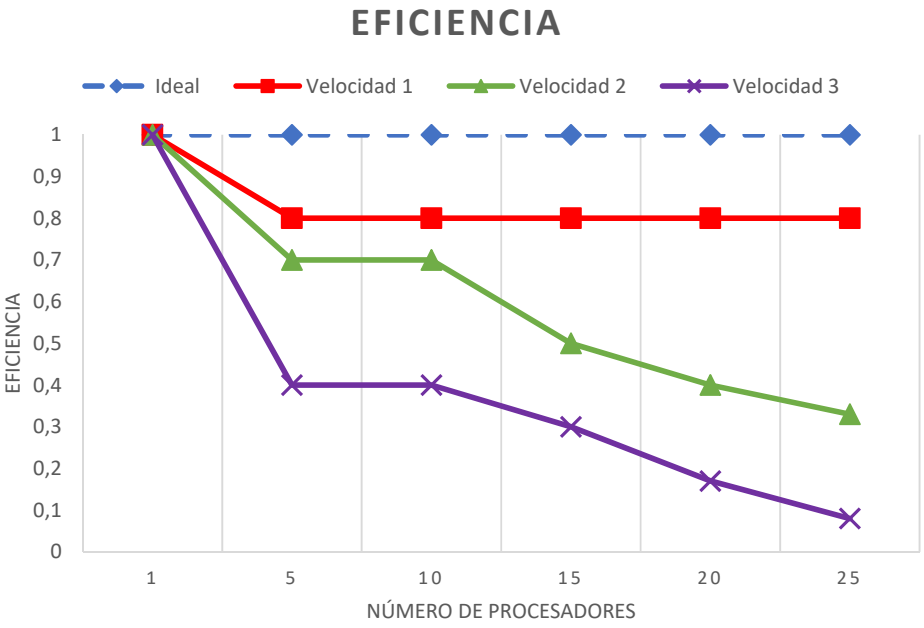


Ilustración 4 Tabla de Eficiencia. Adaptado de https://nex.nasa.gov

Entre las aplicaciones más comunes están las simulaciones aerodinámicas, el estudio de tsunamis y terremotos, la predicción del tiempo o la secuenciación del genoma humano, en el estudio de las interacciones entre proteínas y el modelado de las moléculas que poseen relación con la creación de nuevos fármacos

Ilustración 5 Supercomputador 10-Petaflow. Recuperado de http://www.extremetech.com

Se necesitaba conseguir un mayor almacenamiento, aprovechamiento de recursos, potencia de cálculo, etc. Al combinar los recursos computacionales de varias entidades u organizaciones, esto era necesario ya que con un único nodo no se puede realizar grandes volúmenes de trabajo; pero al combinar varios nodos trabajando si lo logran. Esto, en naturaleza es la Computación **Grid,** es un paradigma de la computación distribuida que fue propuesta por Carl Kesselman y Ian Foster a mediados de años noventa. (Foster & Kesselman, 1998)

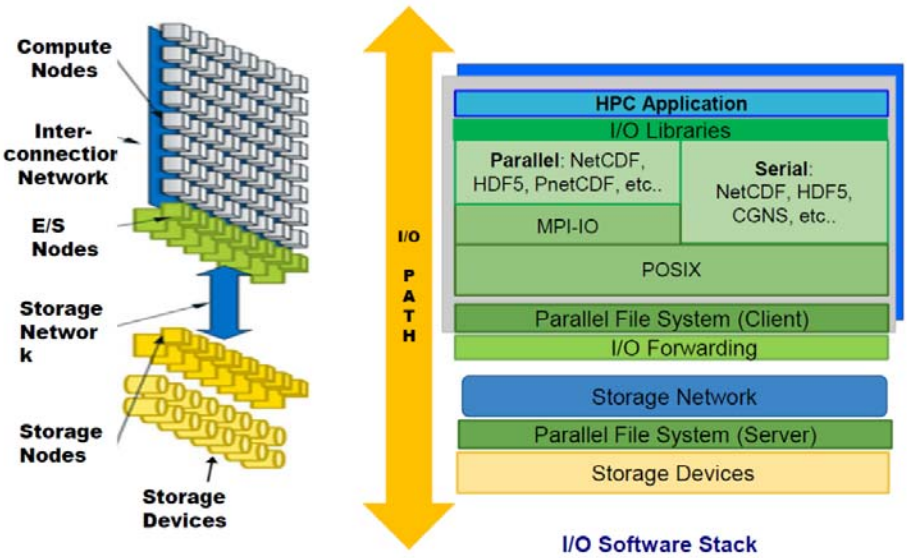


Ilustración 6 Grilla Computacional

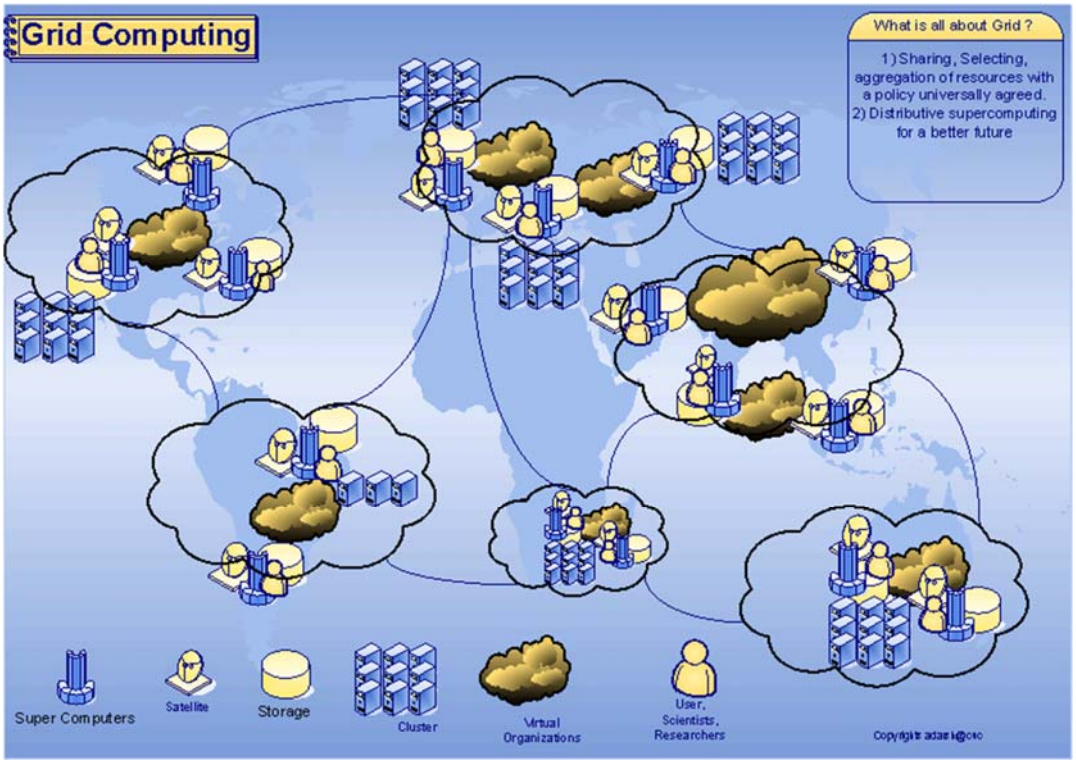


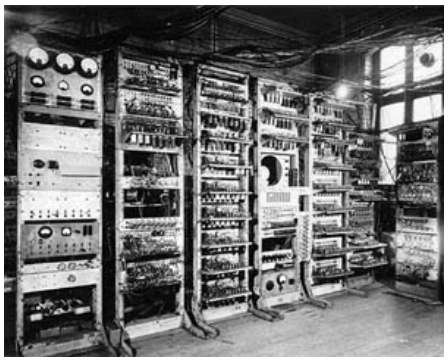
Ilustración 7 Grilla Computacional de Clúster Distribuidos. Recuperado de: http://adarshpatil.com

Una grilla computacional es un sistema que:

- Utiliza un estándar abierto de protocolos de propósito general e interfaces.
- Tiene recursos coordinados y no están sujetos a un único control.
- No ofrece las cualidades triviales de un servicio cualquiera.

Su principal objetivo es crear la ilusión de una simple, pero poderosa organización virtual de ordenadores heterogéneos en red, para compartir diversas combinaciones de recursos (diferentes aplicaciones de usuario y distintos sistemas operacionales).

1.1 Evolución de la Tecnológica Paralela



Collosus 2, 1er computador paralelo: 50.000 operaciones x seg (UK)

IBM NORC, 67.000 operaciones x seg (USA)

MEGAFLOP COMPUTER

1938 — **1946** — **1948** — **1954** — **1964**

Zuse Z1 (Ale), primer computador mecánico: 1 operaciones x seg

ENIAC (USA), 5.000 operaciones x seg

IBM 7030 “Stretch”, 1.2 MFLOPS (USA)

GIGAFLOP COMPUTER

Cray-2/8: 3.9 GFLOPS (USA)

1984

1985 - 89

1964

GIGAFLOP COMPUTER
M-13: 2.4 GFLOPS
(URSS)

TERAFLOP COMPUTER

ASCI Red/9152: 1.338 TFLOPS
(USA)

TERAFLOP COMPUTER

1.2 Línea de Tiempo HPC

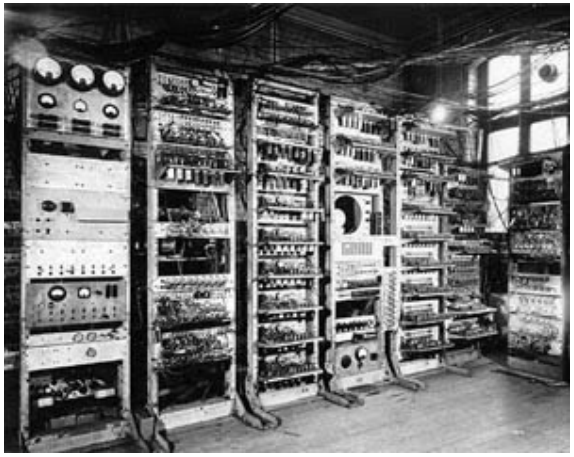
Sunway TaihuLigth (China, 2017):
93×10^{15} operaciones/s.

ASCI Red/9152:
10^{12} operaciones/s. (USA)

Jaguar (LANL, USA, 2010):
10^{15} operaciones/s

M-13: 10^{9} operaciones/s. (URSS)

IBM 7030 "Stretch“ IBM 7030 "Stretch“(LANL, USA, 1961):
10^{6} operaciones/s.



Colossus Mark 2 (UK, 1946):
5.0 operaciones/s.

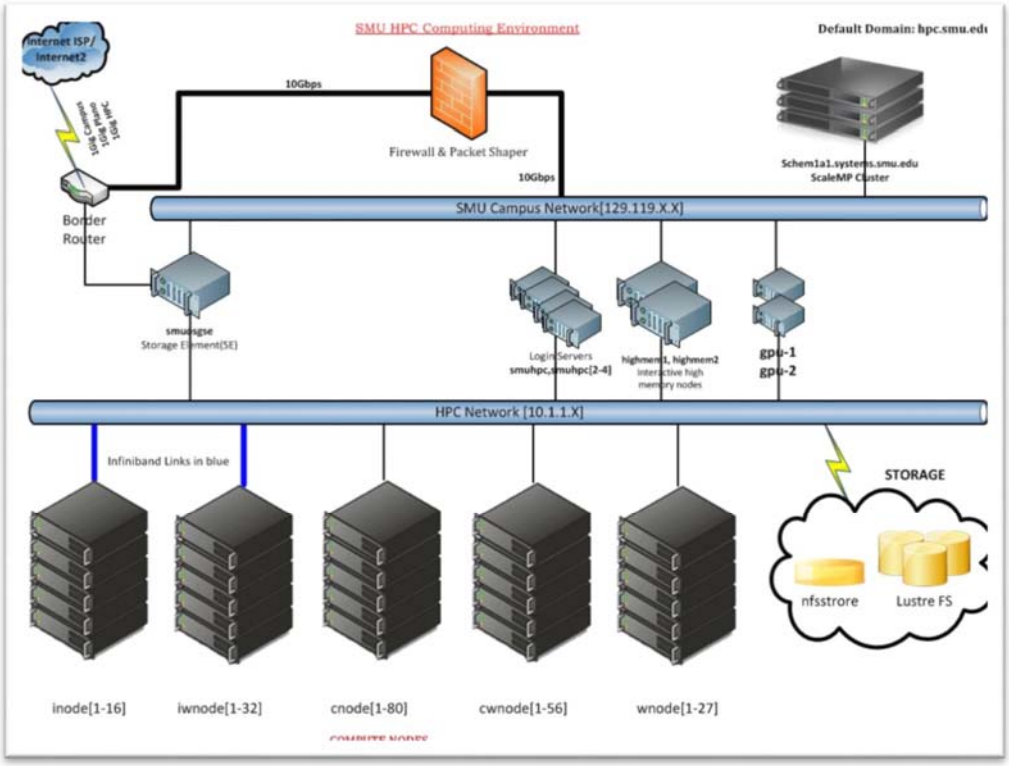


Ilustración 8 Diseño de una arquitectura HPC. Recuperado de: http://runge.math.smu.edu

Analicemos la ilustración 9 en términos de bytes:

Megabyte	**Mflop/s 106 flop x seg**	**Mbyte: 220 hasta 1048576 - 106 bytes**
Gigabyte	Gflop/s 109 flop x seg	Gbyte: 230 hasta 109 bytes
Terabyte	Tflop/s 1012 flop x seg	Tbyte: 240 hasta 1012 bytes
Petabyte	Pflop/s 1015 flop x seg	Pbyte: 250 hasta 1015 bytes
Exabyte	Eflop/s 1018 flop x seg	Ebyte: 260 hasta 1018 bytes
Zettabyte	Zflop/s 1021 flop x seg	Zbyte: 270 hasta 1021 bytes
Yottabyte	Yflop/s 1024 flop x seg	Ybyte: 280 hasta 1024 bytes

Tabla 1 Escala de unidades de medida en HPC en bytes. Fuente: los autores.

El High Performance Computing (HPC) es una de los paradigmas de computación que es usada para crear supercomputadoras. Se caracteriza por la necesidad de gran poder de computo según lo podemos ver en la tabla No. 1. en periodos de tiempos muy cortos, otra característica es dividiré en pequeños pedazos para que cada pieza pueda ser ejecutada simultáneamente por procesadores independientes. (Timtthy, 2010)

El uso de computadores de HPC es fundamental para que los científicos puedan simular problemas complejos. Los problemas son cada vez más costosos computacionalmente:

- Los sistemas de HPC son requeridos para realizar cálculos.
- Es fundamental aprovechar el paralelismo en todas las fases.

Las áreas críticas de investigación usan cada vez más datos. El acceso a los datos es un gran desafío:

- Usar el paralelismo en la E/S (Es el movimiento de datos desde memoria principal a disco y viceversa. Es realizada por medio de Patrones de E/S y números de ficheros; la cual es realizada siguiendo la arquitectura E/S y es almacenada en los dispositivos de E/S) para obtener mayor rendimiento.
- Proporcionar librerías de E/S amigables y eficientes.
- Comprender y sintonizar la E/S.

