Performance tests on the CPU and GPU Cluster using N-body simulation

Noemi Maritza Lapa Romero, Bach. en Ciencias de la Computación[1], Jose Antonio Fiestas Iquira, Dr. Astrofísico[1], Alonso Tenorio Trigoso, MSc. en Ingeniería Informatica[1], y Yuri Nuñez Medrano, MSc. en Ingeniería de Sistemas[1]
[1]Universidad Nacional de Ingeniería, Perú, nlapar@uni.pe, jose.fiestas@uni.edu.pe, atenoriot@uni.edu.pe, ynunezm@uni.edu.pe

Abstract– ***In recent decades, the field of computing has been one of the areas that has developed the most and that allows to obtain other fields in which it is needed, simulate, cure or summarize data in the field of research, business and entertainment***

With the passage of time, this amount of processing increases more and more, the current needs of users, for example, upload approximately 400 hours of videos, YouTube every minute, 300 million photos on Facebook daily for each event generated in CERN generates 1 Mb of data, approximately 600 million events per second, so it has been necessary to increase the computational power and as it became increasingly difficult to have this computing capacity in a single machine, we started to work implementing computers, computers in Grid, and create technologies such as Big Data for the storage and processing of data.

In this paper we present the implementation of a hybrid computer cluster that includes the use of CPUs and GPUs, details the construction and is used for GPUs, as well as an analysis of the results of the performance tests executed on the cluster.

Keywords- Cluster, Parallel Computing, CPU, GPU, n-BODY MPI, CUDA.

Digital Object Identifier (DOI):http://dx.doi.org/10.18687/LACCEI2018.1.1.494
ISBN: 978-0-9993443-1-6
ISSN: 2414-6390

Pruebas de Rendimiento Sobre el Clúster de CPUs y GPUs Empleando Simulación N-body

Noemi Maritza Lapa Romero, Bach. en Ciencias de la Computación[1], Jose Antonio Fiestas Iquira, Dr. Astrofísico[1], Alonso Tenorio Trigoso, MSc. en Ingeniería Informatica[1], y Yuri Nuñez Medrano, MSc. en Ingeniería de Sistemas[1]
[1]Universidad Nacional de Ingeniería, Perú, *nlapar@uni.pe, jose.fiestas@uni.edu.pe, atenoriot@uni.edu.pe*, *ynunezm@uni.edu.pe*

Resumen– En las últimas décadas el campo de la computación ha sido una de las áreas que más ha evolucionado y a su vez permitió que se desarrollaran otros campos en los cuales se necesitaba procesar, simular, curar o resumir data tanto en el campo de la investigación, empresa y entretenimiento

Con el paso del tiempo esta cantidad de procesamiento fue incrementándose constantemente debido a las necesidades actuales de los usuarios y la creciente cantidad de aplicaciones disponibles a éstos, por ejemplo, se suben aproximadamente 400 horas de videos a YouTube cada minuto [1], 300 millones de fotos a Facebook diariamente [2] y por cada evento producido en un experimento del CERN se genera 1 Mb de datos, ocurriéndose aproximadamente 600 millones de eventos por segundo [3] por lo que ha sido necesario incrementar el poder computacional y al hacerse cada vez más difícil tener todo esta capacidad de cómputo en una sola máquina se comenzó a trabajar implementado clústeres de computadores, computadoras en Grid, y crear tecnologías como Big Data para el almacenar y procesar los datos.

En el presente trabajo se presenta la implementación de un clúster híbrido de computadores la cual incluye el uso de CPUs y GPUs, se detalla la construcción y configuración, las pruebas de rendimiento usadas para la comparación de uso y no uso de GPUs, así como un análisis de los resultados de las pruebas de rendimiento ejecutadas sobre el clúster.

Keywords-- Cluster, Computación Paralela, GPU, n-BODY, MPI, y CUDA.

I. Introducción

Actualmente, los clústeres de sólo CPUs no resultan rentables debido al alto costo en comparación con tecnologías actuales, es decir, en vez de agregar un nodo nuevo a un clúster, ha sido más productivo y económico adicionar coprocesadores. Con la creación de las GPUs, que en un inicio tenían su uso principal en la renderización de imágenes y fines gráficos hasta llegar a la integración de éstas en los clústeres obteniéndose grandes beneficios en ahorro de costo y energía comparados con el rendimiento obtenido.

