Computación de Alto Desempeño Aplicada en Técnicas de Simulación de Proteínas

Fabio Antonio Avellaneda Pachón

Trabajo de grado para optar al título de Magíster en Ciencias Biológicas

Grupo de Bioquímica Molecular Computacional y Bioinformática
Facultad de Ciencias
Pontificia Universidad Javeriana
Junio, 2009

Computación de Alto Desempeño Aplicada en Técnicas de Simulación de Proteínas

Fabio Antonio Avellaneda Pachón

APROBADO

Leonardo René Lareo, Ph. D.
Director

Orlando Emilio Acevedo, Ph. D
Director

Luz Mary Salazar Pulido, Ph. D.
Jurado

Janeth González Santos, Ph. D.
Jurado

Juan Pablo Garzón Ruiz, M.Sc
Jurado

Contenido

Contenido i
Índice de Tablas iii
Índice de Gráficas iv
Introducción 1
Objetivos 2
General 2
Específicos 2
Estado del Arte 3
Necesidad de Cálculo 3
Computación de Alto Desempeño 4
Cluster 7
Grilla 7
Computación de Alto Desempeño en Colombia y en el Mundo 8
Bioinformática y Biología Computacional 11
Dinámica Molecular Computacional 12
DESCRIPCIÓN DEL EXPERIMENTO 14
SELECCIÓN DE PROTEINAS 14
VARIABLES INDEPENDIENTES 16
VARIABLES DEPENDIENTES 16
SISTEMAS DE CÓMPUTO 17
RACK INGENIERÍA 17
PC DE ESCRITORIO 17
SALA BD 17
SALA B 18
SALA A 18
SUPERCLUSTER 18
MONTAJE DEL EXPERIMENTO 18
PREPARACIÓN DE LAS PROTEINAS PARA SU MINIMIZACIÓN 19

CONFIGURACIÓN DE LA MINIMIZACIÓN 19
Preparación del ambiente distribuido 20
RESULTADOS Y ANÁLISIS 23
Resumen de Varianzas 23
Respecto a los ambientes de ejecución 24
Respecto a las características de las proteínas 26
Respecto a la estabilización 31
Speed Up y Eficiencia 32
Conclusiones y Trabajo Futuro 34
Desarrollo de software especializado para problemas específicos 39
Script para descarga de proteínas 39
Scripts para preparación de proteínas 39
Generador de estructura de directorios para simulaciones 40
Consolidador de wallclock 41
Consolidador de graficas en Excel 42
Bibliografía 43

Índice de Tablas

Tabla 1. Características de cada uno de los servidores del Rack 17
Tabla 2. Resumen de varianzas para el Rack de Ingeniería en configuración secuencial 23
Tabla 3. Resumen de varianzas para el Rack de Ingeniería en configuración alternada 23
Tabla 4. Información de las proteínas utilizadas en la simulación. 26
Tabla 5. Resumen de los tiempos obtenidos luego de la ejecución de las minimizaciones. 27
Tabla 6. Comparación entre el tiempo mínimo obtenido y el tiempo de estabilización de la simulación. 32
Tabla 7. Eficiencia y Speed Up en tiempo mínimo y en tiempo de estabilización 33
Tabla 8. Ejemplo de archivo generado por el consolidador de resultados 42

Índice de Gráficas

Gráfica 1. Cantidad de átomos en la simulación Vs. Cantidad de procesadores necesarios para alcanzar el mínimo tiempo de ejecución. Datos tomando en cuenta la media móvil con periodo 3 28
Gráfica 2. Peso Molecular Vs. Cantidad de procesadores necesarios para alcanzar el mínimo tiempo de ejecución. Datos tomando en cuenta la media móvil con periodo 3 29
Gráfica 3. Cantidad de residuos Vs. Cantidad de procesadores necesarios para alcanzar el mínimo tiempo de ejecución. Datos tomando en cuenta la media móvil con periodo 3 29
Gráfica 4. Porcentaje de Hélices Alfa Vs. Cantidad de procesadores necesarios para alcanzar el mínimo tiempo de ejecución. Datos tomando en cuenta la media móvil con periodo 3..... 30
Gráfica 5. Porcentaje de hojas plegadas Beta Vs. Cantidad de procesadores necesarios para alcanzar el mínimo tiempo de ejecución. Datos tomando en cuenta la media móvil con periodo 3 31

Introducción

La investigación en sistemas bionanométricos carece de herramientas de visualización, pues su naturaleza hace difícil la visualización con microscopía, cristalografía, NMR, de manera que es necesario acudir a sistemas de simulación para poder estudiar los fenómenos que allí acontecen[1].

Una de las grandes limitaciones para la realización de los procesos de simulación de sistemas complejos es la capacidad de cómputo que poseen las máquinas actuales, las cuales toman grandes tiempos para generar un resultado, solución y/o visualización[2]. Específicamente en el área de bioinformática, la simulación molecular y su correspondiente visualización han sido puntos críticos en el avance de la investigación[1, 3].

Muchos de los problemas de frontera en la bionanotecnología no tienen aún aproximaciones experimentales para ser solucionados o su experimentación es muy costosa, dejando las herramientas computacionales como única alternativa para el descubrimiento y generación de conocimiento en esos tópicos, mediante la simulación. De esta manera, se han tratado problemas como la caracterización de la α-Hemolisina[4], mecanismos de plegamiento de arreglos nucleosomales[5], modelos de interacción y evolución bacteriana[6], entre otros.

Dentro del campo de la informática aparecen continuas evoluciones y nuevos paradigmas para la solución de problemas. Desde esta perspectiva, se han alcanzado notables desarrollos para mejorar la eficacia y eficiencia de los sistemas computacionales mediante el uso de sistemas de cálculo en paralelo utilizando sistemas distribuidos como clusters [7] y sistemas de computación en grilla[8, 9].

En el grupo de investigación Sistemas de Información, Sistemas Distribuidos y Redes (SIDRe) [10] de la Facultad de Ingeniería de la Pontificia Universidad Javeriana, se cuenta con el conocimiento para desarrollar e implementar sistemas de cómputo de alto desempeño para la solución de problemas con alta exigencia computacional. Por otra parte, el grupo de Bioquímica Computacional Estructural y Bioinformática (BCEB)[11] de la Facultad de Ciencias de la Pontificia Universidad Javeriana cuenta con una serie de problemas de simulación molecular que requieren de sistemas computacionales eficientes y eficaces en su solución. La integración de las ciencias biológicas y la ingeniería de sistemas es de esencial importancia para identificar cuáles son las mejores opciones para abordar la simulación de sistemas nanomoleculares biológicos con los recursos disponibles para investigación, en particular en la Pontificia Universidad Javeriana.

Dado lo anterior, este trabajo busca estudiar, proponer, caracterizar y evaluar un sistema de procesamiento en paralelo para la solución de problemas biológicos mediante la simulación, utilizando y ampliando la infraestructura computacional actual con que se cuenta en el grupo de investigación SIDRe de la Pontificia Universidad Javeriana para investigación, de manera que se pueda obtener una aproximación a la simulación de un problema de frontera en bioquímica de

proteínas. Vale la pena aclarar que aunque este trabajo se encuentra enmarcado en las necesidades computacionales que las ciencias biológicas presentan, es extensible a todo tipo de problemas.

Con los hallazgos de este trabajo se pretende demostrar que con los recursos disponibles en una institución académica es posible llevar a cabo experimentos que requieran de alto poder de cómputo sin necesidad de hacer grandes inversiones. Más concretamente, en el caso de la Pontificia Universidad Javeriana la intención está en demostrar la necesidad de establecer políticas que permitan implementar un sistema de cómputo de alto desempeño con los recursos existentes en el campus. Con esto, poder recibir el apoyo institucional para posteriormente llevar la iniciativa a otras universidades con el fin de crear un supercomputador en nuestra ciudad, o mejor aún, en nuestro país.

