Construcción e Implementación de un Clúster con máquinas PCs recicladas

César Martín Cruz Salazar† y José A. Fiestas‡
CTIC(Centro de Tecnologías de Información y Comunicaciones).
Universidad Nacional de Ingeniería;
†*ccruz@uni.edu.pe*
‡*fiestas@ari.uni-heidelberg.de*

Recibido el 3 de Octubre de 2014; aceptado el 5 de Noviembre de 2014

En este trabajo se presenta la implementación de un clúster de Computadoras para dotar a CTIC UNI con una Máquina Computacional de Alto Rendimiento, usando como base de hardware computadoras personales recicladas, de bajo costo y software de código abierto como LINUX. Asimismo, se describe la arquitectura del clúster de máquinas usada para la implantación del clúster CTIC. Finalmente, se presenta la evaluación del clúster implementado y pruebas de rendimiento utilizando sofware escalable de última generación.
Palabras Claves: LINUX clúster, Computación Paralela, Computación Distribuida, Cómputo Grupal, Computación de Alto Rendimiento.

We present in this paper the implementation of a cluster of computers, with the aim to provide to CTIC UNI with a High Performance Computing Machine, using as a hardware basis, inexpensive, recycled personal computers and open source software such as Linux. Moreover, the architecture of the cluster machines used for the implementation of cluster CTIC is described. Finally, evaluation of the implemented cluster and benchmarks are done using scalable, last generation software.
Keywords: LINUX Cluster, Parallel Computing, Distributed Computing, Cluster Computing, High Performance Computing.

1 Introducción

En la UNI, hemos querido desde hace mucho tiempo tener un clúster para que se utilice en los proyectos de investigación sobre HPC (High Performance Computing) llevados a cabo en la universidad. Se hicieron muchos intentos que fracasaron hasta que al fin se tuvo la idea de desarrollar un clúster con máquinas PCs (Personal Computers) recicladas aprovechando la experiencia ganada el año pasado al desarrollar un clúster con minicomputadores Raspberry Pi de bajo costo [1] . Existe una demanda continua de velocidad en los sistemas computacionales. Esta demanda de velocidad en general es requerida en áreas como cálculo numérico, simulación y modelado numérico de problemas científicos, procesamiento de imágenes y datos en general, en gran variedad de aplicaciones de ingeniería y ciencias básicas. Usualmente dar solución a problemas en estas áreas implica numerosos y repetitivos cálculos matemáticos sobre un conjunto de datos, que usando técnicas de programación paralela, se pueden resolver en un tiempo aceptable. Las plataformas computacionales necesarias para abordar la resolución de estos problemas son las llamadas Supercomputadoras, que poseen características especiales de hardware que las hacen, en general, extremadamente costosas y por ende, difíciles de adquirir y de mantener actualizadas, tanto en su plataforma de hardware como de software, por entes como las universidades públicas peruanas. Con los avances tecnológicos en diseño y construcción de microprocesadores de bajo costo, como los usados en las computadoras actuales, hoy día, es posible contar con gran capacidad de cómputo en una PC.

2 Descripción y Antecedentes

2.1 Clúster

Un clúster de computadoras es simplemente un conjunto de máquinas interconectadas entre sí. En este sentido, el clúster no se diferencia de una red de computadoras. Lo que distingue al clúster de una red es que todos los elementos de procesamiento que lo conforman se dedican a la solución de un problema computacional complejo utilizando programación paralela. En la actualidad, se cuenta con la tecnología necesaria para construir un clúster de máquinas capaz de competir con algunas supercomputadoras, en cuanto a su capacidad de cómputo y almacenamiento, a un costo sustancialmente menor usando como elementos de procesamiento a PCs o estaciones de trabajo de bajo costo. De acuerdo a su finalidad, se pueden definir tres diferentes tipos de clúster.