Se describirá qué es el paralelismo, qué son los clústeres de computadores, su importancia y qué papel desempeña en la actualidad, así mismo, lo que es un GPU y se menciona las técnicas y tecnologías usadas para el desarrollo de la investigación.

Finalmente, se muestra el algoritmo utilizado para la evaluación del clúster, se presentan los resultados obtenidos en las experiencias y el análisis de los resultados.

Digital Object Identifier (DOI): http://dx.doi.org/10.18687/LACCEI2018.1.1.494
ISBN: 978-0-9993443-1-6
ISSN: 2414-6390

II. Objetivos

El objetivo de este trabajo es dar a conocer las características y ventajas de la implementación de un clúster que incluya el uso de CPUs y GPUs que vienen siendo una alternativa viable para el crecimiento de los supercomputadores actuales.

Para el desarrollo de este trabajo se disponía de un clúster de 8 nodos que tienen como Sistema Operativo CentOS 6.8 a los cuales se les agregarán las tarjetas gráficas NVidia GeForce 710 y se procederá con las pruebas de rendimiento propias del equipo.

III. Computación Paralela

Se puede entender como diferentes núcleos de computadores o procesos trabajando para una tarea en conjunto en los cuales cada proceso se encarga de una porción específica del problema y los procesos pueden intercambiar información entre ellos [4].

Existen diferentes tipos de paralelismo, tales como el paralelismo a nivel bit, a nivel de instrucción, datos y tareas, a continuación describiremos estos dos últimos:

- Paralelismo de data, cuando cada proceso ejecuta la misma tarea (porción de código) sobre diferentes datos. Como ejemplo, podemos tener las operaciones matriciales.
- Paralelismo de tarea, cuando cada proceso ejecuta diferentes tareas, no necesariamente de una misma función o código, como ejemplo podemos tener el Pipeline de una computadora.

A. Ley de Amdahl

La Ley de Amdahl [4] explica que el tiempo de máxima paralelización de un código está siempre limitado por la parte del código secuencial, es decir, sea F_s la fracción secuencial del código y F_p la fracción paralelizable del código, luego $F_s - F_p = 1$, entonces si T_1 es el tiempo con p procesadores y T_1 el tiempo total del programa (fracción secuencial más fracción paralela), entonces:

$$T_p = T_1 .(F_{s+} F_s/P)$$

B. Arquitectura de Computadores Paralelos

Se mostrarán a continuación dos enfoques para la taxonomía de las computadoras paralelas [4]

- SISD, Single Instruction Single Data, arquitectura en la cual solo un procesador ejecuta un flujo de instrucciones sobre un grupo de datos almacenados en una única memoria.
- SIMD, Single Instruction Multiple Data, arquitectura en la cual un flujo de instrucciones es ejecutado en múltiples procesadores con diferentes datos (la data está en memorias independientes).
- MISD, Single Instruction Multiple Data, arquitectura en la cual distintos flujos de instrucciones son ejecutados en múltiples procesadores sobre un grupo de datos almacenados en una única memoria.

De acuerdo a la organización física de procesadores y memoria, se divide en las siguientes: [12]

- UMA, Uniform Memory Access, todas las memorias están a la misma distancia del procesador. Está limitado a un número pequeño de procesadores. Un ejemplo de estos son los CMP (Chip multiprocesadores, específicamente los Intel Core).
- NUMA, Non-Uniform Memory Access, una porción de la memoria está situada con cada procesador, lo que la hace de más rápido acceso. Un ejemplo de estos son los CMPs (Chip multiprocesadores, específicamente los Intel Nehalem, a partir del Core i7).

IV. GRAPHICS PROCESSING UNIT (GPU)

La Unidad de Procesamiento Gráfico, GPU, es un microprocesador especializado y altamente paralelo, diseñado en sus inicios para acelerar el renderizado 2D y 3D, lo cual era tarea de la CPU.

En los últimos 17 años las GPUs han tenido un gran incremento en su rendimiento y la cantidad de aplicaciones en las que pueden ser usadas, por ejemplo, en las GPUs actuales ya no se busca solo el procesamiento gráfico, sino que gracias a su ancho de banda de memoria (que supera al de las CPUs) y a las instrucciones aritméticas que presenta se ha generado una comunidad de investigadores que han aprovechado este potencial y lo están plasmando en lo que se conoce como General Purpose Graphic Processing Unit, GPGPU, o GPU computing, el cual ha posicionado a las GPU como una alternativa a los microprocesadores tradicionales en la computación de alto rendimiento.