Para ello, primero se dará una breve descripción del estado del arte de la computación de alto desempeño, donde se puede encontrar una breve reseña del desarrollo de la computación hasta llegar a los sistemas existentes en la actualidad tanto en Colombia como en el mundo. Seguidamente se describe el contexto biológico y se denota la necesidad de la computación de alto desempeño en problemas de investigación.

Posteriormente se hace la descripción del experimento donde se indican los criterios de selección de las proteínas y las variables (dependientes, independientes e intervinientes) contempladas en el mismo. A su vez, se describen brevemente los sistemas de cómputo utilizados y su configuración general.

Luego, en el capítulo denominado montaje del experimento, se explica la forma en que las proteínas fueron preparadas para la simulación y se describe brevemente la configuración de dicha minimización. También se puede observar la forma como fueron preparados los ambientes computacionales para la ejecución.

Finalmente se encuentra el capítulo donde se obtienen los resultados y se hacen diversos cálculos para derivar información adicional de gran utilidad y de esta manera obtener las conclusiones y posibles trabajos que se derivan del actual, y que conforman el penúltimo capítulo, seguido de la descripción de los programas desarrollados a lo largo de esta investigación

Objetivos

General

Realizar un análisis comparativo de un sistema paralelo y distribuido aplicado a la solución de problemas de dinámica molecular que involucren alta carga computacional.

Específicos

- Implementar el sistema de procesamiento.
- Escoger las técnicas de simulación de proteínas e implementarlas en sistema desarrollado.
- Comparar y analizar el desempeño de los diferentes procesos escogidos.

Estado del Arte

Necesidad de Cálculo

La necesidad de calcular ha sido una fuerza conductora para la innovación a lo largo de los siglos[12]. Los computadores electrónicos fueron precedidos en sus inicios por versiones mecánicas, en donde la máquina analítica de Babbage representó el primer intento por desarrollar un poder para realizar cálculos significativamente mayor al que puede ser logrado a mano. A pesar de los problemas inherentes a su diseño y posibilidad de desarrollo para su época, el trabajo de Babbage dio predicciones sobre la tecnología del futuro cercano, incluyendo las máquinas tabuladoras mecánicas usadas por el gobierno de los Estados Unidos a principios del siglo 20 para manejar las estadísticas del censo. Tiempo después, la International Business Machines (IBM) se dedicó a construir y mantener máquinas tabuladoras electromecánicas.

Durante la Segunda Guerra Mundial aparecieron los primeros computadores electrónicos. Surge el Electronic Numerical Integrator and Computer (ENIAC) en 1948, conformado por 18.000 tubos al vacío y una velocidad de operación de 100.000 ciclos por segundo. Se estima que una persona trabajando con una calculadora mecánica puede ejecutar una operación de 16 dígitos por minuto, incluyendo el tiempo para las operaciones de entrada y salida (ingresar los números y almacenar el resultado), de manera que el desempeño para el sistema humano-calculadora podría ser de alrededor de 1/16 FLOP (Floating point OPeration(s) per Second – operación de punto flotante por segundo). El ENIAC podía desarrollar una multiplicación en 2.8 ms, una división en 24 ms, y una suma en 0.2 ms, de forma que su tasa de FLOPS varía entre 40 y 5000. Dado lo anterior, el ENIAC operaba de 240 a 300.000 veces más rápido que un sistema humano-calculadora.

Posterior al ENIAC y gracias a la arquitectura propuesta por Von Newmann, se inició la investigación enfocada en microprocesadores, memorias, almacenamiento masivo y software y desempeño computacional. La primera generación de computadores, alrededor de 1975 fue ejemplificada por el Altair 8800 (que usaba un procesador Intel 8008 operando con palabras de 8 bits a una velocidad de 2MHz) y el IMSAI 8080, los cuales se encontraban en la forma de kit que debía ser ensamblado y probado por el usuario. Tenían una terminal de entrada y salida estilo teletipo y el almacenamiento de programas estaba usualmente sobre tarjetas perforadas o cintas. La memoria era muy costosa, lo cual limitaba al computador a tener de entre 4 a 16Kb.

Fue hasta la aparición de los computadores listos para usar como el Commodore PET, Apple II y Radio Shack TRS-80 (alrededor de 1977), que los computadores empezaron a estar disponibles para una mayor cantidad de audiencia. En esta segunda generación de computadores se dieron fuertes cambios en la industria del software y del hardware; la memoria disponible en estos sistemas se incrementó a 64Kb o más, y el teletipo fue reemplazado por los monitores o las

impresoras de matriz de punto. El almacenamiento masivo también evolucionó hacia las unidades de disco flexible. Respecto al software nacen lenguajes como FORTRAN, C y Pascal.

La tercera generación de computadores nace con la introducción del IBM PC (1981), y continuó con el Macintosh de Apple, Amiga de Commodore y PC-AT de IBM. Estos computadores trabajaban a velocidades de 20MHz o más y contaban con procesadores de 16 o incluso 32 bits.

Posteriormente surge la cuarta generación de los computadores personales basados en procesadores de 32 bits como el MacII, el IBM System 2, y Compaq. Estas nuevas máquinas eran capaces de direccionar la antes inimaginable cantidad de 16Gb, cuando lo común era encontrar computadores con 5Mb. Dadas estas características, se hablaba de "mainframes on chips" o "personal mainframes" ejemplificados por el Intel 80486[13].

En la actualidad, el mercado enfocado a personas que usan sus computadores personales para aplicaciones con gráficos en 3D, multimedia, juegos, entre otras ha impulsado el desarrollo de procesadores de alto desempeño económicos[14].

Computación de Alto Desempeño

El desarrollo de los computadores de gran escala, científicos o mainframes inició con la compañía Control Data Corporation (CDC), quienes lanzaron el CDC6600 en 1964 diseñado por Seymour Cray; dicho computador operaba a una velocidad de 9MFLOPS. Burroughs, otro de los constructores de mainframes, desarrolló el ILLIAC IV, primer computador en usar un diseño paralelo (o no von Neumann). Tenía 64 computadores escalares idénticos operando en paralelo. Es aquí donde empiezan a aparecer sistemas de cómputo de alto desempeño: sin que aún se conocieran con este nombre, ya se buscaba la realización de cálculos en periodos cortos de tiempo, aprovechando al máximo las características del hardware y de hecho, manteniendo la investigación para superar los picos de procesamiento logrados en cada instante.

Posteriormente, el concepto de RISC (Reduced Instruction Set Computer) fue introducido por IBM en 1971, al tiempo que Seymour Cray dejó CDC para formar la compañía Cray Research. En 1976 Cray lanzó el Cray-1, construido utilizando arquitectura vectorial y forma cilíndrica para acortar la conexión entre componentes. Proveía un desempeño de 100 MFLOPS. Esta innovación continuó con una serie de computadores Cray, incluyendo el Cray X-MP que tenía dos Cray-1 conectados en paralelo, con un desempeño tres veces mayor al Cray-1.

Por otra parte, Immos, una compañía del Reino Unido, desarrolló el transputer. La Thinking Machines Corporation fue fundada específicamente para desarrollar computadores masivamente paralelos. Subsecuentemente en 1985, esta compañía introdujo la "Connection Machine", la cual tenía 16.384 procesadores paralelos y una velocidad tope de 600 MFLOPS. Usando cuatro veces este número de procesadores, se alcanzó el pico de 1 GFLOPS. Cray-2 mas adelante alcanzó dicho

tope y de hecho en 1995 lanzó el Cray Y-MP C90, equipado con 16 procesadores, y una velocidad de 16 GFLOPS.

