Utilización de Cluster de GPU en HPC. Un caso de estudio

Erica Montes de Oca, Laura De Giusti, Franco Chichizola, Armando De Giusti, Marcelo Naiouf

Instituto de Investigación en Informática LIDI (III-LIDI)
Facultad de Informática – Universidad Nacional de La Plata
La Plata, Buenos Aires, Argentina
{emontesdeoca,ldgiusti,francoch,degiusti,mnaiouf}@lidi.info.unlp.edu.ar

Resumen. En este trabajo se realiza un análisis del uso de las arquitecturas GPU y un cluster de GPU, teniendo como caso de estudio el problema de atracción gravitacional de los cuerpos celestes. Se presentan las implementaciones para ambas arquitecturas, empleando para la primera CUDA y para la segunda MPI-CUDA. Se detallan y analizan los resultados experimentales, mostrando una buena aceleración obtenida utilizando un cluster de GPU. Además, se compara la aceleración de la aplicación ejecutada en un cluster de CPU y en el cluster de GPU; se observa claramente que el uso de este último logra una aceleración similar al uso del cluster de CPU pero con tiempos de ejecución significativamente menores.

Palabras claves: procesamiento paralelo, GPGPU, GPU, Cluster de GPU, Cluster de CPU, CUDA, MPI, N-Body.

1 Introducción

Si bien las GPU (*graphic processing unit*) fueron concebidas para dedicarlas al procesamiento gráfico, actualmente son muchas las aplicaciones que se han adaptado a esta arquitectura, alcanzando las mismas una aceleración significativa en el ámbito de HPC. La velocidad de procesamiento que se puede obtener con estas nuevas plataformas paralelas es indiscutible [1] debido a su diseño. Las GPUs, a diferencia de las CPUs, tienen una mayor cantidad de transistores dedicados al procesamiento de datos, en cambio las CPUs dedican muchos de sus transistores para flujo de control y grandes cachés [2] [3]. Si bien hasta el presente, el conjunto de problemas de propósito general resolubles en placas gráficas es menor que aquellos que pueden resolverse computacionalmente en las CPUs, estas aplicaciones se han visto altamente beneficiadas disminuyendo el tiempo de procesamiento utilizando sólo una GPU. Por otra parte, a medida que las GPUs se instalaron como una nueva arquitectura paralela han surgido los cluster de GPU [4].

La complejidad de las arquitecturas heterogéneas actuales basadas sólo en CPUs, sumado a la aceptación de CUDA como modelo de programación paralela de propósito general [5], han hecho que la GPU sea una arquitectura paralela opcional para la resolución de problemas de alta demanda computacional.

Al hacer mención a cluster de GPU, también se hace referencia a una arquitectura heterogénea en la cual, por un lado se tiene un subsistema conformado por varios cores CPUs y su sistema de memoria y entrada/salida, mientras que por otro lado, se encuentra el subsistema GPU con sus memorias on y off chip. Estos dos sistemas se comunican por medio de un bus PCI-E con una velocidad de ancho de banda baja comparada con las velocidades de comunicación de los sistemas de memoria. Por lo cual, el cuello de botella es la comunicación que existe entre dichos subsistemas [6] [7] [8].

La aceleración que puede obtenerse con el uso de un cluster de GPU es dependiente de la aplicación. Por ejemplo en [9], se estudia el algoritmo de cifrado AES, implementado para una GPU y para un cluster de GPU. En ese caso, el uso del cluster no permite obtener una disminución de los tiempos totales de la aplicación, porque el tiempo de procesamiento no logra solapar al tiempo de comunicación, por lo que es más conveniente el uso de una sola GPU, realizando con la misma la ejecución de varias instancias del problema.

Para el uso de cluster de GPU se debe buscar disminuir la cantidad de datos a comunicar desde la CPU o host a la GPU o device, en aplicaciones en las cuales la cantidad de información a procesar es muy grande. Un problema de alta demanda computacional muy estudiado es el N-Body, que se caracteriza por ser adaptable a varios campos científicos [10].

En este trabajo se presenta la resolución del problema de N-Body utilizando una GPU y un cluster de GPUs. La Sección 2 plantea el problema a resolver. En la Sección 3 se describen las soluciones implementadas. La Sección 4 contiene los resultados experimentales. La Sección 5 presenta las conclusiones y los trabajos futuros.