En la presente sección se detalla las características que han llevado a la GPU a su apreciación actual y la evolución de la misma.

La GPU está diseñada para una clase particular de aplicaciones con las características siguientes [5]

- Grandes requerimientos computacionales, por ejemplo, la renderización de imágenes en tiempo real requiere gran rendimiento computacional.
- Paralelismo requerido, esto se puede expresar en que el programa tiene granularidad fina, por lo cual el paralelismo evidente y necesario por la cantidad de datos.
- El rendimiento es más importante que la latencia, dado que las operaciones en un procesador actual toman escalas de nanosegundos y el sistema visual humano funciona a escala de tiempo de milisegundos la latencia acumulada por las operaciones no son significantes.

GPU Computing

Para el desarrollo de esta sección se describirá primero como se programa en una GPU para trabajar sobre gráficos, luego la programación para GPGPU hace unos años y el enfoque actual [5].

Programación GPU para gráficos, comenzamos con el mismo pipeline gráfico ya definido, prestándole mayor atención a los pasos programables del mismo:

- El programador especifica la geometría que cubre la región en la pantalla. El rasterizador genera un fragmento en cada ubicación de píxel cubierta por esa geometría.
- Cada fragmento pasa por el Shader correspondiente.
- El programa de fragmentos captura su valor mediante una serie de operaciones matemáticas y lecturas de una "memoria global de texturas".
- La imagen resultante puede ser usada como textura para futuros pasos de pipeline gráfico.

Programación GPU para Propósitos Generales (GPGPU) - Antiguo, posee exactamente los mismos pasos mencionados en el ítem anterior, pero con diferentes terminologías. Para mejor interpretación se puede usar el ejemplo de una simulación de fluido en una región, en cada paso se debe calcular el siguiente estado del fluido en cada punto de la región, teniendo en cuenta la interacción con los fluidos de alrededor:

- El programador especifica la geometría de la primitiva que cubre el dominio computacional de interés. La rasterización genera un fragmento en cada lugar de pixel cubierta por esa geometría. (Para el ejemplo mencionado, la primitiva debe cubrir una región de fragmentos igual al tamaño del dominio de la simulación del fluido).

- Cada fragmento es procesado en el Shader por un programa SPMD de propósitos generales (Para el ejemplo mencionado, cada punto en la región ejecuta el mismo programa para calcular el estado del fluido).
- El programa de fragmentos calcula el valor del fragmento mediante operaciones matemáticas y acceso (obtención de datos) a la memoria global. (Para el ejemplo del fluido, en cada paso se puede obtener el estado de los vecinos en el paso previo para la obtención de su valor actual)
- El valor obtenido actualmente puede ser usado para futuras iteraciones. (Para el ejemplo, el estado actual del fluido será usado para la siguiente iteración).

Programación GPU para Propósitos Generales (GPGPU) - Actual, a pesar de que para el enfoque de las aplicaciones para GPGPU ya no era necesario el procesamiento de gráficos las aplicaciones aún usaban las APIs gráficas, así como aún se trabajaba en términos del pipeline gráfico con las partes programables solo en pasos específicos del pipeline. Actualmente los programas sobre GPU tienen la siguiente estructura:

- El programador define directamente el dominio de computacional de interés en una estructura grid de threads.
- Un programa SPMD se ejecuta para calcular el valor correspondiente en cada thread.
- El valor de cada thread es calculado mediante una combinación de operaciones matemáticas con acceso a lectura y escritura en la memoria global, a diferencia de los otros dos métodos, en este enfoque el mismo búfer puede ser usado para ambas aplicaciones permitiendo una mayor flexibilidad de los algoritmos a usarse.
- El resultado obtenido se almacena en el búfer y puede ser accedido en cálculos posteriores.

IV. Técnicas Utilizadas

Se explican las técnicas y tecnologías usadas para el desarrollo de la investigación.