1.3 Tipos de Problemas que soluciona

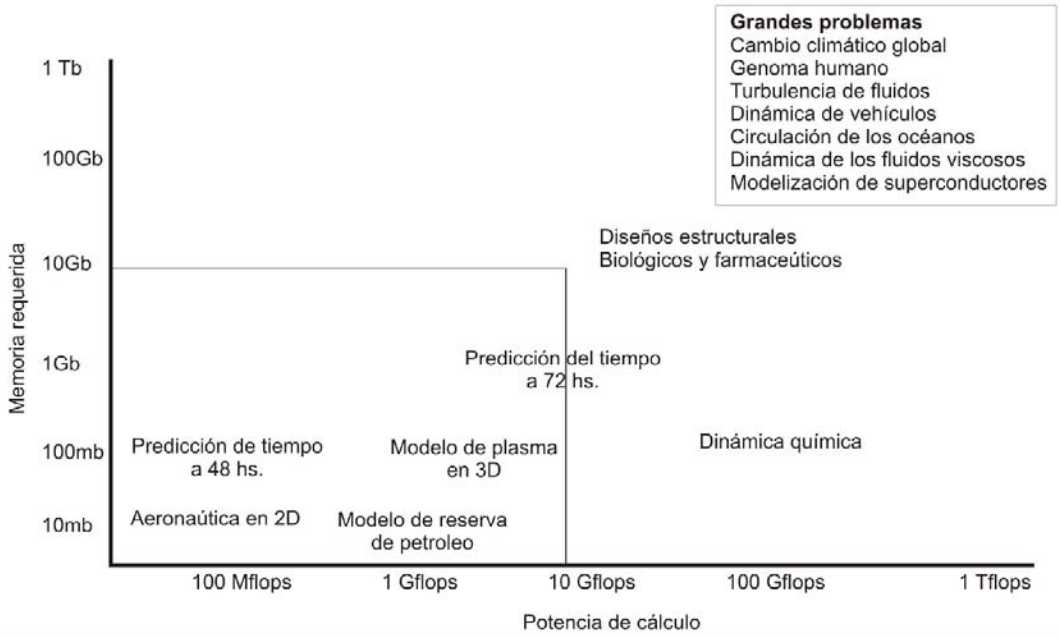


Ilustración 9 Problemas que Soluciona HPC. Recuperado de: https://www.fing.edu.uy

Los sistemas HPC su enfoque principal es de brindar rendimiento y disponibilidad. Deben ser capaces de satisfacer las grandes y crecientes demandas de procesamiento; para poder lograr estos objetivos se deben tener en cuenta los siguientes principios:

La **confiabilidad**, en donde se evalúa la robustez y la capacidad de auto-administración, desde bajo nivel (Chip) hasta el más abstracto (Aplicaciones), en donde se deben proveer sistemas de alto rendimiento que permitan asegurar los altos niveles de calidad de servicio.

La **eficiencia**, se debe evaluar tiempos de ejecución y el uso de recursos al momento de explotar el procesamiento en paralelo masivo. (Nesmachnow, Iturriaga, & Rocchetti, 2017)

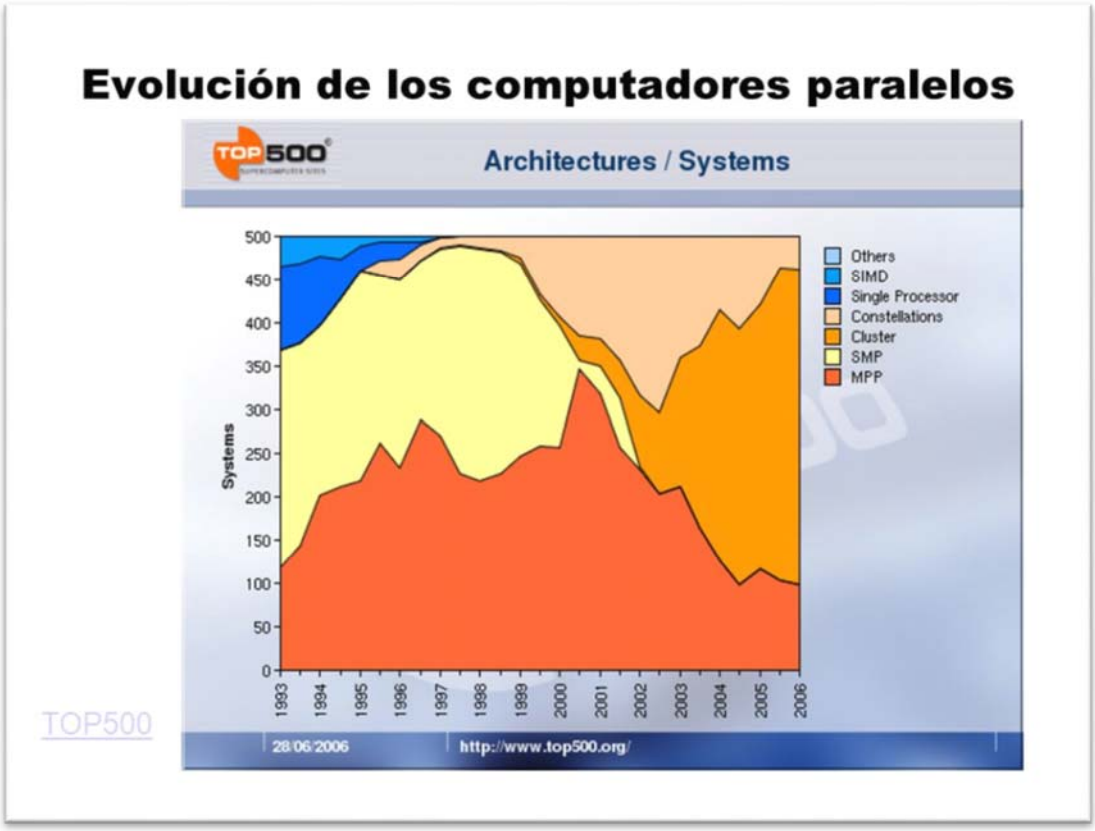


Ilustración 10 Los datos a través del tiempo, evolución de los computadores paralelos. Recuperado de: (top500, 2006).

Al analizar la ilustración N°10, podemos comprobar la evolución de los computadores paralelos a través del tiempo. En el año (2006), se generaron aproximadamente 450 sistemas y se pronostica que para el 2020 este número aumentará significativamente debido a la gran cantidad de datos a analizar y procesar.

Como podemos apreciar en la imagen No.10, podemos observar una estructura básica de un sistema de alto nivel computacional, no quiere decir que no se pueda construir uno con más bajos requerimientos, es posible la construcción de un sistema con virtualBox, el cual nos permite gestionar memoria RAM, espacio y los cores de procesamientos, es un muy buen simulador de entornos, motivo por el cual, es el tiempo de incursionar en la investigación, el estudio, y prototipos¸ desarrollo de metodologías, y herramientas que no ayuden y permitan cumplir con este nuevo reto.

Entre las razones principales que nos motiva a incursionar en estos ambientes, se encuentra la necesidad de aprender a manejar estas herramientas y tecnologías que no son fuertes en Colombia, pero se requieren y son necesarias para el desarrollo de nuestra Nación.

Capítulo 2

2 Diseño y Montaje Estructura

2.1 Diseño

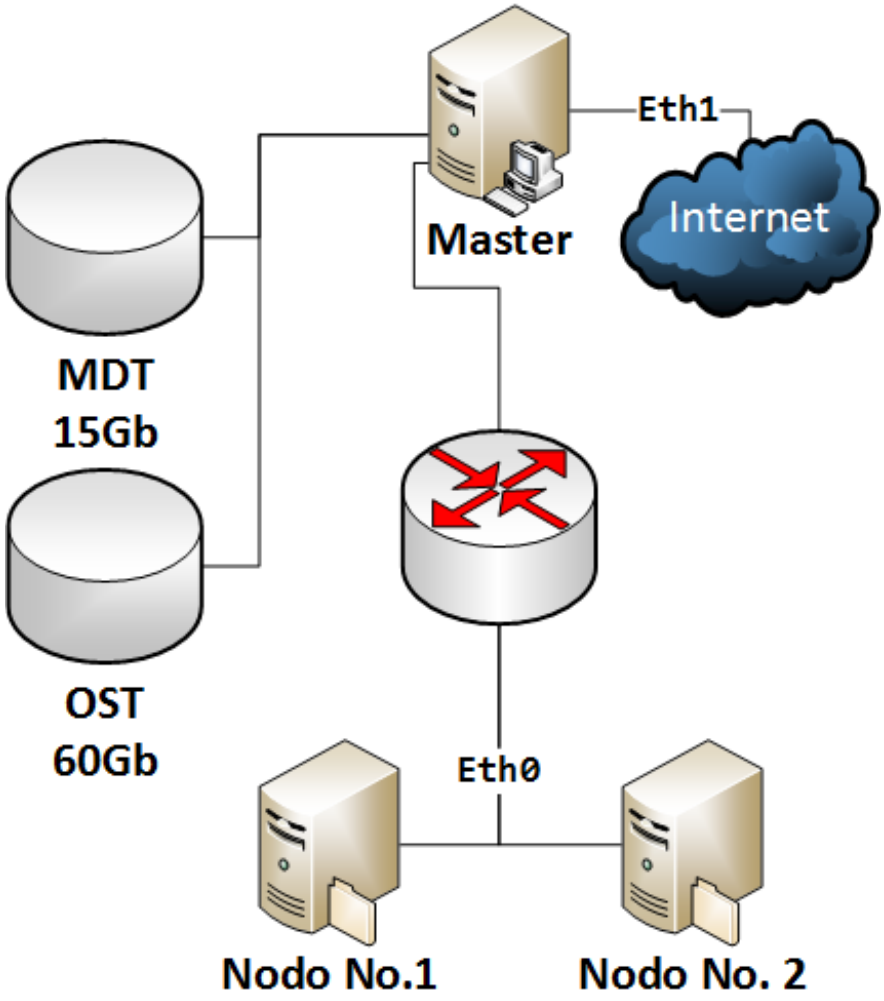


Ilustración 11 Diseño y estructura en Virtual Box. Recuperado de: el Autor

.

Analizando la imagen 12 podemos visualizar la forma en la cual se debe crear el primer modelo y configuración del clúster de altas prestaciones, en el indicamos donde va instalado un sistema de archivos paralelos como lustre y que distribución debemos instalas en cada equipo, adicional a este configuramos una red interna y una red con salida a internet; adicional a esto configuramos un arreglo de discos que van a almacenar la información resultante. Este proceso se explicará más detalladamente línea abajo.

Ya que los datos son analizados y almacenados en sistemas computacionales. Debemos analizar dos problemas principales:

El almacenamiento: Para poder guardar grandes volúmenes de datos, se debe recurrir a bases de datos distribuidas, con esto mejoramos el espacio que se requiere para almacenar dichos datos

La velocidad: Como los datos deben ser procesados¸ analizados, depurados, etc. La velocidad debe ser muy eficiente, por lo tanto, se debe recurrir a sistemas distribuidos para poder dar solución a esta situación, adicional a esto la información manejada es en tiempo real y como característica más relevante es el acceso a la información de forma inmediata.

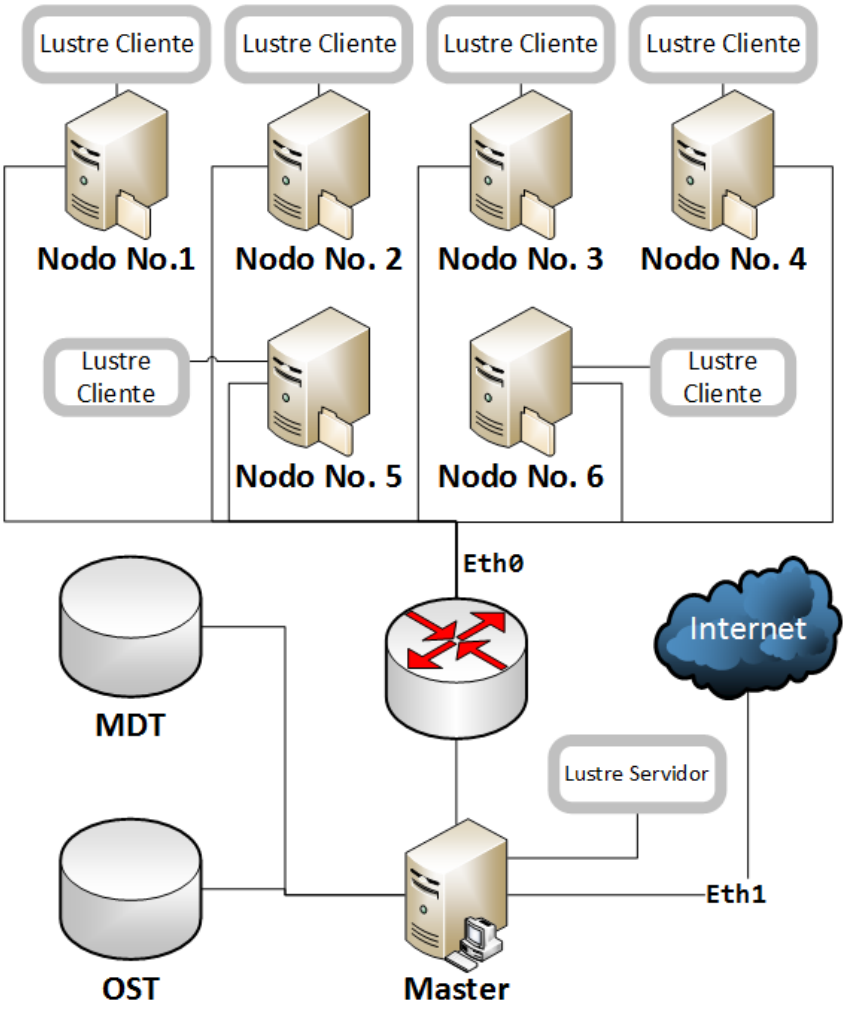


Ilustración 12 Diseño y estructura física Institución Universitaria Politécnico Grancolombiano. Recuperado de: el Autor

2.2 Planteamiento del problema

El principal objetivo de nuestro proyecto es poder trabajar en un ámbito real que nos sirva para el desempeño de nuestra carrera universitaria en la vida laboral, la clusterización, manejo de servidores, paralelismo, uso de nodos y demás. Se están convirtiendo en los puntos más fuertes de los sistemas ya que con la llegada de los teléfonos inteligentes, computadoras más avanzadas y otros aparatos tecnológicos, servidores de alto rendimiento para consulta de documentos como lo son todos aquellos que generan estas nuevas tecnologías.

Se está demostrando y encontrando que es necesario el uso y acceso en tiempo real a grandes volúmenes de información. Una dificultad que presenta el aprender y brindar mantenimiento a este tipo de plataformas, son sus grandes costos ya que no es fácil adquirirlos para una persona natural, pero al tener la oportunidad de aprender en esta etapa de nuestras vidas y directamente con el apoyo de nuestro docente queremos asegurar la gran experiencia que vamos a obtener en este ámbito y que de la seguridad a las personas de nuestras compañías que se cuenta con el manejo de estas herramientas de forma óptima.

2.3 Objetivos

2.3.1 Objetivo general

Realizar por medio de validación experimental un prototipo para la medición de la eficacia de entrada y salida en un clúster HPCC (High-Performance Computing Clúster) sobre OrangeFs.

2.3.2 Objetivos específicos

- Revisión de literatura científica y construcción del estado del arte en el documento maestro.
- Montaje y ensamble de un clúster de alto rendimiento incluyendo la distribución de arreglos de discos externos.
- Instalación de software para la medición edición de entradas y salidas en paquetes de datos y construcción de curvas con base en el análisis apoyado por el director.
- Construcción de un artículo científico y envío a una revista de talla internacional.
- Elaboración del documento final del proyecto.

2.4 Delimitación

2.4.1 Tiempo

Fase	Fecbrero	Marzo	Abril	Mayo
Consulta Bibliográfica	✓	✓	✓	
Elaboración del artículo y del documento	✓	✓	✓	✓
Desarrollo y diseño del modelo	✓	✓	✓	
Test			✓	✓
Preparación y diseño de Sustentación				✓

Tabla 2 Cronograma ejecución del desarrollo de la tesis. Fuente: los autores.