La tendencia de este tipo de computación se dividió en dos corrientes: una que incorpora grandes clusters de computadores y estaciones de trabajo. La otra involucra un gran número de procesadores altamente integrados, y utilizando memoria compartida. También ha evolucionado un híbrido entre estas dos tendencias, donde máquinas multiprocesador están interconectadas conformando un cluster[15].

Se pueden clasificar los sistemas de computación de alto desempeño en dos categorías principales: Alta Disponibilidad y Cómputo Intensivo. Alta disponibilidad se enfoca en los sistemas tolerantes a fallos, donde las aplicaciones tienen que estar disponibles en todo momento, como sucede con los sistemas de transacciones bancarias. Por otra parte, los sistemas de cómputo intensivo se enfocan en resolver problemas en el menor tiempo posible, y generalmente involucran la paralelización de la mayor parte del problema con el fin de resolver varias partes al tiempo para luego obtener la solución. El presente trabajo trata la computación de alto desempeño para cómputo intensivo.

Este tipo de computación se ha empleado tradicionalmente en el diseño de alas para aviones, motores de combustión interna, construcciones, etc; sin embargo, cada día la investigación en este tipo de sistemas demuestra su importancia, pues está impulsando el desarrollo en diferentes áreas del conocimiento y la industria mediante la simulación.

Durante los años 80, investigadores de múltiples disciplinas también empezaron a reunirse para atacar problemas en ciencias e ingeniería para los cuales la infraestructura computacional de gran escala proveía una herramienta fundamental para alcanzar nuevos descubrimientos científicos[16].

Algunos ejemplos de lo anterior, se muestran a continuación[17]:

- El diseño de sistemas micro y nano electromecánicos (MEMS y NEMS) reúne una mezcla de fenómenos cuánticos, dinámica molecular y modelos estocásticos y continuos con procesos físicos como conducción, radiación y mecánica estructural, todo en un solo sistema.
- La secuenciación del genoma humano abrió nuevas fronteras en la bioinformática. La caracterización funcional y estructural de los genes y proteínas mantienen la promesa de entender los procesos biológicos, y de esta manera desarrollar nuevas medicinas y curas para enfermedades y condiciones médicas.
- Los avances en física y química computacional se han enfocado a entender procesos en la escala que comprende desde los fenómenos cuánticos hasta las estructuras macromoleculares. Esto ha resultado en el diseño de nuevos materiales y el entendimiento de rutas químicas, entre otros.
- Respecto a conjuntos de datos extremadamente grandes, es posible encontrar bases de datos de proteínas y genes como PDB, SwissProt, ENTREZ y NDB. Podría pensarse en tener todo el poder de cómputo necesario para analizar los datos en una sola máquina, pero podría ser imposible reunir todos los datos para trabajar eficientemente.

A su vez, se han publicado numerosos artículos referentes al uso de la computación de alto desempeño en problemas complejos como la simulación del ribosoma, con aproximadamente 2.64 millones de átomos[18], el descubrimiento de ligandos para la proteína FKBP12[19], el cálculo del espectro de absorción de rayos X de sólidos[20], cálculo de maniobras aerodinámicas para proyectiles en vuelo libre[21]; incluso se ha resaltado la necesidad de las simulaciones y cálculos mencionados anteriormente para mejorar la pedagogía con el uso de ejemplos más cercanos a la realidad, para la educación primaria y secundaria[22].

Dado lo anterior, diseñar sistemas que involucren la interacción de fenómenos de diversa naturaleza, analizar sus resultados y visualizarlos en tiempos menores a los que se podría tener en un solo computador requiere no solo del diseño de algoritmos innovadores sino también del desarrollo de poder computacional a gran escala.

La búsqueda de alto poder de procesamiento ha llevado la investigación incluso a trabajar sobre los componentes internos del computador para procesar datos sobre ellos, así no hayan sido concebidos originalmente para tal fin. Este es el caso del hardware gráfico, donde se ha trabajado en el cálculo de propósito general sobre unidades de procesamiento gráfico[23].

La aplicación mejor conocida de la computación de alto desempeño en las ciencias de la vida es el Proyecto Genoma Humano. Más de 1000 CPUs fueron utilizadas por los diferentes grupos en el análisis de los datos a finales de los años 90[24]. Este tipo de computación se ha convertido en una tecnología accesible para compañías farmacéuticas grandes, y más recientemente, para la comunidad biotecnológica[24].

Uno de los paradigmas de equidad de acceso en la edad de la información es que las universidades tienen una responsabilidad moral de proveer adecuados recursos computacionales a sus estudiantes [25]. La idea de utilizar tiempos de computador ocioso para otras tareas no es nueva, ha sido explotada, por ejemplo, por el proyecto Seti@Home y ha probado ser muy exitosa [25].

Típicamente, tener un supercomputador es un privilegio que pocas universidades o centros de investigación pueden tener, pues son equipos muy costosos que pueden requerir adecuaciones especiales para su instalación y además necesitan de personal experto para su mantenimiento y administración. Adicionalmente, para instituciones en países como Colombia, es más cómodo construir un sistema de cómputo de alto desempeño a partir de pocos computadores, con la posibilidad de poderlo extender con el paso del tiempo, en vez de tener que disponer de una elevada cantidad de dinero para la compra de una máquina gigantesca, sin posibilidades sencillas de crecimiento.

Los dos sistemas de computación de alto desempeño más utilizados hoy en día corresponden a la computación en cluster y a la computación en grilla, e incluso se utilizan en forma combinada para diversas tareas como predicción de clima, pero su mayor enfoque ha sido el resolver problemas biológicos como el descubrimiento de ligandos[19], descubrimiento y diseño de drogas[26], etc.

Cluster

Un sistema de cómputo en cluster es un tipo de sistema de procesamiento en paralelo y/o distribuido que consiste en una colección de computadores interconectados, los cuales trabajan juntos como un recurso integrado[27]. Los clusters de computadores convencionales son denominados *Beowulf*, diseñados para computación paralela de alto desempeño sobre computadores personales; este tipo de clusters es comúnmente usado para soportar computación científica[26].

El surgimiento de Linux y el concepto de cluster *Beowulf* permitieron la introducción de computación paralela de alto desempeño a costos bajos a los pequeños grupos de investigación, mientras que muchos laboratorios grandes creaban y operaban grandes clusters de este tipo. Los clusters son relativamente fáciles de mantener y actualizar dado que están construidos usando componentes individuales disponibles por muchos proveedores. Otra ventaja del uso de clusters es que su tamaño puede crecer al tiempo que crecen las necesidades computacionales y la disponibilidad de recursos financieros[28].

Grilla

La grilla computacional es la infraestructura computacional y de administración de datos que proveerá el apoyo electrónico para una sociedad global que involucra negocios, gobierno, investigación, ciencia y entretenimiento[16]. Se define como una infraestructura computacional que provee acceso confiable, consistente, difundido y no costoso a capacidades computacionales de alto desempeño[29]. Ha sido identificada como aquella con mayores beneficios potenciales para la bioinformática, particularmente en el área del análisis de genomas y la genómica comparativa[30, 31].

La analogía de la computación en grilla corresponde a la malla eléctrica. La meta final es que un día sea posible conectar todos los dispositivos en la grilla computacional para acceder a otros recursos de forma transparente, de la misma manera como se hace con la malla eléctrica hoy en día[32], es decir, los usuarios pueden consumir recursos computacionales tan fácil como conectando sus dispositivos a cualquier toma de red. Los participantes en las organizaciones virtuales agregan sus recursos disponibles a un pool compartido y los usan de acuerdo a sus necesidades[33].

Los problemas inherentes a la conducción de colaboración geográficamente dispersa y multidisciplinaria brindó a los investigadores experiencia en coordinación y distribución, dos conceptos fundamentales en la computación en grilla.

