

Universidad
Rey Juan Carlos

TESIS DOCTORAL

Simulación Multi-GPU escalable de dinámica molecular en arquitecturas de computación de alto rendimiento

Autor:

Gonzalo Nicolás Barreales

Director:

Dr. Alberto Sánchez Campos

Programa de Doctorado en Tecnologías de la Información y las Comunicaciones

Escuela Internacional de Doctorado

Abril 2021

Nothing in life is to be feared, it is only to be understood. Now is the time to understand more, so that we may fear less.

- Marie Curie

Agradecimientos

Quiero empezar haciendo especial mención al Grupo de Modelado y Realidad Virtual (GMRV) de la Universidad Rey Juan Carlos, donde se ha desarrollado esta tesis, al Barcelona Supercomputing Center (BSC) y a Amazon Web Services (AWS) por su colaboración para poder realizar los experimentos mostrados. Me gustaría empezar agradeciendo a todas las personas que componen GMRV por acogerme desde el primer momento y por prestarme su ayuda y apoyo durante todo este tiempo. En especial quiero dar las gracias al director de esta tesis, Alberto Sánchez, por toda la dedicación y paciencia que ha tenido a lo largo de estos años. También me gustaría agradecer a todos mis compañeros de MSLab y del despacho 019, y a todas las personas que han pasado alguna vez por aquí y han dejado su huella, por todos los momentos que hemos pasado juntos y todo lo que he aprendido con ellos. Quisiera dar las gracias especialmente a Marcos Novalbos por su inestimable ayuda en el desarrollo de este trabajo y por todas las horas que se ha pasado conmigo depurando código, y a Aaron Sújar por todo su apoyo a la hora de escribir y publicar, y todo el tiempo que ha invertido en ayudarme mejorar mis dotes de escritor.

También quiero mostrar mi agradecimiento a toda mi familia, que siempre me han apoyado y se han interesado por mi trabajo durante todos estos años. A mis padres Nuria y Gonzalo, y a mi hermano Diego, muchas gracias por vuestro apoyo incondicional y por enseñarme y transmitirme que con esfuerzo y dedicación todo se puede conseguir. Y por su puesto a Eva, que lleva en mi vida más tiempo del que puedo recordar y me ha apoyado, ayudado y aguantado todos estos años, muchas gracias.

Por último, quisiera dar las gracias a todo el que lea estas palabras por dedicar su tiempo y esfuerzo a revisar este trabajo.

Muchas gracias a todos.

Resumen

Actualmente en el mundo de las tecnologías de la información y las comunicaciones están tomando una gran relevancia los sistemas que permiten realizar cálculos de manera paralela, ya sea utilizando diversos equipos conectados a una misma red, lo que llamamos clúster, o mediante un solo dispositivo con múltiples procesadores. Tanto es así que en la actualidad existen grandes centros de supercomputación destinados a realizar grandes cantidades de cálculos con el objetivo de poder resolver problemas computacionalmente muy costosos. Dispositivos como las GPUs (Graphics Processor Unit) se han ido adaptando a este escenario para poder realizar de forma paralela cálculos de ámbito genérico de una forma eficiente, fragmentando el problema y realizando cálculos menos costosos en cada uno de sus procesadores.

Estas técnicas de computación paralela encajan muy bien con diversos algoritmos de simulación mecánica. Concretamente, las técnicas de simulación basadas en partículas, como la simulación de fluidos o la simulación de dinámica molecular, tienen las características deseables para poder utilizar estas arquitecturas multi-procesador; siendo la principal la necesidad de realizar miles de cálculos que implican las mismas operaciones sobre conjuntos de datos diferentes. Por esta razón el uso de GPUs es ideal para esta aplicación. Esta tesis se va a centrar en el uso de estas arquitecturas multi-procesador para llevar a cabo el proceso de simulación de dinámica molecular de forma escalable sobre las mismas. En este tipo de simulación los átomos se representan como partículas, las cuales pueden estar enlazadas a otras partículas, y además existen interacciones entre ellas no relacionadas con los enlaces, como son las fuerzas electrostáticas y las fuerzas de Van der Waals. Para cada átomo es necesario calcular todas las fuerzas que actúan sobre este en cada paso de tiempo, lo

que se puede hacer en un procesador de la GPU. De esta forma una sola GPU puede calcular las interacciones atómicas de un sistema molecular compuesto por miles de átomos en un tiempo mucho mas reducido que el que utilizaría una CPU. Por lo tanto, teóricamente un clúster compuesto de múltiples GPUs seria capaz de realizar estos mismos cálculos para millones, o incluso miles de millones de átomos en un tiempo similar. Sin embargo, al paralelizar la simulación de un sistema molecular es necesario disponer de cierta duplicidad en los datos, lo que reduce la escalabilidad del sistema. Esto es debido a que los cálculos de las fuerzas de simulación dependen tanto de los átomos situados alrededor de cada uno de ellos en un radio predefinido, así como de todos los demás átomos del sistema. Por lo tanto, cada GPU debe almacenar los datos de los átomos para los que tiene que calcular las fuerzas, así como los datos de los que se encuentran alrededor que pueden están localizados en otra GPU. A lo largo de esta tesis se mostrarán diversas técnicas para optimizar la paralelización de simulación de dinámica molecular, resolviendo la problemática expuesta de forma escalable y llegando a simular un sistema molecular de más de 1000 millones de átomos en solo 128 GPUs.

La simulación de dinámica molecular en arquitecturas de computación masiva, como es un entorno Multi-GPU, no solo implica la problemática de realizar los cálculos en paralelo de manera óptima, sino que también es necesario gestionar la infraestructura donde se va a llevar a cabo la ejecución. Este tipo de arquitecturas se encuentran generalmente en grandes centros de supercomputación o en servicios de cloud computing. El acceso a los centros de supercomputación puede ser bastante costoso y, además, su disponibilidad es limitada. Esto hace de los entornos cloud una buena alternativa al uso de los supercomputadores. Sin embargo, estos entornos también suponen un elevado coste si se desea contratar el uso de GPUs, lo que hace necesario tener la capacidad de gestionar su uso para que sea lo más eficiente posible. Otro punto a tener en cuenta a la hora de realizar la simulación es el preprocesado

de los datos de los sistemas moleculares. Los datos de estos sistemas se pueden, o bien modelar, o bien obtener de repositorios públicos que se encargan de recopilar esta información. Estos datos por lo general se obtienen con fines científicos en diversos campos, no solo en el de simulación. Esto supone que pueden estar incompletos o no ser realistas en términos físicos, por lo que es necesario tratarlos previamente para poder realizar una simulación correcta. Estos puntos se tratan a lo largo de la presente tesis, detallando el desarrollo de una aplicación SaaS que permite gestionar los recursos del entorno cloud de la forma más eficiente posible. Esta herramienta además permite automatizar tanto el proceso de simulación como el preprocesado de los datos molecular, simplificando estos procesos para el usuario y permitiendo que los usuarios no especializados en dinámica molecular puedan llevar a cabo la simulación de una manera más sencilla.

Abstract

Currently, in the field of information and communication technologies, systems that allow massive parallel calculations, either by using several computers connected by a network, named clusters, or employing a single device with a multi-processor, are gaining great relevance. In fact, there are large supercomputing centers designed to perform a huge amount of calculations in order to solve problems very expensive in computational terms. Devices such as GPUs (Graphic Processor Unit) are able to perform parallel calculations of generic scope efficiently, fragmenting the problem and performing less costly calculations on each processor.

These parallel computing techniques fit very well with multiple mechanical simulation algorithms. Specifically, particle-based simulation techniques, such as fluid simulation or molecular dynamics simulation, have the desirable characteristics to be able to use these multi-processor architectures, e.g. the need to perform thousands of computations involving the same operations on different datasets. For this reason, the usage of GPUs is ideal for this application. This thesis is focused on the use of these multi-processor architectures to carry out the molecular dynamics simulation process in a scalable way. In this type of simulation, atoms are represented as particles, which can be bonded to other particles. There are also interactions among them that are not related to bonds, such as electrostatic forces and Van der Waals forces. For each atom is necessary to calculate all the forces acting on it at each time step, which can be computed on a GPU processor. In this way, a single GPU can calculate the atomic interactions of a molecular system composed of thousands of atoms in a much shorter time than a CPU. Therefore, theoretically, a cluster composed of multiple GPUs would be able to perform these same calculations for millions,

or even billions of atoms in a similar time. However, parallelizing the simulation of a molecular system requires some duplication of data, which reduces the scalability of the system. This is due to the calculations of the simulation forces depend on both, the atoms located around each atom within a predefined radius, and all the other atoms located in the system. This means that each GPU must store data of the atoms for which it calculates the forces, as well as data of the surrounding atoms that may be located on another GPU. Several techniques to optimize the parallelization of molecular dynamics simulation have been designed in this Ph.D. thesis, solving the exposed problem and reaching to simulate a molecular system of more than 1000 million atoms in only 128 GPUs.

Molecular dynamics simulation in massive computing architectures not only involves the problem of performing parallel computations optimally, but it is also necessary to manage the infrastructure where the execution is carried out. These types of architectures are generally found in large supercomputing centers or cloud computing services. Access to supercomputing centers can be quite expensive and availability is limited. This makes cloud environments an option to consider. However, these environments are also expensive if users want to use GPUs, which makes it necessary to have the ability to manage these environments as efficiently as possible. Another point to consider when performing simulation is the preprocessing molecular systems data. Data from these systems can either be modeled or obtained from public repositories that collect this information. These data are usually obtained for scientific purposes in multiple fields. This implies that this data may be incomplete or unrealistic in physical terms, so it is necessary to treat them previously in order to be able to perform a correct simulation. These points are discussed throughout this Ph.D. thesis, detailing the development of a SaaS application that allows users to manage the resources of the cloud environment efficiently way. This tool also allows users to automate both the simulation process and the preprocessing of molecular

data, simplifying these processes and enabling users not specialized in molecular dynamics to carry out the simulation in a simpler way.

Índice general

Agradecimientos **III**

Resumen **V**

Abstract **IX**

1. Introducción **1**

1.1. Introducción 1

1.2. Antecedentes 3

1.3. Hipótesis y objetivos 4

1.4. Organización del documento 6

2. Estado del arte **11**

2.1. Dinámica molecular 11

2.1.1. Fuerzas implicadas en dinámica molecular 12

2.1.2. Integradores o métodos de integración numérica 14

2.1.2.1. Método de Euler explicito 15

2.1.2.2. Método de Verlet 17

2.1.2.3. Método RESPA 18

2.2. Dinámica molecular en arquitecturas paralelas 18

2.2.1. Particionado 20

2.2.2. Fuerzas de enlace 20

2.2.3. Fuerzas electrostáticas de corto alcance y Van der Walls . . . 21

2.2.4. Fuerzas electrostáticas de largo alcance 22

2.2.4.1. *Multilevel Summation Method* (MSM) 24

2.3. Simuladores en arquitecturas HPC 26
2.4. Computación cloud 28

3. Planteamiento del problema **31**
3.1. Dinámica molecular en arquitecturas masivamente paralelas 31
3.2. Arquitectura en entornos de cloud computing 32
3.3. Proceso de datos para Dinámica molecular 33

4. Simulación de Dinámica Molecular en entornos HPC **35**
4.1. Dinámica molecular escalable en entornos Multi-GPU 35
4.1.1. Datos compartidos de átomos 36
4.1.2. Datos de enlaces compartidos 40
4.1.3. Optimizaciones para sistemas moleculares pequeños 43
4.2. Algoritmo de simulación en paralelo 43
4.2.1. Arquitectura Hardware Multi-GPU 50
4.3. Procesado de datos moleculares 51
4.3.1. Topología de los sistemas moleculares 53
4.3.2. Solución en agua de los sistemas moleculares 53
4.3.3. Equilibrio energético del sistema molecular 54

5. Dinámica molecular en entornos de Cloud Computing **55**
5.1. Despliegue de arquitecturas en servicios Cloud 56
5.2. Proceso de ejecución e interacción con el usuario 60

6. Evaluación **63**
6.1. Rendimiento del algoritmo de dinámica molecular 63
6.2. Evaluación en entornos de Cloud Computing 79

7. Conclusiones y líneas futuras **91**

7.1. Conclusiones 91

7.1.1. Simulación paralela de dinámica molecular 92

7.1.2. Gestión y preprocesado para simulación de dinámica molecular 93

7.2. Trabajo futuro 95

7.2.1. Balanceo de carga 95

7.2.2. Cálculo de fuerzas electrostáticas de largo alcance 95

7.2.3. Ampliación de herramienta cloud computing 96

Bibliografía **97**

Índice de figuras

2.1. Comparación de particionado espacial binario y lineal 21
2.2. Diagrama con las principales operaciones de MSM. 25

4.1. Distribución en 2D del sistema molecular entre los diferentes ECs. . 37
4.2. Tipos de celdas. 39
4.3. Enlaces distribuidos entre dos ECs. 41
4.4. Relación entre átomos y enlaces. 49
4.5. Arquitectura remota con GPU-direct. 51
4.6. Preprocesado realizado para pasar de los datos moleculares iniciales a un sistema molecular equilibrado y preparado para llevar a cabo la simulación. 52

5.1. Diagrama de la arquitectura en el entorno de Cloud Computing. . . . 57
5.2. Flujo de trabajo de la herramienta en todas sus etapas. 61

6.1. Tiempo de simulación utilizando 8 millones de átomos en moléculas de agua de forma distribuida. 66
6.2. Tiempo de actualización de los datos de un sistema molecular compuesto por 8 millones de átomos de forma distribuida formado solo por moléculas de agua. 67
6.3. Resultados obtenidos simulando 100 pasos con un bloque de moléculas de agua compuesto de 8 millones de átomos con el algoritmo propuesto y NAMD en un entorno de pruebas de AWS con GPUs Nvidia K80. 68

6.4. Porcentaje de tiempo necesario para actualizar los átomos y los enlaces en relación a la simulación del sistema de moléculas de agua compuesto por 8 millones de átomos en un entorno de AWS formado por GPUs Nvidia K80. 69
6.5. Uso de memoria del algoritmo propuesto por cada EC para el conjunto de moléculas de agua simulado en la Figura 6.6. 70
6.6. Rendimiento en comparación con NAMD en escala logarítmica, incrementando el número de GPUs y de átomos por EC para simular moléculas de agua con decenas de millones de átomos. 71
6.7. 12 estructuras moleculares del sistema 4V4L, compuesto por aproximadamente 7.5 millones de átomos, incluyendo las moléculas de agua que lo envuelven. 73
6.8. Sistema molecular 3J3Q, cápsida del virus HIV-1 compuesta por cerca de 66 millones de átomos, incluyendo el agua que lo rodea. 74
6.9. Comparación del tiempo de simulación de la solución propuesta y de NAMD utilizando los sistemas moleculares 12x4V4L y 3J3Q en el entorno de pruebas de AWS con GPUs Nvidia K80. 75
6.10. Uso de memoria por EC en particiones 3D binarias distribuidas de forma equitativa entre los diferentes ECs de la solución propuesta utilizando los sistemas moleculares 12x4V4L y 3J3Q en el entorno de pruebas de AWS con GPUs Nvidia K80. 76
6.11. Gigamolécula compuesta por 96 virus 4UDF, replicada para obtener 1000 millones de átomos incluyendo el agua que lo rodea. 77
6.12. Comparación de resultados de ambas variantes del algoritmo de simulación en el clúster Minotauto utilizando un sistema de 8 millones de átomos formado por moléculas de agua. 78
6.13. Estructura molecular 4V4L obtenida del Protein Data Bank. 80

6.14. Diagrama detallado de la arquitectura del entorno de pruebas utilizado. 81
6.15. Relación de las claves de acceso a AWS. 82
6.16. Creación de un espacio de trabajo para un sistema molecular. 83
6.17. Proceso de carga del sistema molecular. 84
6.18. Etapa de creación de PSF. 85
6.19. Etapa para añadir una solución de agua. 86
6.20. Etapa de equilibrado del sistema molecular. 87
6.21. Configuración de una simulación de dinámica molecular personalizada. 88
6.22. Configuración de la simulación de dinámica molecular. 89

Índice de tablas

4.1. Comparación entre un diccionario de datos almacenado en GPU y una tabla hash, accediendo a un millón de números aleatorios en una Nvidia GeForce GTX 1070 42

6.1. Medidas obtenidas al simular un sistema de moléculas de agua en 128 GPUs Nvidia Tesla K80. 72
6.2. Medidas obtenidas al simular un sistema de moléculas de agua en 96 GPUs Nvidia Tesla V100. 72
6.3. Medidas obtenidas al simular 1 billón de átomos con 96 virus 4UDF, replicados en 96 GPUs Nvidia Tesla V100. 78
6.4. Tiempo de ejecución para cada etapa del proceso. 90

Índice de Algoritmos

1. Algoritmo para la resolución del método RESPA 19
2. Integrador Verlet/RESPA para Multi-GPU. 45
3. Identificación de los átomos que pasan a estar localizados en otro EC. 45
4. Obtención de los identificadores de los enlaces que se deben enviar a la interfaz. 46
5. Obtención de los identificadores de los átomos que se van a compartir con las celdas de interfaz de otros ECs. 47

Capítulo 1

Introducción

1.1. Introducción

La simulación visual de efectos mecánicos está tomando cada vez más fuerza en diferentes áreas. Esto es debido a que obtener una realimentación visual adecuada puede significar una toma de decisiones más ágil y eficiente en cualquier investigación, no solo en el caso de experiencias interactivas como videojuegos, o animaciones utilizadas en televisión, publicidad o cine, sino también en herramientas de diseño que pueden utilizarse con fines como son médicos, comerciales o científicos.

Diversos campos de estudio requieren simular el comportamiento de miles de elementos (partículas) que interactúan entre si, como es el caso de la física de fluidos o la dinámica molecular. En estos campos de estudio, la simulación implica una gran cantidad de datos y cálculos debido a la alta complejidad de los algoritmos necesarios para su resolución, lo que puede suponer el uso una gran cantidad de tiempo y recursos. Para resolver este tipo de problemas se utilizan técnicas de procesamiento paralelo que permiten reducir el tiempo requerido para realizar los cálculos. Este tipo de técnicas aprovechan las arquitecturas HPC (*High Performance Computing* o Computación de Altas Prestaciones) para realizar la simulación, ya sea mediante entornos de supercomputación o servicios en la nube. En concreto, esta tesis profundizará en la creación de técnicas escalables de procesamiento paralelo y distribuido en entornos Multi-GPU para resolver problemas complejos de simulación de dinámica

molecular.

En concreto, en el campo de la dinámica molecular suceden múltiples interacciones a nivel atómico de sistemas compuestos por un gran número de partículas [52]. La simulación de dinámica molecular trata de recrear, mediante el uso de recursos computacionales, estas interacciones para estimar el comportamiento de los sistemas moleculares. Este cálculo no puede realizarse de forma analítica, sino que requiere de un proceso complejo de simulación. Entre otras cosas, la resolución de dinámica molecular permite predecir la estructura de las proteínas y su comportamiento para el diseño de nuevos medicamentos. Por ejemplo, la industria farmacéutica trata de anticipar la afinidad de unión de un ligando en un receptor [51] a partir de las interacciones producidas entre todos los átomos para poder analizar sus propiedades antes de sintetizarlas.

La dinámica molecular considera las características de los átomos y las fuerzas de interacción entre ellos para reproducir de manera físicamente correcta su movimiento durante un tiempo determinado. La utilización de métodos de discretización permite dividir la simulación en pasos o *time-steps*, calculándose en cada *time-step* las fuerzas para cada uno de los átomos del sistema, además de las nuevas posiciones de cada uno de ellos mediante integración numérica. Solucionar un problema de dinámica molecular supone un elevado coste, tanto a nivel computacional como en uso de memoria.

Los supercomputadores son la plataforma ideal para resolver problemas de dinámica molecular. Tanto es así que los sistemas moleculares más grandes que se han simulado hasta la fecha han utilizado los supercomputadores de las posiciones más altas del TOP 500 [57], con cientos de miles de procesadores [26, 33, 19]. Asimismo, la dinámica molecular es un problema masivamente paralelo que encaja muy bien con arquitecturas de una sola GPU. Sin embargo, no se han abordado de forma completa soluciones en entornos Multi-GPU, los cuales se encuentran en la mayoría de

centros de supercomputación actuales. Además, las GPUs están bastante limitadas en cuanto a su capacidad de memoria, y simular grandes sistemas moleculares obliga a distribuir el proceso para reducir su coste computacional.

La simulación de dinámica molecular necesita, además de una alta capacidad computacional, un tratamiento previo de los datos sobre los que se va a trabajar. Esto es debido a que los datos moleculares disponibles en repositorios abiertos, como es el PDB (*Protein Data Bank* [10]), contienen únicamente información acerca de la posición de cada átomo y la molécula a la que pertenece. Para llevar a cabo la simulación hay que tratar estos datos a lo largo de una serie de procesos, aportando los parámetros que influyen en las interacciones atómicas. Este proceso se puede llevar a cabo en entornos Multi-GPU tanto de supercomputación como de cloud computing. Sin embargo, el uso de estos últimos implica la necesidad de gestionar todos los recursos, el software, la configuración del servicio y el almacenamiento de los datos de la simulación por parte del usuario, lo que puede ser tedioso para expertos no relacionados con el campo de las TIC.

1.2. Antecedentes

El objetivo actual de la investigación en simulación consiste en desarrollar nuevas técnicas que permitan acelerar el proceso sin comprometer la precisión del resultado. El uso de las GPUs como elementos de cómputo está cada vez más extendido. Tanto es así que los grandes centros de supercomputación del top 500 [57] basan gran parte de su capacidad computacional en su utilización. Esto supone que es posible disponer de entornos Multi-GPU para acelerar los cálculos, sobre todo para aquellas simulaciones complejas que requieren tiempos de ejecución muy elevados. Dentro de esta área cumplen estas características la simulación de sistemas de partículas como los utilizados en simulación de dinámica molecular, entre otros. La utilización de estos

entornos permite realizar múltiples cálculos de forma paralela, lo que los convierte en la herramienta adecuada para el cálculo de simulaciones complejas. Concretamente en esta tesis se va a tratar la simulación de dinámica molecular en arquitecturas Multi-GPU, utilizando entornos de supercomputación y computación cloud.

El objetivo principal de esta tesis es aplicar la capacidad de cálculo de las tarjetas gráficas, de forma que se pueda reducir drásticamente el tiempo de simulación sin que afecte a la precisión del resultado obtenido. Este trabajo parte del desarrollo de una tecnología de simulación propia de dinámica molecular altamente eficiente en entornos locales Multi-GPU, donde cada GPU es tratada como un procesador propio, frente a otras propuestas que hacen uso de sistemas de supercomputación tradicionales y, en el caso de utilizar GPUs, lo hacen únicamente como co-procesadores. Esta propuesta permitió una reducción del tiempo de cálculo de las simulaciones como demuestran Novalbos et al. en trabajos anteriores [42, 41]. Sin embargo, esta tecnología no es completamente escalable y está planteada únicamente para entornos locales, con un reducido número de GPUs, lo que supone una limitación a la hora de tratar de simular grandes sistemas moleculares.

1.3. Hipótesis y objetivos

La simulación de sistemas de partículas masivos, como es el caso de la dinámica de fluidos o de la dinámica molecular, implica un coste computacional elevado. Este tipo de simulaciones se basan en el calculo de un gran número de fuerzas que actúan sobre cada una de las diferentes partículas del sistema. Existen múltiples herramientas que pueden reducir los tiempos de ejecución de las simulaciones paralelizando parte de los cálculos en entornos de supercomputación o en servicios de cloud computing. La mayor parte de estos entornos utilizan GPUs para aumentar su capacidad de procesamiento y reducir drásticamente los tiempos de ejecución. Sin

embargo, aunque hay diversos simuladores de dinámica molecular que utilizan estos elementos para agilizar sus cálculos, solo utilizan las GPUs para las operaciones más costosas, o bien su arquitectura limita el número de GPUs que se pueden utilizar. Por estas razones, esta tesis plantea comprobar si **"los algoritmos de simulación de dinámica molecular pueden ser diseñados de forma completamente escalable para entornos Multi-GPU, donde se utilice la GPU como el elemento de computo principal para todas las interacciones y no se limite el número de GPUs implicadas"**. Para poder demostrar esta hipótesis se desarrollarán algoritmos que optimicen y reduzcan todo lo posible la comunicación entre GPUs sin comprometer la escalabilidad del sistema ni el uso de memoria.

A partir de la hipótesis enunciada se pueden extraer los siguientes objetivos:

- Desarrollar un simulador de dinámica molecular capaz de llevar a cabo la ejecución del sistema íntegramente en entornos Multi-GPU, de forma que las GPUs no se utilicen como meros coprocesadores, sino como el elemento de computo principal. De esta forma la CPU solo se encarga de gestionar los procesos de las GPUs así como la comunicaciones entre estas.

- Demostrar las ventajas de las técnicas de procesamiento masivamente paralelo aplicadas al campo de la simulación con respecto a otros algoritmos existentes en la actualidad, tanto a nivel de rendimiento de como de escalabilidad, utilizando para ello entornos de supercomputación reales, ya sea en centros de supercomputación o en servicios de cloud computing.

- Desarrollar una herramienta que permita automatizar todos los procesos implicados en la simulación, proveyendo de la capacidad de crear y gestionar de forma eficiente una infraestructura adecuada en un entorno de cloud computing.

- Simplificar la gestión y ejecución de todo del proceso de simulación, incluyendo todo el ciclo de preparación necesario. Por ejemplo, llevar a cabo un balanceo inicial para asegurar un estado físicamente correcto, de forma que la simulación se pueda llevar a cabo a partir de la información básica obtenida de repositorios públicos existentes o de un modelado manual del sistema molecular.

1.4. Organización del documento

Dada esta lista de objetivos definida en la sección anterior podemos dividir esta tesis en dos partes diferenciadas. La primera y principal es el desarrollo del algoritmo de simulación de dinámica molecular, en el que no solo se buscará una mejora de rendimiento con respecto a los algoritmos actuales, sino también una escalabilidad prácticamente lineal a medida que se aumente el número de GPUs implicadas. Además, se compararán tanto el rendimiento como la escalabilidad en entornos de supercomputación y de cloud computing con otros simuladores conocidos.

La segunda parte de la tesis consistirá en la definición del pipeline de datos y la automatización del proceso de simulación en un entorno cloud. Nótese que al obtener los datos los de sistemas moleculares desde repositorios públicos, solo se dispone de los datos atómicos más básicos y deben ser tratados para poder obtener los tipos de enlaces que contiene cada una de las moléculas, así como saber que átomos están implicados en estas. Además, habitualmente los sistemas moleculares se deben encontrar en un solvente, y el sistema completo debe ser estable en términos de energía. Todo este proceso se automatizará para su ejecución en un entorno de computación cloud, de manera que se reserven únicamente los recursos necesarios en función de los requisitos de cada uno de los pasos del pipeline. Asimismo, la herramienta permitirá configurar y realizar la ejecución final de la simulación en un entorno cloud distribuido, habilitando los recursos según los parámetros definidos

por el usuario

El documento se divide en varios capítulos, cada uno de los cuales se compone de diferentes secciones. El Capítulo 2, *Estado del arte* muestra las diferentes técnicas de simulación y paralelización necesarias para llevar a cabo la simulación molecular, además de los diferentes entornos y arquitecturas para su ejecución. Este capítulo se divide en las secciones:

- Sección 2.1, Dinámica molecular: Expone las técnicas existentes de simulación de dinámica molecular, discutiendo cuales se adaptan mejor a la simulación en entornos Multi-GPU.
- Sección 2.1.1, Fuerzas implicadas en dinámica molecular: Estudia las distintas fuerzas que afectan a los átomos de un sistema molecular complejo.
- Sección 2.1.2, Integradores o métodos de integración numérica: Muestra los distintos métodos de integración numérica que se adaptan a esta rama de simulación.
- Sección 2.2, Dinámica molecular en arquitecturas paralelas: Estudia las técnicas actuales de simulación de dinámica molecular para entornos en paralelo.
- Sección 2.3, Simuladores en arquitecturas HPC: Compara los distintos simuladores de dinámica molecular más relevantes que hacen uso de arquitecturas HPC.
- Sección 2.4, Computación cloud: Muestra las capacidades de los servicios de computación en la nube aplicados a la simulación de dinámica molecular.