2 Problema de alta demanda computacional: N-Body

El problema N-Body ha sido muy estudiado por su aplicación en varios campos científicos [10]. En astrofísica, dicho problema es utilizado para calcular la fuerza de atracción gravitacional de los cuerpos en el espacio [11]. El presente artículo se basa en la resolución de este problema centrado en la Teoría de Newton sobre su Ley de Atracción [12].

Para la resolución del problema se requiere de la siguiente información básica: la masa, la velocidad, la posición y la fuerza de atracción inicial de cada cuerpo. Siendo la Ecuación (0) el cálculo central de procesamiento:

$$F = \frac{G \times m_i \times m_j}{r^2} \tag{0}$$

con r = distancia, G = constante gravitacional

La fuerza de atracción gravitacional nos permite calcular posteriormente la nueva posición y la nueva velocidad de cada uno de los cuerpos en el espacio de simulación [13][14].

El algoritmo utilizado se denomina *all pair* o simulación directa. En dicho algoritmo, todos los cuerpos necesitan conocer la fuerza de atracción gravitacional del resto de los cuerpos que conforman el espacio para poder calcular su movimiento. Por esto, la complejidad de la solución es de $O(N^2)$ [15].

Este algoritmo puede ser adaptado a la GPU por tratarse de una solución paralela de datos, siendo el cálculo del movimiento de cada cuerpo independiente en cada paso de simulación [16].

En trabajos anteriores, se ha mostrado que la resolución del presente caso de estudio usando una GPU se ve beneficiada de manera significativa, al compararla con las versiones paralelas desarrolladas en memoria distribuida, memoria compartida y en la combinación de ambas para la arquitectura CPU [14].

3 Descripción de las soluciones

Basada en la solución presentada en trabajos previos [14][17], se realiza la adaptación de la misma para ser ejecutada en múltiples GPUs. La ejecución en un cluster de GPUs, requiere la utilización de una combinación de MPI con CUDA.

Por cada proceso MPI creado se tiene asociada una GPU (en este trabajo la cantidad máxima de GPUs es ocho). Utilizando un esquema master/slave, el proceso master es el encargado de inicializar la información de los cuerpos a distribuir entre los procesos esclavos. Los N cuerpos se distribuyen en P procesos, por lo que cada proceso debe computar un total de N/P elementos. A su vez, cada proceso se encarga de enviar por medio de PCI-E dichos datos a sus respectivas GPUs, siendo éstas las que realmente realizan cómputo, ya que los procesos MPI se limitan a comunicar entre ellos los datos resultantes de cada paso de simulación, para que cada GPU conozca la información calculada por el resto de los procesos.

En la GPU, los datos son almacenados en la *memoria global* (*off-chip*) y luego son cargados a la *memoria shared* (*on-chip*) en porciones de 256 datos. La copia de los datos a esta última memoria, así como la reserva de espacio en la misma, es tarea del programador.

Cabe destacar, que al inicio del algoritmo se le debe enviar los N datos a la GPU. Una vez que la GPU ha calculado la fuerza de atracción de los cuerpos que le correspondían, enviará dichos datos a la CPU. Paso siguiente, el proceso MPI, comunicará dichos datos al resto de los procesos que conforman el problema; recibiendo al mismo tiempo los datos calculados por los demás. Dicho procesamiento se repetirá t pasos, con t definido por el usuario.

4 Resultados experimentales

En esta sección se presentan los resultados obtenidos a partir de la ejecución del problema planteado en un cluster de multicore con procesadores Quad Core Intel i5-

2300 de 2,8 GHz, con caché de 6MB y Sistema Operativo Debian de 64 bits; cada nodo del cluster cuenta con una GPU Geforce TX 560TI, con 384 procesadores con una cantidad máxima de 768 de threads y 1 GB de memoria RAM.

La cantidad de cuerpos para cada simulación varía entre: 256000, 512000, 1024000, 2048000, 4096000 y 8192000 cuerpos para cinco pasos de simulación. Los datos obtenidos son el resultado de un promedio de 15 ejecuciones.

El cluster de GPU está conformado por un máximo de 8 máquinas con las GPUs descriptas anteriormente.

