Cluster de Alto Rendimiento

Daniel Jiménez
alejo3479@gmail.com

Andres Medina
wellvu@hotmail.com

Abstract

En este artículo se realiza una introducción a los conceptos y definiciones que se necesitan para el desarrollo de un cluster de computadoras, así como los componentes y las funciones del mismo, también se calculan los tiempos de proceso de dos programas ejecutados con diferentes condiciones en el cluster diseñado.

Keywords – cluster, nodo, balance de carga, Middleware, programación paralela.

I. Introducción

El término cluster (grupo o racimo) se define como el conjunto de computadores que se comportan como si fuesen un único computador.

La tecnología de clusteres ha evolucionado en apoyo de actividades que van desde aplicaciones de supercómputo y software de misiones críticas, servidores web y comercio electrónico, hasta bases de datos de alto rendimiento, entre otros usos.

El cómputo con clusteres surge como resultado de la convergencia de varias tendencias actuales que incluyen la disponibilidad de microprocesadores económicos de alto rendimiento y redes de alta velocidad, el desarrollo de herramientas de software para cómputo distribuido de alto rendimiento, así como la creciente necesidad de potencia computacional para aplicaciones que la requieran.

II. Clasificación de los Clusteres

Un cluster puede estar constituido por dos o más computadores, de los cuales se espera que presente uno o diferentes combinaciones de los siguientes servicios:

- Almacenamiento
- Alta disponibilidad
- Balance de carga
- Alto rendimiento

A. Almacenamiento

Un cluster de almacenamiento proporciona una imagen de sistema de archivos consistente a lo largo de los servidores en el cluster, permitiendo que los servidores lean y escriban de forma simultánea a un sistema de archivos compartido.

Un cluster de almacenamiento simplifica la administración de almacenamiento al limitar la instalación de aplicaciones a un sistema de archivos. Asimismo, con un sistema de archivos a lo largo del cluster, un cluster de almacenamiento elimina la necesidad de copias de más de los datos de la aplicación y simplifica la creación de copias de seguridad y recuperación contra desastres.

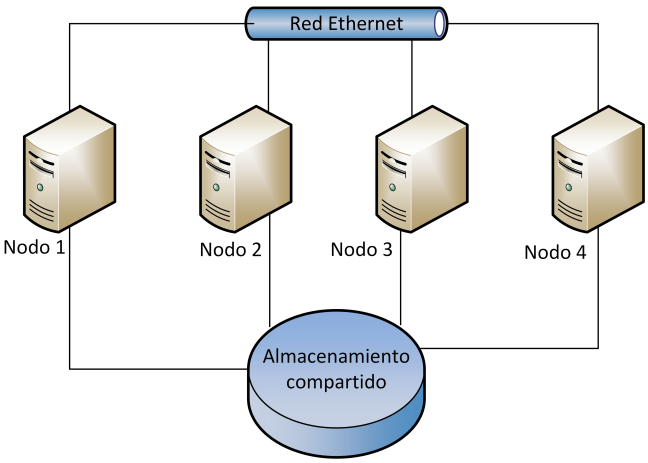

Figura 1: Cluster de Almacenamiento

B. Alta Disponibilidad

Los cluster de alta disponibilidad proporcionan continua disponibilidad de los servicios a través de la eliminación de la falla por un único elemento y a través del proceso de recuperación en contra de fallos al trasladar el servicio desde el nodo de cluster erróneo a otro nodo completamente funcional.

Generalmente, los servicios en los cluster de alta disponibilidad leen y escriben datos a través de la lectura y escritura a un sistema de archivos montado. Así, un cluster de alta disponibilidad debe mantener la integridad de los datos cuando un nodo recibe el control del servicio desde otro nodo.

Los nodos erróneos no son vistos por los clientes fuera del cluster. Los cluster de alta disponibilidad son conocidos también como cluster con recuperación contra fallas.

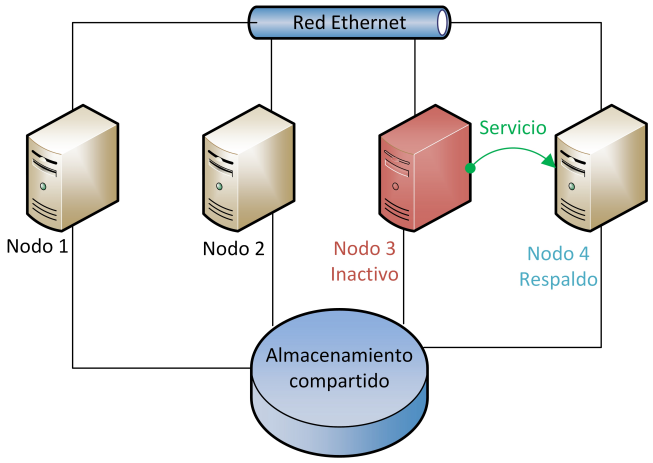

Figura 2: Cluster de Alta Disponibilidad

C. *Balance de Carga*

Los cluster de balance de carga responden a peticiones de servicios de red desde diferentes nodos para balancear las peticiones a los largo de los nodos del cluster.

El balance de carga proporciona escalabilidad económica porque se puede configurar el número de nodos de acuerdo con los requerimientos de balance de carga. Si un nodo en un cluster de balance de carga falla, el software de balance de carga detecta la falla y asigna las peticiones a otros nodos en el cluster.

Los nodos erróneos en un cluster de balance de carga no son visibles desde los clientes fuera del cluster.

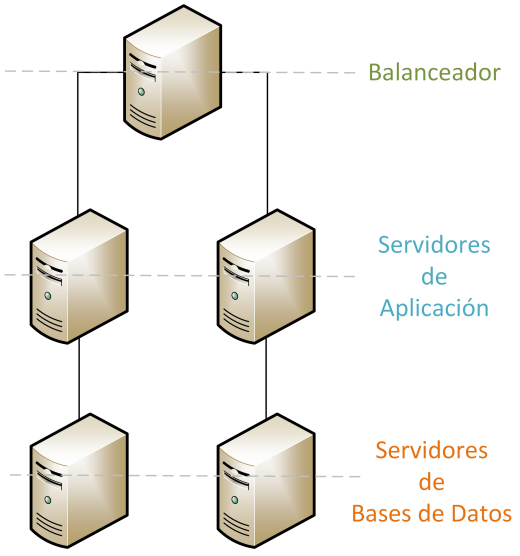

Figura 3: Cluster de Alta Disponibilidad

D. *Alto Rendimiento*

Los cluster de alto rendimiento utilizan los nodos para ejecutar cálculos simultáneos. Un cluster de alto rendimiento permite que las aplicaciones trabajen de forma paralela, mejorando así el rendimiento de éstas. Los cluster de alto rendimiento son conocidos como cluster computacionales o computación de red.

