Implementación de un prototipo de plataforma tecnológica tipo cluster, para la computación de algoritmos realizados en mpi presentados en el campus universitario.

Daniel Adolfo Diaz Giraldo
Anderson Alberto Ochoa Estupiñán

Implementación de un prototipo de plataforma tecnológica tipo cluster, para la computación de algoritmos realizados en mpi presentados en el campus universitario.

Daniel Adolfo Diaz Giraldo
Anderson Alberto Ochoa Estupiñán

Trabajo de grado presentado como requisito
parcial para optar al título de
Ingeniero de Sistemas y computación

Pereira, Diciembre de 2015
UNIVERSIDAD TECNOLÓGICA DE PEREIRA
Programa de Ingeniería en Sistemas y Computación.
Grupo de investigación Sirius

Implementación de un prototipo de plataforma tecnológica tipo cluster, para la computación de algoritmos realizados en mpi presentados en el campus universitario.

Anderson Alberto Ochoa Estupiñán

Director: Ramiro Andrés Valencia Barrios

Pereira, Diciembre de 2015
Programa de Ingeniería Eléctrica.
Universidad Tecnológica de Pereira
La Julita. Pereira(Colombia)
TEL: (+57)(6)3137122
www.utp.edu.co
Versión web disponible en: *http://recursosbiblioteca.utp.edu.co/tesisd/index.html*

Agradecimientos

En este pequeño espacio quisiéramos agradecer a todos los docentes que nos dieron la mano a lo largo de la carrera, nuestras familias que fueron nuestro apoyo y a todos los compañeros que tuvimos a lo largo de la misma.

Resumen

Actualmente, la utilización de medios informáticos a lo largo de la academia se ha vuelto cada vez más imprescindible a la hora de representar cualquier tipo de modelo computacional, ya sea como herramienta para la representación de datos o como núcleo y fuerza de trabajo para elaborar algoritmos de algún propósito en específico. La evolución del computo científico está encaminada a procesos de tipo hpc [1], donde cada vez, son más los algoritmos que dependen de grandes y costosas infraestructuras para poder funcionar, no está de más entonces, concebir una manera de utilizar los recursos a disposición para mejorar la producción académica e intelectual de nuestra universidad.
La idea concéntrica planteada en este proyecto de grado es modular un sistema escalable para aprovechar recursos computacionales de componentes comunes (Pc's a lo largo de todo el campus universitario) usando mpi[2] como base para la comunicación de estos componentes, todo esto accesible de manera web, encaminado a la obtención de mejores resultados en los procesos investigativos y académicos emergentes en el campus universitario.
Éste documento contendrá paso a paso el proceso de elaboración de un cluster computacional funcional bajo arquitecturas estables que serán descritas de manera puntual; todos esto sustentado en tecnologías open-source[3] . Las razones por las cuales el proyecto descrito, presente factibilidad, es debido concretamente a los siguientes beneficios:

- Computar algoritmos con una optimización notable.
- Accesibilidad,portabilidad,eficiencia, dado la posibilidad de acceder a través de una plataforma web.
- Disminución de costos de tecnologías especializadas.

[1] High performance Computing, es el uso de procesamiento en paralelo para la ejecución de programas de aplicación avanzada de manera eficiente, fiable y rápida

[2]Message Passing Interface, para la comunicación entre nodos de un sistema distribuido

[3] software o hardware distribuido y desarrollado libremente (acceso al código fuente) y en cuestiones éticas.

- Mejorar la eficiencia e inteligencia colectiva de la universidad, puntualmente de los grupos de investigación.
- Mejorar el uso de los nodos computacionales dentro de la universidad.

Finalmente se dejara el código fuente de la plataforma web, que estará a disposición del grupo de investigación sirius y el campus en general.

Palabras claves: cluster, web, mpi, hpc, sirius, procesamiento paralelo, plataforma, web.

Tabla de Contenido

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

1.1. Planteamiento del problema

Actualmente el campo de la computación distribuida se ha afianzado como una de las mejores maneras de computar algoritmos de alta demanda, tanto en campos de la ciencia como la biología, física, medicina, bioinformática, entre otras, utilizando diferentes estrategias informáticas. La optimización de algoritmos[12] es un tema que ayudó al desarrollo de herramientas para utilizar un conjunto coordinado de recursos dispersos que unitariamente no aportaban mayor poder computacional, pero en conjunto, resolvían las necesidades emergentes. Las técnicas actuales de computación de alto rendimiento ostentan la utilización de recursos comunes, presentes, en cada equipo de cómputo promedio, un ejemplo de ello son las vitales CPU o las unidades de procesamiento gráfico (GPU).

Proveedores como Amazon, Nvidia entre otros, ofrecen servicios de computación distribuida rentados por tiempos o por convergencia de solución algorítmica. Estos servicios ofrecen grandes cantidades de FLOPS[1] a disposición; además de las soluciones competitivas y administrativas que involucran las diferentes compañías de este tipo computación distribuida; lo que presenta inconvenientes respecto a lo que se desea computar. Se resalta entonces que este tipo de proveedores tengan un mercado selecto, establecido y predeterminado lo que hace que el costo para pequeñas instituciones sea elevado y no se acomode a las necesidades puntuales de éstas.

En lo que concierne a la universidad, la presencia o necesidad en los últimos años de este tipo de algoritmos ha ido creciendo a grandes pasos. Grupos de investigación, semilleros, como

[1]FLoating-point Operations Per Second.Medida de la eficiencia computacional. Unidad que determina la cantidad de operaciones que se pueden realizar en un segundo

docentes o alumnos han presentado algoritmos de diferentes tipos algunos ya distribuidos o que se podrían optimizar con técnicas de alto rendimiento computacional.

1.2. Justificación

Gracias a las investigaciones realizadas por Bernardo Cuervo en su proyecto de grado[1] y por la retroalimentación obtenida por de diferentes docentes, estudiantes e investigadores, se dieron a conocer las diferentes problemáticas que tienen que ver con el tema.
Los enfoques de procesamiento de HPC son enfoques que recientemente aparecen dada la demanda del campo científico que han marcado un hito en la historia de la computación aún más en el ámbito investigativo. Los altos costos que demanda este tipo de computación (como también el HTC[2]), junto a la combinación de los enfoques mencionados con tecnologías como grids[3] o mallas superponen una gran costo de manutención e implementación donde muchas instituciones educativas no pueden mantener. Una vez planteadas las condiciones generales de la universidad en el campo de computación distribuida, seguido de las ideas presentadas por grupos de investigación dentro del campus, junto a proyectos de grados anteriores, se ha visto la posibilidad de implementar una plataforma tecnológica del tipo cluster[4] la cual podría ayudar e incursionar en el campo distribuido ya señalado , dando con esto ventajas investigativas, sumamente importantes a la hora de obtener resultados en los tiempos de ejecución de algoritmos. Esta idea tiene como objetivo acarrear con el desarrollo tecnológico (infraestructura y adecuación de equipos) todo esto sustentado en la infraestructura actual alojada en equipos de cómputo similares a los usados en las salas de sistemas del centro de recursos informáticos y educativos (CRIE)[5]. Se mantiene la idea principal de diseñar, elaborar e implementar en un prototipo de infraestructura con un entorno computacional estable, accesible y elaborable dentro de los recursos actuales del ambiente de trabajo.
Sentando lo anterior, se le brindará a todo el campus la oportunidad de acceder a una plataforma para el procesamiento de algoritmos de alta carga computacional con una disponibilidad razonable.
La plataforma será capaz de interpretar a nivel de código lo implementado en la librería de pasos de mensajes MPI.

[2]"High-Throughput ComputingÇomputación que dispone generalmente de una gran cantidad de recursos en un corto periodo de tiempo

[3]Conjunto de recursos informáticos localizados espacialmente en diferentes zonas que se coordinan para alcanzar un objetivo común

[4]Computadoras conectadas entre sí que se comportan como si fuesen una misma computadora

[5]Centro de Recusos Informáticos y Educativos

1.3. Objetivos

1.3.1. General

Diseñar e implementar un prototipo de plataforma tecnológica distribuida, capaz de computar algoritmos de alta demanda computacional presentados en el campus universitario, soportados a través de la librería MPI.

1.3.2. Específicos

- Obtención de requerimientos, análisis de requisitos, limitaciones, especificaciones y diseño sistemático del prototipo.
- Desarrollo del prototipo de plataforma distribuida dentro del campus universitario.
- Implementación del prototipo de plataforma funcional.

1.4. Estado del arte

1.4.1. Definición cluster

" Un cluster cuya traducción al castellano sería ('racimo', 'conjunto' , 'grupo' o 'cúmulo') es una solución computacional conformada por un conjunto de sistemas computacionales muy similares entre si (grupo de computadoras), interconectados mediante alguna tecnología de red de alta velocidad, configurados de forma coordinada para dar la ilusión de un único recurso; cada sistema estará proveyendo un mismo servicio ejecutando una (o parte de una) misma aplicación paralela. La característica inherente de un cluster es la compartición de recursos: ciclos de CPU (Central Processing Unit), memoria, datos y servicios."[9]. Los clusters están conformados por nodos genéricos single-core o multicore, que a su vez podrían usar hardware de procesamiento como la CPU o la GPU[6]. Dichos nodos pueden estar exclusivamente dedicados a labores del cluster o pueden destinar una parte para ello. Actualmente ocupan un papel crucial en el campos científicos, tecnológicos, comerciales ya que aportan una plataforma capaz de computar grandes lotes de información utilizada en cada tipo de cómputo. En otras palabras es la unión de varias computadoras (nodos) a través de una red, que parecen ante el usuario como un solo equipo de cómputo y que trabajan en conjunto de manera funcional, gracias a alguna aplicación.

[6]GPUS: (graphics processing unit) circuito especializado en el procesamiento de operaciones gráficas

1.4.2. Características sistemas distribuidos

Los sistemas distribuidos suponen un paso más en la evolución de los sistemas informáticos, entendidos desde el punto de vista de las necesidades que las aplicaciones plantean y las posibilidades que la tecnología ofrece. Antes de proporcionar una definición de sistema distribuido resultará interesante presentar, a través de la evolución histórica, los conceptos que han desembocado en los sistemas distribuidos actuales, caracterizados por la distribución física de los recursos en máquinas interconectadas. Un sistema distribuido que pretenda ofrecer una visión de sistema único deberá cumplir las propiedades que se presentan a continuación.[13]:

Transparencia

El objetivo esencial de un sistema distribuido es proporcionar al usuario y a las aplicaciones una visión de los recursos del sistema como gestionados por una sola máquina virtual. La distribución física de los recursos es transparente. Pueden describirse diferentes aspectos de la transparencia:

- De **identificación:** Los espacios de nombres de los recursos son independientes de la topología de la red y de la propia distribución de los recursos. De esta forma, una aplicación puede referirse a un recurso con un nombre independientemente de en qué nodo se ejecute. Por ejemplo, en NFS (concepto que se verá mas adelante) un sistema de ficheros remoto se monta en un punto del sistema de ficheros local. Que una instalación de NFS proporcione transparencia de identificación a las aplicaciones depende de que todos los nodos cliente tengan montado el sistema remoto en el mismo punto de montaje, lo que es generalmente responsabilidad del administrador.

- De **ubicación física de los recursos:** Ni los usuarios ni las aplicaciones conocen en qué nodo reside el recurso accedido, o si éste es local o remoto. Esto implica también que los recursos pueden migrar entre nodos sin que las aplicaciones se vean afectadas.En NFS, podemos migrar un sistema de ficheros remoto de un nodo a otro y la transparencia de la ubicación se preservará si se actualizan convenientemente las tablas de montaje de los nodos cliente.

- De **replicación:** Ni los usuarios ni las aplicaciones conocen cuántas unidades hay de cada recurso, ni si se añaden o eliminan copias del recurso. En NFS los clientes gestionan caches locales de los ficheros remotos. El “caching” es una forma restringida de replicación que requiere alguna forma de validación (en NFS mediante encuesta) por los clientes para preservar la consistencia6. Aún así, la semántica UNIX que NFS

pretende ofrecer a las aplicaciones puede verse a veces comprometida cuando varios clientes escriben sobre un mismo fichero.

- De **paralelismo:** Una aplicación puede ejecutarse en paralelo, sin que la aplicación tenga que especificarlo, y sin consecuencias sobre la ejecución, salvo por cuestiones de rendimiento. Esta propiedad afecta a los sistemas que permiten distribuir procesos y memoria. En el caso de un sistema de ficheros, sólo es relevante cuando las aplicaciones bloquean temporalmente el acceso a ficheros, lo que se especifica de forma explícita (mediante primitivas de lock y unlock). Las últimas versiones de NFS incluyen este mecanismo.

- De **compartición:** El que un recurso compartido intente ser accedido simultáneamente desde varias aplicaciones no tiene efectos sobre la ejecución de la aplicación. Como se resalto anteriormente, en NFS esta propiedad puede verse afectada por la existencia de caching, en particular si los periodos de la encuesta de validación son elevados.

- De **rendimiento:** Inevitablemente, implementar las propiedades de los sistemas distribuidos será a costa de una pérdida de rendimiento. Por lo tanto, generalmente es necesario buscar soluciones de compromiso. Así, en NFS la minimización del periodo de validación de la cache permitiría hacer casi totalmente transparente la existencia de caches, a costa de incrementar el tráfico de la red y en consecuencia las latencias.

Escalabilidad

Una de las características de los sistemas distribuidos es su modularidad, lo que le permite una gran flexibilidad y posibilita su escalabilidad, definida como la capacidad del sistema para crecer sin aumentar su complejidad ni disminuir su rendimiento. Uno de los objetivos del diseño de un sistema distribuido es extender la escalabilidad a la integración de servicios.

La escalabilidad presenta dos aspectos. El sistema distribuido debe (1) proporcionar espacios de nombres suficientemente amplios, de forma que no supongan una limitación inherente, y (2) mantener un buen nivel de rendimiento en el acceso a los recursos cuando el sistema crece.

- **Espacios de nombres:** Los espacios de nombres, al igual que en los sistemas centralizados, pueden identificar objetos de diferente naturaleza, como ficheros, procesos, variables, o incluso direcciones de memoria (en los sistemas de memoria compartida distribuida, DSM). En el caso de los espacios lineales, como la memoria, existe una

limitación inherente asociada al tamaño del nombre, de forma que hoy en día es razonable plantear la insuficiencia de los espacios de direcciones de 32 bits. En otros casos, los espacios de nombres son jerárquicos y por lo tanto escalables por naturaleza.

- **Complejidad-rendimiento:** El crecimiento de un sistema distribuido puede introducir cuellos de botella y latencias que degradan su rendimiento. Además del incremento de los costes de comunicación por el aumento de la distancia física entre los componentes del sistema, la complejidad estructural de los algoritmos distribuidos es a menudo más que lineal con respecto al tamaño del sistema. Es necesario, por tanto, establecer compromisos entre tamaño del sistema, rendimiento y complejidad.

Fiabilidad y tolerancia a fallos:

La fiabilidad de un sistema puede definirse como su capacidad para realizar correctamente y en todo momento las funciones para las que se ha diseñado. La fiabilidad se concreta en dos aspectos:

- **Disponibilidad:** Es la fracción de tiempo que el sistema está operativo. El principal parámetro para medir la disponibilidad es el tiempo medio entre fallos (MTBF), pero hay que considerar también el tiempo de reparación. La disponibilidad se puede incrementar de dos formas: (a) utilizando componentes de mayor calidad, y/o (b) con un diseño basado en la replicación de componentes que permita al sistema seguir operando aún cuando alguno(s) de ellos falle(n). Ambas alternativas incrementan el coste del sistema; sin embargo, en el estado tecnológico actual, la replicación resulta, en general, menos costosa. Los sistemas distribuidos proporcionan inherentemente la replicación de algunos recursos (por ejemplo, unidades de proceso), mientras que otros normalmente compartidos (por ejemplo, un servidor de ficheros) pueden replicarse para aumentar la disponibilidad. Por otra parte, la ausencia de fallos en los componentes de un sistema, tanto hardware como software, nunca puede garantizarse, de modo que, más allá de unos límites, la replicación es necesaria para seguir incrementando la disponibilidad, ya que la probabilidad de fallo disminuye como una función exponencial de la replicación. Por ejemplo, dada una probabilidad de fallo de un 1 % en un componente (en un periodo de tiempo dado), si se monta un sistema replicado con cuatro componentes idénticos, la probabilidad de que fallen en ese periodo los cuatro componentes disminuiría a 0,000001 %.