Wide-area year (I-WAY) es considerada la primera grilla moderna, desarrollada como un proyecto experimental de demostración. En 1995, durante una conferencia sobre supercomputación, los pioneros en investigación se reunieron para añadirse al proyecto en un ambiente distribuido nacionalmente con más de 17 sitios interconectados. Más de 60 aplicaciones fueron desarrolladas para la conferencia y ejecutadas sobre la I-WAY, así como una infraestructura rudimentaria de software para grilla, con el fin de proveer acceso, seguridad, coordinación de recursos y otras actividades. Dicho evento se convirtió en la semilla para la primera generación de investigadores y proyectos referentes a grillas computacionales.

I-WAY abrió la puerta para actividades considerables en el desarrollo de software para computación en grilla. Los proyectos de infraestructura Globus y Legion exploraron aproximaciones para proveer infraestructura básica a nivel de sistema, mientras que el proyecto Condor experimentó en el tema de planificación de alto rendimiento.

A finales de 1990, los investigadores en computación en grilla se asociaron en el Grid Forum, expandiéndose posteriormente en el Global Grid Forum, en donde muchas de las investigaciones más recientes están evolucionando para convertirse en estándares base para el futuro de las grillas computacionales.

Actualmente la grilla se ha convertido en un tema global, con colaboración mundial entre investigadores de Estados Unidos, Europa y Asia-Pacífico. Agencias patrocinadoras, vendedores comerciales, investigadores académicos, centros nacionales y laboratorios se han reunido para formar una comunidad de amplia experiencia con enormes intereses en construcción de grillas computacionales[16].

Las ciberinfraestructuras hoy proveen un ambiente global distribuido que permite ciencia electrónica con datos, conocimiento y computo intensivo multidisciplinario y geográficamente disperso[33]. La computación en grilla es una de las ciberinfraestructuras más atractivas para alcanzar la escalabilidad en el poder de computo necesario en la bioinformática, pues permite compartir recursos tales como computadores, bases de datos, aplicaciones, servicios y conocimiento en una forma segura; también es atractiva para la formación de comunidades conocidas como organizaciones virtuales[33, 34].

Mediante la computación en grilla se han tratado diversas clase de problemas, como la optimización multi-objetivo mediante búsqueda enumerativa[35], comparación de genomas microbiales[29, 36], entre muchos otros.

Computación de Alto Desempeño en Colombia y en el Mundo

El proyecto TOP500 inició en el año 1993 con el objetivo de proveer bases confiables para detectar y hacer seguimiento a las tendencias en la computación de alto desempeño. La última lista generada por top500, en junio de 2008, muestra el más alto avance en la computación de alto

desempeño hasta el momento: el rendimiento de más de 1 petaflop/s. este límite ha sido alcanzado por el sistema Roadrunner, construido por IBM para el departamento de Energía de los Estados Unidos

El sistema Roadrunner está construido con versiones avanzadas del procesador usado en el Sony PlayStation 3, y ha desplazado al sistema BlueGene/L, el cual, con un rendimiento de 478.2TFop/s, había ocupado el puesto número uno desde noviembre de 2004; cuenta con 73.728Gb en memoria y 212.992 procesadores PowerPC 440 700 MHz. IBM, y su sucesor, el BlueGene/P, escala al menos a 262.144 nodos con procesadores quadcore, con un pico de rendimiento de 3.56 petaflops [37, 38].

Sobre el sistema IBM Blue Gene/L, concebido originalmente para ser usado en aplicaciones relacionadas con las ciencias de la vida, se han ejecutado numerosos problemas concernientes a la biología molecular computacional como la identificación de posibles medicamentos[26], dinámica molecular *ab initio*[39], entre otras.

En Suramérica, Petrobrás en Brasil ocupa el puesto número uno con un cluster HP de 1.024 procesadores capaz de desarrollar 6,21 TFop/s. Sin embargo, el solo número de procesadores no indica el rendimiento del sistema: la agencia meteorológica de Japón cuenta con dos clusters, cada uno de 80 procesadores Power5+ de 2.1Ghz capaces de desarrollar 9.036TFop/s[40].

Retomando el reporte de Top500, los seis primeros puestos de Suramérica los ocupa Brasil, seguido por Venezuela en el séptimo lugar, luego nuevamente Brasil. En general, de los 22 lugares que tiene Suramérica, 16 le corresponden a Brasil, 4 a Venezuela, 1 a Perú y el último lugar a Colombia. El único sistema de cómputo de alto desempeño reportado por Colombia le pertenece a Colombia Móvil S.A, empresa que en el 2004 obtuvo el puesto 476 con un sistema HP Integrity Superdome con 128 procesadores, y un pico de 768 GFlops. Sin embargo, actualmente TOP500 señala no tener registros de este sistema[40].

Algunos países en Suramérica han publicado en sus páginas institucionales información sobre sus sistemas de cómputo de alto desempeño y proyectos asociados: En Brasil se tiene el proyecto: AMSUD – GBRAMS. South-American Grid for the Brazilian Regional Atmospheric Modeling System[41]. STIC-Amsud es un programa de cooperación científico-tecnológica en el que participan Francia, Argentina, Brasil, Chile, Perú y Uruguay y cuyo objetivo es promover y fortalecer la colaboración y la creación de redes de investigación y desarrollo en el ámbito de las ciencias y tecnologías de la información y comunicación (CTICs), a través de la presentación de proyectos conjuntos[42].

En Chile, la Fundación Ciencia para la Vida (organización creada para promocionar la adopción y uso de la innovación científica) adoptó la plataforma de computación de alto desempeño de Microsoft para incrementar su capacidad de procesamiento en investigaciones de alto impacto para el desarrollo científico del país[43].

A su vez, en la Universidad de Chile se encuentra el centro de modelamiento matemático, encargado de usar las matemáticas para la solución de problemas de las ciencias, la industria y las políticas públicas. Desde el 2005 cuenta con un sistema de cómputo de alto desempeño conformado por 16 servidores HP con 32 procesadores Intel Itanium II[44, 45].

En Ecuador, la Universidad de San Francisco de Quito cuenta con un laboratorio de Computación de Alto Desempeño, en el Colegio de Ciencias e Ingeniería, conformado por 6 servidores Sun Fire V240 y 4 estaciones de trabajo Sun Blade 1500[46].

En Colombia, luego de la iniciativa del gobierno con la Agenda de Conectividad, se ha dado especial énfasis a la Computación en Grilla. Es así como se generó la dinámica en torno a las redes de alta velocidad para conectar los sistemas de cómputo de alto desempeño en todo el país, y a su vez, con el mundo.

"La Red Universitaria Metropolitana de Bogotá, RUMBO, tiene como objeto general agrupar en una red de alta velocidad a las instituciones de educación superior de Bogotá, con el objeto de participar activamente en el proyecto Internet 2, promovido por la Agenda de Conectividad a través de la Red Académica de Tecnología Avanzada RENATA"[47].

Por su parte, "RENATA es la red de tecnología avanzada que conecta, comunica y propicia la colaboración entre las instituciones académicas y científicas de Colombia con las redes académicas internacionales y los centros de investigación más desarrollados del mundo. El gran valor agregado de RENATA radica en el poder de comunicación y colaboración entre sus miembros […] es administrada por la Corporación RENATA, de la cual son miembros las Redes Académicas Regionales, el Ministerio de Comunicaciones, el Ministerio de Educación y Colciencias"[48].

Gracias a lo anterior, las universidades han empezado a organizarse para construir sus sistemas de cómputo de alto desempeño, para luego ponerlos a disposición de los demás investigadores de Colombia y el mundo a través de RENATA. Sin embargo este proceso ha sido lento con referencia a otros países. Reportados se puede encontrar el grupo de Investigación y desarrollo en computación de Alto Rendimiento de la Universidad Pontificia Bolivariana en la Seccional Bucaramanga, en cuya información se observa un cluster conformado por 10 computadores de escritorio[49].