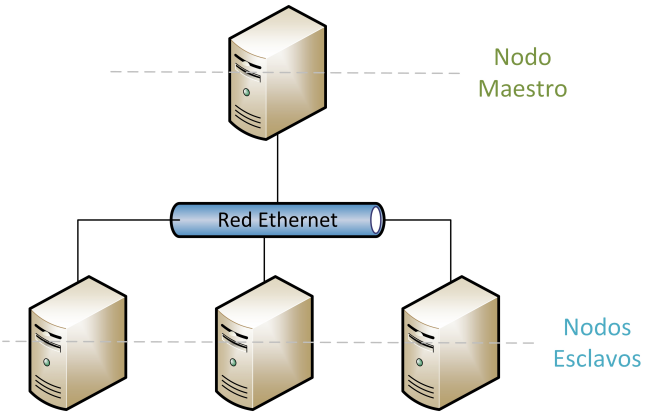

Figura 4: Cluster de Alto Rendimiento

III. Componentes de un Cluster

- Nodos
- Almacenamiento
- Sistemas operativos
- Conexiones de red
- Middleware
- Protocolos de comunicación y servicios
- Aplicaciones
- Ambientes de programación paralela

A. *Nodo*

Nodo es un punto de intersección de dos o mas elementos, dependiendo del paradigma en que se encuentre; un nodo en redes de computadores es un servidor, mientras que para estructuras de datos dinámicas, un nodo es un registro que contiene un dato de interés.

En el ámbito de programación paralela, un nodo puede ser de dos tipos:

Dedicado: Es un computador sin periféricos de salida, es decir no tiene conectado ni el teclado, ni mouse, ni monitor, solamente se encuentra conectado a los demás nodos. Este tipo de nodos solo realizan trabajo del cluster, no hacen ninguna tarea individualmente.

No dedicado: Es un computador con periféricos de salida, los mas importantes son el teclado y el monitor, ademas se encuentra conectado a los demás nodos. En este tipo de nodo se realizan dos tipos de tareas, las relacionadas con el cluster y las tareas individuales del nodo, estas ultimas se realizan como ultima prioridad, por tanto se efectúan en los periodos de reloj libres que dejan las tareas del cluster.

B. Almacenamiento

El almacenamiento en un cluster es un punto muy delicado y a tomarse muy en cuenta, ya que este puede ser compartido o individual en cada nodo. El problema mas importante esta en la sincronización de lectura/escritura así como el tiempo que usa cada nodo ya que el bus es compartido y el acceso es único.

El protocolo mas usado para el acceso al almacenamiento es NFS[1]. En cuanto a hardware, se pueden utilizar discos duros por cada nodo (sistema tipo DAS[2]), o bien un único disco compartido entre los nodos (sistema tipo NAS[3]). El sistema compartido obtiene el acceso a través de los protocolos CIFS, NFS, FTP o TFTP.

C. Sistema Operativo

El sistema operativo debe ser multiproceso, ya que es el que permitirá una gestión eficiente de los recursos en cada nodo, tanto para el balanceo de carga en cada nodo, como para la eficiencia del cluster en su totalidad. Los mas comunes utilizados en clusteres son:

- GNU/Linux
- Unix
- Windows
- Mac OS
- Solaris
- FreeBSD

D. Conexiones de Red

Para un buen rendimiento del cluster la conexión entre los nodos debe ser lo menos compleja posible, para incrementar la velocidad en el intercambio de información. Para ello se utilizan diferentes tecnologías en los adaptadores de red.

Ethernet: Es la mas utilizada en la actualidad, aun así no es la mas eficaz ya que limita el tamaño de paquete, realiza una excesiva comprobación de errores y los protocolos no son muy avanzados, un ejemplo es el protocolo TCP/IP.

La solución a algunos de estos problemas puede ser el uso de Gigabit Ethernet (1 Gbit/s) o mejor aun 10 Gigabit Ethernet (10 Gbit/s), con una latencia de 30 a 100 μs.

Myrinet: Si se desea una red de baja latencia, esta es la elección perfecta ya que llega a tener de 3 a 10 μs, con una velocidad de transferencia de 2 a 10 Gbit/s. Los protocolos sobre esta red son MPICH-GM, MPICH-MX, Sockets-GM y Sockets MX,

InfiniBand: Es la red con mayor ancho de banda, 96 Gbit/s y latencia de 10 μs. Define una conexión entre un nodo de computación y un nodo de I/O. La conexión va desde un HCA[4] hasta un TCA[5]. Se está usando principalmente para acceder a arrays de discos SAS.

SCI: [6]Es una tecnología bastante escalable, se aplica en las topologías de anillo (1D), toro (2D), e hipercubo (3D) sin necesidad de un switch. Se tienen tasas de transferencia de hasta 5,3 Gbit/s y latencia de 1,43 μs.

E. Middleware

Es un software que produce la interacción entre el sistema operativo y las aplicaciones. Mediante middleware se tiene la sensación de que se esta utilizando una ordenador muy potente en lugar de varios comunes. Es el encargado de congelar o descongelar procesos, balancear la carga, etc.

También mediante esta herramienta se pueden conectar mas nodos al cluster y utilizarlos sin tener que realizar una tarea compleja para poder distribuir los procesos con los nuevos nodos. Existen varios tipos de este software algunos ejemplos son: MOSIX, OpenMOSIX, Cóndor, OpenSSI, etc.

F. Protocolos de Comunicación y Servicio

Protocolo es un conjunto de reglas que permiten que dos o mas entidades se comuniquen para transmitir cualquier tipo de información por medio de un enlace físico.