- **Tolerancia a fallos:** Aún con una alta disponibilidad, un fallo en un momento determinado puede tener consecuencias desastrosas. Por ejemplo sistemas de tiempo real

críticos que controlan dispositivos vitales (por ejemplo en medicina, centrales nucleares...). Es decir, aunque la replicación aumenta la disponibilidad, no garantiza por sí sola la continuidad del servicio de forma transparente. La tolerancia a fallos expresa la capacidad del sistema para seguir operando correctamente ante el fallo de alguno de sus componentes, enmascarando el fallo al usuario o a la aplicación. Por lo tanto, la tolerancia a fallos implica (1) detectar el fallo, y (2) continuar el servicio, todo ello de forma transparente para la aplicación (transparencia de fallos).

Consistencia

La distribución de recursos introduce importantes beneficios. Por una parte, contribuye al incremento del rendimiento a través del paralelismo y promoviendo el acceso a copias locales del recurso (disminuyendo los costes de comunicación). Por otra, como se acaba de ver, la replicación aumenta la disponibilidad, siendo la base para proporcionar tolerancia a fallos. No obstante, distribuir recursos acarrea algunos problemas. Por una lado, la red de interconexión es una nueva fuente de fallos. Además, la seguridad del sistema es más vulnerable ante accesos no permitidos. Pero el problema de mayor complejidad es el de la gestión del estado global para evitar situaciones de inconsistencia entre los componentes del sistema. Los nodos del sistema se hallan físicamente distribuidos, por lo que la gestión del estado global depende fuertemente de los mecanismos de comunicación, a su vez soportados por una red sujeta a fallos. Como se ve en la actualidad, la gestión de la consistencia puede basarse en una buena sincronización entre los relojes de los nodos o en mecanismos de ordenación de eventos (relojes lógicos). La distribución física hace, en general, inviable la utilización de un reloj global que aporte referencias absolutas de tiempo, lo que permitiría una ordenación total de los eventos y, por lo tanto, de las transiciones de estado en cada nodo.

1.4.3. Clasificación de los clusters

La base de la clasificación de los clusters están establecidas por su uso y servicios que ofrecen.

Según su aplicación:

- **Alta disponibilidad**: Son clusters que tienen como finalidad asegurar la fiabilidad de un sistema, son utilizados cuando la disponibilidad es un factor clave a la hora de computar esquemas, un ejemplo de ello son las transacciones de comercio electrónico en tiempo real. Estos sistemas aseguran la disponibilidad manteniendo servicios activos

compartidos entre si, a su vez, son monitorizados de manera constante. Generalmente se clasifican en 2 subclases:

- **Alta disponibilidad de infraestructura**: frente a una falla de hardware, el software especializado sera capaz de iniciar de nuevo los servicios en otros componentes activos del cluster (failover), cuando dicho componente se recupere, el software sera capaz de migrar nuevamente los servicios a la maquina original, esto es failback. Esta capacidad de recuperación automática minimiza la percepción de fallo por parte de los usuarios.
- **Alta disponibilidad de aplicación**: frente a una falla de software o hardware, el sistema sera capas de proceder de manera correcta salvaguardando la integridad y el estado de correctitud del cluster arrancando de nuevo los servicios en los demás nodos del sistema. Esto garantiza integridad de la información, ya que intenta salvaguardar la información de la mejor manera.

- **Alto rendimiento**: Para la ejecución de algoritmos que demanden gran cantidad de recursos, un ejemplo de ellos son los programas de Rendering, usados en el procesamiento de imágenes. Se distinguen por ser una tecnología que trabaja en un conjunto de computadoras de manera paralela, prioritariamente son usados en problemas de gran complejidad y de plataformas de costo elevado.
- **Según el propietario de los nodos**:
  - **Nodos dedicados:** nodos exclusivos para uso del cluster, generalmente se encuentran conectados por lan dentro de una zona común y sin otro componente conectado como periféricos.
  - **Nodos no dedicados:** nodos con uso genérico que hacen parte del cluster pero pueden realizar otras tareas.
- **Según la configuración del cluster:** Son diversos los factores que intervienen directamente en la configuración de un cluster, entre los mas importantes se destancan dos: el tipo de computo que tendrán y el propietario de los nodos.
  - **Según su propietario:**
    - **Homogenéos :** Conjunto de maquinas bajo arquitecturas de software y hardware idénticas.

- **Heterogéneos:** Conjunto de nodos que no comparten relación entre sus características de hardware y software.

- **Según el tipo de cómputo:**
  - **High Performance (Alto rendimiento) :** Para tareas que demanden alto poder computacional (memoria, procesamiento, en algunos casos almacenamiento). También pudiendo comprometer la demanda de recursos por largos periodos de tiempo. Son pensados para solventar problemas de cómputo pesados, además de ser costosos. Se basan el la premisa de dividir los problemas en partes mas pequeñas, dando así poderes de computo mucho mejores que las supercomputadoras dedicadas mucho mas costosas. Para poder beneficiarce de la división de problemas los programadores entonces, están obligados a realizar sus implementaciones pensando en paralelizar sus algoritmos, por otra parte, las tecnologías actuales han permitido el uso de nuevos tipos de computación como sistemas embebidos y las GPUs que permiten solucionar de distintos enfoques y de manera optimizada problemas computacionales de diferente tipo.
    Para que un problema sea del tipo paralelizable se debe solucionar el cómo se comunicaran los procesos divididos dentro del cluster, para ello, se usan librerías especializadas como PVM (Parallel virtual machine) [7] o MPI (Message passage interface)[8], son usados generalmente por la comunidad científica. Por otra parte se puede usar un modelo de memoria compartida como lo es OpenMP[9].
  - **High Availability (Alta disponibilidad):** Máxima disponibilidad de los recursos del cluster, seguido de un rendimiento sostenido[14]. Son usados generalmente por el sector comercial.
  - **High Throughput (Alta eficiencia):** : según el proyecto de la universidad de Universidad Católica de Temuco [2] "Son clusters cuyo objetivo de diseño es el ejecutar la mayor cantidad de tareas en el menor tiempo posible. Existe independencia de datos entre las tareas individuales. El retardo entre los

[7]PVM : consiste en una librería de paso de mensajes con un ambiente de ejecución y administrador de tareas y recursos, usada comúnmente en clusters heterogéneos ha sido desarrollado desde 1989 por Üniversity of Tennessee, Oak Ridge National Laboratory and Emory University."

[8]MPI: libreria de paso de mensajes, define un estándar de comunicación para aplicaciones paralelas, desarrollado en 1992 en Williamsburg, Virginia.

[9]Open Multi-Processing: es un API que soporta múltiples plataformas de multiprocesammiento de memoria compartida.

nodos del cluster no es considerado un gran problema", su principal tarea es asegurar la resolución de problemas en el menor tiempo posible, minimizando el acaparamiento de recursos por una sola aplicación en una medida de tiempo considerable. Usados por la comunidad científica y militar.

- **Clasificación según su disposición y contribución al cluster:**

  La disponibilidad de los nodos es vital para lograr el objetivo común del cluster, no todos los cluster son dedicados ya que serlo implica un costo que no seria viable para proyectos de ámbito no lucrativo, ejemplo de ello es el proyecto SETI@home[15] que usa los ordenadores de usuarios a lo largo del mundo, descargando información de un servidor central y procesando estos datos durante sus tiempos muertos (tiempos en los cuales el ordenador no esta en uso), luego se envían los resultados de vuelta al servidor.

  Para clusters cooperativos donde es descentralizado su dominio y del tipo beowulf (se explicará más adelante), la disponibilidad está determinada por los tiempos que este nodo puede aportar (en base a su capacidad), de otro modo, si el ordenador esta realizando una tarea diferente a la de un calculo del cluster no sera prestado, lo que se denomina COW y no hará parte del poder de procesamiento. Por otra parte, algunos programas de balanceo de carga permiten el uso persistente de los ciclos de procesamiento (esta técnica se denomina NOW) asegurando mayor la persistencia de participación en todos los nodos[16].

- **Clasificación según tamaño:**[4] "Los **meta clusters**, son los clusters de clusters, es decir, los meta clusters son usualmente un grupo de clusters que están geográficamente distribuidos, ya sea a nivel nacional o al rededor del mundo, pero pueden ser tratados como un solo recurso por software especial y muy avanzado. Los meta clusters pueden ser también homogéneos o heterogéneos. Los meta clusters son llamados también super clusters. "

Clusters Beowulf:

Consiste en una tecnología que agrupa computadoras basadas en sistemas linux (además de otras herramientas) para formar parte de un sistema de computo mas potente o de un cluster, su primera aparición fue en el año de 1994 y ha sido una tecnología con mucha acogida debido a su facilidad de implementacion y coster relativamente bajos. Una de las diferencias principales entre Beowulf y un cluster de estaciones de trabajo (COW, cluster of workstations) es el hecho de que Beowulf se comporta más como una sola máquina que muchas estaciones de trabajo. Este sistema consiste de un nodo maestro y uno o más nodos

esclavos conectados a través de una red ethernet u otra topología de red. Esta construido con componentes de hardware comunes en el mercado, similar a cualquier PC capaz de ejecutar linux, adaptadores de Ethernet y switches estándares. Como no contiene elementos especiales, es totalmente reproducible [2].

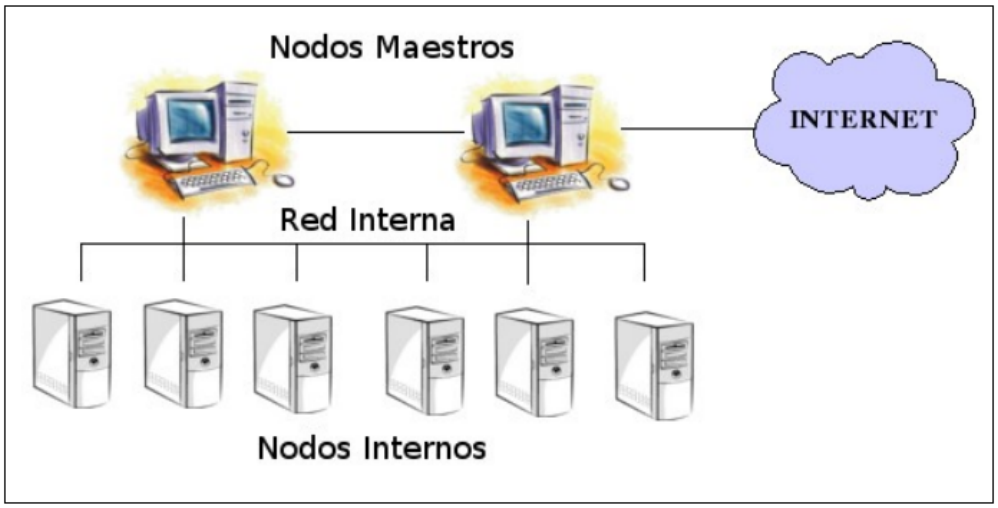


Figura 1.1: Arquitectura del cluster Beowulf [2]

1.4.4. Componentes de un cluster

Para poder cumplir su labor, un cluster necesita de componentes básicos para lograr su objetivo y optimizarlo, algunos de estos componentes son:

Arquitectura

la siguiente sección es tomada del proyecto de pregarado de Erick Samuel Gutiérrez perteneciente a la universidad Veracruzana de Mexico [14] " un cluster está formado por nodos de cómputo y una red de comunicación, dichos nodos puede ser un ordenador convencional o un sistema multiproceso con su propia memoria y sistema operativo. Cabe mencionar que estos nodos pueden estar incluidos en una sola cabina (llamada también rack"), o conectados por medio de una LAN.

Los componentes que forman un sistema de este tipo son.

- Un conjunto de ordenadores de altas prestaciones.

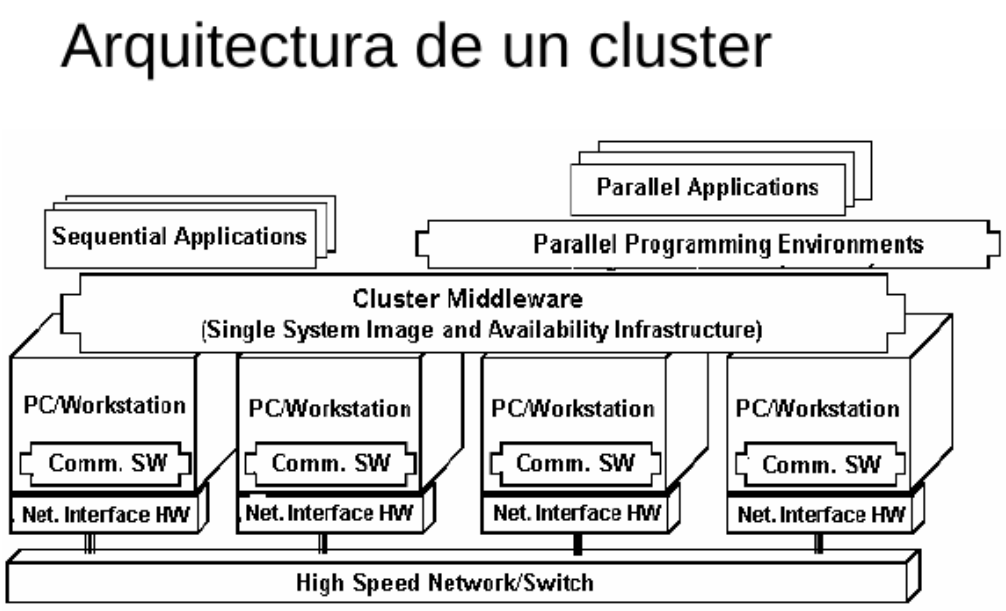


Figura 1.2: Arquitectura Paralela Basada en Clusters [3]

- Sistemas operativos basados en microkernel o estratificados.
- Redes de interconexión de altas prestaciones (myrinet, Gigabitt, Infiniband).
- Tarjetas de conexión a red de alta velocidad.
- Protocolos y servicios de comunicación a alta velocidad.
- Midleware que está compuesto por dos subniveles de software, el cual se explicará con más detalle en el siguiente punto.
- Entornos y herramientas de programación paralela, compiladores paralelos, Java, PVM, MPI, estos componentes se mostró en la figura anterior.

Midleware

En la computación basada en clusters el midleware es el software especializado que sirve de interfaz entre el sistema operativo y el hardware con la finalidad de:

- Proveer escalabilidad: De esta manera se le pueden unir mas componentes computacionales (nodos) de manera sencilla y fácil al sistema.

- Herramientas de optimización y rendimiento del sistema.
- Interfaz única de acceso al sistema.
- La imagen de sistema única (SSI Single System Image)[10] que ofrece a los usuarios un acceso unificado de todos los recursos del sistema, es un concepto que se basa en el fácil uso y administración de clusters especializados.
- Disponibilidad del sistema que permite servicios como puntos de chequeo, recuperación de fallos, soporte para tolerancia a fallos.

Es importante señalar que la imagen de sistema único proporciona al usuario una visión unificada del cluster como un único recurso o sistema de cómputo. Esta imagen está soportada precisamente por el nivel midleware que está entre el sistema operativo y las aplicaciones de usuario y tiene básicamente dos subsistemas que son:

- **Infraestructura de imagen del sistema único:** Esta presente en los nodos del sistema propiciando acceso invariante a los recursos disponibles del sistema.
- **Infraestructura de disponibilidad del sistema:** Posibilidad servicios propios del cluster como puntos de chequeo, recuperación automática de fallos y soporte para tolerancia a fallos.