A. *Clúster Chess*

El Cluster donde de desarrollo es llamado Chess, el cual es un conjunto de 8 computadores HP interconectados por una router y dos switches, construido en las instalaciones de Centro de Tecnologías de la Información de la Universidad Nacional de Ingeniería CTIC-UNI por investigadores del centro entre junio y julio del 2016, el nombre del clúster viene por la jerarquía de trabajo que pensaron para el mismo, el nodo King que controla los nodos Pawn, nodo Queen que se encargaría de verificar que todo funcione de la manera correcta, y tomaría el papel del nodo King en caso de fallos, y 6 nodos Pawn que se encargarían del proceso.
El propósito del clúster es instalar software para el desarrollo de Big Data. Físicamente, la organización del clúster es de forma estrella, es decir, todos los nodos están al mismo nivel conectados por un router.
Posteriormente se agregaron las tarjetas gráficas y se procedió a la instalación de todo el software para el desarrollo de aplicaciones en CUDA en el clúster.

TABLA I
Características del hardware

	Descripción
Modelo	HP EliteDesk 800 G1 SFF
Disco Duro	1TB
RAM	8 GB
Procesador	Intel® Core™ i7-4790 CPU
Cantidad de Nucleos Fisicos	4
Cantidad de Hilos	4
Total de Cores	8
Tarjeta gráfica	GeForce GT 710/ PCI-E 2.0/DDR3/192 cores
Tipo de Sistema Operativo	64-bit
Sistema Operativo	CentOS 6.8

B. *MPI*

De las siglas Message Passing Interface, es la primera librería de paso de mensajes, entre procesos que no comparten datos. Dada la característica de que no precisa de memoria compartida para su funcionamiento es muy usual que se use en sistemas distribuidos [6].

C. *CUDA*

CUDA es una arquitectura de cálculo paralelo de NVidia que aprovecha la gran potencia de la GPU (unidad de procesamiento gráfico) para proporcionar un incremento extraordinario del rendimiento del sistema.

La plataforma de cálculo paralelo CUDA proporciona unas cuantas extensiones de C y C++ que permiten implementar el paralelismo en el procesamiento de tareas y datos con diferentes niveles de granularidad. El programador puede expresar ese paralelismo mediante diferentes lenguajes de alto nivel como C, C++ y Fortran o mediante estándares abiertos como las directivas de OpenACC. En la actualidad, la plataforma CUDA se utiliza en miles de aplicaciones aceleradas en la GPU [7].

C. *Modelo N-body*

Las simulaciones gravitacionales de N-body (N-cuerpos), son soluciones numéricas a ecuaciones de los movimientos para N partículas que interaccionan gravitacionalmente.

Un algoritmo muy simple para este proceso es:

- Escoger un pequeño intervalo de tiempo.
- Calcular las fuerzas sobre cada partícula con los datos de todas las otras partículas.
- Actualizar la posición de cada partícula, como si la fuerza de esta permaneciera constante a lo largo de este intervalo de tiempo.

ALGORITMO N-body

Result: Posición para cada partícula
foreach particle i do
  foreach particle j do
    let $\bar{r}_{ij}$ be the vector between *i* and *j*; then the force on *i* because of *j*
    is $f_{ij} = -\bar{r}\frac{m_i m_j}{|r_{ij}|}$ (where m_i, m_j are the masses and $f_{ji} = -f_{ij}$)
  end
end

IV. Experimentación y Resultados

Los resultados obtenidos en las experiencias utilizando las técnicas explicadas previamente.

A. *Resultados de pruebas de rendimiento.*

Para estudiar la escalabilidad del clúster en este trabajo se realizaron las pruebas de rendimiento usando 1, 2, 4 y 8 nodos del equipo, cada nodo contiene hasta 4 cores de CPU y 192 cores de GPU. Los resultados son los siguientes.

A continuación se muestran las gráficas obtenidas con los datos del Cuadro de las TABLA II, TABLA III y TABLA IV para 192, 384,768 y 1536 núcleos de GPU.

TABLE II
TIEMPO EN SEGUNDOS DE LA COMUNICACIÓN ENTRE NODOS RESPECTO A LA CANTIDAD DE CUERPOS

cuerpos / nodos	8192	16384	32768	65536	131072	262144
1	0.011	0.022	0.053	0.160	0.502	1.278
2	3.627	6.450	11.098	23.448	46.599	105.140
4	6.437	11.521	20.397	41.756	89.878	196.550
8	10.174	17.956	31.741	63.217	128.310	283.540