El Capítulo 3 *Planteamiento del problema* detalla los diferentes problemas que se deben abordar para llevar a cabo la simulación de dinámica molecular. Se divide en las siguientes secciones:

- Sección 3.1, Dinámica molecular en arquitecturas masivamente paralelas: Muestra los diversos problemas existentes a la hora de de simular dinámica molecular en entornos paralelos.
- Sección 3.2, Arquitectura en entornos de cloud computing: Trata el funcionamiento de los entornos de cloud computing y los problemas que puede suponer utilizar dichos entornos para un usuario no especializado en TIC.
- Sección 3.3, Proceso de datos para dinámica molecular: Especifica la problemática que supone tratar con sistemas moleculares complejos, ya que los datos que se pueden obtener de estos no están preparados para realizar el proceso de simulación.

El Capítulo 4, *Simulación de dinámica molecular en entornos HPC*, muestra los detalles del diseño e implementación de un simulador de dinámica molecular en arquitecturas Multi-GPU. Está compuesto de las secciones:

- Sección 4.1, Dinámica molecular escalable en entornos Multi-GPU: Compila las distintas soluciones que se pueden aplicar a los problemas que supone la simulación de dinámica molecular en arquitecturas HPC.
- Sección 4.2, Algoritmo de simulación en paralelo: Muestra el algoritmo utilizado para aplicar las soluciones especificadas anteriormente.
- Sección 4.3, Procesado de datos moleculares: Recopila todos los procesos necesarios para simular un sistema molecular complejo a partir de los datos básicos de los que se parte.

El Capítulo 5, *Dinámica molecular en entornos de Cloud Computing*, especifica como se realiza el despliegue del simulador en entornos cloud y de alto rendimiento. Se compone de las siguientes secciones:

- Sección 5.1, Despliegue de arquitecturas en servicios Cloud: Describe la arquitectura de una herramienta web para gestionar el uso de servicios de cloud computing para llevar a cabo todo el proceso de simulación.
- Sección 5.2, Proceso de ejecución e interacción con el usuario: Detalla el flujo de ejecución de la herramienta web para la simulación de dinámica molecular.

El capítulo 6, *Evaluación*, muestra los resultados obtenidos con el simulador propuesto. Consta de las secciones:

- Sección 6.1, Rendimiento del algoritmo de dinámica molecular: Detalla las pruebas realizadas con el algoritmo de simulación, comparando su rendimiento con el de otros simuladores de dinámica molecular.
- Sección 6.2, Evaluación de dinámica molecular con servicios de Cloud Computing: Muestra un caso de uso de la herramienta web, realizando el proceso completo de procesado de datos de simulación.

Finalmente, el capítulo 7, *Conclusión y líneas futuras*, analiza los resultados obtenidos a lo largo de la tesis. Se divide en las secciones:

- Sección 7.1, Conclusiones: Compara y discute los resultados obtenidos con los objetivos fijados previamente presentando las conclusiones de la tesis.
- Sección 7.2, Trabajo futuro: Establece las líneas a seguir a partir del trabajo realizado.

Capítulo 2

Estado del arte

2.1. Dinámica molecular

La dinámica molecular se basa en el cálculo de la nueva posición de cada uno de los átomos que componen un sistema molecular de acuerdo con las fuerzas existentes entre cada uno de estos para cada paso de la simulación. Los átomos se definen en un espacio tridimensional que representa un sistema molecular ocupando un volumen definido. El conjunto de átomos de un sistema biomolecular normalmente se encuentra rodeado de moléculas de agua y debe cumplir ciertas condiciones periódicas en los límites.

Estas condiciones de frontera implican que, cuando un átomo sale del volumen de la simulación por uno de los lados, vuelve a entrar por el lado contrario, de forma que la simulación no se ve afectada por ninguna restricción.

Para llevar a cabo una simulación, se divide el tiempo total que se va a simular (del orden de segundos o milisegundos) en pasos de tiempo de varios ordenes de magnitud inferiores, por ejemplo, femtosegundos (fs, 10^{-15} segundos) en el caso de dinámica molecular. Para cada uno de estos pasos de tiempo se calculan todas las fuerzas que afectan a cada uno de los átomos, integrándolas para obtener las nuevas velocidades y posiciones de cada átomo

2.1.1. Fuerzas implicadas en dinámica molecular

Existen tres diferentes tipos de fuerzas internas que actúan sobre los átomos de un sistema molecular, las cuales se describen a continuación:

- Las fuerzas *de enlace* se producen debido a los enlaces de los átomos que conforman la molécula. Los enlaces representan la atracción entre dos átomos que componen una conexión química. La fuerza de estos enlaces se basa en la carga y en el número de electrones que los átomos pueden compartir entre si. Estas fuerzas se modelan como si se tratara de un muelle, y solo actúan en los átomos que componen el enlace. Son las más rápidas de calcular, ya que cada enlace está compuesto por un pequeño número de átomos, y son diferentes dependiendo del número de átomos enlazados (simples, angulares, propios e impropios). Las ecuaciones de energía de las fuerzas de enlace son las siguientes:

$$\begin{aligned} E_{simple} &= k(|r_{ij}| - r_0)^2 \\ E_{angular} &= k_\theta(\theta - \theta_0)^2 + k_{ub}(|r_{ik}| - r_{ub})^2 \\ E_{propio/impropio} &= \begin{cases} k(1 + \cos(n\Phi + \theta)) & \text{if} \quad n > 0 \\ k(\Phi - \theta)^2 & \text{if} \quad n = 0 \end{cases} \end{aligned} \tag{2.1}$$

  Los enlaces *simples* (E_{simple}) se describen como muelles armónicos simples, donde k representan la constante del muelle que enlaza ambos átomos, $|r_{ij}|$ es la distancia entre los átomos i y j, y r_0 es la misma distancia en estado de reposo. Los enlaces *angulares* ($E_{angular}$) se definen como muelles armónicos angulares, donde k_θ en la constante del muelle armónico en ángulo que enlaza los tres átomos, θ es el ángulo formado por los vectores que conectan los átomos $i-j$ y $k-j$, y θ_0 es el ángulo formado por los vectores que conectan los átomos

$i - j$ y $k - j$ en reposo. Finalmente, los enlaces dihédricos e impropios están formados por la unión de cuatro átomos, y se modelan como la unión de un primer conjunto de tres átomos (i, j y k), y un segundo conjunto de otros tres átomos (j, k y l).

- Las fuerzas *electrostáticas* se producen debido a las cargas electrostáticas de los átomos. Se calculan utilizando todas las interacciones entre los átomos del sistema. No es un cálculo directo como las otras fuerzas, ya que para todo el sistema supondría una complejidad $O(n^2)$. Para calcular estas fuerzas se utilizan algoritmos que dividen el sistema y calculan campos de potenciales para, finalmente, obtener la fuerza total que ejerce el sistema sobre cada átomo [47, 29]. La ecuación de energía de las fuerzas electrostáticas es la que sigue:

$$E = \varepsilon_{14} \frac{C q_i q_j}{\varepsilon_0 |r_{i,j}|}$$

donde ε_{14} es el factor de escala para las interacciones 1-4 (pares de átomos conectados por tres enlaces). Este valor es cero para las interacciones 1-2 y 1-3 (pares de átomos conectados por enlaces simples y dobles, respectivamente), y es igual a uno para cualquier otra interacción. C es la constante de Coulomb y es igual a $2{,}31 \times 10^{-19}$ J nm. q_i y q_j son las cargas de los átomos i y j respectivamente. ε_0 es la constante dieléctrica. Finalmente, $|\vec{r}_{i,j}|$ es la distancia entre los átomos i y j.

El cálculo de estas fuerzas se puede dividir en dos tipos que se deben calcular por separado: de corto alcance y de largo alcance.

- Las fuerzas de *Van der Waals* son las resultantes de las interacciones entre átomos. Estas interacciones suceden entre átomos que se encuentra separados, y decaen rápidamente a medida que la distancia aumenta, por lo que es posible

establecer un radio de corte R_c desde el cual la fuerza ejercida se considera despreciable. Las ecuaciones de energía de las fuerzas de Van der Waals son:

$$E_{vdw} = \frac{A}{r_{ij}^{12}} - \frac{B}{r_{ij}^{6}} \tag{2.2}$$

siendo A y B constantes precalculadas utilizando los parámetros σ_{ij} y ε_{ij}, los cuales también se precalculan utilizando los valores σ y ε de los átomos:

$$\sigma_{ij} = \frac{\sigma_i + \sigma_j}{2}$$

$$\varepsilon_{ij} = \sqrt{\varepsilon_i \varepsilon_j}$$

$$A = 4\sigma_{ij}^{12}\varepsilon_{ij}$$

$$B = 4\sigma_{ij}\varepsilon_{ij}$$

donde σ_{ij} es la distancia (finita) entre las partículas i y j donde su potencial es cero, y ε_{ij} es la profundidad del potencial entre las partículas i y j [14].

Al igual que con las fuerzas electrostáticas, podemos dividir las fuerzas de Van der Waals en corto y largo alcance. Sin embargo, como se ha indicado anteriormente, a partir de cierta distancia estas fuerzas se consideran despreciables, por lo que se pueden tratar solo como fuerzas de corto alcance.

2.1.2. Integradores o métodos de integración numérica

El desarrollo de una simulación implica la resolución de un problema dinámico, lo que a su vez supone la resolución de una ecuación diferencial. En este tipo de problemas, el resultado varía en cada instante de tiempo, y a su vez, depende del estado de la simulación en el instante anterior. Por ello, para resolver el problema se parte del estado inicial del sistema siendo $t = 0$ y, a través de las fuerzas implicadas, se calcula el nuevo estado con $t = \Delta t$. Este valor Δt se conoce como *paso de tiempo*

o *time-step* y, dependiendo de la aplicación y de la precisión que se desea obtener, puede ser desde un orden de *fs* o hasta de una magnitud de *ms* o *s*. Podemos definir integrador, o método de integración, como un método numérico para la obtención de las trayectorias de las fuerzas de la simulación, el cual implica la resolución de una ecuación diferencial. Por extensión permite obtener las aceleraciones del sistema y, a partir de ahí, calcular las posiciones de los elementos en cada instante de tiempo.

Según la definición anterior, un método de integración debe servir para resolver Ecuaciones Diferenciales Ordinarias (EDO). Una EDO general, con derivadas con respecto a una variable t se define tal que:

$$f(t, x(t), x'(t), x''(t)...x^{(n)}(t)) = 0$$

de esta forma, se puede definir una EDO lineal como:

$$a_0 x^{(n)}(t)... + a_{n-1}x'(t) + a_n x(t) + t = 0$$

2.1.2.1. Método de Euler explicito

Dada una EDO lineal, se dice que es una EDO de primer orden cuando:

$$x'(t) = f(x(t), t)$$

En el proceso de simulación partimos de que se conoce el estado inicial $x(t_0) = x_0$. A partir del desarrollo en serie de Taylor:

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + ... + \frac{f^{(k)}(a)}{k!}(x-a)^k$$

aproximando la función en $t_1 = t_0 + \Delta t$ y truncando la serie de Taylor se obtiene:

$$x(t_0 + \Delta t) = x(t_0) + x'(t_0)\Delta t + O(\Delta t^2)$$

Lo que permite integrar EDOs de primer orden, siendo $O(\Delta t^2)$ el error producido por el truncamiento de la serie de Taylor, el cual se considera despreciable. Este método de integración se conoce como método de *Euler explícito*, y permite resolver EDOs, y por lo tanto integrar las velocidades de una simulación ($x'(t)$), mediante el siguiente algoritmo:

1. Cálculo de la derivada:
$$x'_0 = x'(t_0) = f(x_0, t_0)$$
2. Integración:
$$x_1 = x'(t_1) = x_0 + x'_0 \Delta t$$
3. Cálculo de la derivada:
$$x'_1 = x'(t_1) = f(x_1, t_1)$$
4. Integración:
$$x_2 = x'(t_2) = x_1 + x'_1 \Delta t$$
5. Etc.

En el caso de la simulación de cualquier fenómeno dinámico se aplica la segunda ley de Newton, la cual se puede expresar como una EDO de segundo orden:

$$m \cdot x''(t) = F(x(t), x'(t), t)$$

Sin embargo, es posible descomponerla en dos EDOs de primer orden:

$$m \cdot v'(t) = F(x(t), v(t), t)$$

$$x'(t) = v(t)$$

Lo que posibilita la integración con el método de Euler explicito, aprovechando que es posible calcular la velocidad como una EDO de primer orden:

$$v(t+\Delta t) = v(t) + \Delta t \cdot v'(t) = v(t) + \Delta t \cdot \frac{1}{m} F(x(t), v(t), t)$$

$$x(t+\Delta t) = x(t) + \Delta t \cdot x'(t) = x(t) + \Delta t \cdot v(t)$$

De esta forma se puede llevar a cabo la integración alternando entre posición y velocidad. Este método es conocido como método de *Euler semi-implícito* o *Euler Simpléctico*.

2.1.2.2. Método de Verlet

Dada una EDO de segundo orden, es posible realizar el desarrollo en serie de Taylor hasta la segunda derivada. El método de integración de Verlet utiliza dos veces el desarrollo de Taylor para integrar la posición:

$$x(t+\Delta t = x(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2 + \frac{1}{6}x'''(t)\Delta t^3 + O(\Delta t^4)$$

$$x(t-\Delta t = x(t) - v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2 - \frac{1}{6}x'''(t)\Delta t^3 + O(\Delta t^4)$$

Al sumar ambas expresiones, por cancelación de términos se obtiene la integración de la posición:

$$x(t+\Delta t) = 2x(t) - x(t-\Delta t) + a(t)\Delta t^2 + O(\Delta t^4)$$

y para obtener la velocidad:

$$v(t+\Delta t) = \frac{x(t+\Delta t) - x(t-\Delta t)}{2\Delta t} + O(\Delta t^2)$$

2.1.2.3. Método RESPA

RESPA (*REference System Propagator Algorithm*) es un método de integración diseñado para utilizar múltiples pasos de tiempo (MTS o *Multi time-step*). Este método se utiliza en dinámica molecular para evitar los cálculos de las fuerzas de mayor complejidad en cada paso de tiempo. La utilización de los métodos de integración vistos previamente implica realizar el cálculo de todas las fuerzas, tanto de enlace como las electrostáticas de corto y largo alcance, en cada paso. RESPA establece una relación entre el número de veces que se calculan estas fuerzas las fuerzas de enlace, las de Van der Waals y las electrostáticas de corto y largo alcance.

Las fuerzas electrostáticas y de Van der Waals necesitan un mayor tiempo de cálculo. Además, permiten utilizar un paso de tiempo superior al utilizado en las fuerzas de enlace sin que disminuya la precisión. Por otro lado, las fuerzas electrostáticas de largo alcance permiten un paso de tiempo aun más alto con respecto a las de Van der Waals. El Algoritmo 1 muestra en pseudocódigo el funcionamiento de este método para un sistema con tres diferentes tipos de fuerzas, las cuales suponen distintos tiempos de cálculo. Podemos definir las fuerzas de enlace como *fuerzas rápidas*, las fuerzas de Van der Walls y las electrostáticas de corto alcance como *fuerzas medias* y las electrostáticas de largo alcance como *fuerzas lentas*. Para cada tipo de fuerza se utiliza un paso de tiempo diferente, siendo estos Δt_r, Δt_m y Δt_l respectivamente. De esta forma, f_l solo se calcula L veces, f_m se calcula $L \cdot M$ y f_r $L \cdot M \cdot R$ veces. Para la integración de la velocidad y la posición se utiliza el método *Verlet* descrito en la sección anterior.

2.2. Dinámica molecular en arquitecturas paralelas

Para llevar a cabo el proceso de simulación de dinámica molecular, los átomos se ubican en un sistema virtual de coordenadas tridimensionales, representando el

Algoritmo 1 Algoritmo para la resolución del método RESPA

```
1: procedure RESPA(Δt_lf)
2:     for i = 1 to L do
3:         v = v + (f_l Δt_l)/(2m)
4:         for j = 1 to M do
5:             v = v + (f_m Δt_m)/(2m))
6:             for k = 1 to R do
7:                 v = v + (f_r Δt_r)/(2m)
8:                 r = r + vΔt_m
9:                 f_r = CalcularFuerzasRapidas()
10:                v = v + (f_r Δt_r)/(2m)
11:            end for
12:            f_m = CalcularFuerzasMedias()
13:            v = v + (f_m Δt_m)/(2m)
14:        end for
15:        f_l = CalcularFuerzasLentas()
16:        v = v + (f_l Δt_l)/(2m)
17:    end for
18: end procedure
```

sistema molecular según sus propiedades. Existen diversos puntos en el proceso de simulación donde se pueden aplicar optimizaciones que permitan paralelizar el problema y mejorar el rendimiento en arquitecturas HPC, tanto Multi-GPU y Multi-CPU, desde el proceso de particionado de los datos hasta la evaluación de las distintas fuerzas que actúan sobre cada átomo.

2.2.1. Particionado

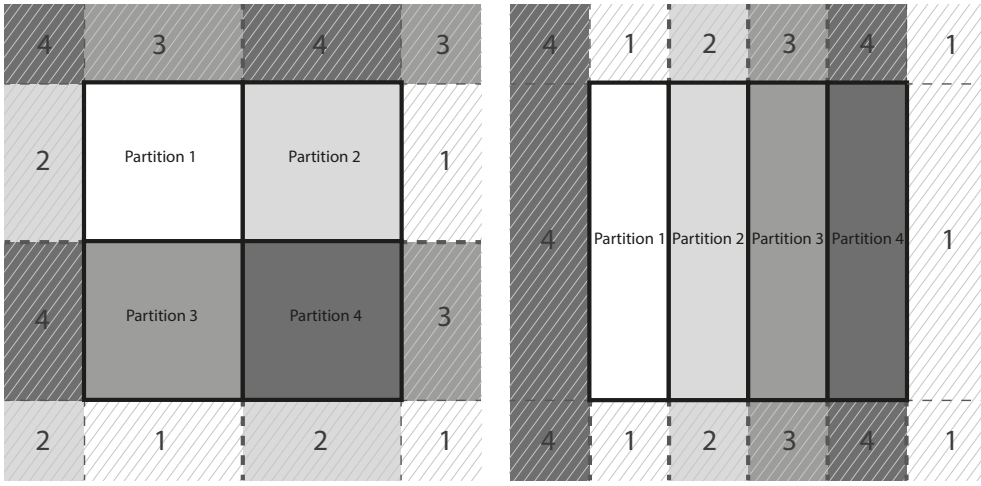
El particionado del sistema es uno de los principales puntos que se pueden optimizar para facilitar la paralelización de los algoritmos de simulación. En función de la arquitectura paralela donde se vaya a ejecutar, se pueden seguir diferentes topologías [42], las cuales afectan al tratamiento de las condiciones periódicas en los límites. Podemos diferenciar entre dos topologías de particionamiento, las cuales se muestran en la figura 2.1:

- Particionamiento lineal (véase figura 2.1(b)): consiste en realizar la división del sistema en torno a un solo eje cartesiano. Esto implica que cada partición solo linda con otras dos particiones, colocadas de forma secuencial. En el caso de condiciones de frontera periódicas cada partición también linda consigo misma, formando una secuencia de particionado cíclica de forma que, existiendo n particiones, la partición $n+1$ pasa a ser la partición 0.

- Particionamiento binario (véase figura 2.1(a)): divide el sistema en torno a todos los ejes de coordenadas utilizados. En el caso de la imagen (bidimensional), se realiza el particionado en torno a dos ejes.

2.2.2. Fuerzas de enlace

Las fuerzas de enlace se deben a la interacción entre un conjunto de átomos unidos entre sí por algún tipo de enlace, los cuales se describen en la sección 2.1.1. Para cada enlace se evalúa la energía que afecta a los átomos en función de las fuerzas descritas. El cálculo de energía para cada átomo se realiza de forma independiente a los demás, y el coste computacional que supone es similar con los enlaces del mismo tipo. Esto hace que la forma más común de paralelizar este cálculo sea subdividir la tarea. Esta



(a) Partición binaria (b) Partición lineal

Figura 2.1: *Comparación de particionado espacial binario **(a)** y lineal **(b)**. Las zonas rayadas representan la periodicidad del volumen de simulación. El sistema de partición binario es el mas indicado para entornos distribuidos, como demuestra [42].*

división permite asignar cada subtarea a los elementos computacionales disponibles y, por lo tanto, paralelizar el proceso en entornos Multi-CPU y Multi-GPU.

2.2.3. Fuerzas electrostáticas de corto alcance y Van der Walls

Las fuerzas electrostáticas y de Van der Walls decaen rápidamente a medida que aumenta la distancia entre átomos. Por eso es posible calcular con exactitud las interacciones entre los átomos dentro de un radio de cutoff (R_c), en las que las fuerzas no han decaído. Las fuerzas dentro de este radio (R_c) son las que se van a tratar en este apartado.

El cálculo de estas fuerzas se puede acelerar dividiendo el sistema en una cuadrícula tridimensional, la cual se actualiza con frecuencia durante todo el proceso de

simulación. Este método se conoce como *cell list* [17]. El algoritmo divide el espacio de simulación en subespacios tridimensionales ("celdas"), siendo el número de celdas dependiente de R_c. Las fuerzas se calculan entre pares de átomos que se encuentran en la misma celda o en alguna de las celdas contiguas. Debido a esto, se puede paralelizar mediante métodos de subdivisión. Puesto que las interacciones de las celdas se pueden calcular de manera individual, es posible calcular en interacciones entre átomos de distintas celdas. Esto permite aprovechar las arquitecturas HPC, como GPUs o clústers, siempre que se utilicen las técnicas de particionamiento adecuadas.

Durante el proceso de simulación los átomos varían su posición, lo que provoca que estos puedan moverse entre las distintas celdas y obliga a la actualización de su contenido a medida que la simulación avanza. Esto supone que se debe emplear parte del tiempo total de la simulación en actualizar los datos de las celdas, lo que afecta al rendimiento. Por esta razón el objetivo es reducir todo lo posible la información que se va a actualizar y prevenir así los posibles cuellos de botella.

2.2.4. Fuerzas electrostáticas de largo alcance

Cada átomo genera una fuerza que afecta al resto de átomos del sistema, lo que significa que para calcular la fuerza total que actúa sobre un solo átomo es necesario iterar sobre cada uno de los demás átomos del conjunto. Esto hace que el cálculo de fuerzas electrostáticas tenga un coste computacional cuadrático ($O(n^2)$). Las fuerzas entre los átomos que están a una distancia superior a R_c se calculan utilizando aproximaciones, con el objetivo de acelerar la simulación. Como las fuerzas decaen rápidamente al superar la distancia R_c, el error producido por esta aproximación es despreciable. Esta distancia R_c es la misma definida en la sección 2.2.3 a partir de la cual las fuerzas de Van der Walls se consideran despreciables y, por lo tanto, en la mayoría de los casos no es necesario calcularlas.

Existen múltiples formas de reducir el coste cuadrático del cálculo de estas fuer-

zas, siempre pasando por aproximar la solución o realizar implementaciones en paralelo [48]. También existen soluciones masivamente paralelas en GPU, pero siempre utilizando la GPU como coprocesador [49].

PME (*Particle Mesh Ewald*) [16] es uno de los métodos mas utilizados para calcular las fuerzas electrostáticas de largo alcance. Los métodos *Lattice Ewald* resuelven los potenciales de largo alcance dividiendo el sistema en una cuadrícula y utilizando una FFT (Transformada Rápida de Fourier). El algoritmo PME estándar utiliza diferenciación espectral y un total de cuatro FFTs por paso de tiempo, mientras que *Smooth PME* (SPME) [21] utiliza interpolación de B-Splines, reduciendo el número de FFTs necesarias a dos. PME es un algoritmo que puede ser paralelizado en una sola GPU, pero no es tan fácil de distribuir en múltiples GPUs, debido a la comunicación que se debe realizar entre todos los átomos para calcular la FFT. Aún así, PME se utiliza en múltiples simuladores de dinámica molecular en paralelo como NAMD [49], GROMACS [31] o ACEMD [30].

Otros métodos de aproximación se basan en algoritmos con cuadrículas. Estas aproximaciones utilizan múltiples niveles de cuadriculado con distinta resolución para calcular los potenciales de largo alcance, con un coste computacional $O(n)$. En dinámica molecular, este tipo de métodos han demostrado ser más eficientes que los vistos previamente [55], así como el *Fast Multipole Method* (FMM) [63], debido a que su escalabilidad es superior manteniendo un error bajo. El *Meshed Continium Method* (MCM) [11] y *Multilevel Summation Method* (MSM) [27] son los métodos de cuadriculado más comunes para el cálculo de las fuerzas de largo alcance. MCM utiliza funciones de densidad para muestrear las partículas y distribuirlas en una cuadrícula, calculando el potencial mediante una ecuación de Poisson. Por otro lado, MSM calcula el potencial directamente en una cuadrícula, utilizando diversas longitudes de escala. Estas escalas se distribuyen sobre una jerarquía de cuadrículas, donde el potencial de los niveles de menor densidad se corrige sucesivamente me-

diante contribuciones de los niveles con mayor densidad, lo que permite obtener el potencial final. Esta aproximación ofrece una mayor escalabilidad que PME o que otros algoritmos de cuadriculado. Aunque en implementaciones con una sola GPU el rendimiento de MSM es notablemente inferior que el de PME [28], en implementaciones Multi-GPU MSM ofrece un rendimiento superior [41]. El desarrollo de este algoritmo para entornos Multi-GPU queda fuera del alcance de esta tesis, ya que en trabajos previos se ha alcanzado la escalabilidad requerida para fuerzas de largo alcance.

2.2.4.1. *Multilevel Summation Method* (MSM)

Para un sistema de partículas con cargas $q_1, ...q_N$ y posiciones $r_1, ...r_N$, la energía potencial electrostática es:

$$U(r_1, ...r_N) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1, j!=i}^{N} \frac{q_i q_j}{||r_i - r_j||}$$

y tiene una complejidad computacional de $O(n^2)$

MSM es un algoritmo rápido para el cálculo de una aproximación de las interacciones electrostáticas con una complejidad de $O(n^2)$. MSM divide el potencial en componentes de corto y largo alcance. El componente de corto alcance se calcula como una interacción directa entre partículas, mientras que el de largo alcance se aproxima mediante una jerarquía de cuadrículas.

Para el termino de largo alcance, primero se distribuyen las cargas de los átomos en la cuadrícula más fina de la jerarquía. Este proceso se denomina *anterpolación*. Para distribuir estas cargas se utiliza una función nodal $\phi(r)$ sobre cada punto de la cuadrícula. Una vez que las cargas se han distribuido en la cuadrícula mas fina, se distribuyen de nuevo en el siguiente nivel utilizando la misma función. A este proceso se le denomina *restricción*, y se repite hasta que se alcanza la cuadrícula más gruesa.

La Figura 2.2 describe el algoritmo MSM completo. A cada nivel, el método calcula sumas directas con las cargas de las cuadrículas vecinas hasta un radio de $2R_c/h_0$, donde h_0 es la resolución de la cuadrícula más fina. Para simulación de dinámica molecular es suficiente una resolución entre 1Å[1] y 3Å (véase [27]). Cabe destacar que la resolución se reduce a la mitad a medida que la cuadrícula se hace mas gruesa, por lo que las sumas directas cubren el doble de distancia con el mismo número de puntos. La suma directa de los potenciales de carga por pares es análoga a la de las fuerzas de corto alcance, con la excepción de que las distancias de la cuadrícula son fijas y se pueden calcular como en un preprocesado, por lo que el cálculo es simplemente una acumulación de cargas ponderadas de la cuadrícula.