Algunos objetivos de esta tecnología son:

- Presentar una completa transparencia al usuario de forma que este no tenga que preocuparse de detalles de bajo nivel en la implementación, ni de cómo gestionar el sistema para optimizar su rendimiento.
- Por otro lado, la escalabilidad del sistema, ya que los clusteres pueden ampliarse fácilmente añadiendo nuevos nodos, las aplicaciones deben ejecutarse de forma eficiente en un rango amplio de tamaños de máquinas.
- Por último, también es importante la disponibilidad del sistema para soportar las aplicaciones de los usuarios, esto, como se menciono anteriormente, por medio de técnicas de tolerancia a fallos y recuperación automática sin afectar a las aplicaciones de los usuarios.

[10]SSI: es un cluster de maquinas que aparentan ser un solo sistema, el concepto es aveces considerado como sinónimo de un sistema operativo distribuido.

Existen diversos tipos de midleware, entre los que se encuentran :

- **MOSIX:** Es una herramienta diseñada para realizar balanceo de cargas en el cluster de forma totalmente transparente de forma tal que los nodos del cluster se comportan como una sola máquina, y de esta manera incrementar el aprovechamiento de cada uno de los nodos. De igual manera también se puede observar el comportamiento del cluster, para esto se utiliza el monitor de carga de MOSIX llamado MON. Otra aplicación que vale la pena mencionar es MPS la cual da un reporte del status de procesos multicomputadores. Por otro lado cabe señalara que es un parche para el kernel de Linux que le da la capacidad de ejecución distribuida de procesos, lo que permite el desarrollo de una supercomputadora.
- **OpenMOSIX:** Es un parche para el kernel Linux que permite a varias máquinas actuar como un sistema multiprocesador grande. Lo que hace OpenMosix es balancear la carga de trabajo entre todos los nodos que forman el clúster : migra los procesos, independientemente de en qué nodo se han originado, al nodo con menos carga de trabajo. Su mayor ventaja es que las aplicaciones no tienen que estar programadas específicamente para OpenMosix ya que trabaja con aplicaciones normales (no paralelizadas), siendo su funcionamiento transparente al usuario. Pero tiene una limitación: sólo migra procesos que no usen memoria compartida, por lo que no migra procesos multi-hilo(característica que permite a una aplicación realizar varias tareas a la vez).

  OpenMosix está formado por los siguientes componentes:

  - Un parche para el kernel Linux.
  - Herramientas para la línea de comandos y para el entorno gráfico.
  - el script de inicio /etc/init.d/openmosix.

  El middleware también debe poder migrar procesos entre servidores con distintas finalidades:

  - **Balancear la carga:** Si un servidor está muy cargado de procesos y otro está ocioso, pueden transferirse procesos a este último para liberar de carga al primero y optimizar el funcionamiento.
  - **Mantenimiento de servidores:** Si hay procesos corriendo en un servidor que necesita mantenimiento o una actualización, es posible migrar los procesos a otro servidor y proceder a desconectar del clúster al primero.

- **Periodización de trabajos :** En caso de tener varios procesos corriendo en el clúster, pero uno de ellos de mayor importancia que los demás, puede migrarse este proceso a los servidores que posean más o mejores recursos para acelerar su procesamiento. ”

Otros factores que influyen en la arquitectura según Leonardi Martín Torralba Morales [4] son :

- **Almacenamiento compartido y almacenamiento distribuido:** Al diseñar un cluster también se debe pensar en la manera de almacenar la información dentro del esquema distribuido:

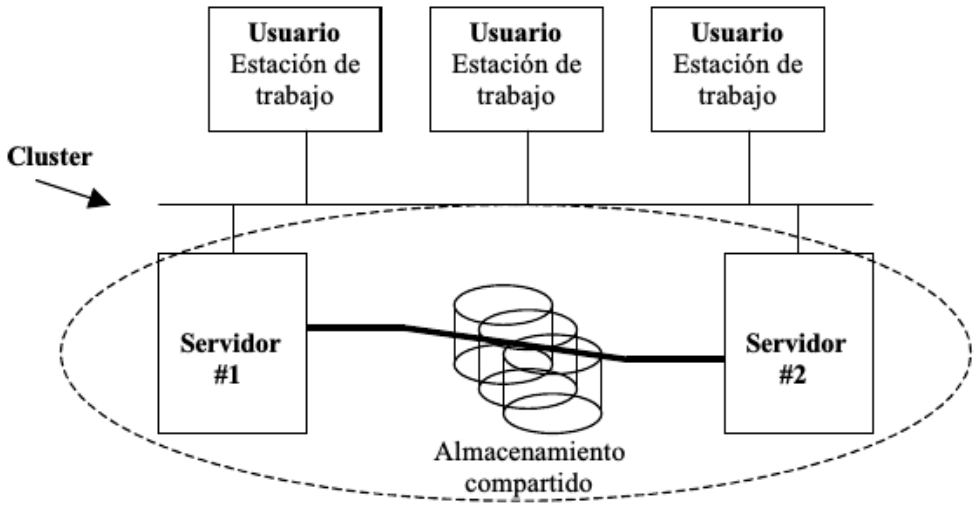


Figura 1.3: Cluster con almacenamiento compartido [4]

En el **almacenamiento compartido**, los nodos se conectan entre si para proveer un servicio de almacenamiento a un disco o arreglo de ellos (RAID), los nodos comparten este arreglo de discos gracias a un software especializado que se encarga de administrar el control de acceso que puede ser concurrente. Este tipo de almacenamiento compartido, es mucho mejor para sistemas que manejan gran cantidad de bases de datos y para sistemas que se encuentran en un mismo lugar.

Por otra parte el **almacenamiento distribuido**, cada nodo hace uso de su disco duro propio, así, cuando otro nodo necesita acceder a información de éste, lo realiza por medio de un mecanismo de paso de mensajes. Gracias a este proceso, se controla

la sincronización e información actual del sistema. Este tipo de almacenamiento se comporta eficientemente en sistemas de acceso constante e independiente a ciertos datos y también para sistemas dispersos o clusters de pocos nodos. Estas arquitecturas se pueden mezclar sin ningún problema. Uno de los protocolos mas populares para este tipo de sistema, es el NFS [11].

- **Memoria compartida y memoria distribuida:** Cuando se tiene un esquema de **memoria compartida** cada unidad de procesamiento tiene acceso a toda la memoria incrementando en gran medida el desempeño, esto se puede llegar a cabo debido a que cada espacio de direccionamiento de cada uno procesos participantes puede ser asignada y manipulada de manera independiente, evitando problemas como concurrencia o fallos, la memoria compartida (shared memory) es una forma implícita de comunicación entre procesos[17]. De otro modo, la **memoria compartida distribuida** [18] es definida como son sistemas que, mediante software, emulan semántica de memoria compartida sobre hardware que ofrece soporte sólo para comunicación mediante paso de mensajes. Este modelo permite utilizar una red de estaciones de trabajo de bajo costo como una maquina paralela con grandes capacidades de pro cesamiento y amplia escalabilidad,siendo a la vez fácil de programar. El objetivo principal de estos sistemas es permitir que un multicomputador pueda ejecutar programas escritos para un multiprocesador con memoria compartida.

  Por otro lado Erick [14] trae en contexto otros 2 tipos de arquitectura de memorias:

- **Arquitectura Non Uniform Memory Arquitecture(NUMA):** Además de la memoria compartida, cada procesador tiene una memoria local para guardar el programa y los datos no compartidos. En este caso disminuye el acceso a la red por parte de cada procesador haciendo más eficiente el acceso a memoria.

- **Arquitectura Cache Only Memory Arquitecture (COMA):** Solo quedan las memorias locales para cada procesador y cada procesador puede acceder a la memoria de cualquier otro, se le conoce como memoria virtual.

Balanceo de carga

El balanceo de carga es una técnica computacional, donde los elementos portadores de recursos se reparten el trabajo de la mejor manera posible generando una mejor prestación de

[11] Protocolo originalmente desarrollado por Sun Microsystems en 1984, permite a un usuario en una computadora del tipo cliente acceder a información a través de una red como si estíbese accediendo al disco local

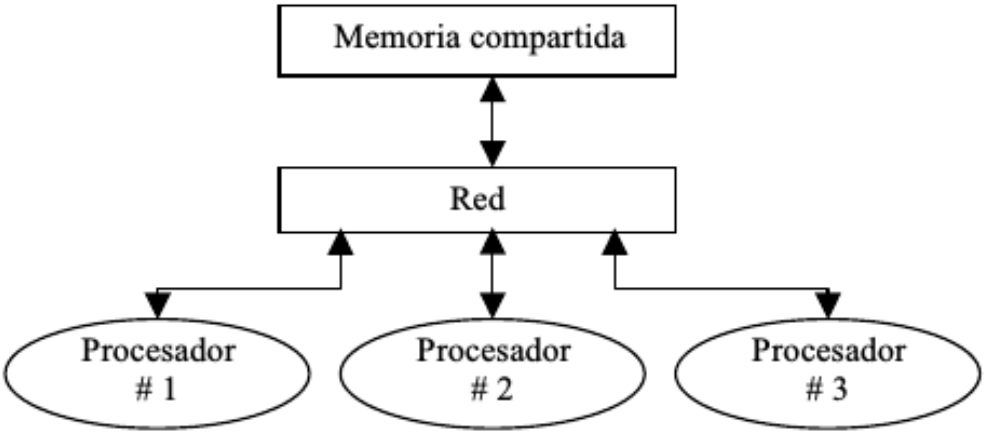


Figura 1.4: Memoria Compartida [4]

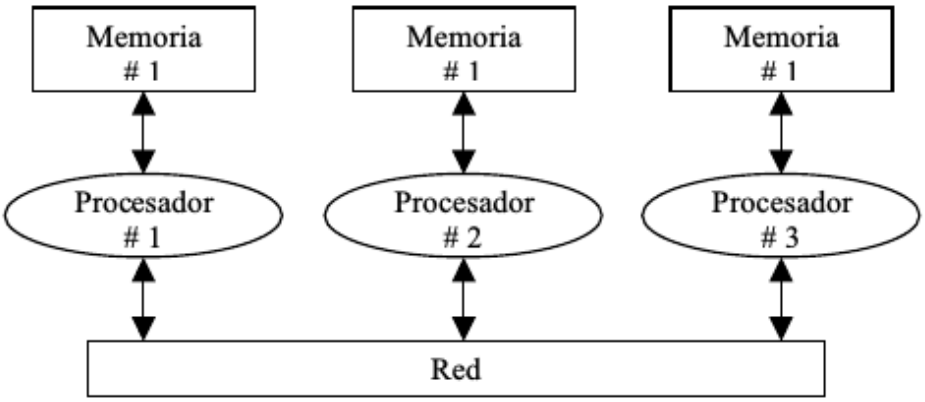


Figura 2.7-4 Memoria distribuida.

Figura 1.5: Memoria Distribuida [4]

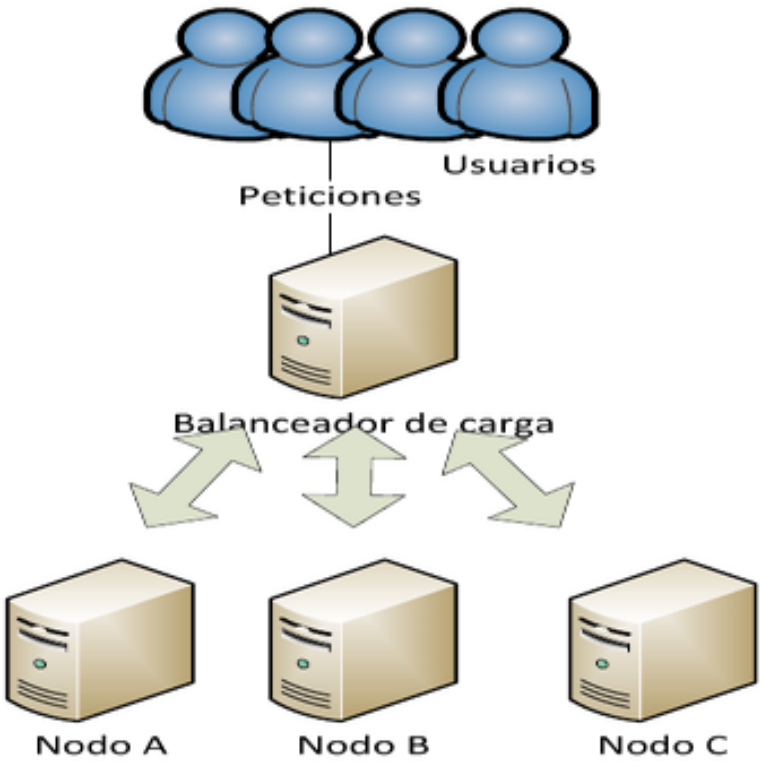


Figura 1.6: Diagrama cluster de balanceo de carga [5]

servicio, en términos de cluster el proyecto realizado por Alfred Gutiérrez Sanmiguel para la universidad Autónoma de Barcelona define el balanceo de carga de un cluster como [5] "Un cluster de balanceo de carga esta compuesto de varios nodos que actúan como la cara visible o front-end de este. Estos nodos se encargan de repartir las peticiones de servicio recibidas entro los diversos nodos existentes. "

La técnica de balanceo se divide en técnicas algorítmicas que reorganizan de manera mas precisa y equitativa el trabajo, encontrando procesos, rutas, mapeos, consistentes en cada componente que sea demandado por recursos. La aplicación de estas técnicas reduce el tiempo que le tarda a las aplicaciones en arrojar resultados. La administración de estas técnicas se hace mas engorrosa si los componentes del cluster son heterogéneos, ya que debe de mantener un estado de correctitud entre diferentes componentes de distinta naturaleza (procesadores, memorias, sistemas operativos , entre otros).

Existen tres clases de algoritmos de balanceo:

- **Balanceo centralizado:** En este tipo de balanceo, se delega un nodo predeterminado para ejecutar los algoritmos y así mantener el estado normal de todo el cluster, este método es poco flexible o escalable ya que para una demanda muy grande de recursos el nodo presentaría un cuello de botella dado que ese solo nodo intentaría suplir la sobre demanda de tareas.

- **Balanceo completamente distribuido:** Aquí cada nodo computacional posee información global del funcionamiento del sistema, intercambiando información con los nodos asociados a ellos, esto con el fin de hacer cambios de manera parcial.

- **Balanceo semi-distribuido:[19]** divide los procesadores en regiones, cada una con un algoritmo centralizado local. Otro algoritmo balancea la carga entre las regiones. El balanceo puede ser iniciado por envío o recibimiento. Si es balanceo iniciado por envío, un procesador con mucha carga envía trabajo a otros. Si es balanceo iniciado por recibimiento, un procesador con poca carga solicita trabajo de otros. Si la carga por procesador es baja o mediana, es mejor el balanceo iniciado por envío. Si la carga es alta se debe usar balanceo iniciado por recibimiento. De lo contrario, en ambos casos, se puede producir una fuerte migración innecesaria de tareas.

Estas técnicas están a cargo de programas complejos, uno de los más usados en implementaciones del tipo beowulf es el framework HTCondor[12] diseñado por la universidad de

[12]HTCondor es un software de código abierto de computación de high throughtput para la paralelización distribuida de grano grueso de las tareas computacionalmente intensivas. Se puede utilizar para manejar la carga de trabajo en un cluster dedicado de computadoras , y / o para aprovechar tiempos muertos de computadoras de escritorio - llamado ciclo de compactación.

Wisconsin-Madison. HTCondor se encarga de gestionar la carga de trabajos de cálculo intensivo, ofrece funcionalidades importantes como, mecanismo de cola de trabajos, política de planificación, esquema de prioridades, seguimiento de recursos y la gestión de estos. Los usuarios envían sus trabajos seriales o paralelos a la plataforma que disponga este framework, HTCondor administra estros trabajos de tal manera de que presenta un informe puntual y hace seguimiento cronológico a la ejecución de las tareas.

- **Recurso:** Componente de hardware o software que hacen parte del cluster.
- **Nodos:** Definición de los componentes computacionales que hacen parte del modelo del cluster, generalmente, son sistemas multi-procesador o estaciones de trabajo (workstations).

