CEIG - Spanish Computer Graphics Conference (2018)
I. García-Fernández and C. Ureña (Editors)

A Prototype of a Scalable Multi-GPU Molecular Dynamics Simulator for Large Molecular Systems

G. Nicolas-Barreales [1], M. Novalbos[1], M. A. Otaduy[1,2] and A. Sanchez[1,2]

[1]Universidad Rey Juan Carlos, Spain
[2]Center for Computational Simulation, Spain

Abstract

Parallel architectures, in the form of multi-core or multiple computers, have produced a major impact in the field of information technology. GPU devices, as an extreme example of parallel architectures, have been adapted to enable generic computation in massively parallel architectures. Molecular dynamics is a problem that fits perfectly such architectures, as it relies on the computation of many similar interactions between atoms. Moreover, large molecular systems require resources that exceed those available in a single computer, even multi-GPU computers. Therefore, the ideal architecture to simulate molecular dynamics is a distributed multi-GPU cluster, which consists of multiple interconnected computers with one or more GPUs each. A molecular dynamics simulation usually needs days, and even weeks of computation time to produce results that represent only a few microseconds of atom interactions. In contrast, distributed multi-GPU clusters allows us to develop an efficient and scalable simulator. This paper aims to develop a prototype of a molecular dynamics simulator for large molecular systems. It uses the GPU as the main computing device, using only the CPU to control the workflow. We have implemented parallel processing techniques to develop a fully scalable system.

CCS Concepts

•***Computing methodologies*** → ***Massively parallel algorithms;*** *Graphics processors;* •***Applied computing*** → *Molecular structural biology;*

1. Introducción

Los sistemas moleculares están compuestos por un gran número de partículas que interactúan entre si a nivel atómico. La industria farmacéutica requiere poder anticipar su comportamiento en base a las interacciones que se producen para analizar sus propiedades. Sin embargo, la síntesis de un sistema molecular requiere de elevados costes y tiempo de realización. La simulación de dinámica molecular trata de recrear dichas interacciones apoyándose en el uso de equipamiento informático. Estas simulaciones permiten anticipar el resultado de una disolución química antes de sintetizarla, lo que supone un ahorro de tiempo y material de laboratorio.

La dinámica molecular [Sch02] tiene en cuenta las características de los átomos para calcular las fuerzas y energías que generan y así poder representar fielmente el movimiento que realizan durante un determinado espacio de tiempo. De esta manera se pueden reproducir los movimientos de los átomos de una forma que no es posible utilizando un método analítico. Aún así, supone un elevado coste computacional y un gran uso de memoria [NGO*13, NGO*14], lo que añadido al cada vez mayor número de átomos analizados hace necesario el desarrollo de soluciones escalables que trabajen de forma paralela utilizando varios equipos. Esto permitiría aumentar la capacidad computacional y reducir drasticamente los tiempos de ejecución. Actualmente la mayoría de los centros de supercomputación existentes disponen de equipamiento para la ejecución de cálculos masivamente paralelos mediante GPUs.

Este trabajo presenta un prototipo escalable de simulador para dinámica molecular para arquitecturas multi-GPU. La mayor parte de los cálculos realizados en GPU se corresponderán con los cálculos de fuerzas, distribuyéndolos entre las distintas GPUs que formen la infraestructura de ejecución. Como diferencia a otros simuladores, las GPUs se utilizaran como dispositivos de cálculo principal, de forma que la CPU se encarga solo de controlar la ejecución.

2. Trabajo relacionado

En las simulaciones de dinámica molecular los átomos se definen en un espacio tridimensional que representa un entorno real dentro de un volumen especifico. Los cálculos abarcan desde la mecánica cuántica hasta la mecánica clásica y mecánica estadística, incluyendo muchos procedimientos ya establecidos. Para calcular las fuerzas se divide la simulación en pequeños pasos de tiempo. La definición de la duración del paso de tiempo es directamente pro-

DOI: 10.2312/ceig.20181149

porcional a la precisión y a la duración de la simulación. Para cada uno de estos pasos se calcula la posición de cada átomo en función de todas las fuerzas que le afectan (resto de átomos) integrándolo con su velocidad para obtener sus posiciones finales.