Por ejemplo, entre un nodo y el sistema de almacenamiento común se puede utilizar el protocolo FTP, entre nodo y nodo se puede utilizar el protocolo UDP y entre el cluster y el cliente se puede utilizar el protocolo TCP/IP.

G. Aplicaciones

En la actualidad existen un gran número de clusteres implementados, los ejemplos más sobresalientes son los servidores de Google, Facebook, Amazon, tareas: etc. Las aplicaciones son muchas y son variadas, se pueden realizar las siguientes

- Predicciones Meteorológicas
- Predicciones Bursátiles
- Simulaciones de Comportamiento Cinemático y Dinámico de componentes mecánicos
- Diseño y análisis de nuevo materiales
- Estudio de fármacos y enfermedades epidémicas

[1] Network File System
[2] Data Analytics Supercomputer
[3] Network Attached Storage
[4] Host Channel Adapter
[5] Target Channel Adapter
[6] Scalable Coherent Interface

- Diseño de Estructuras moleculares
- Creación y renderización de fotogramas de animación
- Investigación en general
- etc

H. Ambientes de Programación Paralela

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

IV. Diseño

A. Hardware

El cluster diseñado es del tipo Beowulf, este es un sistema de cómputo paralelo basado en clusters de ordenadores personales conectados a través de redes informáticas estándar, sin el uso de equipos desarrollados específicamente para la computación paralela. Las características son las mismas en las cuatro PC's y son las siguientes:

Procesador: Intel® Pentium® D CPU @ 3.40 Ghz

RAM de Cache L2: 2 MB

Velocidad de Bus: 800 Mhz

Memoria RAM: 512 MB

Velocidad de Memoria: DDRII @ 333 Mhz

B. Sistema Operativo

Figura 5: S.O. Ubuntu

Es una distribución de Ubuntu para servidores basada en Debian. La diferencia con el anterior sistema es que en este se deben instalar todos los paquetes y configurar cada nodo para funcionar como parte del cluster, diferenciando el nodo maestro de los nodos esclavos.

C. Almacenamiento

El almacenamiento del cluster es el disco duro que se halla en el nodo maestro, este disco tiene una capacidad de 150 Gb @ 133 Mbps bajo la tecnología IDE[7] o ATA[8].

El nodo maestro comparte una carpeta con los nodos esclavos mediante la tecnología libre NAS[9] mediante el protocolo NFS[10].

1) Protocolo NFS

Es un protocolo de nivel de aplicación, de acuerdo con el modelo OSI[11], este protocolo hace posible que un nodo acceda a los archivos de otro como si fueran propios del mismo mediante una conexión de red, este protocolo esta incluido por defecto en los sistemas UNIX y en algunos sistemas Linux.

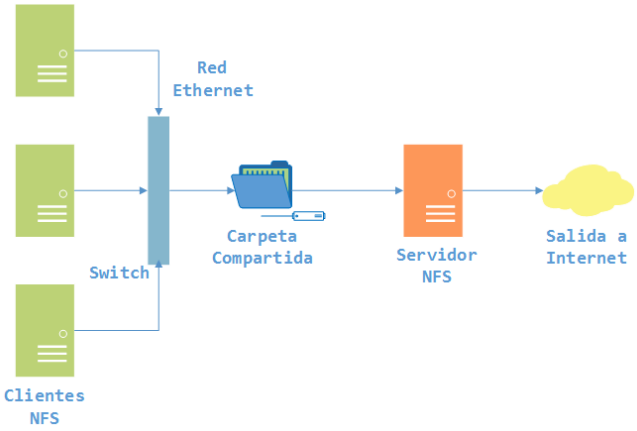

Figura 6: Sistema de Archivos en Red

D. Conexión de Red

Ethernet es el medio más utilizado por su bajo costo y su fácil instalación, esta tecnología es solo recomendable para propósitos de estudio ya que cuenta con detección de colisiones CSMA/CD[12] que incrementa la latencia y reduce la velocidad de la obtención de resultados; esta basado en el estándar IEEE 802.3.

Figura 7: Cable UTP y Conector RJ45

Las direcciones IP's se designan de la siguiente manera:

[7] Integrated Device Electronics
[8] Advanced Technology Attachment
[9] Network Attached Storage
[10] Network File System
[11] Open System Interconnection
[12] Carrier Sense Multiple Access with Collision Detection

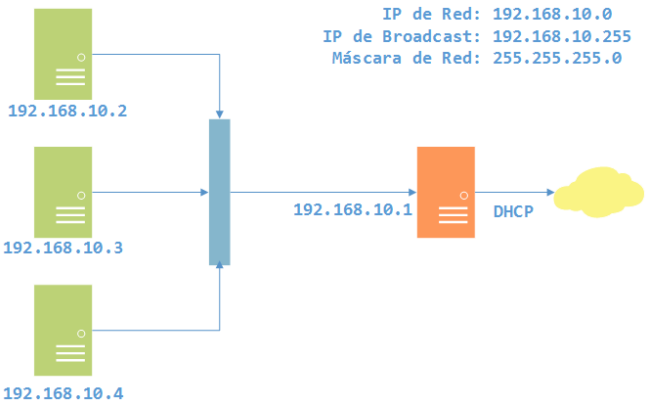

Figura 8: Parámetros de la Red

E. Middleware

En el desarrollo del clúster se utilizó el software SSH[13], este software brinda la posibilidad de conectarse desde el nodo maestro a los nodos esclavos. Una vez que se ha compartido una carpeta o un sistema de archivos se conecta mediante SSH y se inicia un proceso en cada nodo, todo desde el nodo maestro.

Figura 9: Identificación mediante llaves SSH

SSH trabaja de forma parecida a telnet, pero utiliza un cifrado para que la información no sea legible, con la única manera de acceder a esta esta información es por medio de ataques de REPLAY[14].

F. Administrador de Procesos

1) OpenMPD

MPD es un administrador de procesos compatible con MPI hasta la versión 1.2. MPD trata de evitar el envío de procesos a los nodos esclavos, solo cuando el nodo maestro no es suficiente para ejecutar un proceso, entonces es que MPD comienza a mandar tareas, pero lo hace hasta que el siguiente nodo tampoco abastezca; se observa que MPD es eficiente para tareas que necesiten de cálculos muy largos.

