Optimización del Tiempo de Ejecución en Problemas de Dinámica Molecular

Angela Di Serio
Universidad Simón Bolívar, Departamento de Computación y TI,
Caracas, Venezuela, 1080-A
adiserio@ldc.usb.ve

and

María B. Ibáñez
Universidad Simón Bolívar, Departamento de Computación y TI,
Caracas, Venezuela, 1080-A
ibanez@ldc.usb.ve

Abstract

Molecular Dynamics (MD) is a powerful tool used to study the properties of molecular systems and their interactions. MD simulations are computational intensive that requires run on parallel architectures in order to produce results in reasonable time. Because of their dynamic nature, the processors workload change along the simulation. In order to minimize the execution time, the processor workloads need to be reassigned. Recent research has showed that the Generalized Dimension Exchange algorithm improves the total execution time of MD simulation distributing molecules uniformly between processors. Nevertheless, processors can consume different amount of time to execute the balanced workload. In this work, we present a different approach to distribute the workload between the processors based on the execution time. The experiments performed show that the total execution time of the simulation is reduced.
Keywords: Dynamic Load Balancing, Distributed Load Balancing, Parallel Applications, Short Range Molecular Dynamics.

Resumen

Dinámica Molecular (DM) es una herramienta de gran utilidad para el estudio de las propiedades de los sistemas moleculares y de sus interacciones. Es una aplicación de cómputo intensivo que requiere ser ejecutada en arquitecturas paralelas para producir resultados en tiempos razonables. Debido a su naturaleza dinámica, la carga de trabajo de los procesadores cambia a lo largo de la simulación. Por lo tanto, para lograr minimizar el tiempo de ejecución es necesario redistribuir la carga entre los procesadores. Investigaciones recientes han mostrado que el uso del algoritmo Generalized Dimension Exchange mejora el tiempo total de ejecución de DM distribuyendo la carga uniformemente entre los procesadores. Sin embargo, los procesadores pueden consumir cantidades de tiempo diferentes para ejecutar la carga balanceada. En este trabajo presentamos una alternativa diferente para distribuir la carga entre los procesadores basado en el tiempo de ejecución. Los experimentos realizados muestran que la mejora logró reducir el tiempo de ejecución de la simulación de DM.
Palabras claves: Balance de Carga Dinámico, Balance de Carga Distribuido, Aplicaciones Paralelas, Dinámica Molecular de corto alcance.

1. INTRODUCCIÓN

La simulación de Dinámica Molecular (DM) es una técnica de gran utilidad para el estudio de las propiedades de los sistemas moleculares y de sus interacciones. Se utiliza para simular las propiedades de líquidos, sólidos [5],[6]. Es una aplicación de cómputo intensivo que requiere ser ejecutada en arquitecturas paralelas para producir resultados en tiempos razonables.

La simulación consiste en calcular la interacción entre las moléculas de un sistema, afortunadamente las interacciones son independientes y pueden ser calculadas en paralelo, pero generan serios problemas de balance de carga. Inicialmente todos los procesadores tienen la misma cantidad de moléculas. A medida que avanza la simulación, las moléculas cambian de posición y por lo tanto, la carga de trabajo de los procesadores se modifica. La paralelización del algoritmo se vuelve ineficiente después de cierto tiempo. Para poder minimizar el tiempo de ejecución, es necesario mantener balanceada la carga de trabajo de todos los procesadores que participan en la simulación.

En los últimos años se han desarrollado diversos algoritmos de balance de carga para distintas clases de aplicaciones científicas [9],[10],[11]. Las técnicas de balance de carga pueden ser clasificadas en estáticas y dinámicas. Un algoritmo de balance de carga estático asigna la carga a los procesadores durante la compilación. Esta técnica no puede ser aplicada a los problemas de DM por su naturaleza dinámica. Los métodos de balance de carga dinámicos distribuyen la carga durante la ejecución basado en el comportamiento de la aplicación. Uno de los métodos más populares de redistribución de carga para problemas de DM es el método recursivo de bisección geométrico [4],[7]. Sin embargo, este método requiere sincronización global y transferencia intensiva de datos entre los procesadores que resulta en una pobre escalabilidad y altos costos computacionales [9]. Existen técnicas centralizadas y distribuidas para la toma de decisiones, y la migración de la carga puede considerar el dominio global de procesadores o un dominio reducido como por ejemplo el conjunto de los vecinos más cercanos.