En el estado del arte, se han desarrollado diversos simuladores de dinámica molecular como CHARMM [BBM*09], AMBER [SFGP*13], NAMD [PBW*05], GROMACS [AMS*15], ACEMD [HGF09], etc. Entre ellos destacan NAMD y GROMACS que realizan el calculo de fuerzas tanto en CPU y GPU [SPF*07]. Estos simuladores utilizan computación heterogénea introduciendo gran parte de la carga computacional de la simulación en las GPU como coprocesador. Igualmente se puede destacar ACEMD ya que realiza los cálculos de las diferentes fuerzas en GPU, donde cada tipo de fuerza se calcula en una GPU distinta. Esto permite que se aproveche directamente la arquitectura GPU, pero la escalabilidad esta muy limitada al usar sólo una GPU por fuerza (si el número de átomos a analizar es muy elevado posiblemente superará la capacidad de memoria de una única GPU) y a que las comunicaciones se realizan a través de la CPU. En general ningún simulador se ha diseñado para ejecutarse en un entorno masivamente paralelo multi-GPU lo que limita el rendimiento.

Nuestro objetivo es desarrollar un simulador de dinámica molecular utilizando la GPUs como dispositivos de cálculo principal. Se plantea una solución totalmente escalable, tanto en tiempo de ejecución como en memoria. En este caso la CPU se encargará solo de controlar la ejecución, siendo las propias GPUS las que se envíen la información directamente.

3. Dinámica molecular

La dinámica molecular consiste en el cálculo de la posición de cada átomo que compone el sistema molecular en cada paso de tiempo. Para lograr este propósito es necesario calcular las fuerzas que actúan sobre cada uno de esos átomos. Estas fuerzas se pueden dividir en tres clases:

1. Fuerzas de enlace, que son de diferentes tipos en función del numero de átomos afectados por el enlace. Estos enlaces pueden ser simples, ángulos y dihédricos propios e impropios.
2. Fuerzas no enlazadas de corto alcance, que están compuestas por las fuerzas de Van der Waals y las fuerzas electrostáticas generadas por los átomos que se encuentran dentro de un radio de corte R_c. Para la resolución de estas fuerzas se suele subdividir el sistema en celdas mediante el algoritmo CellList [Sch02].
3. Fuerzas no enlazadas de largo alcance, que consisten en las interacciones electrostáticas entre todos los átomos más alejados del radio de corte R_c

4. Dinámica molecular en paralelo en Multi-GPU

Paralelizar un algoritmo de dinámica molecular para ejecutarlo en un entorno multi-GPU no es sencillo y supone ciertos problemas que deben ser abordados para implementar un sistema escalable.

Este trabajo propone el desarrollo de un algoritmo paralelo donde los datos de la simulación se distribuyen directamente entre los distintos nodos, en nuestro caso GPUs. Cada una de ellas determina que datos deben ser compartidos con las otras teniendo en cuenta

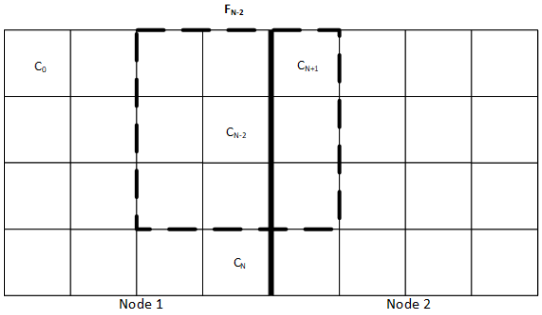
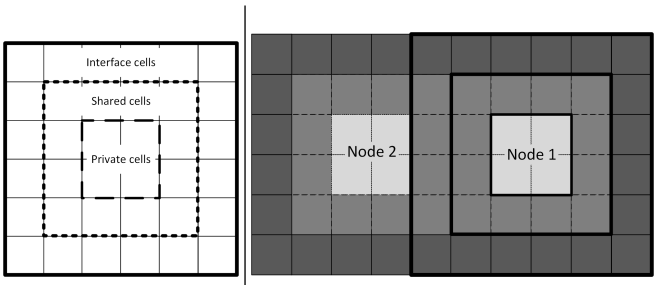