[13] Secure SHell
[14] http://es.wikipedia.org/wiki/Ataque_de_REPLAY

2) Hydra

Hydra es un administrador de procesos compatible con MPI desde la versión 1.3. Hydra al contrario de MPD trata de usar procesadores alejados al nodo maestro, tanto para reducir la carga a este, como para tener mas núcleos activos y usar todos los periodos de clock para realizar los cálculos en todos los núcleos Hydra es eficiente cuando los procesos no son muy pesados y cuando se tiene una gran cantidad de datos.

G. Ambiente de Programación Paralela

Para poder ejecutar programas se necesita de un entorno, en nuestra investigación utilizamos dos diferentes.

1) MPI

MPI[15], es un estándar de paso de mensajes que define la sintaxis y la semántica de las funciones de una librería en un lenguaje en específico, en el cual se diseñan programas que explotan a los multiprocesadores o bien a un sistema multinúcleo. MPI tiene soporte para C, C++, C#, Fortran, Ada, Python, OCaml, Java y código ensamblador.

2) Octave

GNU Octave es un software libre enfocado a los cálculos matemáticos. Es el equivalente al software privativo de Mathworks® MATLAB®. Este software ejecuta scripts que sean compatibles con MATLAB®, es decir que es compatible con archivos de extensión *.m*, es un lenguaje interpretado.

Actualmente octave tiene muchas funciones nativas para el procesamiento paralelo, entre algunas se puede nombrar $parfor$, que básicamente es un bucle for pero que utiliza varios procesadores para realizar la ejecución. Un ejemplo práctico es el siguiente:

- Bucle FOR:

```
for i = 1:8
A(i) = i;
end
```

Salida del procesador: *1, 2, 3, 4, 5, 6, 7, 8*

Si el procesador debe realizar 8 procesos, tomando en cuenta que cada proceso toma $10\mu s$ el tiempo total es de:

$$10\mu s * 8 = 80\mu s$$

[15] Message Passing Interface

- Bucle PARFOR:

```
parfor i = 1:8
A(i) = i;
end
```

Salida del procesador 1: *1, 2*
Salida del procesador 2: *3, 4*
Salida del procesador 3: *5, 6*
Salida del procesador 4: *7, 8*

En este caso tomando en cuenta que cada procesador realiza 2 procesos y que cada proceso toma 10μs el tiempo total es de:

$$10\mu\text{s} \times 2 = 20\mu\text{s}$$

Claramente en el ejemplo anterior se nota que el tiempo se reduce a $\frac{1}{4}$ utilizando 4 núcleos que solamente usando 1 núcleo. Y como este ejemplo existen muchos otros pero este tema sera tratado en los siguientes capítulos.

V. Resultados

A. *Benchmark*

```
HPC Test - - - - - - - - - - - - - - - - -
Quantity of processors = 4
Calcultaion time       = 1.10 seconds
Cluster speed          = 1636 MFLOPS
- - - - - - - - - - - - - - - - - - - - -
Cluster node N00 speed = 409 MFLOPS
Cluster node N01 speed = 409 MFLOPS
Cluster node N02 speed = 409 MFLOPS
Cluster node N03 speed = 409 MFLOPS
- - - - - - - - - - - - - - - - - - - - -
```

Los resultados anteriores indican claramente que el número de operaciones de punto flotante por segundo que puede realizar el cluster es de 1636×10^6, esto es 4 veces mayor al número de operaciones de un simple nodo.

B. *Cálculo de Pi*

El primer caso de estudio es el cálculo de pi con programación paralela. Pi es una razón. Es la respuesta a la pregunta: ¿como se relaciona la distancia a través de un circulo (el diámetro con la distancia a su alrededor)?. Durante milenios se ha sabido que las dos medidas de un círculo están relacionadas. El reto estaba en descubrir como.

Pi es irracional. Su valor se parece a 22/7, pero como ocurre con todos los números irracionales, ninguna fracción puede describirlo perfectamente, y la expresión decimal continuará para siempre sin repeticiones. Luego, como Phi, y e, pi es una constante natural que es imposible conocer completamente. Eso no ha impedido que muchos lo intentaran.

El método Monte-Carlo[16] trata de calcular un valor aproximado de pi, lanzando dardos sobre la diana representada en la siguiente figura.

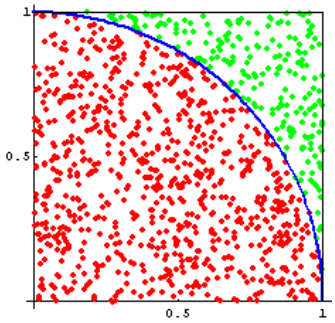

Figura 10: Cálculo de Pi por el método Monte-Carlo

Supongamos que los dardos se reparten uniformemente, entonces la probabilidad de que un dardo caiga en el cuadrante del circulo es:

$$P = \frac{\text{Área del cuadrante}}{\text{Area del cuadrado}}$$

$$P = \frac{\pi}{4}$$

Si lanzamos N dardos sobre el cuadrado, y sea M el número de dardos que caen en el cuadrante. La frecuencia relativa de caída en el cuadrante $\frac{M}{N}$, será aproximadamente igual a $\frac{\pi}{4}$. Por tanto:

$$\pi = \frac{4 \times M}{N}$$

Se utilizó este algoritmo en diferentes ambientes. El balanceador de trabajos que se utilizó es OpenMPD, este evita el retardo que pueda provocarse en la red del cluster; lo que hace es enviar trabajos a los nodos más lejanos siempre y cuando los más cercanos estén ocupados.

Es decir, que si el nodo maestro se configuró para trabajar en conjunto con el cluster y definimos dos procesos, tomando en cuenta que cada nodo tiene dos procesadores, el trabajo se realizara solo en el nodo maestro.

Si definimos tres procesos, el balanceador verifica que dos de los procesos serán realizados en el nodo maestro y enviara el proceso sobrante al nodo más cercano al maestro. Con ello se puede afirmar que solo cuando se definan siete o más procesos es que las cuatro computadoras del cluster trabajaran en conjunto.

Por ello es que este administrador es muy recomendable cuando se utiliza una infraestructura tipo Beowulf. El otro ambiente o condición es cuando se define que el nodo maestro no trabaje en conjunto con el cluster. Esto

[16] http://en.wikipedia.org/wiki/Monte_Carlo_method

tiene un gran efecto como se verifica más adelante, ya que al tener al nodo maestro solo como emisor de trabajos y receptor de resultados se gana un tiempo importante y el proceso se culmina en menor tiempo que cuando este trabaja.