En la Tabla 1 se muestran los tiempos de ejecución en segundos para cinco pasos de simulación del problema en una GPU, para los diferentes tamaños de problema . En la Tabla 2, se muestra el tiempo de ejecución obtenido utilizando el cluster de GPU para cinco pasos de simulación, variando en este caso el tamaño del problema y la cantidad de nodos (GPUs) usadas en el cluster.

Tabla 1. Tiempos de ejecución (en segundos) para la solución en una GPU.

Cantidad de cuerpos	Solución en una GPU
256000	39,38
512000	157,45
1024000	629,37
2048000	2516,77
4096000	10065,70
8192000	40259,87

Tabla 2. Tiempos de ejecución (en segundos) para la solución en cluster de GPU variando la cantidad de nodos (desde 2 hasta 8 GPUs).

Cantidad de cuerpos	2	3	4	5	6	7	8
256000	19,66	13,31	11,41	9,45	8,12	7,20	5,66
512000	76,68	51,88	39,33	31,34	26,95	22,97	21,59
1024000	304,76	204,10	153,63	122,98	104,04	89,14	79,63
2048000	1215,70	813,09	610,15	488,29	409,22	350,20	308,37
4096000	4855,50	3240,80	2432,65	1946,29	1628,41	1394,36	1221,62
8192000	11654,83	11052,64	7773,94	7700,80	6485,66	5561,17	4867,82

Tanto para el caso del uso de una GPU como el cluster de GPU, el bloque de threads utilizado fue de 256.

En la Fig.1 se presenta la aceleración obtenida por el uso del cluster de GPU. La aceleración alcanzada por el uso de más de una GPU se calculó como el cociente del tiempo de ejecución del algoritmo con una sola GPU sobre el tiempo de ejecución del algoritmo en el cluster de GPU.

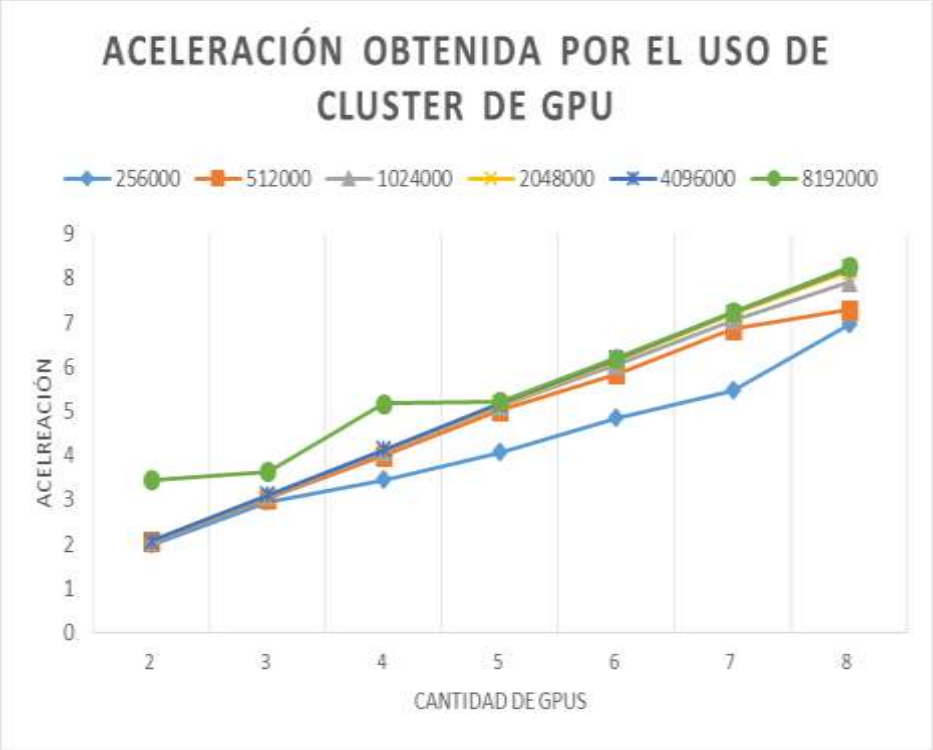

Fig. 1. Aceleración obtenida por el uso del cluster de GPU para 2 a 8 GPUs

Como se puede apreciar en la Fig. 1, la solución ejecutada en un cluster de GPU obtiene una significativa aceleración y, en ninguno de los casos, la aceleración disminuye sino que se mantiene prácticamente lineal para las pruebas realizadas hasta el uso de 8 GPUs.