2.4.2 Alcance

Este documento enfocado en el diseño, montaje e implementación de un sistema de alto computo en paralelo, del cual se puede beneficiar la universidad en varios campos de aplicación, donde se podrá realizar una gran cantidad de operaciones complejas de computo, dando cumplimiento al cronograma mencionado en la Tabla No. 2.

a. PRIMERA ENTREGA
   - Documentación y diseño de la estructura y componentes que serán utilizados.
   - Montaje de la infraestructura técnica con apoyo del director del proyecto e instalación del software básico.

b. SEGUNDA ENTREGA
   - Documentación para el manejo de las herramientas a utilizar.
   - Pruebas con diferentes esquemas o diseños de los componentes

c. TERCERA ENTREGA
   - Evaluación y análisis de resultados, construcción de artículo de investigación y envío a revista científica
   - Documentación final para revisan y aprobación
   - Resumen del montaje.

Capítulo 3

3 Marco teórico

3.1 ¿Qué es Lustre?

Un sistema de archivos de clúster seguro, escalable, altamente disponible y robusto. El objetivo este sistema de ficheros distribuido es que puedan servir a clúster con más de 10000 nodos, proporcionar petabytes de almacenamiento, y la de mover 100 GB/seg con seguridad y gestión de la infraestructura. Lustre es utilizado en siete de los diez grandes supercomputadores del mundo.

Lustre es un sistema de archivos Linux implementado enteramente en el kernel. Su arquitectura se basa en el almacenamiento basado en objetos distribuidos; Se considera a cada archivo almacenado en el sistema de archivos Lustre un objeto. Esto delega la administración de almacenamiento en bloque a sus servidores back-end y elimina problemas significativos de escalabilidad y rendimiento asociados con la gestión coherente de metadatos de almacenamiento de bloques distribuidos.

En Lustre existen en dos tipos de objetos de datos, que son arreglos de bytes simples típicamente usados para almacenar los datos de los archivos POSIX (Portable Operating System Interface X de UNIX), y los objetos de índice, que almacenan valores clave típicamente usados para implementar directorios POSIX llamados metadatos.

Estos objetos son implementados por el dispositivo de almacenamiento de objetos Lustre (OSD Objet Storage Device), una abstracción que permite el uso de diferentes sistemas de archivos back-end, incluyendo ext4 y ZFS (Zettabyte File System). Una única instancia de OSD corresponde a un único volumen de almacenamiento de fondo y se denomina objetivo de almacenamiento. El objeto de almacenamiento, depende del sistema de archivos subyacente para resiliencia al fallo del dispositivo de almacenamiento, pero puede instanciarse en cualquier servidor que se pueda conectar a este almacenamiento para proporcionar alta disponibilidad en caso de fallo del servidor o del controlador.

Los destinos de almacenamiento se exportan como objetivos de metadatos (MDT), se utilizan para las operaciones de espacio de nombres del sistema de archivos que utilizan los objetos (OST) para almacenar datos de archivo. Por lo general, éstos son exportados por servidores configurados específicamente para sus respectivos metadatos o cargas de trabajo de datos, por ejemplo, hardware de almacenamiento RAID 10 y conteos altos de núcleos para servidores de metadatos (MDS) y hardware de almacenamiento RAID6 de alta capacidad y menores cuentas con núcleo para servidores de almacenamiento de objetos. Históricamente, los clústeres Lustre han consistido en un par de nodos MDS configurados para la conmutación por error pasiva/activa y varios OSS configurados para la conmutación por error activa-activa.

Las versiones más recientes de Lustre soportan múltiples MDTs en el mismo sistema de archivos y, por lo tanto, se espera que múltiples nodos MDS con activación activa se vuelvan más comunes.

Los clientes y servidor de Lustre se comunican entre sí utilizando una pila de comunicaciones en capas. Las redes físicas y/o lógicas subyacentes tales como Infiniband o TCP/IP son abstraídas por la capa Lustre Networking, LNet.

LNet proporciona tanto el paso de mensajes como el acceso a la memoria remota (RMA) para un movimiento de datos a gran escala. La capa RPC de Lustre (Ptlrpc) se construye encima de esto para proporcionar robustez cliente-servidor de comunicaciones ante la pérdida de mensajes y fallas de servidor.

El Lustre Distributed Lock Manager (LDLM) es un servicio proporcionado por objetos de almacenamiento además de los servicios de almacenamiento de objetos. (Koziol & Prabhat, 2015)

3.1.1.1 *Componentes de Lustre*

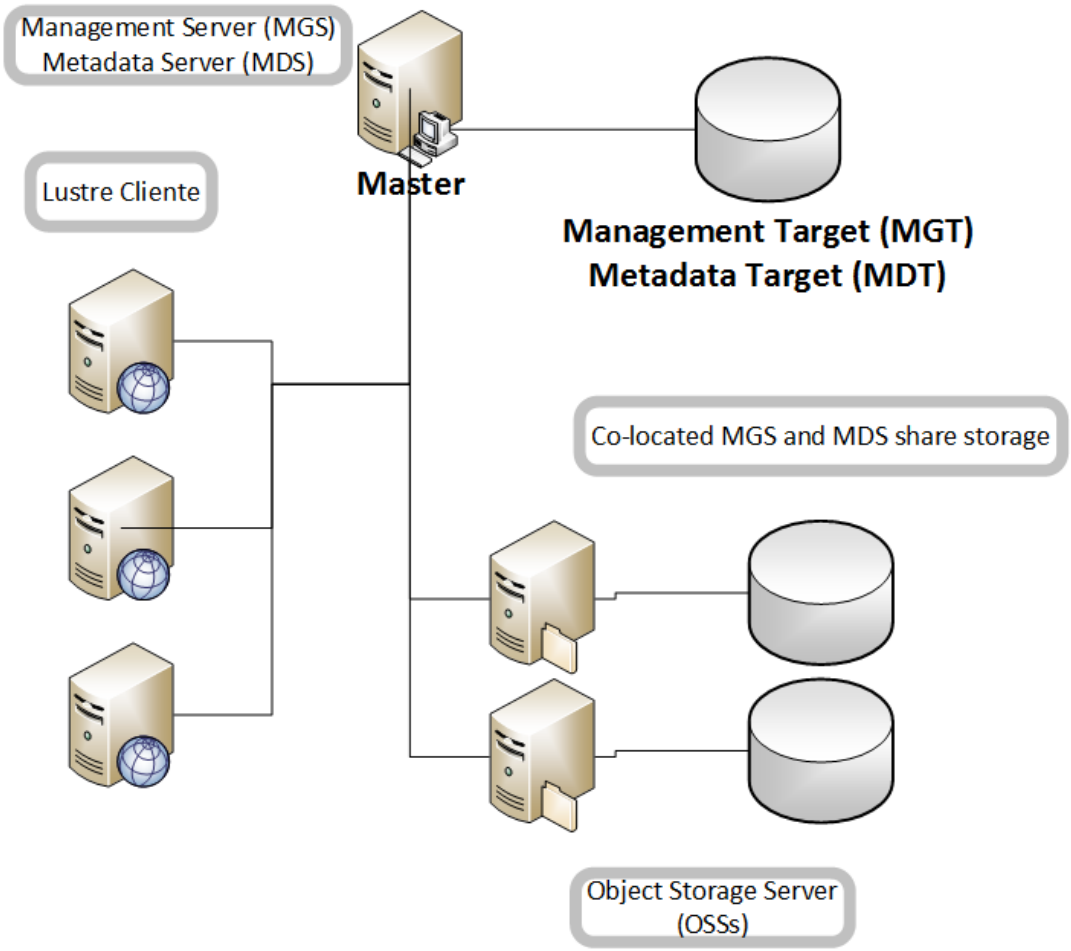


Ilustración 13 Esquema componentes de Lustre. Adaptada de: http://www.systerminal.com

Metadata Server (MDS) – El MDS tiene los metadatos almacenados para los clientes de Lustre. Cada MDS gestiona los nombres y directorios en el sistema de ficheros de Lustre proporcionando la solicitud de red para la manipulación de uno o más MDTs.

Metadata Target (MDT) – El MDT guarda los metadatos como: permisos y la disposición de los archivos, directorios y los nombres de archivo, en algún dispositivo de almacenamiento que está conectado a al MDS.

Object Storage Servers (OSS) – EL OSS ofrece un servicio de E / S de archivos y además de gestionar la red encargada a manejar las peticiones de uno o más OST.

Object Storage Target (OST) – Es el lugar donde se almacenan los datos del fichero del usuario, estos ficheros se graban en unos o más objetos y cada objeto se separa en un OST.

Cliente Lustre – Se ejecuta en el cliente y pueden montar el sistema de fichero Lustre. El software de cliente Lustre proporciona una interfaz entre el sistema de archivos virtual de Linux y los servidores de Lustre. El software de cliente incluye un Management Client (MGC), un Metadata Client (MDC), y varios Object Storage Clients (OSCs), uno correspondiente a cada OST en el sistema de archivos. Un volumen objeto lógico (LOV) agrega las OSCs para proporcionar acceso transparente en todos los OST. Así el cliente no se entera de cómo está organizado.

3.1.1.2 *Funcionamiento de Lustre*

Creo que es interesante conocer cómo funciona Lustre. En Lustre, un archivo almacenado en el MDT apunta a uno o más objetos asociado con un archivo de datos. Cada objeto contiene datos y está almacenado en un OST. Si el archivo MDT apunta a un objeto, todos los datos del archivo se almacenan en ese objeto. Si el archivo apunta a más de un objeto, los datos del archivo son "striped" (partido) a través de objetos (utilizando RAID 0) y cada objeto se almacena en un OST diferente.

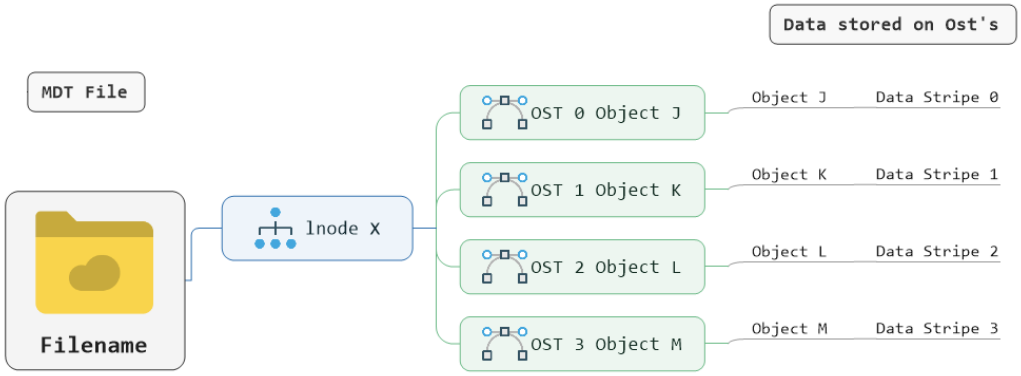


Ilustración 14 Funcionamiento de Lustre. Adaptada de: http://www.systerminal.com

Cuando un cliente abre un archivo, la operación transfiere la estructura del archivo del MDS al cliente. El cliente utiliza esta información para realizar E/S en el archivo e interactúa directamente con los nodos de OSS donde se almacenan los objetos. Cada archivo en el MDT contiene la disposición del archivo de datos asociado, incluyendo el número OST y el identificador de objeto. Los clientes solicitan la disposición de los archivos del MDS y a continuación, realizan operaciones de archivo de E/S mediante la comunicación directamente con el OSS que gestionan los datos del archivo. (Ruiz Navarro, 2014)

3.2 ¿Qué es Ganglia?

Ganglia es un software de monitoreo Open Source, diseñado para escalar a miles de nodos, que comenzó en la UC Berkeley. Cada máquina ejecuta un daemon llamado gmond que recoge y envía las métricas (como velocidad del procesador, uso de memoria, etc.) que recoge desde el sistema operativo a un host especificado. El host que recibe todas las métricas puede mostrarlas y puede transmitir una forma condensada de ellas en una jerarquía. Este esquema jerárquico es lo que permite a Ganglia escalar tan bien. Gmond tiene muy poca sobrecarga que lo convierte en un gran pedazo de código que se ejecuta en todas las máquinas del clúster sin afectar el rendimiento del usuario.

Hay momentos en que toda esta recopilación de datos puede afectar el rendimiento del nodo. "Jitter" en la red (como esto se llama) es cuando muchos mensajes pequeños siguen llegando al mismo tiempo. Hemos descubierto que, al bloquear los relojes de los nodos, esto puede evitarse.

Ganglia ya viene distribuciones Linux de nivel empresarial como Red Hat Enterprise Level (RHEL) y en CentOS. Ganglia surgió de los requisitos para los sistemas de monitoreo de Berkeley (Universidad de California), pero ahora es de uso por organizaciones comerciales y educativas como Cray, MIT, NASA y Twitter.

Se basa en un diseño jerárquico de monitoreo de clúster. Se basa en un protocolo multicast para supervisar el estado dentro de los clústeres y utiliza un árbol de conexiones punto a punto entre los nodos de clúster representativos para monitorear clústeres y agregar su estado. Utiliza las tecnologías más utilizadas, como, XML con la cual realiza la representación de datos, XDR para el transporte de datos de manera portátil y compactos y RRDtool con la cual realiza el almacenamiento y visualización de datos. Su construcción y configuración es compleja, ha sido la portada en un gran número de sistemas operativos y arquitecturas de procesadores adicionada, y actualmente está en uso en más de 500 clúster alrededor del mundo. Se ha utilizado para vincular clúster a través de los campus universitarios y en todo el mundo y puede escalar para manejar clúster con 2000 nodos.

RRDTool es una herramienta de base de datos Round Robin. Fue creado por Tobías Oetiker y proporciona un motor para muchas herramientas de monitoreo de alto rendimiento. Ganglia es uno de ellos, pero Cacti y Zenoss son otros.

Para instalar Ganglia, primero tenemos que tener RRDTool funcionando en nuestro servidor de monitoreo. RRDTool proporciona dos funciones muy interesantes que son apalancadas por otros programas:

- Almacena datos en una base de datos Round Robin. A medida que los datos capturados son muy viejos, la resolución se vuelve menos refinada.
- Puede crear gráficos utilizando argumentos de línea de comandos para generarlos a partir de los datos que ha capturado. (Nicholes, 2008)

3.2.1.1 *Arquitectura de Ganglia*

Ganglia está compuesto por dos daemons y un front-end basado en PHP y algos otros pequeños programas de utilidad

- **Ganglia Monitoring Daemon (gmond):** es un daemon de múltiples hilos que se ejecuta en cada nodo de clúster que desea supervisar. La instalación no requiere tener un sistema de archivos NFS común o un back-end de bases de datos, instalar cuentas especiales o mantener archivos de configuración.

  Gmond tiene cuatro principales responsabilidades:

  - Supervisar los cambios generados en el estado del host.
  - Anunciar cambios más importantes.
  - Escuchar el estado de todos nodos Ganglia a través de un canal unicast o multicast.
  - Responder a las solicitudes en un formato XML del estado del clúster.

  Cada gmond transmite en información de dos maneras diferentes:

  - Unicasting o Multicasting del estado de host en formato de modelamiento de datos externos (XDR) utilizando mensajes UDP.
  - Envío de XML a través de una conexión TCP.

- **Ganglia Meta Daemon (gmetad):** El monitoreo en Ganglia se logra utilizando un árbol de conexiones punto a punto entre los nodos de clúster representativos para agregar el estado de múltiples clústers. En cada nodo del árbol, un Ganglia Meta Daemon (gmetad) periódicamente sondea una colección de fuentes de datos secundarias, analiza el XML recogido, guarda todas las métricas numéricas y volátiles en las bases de datos Round Robin y exporta el XML agregado a través de un socket TCP a los clientes. Las fuentes de datos pueden ser daemons gmond, que representan grupos específicos, u otros daemons gmetad, que representan conjuntos de clústers. Las fuentes de datos utilizan direcciones IP de origen para el control de acceso y se pueden especificar utilizando varias direcciones IP para conmutación por error.