Por otra parte no es estrictamente limitante el número de procesos, aunque se tienen solamente 8 núcleos de trabajo en total, se pueden asignar el número de procesos que se desee a cada uno pero esto tiene un defecto ya que la red tiene un retardo al no contar con conexión de fibra óptica. Por ello se debe hallar un balance entre el número de procesos a asignar de acuerdo al trabajo que se necesite realizar.

El algoritmo de la figura 11 que muestra el proceso de cálculo de Pi en paralelo es el siguiente:

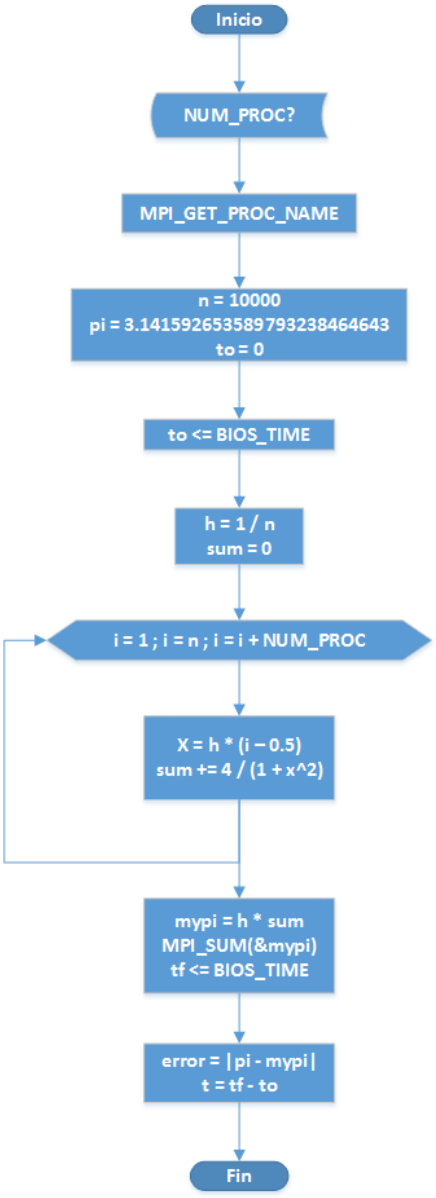

Figura 11: Algoritmo para el cálculo de Pi

Se tienen las siguientes constantes:

$$\pi = 3,41592653589793$$

$$N = 10000$$

1) Procesamiento con el maestro incluido

PC	Procesos	$t_1[\mu s]$	$t_2[\mu s]$	$t_3[\mu s]$	$t_4[\mu s]$	$t_5[\mu s]$	$t_{promedio}[\mu s]$	Error $[\times 10^{-12}]$
1	1	6887	6932	6898	6909	6909	6907	8.333410
1	2	1205	1172	1099	1219	1202	1179.4	8.333387
2	3	215	249	284	168	189	221	8.333387
2	4	252	209	244	209	256	234	8.333307
3	5	121	157	113	95	102	117.6	8.333298
3	6	95	118	124	96	97	106	8.333312
4	7	61	52	43	95	51	60.4	8.333312
4	8	62	67	71	66	52	63.6	8.333312

Tabla 1: Tiempos para cálculo de Pi con nodo maestro

Se puede notar una gran diferencia en los tiempos de cálculo.

$$t_{max} = 6932\mu s \quad (1 \text{ Proceso } / \text{ 1 PC})$$

$$t_{min} = 43\mu s \quad (7 \text{ Procesos } / \text{ 4 PC})$$

$$Incremento = 161{,}21X$$

2) Procesamiento sin el nodo maestro

PC	Procesos	$t_1[\mu s]$	$t_2[\mu s]$	$t_3[\mu s]$	$t_4[\mu s]$	$t_5[\mu s]$	$t_{promedio}[\mu s]$	Error $[\times 10^{-12}]$
1	1	4899	4899	4888	4905	4911	4900.4	8.333410
1	2	1022	1157	1164	1187	1385	1183	8.333387
2	3	737	797	799	743	731	761.4	8.333387
2	4	296	272	254	270	234	265.2	8.333307
3	5	144	195	189	192	137	171.4	8.332980
3	6	147	198	110	135	110	140	8.333312
4	7	50	31	41	41	41	40.8	8.333307
4	8	27	38	32	29	26	30.2	8.333321
4	9	20	20	22	20	16	19.6	8.333325
4	10	16	22	20	16	22	19.2	8.333321
4	11	20	20	20	16	19	19	8.333319
4	12	15	13	10	17	13	13.6	8.333325
4	13	15	10	13	11	13	12.4	8.333334
4	14	14	13	15	14	13	13.8	8.333343
4	15	9	11	13	10	12	11	8.333343
4	16	11	11	8	11	10	10.2	8.333343