1.4.5. Ventajas y desventajas de los clusters

Tomado del proyecto de balanceo de carga [20].

Ventajas

- **Disponibilidad:** Capacidad para continuar operando ante sucesos previsibles.
- **Distribución en paralelo.**
- **Flexibilidad:** Los balanceadores de carga no están amarrados a ninguna arquitectura especifica, en lo que respecta a hardware.
- **Costos: El diseño y montaje requiere inversiones sumamente bajas comparadas con las alternativas de solución, las cuales son de costo elevado.**
- **Escalabilidad:** Capacidad para hacer frente a volúmenes de trabajo cada vez mayores, presentando así un nivel de rendimiento óptimo.
- **Acople:** Transferencia de información y todo tipo de servicio por internet de forma rápida, a bajo costo e ininterrumpible mente.
- **Velocidad:** Incremento del numero de transacciones o velocidad de respuesta ofrecido por los clusters de balanceo de carga.
- **Equilibrio:** Incremento de la confiabilidad y robustez ofrecido por los clusters de alta disponibilidad.

Desventajas:

- Empresas y entidades prefieren seguir utilizando el modelo cliente/servidor tradicional debido al espacio físico o a nuevos problemas que no se daban en la arquitectura tradicional.
- Espacio físico para el montaje de los clusters me mediana o gran embargadura.

1.4.6. Aplicación y prestaciones de cómputos en paralelo

El paralelismo es ampliamente utilizado en multitud de campos para el desarrollo de aplicaciones y estudio de problemas que requieren por su naturaleza trabajar con eficiencia además de tiempos aceptables para la labor académica en cuestión, este tipo de paradigma, divide los problemas (algoritmos) en partes más pequeñas que se resuelven de manera simultanea, Solventando la problemática creciente de la arquitectura monolítica de las computadoras en el pasado. Por otro lado favorece la explotación de recursos en el proceso de cómputo, aumentando el rendimiento de un equipo computacional. Las dos principales características de la computación paralelas son:

- Memoria compartida.
- Paso de mensajes.

Tipos de paralelismo:

- **Funcional:** Se refiere a tareas diferentes que se pueden realizar en paralelo, es inherente en todas las aplicaciones y generalmente tiene un grado de paralelismo bajo.
- **De datos:** Se enfoca a que una misma tarea se ejecuta en paralelo sobre un conjunto de datos, se tiene replicas del mismo programa trabajando sobre partes distintas de los datos, tiene un control centralizado o distribuido, en este caso el grado de paralelismo es muy alto, sin embargo, no está presente en todas las aplicaciones.

El proyecto [14] realizado en la universidad de Veracruz (México) destaca los campos en dónde el paralelismo contribuye de manera significativa:

- **Modelado predictivo y simulación**: Se realiza mediante extensos experimentos de simulación por computador que con frecuencia acarrean computaciones a gran escala para obtener la precisión y el tiempo de respuesta deseado. Entre estos modelados destacamos la previsión meteorológica numérica y la oceanografía

- **El desarrollo industrial**: también reclama el uso de computadores para progresar en el diseño y automatización de proyectos de ingeniería, la inteligencia artificial y la detección remota de los recursos terrestres. En este campo destacamos: la inteligencia artificial y automatización (procesamiento de imágenes, reconocimiento de patrones, visión por computador a, comprensión del habla, deducción automática, robótica inteligente, sistemas expertos por computadoras, ingeniería del conocimiento, etc.)
- **Investigación médica**: En el área médica las computadoras rápidas son necesarias en tomografía asistida, diseño de corazones artificiales, diagnóstico hepático, estimación de daños cerebrales y estudios de ingeniería genética."

Rendimiento computación paralela

Definición según una publicación del profesor Javier Bastida Ibáñez de la universidad de Valladolid [21] "La velocidad con que operan las computadoras se mide por el número de operaciones básicas que pueden realizar por unidad de tiempo." Una de las tantas unidades para medir la velocidad de las operaciones realizadas por una computadora es la llamada **MIPS** (1 millón de instrucciones sobre segundo.), pero como lo afirma Ibáñez [21] esta unidad tiene el inconveniente de que no es homogénea debido a que las instrucciones de un procesador pueden ser mucho más potentes que la de otra computadora. Dentro de las más usadas esta la unidad de **MFLOPS** (1 millón de instrucciones de punto flotante sobre segundo). La unidad generalmente usada en los famosos benchmarks es la **VUP** (VAX unit performance). Una medida bastante utilizada para medir la eficiencia de un sistema, y que sólo afecta parámetros arquitectónicos del mismo sin entrar en la calidad de la tecnología empleada, es el CPI (ciclos por instrucción) que es el número medio de ciclos de reloj necesario para ejecutar una instrucción.

$$CPI = \frac{\text{Número de ciclos de reloj consumidos}}{\text{Número instrucciones ejecutadas}} \tag{1.1}$$

Para un programa de n instrucciones, el tiempo de ejecución estará dado por:

$$t = CPI \cdot n \cdot \frac{1}{f} \tag{1.2}$$

donde f es la frecuencia de reloj, y, por lo tanto, su inverso es el tiempo de ciclo de reloj.

Para medir la eficiencia de un sistema paralelo, se compara el tiempo tomado en el sistema paralelo (N) y el tiempo tomado en solo procesador, a esto se le conoce como **aceleración** (speed up) y esta definido por:

$$S(N) = \frac{t(1)}{t(N)} \tag{1.3}$$

donde *t(1)* es el tiempo que se tardo en ejecutar un proceso en el sistema de un solo procesador y *t(N)* es el tiempo empleado para ejecutar un proceso en el sistema paralelo de N unidades de cómputo. En condiciones ideales, $t(N) = \frac{1}{N}$ dando como resultado que la ganancia de la velocidad (usando ecuación 1.3):

$$S(N)_{\text{ideal}} = \frac{t(1)}{t(N)} = \frac{1}{1/N} = N \tag{1.4}$$

Por otro lado, si se quiere medir el rendimiento del sistema, se compara la ganancia de la velocidad brindada por la ecuación 1.4, dando como resultado la **eficiencia** y estará dada por:

$$E(N) = \frac{S(N)}{S(N)_{\text{ideal}}} = \frac{S(N)}{N} = \frac{t(1)/t(N)}{N} = \frac{t(1)}{Nt(N)} \tag{1.5}$$

Que indica la medida en que se aprovechan los recursos.

1.5. Marco referencial

1.5.1. Marco de antecedentes

Al comienzo del auge computacional, los ordenadores actuaban como componentes separados de propósitos específicos ejecutando rutinariamente programas almacenados en su memoria interna. Ya en los años 60's la comunicación entre computadoras fue implementada para compartir ficheros (sin ningún trasfondo puntual). El primer RFC (RFC 1) [13] fue una propuesta de como máquinas pueden intercambiar información con otras. La primera aplicación de una interconexión entre nodos fue el e-mail en 1972 usando ARPANET[14]. ARPANET dio como base a los sistemas ditribuidos por ende a los grids y clusters computacionales[22]. Según el proyecto[14] los componentes históricos detrás de los sistemas computacionales del tipo cluster son :
El primer producto comercial de tipo cluster fue ARCnet, desarrollada en 1977 por Datapoint

[13]Request For commetns, son especificaciones propuestas por ingenieros de internet que invitan a realizar comentarios públicos.

[14]fue una red de computadoras usadas para la comunicación entre las diferentes instituciones académicas y estatales en USA

pero no tuvieron un éxito comercial y los clusters no consiguieron tener éxito hasta que en 1984 VAX cluster produjera el sistema operativo VAX/VMS.Estos productos no solo apoyan a la computación paralela, sino que también comparten los archivos y dispositivos periféricos. La idea original de estos sistemas era proporcionar las ventajas del procesamiento paralelo, al tiempo que se mantiene la fiabilidad de los datos y el carácter singular.VAX cluster y VMS cluster están aún disponibles en los sistemas de HP OpenVMS corriendo en sistemas Itanium y ALpha.
Otros dos principios comerciales de clusters notables fueron el Thandem Himalaya (alrededor de 1994 con productos de alta disponibilidad) y el IBM S/390 Parallel Sysplex también alrededor de 1994, este con fines empresariales. Por otro lado se encuentra el software de Parallel Virtual Machine (PVM). Este software de fuente abierta basado en comunicaciones TCP/IP permitió la creación de un superordenador virtual (un cluster HPC) realizada desde cualquiera de los sistemas conectados TCP/IP[15]. De forma libre los clusters heterogéneos han constituido la cima de este modelo logrando aumentar rápidamente en FLOPS globalmente y superando con creces la disponibilidad incluso de los más caros superordenadores. El empleo de PC, PVM y redes de bajo costo dio como resultado en 1993 la creación de un cluster desarrollado por la NASA, denominado Discover. Otros hitos relevantes[9]: En 1994, T. Sterling y D. Becker, trabajando en CESDIS (Center of Excellence in Space Data and Information Sciences) bajo el patrocinio del Proyecto de la Tierra y Ciencias del Espacio (ESS), construyeron un cluster de computadoras que consistía de 16 procesadores 486DX4, usando una red Ethernet a 10Mbps, con un costo de 40,000 dolares. El rendimiento del cluster era de 3.2 GFLOPS.Ello s llamaron a su sistema Beowulf, un éxito inmediato, y su idea de proporcionar sistemas en base a COTS (Components Of The Shelve) para satisfacer requisitos de cómputo específicos, se propagó rápidamente a través de la NASA y en las comunidades académicas y de investigación. En la actualidad, muchos clusters todavía son diseñados, ensamblados y configurados por sus propios operadores; sin embargo, existe la opción de adquirir clusters prefabricado.

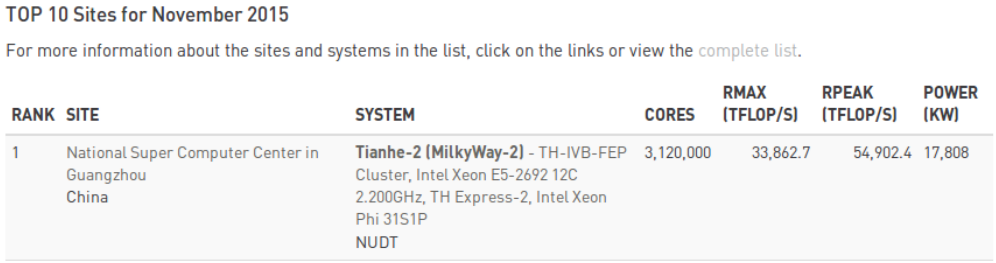
Algunos ejemplos de clusters en la actualidad:

Para dar detalle del poder tecnológico que implica la disposición de un cluster computacional, se lista la maquina mas poderosa actualmente fabricada en el mundo [16]:

Por otro lado, las unidades de computación que pueden prestarse para la elaboración de un cluster son infinitas, como es el caso de las consolas de videojuegos, en Noviembre de 2010 “the Air Force Research Laboratory” creo un poderoso supercomputador conectando 1,760 Sony

[15]Transmission Control Protocol/Internet Protocol, lenguaje de comunicación básico o protocolo de internet.

[16]Top de super computadoras, [fecha de consulta: 15 Noviembre 2015] ,[en línea] http://www.top500.org



TOP 10 Sites for November 2015

For more information about the sites and systems in the list, click on the links or view the complete list.

RANK	SITE	SYSTEM	CORES	RMAX (TFLOP/S)	RPEAK (TFLOP/S)	POWER (KW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808

Figura 1.7: Supercomputadora Tianhe-2 (MilkyWay-2) [6]

PS3s las cuales incluían separadamente 168 GPU's y 84 servidores coordinados en un array en paralelo con capacidad de realizar 500 Teraflops (trillones de operaciones coma flotante por segundo) (500 TFLOPS).[17]. Otro caso interesante es el que propone PUZZLECLUSTER [18], básicamente es un modulo que permite conectar teléfonos modulares (próximos a salir en el mercado) para que aporte poder computacional.

Cifras de otros proyectos

La siguiente sección ha sido tomada del proyecto de grado de Bernardo [1], dentro de su investigación referente a cifras del uso de clusters a nivel nacional.

[17]Supercomputadora PS3, [fecha de consulta: 16 Noviembre 2015] ,[en línea]`http://www.wpafb.af.mil/news/story.asp?id=123231285`

[18]Proyecto Puzzlecluster ,[fecha de consulta: 16 Noviembre 2015] ,[en línea]: `http://www.hpcwire.com/2015/01/27/new-purpose-old-smartphones-cluster-computing/`

Tabla 1.1: Cifra proyectos [1]

Descripción
Nombre del proyecto : Perfiles HMM y genoma de P. infestans. **Área:** Biología. Una de las herramientas más utilizadas para análisis de secuencias biológicas en Hmmer. Con el Laboratorio de Micología y Fitopatología de la Universidad de los Andes, se desarrolló una solución para portar hmmer a una infraestructura distribuida y así poder aprovechar de manera oportunista las salas de cómputo del Departamento de Ingeniería de Sistemas y Computación. Se implementó un cluster virtual basado en Condor (Anteriro version de HTCondor) y se desarrolló una aplicación para permitir la paralelización de las tareas. El cluster fue desplegado en la salas Wuaira del Departamento de ISC la cual consta de 70 máquinas al servicio de los estudiantes. La aplicación requirió la ejecución de 4200 trabajos y los resultados de estas ejecuciones se ilustran en la tabla de resultados.
Nombre del proyecto : JG2A: A Grid-Enabled Object-Oriented Framework for Developing Genetic Algorithms. **Área:** Ingeniería industrial. Java Genetic Algorithm (JGA) es un framework flexible orientado a objetos para el prototipado de algoritmos evolutivos. Es una nueva generación de JGA que explota el paralelismo de los algoritmos genéticos en dos caminos: primero, permite la ejecución paralela de un largo conjunto de instancias (paralización de instancias); y segundo, proporciona paralización de la evaluación de la población. En este proyecto se verificó la paralización de instancias utilizando diferentes parámetros en experimentos de problemas de enrutamiento de vehículos y problemas de diseño de rutas. La paralización de la evaluación de la población es particularmente útil para problemas de optimización donde la función de evaluación objetivo incluye un análisis de la simulación con eventos discretos y elementos finitos. JG2A puede ser desplegado en ambientes computacionales heterogéneos como es el caso de infraestructuras Grid Computing basados en Globus y Condor como el administrador de recursos locales.

Tabla 1.2: Resultados Proyecto Perfiles HMM y genoma de P. infestans. [1]

Infraestructura	Numero de trabajos	Tiempo promedio de CPU X trabajo (min)	Numero de CPUs	Tiempo total (días)
PC	4200	195	2	284,4
Inf. Oportunista	4200	215	140	4,5

Tabla 1.3: Resultados proyecto JG2A [1]

Infraestructura	Numero de trabajos	Tiempo promedio de CPU X trabajo (min)	Numero de CPUs	Tiempo total (días)
PC	2880	50	2	50,1
Inf. Oportunista	2880	52	70	1,5

Herramientas de configuración y administración un cluster

Algunas herramientas (mostradas en la figura 1.8) para realizar configuración y administración en clusters (de ámbito beowulf) son:

- **Pirahna:** Piranha es uno de los productos de clustering de Red Hat Inc; incluye el código del kernel IPVS, herramientas de monitoreo de clusters y herramientas de configuración de cluster basado en la web. La herramienta de monitoreo piranha tiene dos características principales: 1- Heartbeating entre balenceadores de carga de activos y de reserva. 2 - Comprobación de disponibilidad de los servicios en cada uno de los servidores reales.

- **Open Moxis:** Proyecto oficialmente finalizado el primero de marzo del 2008. (link). Fue un sistema de gestión del cluster gratuito que proporciona imagen de sistema único (SSI), por ejemplo, distribución del trabajo automática entre nodos. Permitió a los procesos del programa (no hilos) migrar a las máquinas de la red del nodo que serían capaces de ejecutar ese proceso más rápido (migración de procesos). Es particularmente útil para ejecutar en paralelo e intensivo de entrada / salida (I / O). Fue lanzado como un parche de kernel de Linux, pero también estaba disponible en Live CDs especializados. desarrollo openMosix ha sido terminado por sus desarrolladores, pero el proyecto LinuxPMI continuo desarrollo del antiguo código openMosix [19] .