El proyecto más grande en Bogotá es el Supercomputador Distribuido de la Universidad Nacional, con 40 computadores y 6 servidores. La visión de este proyecto es unir el tiempo ocioso de los computadores de la sede Bogotá (más de 3.500) para brindar tiempo de procesamiento a los investigadores que lo requieran[50].

En la Pontificia Universidad Javeriana el grupo SIDRe[51] del Departamento de Ingeniería de Sistemas ha trabajado sobre los clusters computacionales y de hecho en el año 2002 montó el primer sistema de cómputo de alto desempeño en la Universidad, conocido como el Cluster ASMA, conformado por 12 estaciones de trabajo. Adicionalmente, se cuenta con un cluster de 6 servidores Dell, cada uno con dos procesadores dobles, lo cual ofrece 24 nodos de procesamiento.

Actualmente dicho grupo trabaja en el tema de la computación de alto desempeño en conjunto con otras universidades de Bogotá.

Bioinformática y Biología Computacional

La bioinformática es el campo de las ciencias en el que la biología, las ciencias de la computación y la tecnología de la información convergen para formar una sola disciplina. Abarca cualquier herramienta o método computacional usado para administrar, analizar y manipular grandes conjuntos de datos biológicos y su meta final es permitir el descubrimiento de nuevo conocimiento biológico así como crear una perspectiva global desde la cual unificar los principios de la biología[52]. Actualmente, el proceso de analizar e interpretar datos se conoce como biología computacional[53].

Con el progreso del proyecto Genoma Humano, cada vez más investigadores están involucrados en el dominio de tópicos de la bioinformática, como la predicción de secuencias en un genoma, diseño y descubrimiento de drogas, o la cura de enfermedades. Estudiar la causa de una enfermedad desde el punto de vista de la expresión de los genes puede requerir de demasiado trabajo y recursos si la experimentación biológica se utiliza desde el principio. Una aproximación desde la bioinformática permite disminuir el rango de variables y obtener las referencias más próximas y así reducir el desperdicio por ensayo y error del proceso experimental.

Particularmente, como los problemas en bioinformática están relacionados con cómputos y datos masivos, es necesario tener buenos algoritmos y computación de alto desempeño[54]. Sin duda, la bioinformática puede ser considerada como una de las fuerzas que están cambiando la forma en la cual las ciencias son conducidas en la actualidad[33].

La bioinformática ha tenido muchos problemas computacionales, lo cual ha estimulado la sinergia en la investigación y el desarrollo en técnicas en áreas de minería de datos, estadística, análisis de imágenes y patrones y visualización[55].

Dado que la bioinformática emerge como una clase importante de aplicaciones de la computación científica, se evidencia que los avances en este campo tienen su mayor aplicación en la investigación farmacéutica, predicción de la estructura de proteínas y el desarrollo de terapias basadas en la genética. Se espera que el desempeño de las aplicaciones bioinformáticas se convierta en un factor importante que defina el futuro de los sistemas de cómputo de alto desempeño[56].

Dinámica Molecular Computacional

En la actualidad se conoce la estructura tridimensional de solo un pequeño conjunto de proteínas debido a que para encontrarla, se necesita experimentación en laboratorio, la cual resulta costosa. Determinar dicha estructura en forma computacional es un problema importante pues acelera la investigación y reduce el análisis experto[57].

La simulación por mecánica y dinámica molecular es el nombre de la técnica en la cual las trayectorias de un sistema de partículas interactuantes se calcula resolviendo numéricamente sus ecuaciones de movimiento[42, 58]. Ha emergido como una herramienta viable y una poderosa opción de la bioquímica para la comprensión de las interacciones y particularmente para el diseño asistido por computador de drogas basado en la estructura molecular[59].

Por dinámica molecular no solo se simulan procesos biológicos. También pueden simularse otros procesos físico-químicos como la formación de clusters en el proceso de fabricación de nanotubos de carbono[60].

La dinámica molecular permite estudiar la dinámica de macromoléculas grandes, incluyendo los sistemas biológicos. Los eventos dinámicos juegan un papel importante en controlar procesos que afectan las propiedades funcionales de las biomoléculas. El diseño de drogas es comúnmente usado en la industria farmacéutica para probar las propiedades de una molécula en el computador sin necesidad de sintetizarla (lo cual es mucho más costoso)[57]. Ha sido ampliamente usada para simular propiedades de los líquidos, sólidos, y moléculas en investigaciones de diversas disciplinas, incluyendo las ciencias de los materiales, tecnología biológica, transferencia de calor y masa, entre otras[61].

La dinámica molecular es muy importante para la investigación biomédica pues hace posible simular el comportamiento de una macromolécula biológica *in silico*. De todas maneras es un proceso costoso, pues la simulación de algunos nanosegundos de dinámica para una macromolécula grande como una proteína, puede tardar mucho tiempo debido al gran número de operaciones que se necesitan para resolver las ecuaciones de Newton[62].

Aunque la dinámica molecular es simple en su configuración conceptual, es muy demandante en tiempo de computación si se intentan simular sistemas muy grandes o para tiempos muy extensos[28].

Diseñar una simulación de dinámica molecular puede frecuentemente encontrar límites en el poder computacional. El tiempo de simulación es dependiente del tiempo de integración, por tanto, para hacer el mejor uso del tiempo en computador debería escogerse un tiempo de integración grande. Pero un tiempo de integración grande causa imprecisión en el proceso de integración, y errores por discretización[62].

Con el entendimiento del funcionamiento de los sistemas biológicos, la importancia de las simulaciones biomoleculares se ha incrementado significativamente[63]. Dichas simulaciones

originalmente se enfocaban en dinámica de proteínas, diseño de drogas y plegamiento de proteínas. Recientemente, el progreso se ha enfocado en simular los cambios conformacionales en complejos proteínicos grandes. La atención se centra en entender la expresión de los genes, la cual es el principal punto de atención de la biología molecular[18].

El modelado molecular es una metodología clave para la investigación y desarrollo en bionanotecnología, pues provee imágenes nanoescalares con una resolución atómica e incluso electrónica, predice la interacción nanoescalar de combinaciones poco familiares entre materiales biológicos e inorgánicos, y evalúa estrategias para el rediseño de polímeros con usos nanotecnológicos. La nanoescala es muy pequeña para el microscopio óptico pero muy grande para la cristalografía de rayos X; muy heterogénea para la resonancia magnética nuclear y muy "húmeda" para la microscopía electrónica. Esto hace que la simulación por computador brinde la posibilidad de tener imágenes de fenómenos nanoescalares, todo gracias a los avances en software y hardware[1].

Como una primera aproximación, el grupo SIDRe del Departamento de Ingeniería de Sistemas de la PUJ, desarrolló un simulador molecular didáctico del proceso de creación de una proteína[64], dentro del proyecto ASMA[65].

Por otra parte, en la Pontificia Universidad Javeriana, el grupo de Bioquímica Computacional Estructural y Bioinformática de la Facultad de Ciencias trabaja en el tema de la Mecánica y Dinámica molecular.

Existen muchos programas para dinámica y mecánica molecular, e identificación de sitios de interacción de drogas como EUDOC[66], Car-Parrinello[67]. En el presente trabajo se utilizó el programa NAMD [68] pues ha sido ampliamente usado en los problemas que involucran dinámica molecular, gracias a su escalabilidad y rendimiento; con él se han realizado varias simulaciones como bicapas mixtas DMPC/DPPC [69], reconocimiento de receptores [70], bicapas de lípidos [71] entre muchas otras. También ha sido utilizado para hacer comparaciones de desempeño entre sistemas como el sistema BlueGene/L [63], sistemas multi core [72], etc. Actualmente, el más alto grado de escalabilidad en la literatura publicada ha sido el alcanzado por NAMD[73].