Figura 1: *Cálculo de fuerzas no enlazadas de corto alcance mediante celdas. Es necesario tener los datos de las celdas compartidas de los nodos vecinos para poder realizar los cálculos, ya que en la zona de frontera de cada nodo la fuerza aplicada sobre los átomos depende de otros átomos que se encuentran en otro nodo.*

que los átomos se pueden mover a lo largo de la simulación, de forma que se reduzca el uso de memoria todo lo posible. Para realizar el particionado, tenemos en cuenta el funcionamiento de las *fuerzas no enlazadas de corto alcance* en base a celdas como se especifica en el algoritmo *CellList* [Sch02]. Para que el cálculo de las mismas sea correcto, se deben duplicar los datos de las celdas que se encuentran en las fronteras de las particiones (ver Figura 1). Para ello las celdas se clasifican en tres tipos, lo cual se puede ver gráficamente en la Figura 2a):

1. Celdas privadas: celdas que abarcan los átomos que no están compartidos con otros nodos.
2. Celdas compartidas: todos los átomos que se encuentren en estas celdas se comparten con otros nodos. Cada paso de tiempo se envían las posiciones nuevas de cada átomo que esté en estas celdas para que el resto de particiones pueda hacer los cálculos correctos. Cada cierto numero de pasos de tiempo se actualizan los átomos contenidos en estas celdas.
3. Celdas de interfaz: son las celdas que contienen los átomos compartidos por los nodos de alrededor. En cada paso de tiempo se reciben las nuevas posiciones de estos átomos. Cada cierto numero de pasos de tiempo se reciben los átomos nuevos para estas celdas. Los átomos de estas celdas solo se utilizan para cálculos de fuerzas de los átomos de celdas propias del nodo, ya que los cálculos de los de interfaz se harán en su nodo de origen.



(a) *Celdas en los nodos* **(b)** *Relación entre las celdas de los nodos*

Figura 2: *Tipos de celdas para particionado.*

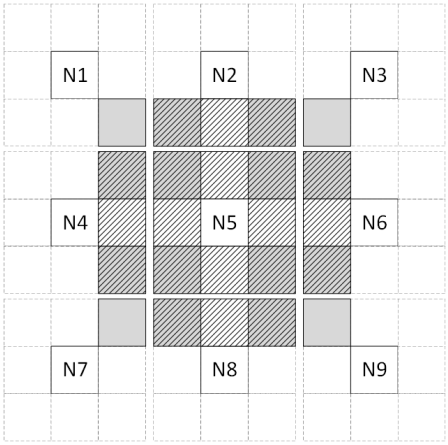


Figura 3: *Distribución de información entre vecinos. Cada nodo comparte ciertas celdas con los nodos vecinos: 8 vecinos en 2 dimensiones (26 si es en 3D). En el caso mostrado, el nodo central N5 tiene varias zonas de datos compartidos. Por un lado, tomando como ejemplo la esquina colindante con el nodo N1, comparte los datos de esa zona con los nodos N1, N2 y N4 (color gris). Además comparte otros datos con los nodos N2 y N4 (rayado). Podemos ver que esto se realiza de forma similar en el resto de sus zonas compartidas.*