Trabajos anteriores [2], [3], nos permitieron concluir que el método de balance de carga dinámico distribuido conocido como Generalized Dimension Exchange (GDE) logra mejoras en el tiempo total de ejecución de la simulación de DM distribuyendo uniformemente el número de moléculas entre los procesadores. Sin embargo, se ha observado que procesadores con igual número de moléculas consumen diferentes cantidades de tiempo en realizar la simulación y en ocasiones la diferencia en tiempo es significativa. En este trabajo se presentan mejoras al balanceador basado en GDE que corrigen este comportamiento de los procesadores en nuestra aplicación.

El trabajo se encuentra organizado de la siguiente manera. En la sección 2, se describe en qué consiste la simulación de Dinámica Molecular. En la sección 3, se presentan los componentes de un balanceador de carga genérico y las decisiones específicas que se toman para resolver el problema planteado. Los experimentos realizados se muestran en la sección 4 y finalmente, la sección 5 contiene las conclusiones.

2. SIMULACIÓN DE DINÁMICA MOLECULAR

Dinámica Molecular es una herramienta computacional usada para simular las propiedades de líquidos y sólidos. Cada una de las *N* moléculas en la simulación es tratada como una masa puntual y las ecuaciones de Newton son integradas para calcular su movimiento. Diversas técnicas han sido desarrolladas para paralelizar simulaciones de DM de corto alcance. La que mejor se adapta a la aplicación es la descomposición espacial de Plimpton [8] en donde cada procesador es responsable de simular una porción del dominio del sistema. Para ello, el espacio de simulación es dividido en pequeñas cajas de tres dimensiones y cada una de estas cajas es asignada a un procesador.

Durante la evolución de la simulación, las moléculas se mueven y pueden abandonar el espacio del procesador y entrar en el espacio de otro. Las moléculas son reasignadas a los nuevos procesadores a medida que se desplazan por el espacio de simulación. Para poder calcular las fuerzas sobre las moléculas, es necesario que lo procesadores conozcan las posiciones de las moléculas que se encuentran en los procesadores vecinos. El tamaño de las cajas asignadas a cada procesador dependerá del número de moléculas y del número de procesadores que se usen en la simulación.

En el algoritmo de descomposición espacial , cada procesador mantiene una lista con las moléculas asignadas y otra con las moléculas que se encuentran en los procesadores vecinos y que están dentro de un cierto alcance. Una iteración del algortimo incluye las siguientes actividades:

1. Construcción de la lista de moléculas que salieron del espacio del procesador y envío a sus vecinos. A su vez, el procesador recibe la lista de moléculas que salieron de otros procesadores y entraron a este espacio. Esto se lleva a cabo cada 20 iteraciones
2. Construcción de la lista de moléculas vecinas
3. Cálculo de las fuerzas de las moléculas usando la lista de vecinos
4. Actualización de las posiciones de las moléculas

5. Comunicación de las nuevas posiciones de las moléculas

El problema que surge con este tipo de paralelización es que la simulación puede volverse ineficiente después de un cierto tiempo, debido al desplazamiento de las moléculas. La ineficiencia puede ser corregida redistribuyendo la carga entre los procesadores, i.e. aumentando o disminuyendo el espacio asignado a los procesadores.

3. BALANCE DE CARGA DINÁMICO

El objetivo de los algoritmos de balance de carga es mantener la carga de trabajo de los procesadores aproximadamente igual. Para lograr este objetivo es importante identificar cuatro componentes que usualmente conforman un balance de carga dinámico:

- Regla de Medida de Carga. Los algoritmos de balance de carga dinámico se basan en información relacionada con la carga de los procesadores. Típicamente se caracteriza mediante un índice de carga que suele ser un entero no negativo cuyo valor es cero si el procesador está libre, y que toma valores mayores que cero a medida que el procesador tiene una carga mayor.

- Regla de Intercambio de Información. Especifica el instante de tiempo en el cual se llevará a cabo la recolección y el mantenimiento de la información de la carga de trabajo de los distintos procesadores.