- **Ultra Monkey:** [20] Ultra Monkey es un proyecto para la creación de servicios de red equilibradas y de alta disponibilidad. Un ejemplo de ello es un grupo de servidores web que aparecen como un único servidor web para los usuarios finales. El servicio puede ser prestado para los usuarios finales en todo el mundo conectado a través de internet,

[19]Definición OpenMoxis, [fecha de consulta: 18 Noviembre 2015] ,[en línea]`https://en.wikipedia.org/wiki/OpenMosix`

[20]Qué es ultramonkey?,[fecha de consulta: 18 Noviembre 2015] ,[en línea] http://www.ultramonkey.org/

	Descripción	Alternativas
Software para clustering	Alta disponibilidad	Pirahna Open Mosix
	Balanceo de carga	Ultra Monkey Pirahna Open Mosix
	Configuración e instalación	FAI SIS System Image System Installer System Configuration
	Monitorización e Instalación	LVSmonn, Sincopt, Fsync, Ghosts, y Pconsole.
	Monitorización	Mon Heartbeat Fake Coda Ganglia

Figura 1.8: Algunas herramientas administrativas para clusters [7]

o para usuarios empresariales conectados a través de una intranet. Ultra Monkey hace uso del sistema operativo Linux para proporcionar una solución flexible que se puede adaptar a una amplia gama de necesidades. Desde pequeños grupos de dos nodos a los grandes sistemas que sirven a miles de conexiones por segundo.

- **FAI (Fully Automatic Installation):** [21] FAI es un sistema no interactivo para instalar, personalizar y gestionar los sistemas Linux y configuraciones de software en los equipos, así como máquinas virtuales y entornos chroot, desde pequeñas redes de infraestructuras a gran escala, como clusters y entornos de nube. Es una herramienta para el despliegue masivo desatendida de Linux.

- **LVSmon:** [22] LVSmon es un demonio de monitoreo de clusters escrita originalmente con la intención de reemplazar a las herramientas como ldirectord y mon en relación con el mantenimiento de tablas LVS . LVSMon es segura, confiable y escalable. LVSmon nunca dejara el sistema. Nunca le dará una salida falsa . Ni permitirá que las personas (usuarios finales) se hagan cargo del sistema. Utiliza muy poca carga de CPU, tiene un tamaño reducido y escala mejor que la mayoría de aplicaciones de este tipo, incluso teniendo en cuenta el hecho de que se ejecuta en un solo hilo.

- **Pconsole:** [23] PCONSOLE es una herramienta administrativa para trabajar con grupos de máquinas. PCONSOLE le permite conectar cada nodo del cluster al mismo tiempo, además usted puede escribir sus comandos administrativos en una ventana especializada que 'multiplica' la entrada a cada una de las conexiones que se han abierto.

- **Ganglia:** [24] Es un sistema de control distribuido escalable para los sistemas de computación de alto rendimiento, como los clusters y Grids. Se basa en un diseño jerárquico dirigido a federaciones de clusters. Aprovecha las tecnologías ampliamente utilizado como XML para la representación de datos, XDR para el transporte de datos compacto, portátil y RRDtool para el almacenamiento de datos y visualización. Utiliza diseñado cuidadosamente las estructuras de datos y algoritmos para lograr muy bajos los gastos generales por nodo y alta concurrencia

- **Heartbeat:** [7] Funciona enviando periódicamente un paquete, que si no llegara, indicaría que un servidor no está disponible, por lo tanto se sabe que el servidor ha

[21] FAI web page, [fecha de consulta: 18 Noviembre 2015] ,[en línea] http://fai-project.org/

[22] LVSmon definición ,[fecha de consulta: 18 Noviembre 2015] ,[en línea] http://www.scaramanga.co.uk/lvsmon/

[23] Definición Pconsole,[fecha de consulta: 18 Noviembre 2015] ,[en línea] http://www.heiho.net/pconsole/

[24] Definición Ganglia ,[fecha de consulta: 20 Noviembre 2015] ,[en línea] http://ganglia.sourceforge.net/

caído y se toman las medidas necesarias. Se recomienda el uso de puertos serie puesto que están aislados de las tarjetas de red. Soporta múltiples direcciones IP y un modelo servidor primario/secundario. Los mensajes de Heartbeat se envían por todas las líneas de comunicación a la vez, de esta manera, si una línea de apoyo cae, se avisará de ese problema antes de que la línea principal caiga y no haya una línea secundaria para continuar el servicio. Heartbeat también se preocupa por la seguridad, permitiendo firmar los paquetes con CRC de 32 bits, MD5 y SHA1.

Servicios Clustering

Debido al auge tecnológico, la demanda de plataformas tecnológicas para computar algoritmos paralelos ha venido en creciente apogeo. A continuación se presentara una lista aplicaciones comerciales que brindan este servicio.

- **Amazon:** Amazon Elastic Compute Cloud (Amazon EC2) es un servicio web que proporciona capacidad informática con tamaño modificable en la nube. Está diseñado para facilitar a los desarrolladores la informática en la nube escalable basada en web. Ésta plataforma funciona bajo StarCluster que englosa un conjunto de herramientas de código libre para clusters, StarCluster trabaja bajo licencia LGPL.

- **Dell:** [25] Soluciones de Dell ™ HPC utilizan bloques de construcción mejores en su clase, probados y validados por los ingenieros de Dell, para que pueda implementar y empezar a utilizar sus recursos de TI rápidamente. Nuestros arquitectos de soluciones trabajan duro para entender el entorno en términos de requisitos funcionales, técnicos y operativos, por lo que pueden integrar soluciones HPC completos que se pueden replicar y empleados como arquitecturas de racimo de clase mundial.

- **HP:** [26] HP Cluster Platforms, Estos sistemas combinan la flexibilidad de una solución personalizada con la simplicidad, la fiabilidad y el valor de un producto preconfigurado, incorporado de fábrica. Características : 1 Amplia selección de procesadores, interconexiones del clúster y middleware. 2 Fábrica integrado y probado, instalación desde el sitio web. 3 Elección de los estilos de embalaje: diseño modular denso o ampliable hasta 1024 nodos. 4 Con el respaldo de garantía y soporte de HP, y uniformemente

[25] Dell HPC web page, [fecha de consulta: 22 Noviembre 2015] ,[en línea] https://www.dell.com/learn/us/en/555/high-performance-computing-strategy

[26] HP cluster site,[fecha de consulta: 22 Noviembre 2015] ,[en línea] http://www8.hp.com/us/en/products/servers/scalable-systems/clusterplatform.html

construido, con especificaciones internacionales. 5 selección de software integral, probado y verificado por HP y / o sus socios. 6 Servicios de despliegue rápido, incluyendo la instalación de software opcional, formación one-site y apoyo a la ejecución. 7 opción Nvidia Tesla GPU - hasta 6 GPUs por 2u.

- **IBM:** [27] IBM Platform HPC, proporciona un conjunto completo de capacidades de computación (HPC) de gestión del rendimiento técnico y alta en un solo producto. El rico conjunto de características out-of-the-box a los administradores de TI y los usuarios reducir la complejidad de la implementación, mejorar la gestión y el uso de su entorno informático y la mejora de su tiempo a los resultados sin olvidar la reducción los costos.

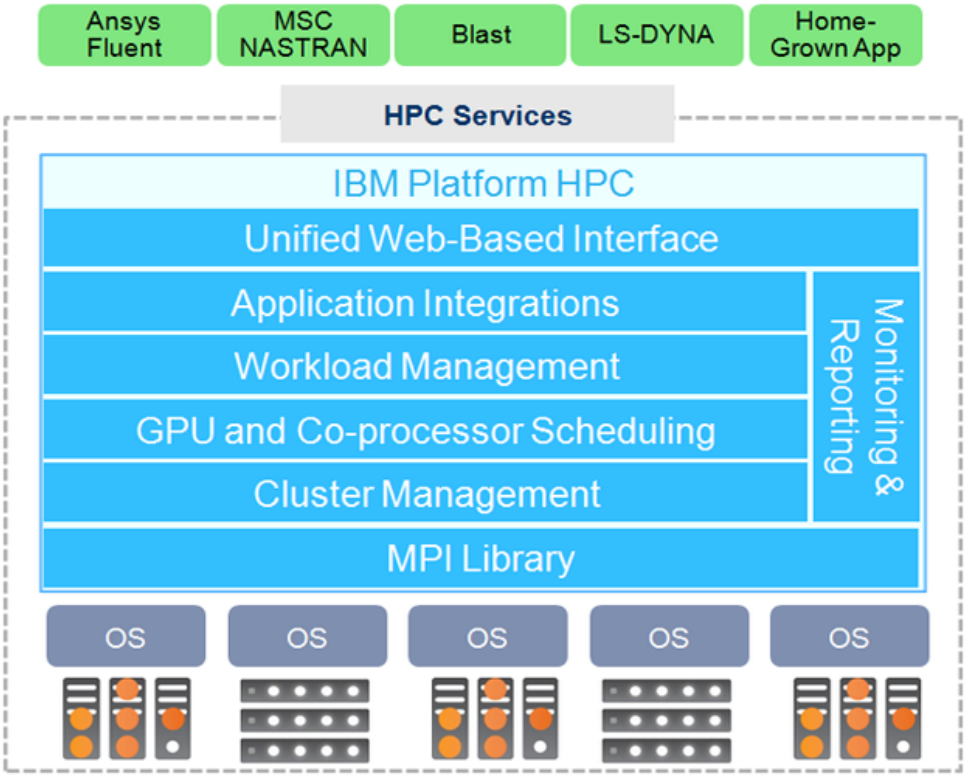


Figura 1.9: Componentes servicio de cluster por IBM [8]

[27]IBM platform computing, [fecha de consulta: 22 Noviembre 2015] ,[en línea] http://www-03.ibm.com/systems/platformcomputing/products/hpc/

1.5.2. Marco Teórico

Ley de Amdahl

La ley de Amdahl [28] [21] (1967) pone un límite superior a la ganancia en velocidad, y por tanto también a la eficiencia, de un sistema paralelo, donde que los procesos suelen tener partes que no pueden ser ejecutadas en paralelo, sino sólo de forma secuencial pura. Del tiempo total de ejecución del proceso, llamaremos s a la parte que no puede ser paralelizada y p al resto (paralelizable). Suponiendo condiciones ideales en la parte paralelizable del proceso p, tendremos que los tiempos mínimos de ejecución del proceso con un solo procesador y en el sistema paralelo con N procesadores serán:

$$t(1) = s + p \tag{1.6}$$

$$t(N) = s + \frac{p}{N} \tag{1.7}$$

Por lo tanto, la ganancia de velocidad máxima estará dada por:

$$S(N)_{\text{máx}} = \frac{s+p}{s+\frac{p}{N}} = \frac{1}{\frac{s}{s+p} + \frac{p}{N(s+p)}} \tag{1.8}$$

Además la fracción de tiempo no paralelizable f esta dada por:

$$f = \frac{s}{s+p} \tag{1.9}$$

Reemplazando en la ecuación 1.8 :

$$S(N)_{\text{máx}} = \frac{1}{f + \frac{1-f}{N}} = \frac{N}{Nf + 1 - f} = \frac{N}{1 + (N-1)f} \tag{1.10}$$

Entonces la ganancia máxima tenderá a :

$$S(N)_{\text{límite}} = \lim_{N\to\infty} \frac{N}{1 + (N-1)} = \frac{1}{f} \tag{1.11}$$

[28] Estadounidense de origen noruego, arquitecto computacional y una de las personalidades más importante y excéntricas en la historia de la informática y la computación. Fundó cuatro compañías tecnológicas en diferentes ámbitos, la mayoría de las cuales, se fundaron con el objetivo de competir con otras compañías donde él mismo había trabajado o incluso creado anteriormente, y que ya había abandonado.

Esta limitación de la fracción no paralelizable f recibe el nombre de **cuello de botella secuencial** de un proceso. Ahora, gracias a el limite anterior sabremos que (en la ecuación de eficiencia 1.5) estará dada por :

$$E(N)_{\text{máx}} = \frac{S(N)_{\text{máx}}}{N} = \frac{1}{1 + (N-1)f} \tag{1.12}$$

Si el numero de procesadores crece indefinidamente, la eficiencia máxima tenderá a:

$$E(N)_{\text{lím}} = \lim_{N \to \infty} \frac{1}{1 + (N-1)f} = 0 \tag{1.13}$$

"Por lo dicho hasta ahora, la ley de Amdahl es bastante pesimista. Es necesario señalar, sin embargo, que esta ley no tiene en cuenta algunos beneficios de los sistemas paralelos. Por ejemplo, un sistema con varios procesadores que tengan memorias caché propias puede hacer que cada procesador almacene en su caché los datos que está utilizando; sin embargo, un procesador solo, con una memoria caché del mismo tamaño, no podrá hacerlo con todos los datos. Esto puede causar que, en estas condiciones, la velocidad de un sistema paralelo con N procesadores pueda aumentar por encima de N, lo que aumentaría la cota superior propuesta por la ley de Amdahl. Una realidad que sí transmite la ley de Amdahl es que el **rendimiento no aumenta por incrementarse indefinidamente el número de procesadores**."[21]

Ley de Gustafson

La ley de Gustafson [29] (1988) tiene un enfoque diferente al establecido en la ley de Amdahl, pasando a ser una visión mucho mas positiva en cuanto a las ventajas del procesamiento paralelo, esta ley afirma que el volumen del problema no es independiente del número de procesadores, debido a que cuanto mayor sea el numero de procesadores se pueden afrontar problemas de mayor tamaño. La Ley de Amdahl se basa en una carga de trabajo o tamaño de entrada prefijados. Esto implica que la parte secuencial de un programa no cambia con respecto al número de procesadores de la máquina, sin embargo, la parte paralelizable es uniformemente distribuida en el número de procesadores [23].

En muchos casos, cuando el volumen del problema aumenta, solo lo hace su parte paralela, dejando que el cuello de botella secuencial tienda a cero. Algunos problemas tienden a aumentar su tamaño debido a la reducción del poder computacional o el aumento de la

[29] John L. Gustafson (19 de enero , 1955) es un científico estadounidense informático y empresario , principalmente conocido por su trabajo en High Performance Computing (HPC) tales como la invención de la ley de Gustafson , al presentar el primer cluster informático comercial

complejidad espacial o temporal ni hablar de problemas que crecen de dimensionalidad. Si el numero de procesadores aumenta indefinidamente tendremos :

$$\lim_{N\to\infty} f = \lim_{N\to\infty} \frac{s}{s+Np} = 0 \tag{1.14}$$

Donde s es la parte secuencial y p la parte paralela. Luego, se aumenta proporcionalmente el numero de procesadores (parte paralela). En un solo procesador 1.15 y en un sistema paralelo 1.16 :

$$t(1) = s + Np \tag{1.15}$$

$$t(N) = s + p \tag{1.16}$$

Por lo tanto, la ganancia de velocidad es:

$$S = \frac{t(1)}{t(N)} = \frac{s+Np}{s+p} \tag{1.17}$$

Y referente a la fracción no paralelizada (dada por la ecuación 1.9) queda:

$$S = f + N(1-f) = N - (N-1)f \tag{1.18}$$

MPI

Message Passing Interface es uno de los protocolos ampliamente usado en en la computación paralela, teniendo como premisa el paso de mensajes en su desarrollo, haciendo que las aplicaciones sean eficientes y portables. La primera versión aparece en mayo de 1994. MPI ofrece al programador interesado en el paralelismo un conjunto de especificaciones y funcionalidades técnicas ideales para el desarrollo de aplicaciones distribuidas. MPI realiza una conversión de datos heterogéneos como parte transparente de sus servicios, por medio de la definicion de tipos de datos específicos para los procesos de comunicación.