TABLE III
TIEMPO EN SEGUNDOS DEL CÁLCULO DE FUERZA EN LAS GPU RESPECTO A LA CANTIDAD DE CUERPOS

cuerpos / nodos	8192	16384	32768	65536	131072	262144
1 (192 GPU)	8.817	37.045	157.460	695.370	3140.360	14192.000
2 (384 GPU)	5.017	19.481	82.800	352.920	1582.400	7116.800
4 (768 GPU)	3.278	10.941	43.250	185.212	802.218	3582.800
8 (1536 GPU)	2.305	7.055	24.036	96.671	419.9380	1814.400

TABLE IV
TIEMPO TOTAL EN SEGUNDOS DE LA EJECUCIÓN DEL PROGRAMA N-BODY (CPU+GPU) RESPECTO A LA CANTIDAD DE CUERPO

cuerpos / nodos	8192	16384	32768	65536	131072	262144
1 (192 GPU)	11.422	44.431	177.924	759.876	3366.160	14929.000
2 (384 GPU)	10.241	30.194	105.338	411.356	1730.180	7569.800
4 (768 GPU)	10.806	25.174	70.575	247.276	948.538	3939.900
8 (1536 GPU)	13.312	26.958	60.470	172.924	583.154	2191.800

B. *Gráficas de resultados obtenidos*

La Fig, 1 siguiente, se obtiene de comparar el máximo rendimiento que podría brindar el clúster estudiado, es decir, como cada nodo es una máquina i7 con 4 núcleos físicos, tenemos datos para 4 (1 nodo), 8 (2 nodos), 16 (4 nodos), 32 núcleos de CPU (8 nodos), y se comparan con el máximo rendimiento que se obtiene del clúster cuando se usan todas las tarjetas gráficas disponibles, existe una tarjeta gráfica por nodo con 192 núcleos CUDA cada una. Tener presente que por cada nodo CPU solo se usa 1 tarjeta gráfica.

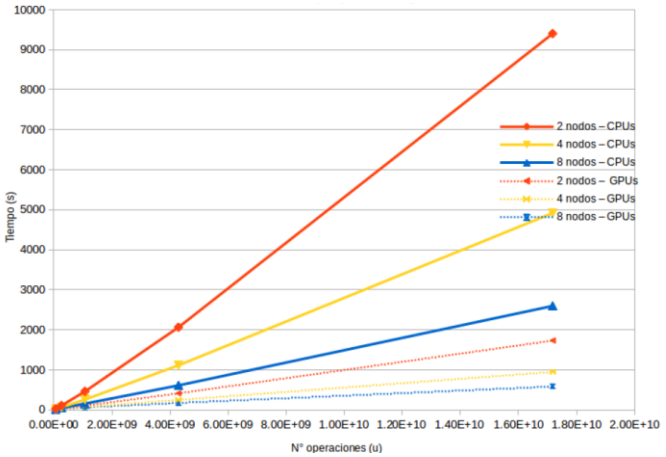

Fig. 1 Simulación N cuerpos para 1, 2, 4, y 8 nodos.

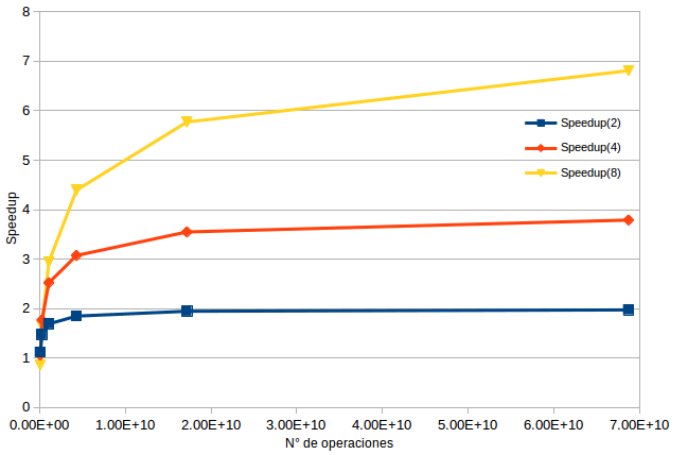

Fig. 2 Speedup vs número de operaciones respecto al incremento de GPUs.