- Sistemas Tolerantes a Fallas
- Sistemas de Alta Disponibilidad
- Sistemas de Cómputo de Alto Rendimiento

2.2 Arquitectura Beowulf

En 1994, la Agencia Espacial NASA comenzó el Proyecto Beowulf [2] en el Centro para la Excelencia en Datos Espaciales y Ciencias de la Información (CESDIS), cuyo resultado fue la construcción de un Clúster de 16 máquinas

destinado al Proyecto de Ciencias Espaciales y Terrestres (ESS) ubicado en el Centro de Vuelo Espacial de Goddard (GSFC). La idea de construir un sistema con elementos de hardware de bajo costo, para satisfacer requisitos computacionales específicos, se difundió rápidamente a través de la NASA y las comunidades académicas y científicas a escala mundial. Un clúster del tipo Beowulf posee una arquitectura escalable de múltiples computadoras, que puede ser usada para realizar cómputo paralelo y distribuido. Estos Clústeres son sistemas construidos con componentes de hardware y software de uso general, es decir, no contienen ningún tipo de hardware especializado. Los nodos están constituidos por cualquier computadora capaz de ejecutar el Sistema Operativo LINUX, y software para el desarrollo de aplicaciones paralelas. En su forma general, un clúster Beowulf consiste de un nodo Maestro (frontend) y uno o varios nodos de cómputo (compute node) llamados nodos Esclavos, interconectados a través de una LAN (Local Area Network). El nodo Maestro controla los nodos Esclavo y también es la consola y "puerta de entrada" al clúster desde el mundo exterior. Los nodos esclavos del sistema se usan sólo para cómputo dentro del sistema. La madurez alcanzada por el Sistema Operativo LINUX y su robustez, la estandarización de librerías GNU para el "paso de mensajes" como MPI (Message Passing Interface), garantizan a los programadores que las aplicaciones que desarrollen, se ejecutarán en futuras versiones de estos elementos de software y por ende en clústeres del tipo Beowulf actualizados, independientemente de la plataforma de hardware. En la figura 1, se muestra la interconexión básica de los nodos en un clúster del tipo Beowulf.

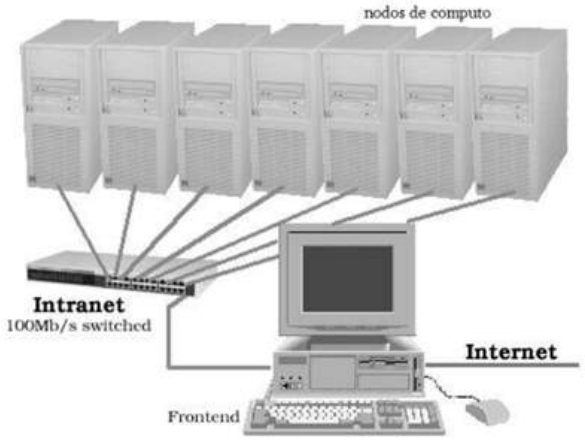

Figura 1. *Interconexión de los nodos en una arquitectura Beowulf.*

Los clústeres Beowulf se comportan como una unidad. En la mayoría de los casos, los nodos cliente no poseen teclados, ratones (mouse) o monitor, y son accedidos desde el nodo Maestro. Así, los nodos Esclavos, representan un componente que posee una Unidad Central de Procesamiento para cómputo y cierta cantidad de memoria que se anexa al sistema. No existe necesidad de que los nodos Esclavos sean accedidos directamente por sistemas externos, ni que éstos accedan al exterior. Por esta razón, los nodos Esclavos forman una red privada (intranet) conectada al nodo Maestro. Para cumplir con esta característica, el nodo Maestro posee una interfaz de red NIC (Network Interface Card) que permite el acceso y uso del clúster desde cualquier estación de trabajo externa. Los usuarios del sistema acceden al nodo Maestro, bien sea desde la consola o vía telnet, secure shell (ssh), etc., donde pueden editar y compilar sus aplicaciones y ejecutar su código en los nodos Esclavos del clúster.

3 Descripción del sistema

En este proyecto, comenzamos recopilando información de la biblioteca y de la Web sobre manuales y libros que nos ayuden a configurar un clúster. Para la configuración utilizamos información diversa pero sobre todo la experiencia ganada anteriormente [1]. Proporcionamos una descripción de la arquitectura del clúster, tanto en términos de sus componentes de hardware y de su entorno de software.