- **Fundamentos de MPI [9] :** El número de procesos requeridos se asigna antes de la ejecución de la aplicación en mpi, sin necesidad crearlos en tiempo de ejecución. Para su identificación a cada proceso se le asigna una variable denominada *rank*. Es así como *rank*, determina el control de ejecución del programa determinando que proceso ejecuta determinado código. En MPI la colección de procesos se define como *comunicator*, los cuales pueden enviar mensajes los unos a los otros. el comunicador más básico se define "MPI-COMM-WORLD" y se define mediante un macro del lenguaje C.

Hay 4 clases de llamadas en MPI:

- Llamadas de inicialización, administración y de terminación de los procesos de comunicaciones.
- Llamadas utilizadas para la transferencia de datos entre un par de procesos.
- Llamadas utilizadas para la transferencia de datos entre varios procesos.
- Llamadas utilizadas para crear tipos de datos definidos por el usuario.

La primera clase de llamadas permiten inicializar la librería de paso de mensajes, identificar el número de procesos (size) y el rango de los procesos (rank). La segunda clase de llamadas incluye operaciones de comunicación punto a punto, para diferentes tipos de actividades de envío y recepción. La tercera clase de llamadas son conocidas como operaciones grupales, que proveen operaciones de comunicaciones entre grupos de procesos. La última clase de llamadas provee flexibilidad en la construcción de estructuras de datos complejos. Formato de un mensaje en MPI:

Figura 1.10: Formato de un mensaje de MPI [9]

El **cuerpo del mensaje** contiene los datos que serán enviados, contiene tres piezas de información:

- **buffer:** Localidad de memoria donde se encuentran los datos de salida o almacenan los datos de entrada.
- **Tipo de dato:** Indica los tipos de datos que se envían en el mensaje.
- **El count:** Es un número de secuencia que junto a los tipos de datos permiten al usuario agrupar ítem de datos de un mismo tipo en un solo mensaje.

La **envoltura** indica el proceso fuente y destino, normalmente contiene la dirección destino conjunto con la fuente, e información que se necesite, consta de 4 partes:

- **Fuente:** Identifica al proceso transmisor.

- **Destino:** Identifica al proceso receptor.
- **Comunicador:** Especifica el grupo de procesos a los cuales pertenecen la fuente y el destino.
- **Etiqueta:** Permite clasificar el mensaje. Es un entero definido por el programador que puede ser usado para distinguir los procesos.

Taxonomía de Flynn

La taxonomía de Flynn es una clasificación de arquitecturas de computadores propuesta por Michael J. Flynn en 1972. Flynn clasifico las arquitecturas según el flujo de datos e instrucciones.

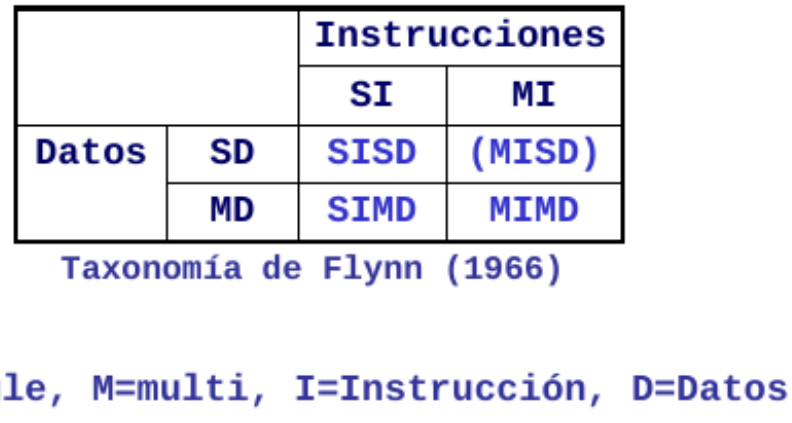


Figura 1.11: Taxonomía de Flynn [10]

- **Single instruction Stream, Single Data Stream (SISD):** Un solo flujo de instrucciones y un solo flujo de datos. Modelo convencional de Von Neumann [30]. El modelo de Von Neumman describe que un procesador es capaz de realizar acciones secuencialmente, controladas por un programa el cual se encuentra almacenado en una memoria conectada al procesador. Este hardware esta diseñado para dar soporte al procesamiento secuencial clásico, basado en el intercambio de datos entre memoria y registros del procesador, y la realización de operaciones aritméticas en ellos.[10]

[30] fue un matemático húngaro-estadounidense que realizó contribuciones fundamentales en física cuántica, análisis funcional, teoría de conjuntos, teoría de juegos, ciencias de la computación, economía, análisis numérico, cibernética, hidrodinámica, estadística y muchos otros campos. Es considerado como uno de los más importantes matemáticos de la historia moderna.

- **Single instruction Stream, Multiple Data Stream (SIMD):** Un solo flujo de instrucciones y varios flujos de datos. Cuenta con una sola unidad de control que asigna instrucciones a los elementos de procesamiento, todas las unidades de proceso ejecutan la misma instrucción. Paralelismo de datos, computación vectorial.

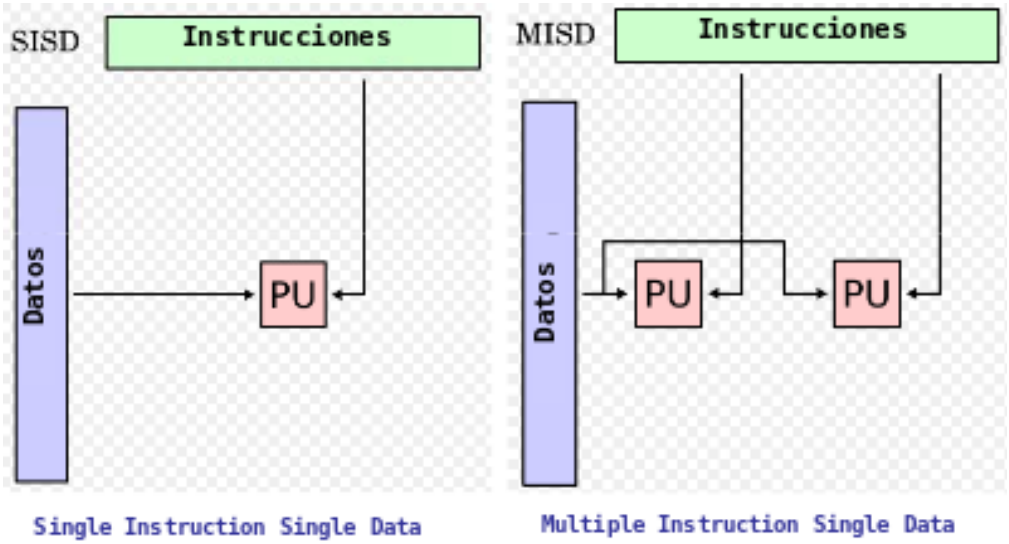


Figura 1.12: Diagrama SISD - MISD [10]

- **Multiple Instruction Stream, Single Data Stream (MISD):** Varios flujos de instrucciones y un solo flujo de datos. Usada en Arrays sistólicos que se definen como : "Una arquitectura y un paradigma aplicables a la computación en paralelo. Los algoritmos paralelos sistólicos exhiben un flujo direccional por el cual los datos se dividen y quedan en cola para pasar por iguales secuencias de procesamiento al moverse de un proceso al siguiente. La metáfora del flujo de sangre ha servido para caracterizar el flujo de datos a lo largo del proceso." [31]

- **Multiple Instruction Stream, Multiple Data Stream (MIMD):** Varios flujos de instrucciones y varios flujos de datos. Modelo general, varias implementaciones.

[31] MISD definition, [fecha de consulta: 1 Diciembre 2015] ,[en línea] http://www.oocities.org/ohcop/systolic.html

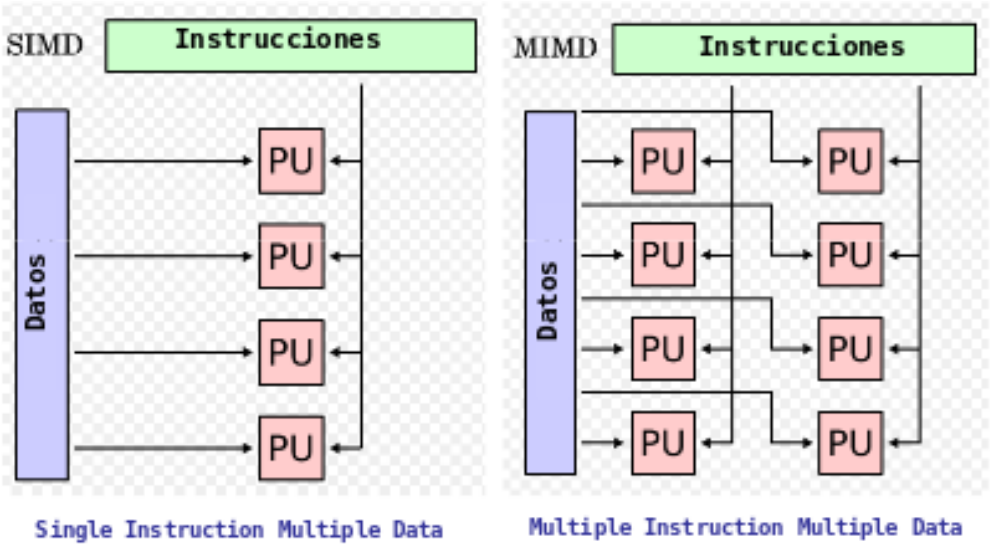


Figura 1.13: Diagrama SIMD - MIMD [10]

1.6. Estructura del trabajo de grado

El trabajo de grado esta organizado de la siguiente forma. En el Capitulo 2 se presenta la implementación del proyecto en cuestión utilizando herramientas y técnicas que serán descritas de manera minuciosa y explicativa, en el capitulo 3 se presentara las convenciones de ingeniería de software para describir el sistema de tipo web en el que se sustenta la implementación. Finalmente se presentan las conclusiones de esta implementación en el capitulo 4.

Capítulo 2

Descripción de la solución

Para solventar el problema descrito 1.1, se ha de detallar de forma explicita la solución del problema conjunto a los apartados plasmados a lo largo de documento, para así alcanzar de manera puntual los objetivos propuestos. Así, Obedeciendo el orden de ideas, se mostrara aspectos como los implementos usados para la elaboración de la plataforma, procedimientos usados y configuraciones. Dentro del abanico de opciones para elegir nuestro sistema operativo central, se decanto el sistema por un sistema Linux, dada su versatilidad, documentación, código abierto y soporte. Por último, la lista de elementos como scripts y archivos de configuración usados en este proceso estarán alojados en la carpeta **ConfigScripts** del **repositorio de trabajo.**[24]

2.1. Elementos usados

Para el desarrollo de la plataforma tecnológica de tipo beowulf se usaron los siguientes elementos:

2.1.1. Hardware

Dentro de los requerimientos de hardware del cluster, se podrán usar WS (Pc's) o cualquier tipo de computadora que cuente con una interfaz de red (sea cual sea el tipo) , para poder comunicarse en red con las otras WS. En el caso de nuestro cluster se usaron las siguientes estaciones de trabajo.

Tabla 2.1: Características estaciones de trabajo de la implementación.

Máquina	**Nombre del host**	**Procesador**	**Memoria Ram**	**HDD**
CM	gc1-ce.utp.edu.co	Intel(R) Core(TM)2 Duo CPU E4600 @ 2.40GHz	2 GB DIMM DDR2 Synchronous 800 MHz	80GB ST380815AS
CL	gc1-ce.utp.edu.co	Intel(R) Core(TM)2 Duo CPU E4600 @ 2.40GHz	2 GB DIMM DDR2 Synchronous 800 MHz	80GB ST380815AS
WN0	gc1-wn0.utp.edu.co	Intel(R) Core(TM)2 Duo CPU E4600 @ 2.40GHz	2 GB DIMM DDR2 Synchronous 800 MHz	80GB ST380815AS
WN1	gc1-wn1.utp.edu.co	Intel(R) Core(TM)2 Duo CPU E4600 @ 2.40GHz	2 GB DIMM DDR2 Synchronous 800 MHz	80GB ST380815AS
WN2	gc1-wn2.utp.edu.co	Intel(R) Core(TM)2 Duo CPU E4600 @ 2.40GHz	2 GB DIMM DDR2 Synchronous 800 MHz	80GB ST380815AS
WN3	gc1-wn3.utp.edu.co	Intel(R) Core(TM)2 Duo CPU E4600 @ 2.40GHz	2 GB DIMM DDR2 Synchronous 800 MHz	80GB ST380815AS
WN4	gc1-wn4.utp.edu.co	Intel(R) Core(TM)2 Duo CPU E4600 @ 2.40GHz	2 GB DIMM DDR2 Synchronous 800 MHz	80GB ST380815AS
WN5	gc1-wn5.utp.edu.co	Intel(R) Core(TM)2 Duo CPU E4600 @ 2.40GHz	2 GB DIMM DDR2 Synchronous 800 MHz	80GB ST380815AS
NFS	gc1-nfs.utp.edu.co	Intel(R) Core(TM)2 Duo CPU E4600 @ 2.40GHz	2 GB DIMM DDR2 Synchronous 800 MHz	80GB ST380815AS

Dónde:

- **CE:** Este tipo de estaciones se encuentran encargadas de la administración lógica del cluster.
- **CL:** Las estaciones CL se encargan de enviar las tareas para que sean procesadas por el sistema, generalmente son estaciones con poder limitado a comparación de las demás máquinas del esquema.
- **WN:** Son los nodos encargados de hacer el trabajo "Pesado". Se ocupan de procesar los trabajos, tareas , peticiones del CL. Entre más WN tenga nuestra infraestructura, mejor será la capacidad de cómputo.
- **NFS:** Network File System , máquina encargada de compartir archivos y configuraciones, asegura transparencia y usabilidad.

2.1.2. Software

Al iniciar el proyecto de GridUTP se establecieron una serie de parámetros, entre ellos, una la lista de software preestablecido con la cual se trabajaría. Se utilizaron los siguientes módulos y/o software para las máquinas :

- **Sistema Operativo:** Debian 7.8.0 32 o 64 bits
- **Administrador de carga de trabajo:** HTCondor 7 o versiones posteriores
- **Librerías,API's,Frameworks:** OpenMPI - MPICH2 - OPENSSH - GCC 5.2.

Nota : La instalación de algunos módulos varía dependiendo de la máquina, más adelante se describirá cuales son los módulos / software que se instalarán en las respectivas máquinas. Es importante resaltar que la instalación de todas las librerías se deberá hacer en la misma ubicación para todos los nodos.

2.1.3. Modelo de red usado

Dentro del modelo de red usado, se decantó por el siguiente esquema, dada la necesidad de facilidad y los elementos aportados por el grupo de investigación Sirius:

Nota: la red red por defecto será 192.168.0.0/24,diferente a la imagen.