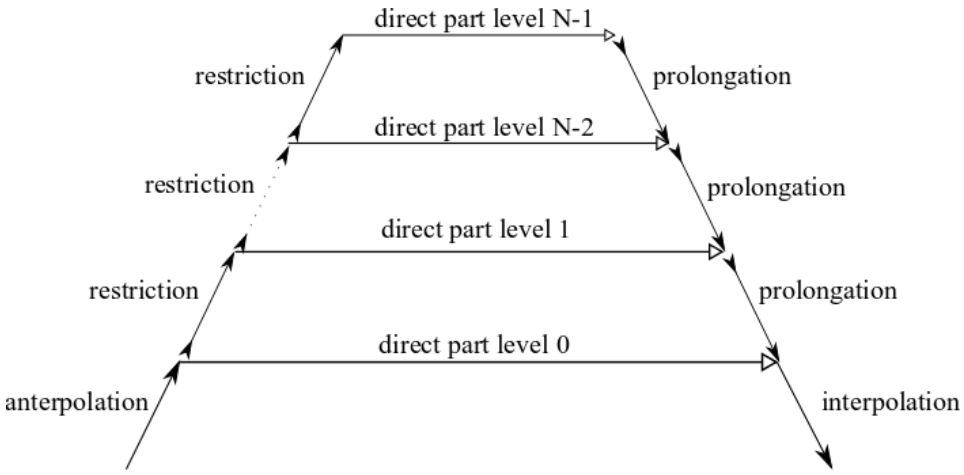


Figura 2.2: *Diagrama con las principales operaciones de MSM. El nivel mas bajo representa los átomos, y el más alto representa la cuadrícula más gruesa*

Existe una versión optimizada para GPU de la suma directa (véase [28]), donde la cuadrícula ponderada se almacena en memoria de constantes y se va cargando en memoria compartida. El algoritmo calcula los niveles más finos en la GPU, mientras

[1] Ángstrom: Unidad de longitud que se emplea generalmente para expresar distancias moleculares y atómicas. Equivale a 10^{-10} metros

que los más gruesos se calculan en CPU.

Una vez que las sumas directas se han calculado en cada nivel, se interpolan los potenciales desde el nivel mas grueso al más fino, acumulando la contribución de cada uno de ellos. Este proceso se llama *prolongación*. Finalmente se interpolan los potenciales de la cuadrícula más fina con los átomos.

2.3. Simuladores en arquitecturas HPC

Existen múltiples algoritmos para la simulación de dinámica molecular, como [34] o [61], y la mayor parte de los simuladores de dinámica molecular [12, 43, 45, 8, 30] se han desarrollado utilizando estos algoritmos. También existen simuladores de dinámica molecular interactivos, cuyo objetivo es acelerar la simulación imponiendo restricciones externas que inclinan la simulación hacia fenómenos de interés científico [56]. Esto es debido a que las técnicas de simulación actuales pueden no llegar a obtener algunos procesos biológicos dada la gran cantidad de tiempo necesario para que ocurran los cambios más grandes. Estos casos se pueden dar cuando se estudia la dinámica de sistemas biológicos complejos [18], que consisten en grandes sistemas moleculares con millones de partículas, como pueden ser las cápsidas de los virus, o incluso un virus completo. Para calcular el gran número de interacciones que tienen estos sistemas, no solo se necesita un periodo de simulación prolongado, sino también una alta cantidad de recursos computacionales. Su ejecución en un entorno masivamente paralelo compuesto por cientos o miles de GPUs puede mejorar en gran medida el rendimiento de la solución.

Actualmente los grandes centros de supercomputación disponen de entornos Multi-GPU. Sin embargo, hasta hace unos años los entornos de supercomputación consistían en un gran conjunto de nodos, equipados con una o múltiples CPUs, interconectados entre ellos. En ese momento los algoritmos de paralelización propues-

tos [46, 20, 58] trataban de sacar provecho de estas arquitecturas, como las primeras versiones del simulador NAMD [45]. Actualmente los algoritmos y simuladores se han adaptado para aprovechar toda la capacidad que ofrecen las GPUs modernas, y mejorar así el rendimiento en el calculo de las interacciones atómicas.

Las versiones más modernas de NAMD [49] tratan la molécula como un conjunto de parches tridimensional, donde el número de estos viene determinado por el tamaño del sistema molecular a simular, independientemente del número de GPUs. Los parches cercanos entre sí se mantienen en el mismo nodo, lo que minimiza las comunicaciones necesarias durante el proceso de simulación. Cada átomo se almacena en el mismo parche que los átomos con los que está enlazado, y cuando los átomos se mueven estos son reasignados a los parches a intervalos regulares, teniendo siempre en cuenta la estrategia de balanceo de carga que el simulador implementa. Por otro lado, GROMACS [8] realiza un particionado espacial del sistema para distribuirlo en arquitecturas con múltiples procesadores. En ambos se utiliza la GPU como coprocesador, y no se realizan todos los cálculos en esta, lo que limita el rendimiento de la simulación al hacer uso de la CPU. Otros simuladores como ACEMD [30] realizan los cálculos de las fuerzas utilizando todo el potencial de la GPU, ejecutando cada uno de los tres tipos de fuerzas en GPUs diferentes. Esto tiene el inconveniente de que el número máximo de GPUs se ve muy limitado.

Otros algoritmos diseñados para sistemas de partículas utilizan técnicas de particionamiento similares para simular sistemas de partículas masivos [50], añadiendo los mecanismos de comunicación necesarios para resolver el problema. Por ejemplo, la simulación de fluidos basada en partículas comparte muchas características con la dinámica molecular. Ambas disciplinas tratan con partículas que interactúan con el resto a nivel global. De esta manera, las simulaciones de fluidos basadas en partículas se pueden resolver mediante el algoritmo Position Based Dynamics [37]. Este algoritmo es fácilmente paralelizable en entornos Multi-GPU [23, 22], mientras las

partículas implicadas en la simulación no estén directamente relacionadas (enlaces) y se muevan libremente entre las diferentes GPUs. Sin embargo, en dinámica molecular, las partículas (átomos) se encuentran unidas físicamente mediante enlaces y su movimiento no es completamente libre, ya que se encuentran directamente ligados al movimiento de los átomos a los que están enlazados. El particionado entonces requiere que cada GPU almacene los átomos que le corresponde simular, además de que pueda identificar en otras GPUs los átomos con los que estos se encuentran enlazados.

2.4. Computación cloud

Llevar a cabo una simulación de sistemas moleculares masivos en arquitecturas Multi-GPU implica la utilización de un entorno de computación de altas prestaciones. Este tipo de entornos suelen encontrarse en grandes centros de supercomputación, donde disponen de cientos de nodos con múltiples CPUs y una o varias GPUs (en clúster). Estos nodos se comunican entre sí a través de un entorno de red de alta velocidad, lo que permite que compartan datos y sean utilizados simultáneamente. Aunque estos clústeres encajan perfectamente con el problema de dinámica molecular, el acceso a estos no es sencillo. Acceder a un supercomputador suele implicar tramites burocráticos, siendo su uso no exclusivo ya que se comparte con otros usuarios. Esto deriva en la necesidad de encolar las tareas que se desean ejecutar, y se depende de un sistema de prioridad que se encarga de decidir en que momento se va a ejecutar el trabajo. Por lo tanto, la ejecución de nuestra tarea se puede retrasar desde unos pocos minutos, hasta varias horas o incluso días dependiendo de nuestro nivel de prioridad y de la cantidad de recursos solicitados.

Por otro lado, las arquitecturas Multi-GPU también están disponibles en entornos de cloud computing. Empresas como Amazon, Google o Microsoft ponen al servicio

de sus usuarios grandes entornos de computación de propósito general, a los cuales se puede acceder por un coste según el uso [60]. La principal ventaja de estos sistemas es que están disponibles en todo momento, sin depender de un nivel de prioridad salvo en casos donde se requieran cantidades extremas de recursos. Sin embargo, contratar servicios cloud conlleva una serie de inconvenientes que generalmente están resueltos en los centros de supercomputación. La mayor y principal diferencia es la infraestructura de la que disponen. Los centros de supercomputación constan de una serie de nodos interconectados entre si, con todo el software instalado y configurado para ser utilizado directamente por los usuarios. En cambio, los entornos de cloud computing en un enfoque de *Infrastructure as a Service (IaaS)* habilitan una serie de maquinas virtuales con los requisitos de hardware y el sistema operativo solicitados. Estas infraestructuras proporcionan un mecanismo de acceso al sistema operativo de la arquitectura elegida, pudiendo gestionar todo el software que se va a utilizar en este. Sin embargo, estos servicios no están enfocados como *Software as a Service (SaaS)* y no proveen de la usabilidad necesaria para alguien no especializado en infraestructura. Los usuarios que quieran utilizarlo tienen que gestionarlo ellos mismos al no poder disponer de manera inmediata del software de comunicación entre las maquinas ni del que se necesita para llevar a cabo la tarea deseada. Esto significa que el usuario debe actuar como ”administrador de sistemas”, instalando y configurando el software necesario. Además de esto, el software para la simulación de dinámica molecular necesita un entorno y una configuración específicos para sacar provecho a todo el hardware. Por otro lado, realizar esta configuración implica cierto grado de experiencia utilizando el entorno cloud, lo que se traduce directamente en coste económico. A esto hay que añadir que, normalmente, los usuarios que requieren realizar la simulación de un sistema molecular no son expertos en administración de sistemas informáticos, pues estas simulaciones se llevan a cabo en el campo de la biología o la química. Existen algunas herramientas que permiten diseñar y automatizar algu-

nos de estos procesos. Por ejemplo, [13] permite simular una infraestructura para un entorno de cloud computing con el objetivo de predecir el coste de utilizar software especifico, pero esa infraestructura debe desplegarla el usuario después en el entorno cloud. También existen otros softwares como Aneka [59] que permite automatizar el despliegue de una infraestructura utilizando servicios de cloud computing. Esta herramienta en particular provee al usuario de una API para el lenguaje .NET, que permite definir la infraestructura y el software a utilizar. Ambos casos requieren de un conocimiento especializado de los servicios cloud por parte del usuario, no eliminan el coste de la configuración de la arquitectura, ni encajan con los requisitos de la simulación de dinámica molecular.

Capítulo 3

Planteamiento del problema

Paralelizar algoritmos de dinámica molecular para simular sistemas moleculares masivos en arquitecturas Multi-GPU supone la necesidad de abordar ciertos retos. Por un lado, es necesario dividir el sistema de la forma más equitativa posible para asignar a cada GPU una subtarea, por ejemplo, un subsistema molecular, tratando de que cada una de ellas tenga una carga de trabajo similar. Por otro lado, esta división del sistema no implica un particionado estricto de los datos ya que, al estar todos los átomos relacionados entre si, se hace necesario que cada GPU almacene no solo los datos relativos a la subtarea que tiene asignada, sino también los datos contenidos en otras GPUs, correspondientes a los átomos circundantes al subsistema que le corresponde. Por último, para poder llevar a cabo el proceso de simulación es necesario disponer de una arquitectura Multi-GPU donde realizar la ejecución, ya sea en entornos de supercomputación o de cloud computing, además de realizar un tratamiento previo de los datos del sistema molecular antes de su ejecución.

3.1. Dinámica molecular en arquitecturas masivamente paralelas

Como se ha indicado al inicio de este capítulo, el primero de los desafíos que supone paralelizar la simulación de dinámica molecular es la necesidad de distribuir el sistema, en este caso, entre las distintas GPUs. Para ello se particiona el sistema

de forma equitativa para asignar a cada GPU cada uno de estos subsistemas. Nótese que estas particiones no son estáticas y los átomos se pueden mover entre las GPUs durante la simulación. Para poder llegar a obtener un sistema totalmente escalable, cada GPU debería mantener en memoria local solamente los datos que necesita para calcular su partición. Esto es necesario para reducir al mínimo el uso de memoria y poder simular sistemas moleculares masivos.

El segundo reto a considerar es que, aunque la simulación de dinámica molecular se puede considerar como un sistema de partículas, esta implica ciertos problemas en el momento de la paralelización que no se encuentran en un sistema de partículas común. Específicamente, el mayor problema es que los átomos se encuentran enlazados físicamente. Un enlace supone una conexión entre dos o más átomos, y el movimiento de uno de estos se traduce en una fuerza que actúa sobre los átomos que lo componen. Por lo tanto, no es posible que cada GPU simule independientemente las fuerzas de enlace para los átomos que le corresponden, ya que necesita los datos almacenados en otra GPU. Esto supone que, cuando los átomos se mueven entre las GPUs, es necesario compartir datos no solo de los átomos, sino también de los enlaces.

Este punto se analiza en el Capítulo 4, tratando el desarrollo un simulador de dinámica molecular capaz de llevar a cabo la ejecución del sistema íntegramente en entornos Multi-GPU y mostrando las ventajas de las técnicas de procesamiento masivamente paralelo aplicadas al campo de la simulación.

3.2. Arquitectura en entornos de cloud computing

La utilización de arquitecturas Multi-GPU supone un desafío, no solo a la hora de desarrollar los algoritmos paralelos que se van a ejecutar, sino también la disponibilidad del entorno. Como se indica en la sección 2.4, este tipo de arquitecturas se

pueden encontrar en entornos de supercomputación o de cloud computing. Un entorno de cloud computing es el que provee de mayor elasticidad. Sin embargo, puede llegar a suponer un gran reto para el usuario debido a la gran carga que implica su utilización y la configuración de este.

Para poder disponer de un entorno cloud es necesario seleccionar y contratar la maquina virtual o instancia que mejor se adapta al proceso. En este caso, la maquina virtual debe disponer de una o múltiples GPUs para realizar cálculos de propósito general. Sin embargo, en una arquitectura Multi-GPU no es posible disponer de más de 16 GPUs en un solo equipo, por lo que se requiere la utilización de múltiples instancias para poder disponer de una cantidad significativamente alta de GPUs. Utilizar múltiples instancias implica una configuración que permita la comunicación entre ellas y así poder distribuir correctamente los datos de la ejecución. Este es un proceso bastante tedioso para usuarios que no son expertos en administración de sistemas.

Las arquitecturas cloud se analizan en profundidad en el Capítulo 5, donde se trata el desarrollo una herramienta que permite automatizar todos los procesos implicados en la simulación, proveyendo de la capacidad de crear y gestionar de forma eficiente una infraestructura adecuada en un entorno de cloud computing.

3.3. Proceso de datos para Dinámica molecular

La simulación de dinámica molecular no solo implica el desarrollo de las técnicas de simulación, sino también la preparación de los datos del sistema molecular que se va a simular. Muchas veces la obtención y el modelado de los datos de un sistema molecular, como puede ser de una proteína, no se realiza solo con el objetivo de llevar a cabo un proceso de simulación, sino con un propósito de realizar un estudio general. Esto supone que los datos pueden no encontrarse preparados para la simulación,

contando solo con datos del propio sistema molecular, fuera de cualquier solvente (lo que no es posible en un entorno realista), y con la posición y forma en la que se encontraba la estructura real cuando se realizó la captura de los datos. Estos datos moleculares se pueden obtener de diversos repositorios abiertos, los cuales solo suelen proporcionar los datos referentes a las posiciones de los átomos y la molécula a la que pertenecen. Por tanto, para poder llevar a cabo una simulación del sistema molecular y estudiar su comportamiento, es necesario modificar estos datos para llevarlos a un estado en el que si podría encontrarse en un entorno real. Por ejemplo, el sistema debe estar ubicado en una solución de agua, donde sí puede existir de forma natural. En resumen, para poder llegar a realizar una simulación antes se debe llevar a cabo el siguiente procesado de datos:

1. Obtener un modelo físico, incluyendo los detalles sobre las interacciones atómicas como los enlaces o cargas electrostáticas.

2. Añadir un solvente (agua) al sistema molecular, dado que con los datos iniciales el sistema estaría ubicado en el vacío.

3. Llevar el sistema molecular a un estado de baja energía, puesto que la inclusión del solvente eleva drásticamente la energía de este a un nivel que no podría existir en la realidad.

Este es un proceso complejo que implica el uso de software especifico de dinámica molecular, y supone un largo periodo de aprendizaje por parte del usuario. Este proceso, al igual que la simulación, se puede ejecutar en entornos Multi-GPU de cloud computing. Este tratamiento de datos se define en profundidad en la Sección 4.3 del Capítulo 4, mostrando un pipeline que permite obtener los datos de los sistemas moleculares en un estado físicamente correcto para que pueda llevarse a cabo la simulación.

Capítulo 4

Simulación de Dinámica Molecular en entornos HPC

4.1. Dinámica molecular escalable en entornos Multi-GPU

La capacidad de paralelizar algoritmos computacionales ha supuesto una gran evolución en el campo de la simulación física, sobretodo si tenemos en cuenta que se puede disponer de grandes centros de supercomputación para su ejecución, que pueden incluir arquitecturas de computación masiva como GPUs. Estas arquitecturas, unidas a los entornos cloud, permiten aumentar la capacidad de computo de manera exponencial, posibilitando en algunos casos entornos de ejecución Multi-GPU. Sin embargo, adaptar los algoritmos de simulación para llevar a cabo su ejecución en estos entornos implica abordarlo desde una nueva perspectiva. Entre otras cosas, en el campo de la dinámica molecular, esta adaptación se traduce en tratar con la escalabilidad necesaria para simular grandes sistemas moleculares en arquitecturas Muti-GPU.

El primer problema que se debe abordar es la necesidad de particionar el sistema molecular para que sea analizado por los diferentes Elementos de Computo (EC, GPUs en este caso), de manera que los datos de los átomos se puedan mover entre estos durante la simulación. Con el objetivo de obtener un sistema totalmente

escalable, cada EC debe mantener sus propios datos en memoria local, minimizando todo lo posible el uso del resto de datos. A pesar de que la dinámica molecular puede clasificarse como un sistema de partículas (átomos en este caso), también conlleva otros problemas en su paralelización que, por lo general, no se encuentran en estos sistemas. La principal diferencia consiste en que los átomos se encuentran enlazados físicamente. Como se indica en la Sección 3.1, un enlace consiste en una conexión entre dos o más átomos, e implica que el movimiento de uno de ellos influye en el resto de los átomos que componen el enlace. Esta diferencia supone que cada EC no puede simular sus propias fuerzas de enlace de manera independiente, sino que necesita conocer los datos de los enlaces compartidos con otros ECs. Cuando los átomos se mueven, la relación con sus átomos enlazados, los cuales pueden estar en otro EC, se debe mantener. Esto hace necesaria la comunicación entre ECs, no solo para intercambiar datos de átomos, sino también de los enlaces con el objetivo de calcular las fuerzas que estos producen. Esta sección aborda el problema tratando de que cada EC pueda, de manera eficiente, calcular por si mismo las fuerzas de los átomos que le corresponden en cada paso de tiempo de la simulación minimizando las comunicaciones entre ECs.

4.1.1. Datos compartidos de átomos

El primer paso para llevar a cabo la simulación paralela es distribuir los átomos entre los distintos ECs. En este caso, la partición se realiza de forma binaria como muestra 4.1. Los átomos se mueven entre los distintos elementos, por lo que cada EC debe comunicar los cambios que se producen para que todos siempre dispongan de los datos correctos. Para ello es necesario que los datos de los átomos localizados en los bordes de cada partición estén duplicados en las particiones contiguas. De esta forma, cada EC es capaz de calcular correctamente las fuerzas de enlace, electrostáticas y de Van der Waals dentro una distancia de corte R_c definida. Para esto se crean tres

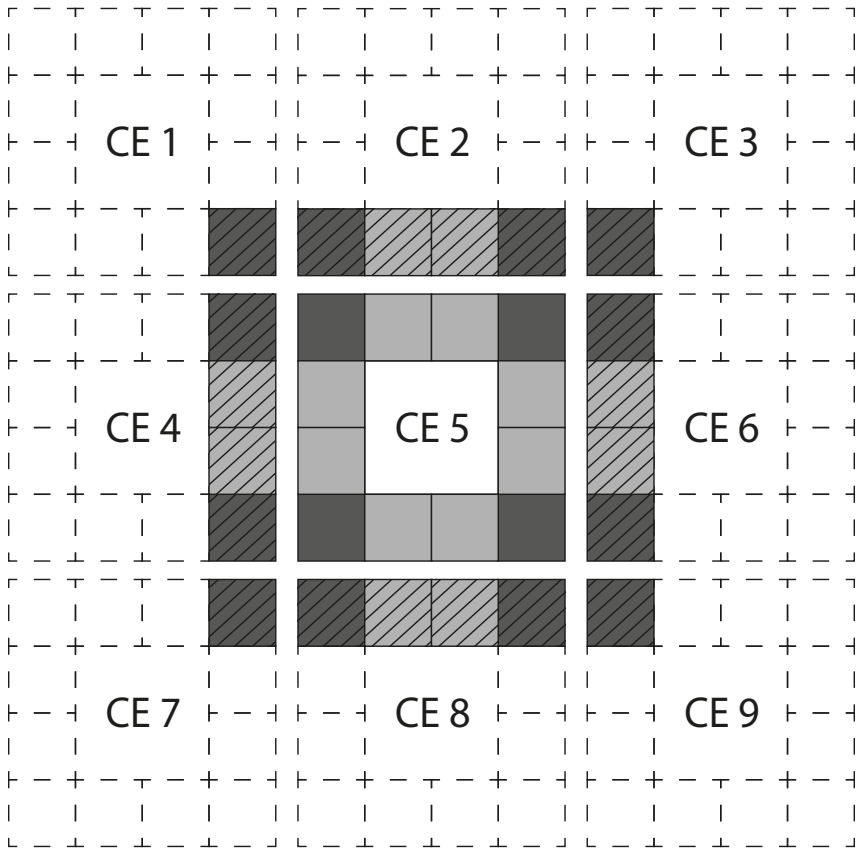


Figura 4.1: *Distribución en 2D del sistema molecular entre los diferentes ECs. Cada EC comparte datos con sus ECs adyacentes (8 en dos dimensiones, 26 en 3 dimensiones). El área de interfaz del EC 5 (zona rayada) contiene los datos compartidos desde otros ECs, que son necesarios para el cálculo de las fuerzas electrostáticas de corto alcance. Además, el EC 5 tiene múltiples áreas de datos compartidos (zona coloreada y sin líneas). Por un lado, comparte los mismos datos con varios ECs (color gris oscuro). Por ejemplo, la esquina adyacente a EC 1 contiene los datos que se comparten con EC 1, EC 2 y EC 4. Por otro lado, se comparten datos de forma individual con un solo EC (color gris claro).*

zonas diferenciadas en cada EC.

- Privada: contiene los átomos que no se comparten con el resto de ECs.

- Compartida: formada por los átomos que se comparten con otros ECs.
- Interfaz: compuesta por todos los átomos que se comparten con otros ECs para poder calcular las fuerzas de los átomos propios del EC, tanto de la zona privada como compartida. La zona compartida y la interfaz tienes los mismos datos, pero en diferentes ECs. Por ejemplo, los átomos almacenados en la interfaz de un EC han sido enviados desde otro EC que tiene dichos datos en la zona compartida. Para estos átomos, las fuerzas se calculan en el EC que los contiene en su zona compartida.

Este sistema de particionamiento permite a cada EC calcular las fuerzas de los átomos que le corresponden sin conocer los datos de los átomos pertenecientes a los otros ECs, lo que supone que cada uno de los ECs solo debe contener los datos indispensables para sus cálculos. Esto hace necesario implementar un sistema de comunicación y actualización de los átomos cuando se mueven entre los ECs.

Para realizar el particionamiento del sistema se utiliza como base el algoritmo *CellList* [52, 62, 17], diseñado para calcular las fuerzas de Van der Waals y fuerzas electrostáticas de corto alcance. Este algoritmo divide el sistema en celdas del mismo tamaño, las cuales se clasifican en las tres zonas vistas anteriormente, como se ve en la Figura 4.2. El tamaño de las celdas es un divisor del radio de corte R_c. Esto significa que, para cada uno de los átomos, es posible agrupar todos los átomos que le afectan dentro de este radio de corte en un subconjunto de celdas. Este conjunto de celdas se encuentra fijo en el sistema, y cada átomo es asignado, al iniciar la simulación, a una de esas celdas en función de su posición global x, y, z dentro del volumen completo del sistema molecular. El contenido de las celdas solo cambia cuando las posiciones de los átomos varían. Cada determinado número de pasos de la simulación, la posición de los átomos dentro de las celdas se recalcula y se actualizan los datos de cada EC si fuera necesario. El radio de corte R_c define el número de celdas

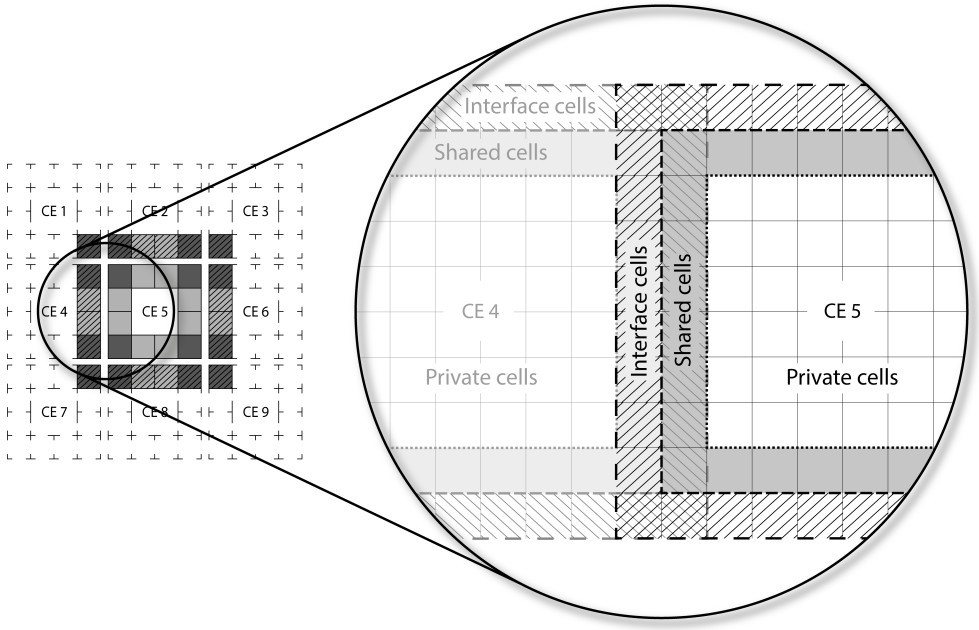


Figura 4.2: *Tipos de celdas. Cada EC contiene los tres tipos de celdas. Las celdas compartidas contienen los átomos que se van a enviar a otros ECs. Las celdas de interfaz (rayadas) almacenan los átomos compartidos por otros ECs, por lo que se solapan con las celdas compartidas de esos otros ECs.*

que se intercambian con otros ECs (zonas interfaz/compartida), con el objetivo de compartir únicamente los datos necesarios para calcular las fuerzas electrostáticas y de Van der Walls [53]. Cada EC calcula estas fuerzas solamente utilizando los datos que ya posee. Se ha de tener en cuenta que cada EC tiene la capacidad de realizar estos cálculos de forma independiente, ya que contiene los datos extra necesarios del resto de ECs almacenados en su propia zona de interfaz. Las fuerzas de los átomos situados en el borde del área compartida, que dependen de los datos de otros ECs, son las que se calculan utilizando ese conjunto de datos extra. Las estructuras de datos utilizadas en el proceso de simulación se han optimizado para reducir el tiempo de comunicación necesario entre ECs, como se describe en la Sección 4.2.