DESCRIPCIÓN DEL EXPERIMENTO

El experimento consiste en la minimización de una serie de proteínas utilizando cinco ambientes computacionales tipo cluster, con computadores agrupados por su capacidad y escalados en su cantidad de procesadores, mas un quinto ambiente que une la totalidad de procesadores de los cuatro ambientes mencionados anteriormente.

Por su parte, las proteínas tienen resultados experimentales y varían tanto en su cantidad de átomos como en su peso molecular. Tres de ellas además han sido usadas para realizar pruebas de rendimiento de diferentes sistemas de cómputo de alto desempeño a nivel mundial.

Inicialmente se realizó un análisis bibliográfico que permitió una profundización en el conocimiento de las arquitecturas para procesamiento paralelo y distribuido existentes enfocadas al contexto de este trabajo. De este análisis se escogieron los sistemas de procesamiento en cluster y en grilla. Respecto al cluster computacional, se implementó uno tipo Beowulf, mientras que para la grilla se implementó el sistema condor[74].

Posteriormente, tanto en la grilla como en el cluster se implementó el programa de dinámica molecular NAMD y el graficador VMD[75]. Para verificar su correcta instalación y funcionamiento, se ejecutaron los casos de prueba sugeridos en la literatura.

Dado que los problemas de dinámica y mecánica molecular son altamente acoplados, es decir, no se pueden dividir en subproblemas solucionables independientemente, luego de investigar y de implementar la grilla computacional se encontró que su labor es hallar los computadores disponibles en el momento de la ejecución, y con ellos configurar un cluster sobre el cual se solucionará del problema.

Lo anterior quiere decir que la grilla ensamblará automáticamente el cluster para sobre él empezar a trabajar. La configuración de dicho cluster dependerá de la elección de la grilla, lo cual le añade una variable aleatoria al experimento. Por tal razón se eliminó el uso de la grilla y el trabajo se centró en el análisis sobre diferentes clusters conformados con los recursos disponibles en el Departamento de Ingeniería de Sistemas de la Pontificia Universidad Javeriana. Por supuesto, no se desestimula el uso de las grillas computacionales, solamente se busca la reducción de variables intervinientes.

SELECCIÓN DE PROTEINAS

El banco de datos más grande de recursos acerca de la información estructural de las proteínas es el Protein Data Bank (PDB) del Research Collaboratory for Structural Bioinformatics (RCSB)[76]. Los

archivos con las coordenadas de los átomos que conforman las siguientes proteínas fueron descargados de dicho sitio, y para su selección se tuvieron en cuenta los siguientes criterios:

- Tipo de Molécula/cadena (Molecule/Chain Type): se seleccionaron aquellas que sólo tuvieran proteína en su estructura pero no DNA ni RNA. Esto con el fin de tener el menor número de archivos de parámetros posible.
- Peso Molecular (Molecular Weight): para este criterio se establecieron tres grupos de proteínas:
    - De entre 2.000u y 20.000u.
    - De entre 40.000u a 60.000u.
    - De entre 60.000u a 1'000.000u.

  Estos rangos se establecieron tomando en cuenta que el peso más pequeño reportado en PDB a la fecha de consulta era de 2.000u, y el más grande era de 1'000.000u, y que el peso de la mayoría de las proteínas reportadas en PDB está en el intervalo de 2.000u a 60.000u.

- Método Experimental (Experimental Method): Que tuvieran datos experimentales hallados por Resonancia Magnética Nuclear (NMR).
- Tiene ligandos (has ligands): que no tuvieran ligandos.

De acuerdo a lo anterior, se obtuvo:

Peso molecular entre 2.000u y 20.000u: esta consulta arrojó 2.377 candidatas, de las cuales se escogieron:

- 1A5E[77]: TUMOR SUPPRESSOR P16INK4A. 2.103 átomos antes de hidratar. Peso: 16.554,80u. 41.163 átomos en la simulación. 156 residuos, 38% Hélices (7 hélices, 60 residuos).
- 1AFO[78]: GLYCOPHORIN A. 1.322 átomos antes de hidratar. 37.151 átomos en la simulación. Peso: 8.908,92u. Dos cadenas: A: 40 residuos, 67% hélices (1 hélice, 27 residuos), B: 40 residuos, 67% hélices (1 hélice, 27 residuos).
- 1AOY[79]: ARGININE REPRESSOR. 1.239 átomos antes de hidratar. 29.595 átomos en la simulación. Peso: 8.722,19u. 78 residuos, 41% hélices (3 hélices, 32 residuos), 10% hojas beta (2 cadenas, 8 residuos).

Peso molecular entre 40.000u y 60.000u: esta consulta arrojó 11 candidatas, de las cuales se escogieron:

- 1EZO[80]: MALTOSE-BINDING PERIPLASMIC PROTEIN. 5.735 átomos antes de hidratar. 76.004 átomos en la simulación. Peso: 40.741,50u. 370 residuos, 39% hélices (16 hélices, 145 residuos), 9% hoja Beta (15 cadenas, 34 residuos).
- 2HYM[81]: 6.063 átomos antes de hidratar. 99.516 átomos en la simulación.
    - Polymer: 1 Molecule: Soluble IFN alpha/beta receptor Chains: A
    - Polymer: 2 Molecule: Interferon alpha-2 Chains: B

Peso: 43.587,80u. 2 cadenas: A: 212 residuos, 1% hélice (1 hélice, 3 residuos), 42% hojas beta (17 cadenas, 91 residuos). Cadena B: 165 residuos, 62% hélices (9 hélices, 103 residuos).

Peso molecular entre 60.000u y 1'000.000u: esta consulta arrojó tres candidatas, de las cuales se escogió:

- 2JQX[82]: Malate synthase G. 11.282 átomos antes de hidratar. 98.650 átomos en la simulación. Peso: 80.566,10u. 723 residuos, 46% hélices (30 hélices, 339 residuos), 14% hojas beta (35 cadenas, 104 residuos).

Adicionalmente se utilizaron tres pruebas de diferentes sistemas de cómputo de alto desempeño reportados, cuyos archivos de configuración fueron descargados de la página de utilidades de NAMD[1]:

- APOA1[83] - Apolipoproteína A1 -: modelo de una partícula de lipoproteína de alta densidad (HDL)[84]. 92.224 átomos en la simulación y una mezcla de proteínas, lípidos y agua; ApoA1 es una muestra representativa de una simulación moderna de tamaño moderado[63].
- STMV[85]: SATELLITE TOBACCO MOSAIC VIRUS. 1'066.628 átomos en la simulación. Peso: 24.218,20u. 3 cadenas: A: 159 residuos, 6% hélices (3 hélices, 10 residuos), 35% hojas beta (9 cadenas, 57 residuos). B: RNA10 nucleótidos poli A. C: RNA 10 nucleótidos poli U.
- F1-ATPASE[86]: ATP-Sintetasa. 327.506 átomos en la simulación.

VARIABLES INDEPENDIENTES

Como otra variable independiente, aparte del peso molecular de las proteínas escogidas, se tomó el número de procesadores a utilizar para la minimización, y la configuración de los sistemas de procesamiento. En la sección "Equipos de Cómputo" se detallarán los diferentes sistemas de procesamiento utilizados y la cantidad de procesadores de cada uno.

VARIABLES DEPENDIENTES

La principal variable dependiente es el tiempo (en segundos) que toma el sistema en simular por dinámica molecular a cada una de las proteínas. Con base en este tiempo se deriva la información porcentual de mejora en el desempeño del sistema al añadir un procesador, con respecto al procesador añadido en el paso anterior y con respecto a un solo procesador. A su vez, se deriva la información referente al Speed Up y la eficiencia del sistema.