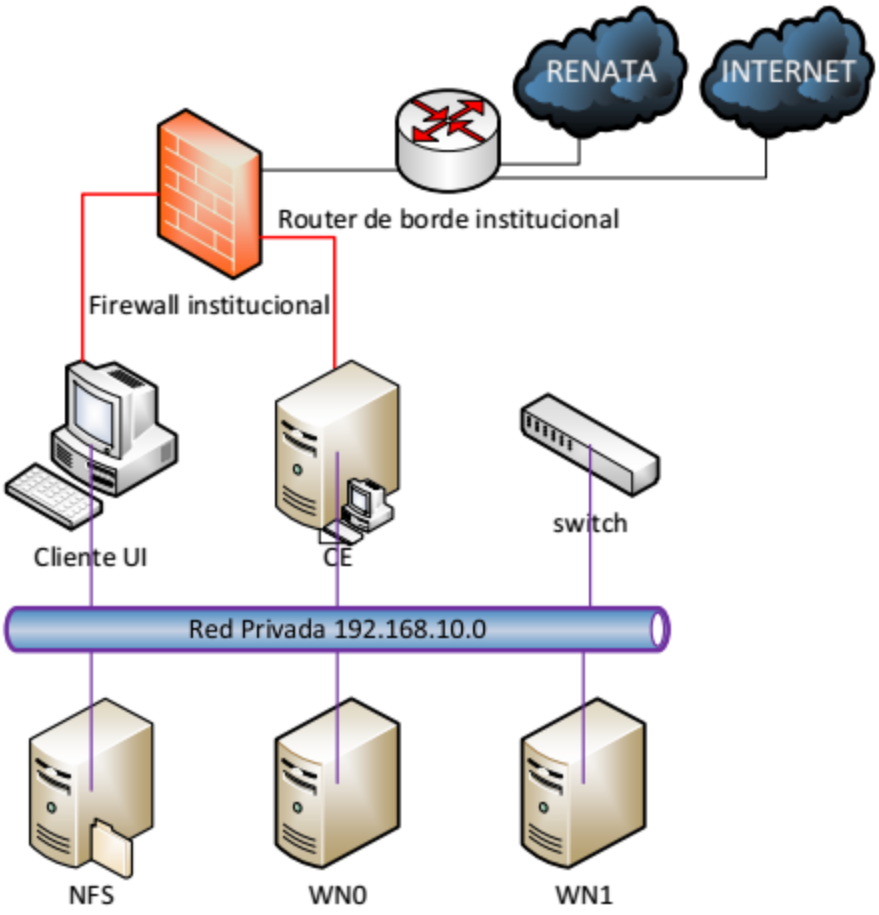


Figura 2.1: Modelo de Red usado en la implementación [11]

El proyecto tiene como meta la escalabilidad para la interconexión con otros clusters a nivel nacional, esto gracias a RENATA [1], pero por razones de alcance solo se describirá el proceso de implementación intra-institucional, omitiendo la interconexión de otros sistemas.

2.2. Configuración de los nodos:

Una ves instalado el sistema linux (que para este proyecto fue Debian), se procede a configurar los nodos de la siguiente manera.

2.2.1. Selección de servidor de repositorios.

Dada la naturaleza de la instalación, el sistema operativo aún se encuentra sin una lista de repositorios predefinida.

```
utp@gsirius-cm:~$ su
root@gsirius-cm:/home/utp# nano /etc/apt/sources.list
```

Comentar todas las líneas y pegar la dirección del servidor favorito, para este caso:

```
deb http://ftp.fr.debian.org/debian stable main contrib non-free
deb http://ftp.debian.org/debian/ wheezy-updates main contrib non-free
deb http://security.debian.org/ wheezy/updates main contrib non-free
```

luego, para que el sistema se percate de los cambios :

```
root@gsirius-cm:/home/utp# apt-get update
```

También se pueden generar los repositorios desde el siguiente enlace: creador de repos.

[1] RENATA es la red nacional de investigación y educación de Colombia que conecta, articula e integra a los actores del Sistema Nacional de Ciencia Tecnología e Innovación (SNCTI) entre sí y con el mundo, a través del suministro de servicios, herramientas e infraestructura tecnológica para contribuir al mejoramiento del nivel de productividad, efectividad y competitividad de la producción científica y académica del país.

2.2.2. Cambiar nombre del host (en todos los nodos, opcional):

Para configurar el hostname se deben modificar los siguientes archivos: /etc/hostname /etc/hosts

```
utp@gsirius-cm:~$ su
root@gsirius-cm:/home/utp# nano /etc/hostname
```

Se edita con el nombre deseado, luego:

```
root@gsirius-cm:/home/utp# nano /etc/hosts
```

Para no tener que reiniciar el equipo para ver los cambios:

```
root@gsirius-cm:/home/utp# /etc/init.d/hostname.sh start
```

Utilizar la convención antes mencionada, para este caso:

- **CM:** gsirius-cm.utp.edu.co
- **CL:** gsirius-cl.utp.edu.co
- **NFS:** gsirius-nfs.utp.edu.co
- **WN0:** gsirius-wn0.utp.edu.co

modificamos el archivo con el nombre pertinente. Nota: luego se clonaran las máquinas WN, por lo que este paso se deberá repetir para cada uno de los nodos.

2.2.3. Añadir usuarios sudoers (opcional,Todos)

primero, se instala sudo en los nodos.

```
utp@gsirius-cm:~$ su
root@gsirius-cm:/home/utp# apt-get install sudo
```

se añaden los usuario a sudoers [2]

```
# User privilege specification
root        ALL=(ALL:ALL) ALL
gsirius-cm    ALL=(ALL:ALL) ALL
# Allow members of group sudo to execute any command
%sudo        ALL=(ALL:ALL) ALL
```

[2]Sudoers, [fecha de consulta: 4 Diciembre 2015] ,[en línea] http://www.pendrivelinux.com/how-to-add-a-user-to-the-sudoers-list/

2.2.4. Instalación Software necesario

```
utp@gsirius-cm:~$ sudo apt-get install gedit
utp@gsirius-cm:~$ sudo apt-get install build-essential
utp@gsirius-cm:~$ sudo aptitude install openssh-server
utp@gsirius-cm:~$ sudo aptitude install mpich2
```

2.2.5. Configuración de red (en todos los nodos) :

Dada la naturaleza de la red se procede a configurar las máquinas con las siguientes direcciones. Se debe de tener en cuenta que para la configuración de la subred se utilizaron 2 interfaces, tanto, para el CM como para el CE. También se utilizó un switch de red.

Tabla 2.2: Características de red de las estaciones de trabajo de la implementación.

Máquina	**Direccion IP**	**Interfaz**	**Gateway**	**Mask**	**DNS**
CM	192.168.0.1	eth0 (sub-red)	192.168.0.1	24	192.168.0.1
CM	10.1.x.xx (privada)	eth1 (Internet)	10.1.4.1	22	10.1.0.10 o 10.1.0.11
CL	192.168.0.2	eth0 (sub-red)	192.168.0.1	24	192.168.0.1
CL	10.1.x.xx (privada)	eth1 (Internet)	10.1.4.1	22	10.1.0.10 o 10.1.0.11
NFS	192.168.0.10	eth0	192.168.0.1	24	10.1.0.10
WN0	192.168.0.3	eth0	192.168.0.1	24	10.1.0.10
WN1	192.168.0.4	eth0	192.168.0.1	24	10.1.0.10
WN2	192.168.0.5	eth0	192.168.0.1	24	10.1.0.10
WN3	192.168.0.6	eth0	192.168.0.1	24	10.1.0.10
WN4	192.168.0.7	eth0	192.168.0.1	24	10.1.0.10
WN5	192.168.0.8	eth0	192.168.0.1	24	10.1.0.10

Para modificar la configuración de red:

```
utp@gsirius-cm:~$ sudo nano /etc/network/interfaces
```

Modificar la configuración de red en 2 interfaces:
Ver archivo anexo **"interfaces"**.
luego:

```
utp@gsirius-cm:~$ sudo service networking restart
```

Nota: Se debe cerciorar que ningún gestor de conexiones esté activo, de ser así, asignar esta configuración a el gestor que se esté usando para evitar solapamientos en configuraciones.

CM y CE tendrán 2 interfaces, para los demás solo será la interfaz eth0 (mayoría de los casos). En este paso se difiere mucho de las instalaciones, la misma institución asignó 2 direcciones de red públicas (CE-CL) para la conexión remota, además se debe solicitar abrir puertos necesarios para la integración de nuestro pool a un grid más grande. Algunos de estos puertos son:

- 22 - SSH
- 80 - Web
- 2119 - Globus Gatekeeper (GRAM)
- 2811 - GridFTP
- 8443 - GUMS/VOMS
- 9443 - WebServices
- 200000:22000 - GLOBUS TCP PORT RANGE
- 9000:9999 - HTCONDOR TCP PORT RANGE
- 443 - 943 - 1194

2.2.6. Configuracíon network switching (solo en el CM)

La siguiente configuración es muy permisiva, pero funcional, para añadir reglas de seguridad a las iptables consultar [3]. Descomentar la linea "net.ipv4.ip_forward1̄ en etcsysctl.conf"

Ver script anexo **"iptables.sh"**.

2.2.7. Asignación ruta por defecto (Todos los WN y NFS)

```
utp@gsirius-wn0:~$ ip route del default
utp@gsirius-wn0:~$ ip route add default via 192.168.0.1
```

[3]Configua iptables , [fecha de consulta: 4 Diciembre 2015] ,[en línea] https://wiki.debian.org/iptables

2.2.8. Actualización sistema operativo(Todos - opcional):

```
utp@gsirius-cm:~$ sudo aptitude update
utp@gsirius-cm:~$ sudo aptitude full-upgrade
```

2.2.9. Instalación NTP

De ser necesario iniciar el demonio ntp con el sistema.

```
utp@gsirius-cm:~$ sudo aptitude install ntp
Para consultar la fecha y hora de nustro sistema:
utp@gsirius-cm:~$ date -R
coordinación Network Time Protocol
```

2.3. instalación de HTcondor (Todos - menos NFS)

2.3.1. Añadir repositorios estables HTCondor(CM-CL-WN):

```
utp@gsirius-cm:~$ sudo echo
"deb http://research.cs.wisc.edu/htcondor/debian/stable/ wheezy contrib"
>> /etc/apt/sources.list
```

2.3.2. Configuración secure shell – ssh (para mpi) – En todos

Primero se crea las llaves públicas, luego deben ser compartidas entre todos los hosts exceptuando el NFS.

```
ssh-keygen -t rsa
Para pasar las llaves entre los usuarios se haces uno por uno
de la siguiente forma:
ssh-copy-id -i ~/.ssh/id_rsa.pub user_name@user_ip_address
Para usuarios comunes que no son root anteponer el comando sudo
sudo ssh-copy-id -i ~/.ssh/id_rsa.pub user_name@user_ip_address
```

Nota: se debe hacer el proceso para los distintos usuarios (de utp a utp o de root a root) logueados como dichos usuarios.A su vez se debe realizar este paso, si se quieren lanzar trabajos en el universo paralelo con OpenMPI

2.3.3. Añadimos llave del repositorio al sistema:

```
utp@gsirius-cm:~$ wget -qO -
http://research.cs.wisc.edu/htcondor/debian/HTCondor-Release.gpg.key |
sudo apt-key add -
```

2.3.4. Actualizamos e instalamos:

La instalación agregará automáticamente el usuario 'condor' y el grupo si este no existe.

```
utp@gsirius-cm:~$ sudo apt-get update
utp@gsirius-cm:~$ sudo apt-get install condor
```

Configuración del Network File System (NFS)

Primero instalamos el servidor NFS en el ordenador dedicado para dicha labor

```
utp@gsirius-cm:~$ sudo apt-get install nfs-kernel-server
```

Se crea el directorio en el que se almacenara la carpeta que se va a compartir. El directorio debe ser creado en un path que todos los equipos puedan usar de la misma forma (El mismo directorio con el mismo path debe ser creado en todos los equipos), por ejemplo en este caso se creó la carpeta en la raíz de directorio (/).

```
utp@gsirius-cm:~$ mkdir /exports
```

y la carpeta se desea compartir.

```
utp@gsirius-cm:~$ mkdir /exports/condor
```

Se modifica el archivo exports para que se sepa que directorio compartir.

```
utp@gsirius-cm:~$ nano /etc/exports
```

Quedando:
Ver archivo anexo **"exports"**.
No es necesarios especificar anonuid o anongid, pero por razones de seguridad se hace. Luego se modifica el archivo Hots.allow para que los nodos puedan hacer uso de la carpeta previa:

```
utp@gsirius-cm:~$ nano /etc/hosts.allow
```

Se especifica el rango de direcciones IP que tienen permitido hacer uso de los recursos a compartir quedando así:

Ver archivo anexo **"host.allow"**.

```
utp@gsirius-cm:~$ exportfs -a
```

2.3.5. Instalación NFS (WN-CL-CM)

Instalamos el cliente NFS en los demás nodos

```
utp@gsirius-cm:~$ apt-get install nfs-common
```

Ahora creamos el directorio en todos los hosts el cual debe tener el mismo path que la creada en el NFS.

```
utp@gsirius-cm:~$ mkdir /exports/condor
```

Modificamos el archivo fstab para añadir la carpeta del NFS

Ver archivo anexo **"fstab"**.

Finalmente montamos el directorio:

```
utp@gsirius-cm:~$ sudo mount
192.168.0.10:/exports/condor /exports/condor
```

2.4. Prueba funcionamiento MPI

Para probar el funcionamiento de los algoritmos escritos en mpi, se procede a crear el archivo dentro de la carpeta compartida previamente creada y montada.

Compilamos nuestro programa en mpi:

```
utp@gsirius-cm:/exports/condor ~$ mpic++.mpich primos.c -o primos
```

Por último procedemos a ejecutar nuestro programa:

```
utp@gsirius-cm:/exports/condor ~$ mpirun -n 4 ./primos
```

Cabe resaltar que la anterior instrucción correrá 4 procesos en la máquina que ejecutó la orden, para distribuir la tarea en los demás nodos, se necesita especificar las máquinas disponibles para el cálculo.

El formato del archivo que especificara los nodos disponibles será:
[Nombre de la máquina:][Nucleos disponibles][usuario=]

```
utp@gsirius-cm:/exports/condor ~$ cat machinefile
gsirius-wn2.utp.edu.co:2 user=wn2
gsirius-wn4.utp.edu.co:2 user=wn4
```

Por último se especifica el archivo creado con la línea: –hostfile

```
utp@gsirius-cm:/exports/condor ~$ mpirun -n 4
--hostfile [nombre del archivo] ./primos
```

2.5. Configuración HTcondor

Luego de instalar HTCondor en toda la jerarquía de cómputo, se procede a definir los perfiles de cada máquina.

2.5.1. lista de archivos de configuración

Por cuestiones de seguridad y comodidad,los demonios que correrán en cada máquina se especificaron en un archivo compartido en el NFS. Por otra parte, se configuraron las demás opciones en un archivo local existente cada máquina.

```
/etc/condor/condor_config

Configuración general,creado por la instalación de HTCondor .

/Carpeta_compartida_nfs/condor_config.xx

Dónde xx puede ser [headnode-interactive-worker]
(Configuración de los demonios (útil cuando se tiene un pool
de máquinas muy grande).Creado por nosotros (opcional)).
```

Se debe tener en cuenta, que dentro de la configuración de HTCondor se deben tener los perfiles mencionados en el apartado anterior y cuyos demonios para cada perfil serán:

```
WORKER (WN): MASTER, STARTD
CLIENT (CL): MASTER, SCHEDD
COMPUTER MASTER (CM): MASTER, COLLECTOR, NEGOTIATOR, SCHEDD
```

Luego se procede a la creación del archivo que correrá los demonios en cada máquina:

- **Headnode (CM):**

  ```
  utp@gsirius-cm:~$ su
  root@gsirius-cm:/home/utp# nano
  /exports/condor/config/htcondor/condor_config.headnode
  ```

  Quedando así:

  ```
  #Configuración - GridUTP
  #Para más info:
  #http://research.cs.wisc.edu/htcondor/manual/v8.2/3_3Configuration.html
  #Lista de demonios para el CM
  DAEMON_LIST = MASTER, COLLECTOR, NEGOTIATOR, SCHEDD
  ```

- **Interactive nodes (CL):**

  ```
  utp@gsirius-cm:~$ su
  root@gsirius-cm:/home/utp# nano
  /exports/condor/config/htcondor/condor_config.interactive
  ```

  Quedando así:

  ```
  # Configuración - GridUTP
  # Para más info:
  #http://research.cs.wisc.edu/htcondor/manual/v8.2/3_3Configuration.html
  # Lista de demonios para el Cliente UI
  DAEMON_LIST = MASTER, SCHEDD
  ```

- **Worker nodes (WN):**

```
utp@gsirius-cm:~$ su
root@gsirius-cm:/home/utp# nano
/exports/condor/config/htcondor/condor_config.worker
```

Quedando así:

```
# Configuración - GridUTP
# Para más info:
#http://research.cs.wisc.edu/htcondor/manual/v8.2/3_3Configuration.html
# Lista de demonios para el WN
DAEMON_LIST = MASTER, STARTD
```

Se Procede a realizar los links simbólicos para aislar aún más las configuraciones de cada máquina,esto con el fin de mejorar la seguridad,pero deja de lado la practicidad.

```
root@gsirius-cm: /exports/condor/config/htcondor#
ln -s condor_config.headnode condor_config.gsirius-cm
root@gsirius-cm: /exports/condor/config/htcondor#
ln -s condor_config.interactive condor_config.gsirius-cl
root@gsirius-cm: /exports/condor/config/htcondor#
ln -s condor_config.worker condor_config.gsirius-wn0
root@gsirius-cm: /exports/condor/config/htcondor#
ln -s condor_config.worker condor_config.gsirius-wn1
root@gsirius-cm: /exports/condor/config/htcondor#
ln -s condor_config.worker condor_config.gsirius-wn2
root@gsirius-cm: /exports/condor/config/htcondor#
ln -s condor_config.worker condor_config.gsirius-wn3
root@gsirius-cm: /exports/condor/config/htcondor#
ln -s condor_config.worker condor_config.gsirius-wn4
root@gsirius-cm: /exports/condor/config/htcondor#
ln -s condor_config.worker condor_config.gsirius-wn5
```

Dependiendo de este perfil, se debe linkear desde el archivo de configuración \etc \condor\ condor_config de cada máquina. Reemplazando o adicionando luego de la siguiente línea "## Where is the machine-specific local config file for each host?" por:

```
##  Where is the machine-specific local config file for each host?
LOCAL_CONFIG_FILE = /exports/condor/config/htcondor/condor_config.\$(HOSTNAME)
```

Si se tiene un pool muy grande y no se quiere especificar o hacer links simbólicos, cambiar .$(HOSTNAME), por el .worker ó .interactive ó .headnode, según sea el caso; recordar que este archivo se debe proteger para evitar colapsos del sistema.