3.1 Especificaciones de Máquinas PCs empleadas

Se emplearon 27 máquinas PCs del año 2008 igual al como aparece en la figura 2.

Figura 2. *Máquina PC utilizado en el Clúster CTIC.*

Cada PC tiene un procesador Core 2 Duo de 2.33 Gigahertz de 2 núcleos, 1 Gigabyte de RAM y 320 gigabytes de disco duro. El clúster-CTIC construído se muestra en la figura 3.

Figura 3. *Clúster CTIC construído con PCs Core 2 Duo de 2.33 Gigahertz de 2 núcleos, 1 Gigabyte de RAM y 320 gigabytes de disco duro.*

3.2 Software y Configuración del Clúster

El software instalado es UBUNTU 14.04. Se confeccionó un manual de configuración basado en los documentos y libros consultados en la web y en la biblioteca y este se utilizó a lo largo de toda la configuración del Clúster [3], [4].

3.2.1 Cambios realizados en las PCs

- Se retiro de las lectoras de DVD, las disqueteras y tarjeta modem de las PCs que conforman el clúster, para lograr mayor ventilación al interior de las PCs y menos consumo de energía. Con esto también pensamos lograr mayor estabilidad en las PCs.

- Se quitaron todas las interfaces gráficas de los esclavos del 1 - 26, luego apagamos, desconectamos los enchufes y volvimos a conectar, logrando que prendieran todos los esclavos a la primera, cosa que nunca había pasado. Eliminando la interfaz gráfica a los esclavos se logró reducir la memoria usada a casi la tercera parte, pasando de un aproximado de memoria usada con interfaz gráfica de 390 Mb a 130 Mb de memoria usada sin interfaz gráfica de un total de 960 Mb.

- Instrucciones que utilizamos para quitarle la interfaz gráfica:

  – Modificar el archivo

    `sudo nano /etc/default/grub`

  – En el archivo cambiar la línea:

    `GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"`

    por:

    `GRUB_CMDLINE_LINUX_DEFAULT="text"`

  – Guardar el archivo

  – Actualizamos el grub:

    `sudo update-grub`

  – Reiniciamos el computador. El computador prenderá normalmente sin interfaz gráfica y si se desea abrir el terminal se debe pulsar Ctrl + Alt + F1.

3.3 Problemas que se tienen con las PCs

En el clúster se cuelgan algunos nodos Esclavos. Al momento de escribir este artículo todavía se esta intentando resolver este problema.

3.4 Como montamos la ruta compartida de trabajo(/mnt/cctic) en cada una de las PCs(nodos) del Clúster

- Editar el archivo: /etc/fstab en cada nodo. [3], [5] Esto montará automáticamente el directorio /mnt/cctic al iniciar el sistema. Agregar la línea (al final del archivo):

```
master2:/mnt/cctic/  /mnt/cctic/  nfs
defaults 0 0
```

- En donde: El primer elemento es el directorio fuente (nótese la sintaxis servidor:directorio). El segundo es el directorio donde se montará en el nodo. El tercero indica el tipo de sistema de archivos. El cuarto indica parámetros de montaje. Los dos últimos indican el orden en que se debe respaldar el sistema de archivos.

- Después aplicar: sudo mount -a en cada nodo, para que se monten todos los dispositivos, incluyendo la última modificación en fstab.

- Por último, reiniciar los nodos.

3.5 Velocidad de cómputo del Clúster

Se ejecutó un programa para el cálculo del valor de pi(3.14...) con 800 millones de rectángulos. Usando el Clúster CTIC con 54 nodos, este cálculo se hizo en un tiempo de 0.56 segundos como se muestra en la Figura 4.