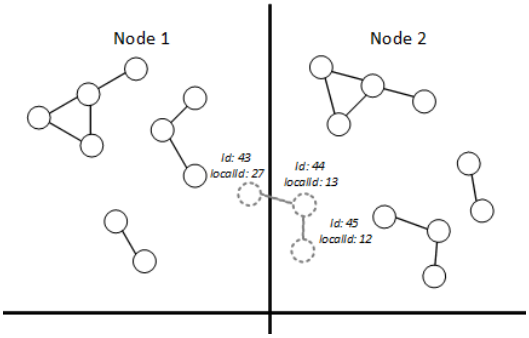


Figura 4: *Enlaces distribuidos en dos nodos. Cuando un átomo (ej. id. 43) pasa de la zona de celdas privadas a la zona compartida hay que detectarlo y enviarlo. Además se debe enviar el enlace correspondiente a dicho átomo, con todos sus datos asociados (enlace formado por ids. 43, 44 y 45) para poder realizar los cálculos necesarios en los nodos con los que está compartido. Para poder identificar los átomos dentro del sistema utilizaremos dos tipos de identificadores: globales, que identifican el átomo para todo el sistema; y locales, que identifican el átomo dentro de cada nodo. En el momento de enviar un átomo se debe convertir el identificador local a global, lo que implica cierta complejidad computacional y aumenta el uso de memoria (véase sección 4.1).*

Las rejilla de celdas del sistema es fija, y se asigna cada átomo a cada celdas según su posición global *x, y, z* en un momento dado. Los átomos de la molécula se distribuyen entre los diferentes nodos replicando la información de los átomos de aquellas celdas que son necesarias, tal y cómo se observa en la Figura 3.

Por otro lado, las fuerzas de enlace indican que hay una relación a tener en cuenta entre dos, tres o cuatro átomos. Así, cada átomo está asignado al menos a un enlace. Como los átomos están conectados entre si a través de enlaces, si uno de los átomos pasa de una celda a otra que pertenece a una partición de otra GPU se deben enviar los enlaces asociados para poder realizar los cálculos de fuerzas de cada átomo, cómo se ve en la Figura 4. Este envío se realizará cada cierto numero de pasos de tiempo.

Por último, las fuerzas no enlazadas de largo alcance también se ven afectadas por el particionado. Hay distintos algoritmos como PME [DYP93] y MSM [HWP*15] que resuelven este problema. Hemos elegido MSM para su resolución ya que ofrece mejor escalabilidad que PME al escalar un sistema de tamaño modesto, menos de 100K átomos, a más de mil procesadores (esto demuestra la idoneidad de MSM para la simulación paralela a gran escala [HWP*15, HSS09]). Para aplicaciones prácticas la precisión de MSM es igual que PME para cálculos de propiedades de densidad del agua, constante de difusión, constante dieléctrica, tensión superficial, función de distribución radial y factor de Kirkwood dependiente de la distancia.

4.1. Tabla hash

Puesto que el particionado del sistema requiere de algunos datos duplicados entre los diferentes nodos es necesario poder identificar cada átomo de manera global, para saber que átomos pasar con cada uno de los enlaces. Mantener una relación de todos los átomos en todos los nodos tiene un alto coste de memoria (podría ser hasta superior del espacio de memoria disponible en cada GPU si el tamaño molecular es suficientemente elevado), por lo que se ha optado por identificar los átomos, tanto de forma local al propio nodo como de forma global. Este método presenta el inconveniente de tener que convertir entre identificadores locales y globales. Su conversión se realiza utilizando una tabla hash habiendo optado por usar la implementación CUDPP de tabla hash de Nvidia en GPU [cud], que permite reducir drasticamente el tiempo de ejecución y el espacio ocupado en memoria.

La Figura 5 muestra que el tiempo utilizando la tabla hash es prácticamente despreciable en comparación con el tiempo total de ejecución. Esto ayuda a mejorar significativamente la escalabilidad con respecto a este punto.