Nota: También se puede especificar los demonios que se quieren correr de la misma manera, pero en el archivo de configuración \etc\condor\condor_config de cada una de las máquina existentes en el pool.

CM

Ver script anexo **"condor_config.headnode"**

CL

Ver script anexo **"condor_config.interactive"**

WN

Ver script anexo **"condor_config.worker"**

Para probar el sistema procedemos a ejetutar en la terminal la siguiente línea:

```
utp@gsirius-cm:/exports/condor ~$ condor_status
```

Mostrando exitosamente los recursos disponibles de nuestro cluster:

```
Name               OpSys  Arch   State  Activity LoadAv Mem   ActvtyTime

slot1@gsirius-wn2. LINUX  X86_64 Unclaimed Idle 0.000  993  1+16:14:56
slot2@gsirius-wn2. LINUX  X86_64 Unclaimed Idle 0.000  993  1+16:15:17
slot1@gsirius-wn4. LINUX  X86_64 Unclaimed Idle 0.000  993  1+16:15:02
slot2@gsirius-wn4. LINUX  X86_64 Unclaimed Idle 0.020  993  1+16:50:23
            Total Owner Claimed Unclaimed Matched Preempting Backfill



X86_64/LINUX     4     0       0         4       0          0        0

       Total     4     0       0         4       0          0        0
```

HTCondor provee una serie de directivas para controlar los trabajos dentro del sistema, estas directivas deben aparecer en un archivo llamado "archivo_submit" que es el archivo por excelencia para la ejecución de cualquier trabajo en un cluster administrado por HTCondor. El archivo "submit" más básico posee:

- #: Comentarios.
- Universe: Entorno en el cuál correrá nuestro programa.
- Executable: ejecutable.
- input: archivos. (STDIN)
- arguments : Números,cadenas ... (STDIN)
- output: Salidas de nuestro programa. (STDOUT)
- error: Errores presentados en su ejecución. (STDERR)

El archivo más básico :

```
##############################################
##   submit description file for a parallel program
# submit.sub
##############################################
universe = parallel
executable = /bin/sleep
arguments = 5
error = errorfile
machine_count = 2
queue
```

Para probar la comunicación del sistema se procede a ejecutar un ejemplo básico que invocará una rutina básica del sistema operativo. La siguiente línea envía una tarea al sistema.

```
utp@gsirius-cm:/exports/condor ~$ condor_submit submit.sub
```

De ser una ejecución limpia, el archivo errorfile deberá estar totalmente en blanco.

2.6. Mpi & HTCondor

Para poder ejecutar un algoritmo MPI conjunto a HTCondor se debe ejecutar un script especial para enlazar los requerimientos del ejecutable, HTCondor toma de manera inicial la configuración contenida en este script, pasando por configuraciones como ssh y directorios de librerías necesarias para la ejecución, el script usado en el proyecto es el siguiente:

Ver script anexo **"MPIwrapper.sh"**

A continuación se presentará el código usado para esta prueba:

Ver código anexo **"PrimosMPI.c"**

Para la ejecución de MPI se debe cambiar el archivo "submit", quedando:

Ver código anexo **"submit.sub"**

Por ultimo, ejecutamos la tarea con HTCondor:

```
utp@gsirius-cm:/exports/condor ~$ condor_submit submit.sub
```

Capítulo 3

WebMPI

En este capítulo se ahondara la implementación de la interfaz multiusuario denominada web mpi desde el lado de la ingeniería de software, se plantea todo el apartado de este capítulo en un documento alojado en la carpeta **Documents** en el **repositorio de trabajo.**[24]

Ver anexo **"Documento de ingeniería de software"**

Capítulo 4

Conclusiones

El presente trabajo demuestra técnicamente la posibilidad de realizar una implementación de bajo costo con elementos comunes (ver sección 2.1.1) presentes en las salas de sistemas de la universidad cumpliendo con el objetivo propuesto relacionado.

Lo que respecta a la plataforma, una implementación en windows supone más problemas de los imaginados, primero no cuenta con soporte nativo para NFS por lo que se dependera del uso de software de terceros, segundo, muchas rutinas de condor no están soportadas en este sistema operativo, es por eso, el uso de herramientas libres se concibió en una decisión acertada ya que contrajo una amplia base de conocimiento, ahorro de tiempo, además de dinero, encontrando implementaciones existentes de todo tipo con procesos detallados.

La implementación e instalación de un framework como Htcondor supuso un verdadero reto a la hora de interconectar las maquinas y ponerlas a punto, dada la susceptibilidad de la comunicación y configuración de sus demonios, y la desactualización de algunos archivos de configuración. Por el lado de mpi, fue fácil su adecuación al sistema, gracias al gran soporte del sistema operativo Debian y a su amplia lista de paquetes compatibles. Referente a la combinación de mpi y Htcondor, no se pudo lograr ejecutar el wrapper necesario para que pudiese funcionar, ya que este, esta totalmente desactualizado comparado con la versión actual de HTCondor.

Además de ser posible el desarrollo de una implementación intuitiva, se ha incorporando una abstracción de tipo web escrita en javascript (del lado del cliente) y nodejs (del lado del servidor), los cuales fueron fáciles de entender y de configurar por su facilidad de aprendizaje y amplia documentación disponible actualmente sobre cualquier tema en particular.

Por medio a este desarrollo web se re-asegura el principio de transparencia de los sistemas distribuidos, dando una verdadera ventaja para las personas que no quieren preocuparse por agentes externos dentro de sus programas, también, incrementando el tiempo productivo de los usuarios contando con la comodidad de ejecutar su código desde cualquier lugar.

Al comienzo se empezó a hacer pruebas bastante sencillas haciendo uso de un "Hola mundo"programado en C y con mpi. Lo que muestra el algoritmo a la hora de ser ejecutado es:

```
Hello world from processor gsirius-wn4, rank 0 out of 1 processors
```

Fue una tarea satisfactoria ya que el algoritmo se hizo correr en todas las maquinas y no se presentó ningún problema.

Después se hizo uso de un algoritmo encargado de encontrar los números primos que hay en las potencias de 2 hasta un numero N que uno es el encargado de cambiarlo, en este caso el valor escogido fue 140000 para poder encontrar los primos hasta 2 en la 17 potencia.

Entregando la cantidad de hilos usados y el tiempo gastado encontrando los primos en las potencias de 2. En este caso con un solo hilo demoró 5.10558 segundos en hacer el trabajo; y aquí son mostradas las pruebas realizadas con 1 y más hilos:

- Con 1 hilo se tardó 5.10558
- Con 3 hilos se tardó 2.56046
- Con 5 hilos se tardó 1.28632
- Con 7 hilos se tardó 0.857135

Lo que se observa es que el algoritmo mejoró cuando se aumentaron los hilos; pero hay que tener cuidado porque hay algoritmos que empeoran cuando son más los hilos encargados de hacer el trabajo.

Por esto se plantea que la paralelización de algoritmos es como el futuro de la computación.

4.1. Futuros trabajos de investigación

- **Plataforma de tipo SaaS:** Una mejor implementación, conjunto a una mejor infraestructura, representaría una plataforma atractiva para la ejecución de algoritmos de niveles industriales, la elaboración de un sistema como SaaS (software como servicio) proporcionaría la interfaz necesaria para disponer esos recursos para diversos objetivos, uno de estos con fines económicos.

- **Utilización de dockers:** Para la labor distribuida, la obtención de recursos en nuestro proyecto se basa en la disposición total de un entorno que independiente o no de que se ejecuten tareas, se basará en formatos de compilación y ejecución pre-establecidas. La idea de la utilización de dockers es dotar de ambientes cambiantes capaces de atender múltiples peticiones con múltiples requerimientos, adaptando el espacio de trabajo a tal punto de generar una instancia optimizada para que solo se dispongan las herramientas puntuales cuando se separan requerimientos para los ejecutables. También la utilización de dockers garantizará un estándar de seguridad más alto que el brindado en el prototipo de implementación aquí vista, por otro lado, la distribución de recursos sería totalmente heterogénea basándose exclusivamente en una implementación que correría en los sistemas operativos más comunes (gracias a la máquina virtual presente en algunas implementaciones de docker), por último, la utilización de recursos sería mucho mejor dado que dentro de un solo ordenador podría estar alojado toda la infrastructura administrativa aprovechando los demás nodos para labores de ejecución.

- **Grid UTP:** Este prototipo de implementación a la larga es el inicio de una otra más escalable y viable, que podría estar presente en la infraestructura de todo el campus universitario, siendo una verdadera herramienta de ejecución de algoritmos que beneficiaría enormemente la academia.

Bibliografía

[1] B. C. BENÍTEZ, "Diseño de una plataforma computacional tipo cluster, con el aprovechamiento de las salas del bloque r, para el procesamiento de información de los grupos de investigación de la utp." *UNIVERSIDAD TECNOLÓGICA DE PEREIRA*, pp. 1–109.

[2] M. A. Castro, "Estructuración de un cluster beowulf," *Universidad Católica de Temuco*, pp. 1–37. [Online]. Available: https://xxito.files.wordpress.com/2008/11/trabajo-final-beowulf.pdf

[3] J. L. Bosque, "Arquitecturas paralelas basadas en clusters," *Universidad Rey Juan Carlos*, 2006. [Online]. Available: http://dac.escet.urjc.es/docencia/Doctorado/CPBC/sesion1.pdf#page=1&zoom=auto,-21,852

[4] L. M. T. Morales, "Construcción de un cluster moxis: Pruebas con simulación de halos," *Universidad Tecnológica de Mexaca*, pp. 14–18, 2002. [Online]. Available: http://mixteco.utm.mx/~resdi/historial/Tesis/Tesis_Leonardi.pdf

[5] A. G. Sanmiguel, "Clúster de alta disponibilidad y balanceo de carga sobre un servidor web," *Universisdad autonoma de barcelona*, pp. 39–40. [Online]. Available: http://www.recercat.cat/bitstream/handle/2072/219063/GutierrezSanmiguelAlfred-ETISa2011-12.pdf?sequence=1

[6] "Top 500 website," 2015. [Online]. Available: http://www.top500.org

[7] F. M. G. Padilla, "Diseño de una solución para servidores de alta disponibilidad y balanceo de carga con open source," *UNIVERSIDAD ALFREDO PÉREZ GUERRERO UNAP*, pp. 50–60, 2011.

[8] "Ibm hpc website," 2015. [Online]. Available: http://www-03.ibm.com/systems/platformcomputing/products/hpc/

[9] B. C. I. P. Mejía N. David, Fernández A. Diego, "Desarrollo de aplicaciones paralelas para clusters utilizando mpi (message passing interface)," *Escuela Politécnica Nacional*, pp. 1–10. [Online]. Available: http://www.researchgate.net/profile/David_Mejia6/publication/27557530_Desarrollo_de_aplicaciones_paralelas_para_clusters_utilizando_MPI_%28Message_Passing_Interface%29/links/02e7e5383941d05055000000.pdf

[10] F. de ingeniería, "Computación de alta performance," *Universidad de la República - Uruguay*, pp. 7–9, 2009.

[11] "Manual de configuración de un clúster con htcondor," 2014. [Online]. Available: https://drive.google.com/a/sirius.utp.edu.co/file/d/0B0CLFmiGU6LjSzhyOTloS3pjS00/edit

[12] G. I. G. Echeverry, "Tópicos sobre la optimización de algoritmos," *Ingeniería en investigación*, pp. 55–62. [Online]. Available: http://www.bdigital.unal.edu.co/23891/1/20976-70978-1-PB.pdf

[13] A. Lafuente, "introducción a los sistemas distribuidos," *UPV/EHU*, pp. 3–10. [Online]. Available: http://www.sc.ehu.es/acwlaroa/SDI/Apuntes/Cap1.pdf

[14] E. S. G. Rendón, "Propuesta de implementación de un clúster de altas prestaciones," *UNIVERSIDAD VERACRUZANA*, pp. 1–128. [Online]. Available: http://docplayer.es/1638546-Universidad-veracruzana-tesina-licenciado-en-\sistemas-computacionales-administrativos.html

[15] M. Creel, "Multi-core cpus, clusters, and grid computing:a tutorial," *Universidad Autónoma de Barcelona*, pp. 11–12. [Online]. Available: http://cook.rfe.org/grid.pdf

[16] A. A. Navas, "Arquitectura e ingeniería de computadores," *ESI-UCLM*, pp. 1–43. [Online]. Available: http://www.inf-cr.uclm.es/www/sbenito/AIC/Cursos%20Anteriores/Curso%202004-2005/Transparencias/Clusters.pdf

[17] U. de Almería, "Práctica 5 memoria compartida," *Universidad de Almería*, pp. 1–3. [Online]. Available: http://www.ual.es/~rguirado/so/practica5

[18] F. M. Montoya, "Memoria compartida distribuida en ambientes de bajo costo," *INSTITUTO TECNOLÓGICO DE COSTA RICA*, pp. 46–47, 2001. [Online]. Available: http://campuscurico.utalca.cl/~fmeza/papers/tesisMSc.pdf

[19] ecured.cu. [Online]. Available: http://www.ecured.cu/Balanceo_de_cargas

[20] E. G. R. Maria Mercedes Sinisterra, Tania Marcela Henao, "Cluster of load balancing and high availability for web and mail services," *Universidad Libre, Seccional Cali*, pp. 1–10, 2012. [Online]. Available: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=13&ved=0CCoQFjACOApqFQoTCNW08uickckCFcUrJgodxsAJng&url=http%3A%2F%2Fdialnet.unirioja.es%2Fdescarga%2Farticulo%2F4364562.pdf&usg=AFQjCNGH0CsrKRtC1sL2q4xSrJYoA8iz4g

[21] J. B. Ibáñez. (2013) Aspectos generales arquitectura de computadoras. [Online]. Available: http://www.infor.uva.es/~bastida/Arquitecturas%20Avanzadas/General.pdf

[22] M. L. Liu, *Computación distribuida fundamentos y aplicaciones*, 2004.

[23] M. Tosini. (2015) Arquitectura de computadoras ii. [Online]. Available: http://www.exa.unicen.edu.ar/catedras/arqui2/arqui2/filminas/Rendimiento%20de%20sistemas%20paralelos.pdf

[24] D. D. G. Anderson Ochoa. [Online]. Available: https://github.com/aaochoa/webmpi/tree/master/