- Regla de Inicio de Balance. Establece cuándo se debe iniciar un proceso de balance de carga. La regla de inicio debería tomar en cuenta el costo de la operación de balance versus el beneficio en cuanto a rendimiento que se obtiene una vez realizado el balance.

- Operación de balance. Puede ser definida mediante tres reglas: regla de localización, regla de distribución y regla de selección. La regla de localización determina el grupo de procesadores o dominio de balance que intervendrán en la operación de balance con respecto a un procesador dado. La regla de distribución determina cómo se redistribuye la carga entre los procesadores que se encuentran en el dominio de balance. La regla de selección determina la carga más conveniente que será migrada entre los procesadores.

3.1 REGLA DE MEDIDA DE CARGA

La cantidad de trabajo que realiza un procesador está en relación directa con el número de moléculas que pertenecen a su espacio de simulación, es por ello que la medida natural de carga sea el número de moléculas que posee un procesador. Esta medida de carga ha sido utilizada con buenos resultados en [2] y [3] pero no considera los efectos de la aglomeración de moléculas en términos de cantidad de trabajo a realizar. El incremento en el número de moléculas en un área hace que el número de vecinos de las moléculas de esa zona crezca , aumentando exponencialmente la cantidad de trabajo. Moléculas con un mayor número de vecinos ocasionan un mayor volumen de trabajo. En este sentido, parece más conveniente usar medidas de carga ligadas al tiempo. Dado que la simulación se hace a intervalos de 20 iteraciones, utilizaremos como medida de carga el tiempo que el procesador se demora en ejecutar las últimas 20 iteraciones de la simulación (T_{actual}) dado el número de moléculas presentes en el espacio de dicho proceso.

3.2 REGLA DE INTERCAMBIO DE INFORMACION

La carga de trabajo de los distintos procesadores será reportada cada vez que la simulación recalcula la lista de moléculas vecinas, es decir, cada 20 iteraciones.

3.3 REGLA DE INICIO DE BALANCE

La métrica usada para decidir si el sistema está balanceado o no es el Coeficiente de Variación (COV) de los tiempos de ejecución de las últimas 20 iteraciones de la simulación. Esta métrica permite escoger el nivel de desbalance en el cual será activado la operación propiamente dicha de balance de carga. Cuando el coeficiente de variación de los tiempos exceda el valor prefijado entonces se activará la operación de balance.

3.4 OPERACION DE BALANCE

Para llevar a cabo la operación de balance propiamente dicha, se escogió el método Generalized Dimension Exchange (GDE) [11]. GDE permite calcular el flujo de carga que deberá migrar entre los distintos nodos para alcanzar un balance de carga. El método GDE puede ser clasificado como un algoritmo de balance de carga determinístico y que sólo permite comunicaciones con los vecinos. Cada procesador compara su carga con la de sus vecinos uno tras otro. En cada una de estas comparaciones, el procesador trata de igualar su carga con la de sus vecinos. Esta operación es repetida hasta que la información de la carga de trabajo se propaga hasta todos los nodos de la red. En este punto surgen dos interrogantes: cuánta carga de trabajo debe recibir o ceder un procesador y cuántas veces se debe intercambiar la información para propagar la información hasta todos los nodos de la red. Esto dependerá de la topología escogida. En nuestro caso se escogió una topología tipo cadena. El flujo de carga

entre dos procesadores vecinos será igual a la cantidad de moléculas que debe migrar de forma tal que el tiempo necesario para procesar la nueva carga sea aproximadamente igual en los dos procesadores involucrados. Este proceso de intercambio y de igualación de carga entre los procesadores se repite *(diametro de la cadena / 2)* veces [11].