Esta última capacidad es natural para agregar datos de los clústeres ya que cada daemon de gmond contiene todo el estado de su clúster.

- **Ganglia PHP Web Front-end:** El front-end de Ganglia ofrece una visión de la información recopilada a través de páginas web en tiempo real. Ganglia muestra los datos de una manera representativa y visual para los administradores. Aunque el front-end de Ganglia comenzó como en sus inicios como una simple página HTML, y con el tiempo ha evolucionado en un sistema dinámico para que el front-end tenga un buen " look and feel" para que se pueda se pueda personalizar fácilmente, presenta una visión general de todos los nodos dentro de una cuadrícula vs todos los nodos de un clúster, brinda la posibilidad de profundizar en nodos individuales y nos presenta tanto vistas textuales como gráficas que mantiene un colorido historial de todos los datos recogidos. (Ganglia Monitoring, 2014)

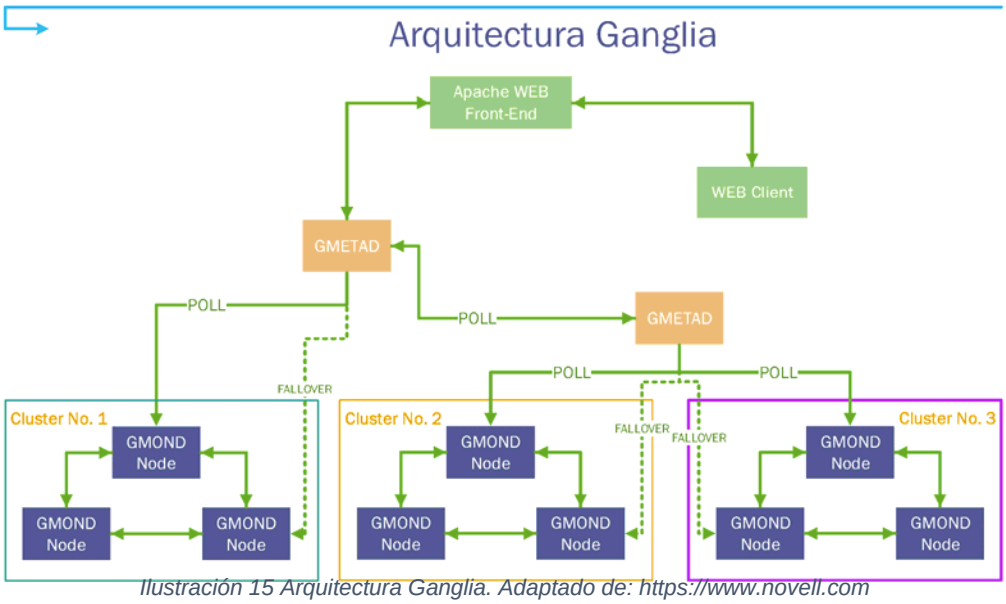


Ilustración 15 Arquitectura Ganglia. Adaptado de: https://www.novell.com

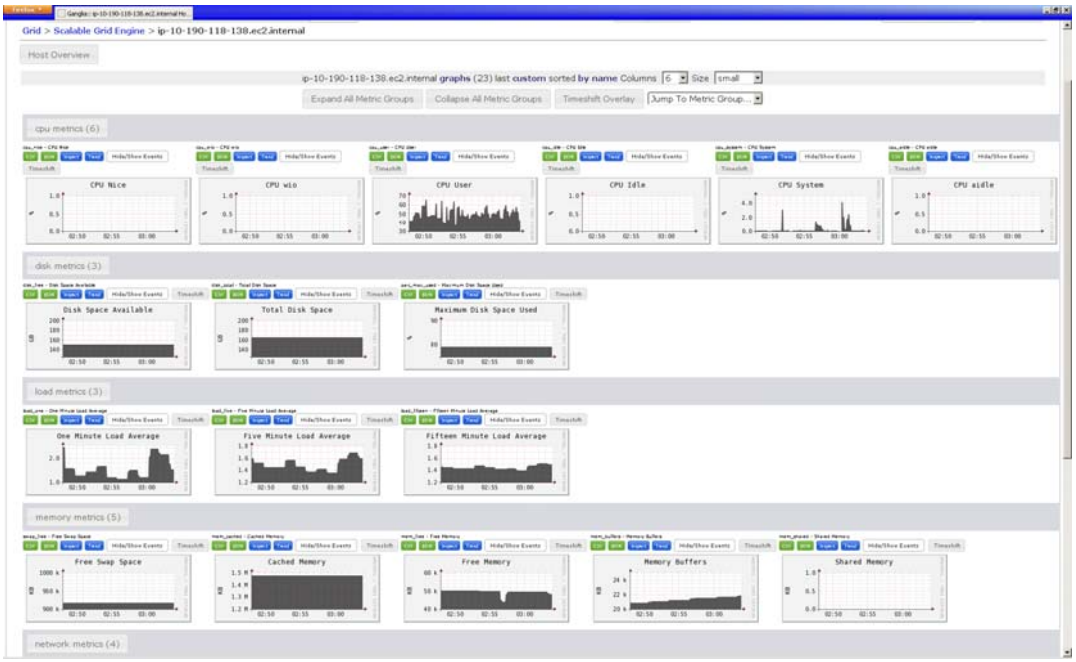


Ilustración 16 Ganglia corriendo en un clúster Scalable Grid Engine HPC en Amazon EC2. Recuperado de: https://en.wikipedia.org

3.3 ¿Qué es Nmon?

Nigel's Monitor (Nmon) Software de monitoreo que se desarrolló para AIX, y posterior a ello se adaptado para que pueda usarlo en Linux, el cual se convierte en un software fundamental para el administrador.

Permite mostrar en pantalla diversos indicadores que muestran el estado del sistema, en tiempo real o guardando los datos en archivos para su posterior revisión en modo de gráficos.

3.3.1.1 *Características de Nmon*

Nmon supervisa los datos de rendimiento del sistema, incluyendo:

- Uso actual de la CPU
- Paginado de la memoria
- Estadísticas de la cola de ejecución de procesos y del kernel
- escritura de las interfaces de red, transferencias, I/O, y tasas de lectura
- escritura de los discos, transferencias, I/O, y tasas de lectura
- Uso de memoria
- Espacio libre en archivos del sistema
- Network File System (NFS)

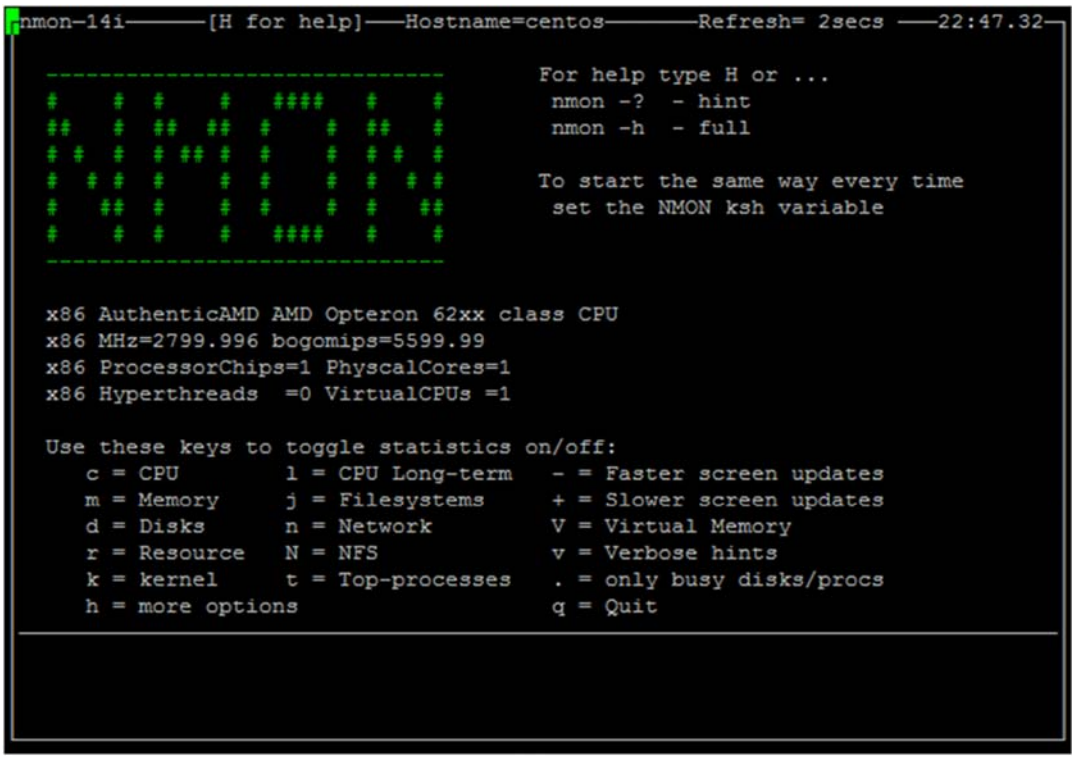


Ilustración 17 Inicio Nmon. recuperado de: https://devops.profitbricks.com/

Los comandos para indicar que deseamos monitorear son:

- ✓ **c**: (**CPU**) Muestra los detalles de la CPU.
- ✓ **m:** **(Memory)** Muestra el estado de la memoria, tanto física como del área de intercambio.
- ✓ **d**: (**Disk**) brinda información de los discos que se encuentran uso: espacio libre, tamaño, tipo de disco.
- ✓ **r**: (**Resources**) Recursos del sistema en donde se incluye la velocidad de la CPU.
- ✓ **k:** **(Kernel)** Muestra las estadísticas del Kernel y otros datos.
- ✓ **h:** **(more options)** Más opciones.
- ✓ **l**: (**CPU Long-**term) muestra un histórico del performance del CPU en %.
- ✓ **j**: (**Filesystems**) permite ver detalladamente de los sistemas de archivos que se encuentran.
- ✓ **n**: (**Network**) permite ver la información de las cada una de la interface de red.
- ✓ **t:** **(Top-processes)** Lista de los procesos con que tiene un alto uso de los recursos.

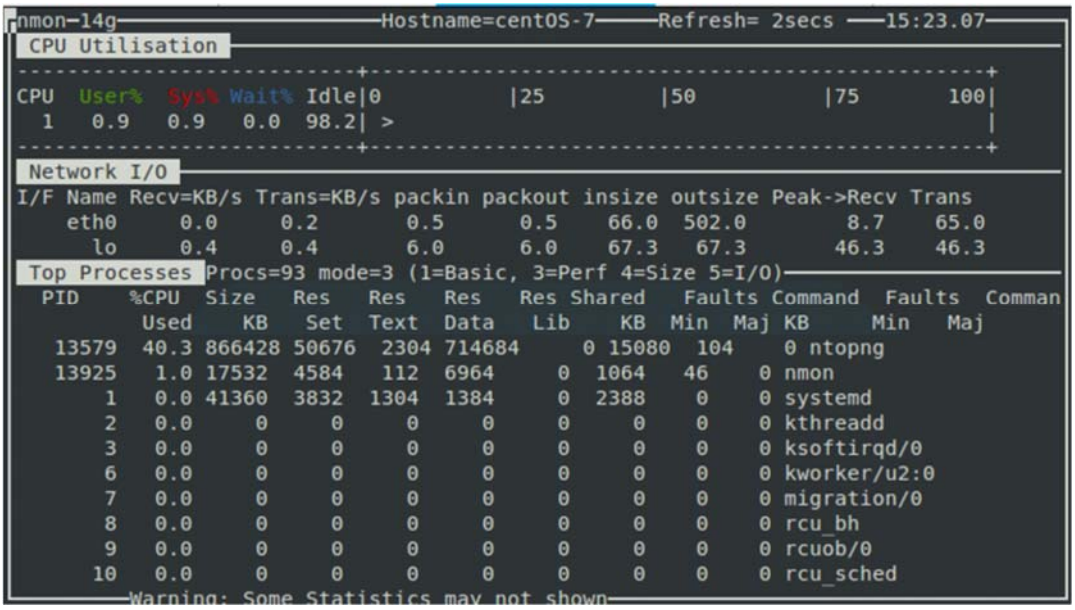


*Ilustración 18 Comprobar las estadísticas de la red pulsando **n**. Recuperado de: https://devops.profitbricks.com*

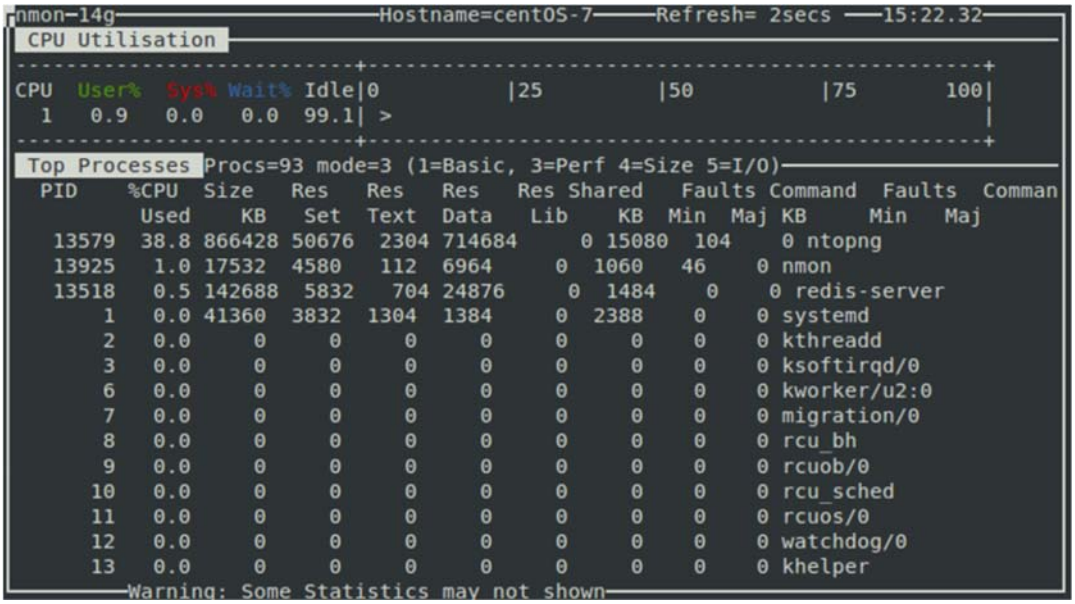


*Ilustración 19 Comprueba la CPU por procesador pulsando **c**. Recuperado de: https://devops.profitbricks.com/*

3.4 ¿Qué es IOR?

IOR está diseñado para medir el rendimiento de E/S del sistema de archivos paralelo en el nivel POSIX y MPI-IO. "IOR" significa "Interleaved Or Random". Este programa paralelo realiza escrituras y lee desde/hacia archivos bajo varios conjuntos de condiciones e informa de las tasas de rendimiento resultantes.

A continuación, se describirá el diseño del IOR y sus parámetros, en la siguiente imagen se mostrará la relación entre la estructura de archivos y los procesadores al escribir en un archivo compartido.