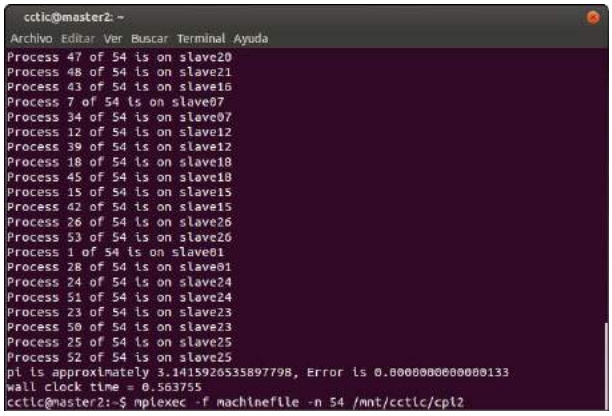

Figura 4. *Ejecución de un programa para el cálculo de pi en el Clúster CTIC, usando 54 nodos.*

3.6 Comparación con una PC de procesador i7 de 8 núcleos físicos

Se realizó una comparación del clúster CTIC con una PC de escritorio actual con procesador Intel i7 de última generación y 8 núcleos adquirida el año 2014 en donde se ejecutó el mismo programa del cálculo de valor de PI. Se obtuvo como resultado un tiempo de 0,77 segundos como se muestra en la Figura 5.

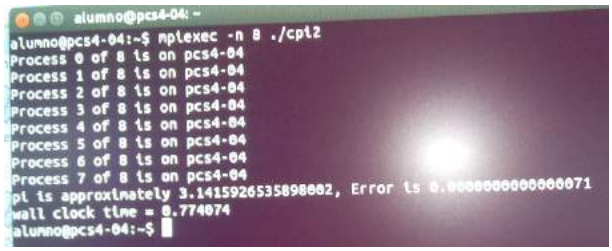

Figura 5. *Ejecución del mismo programa en una PC actual.*

Comparando ambos resultados, se concluye que clúster CTIC es más rápido que la PC de procesador i7.

3.7 Programa para el cálculo de pi

```
/*(C)2001 by Argonne National Laboratory.
*/
#include "mpi.h"
#include <stdio.h>
#include <math.h>
double f(double);
double f(double a)
{
return (4.0/(1.0 + a*a)) ;
}
int main(int argc, char *argv[])
{
int n, myid, numprocs, i ;
double PI25DT=3.14159265358979323846264 3;
double mypi, pi, h, sum, x;
double startwtime=0.0, endwtime;
int namelen;
char processor_name[MPI_MAX_PROCESSOR_NAME];
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
MPI_Comm_rank(MPI_COMM_WORLD,&myid);
MPI_Get_processor_name(processor_name,&namelen);

fprintf(stdout, "Process %d of %d is on %s \n",
myid, numprocs, processor_name);
fflush(stdout);
n=800000000; // # de rectángulos que se modificó
// por defecto viene con 10000
```

```
If (myid == 0)
startwtime = MPI_Wtime();
MPI_Bcast(&n, 1, MPI_INT,0,MPI_COMM_WORLD);
h= 1.0/(double)n;
sum = 0.0;
for(i=myid + 1; i <= n; i+=numprocs)
{
x= h*((double)i - 0.5);
sum += f(x);
}
mypi = h*sum;
MPI_Reduce(&mypi,&pi,1,MPI_DOUBLE,
MPI_SUM,0,MPI_COMM_WORLD);
if (myid=0){
endwtime= MPI_Wtime();
printf("pi is approximately %.16f,
Error is %.16f\n", pi,
fabs(pi - PI25DT) );
printf("wall clock time = %f\n",
endwtime-startwtime);
fflush(stdout);
}
MPI_Finalize();
return 0;
}
```

Figura 6. *Código en paralelo usado para el cálculo de pi.*

4 Pruebas de Rendimiento con simulaciones de N-cuerpos

A continuación se compruebaa el correcto funcionamiento del cluster CTIC al someterlo a un test de simulacion con software de última generación, utilizando un algoritmo de N-cuerpos, diseñado para la Astrofísica y fundamentado en los siguientes puntos:

- Número elevado de cuerpos N (e.g. estrellas), lo que hace que el tiempo de cálculo sea de orden O(N^2)
- No existe una solución analítica para este tipo de problema, es decir, no hay aproximación disponible para el cálculo directo de fuerzas.
- Requerimiento de memoria es proporcional a N^2