El cálculo del flujo de carga a migrar entre los distintos nodos puede ser descrito de la siguiente forma:
Sea

- P_i el procesador i en donde se ejecuta la aplicación (0 = *i* = n-1)
- $T_{actual}(P_i, j)$ Tiempo de ejecución en el procesador *i* del *j*-ésimo grupo de 20 iteraciones
- $M(P_i, j)$ Número de moléculas asignadas al procesador *i* en el *j*-ésimo grupo de 20 iteraciones
- *Moléculas* Número total de moléculas

Lo que se desea es que en un intervalo cualquiera de iteraciones se cumpla que

$$T_{actual}(P_0, j) \cong T_{actual}(P_1, j) \cong \ldots \cong T_{actual}(P_{n-1}, j)$$

De esta forma se garantiza que todos los nodos ejecuten de forma sincronizada y que finalizarán su ejecución aproximadamente al mismo tiempo. El problema que deseamos resolver puede reformularse de la siguiente forma para el j-ésimo grupo de 20 iteraciones

$$\textit{Minimizar } y = \textit{máximo } \{ \; M(P_0, j) * T_{actual}(P_0, j-1) / M(P_0, j-1) \;,$$

$$M(P_1, j) * T_{actual}(P_1, j-1) / M(P_1, j-1) \;, \ldots,$$

$$M(P_{n-1}, j) * T_{actual}(P_{n-1}, j-1) / M(P_{n-1}, j-1) \}$$

Sujeto a:

$$M(P_0, j) + M(P_1, j) + \ldots + M(P_{n-1}, j) = \textit{Moléculas}$$

$$M(P_i, j) = 0 \;\; \forall i \quad \text{y} \quad M(P_i, j) \text{ entero}$$

donde $T_{actual}(P_i, j-1) / M(P_i, j-1)$ es un aproximado de lo que costó ejecutar el grupo anterior de 20 iteraciones para una molécula en el nodo P_i.

Este problema puede ser resuelto usando GDE, en donde en cada iteración *m* del algoritmo se resuelve el siguiente problema:

Para dos nodos vecinos P_i y P_k se desea que

$$M(P_i, j) * T_{actual}(P_i, j-1) / M(P_i, j-1) = M(P_k, j) * T_{actual}(P_k, j-1) / M(P_k, j-1) \quad \text{y}$$

$$M(P_i, j) + M(P_k, j) = M(P_i, j-1) + M(P_k, j-1)$$

Al resolver estos sistemas de ecuaciones obtenemos la carga que debería tener cada uno de los dos nodos para que el tiempo estimado de ejecución para el siguiente grupo de 20 iteraciones sea aproximadamente igual. Este proceso se repite para todos los vecinos. Este proceso corresponde a una iteración del algoritmo GDE. Esto se repite un cierto número de veces que dependerá del diámetro de la topología.

Una vez obtenida la cantidad de carga que debe ser intercambiada entre los procesadores, es necesario migrar las moléculas que sean necesarias. Para migrar moléculas en la descomposición espacial de DM, se redefine el espacio asignado a cada procesador. Para ello, las moléculas de cada procesador son ordenadas de acuerdo a su distancia a los bordes actuales del subespacio de simulación. Una vez ordenadas las moléculas, se mueve el borde del espacio de forma tal que queden fuera del espacio físico del procesador y pasen a formar parte del espacio del procesador vecino.

4. EXPERIMENTOS Y RESULTADOS

El algoritmo fue probado sobre un patrón de prueba que consiste de 32000 moléculas simuladas en un paralelepípedo de tres dimensiones con condiciones periódicas de borde. Se basa en un modelo de Lennard-Jones con densidad reducida $\rho^* = 0.3$ y temperatura reducida $T^* = 0.8$. Dado que esta sustancia es homogénea y no presenta significativos niveles de desbalance, se cambió periódicamente la temperatura de la misma para forzar ciertos

niveles de desbalance. Los experimentos se realizaron sobre un cluster de cuatro nodos duales Pentium III de 800 MHz con 512MB de memoria RAM conectados mediante una red Myrinet.

En la figura 1, se muestran los tiempos de ejecución de cada 20 iteraciones de la simulación sin activación del balance de carga. A partir de la figura se observa que luego de la iteración 12000 las moléculas comienzan a concentrarse en el centro del paralelepípedo (nodos 1 y 2), esto hace que los procesadores 1 y 2 demoren más tiempo que los procesadores 0 y 3 en cada iteración de la simulación.