4.1.2. Datos de enlaces compartidos

Los enlaces entre los átomos del sistema molecular suponen que el movimiento de cualquiera de los átomos provoque un tirón sobre otros debido a las fuerzas de enlace. Cuando los datos se actualizan entre los ECs, es necesario enviar no solo los átomos que cambian de EC, sino también los enlaces a los que pertenecen (ver Figura 4.3). Enviar un átomo a un EC vecino implica que el receptor debe calcular todas sus fuerzas en cada paso de tiempo, incluidas las fuerzas de enlace. Los átomos recibidos se colocan en las celdas de interfaz, por lo que el receptor no debe calcular las fuerzas en ese momento, sino que solo pasan a calcularse cuando los átomos se mueven a alguna de las celdas compartidas o privadas del EC. Debido a esto, el receptor no necesita tener los datos de los enlaces hasta que los átomos no pasen a esta zona privada. Los enlaces, al igual que los átomos, se consideran un elemento más del sistema, con sus propias coordenadas cartesianas x, y, z. Estos también se deben enviar cuando sus posiciones correspondan al área de interfaz de otro EC. A partir de ese punto cada EC almacena sus enlaces y átomos correspondientes y puede calcular las fuerzas de enlace, de Van der Waals y electrostáticas de corto alcance para las celdas de su zona privada.

Un enlace puede combinar dos o más átomos, lo que supone compartir información de átomos que pertenecen a diferentes ECs. Por lo tanto, se necesita identificar cada átomo de manera global para poder mantener una correcta relación entre los átomos que se encuentran enlazados entre si y pertenecen a diferentes ECs. Mantener una lista de todos los átomos en cada EC para disponer de un acceso rápido supondría un alto coste de memoria, lo que supondría una reducción drástica de la escalabilidad del sistema. Como cada EC solo calcula las interacciones para sus átomos de la zona privada y la compartida, es mucho más eficiente almacenar solo los datos necesarios y trabajar con ellos de forma local, sin que exista la necesidad

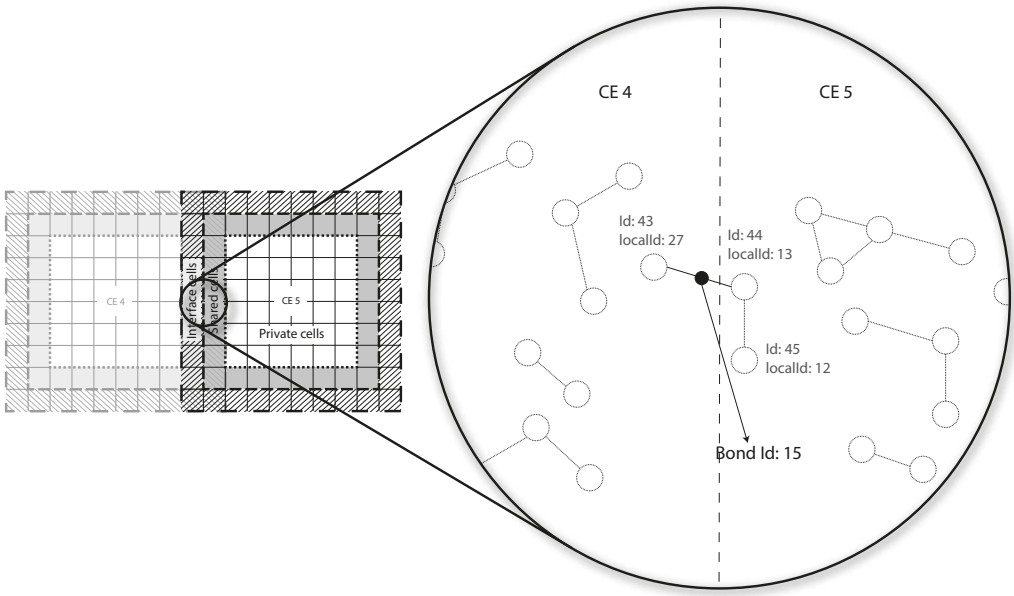


Figura 4.3: *Enlaces distribuidos entre dos ECs. En este caso un átomo se mueve de una celda privada a un área compartida. Los datos de este átomo se deben enviar a los vecinos correspondientes para que puedan calcular sus fuerzas electrostáticas de corto alcance. También es necesario enviar los enlaces que afectan a este átomo cuando se mueve de una celda compartida a un área de interfaz (por ejemplo, id 43), de manera que las fuerzas de enlace se puedan calcular en el EC correspondiente. El EC destinatario debe conocer el identificador de todos los átomos afectados por dicho enlace (en este caso los ids 43, 44 y 45).*

de que cada EC conozca el resto de los datos del sistema. Para disponer de esta visión local y global, se identifican los átomos en el sistema utilizando dos tipos de identificadores: i) globales, que identifican el átomo dentro del sistema molecular completo; y ii) locales, que identifican cada átomo dentro del propio EC. Cada vez que un átomo se envía a un EC, el identificador se debe convertir de local a global, y al recibirlo generar un nuevo identificador local propio para el EC receptor.

Para realizar esta conversión se utiliza una tabla hash en GPU. En concreto

se utiliza una implementación de tabla hash basada en Nvidia CUDPP [9]. Esta implementación permite reducir drásticamente el tiempo de ejecución y el uso de memoria a la hora de realizar la conversión de identificadores. La tabla 4.1 muestra la diferencia de rendimiento entre el uso de un diccionario almacenado en memoria y la tabla hash, utilizando como EC una GPU Nvidia GeForce GTX 1070. El uso del diccionario provee de un acceso mucho más rápido y eficiente cuando este se produce de forma ordenada, siguiendo la estructura interna de la GPU. Sin embargo, no es posible predecir el movimiento de los átomos y obtener sus datos de manera ordenada. Por esta razón, para esta prueba se han realizado accesos a memoria de forma aleatoria, tal como sucede durante el proceso de simulación. La tabla hash utiliza la mitad de tiempo que el diccionario para obtener los datos, requiriendo también menos memoria. La diferencia de tiempo que se puede observar se debe a que los accesos a la memoria de la GPU no se realizan de manera coalescente, penalizando así el tiempo de acceso al diccionario.

Tabla 4.1: *Comparación entre un diccionario de datos almacenado en GPU y una tabla hash, accediendo a un millón de números aleatorios en una Nvidia GeForce GTX 1070*

	Tiempo de acceso	**Uso de memoria**
Diccionario en GPU	4.91 ms	7.63 MBytes
Tabla hash en GPU	2.25 ms	4.77 MBytes

Cuando un EC recibe un nuevo átomo, se crea ese átomo de forma local y se le asigna el identificador global. Este identificador sirve para identificar con que otros átomos de otros ECs está enlazado. Los átomos locales se organizan de forma ordenada según su posición en las celdas para obtener un acceso consecutivo a los datos. Esta disposición de los átomos reduce el tiempo de acceso a memoria y optimiza los cálculos de fuerzas para cada celda.

4.1.3. Optimizaciones para sistemas moleculares pequeños

Aunque el algoritmo de simulación de dinámica molecular funciona según lo esperado con grandes sistemas moleculares y un número elevado de GPUs, con sistemas moleculares más pequeños, como se verá más adelante en la sección 6.1, el tiempo de ejecución es superior al que requieren otros algoritmos de simulación con los que se ha llevado a cabo una comparativa. Esto es debido a que la cantidad de átomos en cada GPU es demasiado baja y no se aprovecha todo el potencial de la distribución.

Sin embargo, no solo supone un bajo uso del potencial de procesamiento de la GPU, sino también de su memoria. Sin embargo, esto abre la posibilidad de incrementar la cantidad de átomos que se almacenan para cada EC, siempre teniendo en cuenta que el mayor coste computacional proviene de las fuerzas electrostáticas de corto alcance y de las fuerzas de Van der Walls. Por ello, aun realizándose el cálculo de estas fuerzas en cada uno de los pasos de simulación, es posible aproximar alguno de los valores necesarios para calcular las fuerzas de Van der Walls y actualizar estos datos cada cierto número de pasos de tiempo (por ejemplo, en cada actualización del contenido de las celdas). Por lo tanto, en la ecuación 2.2 es posible calcular A y B una sola vez y almacenar esos valores para cada átomo en una "tabla de vecindades". Esta aproximación, al actualizarse cada cierto número de pasos, apenas tiene incidencia en la precisión del resultado final, pero permite reducir el tiempo de ejecución a coste de ocupar una mayor cantidad de memoria de GPU.

4.2. Algoritmo de simulación en paralelo

Para resolver las ecuaciones del movimiento y calcular la trayectoria de los átomos del sistema molecular se utiliza el método de integración RESPA, visto en la sección 2.1.2. Esta decisión viene dada por la capacidad que ofrece el método para resolver los distintos tipos de interacciones atómicas con pasos de tiempo de distinto valor,

de forma que los cálculos de mayor complejidad, como son las fuerzas electrostáticas, se pueden realizar un menor número de veces sin que la precisión del resultado se vea comprometida. Las ecuaciones del método de integración son las siguientes:

$$v(t + \frac{\Delta t}{2}) = v(t) + \frac{1}{2m}F(t) \tag{4.1}$$

$$v(t + \Delta t) = v(t + \frac{\Delta t}{2}) + \frac{1}{2m}F(x(t + \Delta t)) \tag{4.2}$$

$$x(t + \Delta t) = x(t) + v(t)\Delta t + \frac{1}{2m}F(t)\Delta t^2 \tag{4.3}$$

donde $v(t)$ la ultima velocidad obtenida de cada átomo y $x(t)$ su ultima posición.

La nueva posición de cada átomo se calcula evaluando las distintas fuerzas que actúan sobre el sistema, siguiendo las ecuaciones descritas en la Sección 2.1.1. El algoritmo 2 muestra la implementación del integrador RESPA descrito en la Sección 2.1.2. Existen diversos datos compartidos entre ECs (átomos y enlaces) por los que se debe disponer de funciones de intercambio para que la simulación se pueda ejecutar en un entorno distribuido. Debido al sistema de particionamiento elegido, estos datos compartidos entre ECs se deben actualizar en cada paso de tiempo de la simulación. Por ejemplo, las nuevas posiciones calculadas para los átomos localizados en el conjunto de las celdas compartidas se deben enviar a los ECs adyacentes para que puedan actualizar las posiciones de los átomos localizados en la zona de interfaz. Esto se debe a que cada EC calcula las fuerzas, velocidades y posiciones de los átomos en sus celdas privadas y compartidas, pero no las de los átomos situados en sus celdas de interfaz. Tras los cálculos de cada step, los ECs envían las posiciones de los átomos, que están situados en su conjunto de celdas compartidas,

Algoritmo 2 Integrador Verlet/RESPA para Multi-GPU.

```
1: procedure STEP(currentStep)
2:     integrateHalfVelocity() ← Equation (4.2)
3:     integratePosition() ← Equation (4.3)
4:     transferPositions(sharedIDs)
5:     if currentStep mód stepUpdate = 0 then
6:         atomIDs[]=identifyEvictedAtomIDs(AtomInfo)
7:         bondIDs[]=identifyEvictedBondsIDs(AtomInfo, BondElements)
8:         transferEvicted(atomIDs, bondIDs)
9:         updateCellList()
10:        sharedIDs[] = identifySharedAtomIds(CellList)
11:        transferPositions(sharedIDs)
12:    end if
13:    computeBondedForces()
14:    computeNonBondedShortRangeForces()
15:    integrateHalfVelocity() ← Equation (4.1)
16:    currentStep = currentStep + 1
17: end procedure
```

Algoritmo 3 Identificación de los átomos que pasan a estar localizados en otro EC.

```
1: procedure IDENTIFYEVICTEDATOMIDS(AtomInfo)
2:     localEvicted[] = computeAtomsEvicted(AtomInfo)
3:     globalEvicted[] = hashTable.retrieve(localEvicted)
4:     mergeSort(globalEvicted)
5:     return globalEvicted
6: end procedure
```

a los ECs adyacentes y, a su vez, reciben en sus celdas de interfaz las posiciones de los átomos desde esos mismos ECs adyacentes. El movimiento de los átomos en cada paso de tiempo es lento, y es inferior al tamaño de una celda, por lo que no es necesario actualizar los identificadores de los átomos contenidos en las celdas en cada paso [62]. Después de un determinado número de pasos (*stepUpdate*), la localización de los átomos y enlaces dentro del cellList debe ser recalculada, enviando los átomos que se han salido de cada EC a sus vecinos correspondientes. Para ello es necesario localizar los átomos y enlaces que han cambiado su posición en el sistema y se debe transferir a otro EC. Esto se realiza con las funciones *identifyEvictedAtomsIDs* y *identifyEvictedBondsIDs*. El algoritmo 3 muestra como se identifican los átomos que han pasado a la zona compartida, y por tanto se deben enviar a la interfaz de otro EC. De manera similar se puede extrapolar al algoritmo 4 considerando los enlaces como un elemento más del sistema, con sus propias posiciones cartesianas dentro del volumen molecular. Una vez identificados, los átomos y los enlaces se transfieren al EC correspondiente.

Algoritmo 4 Obtención de los identificadores de los enlaces que se deben enviar a la interfaz. Los enlaces se consideran un elemento más del sistema, tienen sus propias coordenadas x, y, z y, por tanto, se pueden ubicar en el CellList.

1: **procedure** IDENTIFYEVICTEDBONDIDS($AtomInfo$, $BondElements$)
2: $computeBondsPositions(BondElements, AtomInfo)$
3: $localEvicted[]$=$computeBondsEvicted(BondElements, BondPositions)$
4: $globalEvicted[]$=$hashTable.retrieve(localEvicted)$
5: $mergeSort(globalEvicted)$
6: return $globalEvicted$
7: **end procedure**

Cuando cada EC ha recibido los nuevos átomos y enlaces en su zona de interfaz, se

rehacen las estructuras de datos almacenando los átomos y enlaces locales en sus celdas correspondientes, utilizando la función *updateCellList*. Esto permite llevar a cabo una mejor organización de estos elementos, de acuerdo a sus identificadores locales de en cada EC con el objetivo de mejorar el rendimiento. Al cambiar la localización de los átomos dentro de las celdas se necesita identificar que átomos se encuentran en ese momento en las celdas compartidas utilizando *identifySharedAtomsIds*, que actualiza la posición de los átomos entre ECs. Finalmente, cada EC envía las posiciones de los átomos ubicados en sus celdas compartidas con *transferPositions*. El algoritmo 5 muestra como se identifican los átomos de las celdas compartidas y se envían solo sus posiciones a los mismos átomos situados en las celdas interfaz de los ECs adyacentes.

Para tratar la información que se necesita se utilizan diversas estructuras de datos, optimizando el cálculo de fuerzas y el intercambio de datos entre ECs. Más específicamente, se utilizan las siguientes estructuras de datos para almacenar los

Algoritmo 5 Obtención de los identificadores de los átomos que se van a compartir con las celdas de interfaz de otros ECs.

```
1: procedure IdentifySharedAtomIds(CellList)
2:     sharedLocalIds[]
3:     sharedCells[] = CellList.getSharedCells()
4:     for each item cell in sharedCells do
5:         sharedLocalIds.add(cell.getAtomIds())
6:     end for
7:     sharedGlobalIds[] = hashTable.retrieve(sharedLocalIds)
8:     mergeSort(sharedGlobalIds)
9:     return sharedGlobalIds
10: end procedure
```

átomos en cada EC:

- *AtomInfo* contiene múltiple información acerca de los átomos del EC (posiciones, velocidades, parámetros físicos, e identificadores locales y globales).
- *AtomData* contiene los enlaces a los que pertenece cada uno de los átomos. Se encuentra directamente relacionado con *AtomInfo*.
- *CellList* contiene las celdas de cada EC, ordenado según el tipo de celda. Cada celda almacena los identificadores locales de los átomos que contiene.

Para almacenar los enlaces y su relación con los átomos se utilizan las siguientes estructuras:

- Las estructuras*BondElement, AngleElement, DihedralElement, ImproperElement* contienen los datos de los enlaces para el cálculo de las fuerzas, así como los identificadores locales de los átomos afectados.
- *BondPositions, AnglePositions, DihedralPositions, ImproperPositions* contienen la posición de cada enlace en el espacio tridimensional. Estas posiciones se calculan cada *stepUpdate* pasos, y se utilizan para transferir átomos y enlaces entre ECs.

Cuando se actualizan los datos entre ECs, cada EC calcula la nueva posición de cada átomo y enlace, y almacena estos en *AtomInfo* y *BondPositions* respectivamente. Las posiciones de los átomos se utilizan en la simulación, junto con otros datos contenidos en *AtomInfo*, mientras que *BondPositions* solo se utiliza para realizar la transmisión de datos. Después de un número de pasos predefinido (*stepUpdate*), cada EC calcula la nueva localización de los átomos y de los enlaces en el conjunto de celdas y envía los datos contenidos en las celdas compartidas a los ECs adyacentes. Cada EC actualiza las posiciones de los átomos en la zona de interfaz con los datos

recibidos de los ECs vecinos. Al mismo tiempo envía las posiciones de los átomos en su propia zona compartida, lo que permite a cada EC calcular las nuevas fuerzas con los datos más recientes de la simulación

El algoritmo de simulación calcula en paralelo las fuerzas de enlace utilizando los datos de las estructuras locales de cada EC. Cada átomo local contiene referencias a los enlaces en los que está incluido, y cada enlace contiene referencias a los átomos que interconecta. Para el cálculo de fuerzas de enlace se obtienen los enlaces de cada átomo, y de estos se obtienen todos los átomos involucrados, que son necesarios para llevar a cabo el proceso. La figura 4.4 muestra en detalle este proceso.

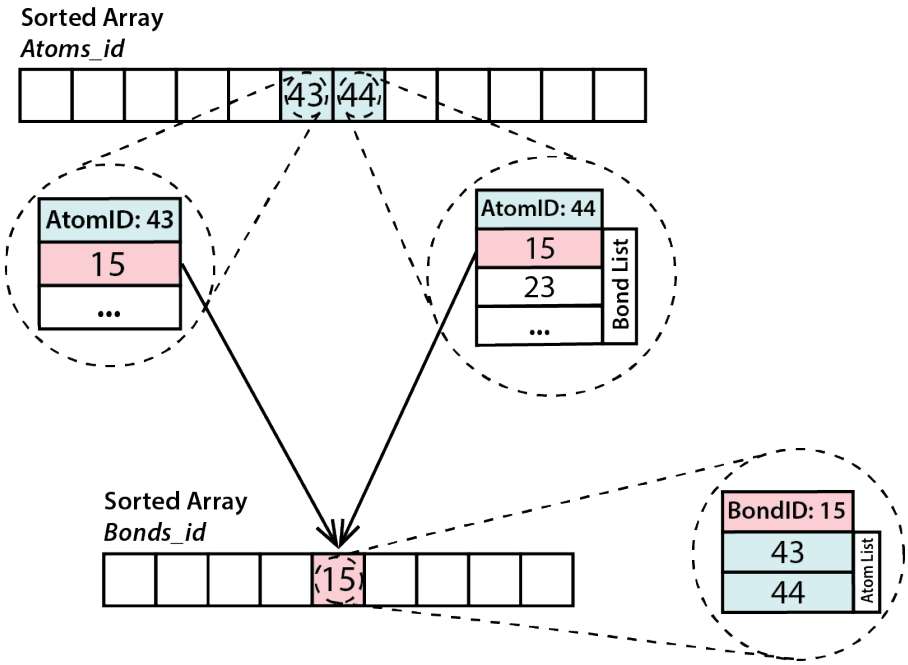


Figura 4.4: *Relación entre átomos y enlaces. Las estructuras de datos facilitan el cálculo de las fuerzas de enlace para cada átomo, de forma que no hay que almacenar datos replicados o inconsistentes.*

4.2.1. Arquitectura Hardware Multi-GPU

El particionado de grandes sistemas moleculares en un entorno Multi-GPU implica la necesidad de un sistema de comunicación hardware. Para este propósito se pueden implementar principalmente dos tipos de arquitecturas Multi-GPU. Una posibilidad es utilizar una arquitectura hardware con un solo equipo y múltiples GPUs, a través de conexiones GPUDirect, NVLink, NVSwitch o PCIe [35]. Esta distribución es eficiente en términos de comunicación entre GPUs, ya que se realiza a través del bus de datos de la placa base. Sin embargo, esta es su mayor limitación, ya que número máximo de GPUs que se pueden utilizar es pequeño (p.e. en una arquitectura NVSwitch solo se pueden conectar hasta 16 GPUs).

Otra posibilidad es utilizar arquitecturas Multi-GPU con las GPUs distribuidas entre diferentes nodos. Cada nodo, que puede contener una o más GPUs, se comunica con los demás utilizando redes avanzadas de alta velocidad. Tradicionalmente, la información se enviaba desde la RAM del nodo emisor a través de la interfaz de red, y los datos se recibían en la RAM del receptor. Este método produce un retraso en la comunicación entre GPUs, debido a que antes de enviar los datos se deben copiar de la GPU a la RAM, y al recibirlos se copian de la RAM a la memoria de la GPU destinataria.

Actualmente los entornos Multi-GPU disponen de una comunicación directa entre GPUs a través de la interfaz de red. Este tipo de comunicación, conocido como GPUDirect RDMA (remote direct memory access), permite a los dispositivos de red acceder directamente a la memoria de GPU, reduciendo drásticamente el tiempo de comunicación entre GPUs al evitar los accesos a RAM (ver Figura 4.5). Además, el estándar MPI (Message Passing Interface) ofrece la posibilidad de realizar este intercambio a través de MPI CUDA-aware, llevando a cabo todo el proceso de forma transparente. MPI CUDA-aware abstrae el sistema de comunicación y permite en-

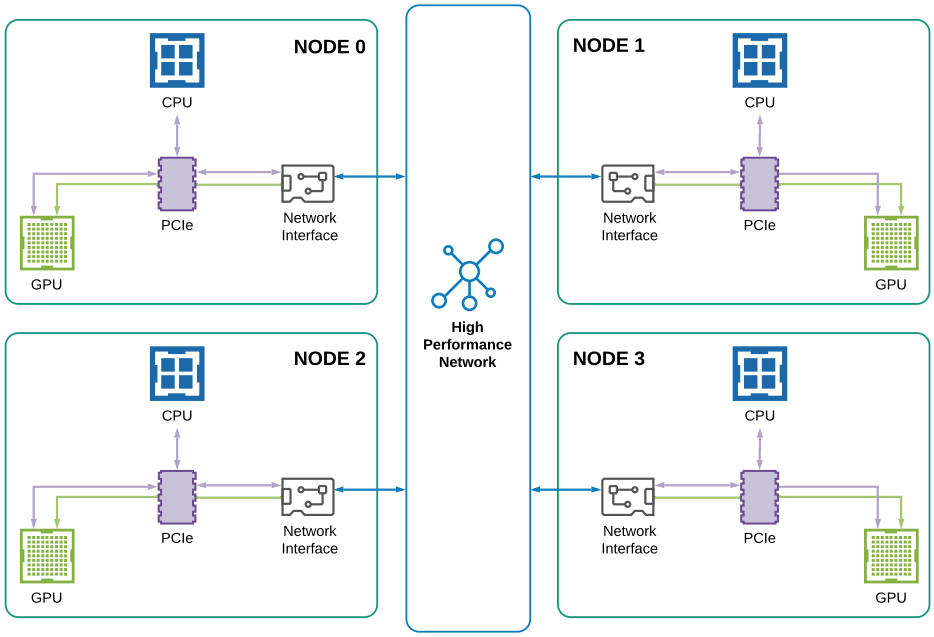


Figura 4.5: *Arquitectura remota con GPU-direct.*

viar y recibir datos entre GPUs directamente, sin que tengan que pasar por la RAM. Esto proporciona la capacidad de desarrollar el simulador sin que influya el tipo de arquitectura disponible: En un solo equipo, conexión de red tradicional o GPUDirect RDMA. Esta abstracción además admite el uso de todas estas arquitecturas de manera híbrida.

4.3. Procesado de datos moleculares

La ejecución de una simulación de dinámica molecular implica, además del propio proceso de simulación, la preparación de los datos moleculares antes de llevar a cabo la ejecución. Para ello se pueden obtener las diferentes estructuras moleculares de repositorios públicos, como Protein Data Bank (PDB) [10], que ofrece los datos estructurales de grandes moléculas biológicas. La información obtenida de estos

repositorios está incompleta debido a que únicamente consta de la descripción del sistema, y solo provee de los datos moleculares básicos como el tipo de átomo y su posición en el espacio tridimensional. Por ello se debe realizar un preprocesado de estos sistemas, el cual se puede dividir en tres pasos que, en conjunto, forman un flujo de datos a aplicar sobre el sistema (véase Figura 4.6):

1. Añadir los detalles de las estructuras moleculares y sus interacciones a nivel atómico.

2. Añadir el sistema molecular a un solvente, siendo agua en la mayoría de los casos.

3. Equilibrar térmicamente el sistema molecular, de forma que este se encuentre en un estado de baja energía [25].

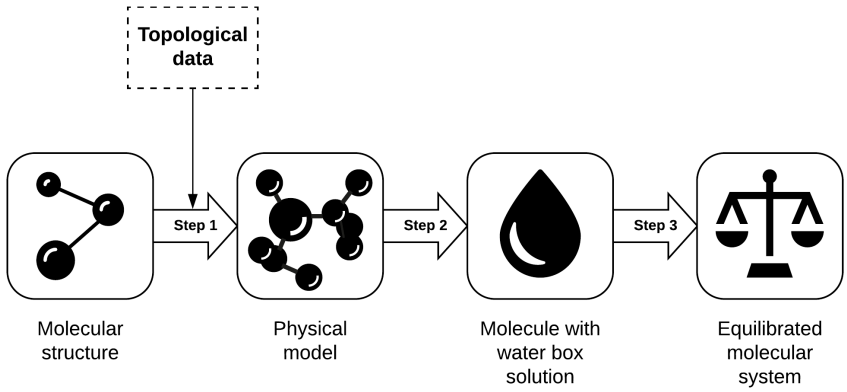


Figura 4.6: *Preprocesado realizado para pasar de los datos moleculares iniciales a un sistema molecular equilibrado y preparado para llevar a cabo la simulación.*

Estos procesos suponen un amplio conocimiento de múltiples softwares específicos de dinámica molecular, por ello en los siguientes apartados se compilan los diferentes

pasos que se deben aplica sobre el sistema para llevar a cabo la simulación de dinámica molecular.

4.3.1. Topología de los sistemas moleculares

La captura de los datos de sistemas moleculares biológicos se realiza con propósitos genéricos, no solo de simulación de dinámica molecular. Por esta razón los repositorios que proveen de estas estructuras biológicas solo proporcionan los datos más básicos del sistema, como el tipo de átomos y sus posiciones. Es necesario llevar a cabo un proceso mediante el cual, utilizando una serie de datos topológicos, se completan las estructuras moleculares obteniendo los diversos datos necesarios para la simulación como los enlaces a los que pertenece cada átomo, su masa o su carga electrostática. Los ficheros de topología utilizados contienen estos datos para cada tipo de composición molecular, de manera que el proceso permite obtener la relación entre dichos ficheros y la estructura del sistema molecular.

Existen diversos paquetes de software que permiten llevar a cabo este proceso. Por ejemplo, el plugin PSFGen [6] desarrollado para VMD [32] permite comparar y relacionar cada átomo del sistema molecular con diversas descripciones topológicas, añadiendo los datos necesarios para simular el sistema.

4.3.2. Solución en agua de los sistemas moleculares

La estructura molecular obtenida en la sección anterior, aunque es topológicamente correcta, no podría existir en la realidad ya que el sistema se encuentra ubicado en el vacío. Al igual que se haría para obtener el plegado del sistema molecular en un entorno real, este se debe encontrar en un solvente (agua en la mayoría de los casos). Para ello se añade un conjunto de moléculas agua al sistema, formando una estructura poligonal, donde la cantidad átomos que se añaden dependerá del tamaño de la estructura molecular a simular. Este proceso se puede realizar con el plugin

Solvate [7] desarrollado para VMD [32].