Se realizaron pruebas para analizar la incidencia de MPI en la solución. Los tiempos de comunicación de los procesos son mínimos y totalmente compensados por el tiempo de procesamiento de la GPU. Además, al aumentar la cantidad de GPUs la fracción de datos comunicados por cada CPU a su GPU (y viceversa), es cada vez más pequeña, por lo que el tiempo de comunicación disminuye.

En trabajos previos [14] se estudió la aceleración obtenida sobre la solución secuencial usando un Cluster de CPU. En la Tabla 3, se presentan los tiempos de ejecución en segundos de la aplicación en el cluster de CPU, para los tamaños de problema de 256000 y 512000 cuerpos en 5 pasos de simulación, con 2, 4 y 8 máquinas. En la Figura 2 y 3, se muestra una comparación de las curvas de aceleración utilizando un Cluster de CPU (usando MPI) y Cluster de GPU, con 2, 4 y 8 máquinas para un tamaño de problema de 256000 y 512000 cuerpos respectivamente en cinco pasos de simulación.

Tabla 3. Tiempos de ejecución (en segundos) para la solución en un cluster de CPU.

Cantidad de cuerpos	2	4	8
256000	9711,28	4856,14	2429,29
512000	38848,18	19430,01	9716,82

Fig. 2. Comparación de la aceleración obtenida en el uso de un cluster de CPU y un cluster de GPU para 2, 4 y 8 máquinas con un tamaño de problema de 256000 cuerpos.

Fig. 3. Comparación de la aceleración obtenida en el uso de un cluster de CPU y un cluster de GPU para 2, 4 y 8 máquinas con un tamaño de problema de 512000 cuerpos.

Mientras que para el caso del uso del cluster de CPU, los tiempos de ejecución se encuentran dentro del rango de media hora, en el cluster de GPU los resultados son obtenidos en unos pocos segundos, evidenciando una significativa ganancia de tiempo al utilizar este tipo de arquitectura. Por otro lado, como puede observarse en ambos gráficos, las curvas de aceleración obtenidas para los dos tipos de cluster (de CPUs y de GPUs) tienen un comportamiento similar al incrementar la cantidad de nodos .

Cabe mencionar que para realizar una comparación significativa respecto a la escalabilidad en ambos tipos de cluster al incrementar el tamaño de la arquitectura (cantidad de nodos), se deben utilizar mayor cantidad de máquinas. Sin embargo, en la actualidad, sólo se cuenta con 8 GPUs para realizar la experimentación.

5 Conclusiones y trabajos futuros

Las GPUs han comenzado a establecerse como nuevas arquitecturas paralelas. En un principio, su utilización para problemas de propósito general estaba discutida. Sin embargo, el incremento en el desarrollo de aplicaciones sobre GPU y el nacimiento de CUDA para facilitar la adaptación de las mismas a las GPUs ha hecho que se beneficien de la gran aceleración que puede alcanzarse con ellas.

Reducir los tiempos de procesamiento con las arquitecturas CPUs del mercado de hoy nos insume un costo mayor comparado con las arquitecturas GPUs [10], sin mencionar la heterogeneidad de las primeras que requieren de una programación cuidadosa en varios aspectos para lograr la ganancia esperada de su uso.

Si bien el gran cuello de botella que se encuentra al trabajar con GPUs está dado por la comunicación de ésta con la CPU, se ha podido sortear este inconveniente

dividiendo el tamaño del problema en porciones más pequeñas, lo que permite reducir los datos enviados a la GPU.

Los resultados presentados muestran claramente los beneficios del uso del cluster de GPU para el problema N-body. Teniendo una aceleración significativa y sostenida al aumentar la cantidad de GPUs (hasta el máximo de ocho).

Como trabajos futuros se plantea el análisis de otros problemas adaptables a la GPU, para ser ejecutados en un cluster de GPUs, así como analizar la escalabilidad de dicha arquitectura heterogénea, con respecto al número de placas.