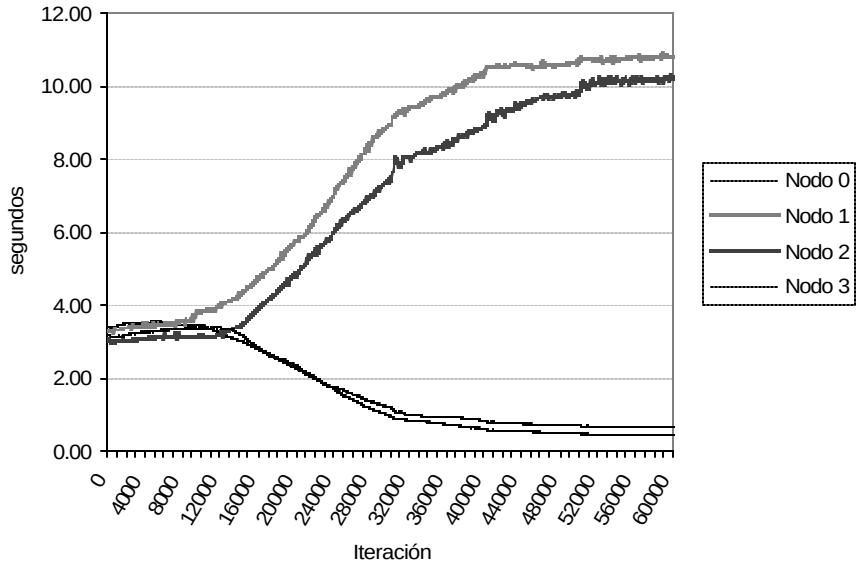

Figura 1. Tiempo de ejecución de cada 20 iteraciones (Tactual) sin Balance de Carga

En la figura 2, se presenta el comportamiento de la sustancia cada 20 iteraciones cuando se aplica el balance de carga. En este caso, el balance de carga se activa cuando el coeficiente de variación de T_{actual} es igual a 0.02. Cuando el COV es bajo, el balanceador se activa ante pequeños desbalances en el tiempo de ejecución de los procesadores. A partir de esta figura, podemos observar que los tiempos se reducen considerablemente en relación a la figura 1, y que los distintos procesadores que intervienen presentan un comportamiento bastante similar.

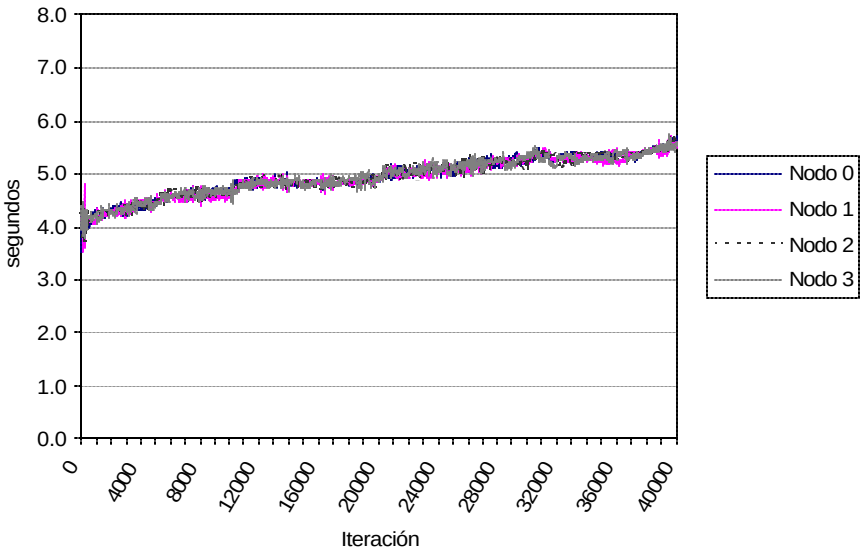

Figura 2. Tiempo de ejecución de cada 20 iteraciones (T_{actual}) con Balance de Carga activado con COV=0.02

En la figura 3, se muestra la variación en el número de moléculas presentes en cada procesador cuando se aplica el balance de carga. Podemos observar que aún cuando la variación entre los tiempos de ejecución de los procesadores es muy baja (se mantiene un balance con un COV=0.02 en tiempo), hay una variación alta en cuanto al número de moléculas (momentos en los que un proceso tiene alrededor de 7000 moléculas mientras otro tiene unas 9000).