4.3.3. Equilibrio energético del sistema molecular

El bloque de agua añadido al sistema es inexacto, ya se que ha insertado de manera independiente al resto de átomos. Esto significa que, teniendo en cuenta la distancia entre los átomos y sus propiedades, el sistema molecular se encuentra en un estado de alta energía (sistema inestable). Un sistema molecular con este nivel de energía no puede existir en el mundo real, por lo que no es posible realizar la simulación. Debido a esto es necesario pasar el sistema a un estado de baja energía, el cual se consigue mediante un proceso de equilibrado molecular [25]. Para este proceso se utiliza el simulador de dinámica molecular NAMD [45], que dispone de una herramienta de equilibrado para el sistema molecular utilizando el método de la desviación cuadrática media (RMSD) [15]

La estructura molecular resultante, la cual ya esta lista para usarse con un simulador de dinámica molecular, incrementa su tamaño en torno a un 40 %. Dependiendo del sistema molecular, su tamaño puede variar desde unos pocos megabytes hasta cientos de gigabytes. Por esta razón, todo el proceso descrito supone un gran coste de recursos computacionales, y su ejecución encaja a la perfección en un entorno de cloud computing

Capítulo 5

Dinámica molecular en entornos de Cloud Computing

La simulación de dinámica molecular implica la necesidad de un entorno de computación de alto rendimiento donde poder llevarla a cabo. En la actualidad, estos entornos se pueden encontrar mayoritariamente en servicios de cloud computing que permiten, bajo demanda, disponer de una gran cantidad de recursos computacionales. El principal inconveniente de estos servicios es la preparación del entorno, tanto la arquitectura y configuración de los componentes hardware requeridos como la instalación y configuración software de simulación.

Este capítulo presenta una solución basada en una interfaz web para ejecutar el software de dinámica molecular en entornos de Cloud Computing, desplegando automáticamente el hardware requerido para cada uno de los pasos de la simulación. Además, esta herramienta permite gestionar estos pasos y almacenar todos los datos en el servicio de almacenamiento de la plataforma Cloud. El principal objetivo de esta solución es abstraer al usuario de los recursos necesarios para llevar a cabo el proceso de simulación

5.1. Despliegue de arquitecturas en servicios Cloud

El principal objetivo de esta solución de software es abstraer al usuario de los recursos necesarios para llevar a cabo la simulación. Para ello se realizan todos los procesos relacionados con la simulación de dinámica molecular directamente en el servicio Cloud. Por esto se plantea el desarrollo de una herramienta basada en una aplicación web, la cual provee de un servicio de dinámica molecular optimizado por el uso de GPUs y se ejecuta en servicios de cloud computing. La figura 5.1 muestra en detalle la arquitectura de la aplicación web propuesta en relación con cada uno de los pasos del procesado previo de los datos. La funcionalidad principal de la aplicación se ubica en un servidor web que, no estando necesariamente localizando en el servicio de computación cloud, utiliza recursos remotos que si se encuentran en el entorno de cloud computing para llevar a cabo todos los pasos previos de la simulación descritos en la sección 4.3, además del proceso de simulación de dinámica molecular. El servidor dispone de un servicio web, el cual actúa como enlace entre la aplicación y la infraestructura remota necesaria para la simulación permitiendo controlar todo el proceso. Este servicio web se puede integrar para controlar los entornos de cloud computing más utilizados en la actualidad, como son Microsoft Azure, Amazon Web Services, Google Cloud Computing, etc. Todas estas plataformas disponen de maquinas virtuales con un hardware preconfigurado (También llamadas instancias). Existen instancias con CPU y GPU para las que se puede solicitar una, múltiples o una fracción de GPU, las cuales se pueden utilizar para crear una infraestructura Multi-GPU para dinámica molecular. Las plataformas de servicios Cloud también proveen de servicios de almacenamiento. A estos servicios se puede acceder remotamente desde la herramienta y las distintas instancias para almacenar los datos moleculares y sus correspondientes metadatos.

El servicio web se implementa como una API REST que permite a la aplicación

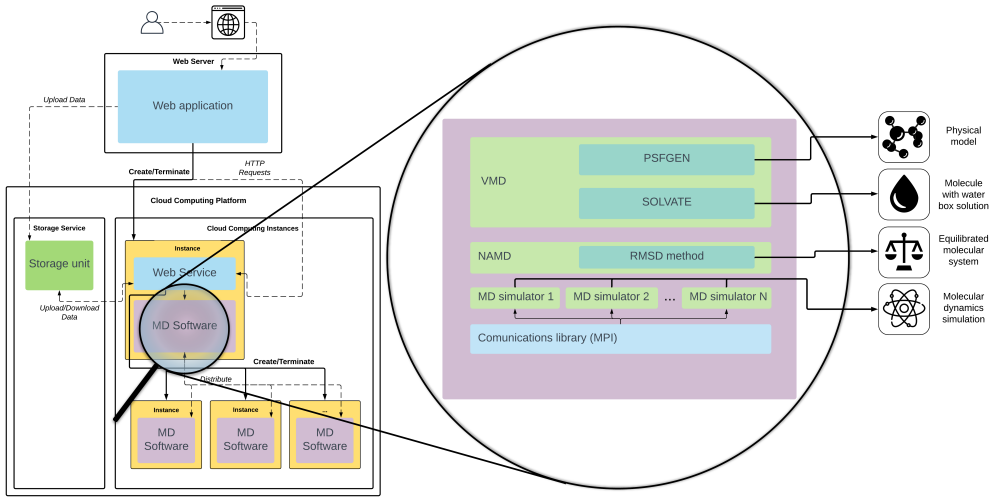


Figura 5.1: *Diagrama de la arquitectura en el entorno de Cloud Computing. En azul se muestran los componentes desarrollados específicamente para esta herramienta. En amarillo aparecen las instancias diseñadas para ejecutar software de dinámica molecular y comunicarse con otras instancias y con la aplicación web. Los componentes marcados en verde muestran los distintos componentes software desplegados en las instancias*

principal realizar peticiones a través del protocolo HTTP, tanto para iniciar como para terminar la ejecución del software y obtener su estado en cada momento. También tiene la capacidad de descargar los datos del servicio de almacenamiento y subir a este los resultados obtenidos después de cada ejecución. Este servicio, al igual que todo el software adicional necesario para la simulación de dinámica molecular, se despliega en una imagen de sistema de desarrollo propio, diseñada específicamente para sacar provecho de la infraestructura Multi-GPU en el entorno cloud. La imagen está basada en un sistema operativo Linux y contiene el servicio web, el software VMD con los plugins PSFGen y Solvate, el simulador de dinámica molecular NAMD con el algoritmo RMSD, librerías de comunicación entre las distintas instancias, como OpenMPI [24], y el software de simulación de dinámica molecular. La herramienta

inicia las instancias de acuerdo con la tarea que se desea llevar a cabo, utilizado solo las que disponen de GPU cuando el uso de esta es necesario. El principal objetivo es que los usuarios no tengan que preocuparse sobre como se inician o finalizan los servicios cloud durante todo el proceso.

Esta propuesta gestiona de manera automática los siguientes recursos en la plataforma Cloud:

- Usuarios. La gestión de usuarios permite a cada uno disponer de su propio espacio, donde pueden administrar sus claves de acceso a la plataforma Cloud. Las claves de acceso se pueden obtener directamente desde el panel de control de la propia plataforma. En cada clave se configuran múltiples permisos para el usuario en la infraestructura cloud.

- Almacenamiento. Debido a que la simulación de dinámica molecular requiere y genera una gran cantidad de información, se delega la gestión de los datos al servicio de almacenamiento de la plataforma Cloud. En la herramienta, por cada clave de acceso, se permite configurar múltiples workspaces en el servicio de almacenamiento, considerando que se necesita uno por cada sistema molecular con el que se trabaja. La aplicación utiliza cada workspace para almacenar todos los datos relacionados con el procesado de los datos y la simulación.

- Instancias. Como se ha visto anteriormente, los usuarios pueden querer ejecutar diferentes etapas del procesado de datos, incluyendo la simulación. Cuando se realiza la ejecución de una etapa, la herramienta automáticamente despliega las instancias solicitadas en la infraestructura cloud, iniciando la imagen del sistema preconfigurada con todo el software de dinámica molecular requerido.

- Comunicación de red. La herramienta controla la comunicación entre las instancias y el servicio del almacenamiento al desplegar la infraestructura. En

el caso de utilizar múltiples instancias con GPU al mismo tiempo para llevar a cabo la simulación, la herramienta gestiona esta comunicación de manera eficiente entre cada instancia. Se debe tener en cuenta que la clave de acceso al entorno cloud debe disponer de los correspondientes permisos para poder comunicar las instancias y los servicios

- Monitorización. Aunque es posible utilizar sistemas de monitorización completos para entornos cloud [36], la herramienta provee al usuario solo de la información relativa al estado de la ejecución, con el objetivo de simplificar su adaptación al entorno. Estos datos se obtienen directamente de la información que provee el servicio web acerca de la infraestructura cloud. Cada una de las instancias transmite a la aplicación el estado de la ejecución, que puede ser:

  1. *Preparing*: Los datos se están descargando desde el servicio de almacenamiento remoto.

  2. *Running*: El proceso se esta ejecutando en el servicio de Cloud Computing.

  3. *Finishing*: La tarea se ha completado con éxito y los datos obtenidos se están subiendo al servicio de almacenamiento.

  4. *Terminate*: La tarea se ha completado y los datos se han guardado correctamente. Al recibir este estado la aplicación finaliza las instancias utilizadas para la tarea.

  5. *Error*: Ha ocurrido un error durante la ejecución del proceso. Este mensaje va asociado a un log de la ejecución. Al recibir este estado la aplicación finaliza las instancias utilizadas para la tarea.

5.2. Proceso de ejecución e interacción con el usuario

La arquitectura definida en la sección anterior permite a los usuarios acceder a la herramienta a través de un navegador web (utilizando smartphones, portatiles, tablets, etc.) para poder ejecutar fácilmente la simulación de dinámica molecular. Como se ha visto, la herramienta encapsula la funcionalidad de simulación de dinámica molecular y la configuración del entorno cloud, permitiendo a los usuarios realizar todo el flujo de trabajo a través de una interfaz amigable y fácil de usar. Todo el software de simulación se ejecuta en la infraestructura Cloud, y los usuarios pueden dejar sus tareas ejecutándose en segundo plano. Este comportamiento encaja con la descripción de *Software as a Service (SaaS).*

La figura 5.2 muestra la interacción entre los usuarios y el sistema propuesto. Los usuarios interactúan con la aplicación a través de una interfaz web mientras el servidor web se encarga de ejecutar todo el conjunto de mecanismos relacionados con los servicios de Cloud Computing. El proceso de ejecución es el siguiente: Primero, el usuario configura la tarea que va a llevar a cabo la aplicación. El servidor web inicia la/s instancia/s requerida/s de acuerdo con la configuración seleccionada, utilizando la imagen de sistema preconfigurada. Una vez que el servidor web comprueba que la comunicación se ha establecido con las nuevas instancias, se envía una petición con el script de ejecución. Las instancias descargan todos los datos necesarios para llevar a cabo la operación desde el servicio de almacenamiento. Cuando la tarea se ha completado, cada instancia sube los resultados obtenidos al servicio de almacenamiento, notifica la finalización de la tarea al servidor web y libera los recursos. En caso de error, la instancia devuelve un mensaje de error junto con les log del software ejecutado. Finalmente, el servidor web notifica al usuario a través de la interfaz web indicando que la tarea ha terminado.

La herramienta propuesta se ha diseñado para utilizar servicios de cloud compu-

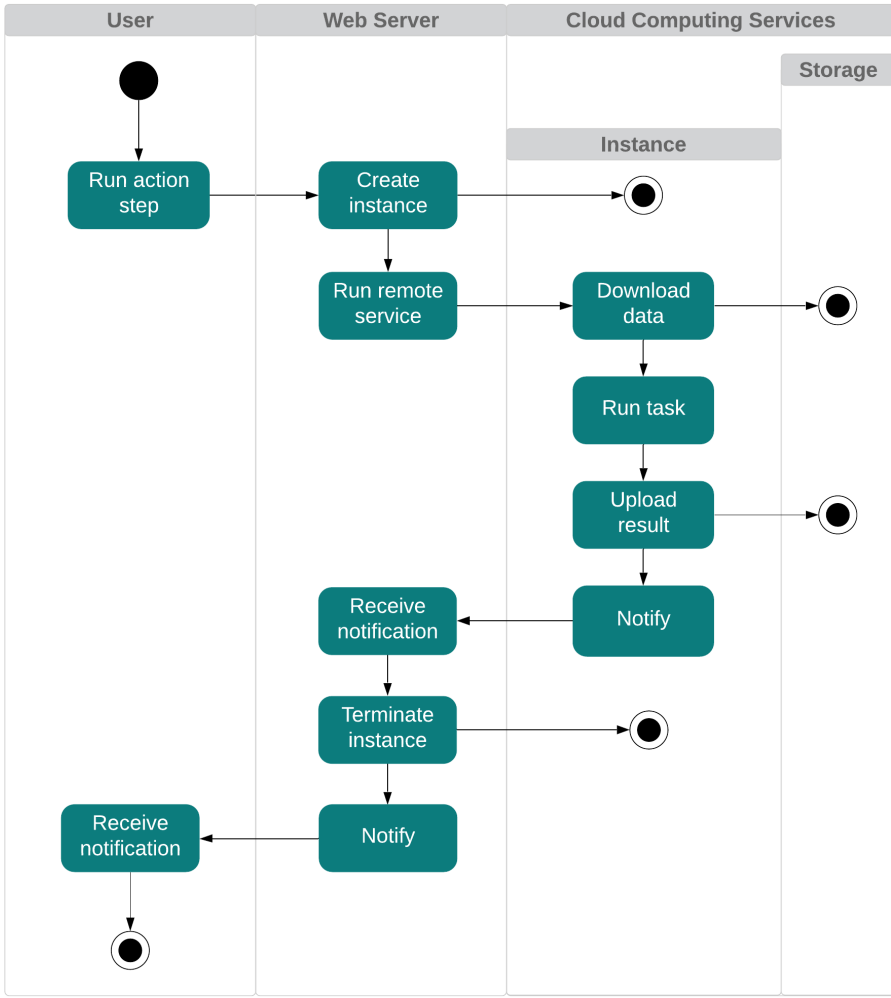


Figura 5.2: *Flujo de trabajo de la herramienta en todas sus etapas, desde el momento en que el usuario inicia una tarea hasta que esta ha terminado y se notifica el estado*

ting, reduciendo todo lo posible su coste evitando gastos innecesarios en ejecuciones fallidas o que finalizan antes de los esperado. Además, la aplicación permite a los usuarios realizar la ejecución de tareas de dinámica molecular que necesitan grandes

tiempos de ejecución de forma desatendida. La herramienta es robusta frente a fallos de comunicación con el servicio cloud o desconexiones del servidor web. En cada paso del proceso de simulación, la aplicación almacena el estado de ejecución y los IDs de cada una de las instancias implicadas. Esto significa que, si el servidor web se desconecta, es posible recuperar el estado de ejecución sin sufrir perdidas en el progreso. También se tratan los errores que suceden durante la ejecución del software que se está gestionando. En caso de fallo, la aplicación recibe el estado correspondiente de la instancia, el cual contiene los datos de log del software que se estaba ejecutando. La información acerca del log se muestra al usuario a través de la interfaz web.

Capítulo 6

Evaluación

En este capítulo se analizan los resultados obtenidos aplicando las técnicas especificadas en las diferentes secciones de la tesis, desde el rendimiento y escalabilidad del algoritmo de dinámica molecular en comparación a otros simuladores que existen en la actualidad, hasta un caso de uso utilizando la herramienta de gestión de dinámica molecular y recursos de servicios de cloud computing

6.1. Rendimiento del algoritmo de dinámica molecular

Esta sección analiza en profundidad la escalabilidad y los beneficios que ofrece la solución propuesta en el Capítulo 4. Este análisis trata de mostrar el rendimiento del algoritmo de simulación comparándolo con el simulador NAMD, ampliamente utilizado en el campo de dinámica molecular. La elección de NAMD se debe a que provee una gran variedad de opciones, particularmente la capacidad de desactivar las fuerzas electrostáticas de largo alcance, las cuales no resuelve el algoritmo propuesto. Desactivando estas fuerzas es posible llevar a cabo una comparación equitativa entre ambas soluciones. NAMD permite simular los bloques de agua como si se tratara de un solido-rígido, por lo que se debe deshabilitar esta opción para que se calculen las fuerzas de todas las moléculas del sistema a nivel atómico. Tanto el algoritmo propuesto como NAMD calculan las fuerzas de enlace y las fuerzas electrostáticas de corto alcance en todos los pasos de la simulación. Además, con ambos sistemas se

obtiene la misma precisión en todos los escenarios propuestos. Obsérvese que, debido a la ausencia de las fuerzas de largo alcance, no es posible medir la validación de la estructura molecular a largo plazo. Por esta razón se limita el número de pasos a 100 en todas las comparaciones realizadas.

Para la simulación se realizan dos tipos de comparativas. Por un lado, se simulan únicamente moléculas de agua, ya que permiten obtener una gran variedad en el número de átomos. También se varía el número de ECs a utilizar para medir el rendimiento, el speedup y el uso de memoria de la GPU. Por otro lado, se utilizan sistemas moleculares conocidos para comparar el rendimiento y el uso de memoria variando el tamaño del sistema y el tipo de GPUs en entornos reales. En todas las comparaciones se distribuye el sistema molecular utilizando un sistema de partición binario en tres dimensiones, al igual que ocurre en trabajos previos con algoritmos similares [42], y siempre teniendo en cuenta las condiciones periódicas de frontera. Como se ha mencionado, se limitan todas las simulaciones a 100 pasos, lo que permite obtener medidas de tiempo reales sin gastar demasiados recursos. Se actualizan los átomos y los enlaces entre los distintos ECs cada 10 pasos. Incrementar este rango produce inestabilidades en las simulaciones, y decrementarlo no supone una diferencia significativa en la precisión del resultado.

Como entornos de pruebas se ha dispuesto tanto de un clúster en el centro de supercomputación Barcelona Supercomputing Center (BSC), así como de dos clústeres formados por diversas instancias con GPU en el servicio de cloud computing Amazon Web Services (AWS). En el caso del BSC, el clúster, llamado Minotauro, consta de múltiples nodos donde cada uno dispone de dos procesadores Intel E5-2649 a 2.53 GHz (6 núcleos), 24 gigabytes de RAM, dos GPUs Nvidia M2090 6GB GDDR5, y dos tarjetas de red QDR Infiniband (40 Gbit). Dadas estas características, la mayor capacidad computacional la ofrecen las GPUs. Por otro lado, se han creado otros dos clústeres en AWS compuestos por múltiples instancias o máquinas virtuales con

GPU. En concreto en AWS se utilizan las instancias p2.xlarge y p3.xlarge, que disponen de GPUs Nvidia Tesla K80 y Nvidia Tesla V100 respectivamente. A pesar de que estas tarjetas gráficas (K80 y V100) están compuestas por dos GPUs cada una de ellas, la virtualización de AWS solamente provee de una por instancia. Debido a esto, cada instancia se ejecuta con una sola GPU, disponiendo de 12 gigabytes de RAM en el caso de la K80 y de 16 gigabytes en la V100.

En un primer caso se realiza una comparación del tiempo de ejecución y de la escalabilidad del sistema en ambos entornos de supercomputación (BSC y AWS). La figura 6.1 compara el rendimiento del algoritmo de simulación utilizando un total de 100 pasos con una configuración de 10 pasos para actualizar los datos entre ECs y un total de 8 millones de átomos formando moléculas de agua. La figura también muestra los intervalos de confianza obtenidos al analizar de manera individual el tiempo de cada EC. Como se espera, todos los ECs utilizan un tiempo de ejecución similar ya que el número de átomos y su carga computacional se encuentra bastante balanceada entre los distintos ECs. Esto es debido a que el sistema molecular está compuesto íntegramente por moléculas de agua, y se localizan en el espacio de manera estable en términos de energía, por lo que todas las regiones del sistema contienen una carga de átomos y enlaces similar. En el caso de un sistema molecular más complejo, con distintas densidades de átomos y de enlaces, el particionado no se realiza de una forma tan balanceada, pero, tal como se verá a lo largo de la sección, no afecta significativamente a la escalabilidad del simulador. Debido al uso de las Nvidia K80, AWS ofrece un mayor rendimiento que el clúster Minotauro en el BSC que usa tarjetas M2090, las cuales disponen de un rendimiento inferior. A medida que el número de ECs se incrementa, la diferencia en el rendimiento decrece debido a que la carga de trabajo individual de cada EC se vuelve menor, y el sistema escala de la forma prevista. Esto da una idea del comportamiento de la ejecución en paralelo en términos de escalabilidad. Además del rendimiento también se ha medido el tiempo

de transferencia de datos. La Figura 6.2 muestra que estas transferencias requieren en AWS un menor tiempo que en el BSC. Cabe destacar que las instancias de AWS pueden estar localizadas en el mismo nodo físico, lo que reduce drásticamente los tiempos de transmisión.

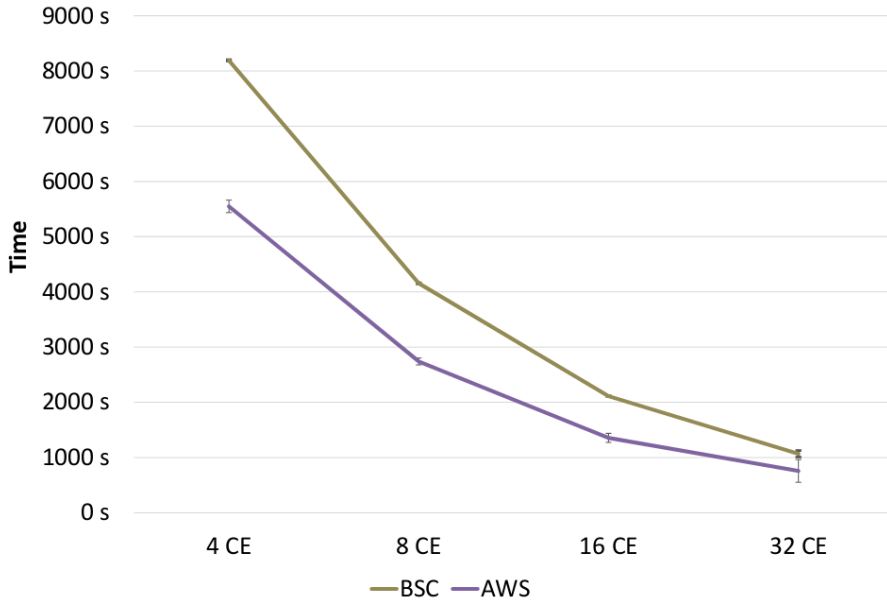


Figura 6.1: *Tiempo de simulación utilizando 8 millones de átomos en moléculas de agua de forma distribuida. Comparación de 100 pasos en el clúster de BSC (Nvidia M2090) y de AWS (Nvidia K80).*

Una vez analizados los resultados en diferentes entornos de ejecución de forma preliminar, se realiza una comparación completa entre el algoritmo propuesto y NAMD, balanceando un sistema compuesto por moléculas de agua (8 millones de átomos en total) en el entorno de AWS formado por instancias con K80. La figura 6.3(a) muestra el rendimiento en ambos simuladores. NAMD ofrece un rendimiento mas alto cuando se utilizan pocos ECs. Utilizando solo moléculas de agua el coste de las fuerzas de enlace es bastante bajo, y el cálculo de estas fuerzas en NAMD uti-

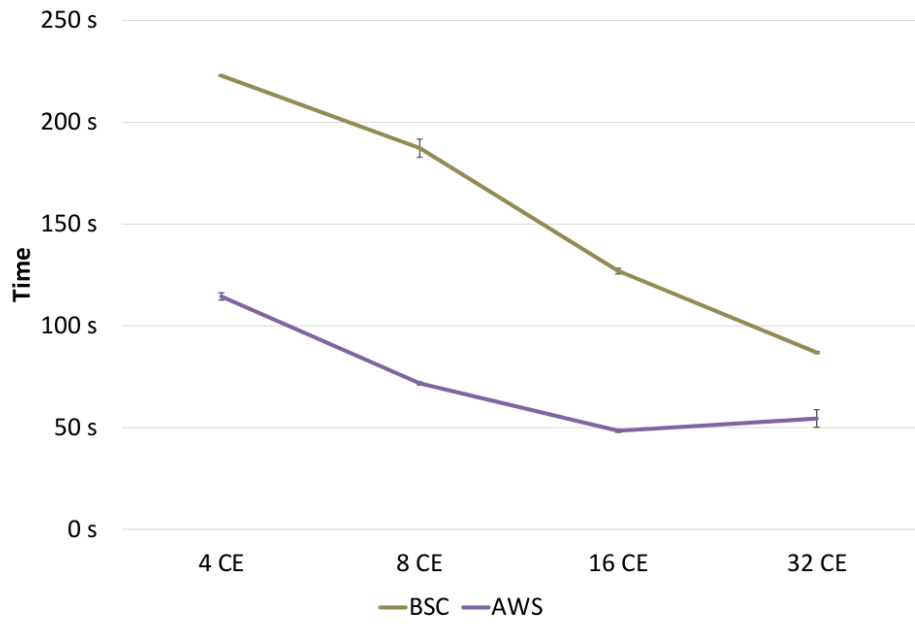
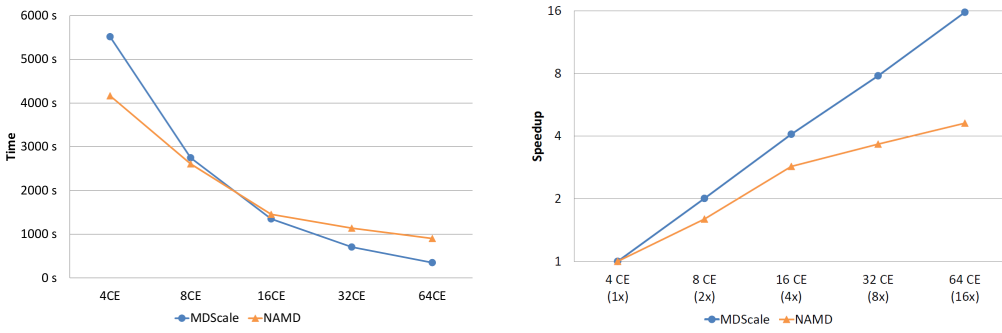


Figura 6.2: *Tiempo de actualización de los datos de un sistema molecular compuesto por 8 millones de átomos de forma distribuida formado solo por moléculas de agua. Comparación de 100 pasos en el clúster de BSC (Nvidia M2090) y de AWS (Nvidia K80).*

lizando CPU es mucho más eficiente. Por otro lado, a medida que el número de ECs aumenta, el rendimiento del algoritmo propuesto sobrepasa el de NAMD debido a su mejor escalabilidad, como se muestra en la figura 6.3(b). El speedup se ha calcula con respecto a la simulación con 4 ECs, ya que el sistema molecular no entra en un menor número de ECs debido a la limitación de memoria. Al incrementar el número de ECs a 64 (x16), el speedup obtenido por el algoritmo es de 15.76, lo que significa que su escalabilidad es prácticamente óptima.

Como se puede apreciar en la Figura 6.4, el tiempo requerido para actualizar los datos entre ECs solo utiliza un pequeño porcentaje del tiempo total de simulación, siendo este entre un 1 y un 6 %. Este porcentaje aumenta a medida que se incrementa



(a) (b)

Figura 6.3: *Resultados obtenidos simulando 100 pasos con un bloque de moléculas de agua compuesto de 8 millones de átomos con el algoritmo propuesto y NAMD en un entorno de pruebas de AWS con GPUs Nvidia K80. (a) muestra el tiempo de ejecución. (b) detalla el speedup, donde la escala es logarítmica en base 2 para que se pueda observar correctamente el speedup lineal del algoritmo. El speedup se calcula utilizando como referencia el menor número de ECs (4 en este caso) que se puede utilizan para simular el sistema de 8 millones de átomos.*

el número de ECs, debido al mayor número de vecinos con los que cada uno debe compartir su información. Aun así, el tamaño de la memoria utilizada por la zona de interfaz de cada EC no excede el 3,25 % del total. Por lo tanto, más del 95 % del tiempo de simulación se emplea únicamente en realizar los cálculos de forma paralela, garantizado una alta escalabilidad.