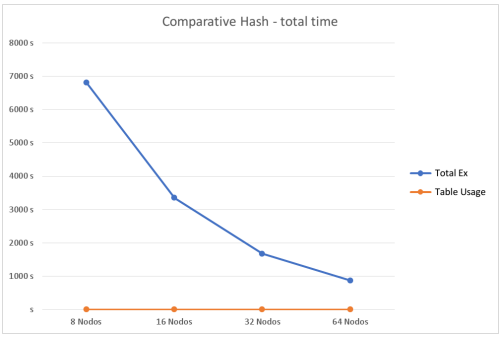


Figura 5: *Uso de la tabla hash vs. tiempo de ejecución.*

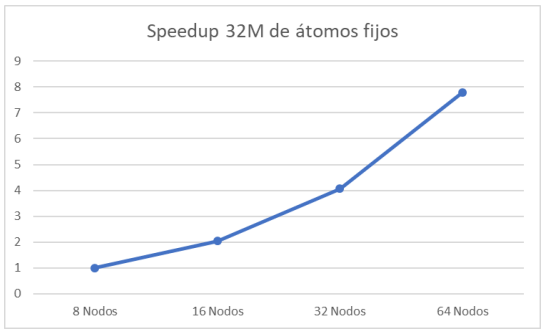
5. Evaluación de escalabilidad

Para realizar pruebas de escalabilidad hemos ejecutado nuestro prototipo en el cluster Minotauro del BSC (Barcelona Supercomputing Center). Hemos realizado dos tipos de baterías de pruebas consistentes en la simulación de un sistema integramente formado por moléculas de agua durante 100 pasos de tiempo, calculando en todos ellos las fuerzas de enlace, la velocidad y la posición de los átomos, y cada 10 pasos las fuerzas no enlazadas de corto alcance y la actualización de las zonas compartidas de los nodos. Se ha dejado como línea futura la ejecución de las fuerzas no enlazadas de largo alcance. La primera prueba consiste en mantener fijo el número de átomos (32 millones) y ejecutar simulaciones variando el numero de nodos para medir la escalabilidad del sistema a medida que reducimos el numero de átomos que se simulan en cada GPU (Figura 6a). La segunda consiste en mantener fijo el numero de nodos (32 GPUs) a medida que vamos añadiendo más átomos a la simulación (Figura 6b). De los resultados obtenidos de estas pruebas se extrae que hemos logrado disminuir todo lo posible los tiempos de envío de datos de la simulación, haciendo que prácticamente todo el tiempo de ejecución sea para el cálculo de fuerzas.

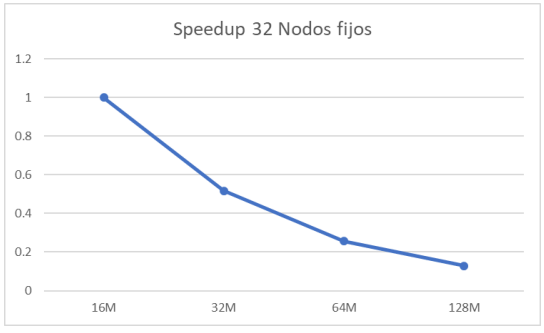
6. Conclusiones

En este artículo se ha presentado un prototipo escalable de simulador para dinámica molecular para arquitecturas multi-GPU, el cual dispone de una comunicación eficiente entre los distintos nodos y hace un uso de memoria reducido. Como resultado tenemos un sistema escalable que utiliza las GPUs como elemento principal de procesamiento.

Nuestro sistema de particionado permite añadir tantas GPUs como sean necesarias de forma distribuida, pudiendo disponer de una o varias en cada uno de los nodos de cómputo. Como limitación de la arquitectura el particionado no garantiza un balanceo de carga entre las distintas GPUS, ya que este se realiza en función de la posición de los átomos, y no de su complejidad. Además todavía no se ha realizado una evaluación teniendo en cuenta la funcionalidad de las fuerzas no enlazadas de largo alcance lo que se plantea cómo trabajo futuro.