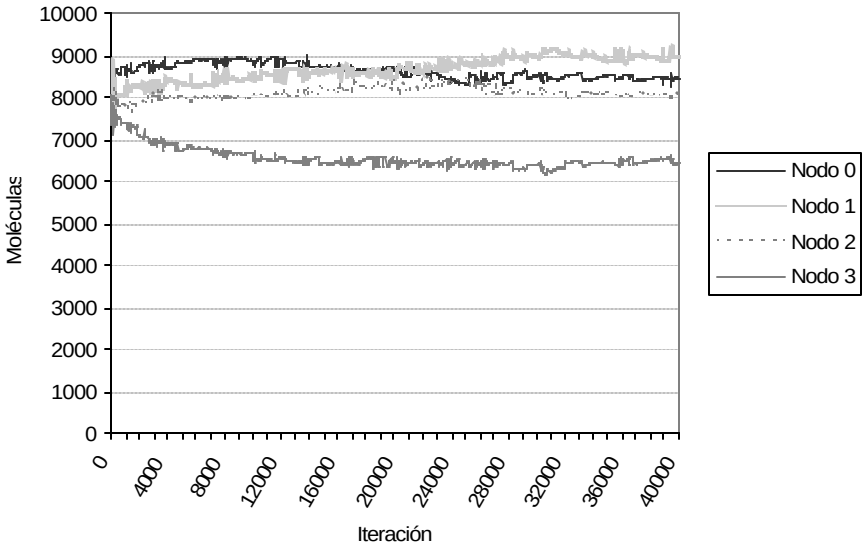

Figura 3. Número de moléculas por procesador con Balance de Carga activado con COV=0.02

La Figura 4 muestra para diferentes márgenes de tolerancia al desbalance, la diferencia en tiempo de ejecución cuando el balanceador utiliza las medidas de carga basadas en moléculas y en tiempo. La medida de carga basada en tiempo permite mejoras de hasta 6.5% sobre la basada en cantidad de moléculas. La escogencia del COV más apropiado depende de la cantidad de *overhead* que puede generar una migración de moléculas en un instante dado de la simulación y la ganancia estimada al lograr el balance del sistema [3].

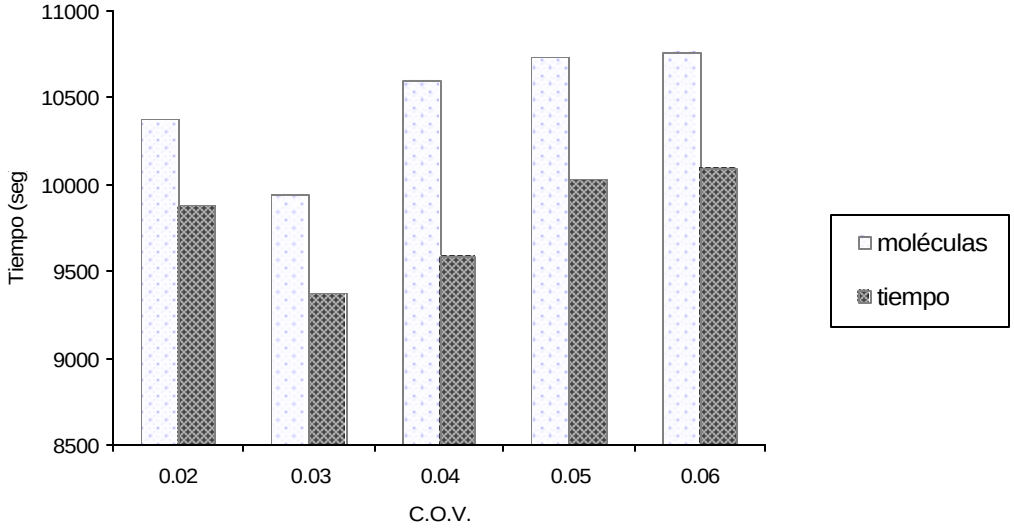

Figura 4. Comparación de tiempos de ejecución para medidas de carga basadas en moléculas y tiempo

La Tabla 1 muestra el tiempo total de ejecución y el porcentaje de disminución de tiempo obtenidos para distintos experimentos llevados a cabo variando el nivel de activación del balance. La disminución en el tiempo está entre 32 por ciento y 43 por ciento. Cuando el balance se activa con COV igual a 0.02, se está incurriendo en un mayor *overhead* debido a la constante migración de moléculas de un subespacio de simulación a otro. Cuando se activa con COV igual a 0.08, se permite un mayor desbalance por mayor cantidad de tiempo pero aún así se obtiene una disminución considerable en el tiempo total de ejcución.