Tabla 2: Tiempos para cálculo de Pi sin nodo maestro

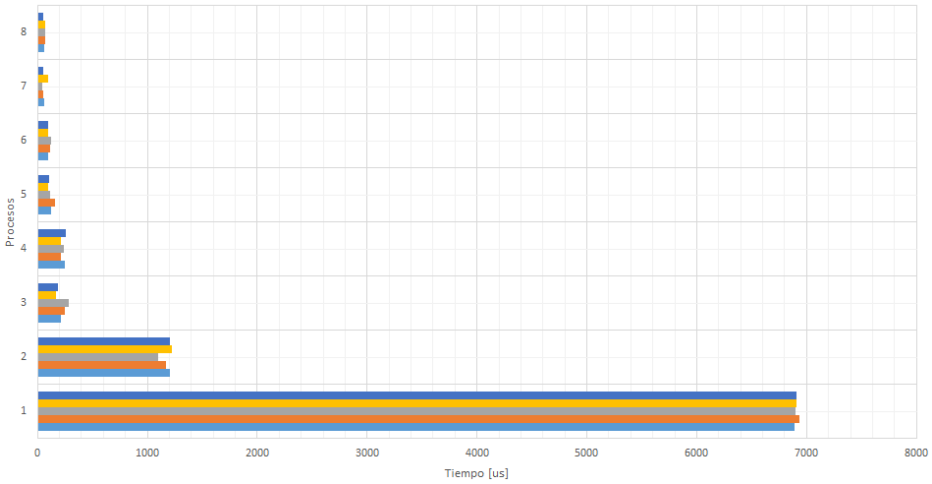

Figura 12: Diferencia de tiempo en 5 tomas

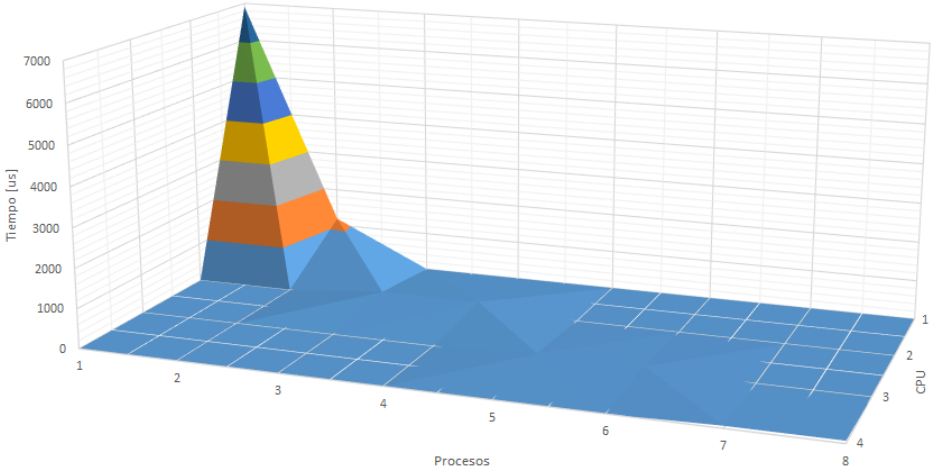

Figura 13: Comparación de tiempo con respecto a procesos y CPU's

Nuevamente se puede notar una gran diferencia en los tiempos de cálculo.

$$t_{max} = 4911\mu s \quad \text{(1 Proceso / 1 PC)}$$

$$t_{min} = 8\mu s \quad \text{(16 Procesos / 4 PC)}$$

$$Incremento = 613{,}88X$$

El aparente incremento de tiempo al usar el nodo maestro tiene una explicación. Cuando el nodo maestro trabaja conjuntamente con el cluster, este realiza 3 trabajos de fondo. Primero envía los trabajos a cada nodo del cluster, después debe realizar la cantidad de trabajos que le son asignados como a cualquiera de los nodos esclavos, y por último debe recibir los resultados de todos los demás nodos e ir sumando parcialmente hasta llegar al resultado final.

Es por ello que cuando el nodo maestro no trabaja junto con el cluster y solo se dedica a enviar trabajos y recibir resultados se obtienen mejores resultados.

C. Estimador de Regresión Nadaraya-Watson

El estimador de Nadaraya-Watson es uno de los mecanismos de Regresión no paramétrica más prestigiosos. Usa un método Kernel de estimación de funciones de densidad. Un Kernel muy usual es la distribución Normal. Por ejemplo una N(0, 1).

Al parámetro h se le denomina ventana y, en realidad, modifica la dispersión de la Normal, si es ésta la que actúa de Kernel. Es una forma original de construir una estimación de la variable dependiente "y" a partir de un valor de una variable independiente "x", basándose exclusivamente en la posición de los valores de la muestra que tenemos.

Se trata de un mecanismo de construcción de la variable "y" ponderando los valores muestrales de esta variable según la distancia que haya desde el valor de "x" a los valores muestrales de la variable independiente. La ponderación se materializa mediante el numerador del Kernel. Supongamos que éste sea la N(0, 1), entonces si

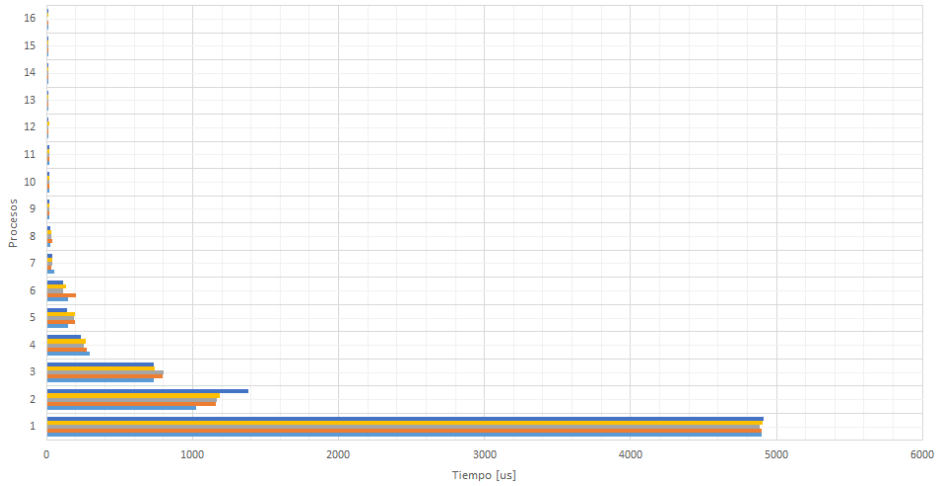

Figura 14: Diferencia de tiempo en 5 tomas

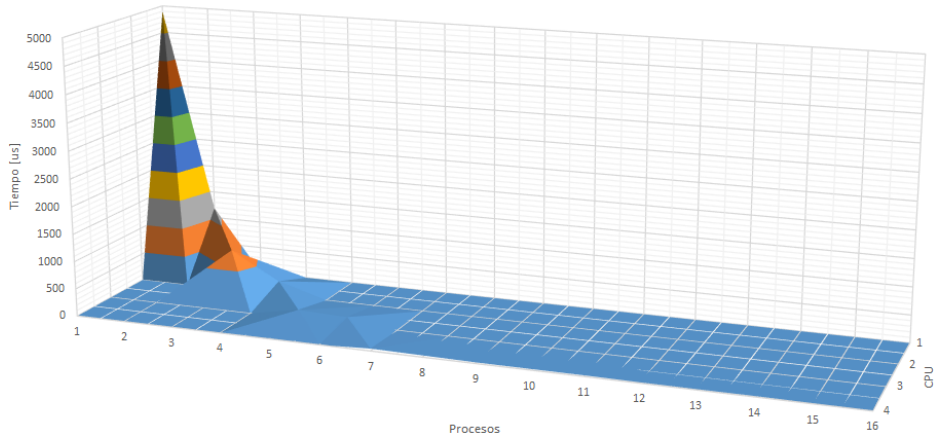

Figura 15: Comparación de tiempo con respecto a procesos y CPU's

el valor de "x" está cerca de un valor muestral de la variable independiente la resta será un valor próximo a cero y tendrá en la Normal un valor grande.