El algoritmo realiza el cálculo de interacciones gravitacionales entre N cuerpos (e.g. estrellas en una galaxia), como se ilustra en la figura 7

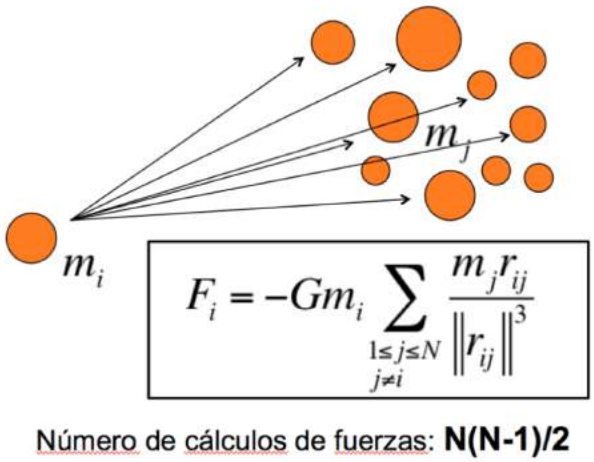

Figura 7. *Cálculo de fuerzas gravitacionales en un modelo de N cuerpos. F_i es la fuerza calculada para la partícula i, m_j es la masa de la partícula j, r_{ij} es la distancia entre partículas vecinas, y G es la constante gravitaroria. Fiestas (2011): IEEE 18th Real-time Conference, Berkeley, CA.*

El pseudo código secuencial de cálculo directo de N-cuerpos se esquematiza en la figura 8 y la función de cálculo de fuerzas se muestra en la figura 9, donde se aprecia el doble bucle en i y j responsable del cálculo de interacción gravitatoria de todas las partículas. Asimismo, se señala la cantidad de operaciones de coma flotante por segundo (FLOPs). El tiempo de ejecución del código en un solo CPU dependerá directamente del número total de partículas durante la simulación.

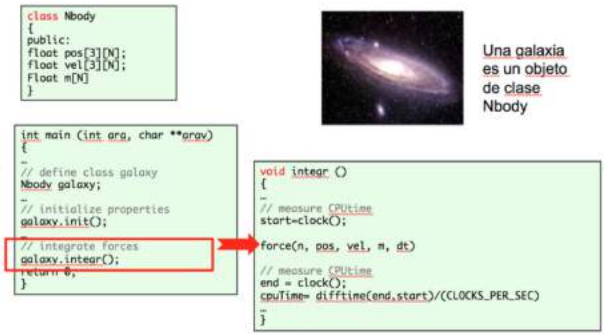

Figura 8. *Pseudo código secuencial del cálculo de fuerzas gravitacionales en un modelo de N cuerpos. Fiestas (2011): IEEE 18th Real-time Conference, Berkeley, CA.*

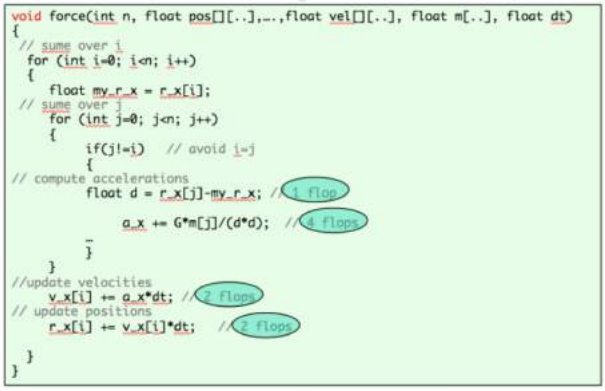

Figura 9. *Función de cálculo de fuerzas en código secuencial. Fiestas (2011): IEEE 18th Real-time Conference, Berkeley, CA.*

Este código es paralelizado utilizando técnicas MPI en el algoritmo antes mencionado. El pseudo código con MPI se muestra en la figura 10, en la cual se observa principalmente el uso de dos directivas MPI. **MPI_Bcast** se utiliza para distribuir la variable *nbody* (número total de partículas) entre todos los procesos, mientras que **MPI_Allreduce** se utiliza para sumar el resultado del cálculo de fuerzas de cada proceso y almacenarlo en la variable *force*.