	Balanceador utiliza medida de carga basada en tiempo				
	COV=0.02	COV=0.03	COV=0.04	COV=0.05	COV=0.08
Tiempo Total	10709.77	9752.77	9777.82	10174.38	11518.74
% disminución	37.47	43.05	42.91	40.59	32.74

Tabla 1. Tiempo total de ejecución y porcentaje de disminución de los tiempos

5. CONCLUSIONES

Aplicaciones dinámicas de cómputo intensivo como DM, requieren la migración de datos de un proceso a otro con miras a garantizar que todos los procesadores tengan una cantidad de trabajo equivalente. Cuando esta migración se logra de manera adecuada, se minimiza el tiempo de ejecución de la aplicación. En DM la cantidad de trabajo a realizar por cada procesador está directamente relacionada con el número de moléculas que tiene en su espacio de simulación. Por tanto, lograr que todos los procesadores mantengan aproximadamente el mismo número de moléculas, corrige las grandes discrepancias de tiempo que se originan de la natural reagrupación de las moléculas durante la simulación. Sin embargo, el tiempo total de ejecución no es óptimo. Nuestro trabajo logró optimizar el tiempo de ejecución utilizando una medida de carga relacionada con el tiempo de ejecución de un fragmento de ejecución en el procesador en particular.

Actualmente nuestros esfuerzos están dirigidos en la aplicación del balance de carga basado en tiempos sobre arquitecturas heterogéneas y en el ajuste de la regla de inicio de balance de carga.

Referencias

[1] Cortes, A., Ripoll, A., Senar, M., Pons P. and Luque, E. *On the Performance of Nearest Neighbor Load Balancing Algorithms in Parallel Systems.* Proceedings of the Seventh Euromicro Workshop on Parallel and Distributed Processing, Funchal, Portugal, Febrero 1999.

[2] Di Serio, A. and Ibánez, M.B. *Distributed Load Balancing for Molecular Dynamics Simulations*. Proceedings of the 16th Annual International Symposium on High Performance Computing Systems and Application, Moncton, New-Brunswick, Canada, June 2002. IEEE Computer Society, 284-289.

[3] Di Serio, A. and Ibánez, M.B. *Evaluation of a Nearest/Neighbor Load Balancing Strategy for Parallel Molecular Simulations in MPI Environment*. Recent Advances in Parallel Virtual Machine and Message Passing Interface. Volume 2474, (Octubre, 2002), pp. 226-233.

[4] Esselink, K., Smit, B. and Hilbers. *Efficient Parallel Implementation of Molecular Dynamics on a Toroidal Network: Multi-particle Potentials*. Journal of Computer Physics. Vol. 106, pp. 108-114, 1993.

[5] Finchman. *Parallel Computers and Molecular Simulation.* Molecular Simulation. Vol. 1, 1987.

[6] Haile, J.M. *Molecular Dynamics Simulation.* Wiley Inters-Science, 1992.

[7] Hegarty, D.F. and Kechadi, T. *Topology Preserving Dynamic Load balancing for Parallel Molecular Simulations.* Super Computing, 1997.

[8] Plimpton, S. *Fast Parallel Algorithms for Short-range Molecular Dynamics.* Journal of Computational Physics. Vol 117, pp. 1-19, 1995.

[9] Sato, N. and Jézéquel, J.M. *Implementing and Evaluating an Efficient Dynamic Load-Balancer for Distributed Molecular Dynamics Simulation.* Proceedings of the 2000 International Workshops on Parallel Processing. IEEE, 2000.

[10] Willebeek-LeMair, M. and Reeves, A. *Strategies for Dynamic Load Balancing on High Parallel Computers.* IEEE Transactions on Parallel and Distributed Systems, 9(3): 235-248, Marzo 1998.

[11] Xu, C. and Lau, F.C.M. *Load Balancing in Parallel Computers. Theory and Practice.* Kluwer Academic Publishers, 1997.