Sin embargo, los valores alejados darán restas grandes en valor absoluto y en la Normal tendrá un valor próximo a cero. Observemos, pues, que el valor de "y" para esa "x" estará muy influido por los valores muestrales cercanos.

$$\hat{g}(x) = \frac{\sum_{t=1}^{n} y_t K[x - x_t/\gamma_n]}{\sum_{t=1}^{n} K[x - x_t/\gamma_n]}$$

$$\hat{g}(x) = \sum_{t=1}^{n} w_t y_n$$

Lo que vemos es que el peso depende de cada punto en la muestra. Para calcular el ajuste en cada punto de la muestra de datos de tamaño n, en el orden de n^2k cálculos realizados, donde k es la dimensión del vector descriptivo de variables x.

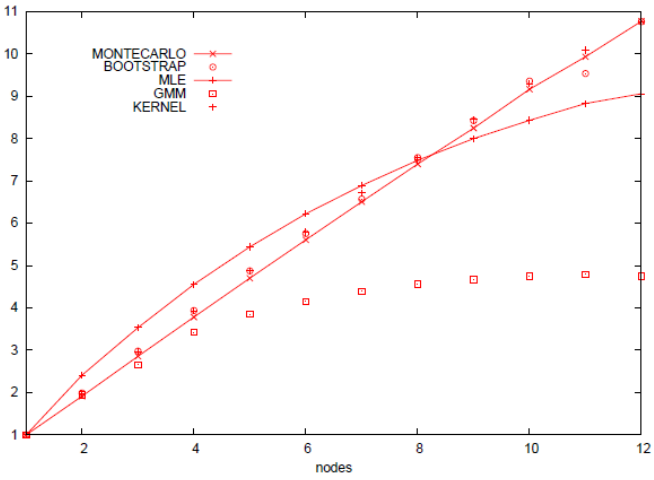

Figura 16: Aceleración mediante paralelización

Racine[17] demuestra que la paralelización MPI puede

[17] Jeffrey Scott Racine - Journal of Applied Econometrics - Abril

ser usada para acelerar el cálculo del estimador de regresión del Kernel, por medio del cálculo de ajustes de porciones de la muestra en diferentes computadoras. La figura 16 muestra el incremento en la velocidad de cálculo para problemas de econometría en un cluster de 12 computadoras.

El incremento para k nodos es el tiempo para finalizar el problema en un solo nodo dividido entre el tiempo para finalizar el problema en k nodos. Se obtiene un incremento de 10X.

Como se aprecia en el gráfico, la regresión Kernel es la que incrementa la velocidad de cálculo, seguida por Montecarlo, Bootstrap, MLE y GMM, en ese mismo orden.

Este algoritmo es el que se uso para la investigación con el soporte del software Octave en tres diferentes condiciones.

1) Cálculo solo en el nodo maestro

Tiempo [s]	**Datos**				
Procesos	1000	2000	5000	7000	10000
2	0,908	2,421	13,821	106,074	264,369
3	0,467	1,201	5,312	10,736	197,566
4	0,322	0,811	3,464	11,979	100,989
5	0,245	0,605	2,606	5,135	45,85
6	0,203	0,51	2,266	4,711	32,678
7	0,176	0,421	1,922	3,995	31,808
8	0,157	0,372	1,646	3,591	18,17

Tabla 3: Tiempos de cálculo utilizando solo el nodo maestro

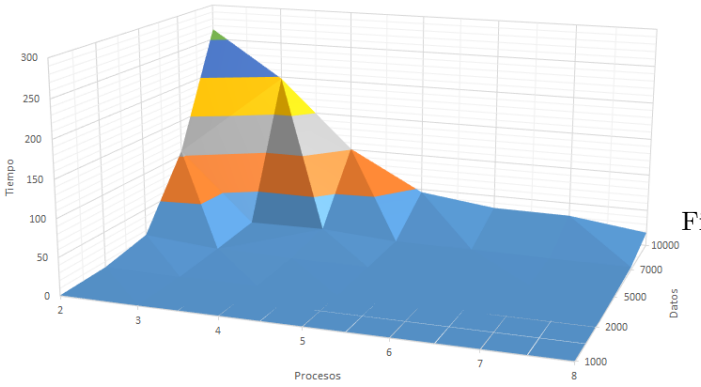

Figura 17: Tiempos de cálculo utilizando solo el nodo maestro

$t_{max} = 264{,}369\text{s}$ (2 Proceso / 10000 Datos)

$t_{min} = 0{,}157\text{s}$ (8 Procesos / 1000 Datos)

2) Cálculo utilizando todo el cluster

Tiempo [s]	Datos				
Procesos	1000	2000	5000	7000	10000
2	0,889	2,292	10,585	23,027	15,38
3	0,461	1,156	2,087	10,2	45,155
4	0,317	0,769	3,313	6,648	22,571
5	0,24	0,588	2,482	4,873	11,082
6	0,19	0,473	2,069	4,263	8,149
7	0,169	0,391	1,716	3,704	10,501
8	0,146	0,349	1,507	3,045	8,486