(a) *Numero de átomos fijos (32M).*



(b) *número de nodos fijo (32 nodos).*

Figura 6: *Speedup. La escalabilidad conseguida es prácticamente lineal a medida que introducimos más nodos. Igualmente el tiempo de ejecución aumenta el doble a medida que metemos más átomos con el mismo número de nodos.*

References

[AMS*15] ABRAHAM M. J., MURTOLA T., SCHULZ R., PÁLL S., SMITH J. C., HESS B., LINDAHL E.: GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX 1-2* (2015), 19 – 25. 2

[BBM*09] BROOKS B. R., BROOKS C. L., MACKERELL A. D., NILSSON L., ET AL.: CHARMM: The biomolecular simulation program. *Journal of Computational Chemistry 30*, 10 (2009), 1545–1614. 2

[cud] Cudpp GPU hash table. [Online] http://cudpp.github.io/cudpp/2.3/hash_overview.html. Accessed: 2018-04. 3

[DYP93] DARDEN T., YORK D., PEDERSEN L.: Particle mesh ewald: An Nlog(N) method for Ewald sums in large systems. *The Journal of Chemical Physics 98*, 12 (1993), 10089–10092. 3

[HGF09] HARVEY M. J., GIUPPONI G., FABRITIIS G. D.: ACEMD: Accelerating biomolecular dynamics in the microsecond time scale. *J. of Chemical Theory and Computation 5*, 6 (2009), 1632–1639. 2

[HSS09] HARDY D. J., STONE J. E., SCHULTEN K.: Multilevel summation of electrostatic potentials using graphics processing units. *Parallel Computing 35*, 3 (2009), 164 – 177. Revolutionary Technologies for Acceleration of Emerging Petascale Applications. 3

[HWP*15] HARDY D. J., WU Z., PHILLIPS J. C., STONE J. E., SKEEL R. D., SCHULTEN K.: Multilevel summation method for electrostatic force evaluation. *Journal of Chemical Theory and Computation 11*, 2 (2015), 766–779. 3

[NGO*13] NOVALBOS M., GONZALEZ J., OTADUY M. A., LOPEZ-MEDRANO A., SANCHEZ A.: On-board multi-gpu molecular dynamics. In *Proceedings of the Euro-Par 2013 Parallel Processing* (2013), Springer, pp. 862–873. 1

[NGO*14] NOVALBOS M., GONZALEZ J., OTADUY M. A., MARTINEZ-BENITO R., SÁNCHEZ A.: Scalable on-board multi-gpu simulation of long-range molecular dynamics. In *Proceedings of the Euro-Par 2014 Parallel Processing* (2014), Springer, pp. 752–763. 1

[PBW*05] PHILLIPS J. C., BRAUN R., WANG W., GUMBART J., TAJKHORSHID E., VILLA E., CHIPOT C., SKEEL R. D., KALÉ L., SCHULTEN K.: Scalable molecular dynamics with NAMD. *Journal of Computational Chemistry 26*, 16 (2005), 1781–1802. 2

[Sch02] SCHLICK T.: *Molecular Modeling and Simulation: An Interdisciplinary Guide*. Springer-Verlag., Secaucus, NJ, USA, 2002. 1, 2

[SFGP*13] SALOMON-FERRER R., GÖTZ A. W., POOLE D., LE GRAND S., WALKER R. C.: Routine microsecond molecular dynamics simulations with AMBER on GPUs. explicit solvent particle mesh ewald. *J. Chem. Theory Comput. 9*, 9 (2013), 3878–3888. 2

[SPF*07] STONE J. E., PHILLIPS J. C., FREDDOLINO P. L., HARDY D. J., TRABUCO L. G., SCHULTEN K.: Accelerating molecular modeling applications with graphics processors. *Journal of Computational Chemistry 28*, 16 (2007), 2618–2640. 2