[1] Archivos de configuración descargados de: http://www.ks.uiuc.edu/Research/namd/utilities/

SISTEMAS DE CÓMPUTO

El experimento se llevó a cabo en un cluster con máquinas heterogéneas, las cuales se dividieron en conjuntos por sus características y ubicación, y se describen a continuación:

RACK INGENIERÍA

Todos los servidores son Dell PowerEdge 1950 con dos procesadores Intel Xeon Dual Core. Se encuentran conectadas entre sí por un switch a 1Gb. Tienen instalado el sistema operativo Linux CENTOs 5.

Nombre del Servidor	Velocidad del Procesador	Memoria RAM	Capacidad de Almacenamiento
Cluster001	2.66 Ghz	16 Gb	600Gb
Cluster002	3.00 Ghz	2 Gb	73 Gb
Cluster003	3.00 Ghz	2 Gb	300 Gb
Cluster004	3.00 Ghz	2 Gb	600 Gb
Cluster005	3.00 Ghz	2 Gb	600 Gb
Cluster006	3.00 Ghz	2 Gb	600 Gb

Tabla 1. Características de cada uno de los servidores del Rack

PC DE ESCRITORIO

Computador Dell Optiplex 745n con procesador Intel Core 2 Duo de 2.4 Ghz, 2 Gb en RAM, 500 Gb en disco duro y tarjeta de red a 1 Gb. Este equipo tiene instalado como sistema operativo Linux CENTOs 5. Este equipo no se utilizó como ambiente separado; únicamente fue incluido en la unión de todos los sistemas, llamado supercluster.

SALA BD

23 computadores Lenovo ThinkCentre con procesador Intel Core 2 Duo de 2.0 Ghz, 2 Gb en RAM, 80 Gb en disco duro y tarjeta de red a 1 Gb. Todos los equipos tienen instalado como sistema operativo Linux Mandriva 7.

SALA B

19 computadores Dell Optiplex GX 260 con procesador Intel Pentium IV de 2.66Ghz, 512 Mb en RAM, 40 Gb en disco duro y tarjeta de red a 1Gb. Estos equipos tienen instalado como sistema operativo Linux Mandriva 7.

SALA A

9 computadores Compaq Evo con procesador Intel Pentium IV de 1.7Ghz, 1 GB en RAM, 40 Gb en disco duro y tarjeta de red a 100 Mb. Estos equipos tienen instalado como sistema operativo Linux Mandriva 7.

SUPERCLUSTER

Es un sistema que integra a los mencionados anteriormente, de manera que en total tiene 101 procesadores. El orden en que se utilizan los procesadores es: primero los del Rack Ingeniería, luego el PC de escritorio, Sala BD, Sala B y finalmente Sala A.

MONTAJE DEL EXPERIMENTO

Gracias al soporte del personal del Departamento de Ingeniería de Sistemas, se contaba con la previa instalación del sistema operativo Linux en los servidores Dell Poweredge (conjunto denominado Rack Ingeniería), y en las máquinas de las salas A, B y BD.

Los programas NAMD y VMD fueron instalados en el computador Dell Optiplex 745, equipo en el cual se hizo el desarrollo del software necesario en el presente trabajo, junto con la preparación de las proteínas a simular. Adicionalmente, NAMD y VMD también fueron instalados en el servidor 1 del Rack Ingeniería, pues éste servidor es el nodo maestro desde el cual se lanzaron las simulaciones hacia los demás ambientes computacionales, de tal forma que todos los resultados quedaran allí almacenados.

Para probar la correcta instalación de los sistemas de computo, bien sea en modo stand alone, como paralelo y distribuido, se elaboró un tutorial de ejecución del programa NADM y el visor VMD (a partir de la guía del usuario de NAMD), haciendo uso de una molécula pequeña cuyos archivos necesarios pueden descargarse del sitio de NAMD. Dicho tutorial fue probado con los estudiantes de maestría y doctorado en ciencias biológicas, en el curso Campos de Fuerza, y está disponible en http://sophia.javeriana.edu.co/modulor/namd/tutorial.pdf.

Adicionalmente se hicieron pruebas sobre un cluster Clustermatic[87], el cual provee optimizaciones a nivel de comunicación en su protocolo de red. Es un tipo de cluster bastante específico pero bastante eficiente. Sin embargo, este cluster no se utilizó en este trabajo, pues su configuración inicial y puesta en marcha de las máquinas no era viable en las salas de cómputo disponibles.

PREPARACIÓN DE LAS PROTEINAS PARA SU MINIMIZACIÓN

Una vez descargados los archivos de coordenadas de los átomos que conforman las proteínas, utilizando scripts escritos en Tool Command Language (TCL) y ejecutados en la consola del programa Visual Molecular Dynamics (VMD)[75], a las proteínas 1A5E, 1AFO, 1AOY, 1EZO, 2HYM y 2JQX se les aplicó el siguiente procedimiento:

1. Se eliminaron las moléculas de agua.
2. Debido a que estos archivos fueron obtenidos experimentalmente, es posible que hagan falta átomos, típicamente los de hidrógeno. Por esta razón con VMD se completaron los átomos faltantes.
3. Se generó el archivo de estructura PSF. Este archivo indica los enlaces entre los átomos.
4. Cada proteína se hidrató en un cubo de agua, cuyas dimensiones comprenden 15Å por fuera del átomo más externo en los tres ejes. Las proteínas muestran su estructura intrínseca y su función en un entorno nativo únicamente; por esta razón es necesario este paso[88].

Los scripts utilizados para hacer la preparación de las proteínas pueden consultarse en el capítulo “Desarrollo de software especializado para problemas específicos”.

CONFIGURACIÓN DE LA MINIMIZACIÓN

Para la minimización por dinámica molecular se utilizó el programa NAMD[68], desarrollado por el Grupo de Biofísica Teórica y Computacional en el Instituto Beckman para el Avance de la Ciencia y la Tecnología de la Universidad de Illinois (Urbana-Champaign). Como entradas para el programa se requiere el archivo de coordenadas de los átomos y el archivo de estructura generados en el paso anterior. Adicionalmente, se necesita el archivo de configuración de la minimización, el cual le va a indicar a NAMD los parámetros para realizar el trabajo. Dichos parámetros fueron:

- Hidratación con un cubo de agua en el vacío.
- No se consideran límites periódicos.
- Archivo de parámetros: par_all27_prot_lipid.inp.
- Temperatura inicial: 300 Kelvin.
- Número de pasos de tiempo entre cada escritura de resultados de energía: 2000.

- Número de pasos de tiempo entre cada escritura de resultados de presión: 2000.
- Escalamiento: 1-4 a 1Å.
- Punto de corte: 12Å.
- Switching activado a una distancia de 10Å.
- No se consideran enlaces rígidos.
- Ejecución por 6000 femtosegundos, es decir, 6 picosegundos.

Como los átomos de hidrógeno en una biomolécula vibran con un periodo de aproximadamente 10 femtosegundos, el paso de tiempo computacional debe ser de alrededor de un femtosegundo para que la integración sea estable y precisa. De todas maneras, el fenómeno de interés puede ocurrir en una escala de microsegundos de manera que una simulación para un periodo de tiempo tan corto requiere alto poder computacional[63].

Todos los parámetros de configuración pueden considerarse variables intervinientes, pero para efectos de este trabajo dichos valores no fueron modificados.