Tabla 4: Tiempos de cálculo utilizando todo el cluster

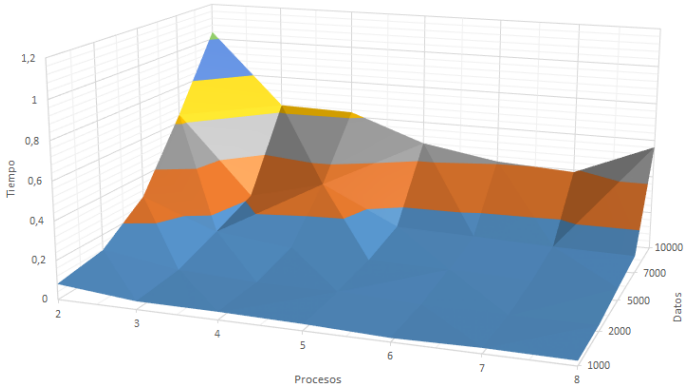

Figura 18: Tiempos de cálculo utilizando todo el cluster

$t_{max} = 45{,}155\text{s}$ (3 Proceso / 10000 Datos)

$t_{min} = 0{,}146\text{s}$ (8 Procesos / 1000 Datos)

2012

3) Cálculo utilizando solo los nodos esclavos

Tiempo [s]	Datos				
Procesos	1000	2000	5000	7000	10000
2	0,08	0,119	0,286	0,643	1,038
3	0,042	0,059	0,133	0,213	0,636
4	0,041	0,041	0,09	0,318	0,626
5	0,034	0,043	0,067	0,119	0,48
6	0,02	0,028	0,064	0,109	0,411
7	0,028	0,025	0,051	0,097	0,388
8	0,025	0,031	0,063	0,092	0,563

Tabla 5: Tiempos de cálculo utilizando solo los nodos esclavos

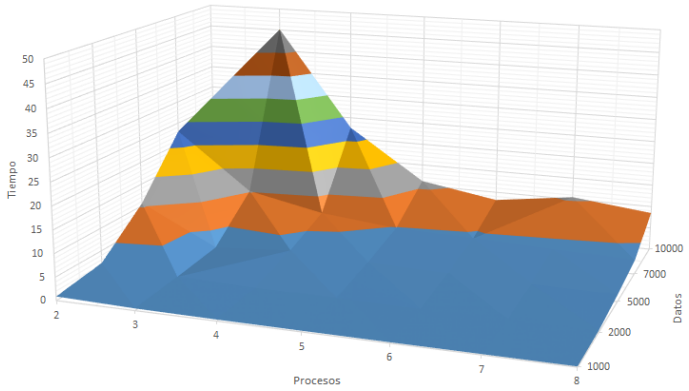

Figura 19: Tiempos de cálculo utilizando solo los nodos esclavos

$$t_{max} = 1{,}038\text{s} \quad \text{(2 Proceso / 10000 Datos)}$$

$$t_{min} = 0{,}02\text{s} \quad \text{(6 Procesos / 1000 Datos)}$$

VI. Conclusiones

- Se comprobó que el benchmarck del cluster es la suma algebraica de los benchmark de cada uno de los nodos que constituyen el cluster.
- Se obtuvo un incremento muy grande en el tiempo de cálculo en los programas con los que se realizó esta investigación.
- Se debe tener cuidado al definir si el nodo maestro trabajará en conjunto con el cluster, ya que de ser así se producen dos tipos de resultados.
  - Cuando el nodo maestro también trabaja se producen colas en la red y esto hace que el tiempo de total se incremente ya que el nodo maestro recibe los resultados de los nodos esclavos solamente después de acabar con su propio trabajo. Pero dado que los cálculos se han realizado en una cantidad mayor de procesos se obtiene un error menor.
  - Cuando el nodo maestro no trabaja con el cluster, solo se encarga de enviar trabajos y recibir resultados, con lo que la red no se encuentra congestionada ya que cada nodo envía sus resultados al maestro y este comienza a procesarlos inmediatamente, y se encuentra en la capacidad de enviar nuevamente trabajos a los nodos que hayan finalizado para que cada nodo del cluster se encuentre activo en todo momento. Dado que los cálculos se realizan en menor cantidad de procesos se obtiene un error mayor, pero este error se puede subsanar enviando una mayor cantidad de paquetes a cada procesador de forma asíncrona.
- Dado que la conexión de red es el medio de comunicación entre los nodos, y gracias a que se utilizó conexión con cable de par trenzado, se puede concluir que el tiempo de proceso se puede disminuir aún mucho más si se utilizara otro tipo de tecnología como fibra óptica.
- Se vió que existen diversas utilidades y librerías que contienen herramientas para programar en paralelo, y no estrictamente obligatorio que estos se ejecuten en un cluster ya que las computadoras actuales tienen dos o más núcleos y dichos procesadores tienen la capacidad para correr los mismos programas, incluso con menor tiempo en ejecución ya que tienen un bus interno compartido entre los núcleos para el intercambio de datos. Es decir muchos de los programas que utilizamos en la vida cotidiana se están ejecutando en paralelo sin que nosotros nos percatemos de ello.
- Por otra parte se puede ver que cuando un programa se ejecuta en muchos núcleos de baja frecuencia, tiene un tiempo de ejecución parecido al mismo programa ejecutado en un solo procesador de alta frecuencia, pero a se puede mencionar a favor de la programación paralela que muchos núcleos de baja frecuencia son mucho mas baratos en conjunto que un solo núcleo de alta frecuencia. Esta es una de las razones por las cuales se dió el cambio de tecnología de los procesadores mononúcleo a multi-núcleo.
- Por último se concluye que cualquier programa optimizado para correr en un ambiente paralelo termina en mucho menor tiempo que en un ambiente serial.

Referencias

[1] Michael R Anderberg. *Cluster analysis for applications*. Inf. téc. DTIC Document, 1973.

[2] Rajkumar Buyya. "High performance cluster computing". En: *New Jersey: F'rentice* (1999).

[3] Daniel Jiménez González. *Multiprocesadores y multicomputadores*. Espa/nol. Inf. téc. Universitat Oberta de Catalunya, 2010.

[4] Landmann. *Red Hat Enterprise Linux 6 Visión general de Cluster Suite*. Espa/nol. Red Hat. 2010.

[5] Thomas Lawrence Sterling. *Beowulf cluster computing with Linux*. MIT press, 2002.

[6] Thomas Lawrence Sterling. *Beowulf Cluster Computing with Windows*. MIT Press, 2002.

[7] Peter Strazdins y John Uhlmann. "A comparison of local and gang scheduling on a beowulf cluster". En: *Cluster Computing, 2004 IEEE International Conference on*. IEEE. 2004, págs. 55-62.