La siguiente comparación consiste en simular moléculas de agua con diferente número de átomos por EC, siempre asegurando un correcto balanceo de carga entre ECs. En concreto se simulan 1, 2 y 4 millones de átomos por cada EC, ejecutando estas pruebas con 2, 4, 8, 16 y 32 GPUs en el entorno de AWS con tarjetas Nvidia K80. La Figura 6.5 muestra el uso de memoria por EC para las moléculas simuladas. El uso de memoria de GPU aumenta ligeramente en el paso de 2 a 4 ECs, donde se estabiliza

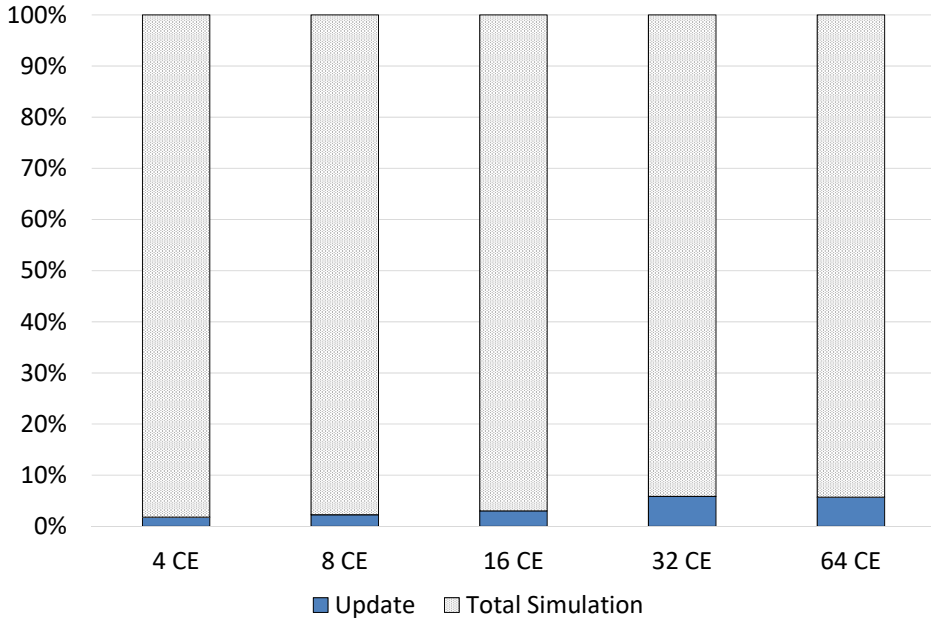


Figura 6.4: *Porcentaje de tiempo necesario para actualizar los átomos y los enlaces en relación a la simulación del sistema de moléculas de agua compuesto por 8 millones de átomos en un entorno de AWS formado por GPUs Nvidia K80.*

asegurando la escalabilidad de memoria. La Figura 6.6 muestra el rendimiento de la simulación (ps/día) obtenido con la solución propuesta y NAMD, utilizando los mismos sistemas moleculares. Esta unidad de medida se utiliza normalmente debido a que es útil para saber cuantos días de simulación son necesarios para completar, por ejemplo, el proceso de síntesis de una proteína. En todos los casos se simula el mismo número de átomos en los dos simuladores. En la comparativa se aprecia que incrementar el número de elementos implica un incremento lineal del número de átomos a simular (por ejemplo, 128 millones de átomos se distribuyen en 4 millones por EC al utilizar 32 GPUs). Al doblar el número de átomos por EC, el tiempo de simulación es inferior al doble, sacando provecho del paralelismo masivo que ofrece

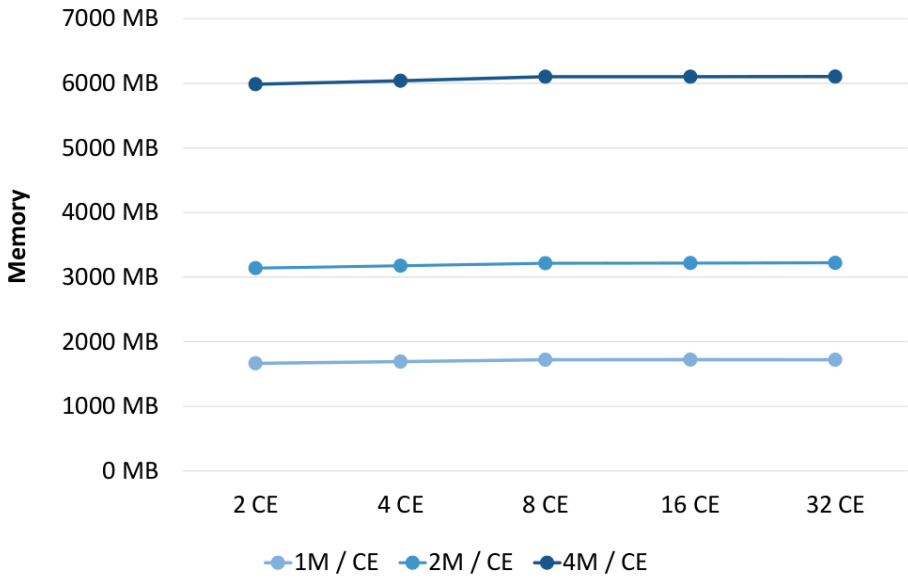


Figura 6.5: *Uso de memoria del algoritmo propuesto por cada EC para el conjunto de moléculas de agua simulado en la Figura 6.6.*

la GPU. Al igual que en el caso anterior, NAMD presenta un rendimiento superior al utilizar una cantidad pequeña de moléculas de agua por EC, debido al reducido número de fuerzas de enlace. A medida que el número de átomos y de ECs aumenta, la capacidad de NAMD decrece mientras que el método propuesto se mantiene estable. Esto permite tratar con simulaciones de dinámica molecular masivas formadas por un gran número de átomos, simplemente añadiendo GPUs a la infraestructura.

Para mostrar la escalabilidad de la solución propuesta, se ha simulado un sistema de moléculas de agua con 1 billón de átomos, distribuidos de forma balanceada en 128 GPUs Nvida K80. La Tabla 6.1 muestra el rendimiento obtenido en el entorno de pruebas descrito. Estas medidas se corresponden con las vistas en las figuras 6.5 y 6.6. La simulación utiliza más de 10 gigabytes de RAM por GPU, de los 12

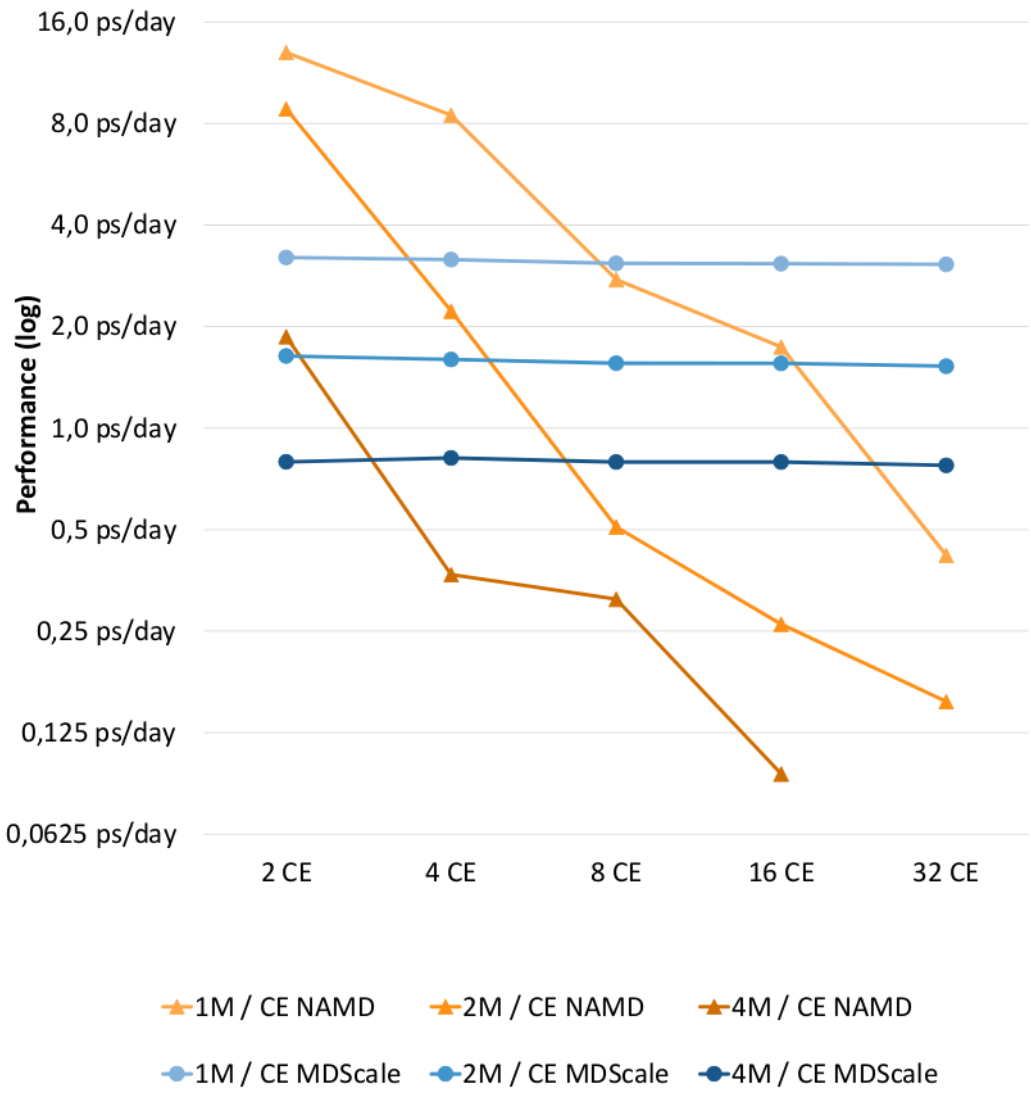


Figura 6.6: *Rendimiento en comparación con NAMD en escala logarítmica, incrementando el número de GPUs y de átomos por EC para simular moléculas de agua con decenas de millones de átomos.*

gigabytes de los que dispone cada una de las K80. El porcentaje de tiempo utilizado en intercambiar e identificar los átomos entre ECs no es significativo y se mantiene

estable con respecto al número de átomos y ECs implicados en la simulación.

Tabla 6.1: *Medidas obtenidas al simular un sistema de moléculas de agua en 128 GPUs Nvidia Tesla K80.*

Rendimiento	Actualización y transferencia	Uso de memoria
0.43 ps/día	6.15 %	10.42 GB/EC

El factor limitante para la escalabilidad en este caso es la memoria disponible en cada GPU. Por ello se ha preparado otro entorno de pruebas, compuesto por GPUs Nvidia Tesla V100, para aumentar la memoria de GPU disponible. Esto permite simular la misma cantidad de átomos (1 billón) utilizando solo 96 tarjetas V100. Además, en las mismas 96 tarjetas V100 se puede simular un total de 1.2 billones de átomos. Como se puede apreciar en la tabla 6.2, el rendimiento obtenido es significativamente superior en comparación con la Tabla 6.1 debido a las mejoras de la arquitectura de la GPU.

Tabla 6.2: *Medidas obtenidas al simular un sistema de moléculas de agua en 96 GPUs Nvidia Tesla V100.*

Núm. Átomos	Rendimiento	Actualizaciones	Uso de memoria
1 Billón	4.8 ps/día	5.78 %	12.32 GB/EC
1.2 Billones	2.92 ps/día	5.23 %	15.63 GB/EC

Una vez se ha demostrado la escalabilidad de la propuesta utilizando sistemas formados por moléculas de agua, se realizan los mismos tipos de pruebas con estructuras biológicas macromoleculares, obtenidas del *Protein Data Bank (PDB)* [10]. Primero se llevan a cabo comparaciones del método propuesto y NAMD utilizando un sistema molecular de tamaño medio. En concretos se simulan 100 paso con fuerzas de enlace y electrostáticas de corto alcance del sistema biomolecular 4V4L [64],

replicado 12 veces (aproximadamente 7.5 millones de átomos), el cual se puede ver en la Figura 6.7. Segundo, se realizan pruebas en un escenario más complejo, simulando un sistema molecular grande como es 3J3Q [44], el cual representa la cápsida de virus HIV-1 (aproximadamente 66 millones de átomos), representado en la Figura 6.8. Cabe destacar que la cápsida del virus HIV-1 ya se ha simulado utilizando NAMD y CHARMM en el supercomputador TITAN, con 3880 nodos Cray-XK equipados con GPU [65]. En ambos sistemas moleculares, el particionado binario genera particiones que se encuentran muy lejos de estar perfectamente balanceadas ya que existen áreas con una mayor densidad de átomos y un mayor número de enlaces.

Ambos sistemas moleculares se distribuyen entre distinto número de ECs, con

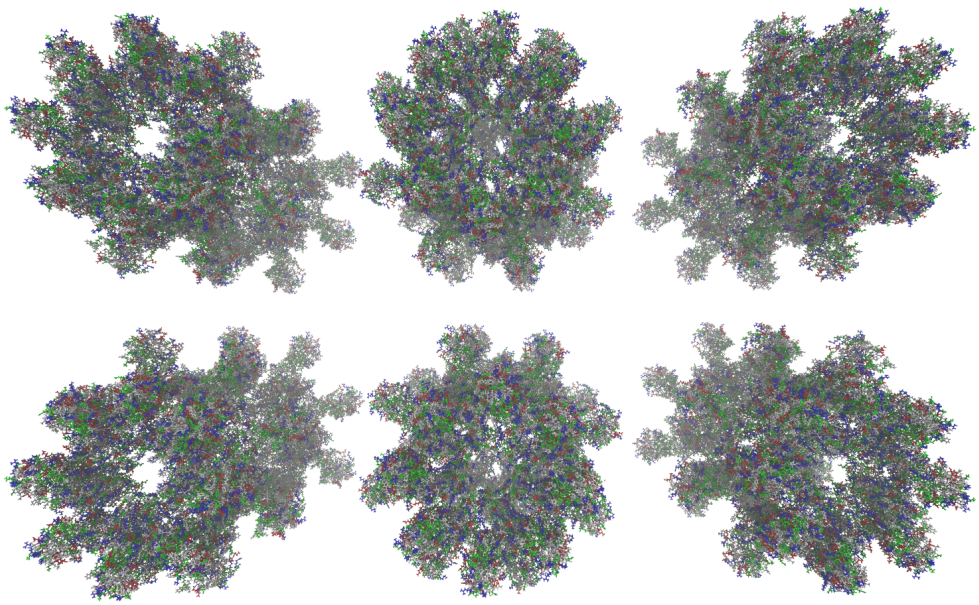


Figura 6.7: *12 estructuras moleculares del sistema 4V4L, compuesto por aproximadamente 7.5 millones de átomos, incluyendo las moléculas de agua que lo envuelven.*

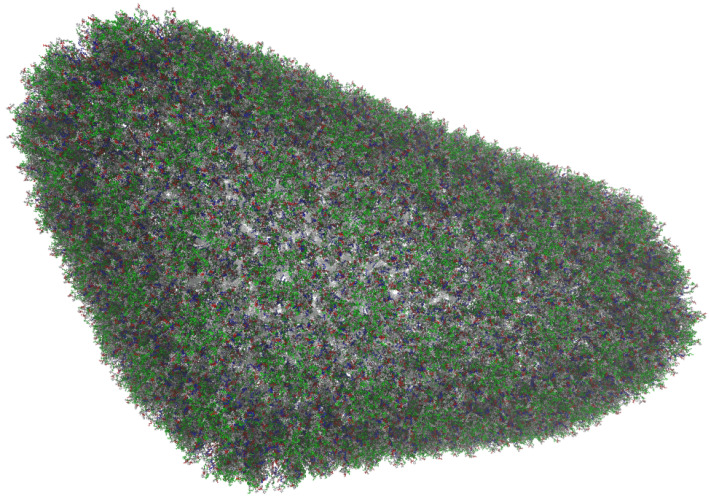


Figura 6.8: *Sistema molecular 3J3Q, cápsida del virus HIV-1 compuesta por cerca de 66 millones de átomos, incluyendo el agua que lo rodea.*

sus respectivas GPUs: 4, 8, 16 y 32 para 4V4L y 16 y 32 para 3J3Q, debido a los requisitos de memoria de cada uno de ellos. Como muestra la Figura 6.9, el rendimiento del método propuesto es superior al de NAMD en todos los casos, a diferencia de lo que ocurre en la Figura 6.3(a). Esto es debido a que, al ejecutar un sistema molecular complejo, NAMD utiliza una gran cantidad de tiempo para llevar a cabo un balanceo de carga cuando tiene un gran número de parches. Para los sistemas 12x4V4L y 3J3Q el speedup obtenido por el algoritmo propuesto es casi lineal: 1.98 y 1.88 respectivamente con 32 GPUs, comparando con la ejecución en 16 GPUs (x2). La diferencia de rendimiento entre las dos pruebas se debe a gran desequilibrio que existe en el sistema molecular 3J3Q. Además, a medida que el tamaño y la complejidad del sistema molecular crecen, la diferencia de rendimiento con respecto a NAMD se hace mucho más notable.

La Figura 6.10 muestra para cada sistema molecular la cantidad de memoria utilizada en cada EC, así como la memoria requerida por la zona de interfaz. Por

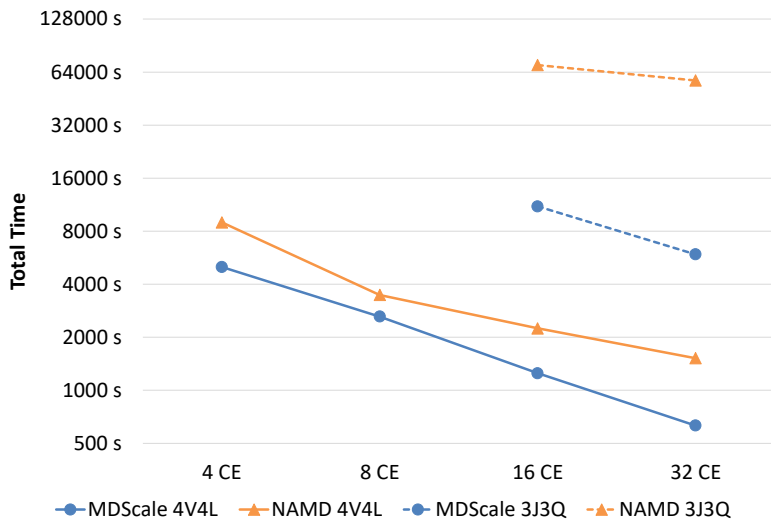


Figura 6.9: *Comparación del tiempo de simulación de la solución propuesta y de NAMD utilizando los sistemas moleculares 12x4V4L y 3J3Q en el entorno de pruebas de AWS con GPUs Nvidia K80. El rendimiento del método propuesto es superior en todos los casos (4, 8, 16 y 32 GPUS) en ambos escenarios.*

ejemplo, el sistema 3J3Q dividido entre 16 GPUs Nvidia K80 utiliza casi toda la memoria disponible en cada GPU. Cabe destacar que los sistemas moleculares se dividen en base a la posición de los átomos de cada una de las moléculas, no de la complejidad de esta. Esto produce que haya ECs con una densidad de átomos mayor y enlaces más complejos y, por lo tanto, que necesiten mas memoria para almacenarlos. Esta variación se representa en la figura mediante líneas que muestran la desviación típica del uso de memoria entre los distintos ECs. Aun con este desequilibrio de carga, como se puede ver el speedup no se ve afectado significativamente. La cantidad total de memoria ocupada decrece casi de forma lineal al incrementar el número de ECs.

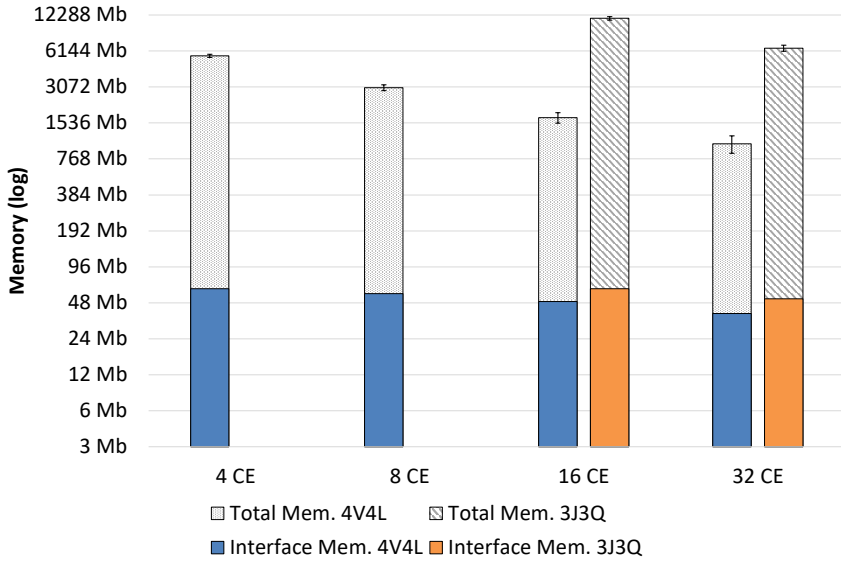


Figura 6.10: *Uso de memoria por EC en particiones 3D binarias distribuidas de forma equitativa entre los diferentes ECs de la solución propuesta utilizando los sistemas moleculares 12x4V4L y 3J3Q en el entorno de pruebas de AWS con GPUs Nvidia K80. El gráfico se encuentra en escala logarítmica para que se aprecie la evolución del uso de memoria. Nótese que la zona de interfaz utiliza una pequeña cantidad de memoria en proporción al tamaño total de la simulación.*

Como cabe esperar, la memoria utilizada por las celdas de la zona de interfaz no decrece al mismo ritmo ya que el tamaño de esta no depende del número de átomos que se simulan, sino del tamaño de las celdas, del radio de corte y del número de vecinos, y por lo tanto del número de ECs. En cualquier caso, se ha de tener en cuenta que el tamaño de la interfaz es muy pequeño (la escala de la figura es logarítmica) en comparación con el espacio de memoria necesario para almacenar los átomos que corresponden a cada EC.

Finalmente se ha diseñado un gran sistema molecular complejo para simular. En

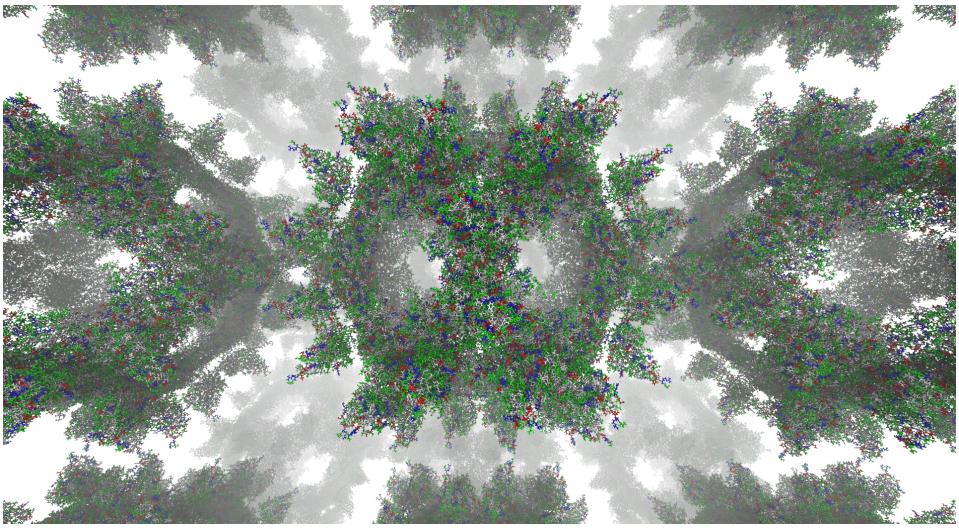


Figura 6.11: *Gigamolécula compuesta por 96 virus 4UDF, replicada para obtener 1000 millones de átomos incluyendo el agua que lo rodea.*

concreto se ha replicado 96 veces el virus 4UDF (*Human Parechovirus Neutralization by Human Monocloal Antibodies*) [54], compuesto por aproximadamente 10,4 millones de átomos contando el agua que lo envuelve), para obtener una gigamolécula de 1 billón de átomos (ver Figura 6.11). La Tabla 6.3 muestra el rendimiento obtenido en una distribución balanceada utilizando un entorno de pruebas con 96 Nvidia V100 (una para cada 4UDF). La simulación utiliza más de 13 GB de memoria de los 16 disponibles en cada V100. Como se espera, el rendimiento decae ligeramente comparado con la simulación de moléculas de agua con el mismo número de átomos y GPUs (ver Tabla 6.2), debido a la elevada complejidad del sistema molecular. Sin embargo, se demuestra que el método propuesto puede trabajar con sistemas moleculares masivos añadiendo tantas GPUs como sea necesario para cubrir el tamaño del sistema molecular y obtener un buen rendimiento

Como se ha podido observar en la figura 6.3, y tal como se ha descrito en la sección 4.1.3, disponer de un número bajo de GPUs en un sistema molecular relativamente

Tabla 6.3: *Medidas obtenidas al simular 1 billón de átomos con 96 virus 4UDF, replicados en 96 GPUs Nvidia Tesla V100.*

Rendimiento	Actualización	Uso de memoria
4.02 ps/día	7.10 %	13.40 GB/EC

pequeño reduce el rendimiento de la simulación considerablemente con respecto a otras soluciones existentes. Por ello, y tras implementar la solución propuesta en 4.1.3, se realizaron nuevas pruebas de la ejecución del simulador. Esta vez se utilizó un entorno de supercomputación como es el clúster Minotauro del BSC (Barcelona Supercomputing Center), dotado de 39 nodos con dos tarjetas gráficas Nvidia Tesla K80 cada uno, lo que se traduce en un total de 156 GPUs si se tiene en cuenta que cada K80 consta de 2 GPUs [5]. En este entorno se realizaron las mismas pruebas que

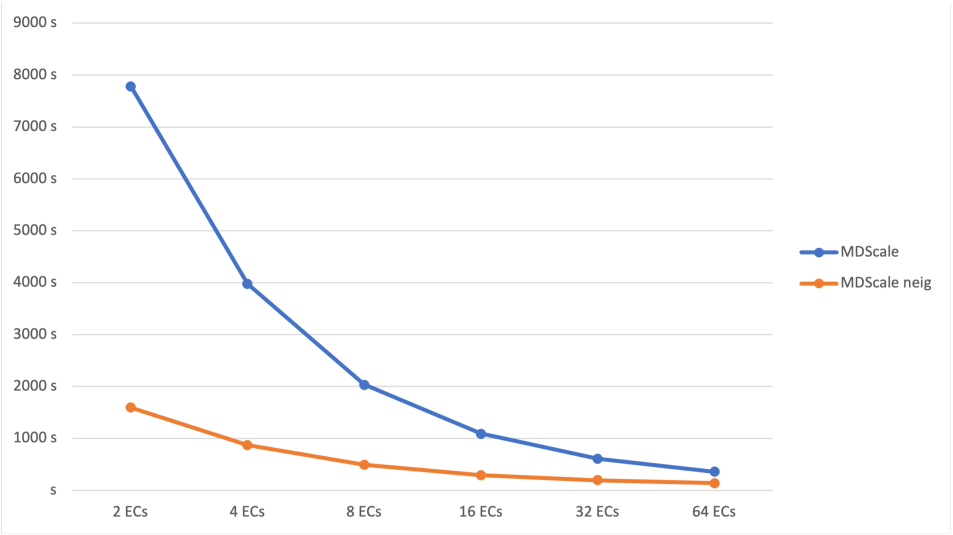


Figura 6.12: *Comparación de resultados de ambas variantes del algoritmo de simulación en el clúster Minotauto utilizando un sistema de 8 millones de átomos formado por moléculas de agua.*

las mostradas en la figura 6.3, utilizando el mismo sistema molecular de 8 millones de átomos compuesto por moléculas de agua, en este caso aplicando la solución con la lista de vecinos propuesta. Como se puede ver en la figura 6.12, utilizando la aproximación de las fuerzas de Van der Walls propuesta, el rendimiento sufre una gran mejora con respecto al algoritmo original. Sin embargo, esta solución aumenta drásticamente el uso de memoria de GPU. Además, esta mejora se produce a coste de reducir la precisión del resultado obtenido, por lo que no es una solución adecuada para simular sistemas moleculares masivos, aunque si permite suplir las carencias del algoritmo original con respecto a los sistemas moleculares mas pequeños o un número reducido de ECs.