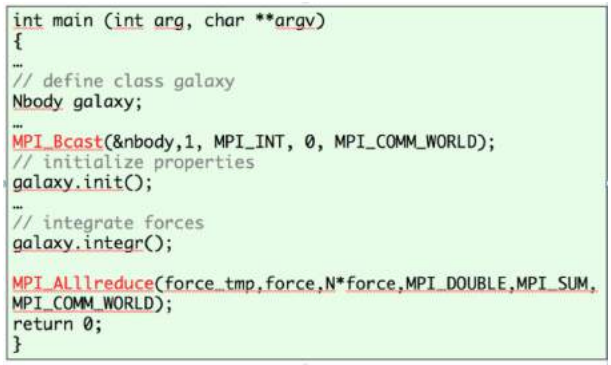

Figura 10. *Pseudo código de cálculo de fuerzas entre N cuerpos con directivas MPI. Fiestas (2011): IEEE 18th Real-time Conference, Berkeley, CA.*

La técnica del cálculo de fuerzas entre N cuerpos, se esquematiza en la figura 11, en la cual se nota claramente una distribución equitativa de la data entre los nodos, así como el flujo de información en el transcurso de las iteraciones temporales.

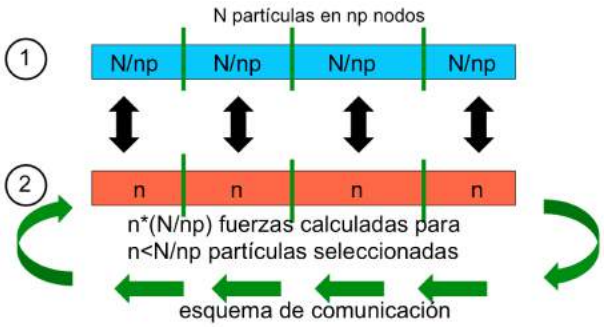

Figura 11. *Mecanismo de memoria distribuida (MPI). Fiestas (2011): IEEE 18th Real-time Conference, Berkeley, CA.*

El software usado para este estudio de rendimiento es llamado **phi-GPU**, un código híbrido escrito en C++, e implementado con directivas MPI para asegurar paralelismo efectivo a utilizarse en supercomputadores [6]. Este software ha sido probado en el Observatorio Nacional de China (Pekín) en un cluster con 85 nodos Dual Intel Xeon y 170 GPUs; en el Centro de Astronomia de Heidelberg (Alemania) en un cúster de 40 nodos, y en la Universidad de Berkeley, USA, en un clúster de 32 nodos [7], [8].

El paralelismo se logró dividiendo equitativamente las partículas entre nodos (directivas MPI_Bcast()) y calculando en cada nodo fuerzas fraccionadas de partículas llamadas *activas* en cada iteración temporal. El número de partículas activas (N_{act}) es normalmente pequeño en comparación al número total de partículas N, pero puede variar en principio entre 1 y N. Todas las fuerzas calculadas que actúan en las partículas *activas* se obtienen finalmente utilizando la directiva MPI_Allreduce() de comunicación.

Los compiladores utilizados son GNU e incluyen adicionalmente soporte de GPUs utilizando librerías de CUDA. A través de MPI se obtiene también soporte de paralelismo MultiGPU, de manera que cada CPU podría utilizar solo un GPU, pero es posible inicializar dos procesos MPI por nodo (para el caso de múltiples GPUs por procesador). La versión actual no soporta directivas OpenMP.