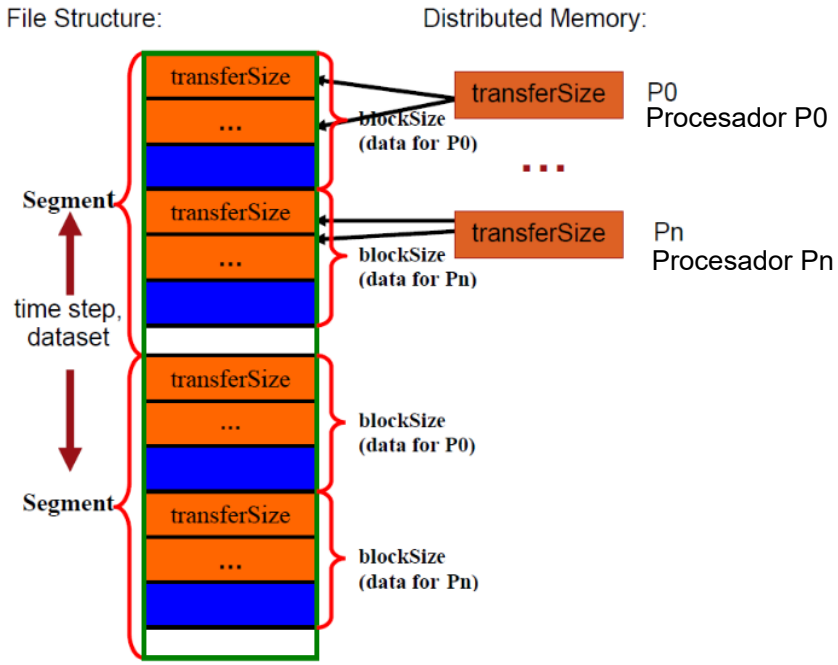


Ilustración 20 El diseño del benchmark IOR para el tipo de archivo compartido. Los bloques se almacenan en archivos separados para el modo de operación de 1 archivo por procesador. Recuperado de: https://cug.org

Se organiza como una secuencia de "segmentos" que representan los datos de una aplicación para un paso de tiempo simulado de una única variable de datos o una secuencia de variables de datos (p. Presión, temperatura, velocidad). Cada segmento se divide uniformemente entre los procesadores que comparten este archivo de datos en unidades denominadas "bloques" para representar el ensamblaje de arreglos realizado por la capa de E/S paralela. El proceso con rango 0 obtiene el primer bloque y el proceso con rango 1 obtiene el segundo bloque y así sucesivamente. (Shan, 2007)

3.5 ¿Qué es OrangeFs?

OrangeFS es un sistema de almacenamiento paralelo, originalmente llamado PVFS y desarrollado en 1993 como parte de una beca de la NASA para estudiar los patrones de E/S de programas paralelos. Es ideal para solucionar los problemas de gran almacenamiento que se encuentran en entornos de HPC, BigData, Streaming de Video, Genómica, Bioinformática. OrangeFS puede ser accedido a través de utilidades del sistema, librerías, MPI-IO y puede ser utilizado por el ecosistema Hadoop como alternativa al sistema de archivos HDFS.

Las aplicaciones a menudo no requieren que OrangeFS esté montado en el VFS, pero el cliente del kernel de OrangeFS permite montarlos. El cliente del kernel se comunica con un demonio en espacio de usuario que a su vez se comunica con el demonio-servidor de OrangeFS que implementa el sistema de archivos. El demonio-servidor no tiene por qué estar ejecutándose en el mismo sistema que el cliente del kernel. Los sistemas de archivo OrangeFS también pueden montarse con FUSE.

Los objetivos de OrangeFs son:

- Ejecutar a nivel de usuario
- Permitir el acceso paralelo a datos y metadatos
- Minimizar los cuellos de botella que limitan el paralelismo
- Proporcionar un cómputo diverso de interfaces de cliente

OrangeFS se puede ejecutar completamente a nivel de usuario, o un pequeño módulo de núcleo puede ser utilizado para montarlo como cualquier otro sistema de Linux, Windows y OSX para el uso de programas y utilidades estándar. Los datos y los metadatos están distribuidos de forma transparente en varios servidores y pueden ser accedidos en paralelo por varios clientes. OrangeFS relaja la semántica de consistencia POSIX cuando es necesario para eliminar cuellos de botella mientras se utiliza un espacio de nombres globales POSIX. Las interfaces de cliente incluyen el montaje a través del núcleo del sistema operativo, MPI-IO, cliente Hadoop JNI y soporte WebDav.

3.5.1.1 *PVFS*

Sistema de archivos paralelos Open Source para sistemas operativos Linux; la cual permite que las aplicaciones en paralelo y serial almacenen y recuperen datos desde un conjunto de servidores o nodos de I/O unidos en una misma red. Para implementar el alto rendimiento en PVFS2 se necesita que el almacenamiento esté compartido en una red de almacenamiento local (Storage Área Network). En 1993 se desarrolló PVFS1 para la máquina virtual paralela (PVM) como parte de una concesión de la NASA. En 1994, el software fue modificado en su estructura para poder utilizar TCP / IP, dirigido a un grupo de equipos digitales Corp.

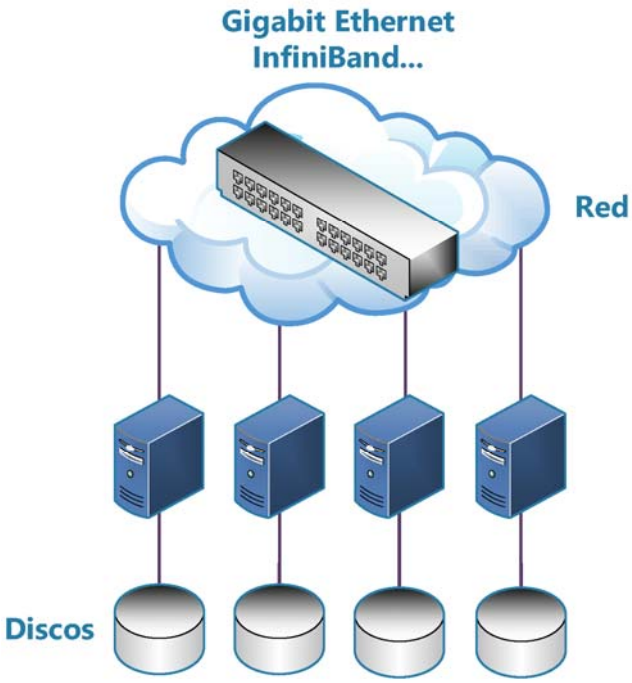


Ilustración 21 Configuración SAN. Recuperado de: https://www.uaeh.edu.mx

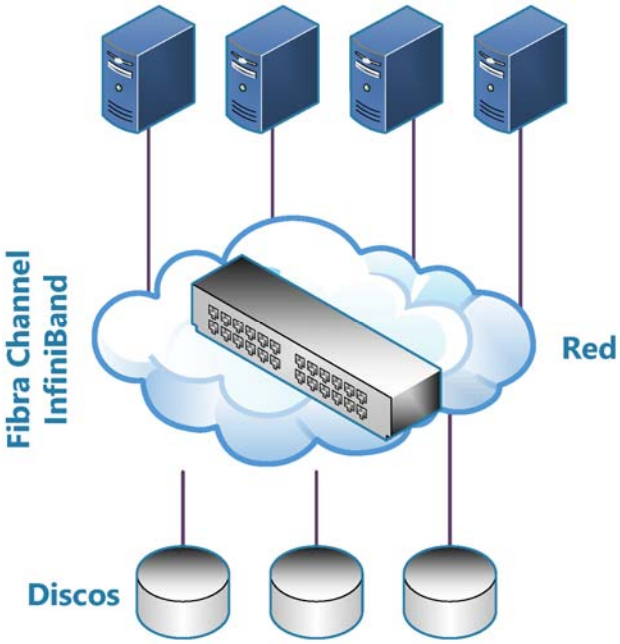


Ilustración 22 Beneficio del almacenamiento de cada nodo del clúster como un sistema virtual de archivos. Recuperado de: https://www.uaeh.edu.mx

3.5.1.2 *PVSF2*

El desarrollo de PVFS2 comenzó en 1999 y fue completado por un equipo de Clemson University, ANL y Ohio Supercomputer Center en 2003. El nuevo diseño ofreció servidores de objetos, metadatos distribuidos, vistas basadas en MPI,

Soporte para múltiples tipos de red y una arquitectura de software diseñada para facilitar la experimentación y la extensibilidad.

3.5.1.3 *ARQUITECTURA DE ARCHIVOS*

El sistema de archivo en paralelo se logró desarrollar a partir de su inicial versión, lo cual mejoro la flexibilidad entre módulos, su modularidad y permito una gran integración con MPI-IO.

3.5.1.4 *ESTRUCTURA HARDWARE*

Los elementos que componen la estructura PVFS2 son:

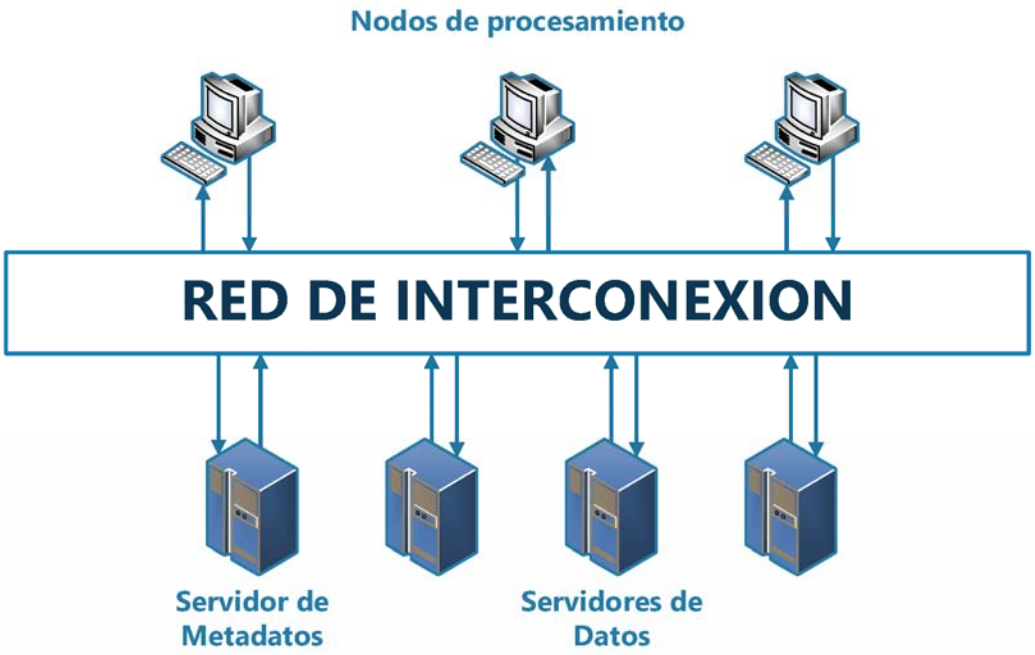


Ilustración 23 Arquitectura de PVFS2. Recuperado de: https://www.uaeh.edu.mx

1. Metaservers (Servidores de metadatos). Se encargan gestionar la información de los metadatos de los archivos. Además, se guardan la distribución, número de datahandles y los atributos.

   Cuando un cliente debe realizar una operación en un archivo particular, se debe comunicar con este servidor para y así obtiene la información que necesita para

poder realizar esta operación. Los metadatos son almacenados por lo general en dos bases de datos: La primera almacena los metadatos, y la segunda almacena los datahandles.

2. Servidores de datos (dataservers). Son denominados de E/S y cuentan con la función de guardar una parte de los archivos de datos del sistema. Se encargan de guardar las rutas de acceso a estos archivos. Cuando se realiza el cálculo del espacio que se encuentra disponible en el sistema de archivos, debe de realizar una suma del espacio que está disponible en cada uno de los servidores. Este cálculo se encarga de recopilar el espacio disponible en todos los servidores, y posterior se realiza la multiplicación el menor de ellos por el total de los servidores que se encuentran disponibles.
3. Clientes: Estos clientes se encargan de acceder al sistema de archivos en sí. Se realiza por medio de una biblioteca que se llama pvfs2lib. También podemos ingresar al sistema a por medio de la biblioteca de MPI-IO.

3.5.1.5 *ARQUITECTURA DEL SOFTWARE*

La capa software de PVFS2 buscan una conexión de todas las capas con todas para no que no exista bloqueo entre ellas.

Capa de interfaz: es una interfaz de programación de aplicaciones que está disponible para los clientes, la cual usa una serie de máquinas de estado para realizar las operaciones que están siendo solicitadas por la capa superior. Estas máquinas de estado crear nuevas operaciones, ejecutarlas y controladas por una capa llamada JOB, y estas operaciones cuentan con identificador asociado.

Capa FLOW, se encarga de utilizar la capa llamada Buffered Message Interface (BMI), y así lograr brindar acceso a la red.

Las capas de software en el cliente son las mismas que las que están en el servidor, menos una capa llamada TROVE la cual solo la tienen los servidores, y se encarga de guardar los datos en los discos de almacenamiento (Tovar, y otros, 2009).

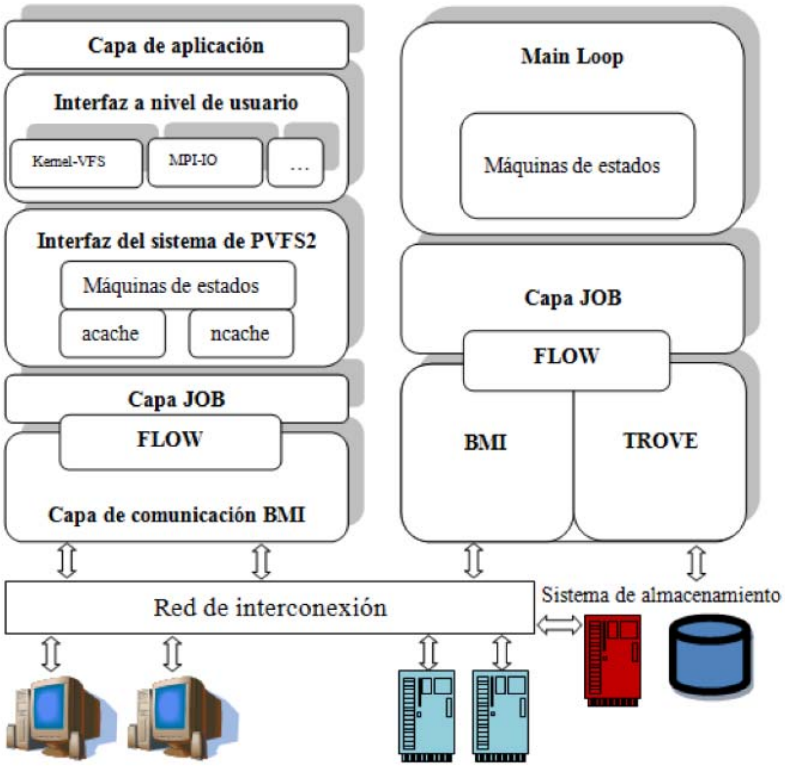


Ilustración 24 Arquitectura software de PVFS2. Recuperado de: https://www.uaeh.edu.mx

4 Montaje Físico

A continuación, se mostrará la instalación de un clúster de Alto Desempeño (HPC- High Performance Clúster) que se instaló en el Politécnico GranColombiano en la ciudad de Bogotá. El objetivo del clúster es tener una infraestructura capaz de correr aplicaciones de cómputo matemático intenso.