6.2. Evaluación en entornos de Cloud Computing

Esta sección presenta un caso de estudio desplegando la solución propuesta en el capitulo 5 en un entorno local y utilizando Amazon Web Services (AWS SDK [4]) como plataforma de ejecución.

Para estas pruebas se ha elegido el sistema molecular 4V4L como ejemplo de sistema complejo, descargado desde PDB (ver figura 6.13). El sistema 4V4L, obtenido de PDB [64] se compone de cerca de 128.000 átomos, alcanzando los 500.000 al final del proceso. El desarrollo del procesado de los datos para construir un sistema molecular que pueda ser simulado supone un coste de RAM de alrededor de 15 gigabytes. En este caso, esta cantidad de memoria se puede alcanzar fácilmente con una instancia de AWS Elastic Compute Cloud (EC2) solo con CPU, como la t2.2xlarge [2]. Los sistemas moleculares más grandes, como es 3J3Q [44], pueden alcanzar los 65 millones de átomos, necesitando hasta 3 Terabytes de RAM, lo que supondría utilizar centros de supercomputación o entornos de cloud computing. Por ejemplo, la instancia EC2 de AWS x1e.32xlarge [3] dispone de 3.9 Terabytes de RAM, que es

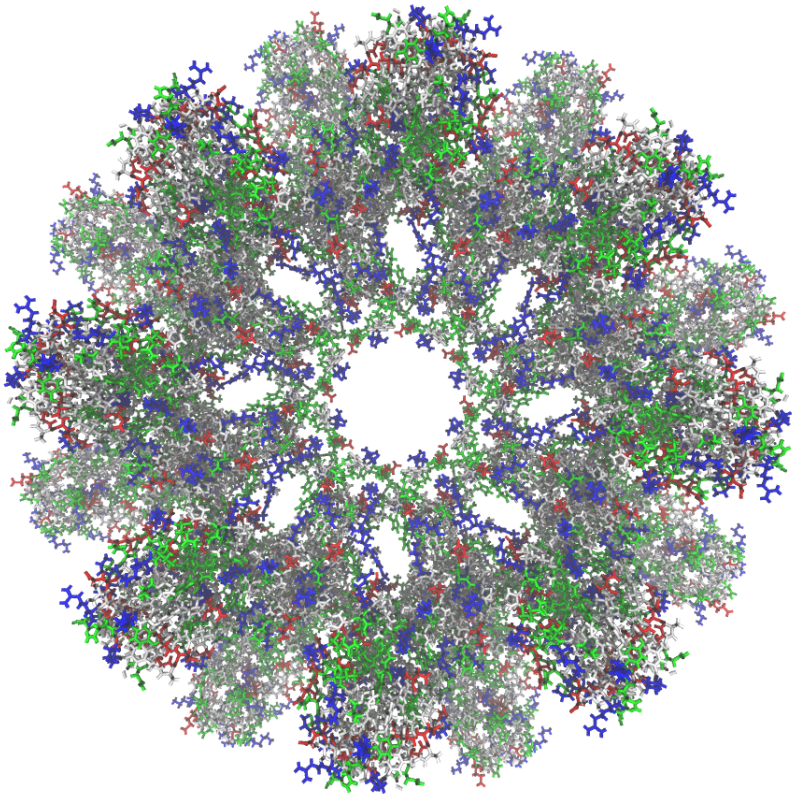


Figura 6.13: *Estructura molecular 4V4L obtenida del Protein Data Bank. Esta imagen ha sido generada utilizando el software VMD.*

suficiente para tratar este sistema molecular. También es posible utilizar instancias t2.2xlarge con 3 terabytes de almacenamiento en disco duro como memoria de intercambio. Esta última configuración es mucho más barata, pero supone un tiempo de ejecución mayor.

Para los procesos de equilibrio térmico y simulación es muy recomendable utilizar instancias EC2 que proveen de GPUs, debido a la alta carga computacional que

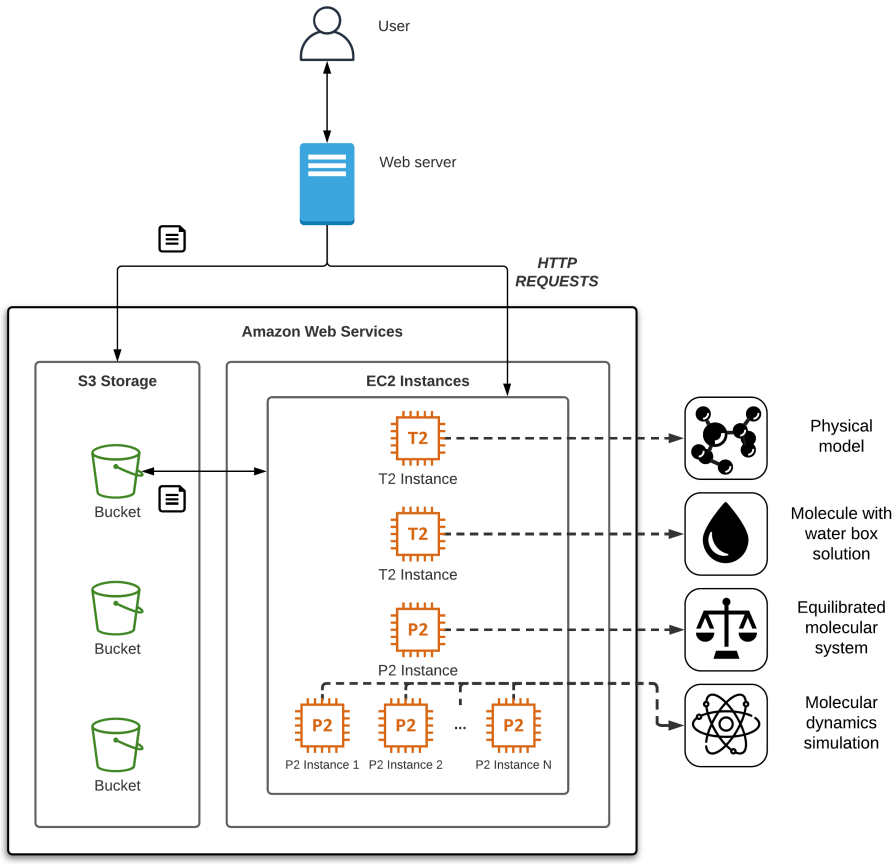


Figura 6.14: *Diagrama detallado de la arquitectura del entorno de pruebas utilizado, basado en la Figura 5.1 y utilizando AWS como entorno de cloud computing.*

suponen estos procesos. Por ejemplo, las instancias EC2 de AWS p2.16xlarge [1] proveen de 16 GPUs Nvidia Tesla K80. Sin embargo, la simulación de grandes sistemas moleculares puede requerir el uso de muchas más GPUs. En un entorno normal puede ser necesario configurar adecuadamente la comunicación de red entre las distintas instancias para disponer de un entorno Multi-GPU funcional, además de instalar y gestionar el software necesario. Esta herramienta basada en una interfaz web permite inicializar, desplegar instancias con el número necesario de GPUs, y establecer la

comunicación entre ellas sin que sea necesaria la intervención del usuario.

Como servicio de almacenamiento cloud se utiliza AWS Simple Cloud Storage Service (S3), que permite una gran escalabilidad y disponibilidad de los datos. En la nomenclatura de AWS, un bucket es similar a un directorio en el servicio del almacenamiento, donde se pueden almacenar datos junto con sus metadatos.

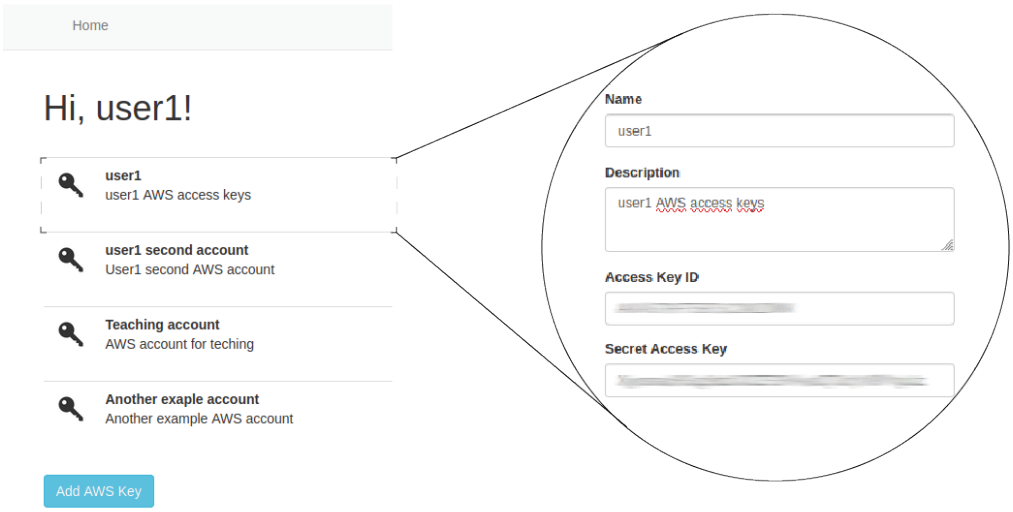
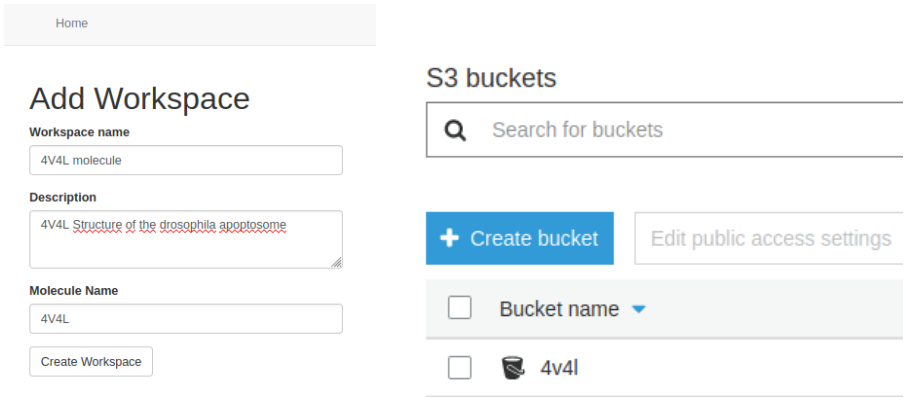


Figura 6.15: *Relación de las claves de acceso a AWS. A la izquierda se muestran las distintas claves almacenadas por un usuario. A la derecha se detalla un ejemplo de una de las claves de acceso a AWS.*

La Figura 6.14 muestra la arquitectura completa del entorno de pruebas ejecutado en una infraestructura de AWS, que es necesaria para el despliegue y uso de la aplicación. En los siguientes párrafos se describirá el proceso completo, desde la gestión de usuarios a la ejecución de la simulación de dinámica molecular, pasando a través de todo el flujo de trabajo utilizando la herramienta propuesta. Se ha de tener en cuenta que no es necesario realizar todos los pasos del proceso, sino que es posible cargar los resultados obtenidos de cualquiera de ellos, además de re-ejecutar cada uno de los pasos sin tener que iniciar de nuevo el proceso completo.

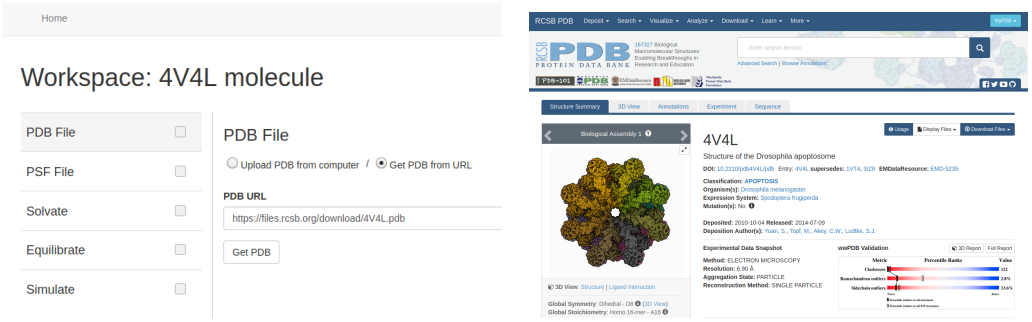


(a) (b)

Figura 6.16: *Creación de un espacio de trabajo para un sistema molecular. (a) muestra como se crea el espacio de trabajo en la herramienta, y (b) la representación de este en un bucket del servicio S3 de AWS.*

El primer paso es registrar un acceso a la plataforma cloud. Los datos de los usuarios están cifrados, y se almacenan en una base de datos SQL en el servidor web. En concreto, a los usuarios se les pregunta por las credenciales para acceder a la plataforma cloud, siendo en este caso AWS. Los usuarios deben disponer de una cuenta de AWS, y haber generado las claves de accesos externas en el servicio Identity and Access Management (IAM) de AWS. El usuario de AWS debe tener permisos para gestionar remotamente instancias EC2 y almacenar datos en S3. Las claves de acceso obtenidas constan de dos partes: un ID de la clave de acceso y una clave de acceso secreta. Cada una de las claves se almacenan asignadas al usuario correspondiente, como se ve en la Figura 6.15. Una vez el usuario ha iniciado sesión en la herramienta, puede empezar a gestionar el entorno de trabajo

En este punto, cada usuario debe configurar un espacio de trabajo asociado a su cuenta, el cual representa un bucket en el servicio de almacenamiento S3 de AWS. La herramienta está planteada para crear un espacio de trabajo por cada sistema



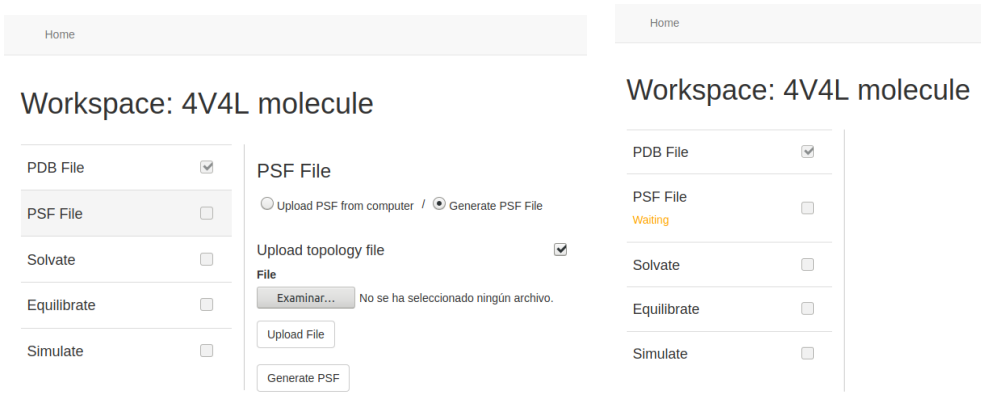
(a) (b)

Figura 6.17: *Proceso de carga del sistema molecular. En (a) se muestra como se provee de un enlace de PDB a la herramienta. (b) Muestra el sistema molecular 4V4L en PDB.*

molecular a simular de forma que el bucket almacena todos los datos relacionados con este sistema. Las instancias EC2 acceden a este bucket para obtener los datos y subir los resultados obtenidos. La Figura 6.16 muestra un espacio de trabajo en la herramienta, junto con su representación correspondiente como bucket de S3.

El siguiente paso consiste en seleccionar los datos del sistema molecular con el que se va a trabajar. Para simplificar este proceso, la herramienta ofrece dos opciones: elegir un archivo local con dichos datos o proveer de un enlace para obtenerlo directamente de PDB, como se ve en la Figura 6.17.

Una vez se han subido los datos moleculares, los usuarios pueden comenzar con el procesado de los datos dependiendo de sus necesidades. La primera etapa que se puede ejecutar consiste en relacionar cada átomo con la molécula a la que pertenece, asignando sus enlaces y sus cargas eléctricas. El usuario puede subir su propio archivo con esta topología establecida. En el caso de no disponer de él, es posible llevar a cabo los cálculos en una instancia EC2 t2 de AWS utilizando el software VMD [32], que permite comparar los datos moleculares utilizando distintas topologías. La



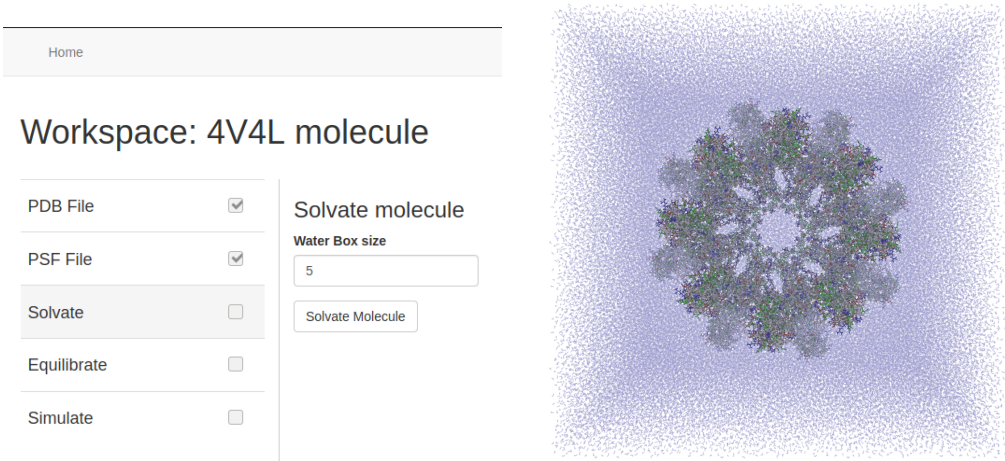
(a) (b)

Figura 6.18: *Etapa de creación de PSF. (a) muestra en detalle la herramienta con sus opciones de configuración. El usuario puede elegir entre utilizar los ficheros de topología que se incluyen por defecto o subir su propio fichero. En (b) se muestra la interfaz de la herramienta cuando la ejecución de esta etapa se encuentra en proceso.*

herramienta se encarga de realizar todo lo relaciona con la instancia. Los usuarios solo deben configurar e iniciar la ejecución desde la interfaz web. El estado de cada etapa se puede consultar en la propia aplicación, que lo obtiene directamente de las instancias EC2. La Figura 6.18 describe como se configura la herramienta para crear un archivo PSF (Protein Structure File). Los archivos PSF contienen toda la información molecular que no está incluida en los datos obtenidos previamente de PDB, como los enlaces o las cargas de los átomos. Estos datos se deben incluir para poder llevar a cabo la simulación de dinámica molecular.

Como se describe en la sección 4.3, un sistema molecular no puede encontrarse en el vacío, sino que debe estar en una solución de agua. La segunda etapa consistirá, por lo tanto, en crear una estructura molecular que forme un sistema cúbico, compuesto por los datos originales rodeados de múltiples moléculas de agua (ver Figura 6.19).

Aunque el tamaño de la estructura depende de la magnitud del sistema original, el usuario puede elegir el margen de moléculas de agua. El software VMD también puede llevar a cabo este proceso en una instancia EC2 t2 de la infraestructura cloud. Al igual que ocurría con la etapa anterior, el usuario puede seleccionar ciertos parámetros, pero es la herramienta la que gestiona todo el proceso.



(a) (b)

Figura 6.19: *Etapa para añadir una solución de agua. (a) muestra una captura donde el usuario puede especificar el tamaño del bloque de agua. En (b), el sistema molecular 4V4L se encuentra dentro del bloque de la solución.*

Introducir las moléculas de agua al sistema provoca que la simulación de dinámica molecular se vuelva inestable. Para resolver esto es necesario balancear el sistema molecular, utilizando por ejemplo NAMD, que provee de una opción para llevar a cabo esta tarea. Al igual que ocurría en etapas anteriores, la herramienta se encarga de gestionar todo el proceso, pero esta vez utilizando instancias EC2 p2 (equipadas con GPU) de AWS. El usuario solo debe configurar una serie de parámetros para realizar una pequeña simulación que lleva al sistema a un estado de baja energía.

Nuevamente la herramienta permite configurar múltiples opciones como la temperatura de simulación, el tamaño del paso de tiempo (en fs), el numero de pasos de simulación y el numero de pasos a los que se reinician los valores de velocidad. En la Figura 6.20 se puede ver un ejemplo de la configuración de esta etapa.

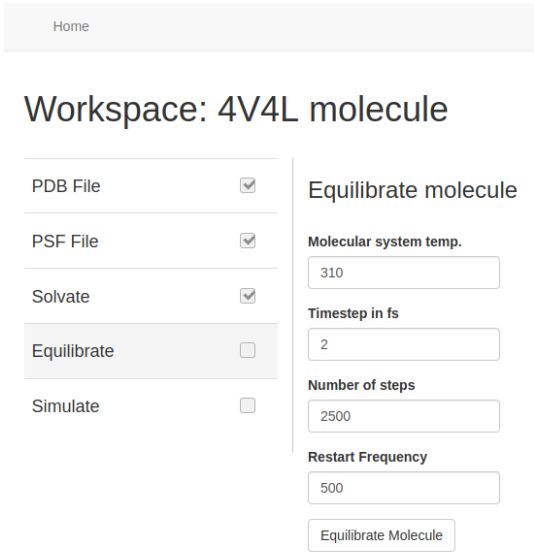


Figura 6.20: *Etapa de equilibrado del sistema molecular. El usuario puede establecer ciertos parámetros de este proceso.*

Una vez se ha pasado por todas las etapas anteriores, el sistema molecular está listo para la simulación. Finalmente, la herramienta permite ejecutar la simulación con diferentes simuladores de dinámica molecular. Se ha configurado una instancia EC2 de AWS, la cual se puede replicar en el entorno cloud conectando todas las copias entre si. En esta instancia se ha instalado NAMD y una implementación del algoritmo descrito en la sección 4.1. Aunque como trabajo futuro se propone incluir otros simuladores como GROMACS o AMBER, la herramienta permite al usuario, si este así lo desea, seleccionar su propia imagen de sistema (Amazon Machine Image en AWS) para realizar el proceso de simulación con su propio software. Para ello se

debe proveer de un script, necesario para ejecutar el simulador (ver Figura 6.21).

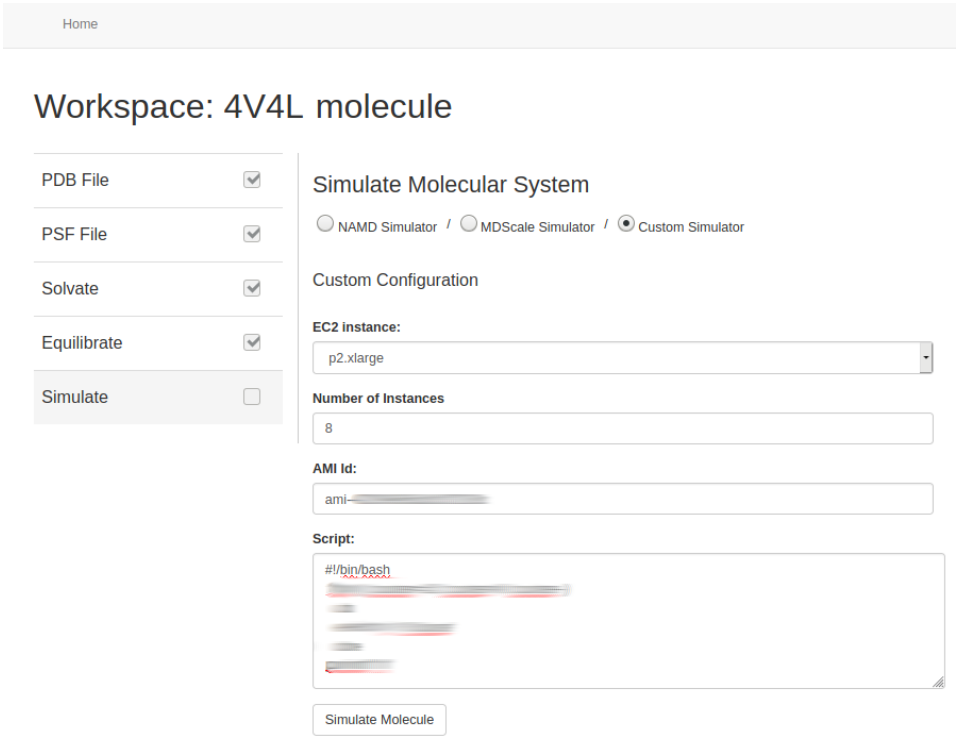
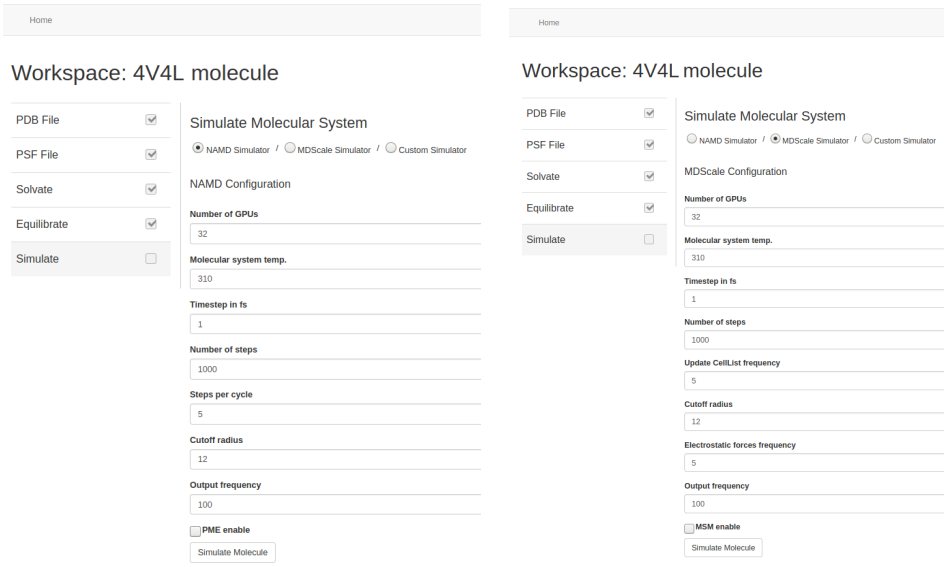


Figura 6.21: *Configuración de una simulación de dinámica molecular personalizada. El usuario puede proveer de su propia AMI, el tipo y el numero de instancias, y un script de ejecución para el simulador.*

El usuario puede establecer una serie de parámetros básicos como la temperatura, el paso de tiempo o la frecuencia a la que se escribe el resultado. Además, dependiendo del simulador de dinámica molecular utilizado, es posible establecer otros parámetros opcionales como el numero de pasos por ciclo, el radio de corte y la frecuencia para calcular las fuerzas electrostáticas de corto alcance, la frecuencia de actualización de CellList y la opción de utilizar algoritmos específicos como PME [16] o MSM [28] para calcular las fuerzas electrostáticas de largo alcance (ver Figura 6.22). Adicionalmente, el usuario puede elegir el numero de GPUs, en concreto la Nvidia Tesla K80 en AWS,

que se van a desplegar para la simulación. La herramienta automatiza la inicialización de las instancias EC2 p2 requeridas, basándose en el numero de GPUs solicitadas y gestiona la comunicación entre ellos. Al finalizar, los resultados obtenidos de la simulación se almacenan en el servicio de almacenamiento S3 de AWS.