Referencias

1. Pritish Jetley, Lukasz Wesolowski, Filippo Gioachin, Laxmikant V. Kalé, Thomas R. Quinn. : Scaling Hierarchical N-body Simulations on GPU clusters. Departamento de Ciencias de la Computación, Universidad de Illinois y Departamento de Astronomia, Universidad de Washington. IEEE 978-1-4244-7558-2 (2010)
2. Chao-Tung Yang, Chih-Lin Huang, Cheng-Fang Lin, Tzu-Chieh Chang.: Hybrid Parallel Programming on GPU Clusters, Departamento de Ciencias de la Computación, Universidad de Tunghai, Taiwan, IEEE 978-0-7695-4190-7/10 (2010)
3. Perez C., Piccoli M. F.: Estimación de los parámetros de rendimiento de una GPU. Mecánica Computacional. Vol. XXIX, pp. 3155-3167 (2010)
4. Volodymyr V. Kindratenko, Jeremy J. Enos, Guochun Shi, Michael T. Showerman, Galen W. Arnold, John E. Stone, James C. Phillips, Wen-mei Hwu. : GPU Clusters for High-Performance Computing. Universidad de Illinois. Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on. 978-1-4244-5011-4. (2009)
5. Chao-Tung Yang, Chih-Lin Huang, Cheng-Fang Lin. : Hybrid Cuda, OpenMP, and MPI parallel programming on multicore GPU clusters, Departamento de Ciencias de la Computación, Universidad de Tunghai, Taiwan, 0010-4655 (2010)
6. Nvidia Corporation: CUDA C Best Practices Guide. (2012)
7. Nvidia Corporation: NVIDIA CUDA C Programming Guide (2011)
8. Lingyuan Wang, Miaoging, Vikram K. Narayana, Tarek El-Ghazawi. : Scaling Scientific Applications on Clusters of Hybrid Multicore/GPU Nodes, The George Washington University, ACM 978-1-4503-0698-0. (2011)
9. Adrian Pousa, Victoria Sanz, Armando De Giusti. Análisis de rendimiento de un algoritmo de criptografía simétrica sobre GPU y Cluster de GPU. Instituto de Investigación en Informática LIDI, Fac. de Informática, UNLP. HPC La TAM 2013. (2013)
10. Peter Berczik, Keigo Nitadori, Shiyan Zhong, Rainer Spurzem, Tsuyoshi Hamada, Xiaowei Wang, Ingo Berentzen, Alexander Veles, Wei Ge. : High performance massively parallel direct N-body simulations on large GPU cluters. Astronomisches Rechen-Institut, ZAH, Heidelberg, Alemania, Main Astronimical Observatory, NASU, Kyiv, Ucrania, Centre for Astronomy, Jeidelberg University, Heidelberg, Alemania. HPC –UA (2011)
11. Tsuyoshi Hamada, Keigo Nitadori. : 190 TFlops Astrophysical N-body Simulation on cluster of GPUs. Universidad de Nagasaki. IEEE 978-1-4244-7558-2 (2010)
12. Bruzzone Sebastian, LFN10, LFN10-OMP y el Método de Leapfrog en el Problema de los N Cuerpos, Instituto de Física, Departamento de Astronomia, Universidad de la República y Observatorio Astronómico los Molinos, Uruguy (2011)

13. Andrews Gregory R.: Foundations of Multithreaded, Parallel, and Distributed Programming. Addison-Wesley (2000)
14. Montes de Oca Erica, De Giusti Laura, De Giusti Armando, Naiuof Marcelo. : Comparación del uso de GPU y cluster de multicores en problemas con alta demanda computacional. Instituto de Invertigación en Informática LIDI, Fac. de Informática, UNLP. CACIC 2012, XVIII Congreso Argentino de Ciencias de la Computación, pág. 267-275. (2012)
15. Jeroen Bédorf. : High Performance Direct Gravitational N-body Simulations on Graphics Processing Units, Universiteit van Amsterdam (2007)
16. Lasr Nyland, Mark Harris, Jan Prins. : Fast N-body simulation with CUDA. GPU Gem 3, cap. 31, pp. 677-695. Addison-Wesley (2007)
17. Montes de Oca Erica, De Giusti Laura, Rodriguez Ismael, De Giusti Armando, Naiuof Marcelo. : Análisis de la escalabilidad y el consumo energético en soluciones paralelas sobre cluster de multicores y GPU para un problema con alta demanda computacional. Instituto de Investigación en Informática LIDI, Fac. de Informática, UNLP. CACIC 2013, XIX Congreso Argentino de Ciencias de la Computación (2013)