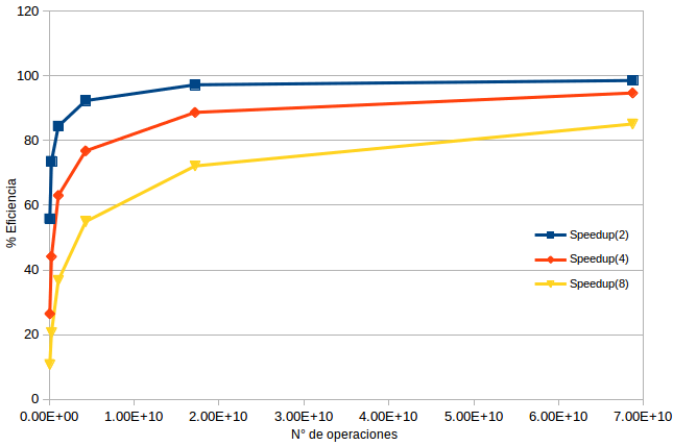

Fig. 3 Porcentaje de eficiencia vs número de operaciones.

VI Análisis

De la TABLA II, TABLA III y TABLA IV se tiene las siguientes observaciones:

- Se observa que cuando la cantidad de cuerpos en el sistema es menor a 65000 cuerpos, el tiempo de cómputo se ve afectado de manera significativa por el proceso de la copia de datos entre el nodo principal y los trabajadores, para su mejor entendimiento, limitado por el tipo de red sobre el que se trabaje, y la conexión que este tenga, que para nuestro caso es una conexión estrella.
- Se observa también que a medida que la cantidad de cuerpos se incrementa, el tiempo total de cómputo, sumados CPU y GPU tienden a tener una curva que se acerca mucho a la curva de solo los cálculos realizados en las tarjetas gráficas, por lo cual se puede inferir que el código está altamente paralelizado, además que se ha definido correctamente las operaciones de grano fino en las tarjetas gráficas, que contrastan con las realizadas por los núcleos de CPU.
- Además, se observa que para los puntos tomados para todos los casos de cantidad de cuerpos que se probaron, siempre hubo una mejora por el incremento de los nodos, aunque por la tendencia no se puede especificar por cuantos incrementos de nodo se seguiría apreciando este incremento.

De las Fig. 1, 2 y 3 se tienen las siguientes observaciones:

- Respecto a la Fig. 1 se puede apreciar que hay escalamiento a lo largo de todas las curvas, independientemente y en conjunto, así pues, se puede notar a simple vista que la cantidad de tiempo para cantidad de cuerpos (para este caso los más visibles son para 65536, 131072 y 262144 cuerpos, el tiempo de cómputo de éstos se reduce a casi la mitad cuando se duplican la cantidad de nodos).
- De la Fig. 1 se puede observar la escalabilidad para cada caso del tiempo que toma para cada cantidad de cuerpos, la cual se presume que sigue una ecuación cuadrática debido a que en verdad la cantidad de tareas a ejecutarse en las tarjetas gráficas vienen dadas por (número de cuerpos)^2, debido a la interacción uno a uno con los demás cuerpos.
- De la Fig. 2, se observa que para el caso de 2 nodos respecto a 1, el valor del speedup se incrementa hasta aproximadamente un valor de 2 para 262144 cuerpos. Para el caso de 4 nodos se observa que este valor crece aproximadamente 4 veces y para el de 8 nodos, se observa que el valor del speedup es aproximadamente 7 veces cuando la cantidad de cuerpos es 262144, esto quiere decir que hasta 8 GPUs en el sistema, el clúster mantiene una escalabilidad ideal esperada, es decir que respecto a 1 nodo si se incremente a 2 nodos, es cómo si se tuviesen 2 nodos ideales, si se tienen 4 nodos se tiene aproximadamente 4 nodos ideales y se tienen 8 se comporta como si fuesen 7 nodos ideales trabajando en conjunto.
- De la Fig. 3 obtenida de la gráfica 2, se observa que la eficiencia se incrementa a medida que se aumenta la cantidad de operaciones, (*número de cuerpos*)^2, se observa que para 262144 cuerpos el porcentaje de la eficiencia se aproxima a 100% para 2, 4 y 8 nodos, así también que la curva de eficiencia reduce su crecimiento cuando se incrementan la cantidad de nodos, por ejemplo, se observa que la curva de 8 nodos no ha alcanzado una eficiencia tan cercana al 100% en contraste con la de 2 y 4 nodos. Esta

reducción se puede inferir que es por la cantidad de cuerpos, pues se observa que la curva de 8 nodos tiene una pendiente más alta de crecimiento que la de 2 y 4 nodos, por lo cual podría inferirse que, si se agregan más cuerpos a la simulación, este obtendría un valor más cercano a 100%.