Se realizaron tests considerando un tiempo total de cómputo T_{tot} por intervalo de tiempo en unidades de código

$$T_{tot} = T_{force} + T_x + T_{MPI} \quad (1)$$

Donde T_{force} es el tiempo de cómputo del CPU, T_x es el tiempo de cómputo de funciones adicionales a la fuerza, y T_{MPI} el tiempo de comunicación para intercambio de datos entre los nodos. Los resultados muestran que dos factores dominan el tiempo de cómputo (aproximadamente el 90%). Estos son: el cálculo de fuerzas T_{force}, y el tiempo de comunicación entre nodos T_{MPI}. Este último contiene una componente que depende del ancho de banda, y otra componente que depende de la latencia. Por ello, el tiempo de cómputo se puede simplificar en

$$T_{tot} \sim T_{force} + T_{MPI} \quad (2)$$

Con el objetivo de calcular la velocidad de cómputo, o performance P, utilizamos la relación

$$P = (\#operaciones)/T_{TOT} \quad (3)$$

o

$$P \sim \gamma N \sum (N_{act})/T_{TOT} \quad (4)$$

Donde T_{TOT} es el tiempo de reloj necesitado en el cálculo de una unidad de tiempo de simulación (tiempo de N-cuerpos). γ representa el número de operaciones de coma flotante utilizando un algoritmo de Hermite por partícula e intervalo de tiempo de N-cuerpos. Basado en un conteo detallado del mismo, obtenemos $\gamma = 60$.

Los resultados de las pruebas de rendimiento del cluster-CTIC se presentan en las figuras 12 y 13. Para estas pruebas se utilizó un modelo inicial de Plummer, utilizado para representar una distribución esférica de estrellas en una galaxia. La figura 12 muestra los tiempos de cómputo versus el número de procesadores (nodos) utilizados. Vemos claramente que el tiempo disminuye, mientras el número de nodos aumenta. Esto debido al paralelismo de tareas a ejecutarse. Notese que el tiempo de cómputo disminuye cada vez mas lentamente, a medida que Np sigue aumentando. Esto indica que hay un incremento de tiempos de comunicación, que desaceleran la disminución del tiempo de cálculo. Esto se ve especialmente con cantidades menores de partículas (N=8K, 16K), donde el incremento de Np es ineficiente.

La figura 13 muestra el rendimiento en número de operaciones de coma flotante por segundo (FLOPs) versus el número de procesadores Np. La linea negra representa la dependencia teórica de FLOPs con Np. La regresión lineal nos da una aproximación de y =1.35*x, lo que se interpreta como un rendimiento de 1.35 GFLOPs (billones de FLOPs por segundo) por CPU. Como era esperado, las pruebas se acercan a esta relación mientras vamos incrementando el número de partículas, ya que el escalamiento se hace mas visible en estos casos. Utilizando 16 procesadores, obtenemos un máximo de 15.75 GFLOPs, lo que representa un 72% del rendimiento teórico esperado. A pesar que este rendimiento esta por debajo de los records mundiales del orden de PFLOPs (trillones de FLOPs por segundo), esto demuestra que es posible construir un cluster de computadores de rendimiento real muy cercano al teórico. Cabe anotar, que el rendimiento en los mejores computadores del mundo, nunca llegan a igualar al rendimiento teórico debido a condiciones técnicas y de implementación del sistema.

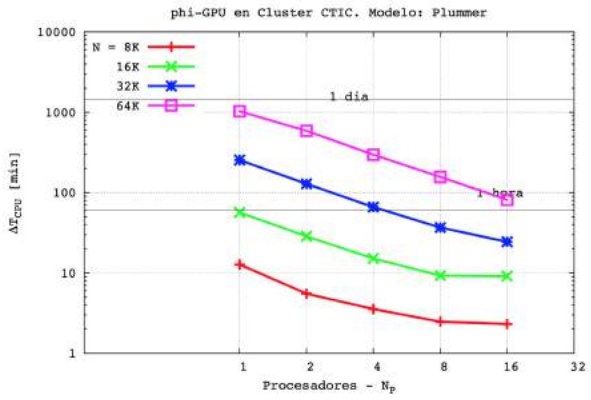

Figura 12. *Tiempos de cálculo en minutos por unidad de tiempo de N-cuerpos, para diferente número de cuerpos (N), y número de procesadores (Np).*