(a) (b)

Figura 6.22: *Configuración de la simulación de dinámica molecular: (a) NAMD, (b) algoritmo propuesto. En ambos casos el usuario especifica los detalles de la simulación y elije el numero de GPUs que va a utilizar.*

Finalmente, se ha llevado a cabo todo el proceso de simulación utilizando el sistema molecular 4V4L, tanto en NAMD como en el algoritmo propuesto, para demostrar las posibilidades de la herramienta. La Tabla 6.4 muestra los tiempos de ejecución para cada etapa del proceso, desde el procesado inicial de los datos hasta la simulación del sistema molecular. El tiempo de despliegue y ejecución en el entorno cloud depende de los recursos necesarios y la disponibilidad de estos en un momento

dado, por lo que no se puede hacer demasiado énfasis en este punto. Gracias a la posibilidad de configurar la simulación, los usuarios pueden realizar test balanceados con distintos tipos de fuerzas a nivel atómico.

Tabla 6.4: *Tiempo de ejecución para cada etapa del proceso, diferenciando entre el tiempo de despliegue de las instancias EC2 y el tiempo de ejecución del simulador de dinámica molecular. La simulación se ha realizado utilizando 8 GPUs, mientras que el proceso de equilibrado de ha llevado a cabo con una sola GPU. El resto de las etapas no requieren del uso de GPUs.*

	Despliegue de instancias EC2	**Tiempo de ejecución**
Generación de PSF	85.6 s	47.9 s
Solución de agua	69.2 s	58.5 s
Equilibrado	153.2 s	3334.3 s
Simulación (NAMD)	304.4 s	295.7 s
Simulación (MDScale)	283.7 s	237.5 s

Capítulo 7

Conclusiones y líneas futuras

7.1. Conclusiones

Esta tesis presenta un algoritmo escalable Multi-GPU para el cálculo de las fuerzas de enlace y de corto alcance en simulación de dinámica molecular, el cual se realiza por completo en GPU. A lo largo de todos los capítulos de este documento se demuestra la eficiencia de la comunicación entre diferentes ECs y el reducido uso de memoria conseguido. El rendimiento del sistema implementado se ha comparado con el simulador NAMD, ampliamente utilizando en el campo, realizando las mismas simulaciones con múltiples sistemas moleculares tanto balanceados como desbalanceados, siendo estos los sistemas moleculares 4V4L y 4UDF replicados varias veces, el sistema 3J3Q y decenas de millones de átomos en bloques de moléculas de agua. El sistema de particionado implementado permite añadir tantas GPUs como sea necesario para cubrir el tamaño del sistema molecular sin perder rendimiento. Como prueba de su escalabilidad se ha llevado a cabo una exhaustiva evaluación, comprobando su rendimiento y uso de memoria con distinto numero de instancias con GPU de Amazon Web Services y en el clúster Minotauro del BSC. Esto ha permitido demostrar la escalabilidad prácticamente lineal del algoritmo, la cual solo se ve mínimamente afectada según la densidad de átomos de cada partición.

El mayor factor limitante de este algoritmo recae sobre el sistema de particionamiento, ya que no es posible garantizar un balanceo de carga entre las GPUs debido a

que está basado en la posición de los átomos y no en la complejidad del sistema molecular. Además, las pruebas realizadas con tan solo 100 pasos de tiempo no permiten ver como evoluciona esta ausencia de balanceo de carga a lo largo del tiempo.

Por lo tanto, dados los resultados evaluados a lo largo de la presente tesis, es posible concluir que se ha demostrado la hipótesis enunciada en la Sección 1.3, puesto que es posible desarrollar un algoritmo de simulación de dinámica molecular totalmente escalable para entornos Multi-GPU, donde se utiliza la GPU como el elemento de cómputo principal para todas las interacciones y sin limitar el número de GPUs implicadas. Esta validación se obtiene a partir de la constatación de una serie de objetivos que se describirán a lo largo de las siguientes secciones.

7.1.1. Simulación paralela de dinámica molecular

Como se ha descrito en el Capitulo 4, se ha implementado un algoritmo de simulación de dinámica molecular totalmente escalable que se ejecuta íntegramente en entornos Multi-GPU. Este algoritmo no utiliza las GPUs meramente como coprocesadores, si no que este es su elemento principal de procesamiento. De esta manera la CPU se encarga únicamente de gestionar la comunicación entre todas las GPUs implicadas, así como del flujo de trabajo de cada una de ellas.

En el Capitulo 6 se ha demostrado que el algoritmo de simulación presenta un método de comunicación eficiente entre los diferentes ECs implicados, además de un uso reducido de memoria. Se ha comparado el funcionamiento con otros simuladores de dinámica molecular, como NAMD, ejecutando las mismas simulaciones de diferentes sistemas, tanto balanceados como desbalanceados. Concretamente se han utilizado sistemas moleculares como 4V4L y 4UDF, que proporcionan cientos de miles de átomos, el sistema 3J3Q, compuesto aproximadamente de 65 millones de átomos y, finalmente, el sistema 4UDF replicado múltiples veces para obtener miles de millones de átomos, llegando a formar una gigamolécula. El sistema de particio-

namiento utilizando permite utilizar en el proceso de simulación tantas GPUs como sean necesarias para cubrir el tamaño del sistema molecular sin que suponga una perdida de rendimiento. Como prueba de esta escalabilidad se ha llevado a cabo una evaluación exhaustiva comprobando tanto el rendimiento como la memoria utilizada en diferentes números de GPUs, tanto en un centro de supercomputación (BSC) como en un entorno de cloud computing (AWS).

Gracias al trabajo realizado se han publicado una serie de artículos científicos que se detallan a continuación:

- A Prototype of a Scalable Multi-GPU Molecular Dynamics Simulator for Large Molecular Systems. Spanish Computer Graphics Conference (CEIG) 2018. The Eurographics Association. [38].
- MDScale: Scalable Multi-GPU bonded and short-range molecular dynamics. Journal of Parallel and Distributed Computing. **JCR: 2.296**. Rank: 35/108 (**Q2** en COMPUTER SCIENCE, THEORY & METHODS). Minor revision [39].

7.1.2. Gestión y preprocesado para simulación de dinámica molecular

Además del desarrollo del algoritmo de simulación, esta tesis presenta una aplicación que facilita la simulación de dinámica molecular, utilizando servicios de cloud computing y automatizando todas las tareas necesarias. Esta aplicación permite a los usuarios gestionar tanto el procesado de los datos moleculares como la simulación del sistema molecular en un entorno de cloud computing Multi-GPU. A través de una interfaz web, el sistema provee de un flujo de trabajo sencillo para llevar a cabo simulaciones de dinámica molecular aceleradas por GPU por usuarios no especializados.

Esta herramienta se ha diseñado con el objetivo de ahorrar tiempo a la hora de preparar el entorno cloud, así como gestionar el gasto de recursos de manera eficiente al preparar el sistema molecular y ejecutar los simuladores. La aplicación consiste en una estructura típica de aplicación web, donde existen un servidor y un servicio web que ofrece una API REST. El usuario interactúa con la simulación a través de un navegador web, ejecutando cada tarea y pudiendo dejarlas desatendidas. La aplicación se encarga de parar cualquier recurso cloud para evitar gastos innecesarios. Por lo tanto, podemos decir que la aplicación provee de un *Software as a Service* (SaaS) que puede ser utilizado por el usuario sin necesidad de conocer y gestionar su infraestructura subyacente.

Además de gestionar los recursos localizados en el entorno cloud, la herramienta se encarga de llevar a cabo todo el procesado o pipeline de datos necesario, partiendo de un sistema molecular básico, que puede estar sin tratar, hasta obtener un sistema totalmente listo para su simulación de forma correcta. De este modo se puede simplificar drásticamente el proceso de simulación para que pueda ser llevado a cabo por usuarios no especializados en el campo de la simulación de dinámica molecular. Dada la finalidad de proporcionar una infraestructura donde ejecutar el software de dinámica molecular, la aplicación debe implementar dicha infraestructura completa dentro del servicio de cloud computing.

Este sistema se ha desarrollado para facilitar el uso de diferentes softwares de dinámica molecular. Se han compilado herramientas como VMD, NAMD, etc. para realizar el flujo de trabajo de dinámica molecular sin necesidad de entender como se configura y ejecuta cada una de ellas. Además, el diseño está totalmente abierto a cambios y mejoras. Es posible reemplazar cualquier software o algoritmos implementados para realizar cualquiera de las tareas del proceso de simulación. Esto puede permitir a los usuarios realizar comparaciones con nuevos algoritmos y técnicas que se vayan desarrollando en un futuro.

A raíz de los objetivos cumplidos de esta sección se ha publicado el artículo científico que se detalla a continuación:

- A Web-Based Tool for Simulating Molecular Dynamics in Cloud Environments. Electronics. **JCR: 2.412**. Rank: 125/266 (**Q2** en ENGINEERING, ELECTRICAL & ELECTRONIC) [40].

7.2. Trabajo futuro

Teniendo en cuenta que se han cumplido los distintos objetivos planteados en esta tesis, se deben destacar algunas mejoras que se podrían realizar como trabajo futuro.

7.2.1. Balanceo de carga

Como se ha visto a lo largo de los diferentes capítulos, se han realizado diversas pruebas, tanto con sistemas balanceados como desbalanceados. Aunque los resultados obtenidos muestran una escalabilidad prácticamente lineal en ambos escenarios, se ha visto que el rendimiento es algo inferior con los sistemas desbalanceados. Por ello uno de los trabajos más destacables de cara al futuro es implementar un balanceo de carga que se realice a la hora de llevar a cabo el particionamiento, además de actualizar las cargas en cada GPU a lo largo de todo el proceso.

7.2.2. Cálculo de fuerzas electrostáticas de largo alcance

Otra de las características de las que no dispone actualmente el simulador es la capacidad de simular fuerzas electrostáticas de largo alcance, puesto que el algoritmo expuesto en esta tesis solo trata el problema de la simulación de fuerzas de enlace y electrostáticas de corto alcance. En trabajos previos en los que se basa esta tesis

[41] se analizan e implementan estas técnicas en entornos Multi-GPU en un solo nodo computacional con resultados satisfactorios. Por tanto, se plantea incorporarla igualmente a la aplicación desarrollada, implementando diferentes posibilidades, como PME o MSM.

7.2.3. Ampliación de herramienta cloud computing

En cuanto a la herramienta desarrollada para la gestión de entornos cloud, esta tiene ciertas limitaciones en el soporte de dichos entornos. La aplicación ha sido diseñada teniendo presente el uso de Amazon Web Services, debido a que ofrece la posibilidad de utilizar instancias con GPU. Sin embargo, se ha diseñado con la intención de ofrecer soporte para otros entornos cloud con diferentes configuraciones y SDKs, como puede ser Google Cloud Platform o Microsoft Azure, o incluso soportar el uso de un clúster local o un centro de supercomputación. En este punto, también es necesario integrar mecanismos de seguridad a la hora de establecer la comunicación entre el servicio web y la infraestructura, además de proveer de protección frente a ataques DDoS, inyección SQL y otros ataques similares.

Por último, el proceso de renderizado de imágenes moleculares es también un proceso altamente costoso y más en entornos distribuidos, si se trata de representar un número elevado de átomos. Esta característica se pretender explorar en trabajos futuros.

Bibliografía

[1] **Amazon EC2 P2 instances**. Accessed: 2021-04-19. Available from: https://aws.amazon.com/ec2/instance-types/p2/. 81

[2] **Amazon EC2 T2 instances**. Accessed: 2021-04-19. Available from: https://aws.amazon.com/ec2/instance-types/t2/. 79

[3] **Amazon EC2 X1e instances**. Accessed: 2021-04-19. Available from: https://aws.amazon.com/ec2/instance-types/x1e/. 79

[4] **AWS SDK for Python (boto3)**. Accessed: 2021-04-19. Available from: https://aws.amazon.com/sdk-for-python/. 79

[5] **BSC Minotauro Cluster**. Accessed: 2021-04-19. Available from: https://www.bsc.es/es/marenostrum/minotauro. 78

[6] **PSFGen plugin for VMD**. Accessed: 2021-04-19. Available from: https://www.ks.uiuc.edu/Research/vmd/plugins/psfgen/. 53

[7] **Solvate plugin for VMD**. Accessed: 2021-04-19. Available from: https://www.ks.uiuc.edu/Research/vmd/plugins/solvate/. 54

[8] Mark James Abraham, Teemu Murtola, Roland Schulz, Szilárd Páll, Jeremy C. Smith, Berk Hess, and Erik Lindahl. **GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers**. *SoftwareX*, **1-2**:19–25, 2015. 26, 27

[9] Dan A. Alcantara, Andrei Sharf, Fatemeh Abbasinejad, Shubhabrata Sengupta, Michael Mitzenmacher, John D. Owens, and Nina

Amenta. **Real-time Parallel Hashing on the GPU**. *ACM Trans. Graph.*, **28**(5):154:1–154:9, dec 2009. 42

[10] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. **The Protein Data Bank**. *Nucleic Acids Research*, **28**(1):235–242, 01 2000. 3, 51, 72

[11] M. Bolten. *Multigrid methods for structured grids and their application in particle simulation*. Dr. (univ.), Univ. Wuppertal, Jülich, 2008. Record converted from VDB: 12.11.2012; Wuppertal, Univ., Diss., 2008. 23

[12] B. R. Brooks, C. L. Brooks, A. D. Mackerell, L. Nilsson, et al. **CHARMM: The biomolecular simulation program**. *Journal of Computational Chemistry*, **30**(10):1545–1614, 2009. 26

[13] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. **CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms**. *Software: Practice and Experience*, **41**(1):23–50, 2011. 30

[14] Wendy D. Cornell, Piotr Cieplak, Christopher I. Bayly, Ian R. Gould, Kenneth M. Merz, David M. Ferguson, David C. Spellmeyer, Thomas Fox, James W. Caldwell, and Peter A. Kollman. **A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules**. *Journal of the American Chemical Society*, **117**(19):5179–5197, 1995. 14

[15] Evangelos A. Coutsias and Michael J. Wester. **RMSD and Symmetry**. *Journal of Computational Chemistry*, **40**(15):1496–1508, 2019. 54

[16] Tom Darden, Darrin York, and Lee Pedersen. **Particle mesh Ewald: An Nlog(N) method for Ewald sums in large systems**. *The Journal of Chemical Physics*, **98**(12):10089–10092, 1993. 23, 88

[17] Matthew Dobson, Ian Fox, and Alexandra Saracino. **Cell List Algorithms for Nonequilibrium Molecular Dynamics**. *Journal of Computational Physics*, **315**, 12 2014. 22, 38

[18] Matthieu Dreher, Marc Piuzzi, Ahmed Turki, Matthieu Chavent, Marc Baaden, Nicolas Férey, Sébastien Limet, Bruno Raffin, and Sophie Robert. **Interactive Molecular Dynamics: Scaling up to Large Systems**. *Procedia Computer Science*, **18**:20–29, 2013. 26

[19] Wolfgang Eckhardt, Alexander Heinecke, Reinhold Bader, Matthias Brehm, Nicolay Hammer, Herbert Huber, Hans-Georg Kleinhenz, Jadran Vrabec, Hans Hasse, Martin Horsch, Martin Bernreuther, Colin W. Glass, Christoph Niethammer, Arndt Bode, and Hans-Joachim Bungartz. **591 TFLOPS Multi-trillion Particles Simulation on SuperMUC**. In *International Supercomputing Conference*, pages 1–12, Leipzig, Germany, 2013. Springer Berlin Heidelberg. 2

[20] M. Eleftheriou, B. G. Fitch, A. Rayshubskiy, T. J. C. Ward, and R. S. Germain. **Scalable framework for 3D FFTs on the Blue Gene/L supercomputer: Implementation and early performance measurements**. *IBM Journal of Research and Development*, **49**(2.3):457–464, 2005. 27

[21] Ulrich Essmann, Lalith Perera, Max L. Berkowitz, Tom Darden, Hsing Lee, and Lee G. Pedersen. **A smooth particle mesh Ewald method**. *The Journal of Chemical Physics*, **103**(19):8577–8593, 1995. 23

[22] M. Fratarcangeli and F. Pellacini. **Scalable Partitioning for Parallel Position Based Dynamics**. *Comput. Graph. Forum*, **34**(2):405–413, May 2015. 27

[23] Marco Fratarcangeli and Fabio Pellacini. **A GPU-Based Implementation of Position Based Dynamics for Interactive Deformable Bodies**. *Journal of Graphics Tools*, **17**:59–66, 01 2015. 27

[24] Edgar Gabriel, Graham E. Fagg, George Bosilca, Thara Angskun, Jack J. Dongarra, Jeffrey M. Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, Ralph H. Castain, David J. Daniel, Richard L. Graham, and Timothy S. Woodall. **Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation**. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, pages 97–104, Budapest, Hungary, September 2004. 57

[25] Marco T Gallo, Barry J Grant, Miguel L Teodoro, Julia Melton, Piotr Cieplak, George N Phillips Jr, and Boguslaw Stec. **Novel procedure for thermal equilibration in molecular dynamics simulation**. *Molecular simulation*, **35**(5):349–357, 2009. 52, 54

[26] Timothy Germann and Kai Kadau. **Trillion-atom molecular dynamics becomes a reality**. *International Journal of Modern Physics C*, **8**:1315–1319, 09 2008. 2

[27] David Hardy. **Multilevel Summation for the Fast Evaluation of Forces for the Simulation of Biomolecules**. 01 2006. 23, 25

[28] David Hardy, John Stone, and Klaus Schulten. **Multilevel Summation of Electrostatic Potentials Using Graphics Processing Units**. *Parallel computing*, **35**:164–177, 03 2009. 24, 25, 88

[29] David J. Hardy, Zhe Wu, James C. Phillips, John E. Stone, Robert D. Skeel, and Klaus Schulten. **Multilevel Summation Method for Electrostatic Force Evaluation**. *Journal of Chemical Theory and Computation*, **11**(2):766–779, 2015. 13

[30] M. J. Harvey, G. Giupponi, and G. De Fabritiis. **ACEMD: Accelerating Biomolecular Dynamics in the Microsecond Time Scale**. *Journal of Chemical Theory and Computation*, **5**(6):1632–1639, 2009. 23, 26, 27

[31] Berk Hess, Carsten Kutzner, David van der Spoel, and Erik Lindahl. **GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation**. *Journal of Chemical Theory and Computation*, **4**(3):435–447, 2008. PMID: 26620784. 23

[32] William Humphrey, Andrew Dalke, and Klaus Schulten. **VMD – Visual Molecular Dynamics**. *Journal of Molecular Graphics*, **14**:33–38, 1996. 53, 54, 84

[33] Kai Kadau, Timothy Germann, and Peter S. Lomdahl. **Molecular Dynamics Comes of Age:. 320 Billion Atom Simulation on BlueGene/L**. *International Journal of Modern Physics C*, **17**:1755–1761, 12 2006. 2

[34] Samuel Kupka. **Molecular dynamics on graphics accelerators**. In *Proceedings of CESCG*, pages 1–4, Castá-Papiernicka Centre, Slovakia, 2006. Institute of Computer Graphics and Algorithms. 26

[35] Ang Li, Shuaiwen Leon Song, Jieyang Chen, Jiajia Li, Xu Liu, Nathan R. Tallent, and Kevin J. Barker. **Evaluating Modern GPU Interconnect: PCIe, NVLink, NV-SLI, NVSwitch and GPUDirect**. *IEEE Transactions on Parallel and Distributed Systems*, **31**(1):94–110, 2020. 50

[36] Jesús Montes, Alberto Sanchez, Bunjamin Memishi, María S. Pérez, and Gabriel Antoniou. **GMonE: a complete approach to cloud monitoring.** *Future Generation Computer Systems, Elsevier. ISSN:0167-739X*, **29**(8):2026–2040, oct 2013. 59

[37] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. **Position Based Dynamics**. *J. Vis. Comun. Image Represent.*, **18**(2):109–118, April 2007. 27

[38] Gonzalo Nicolas-Barreales, Marcos Novalbos, Miguel Ángel Otaduy, and Alberto Sánchez. **A Prototype of a Scalable Multi-GPU Molecular Dynamics Simulator for Large Molecular Systems**. In Ignacio García-Fernández and Carlos Ureña, editors, *Spanish Computer Graphics Conference (CEIG)*, pages 25–28. The Eurographics Association, 2018. 93

[39] Gonzalo Nicolas-Barreales, Miguel-Angel Otaduy, and Alberto Sanchez. **MDScale: Scalable Multi-GPU bonded and short-range molecular dynamics**. *Journal of Parallel and Distributed*, 2021. 93

[40] Gonzalo Nicolas-Barreales, Aaron Sujar, and Alberto Sanchez. **A Web-Based Tool for Simulating Molecular Dynamics in Cloud Environments**. *Electronics*, **10**(2), 2021. 95

[41] Marcos Novalbos, Jaime Gonzalez, Miguel A. Otaduy, Roberto Martinez-Benito, and Alberto Sanchez. **Scalable On-Board Multi-GPU Simulation of Long-Range Molecular Dynamics**. In *Proceedings of the Euro-Par 2014 Parallel Processing*, pages 752–763, Porto, Portugal, 2014. Springer. 4, 24, 96

[42] Marcos Novalbos, Jaime Gonzalez, Miguel Angel Otaduy, Alvaro Lopez-Medrano, and Alberto Sanchez. **On-Board Multi-GPU Molecular Dynamics**. In *Proceedings of the Euro-Par 2013 Parallel Processing*, pages 862–873, Aachen, Germany, 2013. Springer. 4, 20, 21, 64

[43] David A. Pearlman, David A. Case, James W. Caldwell, Wilson S. Ross, Thomas E. Cheatham, Steve DeBolt, David Ferguson, George Seibel, and Peter Kollman. **AMBER, a package of computer programs for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to simulate the structural and energetic properties of molecules**. *Computer Physics Communications*, **91**(1):1–41, 1995. 26

[44] J. R. Perilla, G. Zhao, P. Zhang, and K. J. Schulten. **PDB ID: 3J3Q. Atomic-level structure of the entire HIV-1 capsid**, 2013. 73, 79

[45] James C. Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D. Skeel, Laxmikant Kalé, and Klaus Schulten. **Scalable molecular dynamics with NAMD**. *Journal of Computational Chemistry*, **26**(16):1781–1802, 2005. 26, 27, 54

[46] S. Plimpton and G. Heffelfinger. **Scalable parallel molecular dynamics on MIMD supercomputers**. In *Proceedings Scalable High Performance Computing Conference SHPCC-92.*, pages 246–251, 1992. 27

[47] Steve Plimpton, Roy Pollock, and Mark Stevens. **Particle-Mesh Ewald and rRESPA for Parallel Molecular Dynamics Simulations**. In *Proceedings of the Eighth SIAM Conference on Parallel Processing for Scien-*

tific Computing, pages 1–13, Minneapolis, Minnesota, USA, 1997. Society for Industrial & Applied Mathematics. 13

[48] Christoph Rachinger. **Scalable Computation of Long-Range Potentialsfor Molecular Dynamics**, 2013. 23

[49] Christopher Rodrigues, David Hardy, John Stone, Klaus Schulten, and Wen-mei Hwu. **GPU acceleration of cutoff pair potentials for molecular modeling applications**. In *Proceedings of the 2008 Conference on Computing Frontiers, CF'08*, pages 273–282, 01 2008. 23, 27

[50] E. Rustico, G. Bilotta, G. Gallo, A. Herault, and C. Del Negro. **Smoothed Particle Hydrodynamics Simulations on Multi-GPU Systems**. In *2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, pages 384–391, Garching, Germany, Feb 2012. IEEE Computer Society. 27

[51] Veronica Salmaso and Stefano Moro. **Bridging Molecular Docking to Molecular Dynamics in Exploring Ligand-Protein Recognition Process: An Overview**. *Frontiers in Pharmacology*, **9**:923, 2018. 2

[52] Tamar Schlick. *Molecular Modeling and Simulation: An Interdisciplinary Guide*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002. 2, 38

[53] Loren Schwiebert, Eyad Hailat, Kamel Rushaidat, Jason Mick, and Jeffrey Potoff. **An Efficient Cell List Implementation for Monte Carlo Simulation on GPUs**. *CoRR*, **1408.3764**, 08 2014. 39

[54] S. Shakeel, B. M. Westerhuis, A. Ora, G. Koen, A. Q. Bakker, Y. Claassen, K. Wagner, T. Beaumont, K. C. Wolthers, and S.J. Butcher. **PDB ID: 4UDF. Structural Basis of Human Parechovirus**

Neutralization by Human Monoclonal Antibodies. *J. Virol.*, **89**:9571–9580, 2015. 77

[55] Robert D. Skeel, Ismail Tezcan, and David J. Hardy. **Multiple grid methods for classical molecular dynamics**. *Journal of Computational Chemistry*, **23**(6):673–684, 2002. 23

[56] John Stone, Justin Gullingsrud, and Klaus Schulten. **A System for Interactive Molecular Dynamics Simulation**. In *2001 ACM Symposium on Interactive 3D Graphics*, pages 191–194, New York, NY, USA, 03 2001. ACM. 26

[57] Top500.org. **TOP500 supercomputer sites list statistics**. `https://www.top500.org/statistics/list/`, 11 2018. Accessed: 2021-04-19. 2, 3

[58] Ramkumar V. Vadali, Yan Shi, Sameer Kumar, Laxmikant V. Kale, Mark E. Tuckerman, and Glenn J. Martyna. **Scalable fine-grained parallelization of plane-wave–based ab initio molecular dynamics for large supercomputers**. *Journal of Computational Chemistry*, **25**(16):2006–2022, 2004. 27

[59] Christian Vecchiola, Xingchen Chu, and Rajkumar Buyya. **Aneka: A Software Platform for .NET-based Cloud Computing**. *High Speed and Large Scale Scientific Computing*, **18**, 07 2009. 30

[60] Toby Velte, Anthony Velte, and Robert Elsenpeter. *Cloud Computing, A Practical Approach*. McGraw-Hill, Inc., USA, 1 edition, 2009. 29

[61] Juekuan Yang, Yujuan Wang, and Yunfei Chen. **GPU accelerated molecular dynamics simulation of thermal conductivities**. *Journal of Computational Physics*, **221**:799–804, 02 2007. 26

[62] Zhenhua Yao, Jian-Sheng Wang, G.R. Liu, and Min Cheng. **Improved neighbor list algorithm in molecular simulations using cell decomposition and data sorting method**. *Computer Physics Communications*, **161**:27–35, 08 2004. 38, 46

[63] Rio Yokota, Jaydeep P. Bardhan, Matthew G. Knepley, L.A. Barba, and Tsuyoshi Hamada. **Biomolecular electrostatics using a fast multipole BEM on up to 512 gpus and a billion unknowns**. *Computer Physics Communications*, **182**(6):1272 – 1283, 2011. 23

[64] S. Yuan, M. Topf, L. Dorstyn, S. Kumar, S. J. Ludtke, and C. W. Akey. **PDB ID: 4V4L. Structure of the Drosophila apoptosome at 6.9 angstrom resolution**. *Structure*, **19**:128–140, 2011. 72, 79

[65] Gongpu Zhao, Juan Perilla, Ernest Yufenyuy, Xin Meng, Bo Chen, Jiying Ning, Jinwoo Ahn, Angela M Gronenborn, Klaus Schulten, Christopher Aiken, and Peijun Zhang. **Mature HIV-1 Capsid Structure by Cryo-Electron Microscopy and All-Atom Molecular Dynamics**. *Nature*, **497**:643–646, 05 2013. 73