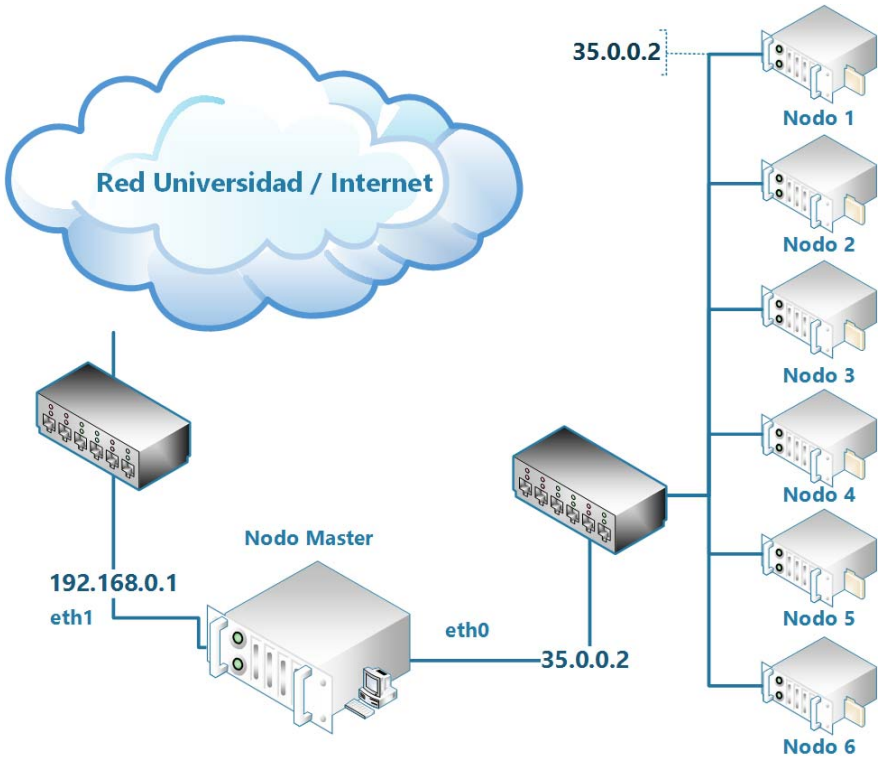


Ilustración 25 Diseño del Clúster

El sistema operativo que se instaló en el nodo máster y con el cual arrancan los nodos de cómputo es el siguiente: CentOS 6.5. Se escogió CentOS, por ser una distribución muy completa que tiene todas las aplicaciones y librerías necesarias para desarrollar, compilar y correr aplicaciones científicas, y está basado en las fuentes de RedHat Enterprise Linux Server (RHEL), lo cual lo hace compatible para las aplicaciones desarrolladas para esa plataforma. El software de clúster escogido fue Rocks Clúster 6.1 (originalmente llamado NPACI Rocks) es una distribución de Linux para clústers de computadores de alto rendimiento. Se escogió Rocks por ser una solución madura y ampliamente probada. Además, esta solución tiene licencia GPL y por lo tanto puede ser usada sin restricciones.

El sistema de archivos usado para el clúster es Lustre, el cual fue desarrollado por Sun Microsystems.

4.1 Visualización de los nodos desde el nodo Master

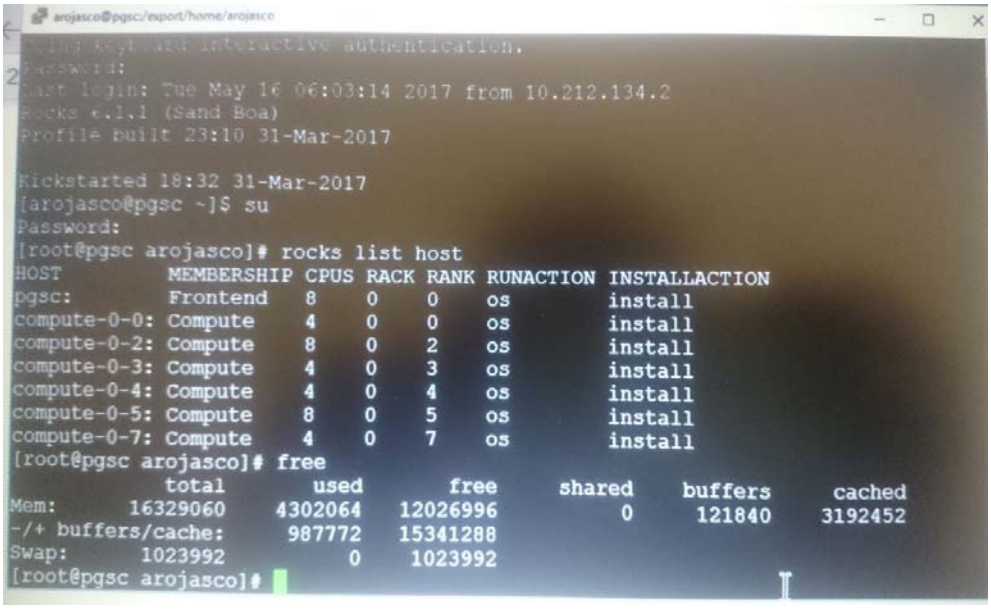


Ilustración 26 vista de los nodos desde el nodo Master

4.2 Shell para montaje Lustre

Se elabora archivo .sh para realizar el montaje automático de lustre en todos los nodos debido a que era muy dispendioso entrar por ssh a cada uno de los nodos y realizar el montaje, motivo por el cual se creó el siguiente archivo:

```
#Antes:
ssh compute-0-0 #Ingresamos a cada nodo por ssh
mount -t lustre 35.0.0.2@tcp:/lustre /lustre #Subimos Lustre por cada
nodo

#Ahora
#!/bin/bash
for i in {1..8}; do
      if [ $i = "1" ]; then
            echo no existe
      elif [ $i = "6" ]; then
            echo no existe
      else
          ssh compute-0-$i
          mount -t lustre 35.0.0.2@tcp:/lustre /lustre
          exit
      fi
done
```

5 Proyectos montados en el Clúster

- Tesis doctoral del Docente Alexis Rojas
- Subordinación de HC en Amazon de Jesús Jinete (Pregrado)
- Trabajo de Investigación de Montecarlo del Docente Diego Arévalo
- Darshan de John Ramírez (Maestría)
- HPC I/O rendimiento de Andrés Cortes (Maestría)
- Render con Cóndor y HPC de Leonardo Forero (Pregrado)
- Grafos en computación paralela para la educación Edwin Malagón (Maestría)
- Diseño y ensamblaje del Clúster de Carlos Andrés Charris y Diego Fernando Meza (Pregrado)

6 Resultados

6.1 Introducción

Después de realizar el montaje del clúster y el correspondiente software se procede a montar en máquinas virtuales una alternativa a Lustre que nos va a reducir la alta concurrencia y cuellos de botella que este genera al momento de la escritura. La solución propuesta fue realizar el montaje de OrangeFs el cual después de muchos inconvenientes se contó con la orientación directa del desarrollador del software permitiendo unos resultados satisfactorios, posterior a esto se procedió a realizar pruebas del benchmark IOR utilizando el sistema de archivos OrangeFs con las siguientes especificaciones:

Prueba Numero 1:

- Cantidad de archivos 1 (1 archivos por nodo).
- Número de repeticiones: 1
- Tamaño xfer: 1024 KB
- Tamaño de los bloques: 200 MB
- Tamaños Agregado al filesystem en la prueba: 200 MB

```
IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Sat Jun  3 22:45:02 2017
Command line used: /opt/ior/bin/ior -v -F -t 1k -b 200m -o
/opt/orangefs/testIOR.txt
Machine: Linux compute-0-0.local
Start time skew across all tasks: 0.00 sec

Test 0 started: Sat Jun  3 22:45:02 2017
Path: /opt/orangefs
FS: 15.3 GiB   Used FS: 23.2%   Inodes: 1.0 Mi   Used Inodes: 11.7%
Participating tasks: 1
Summary:
      api                = POSIX
      test filename      = /opt/orangefs/testIOR.txt
      access             = file-per-process
      pattern            = segmented (1 segment)
      ordering in a file = sequential offsets
      ordering inter file= no tasks offsets
      clients            = 1 (1 per node)
      repetitions        = 1
      xfersize           = 1024 bytes
      blocksize          = 200 MiB
      aggregate filesize = 200 MiB

access    bw(MiB/s)  block(KiB) xfer(KiB)  open(s)    wr/rd(s)
close(s)   total(s)   iter
------    ---------  ---------- ---------  --------   --------   -----
---   --------   ----
write     351.65     204800     1.00       0.000030   0.568495
0.000006   0.568751   0
```

```
read      1844.01    204800     1.00       0.000094   0.108267
0.000006   0.108459   0
remove    -          -          -          -          -          -
0.241451   0

Max Write: 351.65 MiB/sec (368.73 MB/sec)
Max Read:  1844.01 MiB/sec (1933.59 MB/sec)

Summary of all tests:
Operation   Max(MiB)   Min(MiB)  Mean(MiB)     StdDev    Mean(s) Test#
#Tasks tPN reps fPP reord reordoff reordrand seed segcnt blksiz xsize
aggsize API RefNum
write         351.65     351.65     351.65       0.00    0.56875 0 1 1
1 1 0 1 0 0 1 209715200 1024 209715200 POSIX 0
read         1844.01    1844.01    1844.01       0.00    0.10846 0 1 1
1 1 0 1 0 0 1 209715200 1024 209715200 POSIX 0

Finished: Sat Jun  3 22:45:03 2017
```

Prueba Numero 2:

- Cantidad de archivos 16 (8 archivos por nodo).
- Número de repeticiones: 1
- Tamaño xfer: 1024 KB
- Tamaño de los bloques: 10 MB
- Tamaños Agregado al filesystem en la prueba: 160 MB

```
IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Sun Jun  4 00:51:52 2017
Command line used: /opt/ior/bin/ior -v -F -t 1k -b 10m -o
/opt/orangefs/testIOR.txt
Machine: Linux compute-0-0.local
Start time skew across all tasks: 0.11 sec

Test 0 started: Sun Jun  4 00:51:52 2017
Path: /opt/orangefs
FS: 15.3 GiB   Used FS: 23.3%   Inodes: 1.0 Mi   Used Inodes: 11.7%
Participating tasks: 16
Summary:
      api                = POSIX
      test filename      = /opt/orangefs/testIOR.txt
      access             = file-per-process
      pattern            = segmented (1 segment)
      ordering in a file = sequential offsets
      ordering inter file= no tasks offsets
      clients            = 16 (8 per node)
      repetitions        = 1
      xfersize           = 1024 bytes
      blocksize          = 10 MiB
      aggregate filesize = 160 MiB

access    bw(MiB/s)  block(KiB) xfer(KiB)  open(s)    wr/rd(s)
close(s)   total(s)   iter
------    ---------  ---------- ---------  --------   --------   -----
---   --------   ----
write     754.96     10240      1.00       0.082023   0.211867
0.083135   0.211931   0
```

```
read      1158.18    10240      1.00       0.105412   0.138130
0.132494   0.138148   0
remove    -          -          -          -          -          -
0.117477   0

Max Write: 754.96 MiB/sec (791.64 MB/sec)
Max Read:  1158.18 MiB/sec (1214.44 MB/sec)

Summary of all tests:
Operation   Max(MiB)   Min(MiB)  Mean(MiB)     StdDev    Mean(s) Test#
#Tasks tPN reps fPP reord reordoff reordrand seed segcnt blksiz xsize
aggsize API RefNum
write         754.96     754.96     754.96       0.00    0.21193 0 16
8 1 1 0 1 0 0 1 10485760 1024 167772160 POSIX 0
read         1158.18    1158.18    1158.18       0.00    0.13815 0 16
8 1 1 0 1 0 0 1 10485760 1024 167772160 POSIX 0

Finished: Sun Jun  4 00:51:53 2017
```

7 Conclusiones

7.1 Introducción

En el siguiente capítulo se presentarán las conclusiones finales resultantes de este trabajo de investigación. Adicional a esto, se describirán líneas más abajo los futuros trabajos que pueden surgir a partir de la elaboración de este documento, las cuales pueden ser consideradas con el objetivo de continuar trabajando en aplicaciones paralelas.

7.2 Conclusiones Finales

Se logró el diseño y montaje de un clúster de alto rendimiento basados en un prototipo montado en máquinas virtuales, y posterior a esto fue montado sobre servidores físicos; se generó una infraestructura de clúster virtual oportunista, posterior a esto se logró una el montaje de una infraestructura propuesta que permitió la ejecución de diferentes aplicaciones utilizando una estrategia oportunista y sin ningún costo.

Se adecuo el clúster con el software necesario para lograr un alto desempeño en el procesamiento de altos volúmenes de datos lo que permitió apoyar e integrar proyectos de grado de otros compañeros.

Se logra obtener una gran reducción del tiempo que se requiere para obtener resultados y se logran monitorear el comportamiento de los nodos que se encuentran procesando esta información.

Se culminó una investigación paralela en búsqueda de mejorar y reducir los cuellos de botella que genera Lustre debido a la alta concurrencia que se tiene al momento de la escritura, para esto se investigó sobre OrangeFs con el cual se logró tener contacto y acompañamiento con los directos desarrolladores del software para realizar un montaje y realizar pruebas de desempeños entre lustre y OrangeFs.

Se obtuvo un alto conocimiento, manejo y administración sobre las grillas computacionales, sistemas paralelos, monitoreo de nodos, pruebas de desempeño para la toma de decisiones.

Se deja un legado para los próximos estudiantes que deseen sacar provecho de este tipo de infraestructura y aplicaciones paralelas basados en que estamos en una constante evolución y cada vez requeriremos mayor potencia de computo.

Se aprendió y conoció sobre los Sistemas de archivos, el Benchmark que no es más que una serie de técnicas que se utilizan para medir el rendimiento de componentes; se obtuvieron conocimientos sobre monitores y benchmark para medir los impactos I/O, por lo que se debió investigar el uso de algunos comandos y sus funciones con el fin de utilizarlos de forma correcta. También fue muy interesante ver como se comportaba el benchmark en el momento de escribir y leer en el sistema de archivos OrangeFs debido a que lo realiza de una forma paralela utilizando los nodos que agreguemos en el momento

de realizar la prueba, sobre todo ver como expande la cantidad de ficheros y el tamaño que consume en realizar las pruebas de forma paralela.

8 Trabajos Futuros

A continuación, se describirán líneas más abajo los futuros trabajos surgidas a partir de la elaboración de esta investigación, las cuales pueden ser consideradas con el objetivo de continuar trabajando en aplicaciones paralelas:

- Ejecutar aplicaciones que requieran grandes capacidades de procesamiento.
- Buscar que diferentes instituciones a nivel nacional se logren integrar para que puedan compartir sus capacidades de cómputo.
- Sacar mayor provecho de la red de alta velocidad Red Nacional Académica de Tecnología Avanzada.
- Lograr integrarse a proyectos desarrollados a nivel mundial.
- Permitir el desarrollo de proyectos de investigación que requieran o necesiten grandes capacidades de cómputo.
- Permitir que grupos de investigación o instituciones con bajos recursos puedan a acceder a estos recursos para lograr contribuir con sus investigaciones.
- Desarrollo de una infraestructura de grilla institucional entre los diferentes grupos de investigación que existen dentro de la institución.

Bibliografía Y Referencias

Adarsh's Website. (s.f.). Obtenido de http://adarshpatil.com

Allspaw, J., & Robbins, J. (2010). *Web Operations.* O'Reilly Media.

Castro, C. (2016). *Prezi.* Obtenido de https://prezi.com/zpkscl-l6v4u/orangefs/?webgl=0

Chapman&Hall. (2008). *Handbook of Parallel Computing.* Rajasekaran/Reif.

Coulouris, G., Dollimore, J., & Kindberg, T. (Rivera de Loira). *Sistemas Distribuidos Conceptos y Diseño.* 2001: Addison Wesley.

Domingo Gimenéz, F. A., Mantas, J. M., & Vidal, A. M. (2008). *Introducción a la programación paralela.* Paraninfo Cengage Learning.

Follet, M. (05 de 08 de 2015). *SoyAdmin.com*. Obtenido de https://soyadmin.com/2015/02/nmon-un-sistema-de-monitorizacion-que-no-puede-faltarte-en-centos-6-y-7-2/

Foster, I., & Kesselman, C. (1998). *The Grid.* San francisco: Morgan Kaufmann.