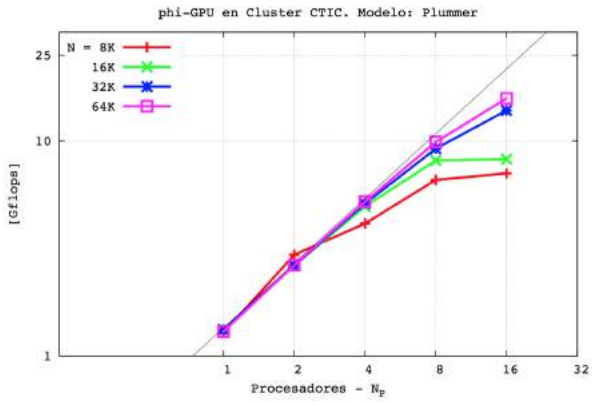

Figura 13. *Rendimiento en Gigaflops versus número de procesadores, para distintas cantidades de cuerpos (N). La relación lineal es teórica. Notese que cuanto más grande es el número total de partículas, el rendimiento se acerca más a la relación teórica.*

5 Conclusiones

El clúster CTIC es una herramienta útil que permitirá a sus usuarios ejecutar sus aplicaciones paralelas de forma eficiente, ofreciendo gran capacidad de cómputo, como lo demuestran los resultados obtenidos al ejecutar el mismo programa en el clúster y en una PC de última generación al momento de escribir este artículo. Es una plataforma computacional altamente escalable y económica tanto para su mantenimiento como para su actualización, en relación con otras plataformas computacionales fabricadas por grandes empresas como SGI, IBM, HP, etc., dado que sus componentes de hardware pueden ser adquiridos en cualquier distribuidor de PCs y el software principal para su funcionamiento es completamente gratuito. Se tiene proyectado incrementar el número de nodos de cómputo del clúster CTIC hasta un total de 30 PCs y añadir una red Gigabit Ethernet para la interconexión entre nodos. Adicionalmente, podemos concluir

- Nuestro software en paralelo permite explotar la eficiencia de procesos al asignar tareas computacionales intensas, que a través del paralelismo MPI, disminuyen en conjunto el tiempo total de cómputo significativamente.
- El hardware utilizado (cluster CTIC), se acerca en un 72% al peak teórico calculado.
- Así como el estudio de fenómenos del espacio es una tarea en la cual se pueden usar las técnicas de supercomputación, también lo son los estudios en medicina, arquitectura, ciencias sociales, marketing, y todo aquel problema en el que la eficiencia se mida por la capacidad de procesamiento de información en el menor tiempo posible.
- Mostramos lo factible que es la implementación de técnicas de HPC y Supercomputación a bajo costo, que han hecho posible este estudio, con un potencial inmenso en la educación en general y la industria. Nuestro país no debe ser ajeno a este desarrollo, sino impulsar el uso de estos instrumentos de aprendizaje a todo nivel.

6 Agradecimientos

Agradezco a Dios, a la UNI, a CTIC y a la Facultad de Ciencias.

1. Cruz Salazar, César Martín. “CRUZ: un Clúster aplicado a la educación, portátil, de bajo costo, usando Raspberry Pi”. Revista REVCIUNI volumen 16, Número 1, páginas 1 - 6. ISSN: 1813-3894, (2013).

2. Sterling, T.; Becker, D.; Dorband, J.; Savarese, D.; Ranawake, U. and Packer, C. BEOWULF: A Parallel Workstation for Scientific Computation. Proceedings of the 1995 International Conference on Parallel Processing (ICPP), páginas 11-14, Agosto, (1997).

3. Perozo Rondón, Freddy J. “Clúster: Implementación y Evaluación”, (2007).

4. Dennis, Andrew K. Raspberry Pi Super Cluster. (2013).

5. Rivera Díaz, Rosa Adriana. “Evaluación de un switch de baja latencia para aplicación en Clúster”, (2011).

6. Berczik P., et al. ApJ, 642, 21, (2006).

7. Spurzem, R.; Berczik, B.; Berentzen, I.; Wei, G.; Xiaowei, W.; Schive, H.; Hamada, T.; Nitadori, K., Fiestas, J., Accelerated Many-Core GPU Computing for

physics and astrophysics on three continents, published by Wiley Wiley, (2011)

8. Fiestas, J. et al., MNRAS, 419, 57, (2012)