VII Conclusiones

Se probó que el algoritmo N-body, usado es óptimo para realizar las pruebas de rendimiento a futuros equipos híbridos. Así también, que con el uso de las GPUs se ha obtenido un incremento de 5 veces la rapidez de cálculo respecto al rendimiento máximo del clúster con sus nodos de CPU.

Se logró determinar que el escalamiento es el esperado hasta los 8 nodos con una tarjeta gráfica cada una, es decir, al duplicar la cantidad de nodos, disminuye aproximadamente a la mitad el tiempo de ejecución de las operaciones sobre la misma cantidad de datos y al duplicar la cantidad de operaciones por nodo, se duplica también la cantidad de tiempo que éste demora en ejecutar. Finalmente, para el escalamiento de las GPUs, se observa que para 2 y 4 nodos con 262144 nodos se obtiene una eficiencia cercana al 100% mientras que para 8 nodos ésta se ve reducida a aproximadamente 85%.

Agradecimientos

El presente trabajo fue desarrollado gracias a la Universidad Nacional de Ingeniería y a los fondos FONDECYT del programa Ciencia Activa (CONCYTEC).

Referencias

[1] T. Gwava, "How much data is created on the internet each day?" https://www.gwava.com/blog/internet-data-created-daily, 2016. [Online; Accessed: 2016-11].

[2] Zephoria, "The top 20 valuable Facebook statistics – updated November 2016." https://zephoria.com/top-15-valuable-facebook-statistics/, 2016. [Online; Accessed: 2016-11].

[3] CERN, "Processing: What to record?" https://home.cern/about/computing/processing-what-record. [Online; Accessed: 2016-09].

[4] V. Eijkhout, E. Chow and R. Van de Geijn, "Parallel computing" in *Introduction to High Performance Scientific Computing*, 2014.

[5] J. Owens, M. Houston, D. Luebke, S. Green, J. Stone and J. Phillips "GPU Computing" in *Proceedings of the IEEE*, 2014.

[6] J. Mantas, "Programación distribuida y paralela" http://lsi.ugr.es/jmantas/pdp/ayuda/mpi_ayuda.php, 2012. [Online; Accessed:2016-11].

[7] Nvidia, "Procesamiento paralelo CUDA" www.nvidia.es/object/cuda-parallel-computing-es.html, 2012. [Online; Accessed:2016-11].

[8] Nvidia Developers, "Nvidia CUDA installation guide for Linux" http://docs.nvidia.com/cuda/cuda-installation-guide-linux/#axzz4LpOly9U7. [Online; Accessed: 2016-07-10].

[9] T. 500, "TOP 10 Sites for November 2016." https://www.top500.org/lists/2016/11/, 2016. [Online; Accessed: 2016-11-20].

[10] H. Obermaier, "The Computer Graphics Pipeline." http://idav.ucdavis.edu/~obermaier/Lecture_Chapter2.pdf, 2014. [Online; Accessed: 2016-11-10].

[11] R. Kilgariff and E. Fernando, "Chapter 30. The GeForce 6 Series GPU Architecture." http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter30.html, 2005. [Online; Accessed: 2016-11-23].

[12] V. Nagarajan, "Types of parallelism." http://www.inf.ed.ac.uk/teaching/courses/pa/Notes/lecture02-types.pdf, 2016. [Online; Accessed: 2016-11].

[13] A. Pozas and M. Curiel, "Renderizado Gráfico Pipeline." http://cidecame.uaeh.edu.mx/lcc/mapa/PROYECTO/libro39/211_renderizado_grafico_pipeline.html, 2016.[Online; Accessed:2016-11-10].

[14] A. Rege, "An Introduction to Modern GPU Architecture." ftp://download.nvidia.com/developer/cuda/seminar/TDCI_Arch.pdf, 2008. [Online; Accessed: 2016-11-23].

[15] S. Che, M. Boyer, J. Meng, D. Tarjan, J. Sheaffer, S. Lee and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing,"in2009 IEEE International Symposium on Workload Characterization (IISWC), pp. 44–54, Oct 2009.

[16] Spurzem, R.; Fiestas, J. ,Berczik, B.; Berentzen, I.; Wei, G.; Xiaowei, W.; Schive, H.; Hamada, T.; Nitadori, K., 2011, Accelerated Many-Core GPU Computing for physics and astrophysics on three continents, published by Wiley