Ganglia Monitoring, S. (2014). *Wikipedia*. Obtenido de https://en.wikipedia.org/wiki/Ganglia_(software)

Gonzalez, S. (s.f.). *Sergio Gonzalez*. Obtenido de http://www.sergio-gonzalez.com/doc/01-pvfs/html/pvfs.html

Henderson, C. (2006). *Building Scalable Web Sites.* O'Reilly Media.

Introducción a los Sistemas Distribuidos. (2004).

Jethva, H. (2016). *Profitbricks The IaaS Company*. Obtenido de https://devops.profitbricks.com/tutorials/analyze-linux-performance-using-nmon-on-centos-7-1/

Koziol, Q., & Prabhat. (2015). *High Performance Parallel I/O.* Berkeley: CRC Press.

Mark, B., & Buyyaz, R. (s.f.). *Cluster Computing at a Glance.* Southsea: University of Portsmouth.

Miah, W. (2015). *Mathematics and Computer Science.* Obtenido de http://www.mcs.anl.gov

NASA, C. A. (2012). *NASA*. Obtenido de https://nex.nasa.gov

NERSC. (2017). Obtenido de http://www.nersc.gov

Nesmachnow, S., Iturriaga, S., & Rocchetti, N. (2017). Obtenido de https://www.fing.edu.uy

Nicholes, B. (2008). *Mivro Focus.* Obtenido de https://www.novell.com

Pacheco, P. (2011). *An Introduction to Parallel Programing.* Burlington: Morgan Kaufmann.

Padua, D. (2011). *Encyclopedia of Parallel Computing.* Springer.

Performance Optimisation and Productivity. (22 de 02 de 2017). Obtenido de https://pop-coe.eu/blog/profiling-io-of-hpc-applications-with-darshan

Prabhat, & Koziol, Q. (2015). *High Performance Parallel E/S.* Illinois: Chapman & Hall/CRC.

República, F. d. (2009). *ARQUITECTURAS PARALELAS.* Uruguay: Facultad de Ingeniería Universidad de la República.

Ruiz Navarro, J. E. (21 de 07 de 2014). *Systerminal*. Obtenido de http://www.systerminal.com

Savchenko, A. (16 de 02 de 2013). *LVEE.* Obtenido de https://lvee.org/en/main

Scheinine, A. (s.f.). *Introduction to Parallel Programming Concepts.* Louisiana: Center for Computational Technology and Information Technology Services.

Shan, H. (2007). *Using IOR to Analyze the I/O performance for HPC Platforms.* Obtenido de https://cug.org

Simarro, C., & Towers, P. (04 de 16 de 2015). *ECMWF.* Obtenido de https://www.ecmwf.int/

Supercomputing, C. L. (2011). *Leibniz Supercomputing Centre*. Obtenido de https://www.lrz.de

Timtthy, J. R. (2010). *A Specimen or parallel Programing.* Parallal Sort Implementation.

top500. (28 de 06 de 2006). Obtenido de https://www.top500.org/

Tovar, E. N., Hernández Palacios, R., Camacho Cruz, H. E., Díaz García, A. F., Anguita López, M., & Ortega Lopera, J. (2009). *Universidad Autónoma del Estado de Hidalgo.* Obtenido de https://www.uaeh.edu.mx

Wikipedia. (15 de 12 de 2016). Obtenido de https://en.wikipedia.org/wiki/OrangeFS

Anexos

Ensamblaje del Clúster

- Procedemos a limpiar y a alistar el rack para empezar a montar casa servidor dentro del rack.

Ilustración 27 Chasis Clúster

- Se procede a montar los servidores sobre los rieles para después lograr hacer su montaje dentro del rack.

Ilustración 28 Montaje del primer nodo

- Después de haber instalado y aprendido a realizar el montaje del primer servidor se procede a realizar el montaje de todos los servidores restantes.

Ilustración 29 Finalización montaje nodos frontal

- Como se dejaron espacios muy grandes entre cada servidor fue necesario desmontar todos los servidores y volverlos a montar, pero esta vez sin dejar espacios entre ellos.

Ilustración 30 Finalización montaje nodos trasera

- Procedemos a realizar el montaje del servidor principal quien va a ser el que llevar el control y administración de los otros servidores.

Ilustración 31 Montaje nodo Master frontal

- Se procede asegurando cada uno de los servidores para que no puedan ser extraídos y que no se vayan a salir del rack.

Ilustración 32 Montaje nodo Master trasero

- Procedemos a realizar el montaje de los Switch para su posterior cableado hacia los servidores

Ilustración 33 Montaje de los Switch

- Se realizan las conexiones necesarias entre los Switches y los servidores.

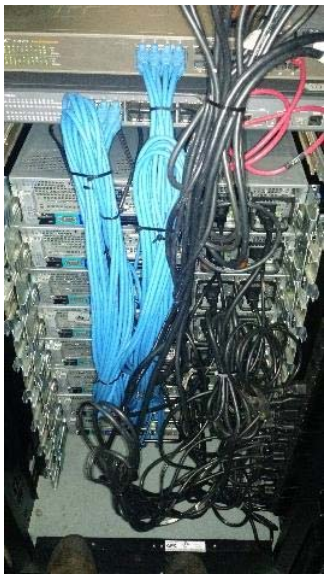


Ilustración 34 Cableado de los Switch con los nodos

- Se organizan los cables (Peinar) para controlar el mantenimiento y distribución.

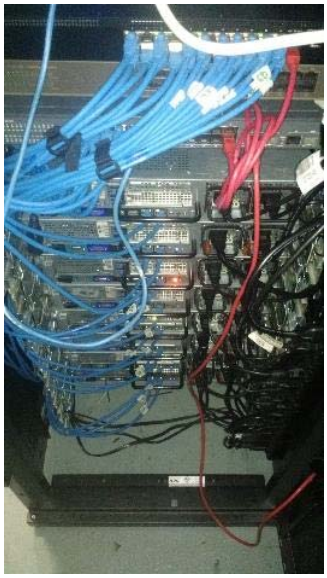


Ilustración 35 Peinado de los cables Parte 1

Ilustración 36 Peinado de los cables Parte 2

- Se alimentan los Switches con energía y se procederá a realizar las pertinentes pruebas.

Ilustración 37 Conexión cables de poder Switch

- Empezamos a encender cada uno de los servidores y a realizar pruebas de funcionamiento

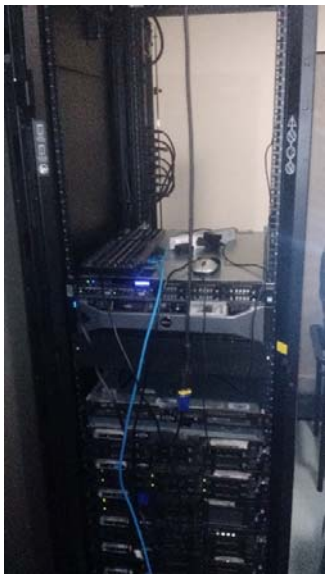


Ilustración 38 Encendida y prueba de la corriente en todos los nodos

- Una vez terminadas las pruebas procedemos a realizar el formateo e instalación del software necesario.

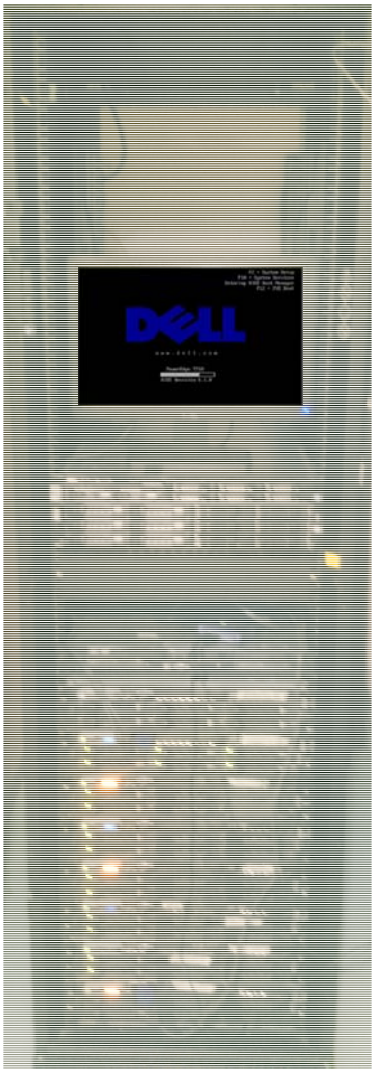


Ilustración 39 Inicio Proceso de Instalación

- Ingresamos al BIOS de cada nodo y empezamos el proceso de formateo a cada servidor.

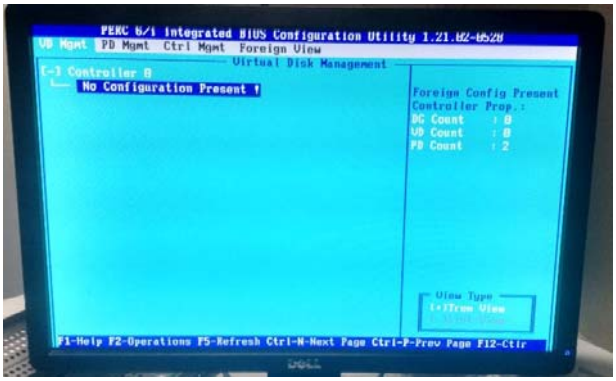


Ilustración 40 BIOS del nodo

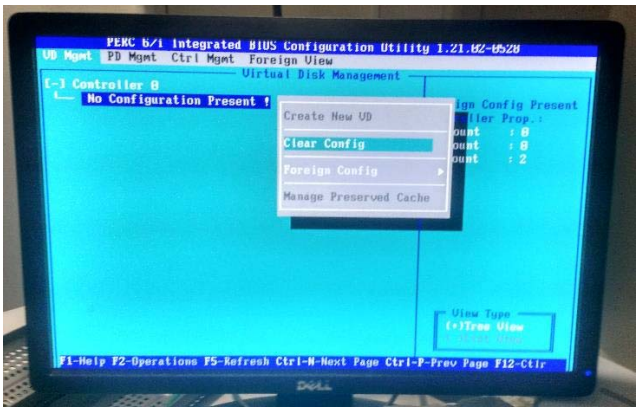


Ilustración 41 Formateo del nodo parte 1

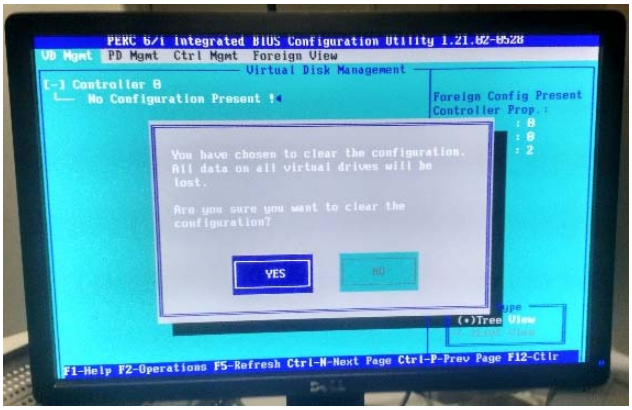


Ilustración 42 Formateo del nodo parte 2

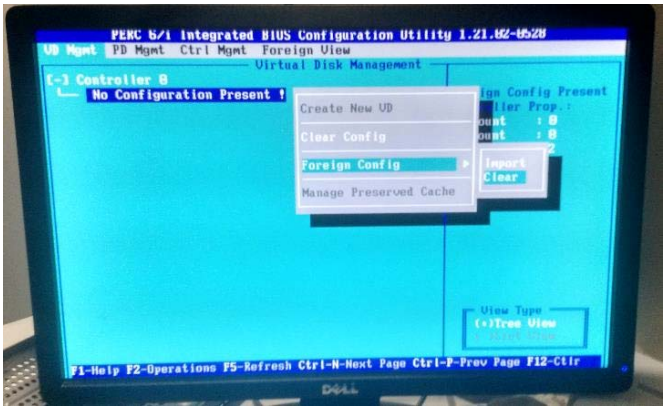


Ilustración 43 Formateo del nodo parte 3

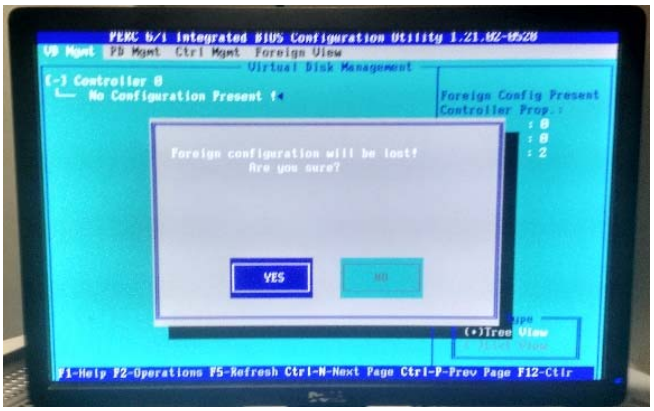


Ilustración 44 Formateo del nodo parte 4

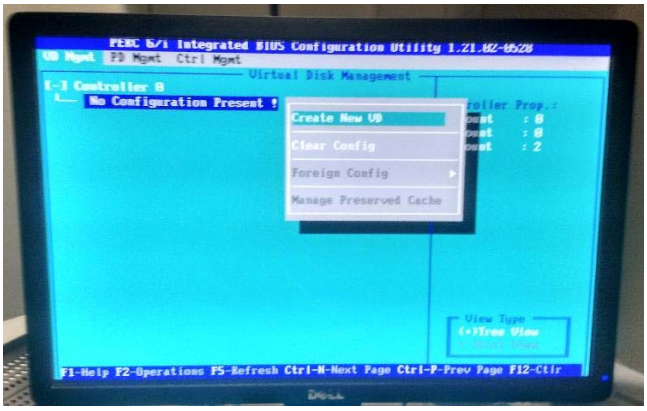


Ilustración 45 Formateo del nodo parte 5

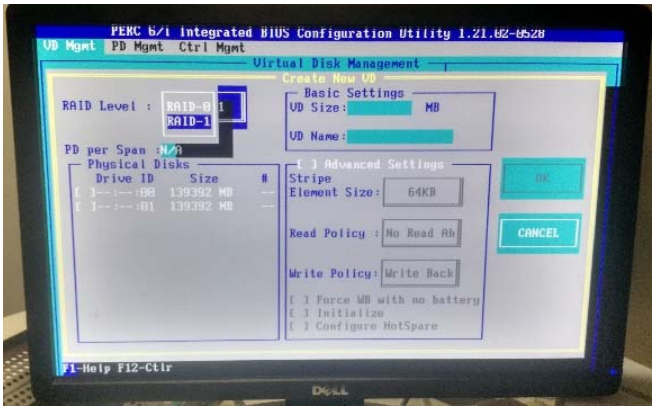


Ilustración 46 Formateo del nodo parte 6

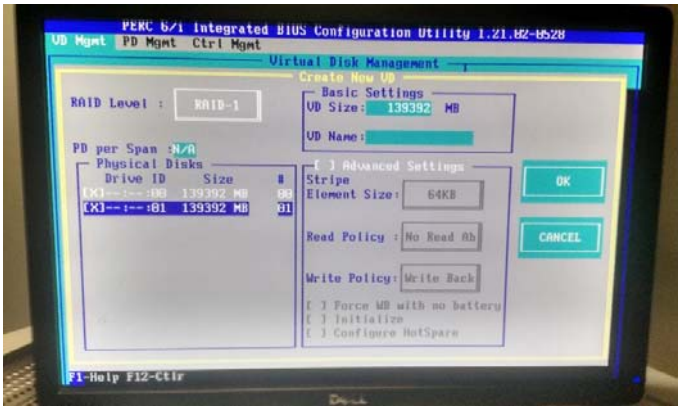


Ilustración 47 Formateo del nodo parte 7

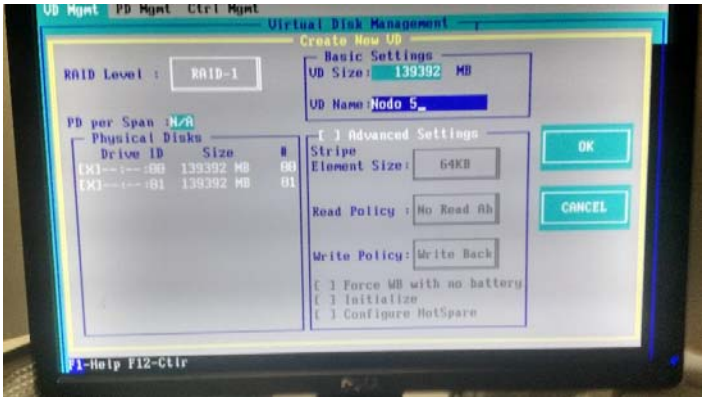


Ilustración 48 Formateo del nodo parte 8

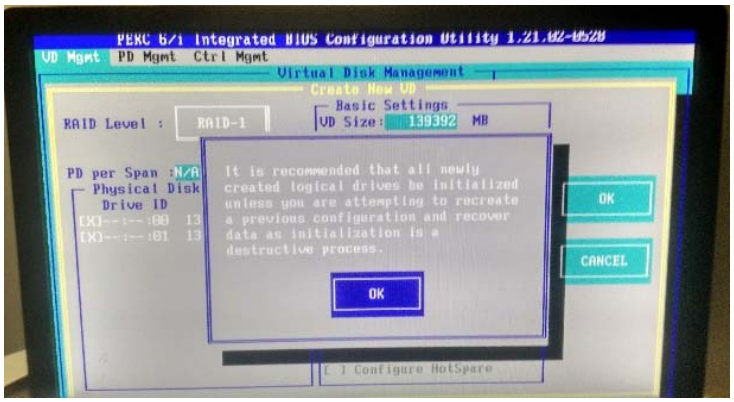


Ilustración 49 Formateo del nodo parte 9

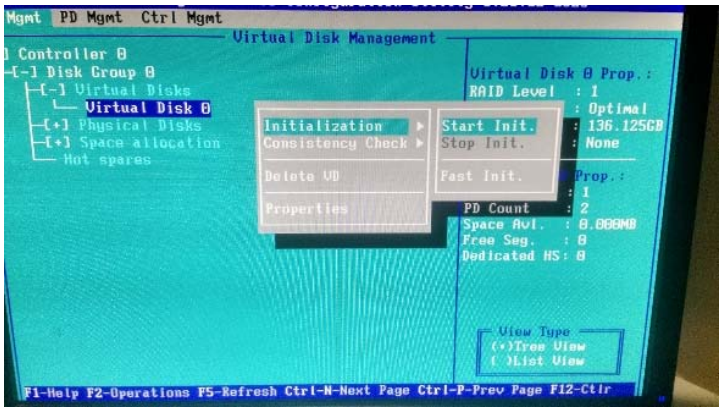


Ilustración 50 Formateo del nodo parte 10

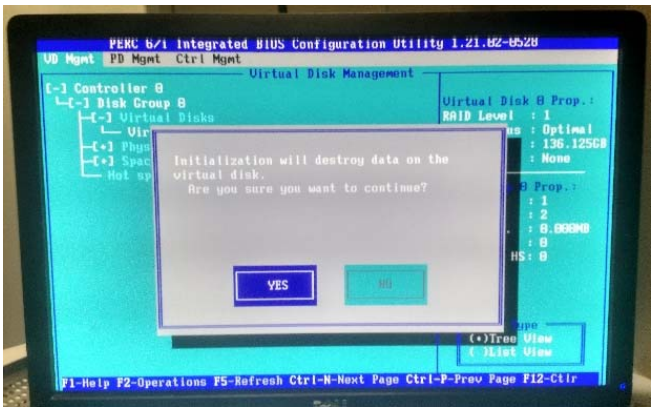


Ilustración 51 Formateo del nodo parte 11

8.1 Configuración de los nodos

Ilustración 52 Configuración nodo 1

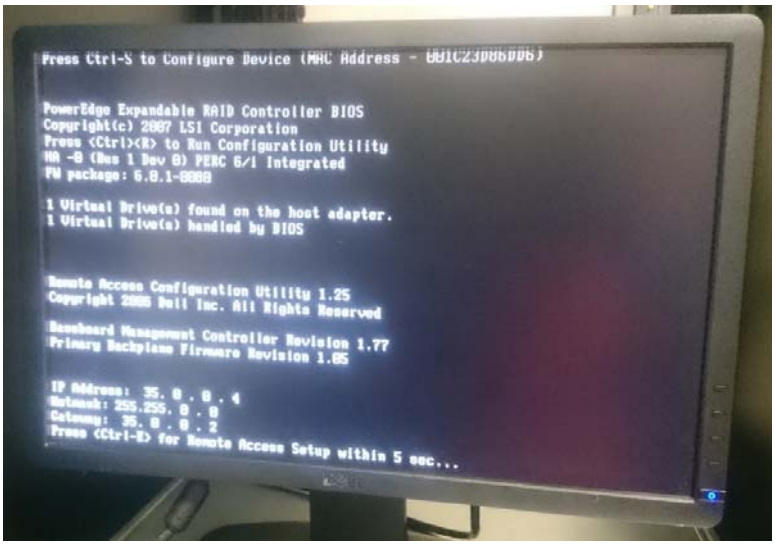


Ilustración 53 Configuración nodo 1

8.1.1.1 *Configuración del Nodo No. 1*

Procesador: Quad-core de 2.00 GHz

Bus de datos: 1333 MHz

Memoria RAM: 8 GB

Cache: 2x6 MB

Dirección IP: 35.0.0.4

Mascara: 255.255.0.0

Gateway: 35.0.0.2

Ilustración 54 Configuración nodo 2

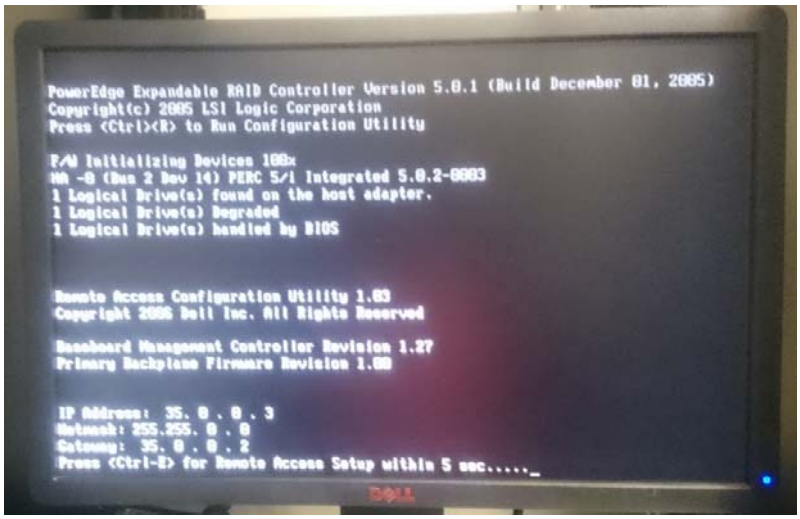


Ilustración 55 Configuración nodo 2

8.1.1.2 *Configuración del Nodo No. 2*

Procesador: Dual-Core de 1.60 GHz

Bus de datos: 1066 MHz

Memoria RAM: 2 GB

Cache: 4 MB

Dirección IP: 35.0.0.3

Mascara: 255.255.0.0

Gateway: 35.0.0.2

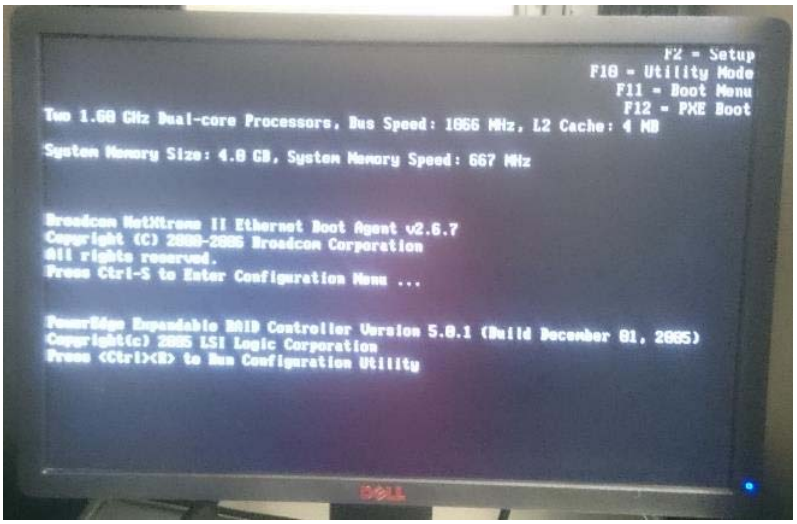


Ilustración 56 Configuración nodo 3

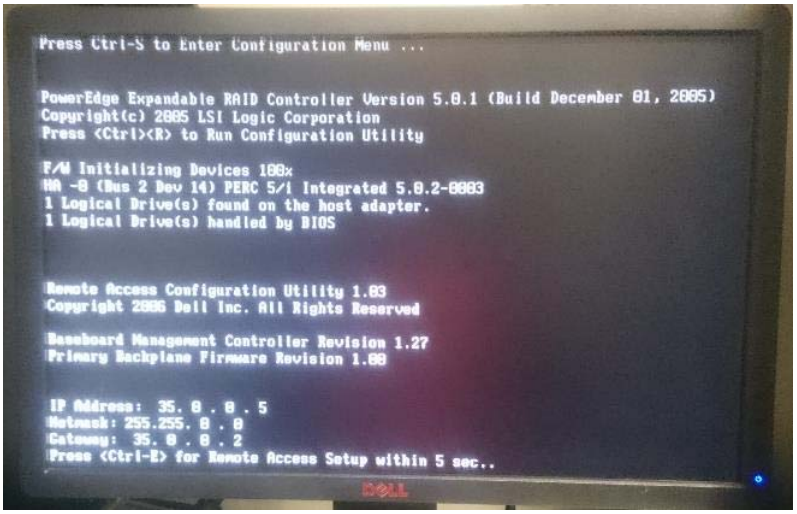


Ilustración 57 Configuración nodo 3

8.1.1.3 *Configuración del Nodo No. 3*

Procesador: Dual-Core de 1.60 GHz

Bus de datos: 1066 MHz

Memoria RAM: 4 GB

Cache: 4 MB

Dirección IP: 35.0.0.5

Mascara: 255.255.0.0

Gateway: 35.0.0.2

Ilustración 58 Configuración nodo 4

Ilustración 59 Configuración nodo 4

8.1.1.4 *Configuración del Nodo No. 4*

Procesador: Dual-Core de 2.00 GHz

Bus de datos: 1333 MHz

Memoria RAM: 8 GB

Cache: 4 MB

Dirección IP: 35.0.0.3

Mascara: 255.255.0.0

Gateway: 35.0.0.2

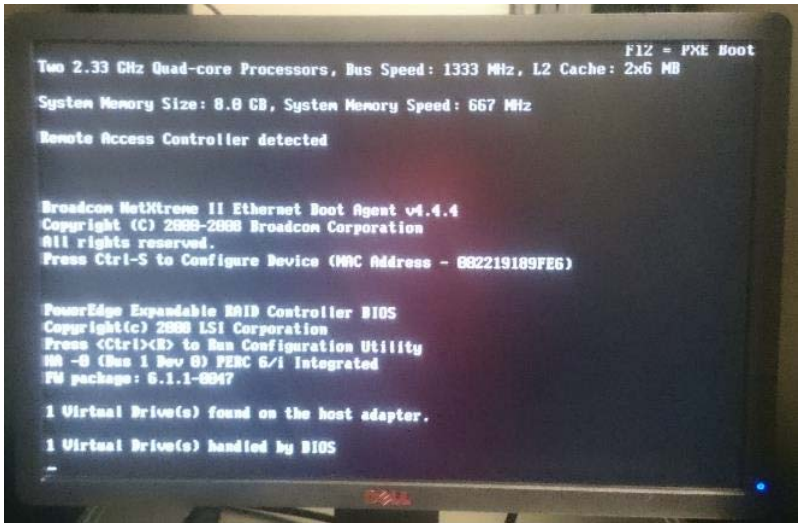


Ilustración 60 Configuración nodo 5

Ilustración 61 Configuración nodo 5

8.1.1.5 *Configuración del Nodo No. 5*

Procesador: Dual-Core de 2.33 GHz

Bus de datos: 1333 MHz

Memoria RAM: 8 GB

Cache: 2x6 MB

Dirección IP: 35.0.0.7

Mascara: 255.255.0.0

Gateway: 35.0.0.2

Ilustración 62 Configuración nodo 6

Ilustración 63 Configuración nodo 6

8.1.1.6 *Configuración del Nodo No. 6*

Procesador: Dual-Core de 1.60 GHz

Bus de datos: 1066 MHz

Memoria RAM: 4 GB

Cache: 4 MB

Dirección IP: 35.0.0.8

Mascara: 255.255.0.0

Gateway: 35.0.0.2

8.2 Configuración del nodo Master

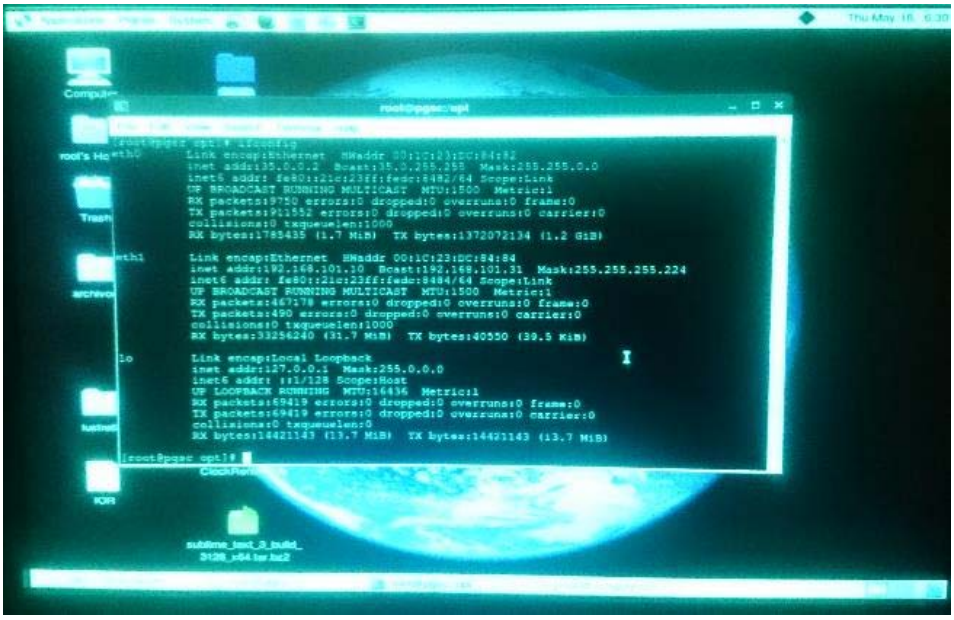


Ilustración 64 Configuración nodo Master

Configuración del Nodo Master

Procesador: Quad-core de 2.00 GHz

Bus de datos: 1333 MHz

Memoria RAM: 16 GB

Cache: 2x6 MB

Dirección IP: 35.0.0.2

Mascara: 255.255.0.0

Gateway: 35.0.0.2