Preparación del ambiente distribuido

Para poder ejecutar una simulación utilizando NAMD en un ambiente distribuido, es necesario (luego de tener preparados los archivos inherentes a la simulación) que todos los equipos involucrados tengan la misma estructura de directorios relacionados con la minimización, que todos tengan instalado NAMD, algún programa que permita acceso remoto con protocolos como rsh o ssh, y un esquema que permita hacer conexión sin necesidad de autenticación interactiva (por ejemplo, llaves públicas y privadas). SSH es un protocolo para conexión remota segura y otros servicios de red seguros que se ejecutan sobre una red insegura[89].

Se escogió entonces el uso de ssh porque brinda mayor seguridad en la conexión, evitando que las contraseñas sean interceptadas en la red, lo cual podría ocasionar ingresos malintencionados a las máquinas. También se escogió el esquema de llaves públicas y privadas, pues es seguro y confiable.

Para poder hacer esto en los 58 computadores que estarían involucrados, se desarrollaron varios scripts que automatizan este proceso, garantizando la homogeneidad en todas las máquinas. Los scripts de este paso se explican en la sección “Desarrollo de Software Especializado para Problemas Específicos”.

Con el fin de facilitar la creación de la estructura de directorios en todas las máquinas, se desarrolló un programa en C++ que permite, dado un archivo de parámetros, crear dicha estructura y así organizar las ejecuciones y réplica de datos en las máquinas que conforman el sistema. La estructura del archivo de parámetros y la forma de ejecutar el programa se explican en la sección “Desarrollo de Software Especializado para Problemas Específicos”.

Una vez el nodo maestro tiene conexión con todas las máquinas, en ellas está creada la estructura de directorios y a su vez cada una tiene los archivos de configuración y de parámetros de las simulaciones, solo falta crear en el nodo maestro un archivo con las direcciones IP de las máquinas que se utilizarán en la minimización.

Como algunos computadores tienen más de un procesador, se copia varias veces su IP para indicar este hecho, de tal forma que al final el archivo tiene 101 líneas; es decir, un nodo equivale a un núcleo de un procesador. De acuerdo a esto, un computador con procesador Core 2 Duo equivale a tener dos nodos de procesamiento. En [90] se demuestra que una máquina con procesador de doble núcleo es equivalente a dos máquinas con procesador de un solo núcleo en términos de desempeño.

Se creó un archivo por cada sistema de procesamiento utilizado. Para el sistema Supercluster (el cual integra a todos los ambientes individuales) el archivo se creó de tal manera que los nodos están en el siguiente orden: Rack Ingeniería, PC de escritorio, Sala BD, Sala B y por último Sala A. Dicho orden se escogió de tal manera que los computadores con procesadores más lentos se utilizaran lo menos posible, es decir, primero se encuentran los nodos más rápidos hasta llegar a los nodos más lentos. A su vez, para el sistema Rack Ingeniería, se crearon dos archivos: secuencial y alternado. En el archivo secuencial, se indica que primero se deben usar todos los procesadores del primer servidor, luego todos los del segundo servidor y así sucesivamente hasta llegar a usar todos los del sexto, mientras que en el archivo alternado, se indica que primero se debe usar un procesador del primer servidor, luego un procesador del segundo servidor, y así sucesivamente hasta llegar a un procesador sexto servidor; esta secuencia se repite cuatro veces hasta usar todos los procesadores de todos los servidores.

El nodo maestro es el servidor cluster01. Para ejecutar las minimizaciones de forma desatendida, se ejecuta el archivo run.sh (generado por el programa que crea la estructura de directorios) de la siguiente manera: `nohup ./run.sh &`. Este comando permite realizar la ejecución sin que sea cancelada por el timeout que genera ssh al detectar inactividad, o que sea cancelado cuando el usuario cierra su sesión con el servidor.

Es importante aclarar que durante el tiempo de ejecución, el único programa que se estaba ejecutando en el cluster era NAMD, aparte de las rutinas del sistema. En la Figura 1 se muestra la topología de la red, con el fin de ilustrar la ubicación lógica de los sistemas de cómputo utilizados y su forma de conexión.

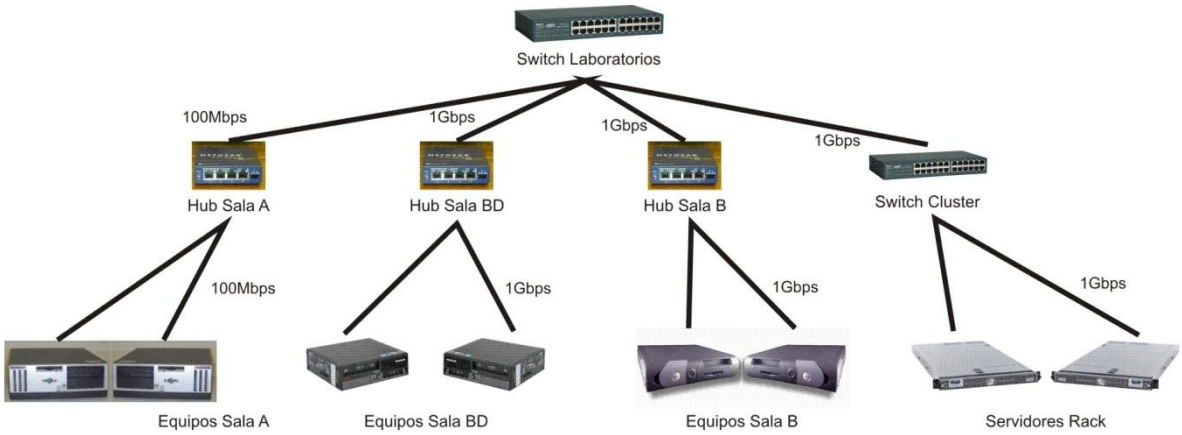


Figura 1. Topología de red

Debido a que las salas A, B y BD son utilizadas por los estudiantes y profesores en el día para su labor académica, y para no tener sobrecarga e interferencia en el tráfico de la red, la puesta en ejecución las simulaciones en los diferentes sistemas se realizó en horarios comprendidos entre las 23H y las 6:30H, teniendo en cuenta que el horario de operación de las salas es de las 7H a 21H.

Por otra parte, para garantizar que los experimentos no son aleatorios, las simulaciones en el Rack Ingeniería se hicieron cinco veces. En la sección de resultados y análisis se mostrará el resumen de las varianzas obtenidas en estas ejecuciones.

Luego de realizar las ejecuciones para cada una de las proteínas en cada uno de los ambientes mencionados anteriormente, se obtuvieron los archivos de salida que genera NAMD. Mediante un programa hecho en C++, se procesó cada uno de los archivos para extraer el tiempo total empleado en la minimización; con este dato, el programa construyó un archivo de resumen separado por tabuladores, donde se muestra para cada proteína y para cada ambiente, el tiempo empleado en la minimización usando diferente cantidad de procesadores.

Finalmente, mediante un programa escrito en Visual Basic for Applications, en Excel, se construyeron las tablas para calcular el porcentaje de mejora en el tiempo de minimización al añadir cada procesador a la ejecución, el porcentaje de mejora que se obtuvo respecto al procesador anterior, el speed up, la eficiencia y la diferencia en las mejoras de porcentajes.

La descripción de los programas mencionados anteriormente se encuentra en la sección “Desarrollo de Software Especializado para Problemas Específicos”.

RESULTADOS Y ANÁLISIS

Resumen de Varianzas

En la siguiente tabla se resumen los valores máximo, mínimo, promedio y mediana de las varianzas obtenidas luego de ejecutar las simulaciones cinco veces en el Rack de ingeniería. Se escogió este ambiente por ser controlado, y tener tráfico independiente de datos lo cual minimiza el ruido en las mediciones.

Proteína	Varianza Mínima	Varianza Máxima	Promedio de Varianzas	Mediana de Varianzas
1A5E	0,13%	10,81%	3,48%	2,90%
1AFO	0,40%	12,56%	3,38%	2,53%
1AOY	0,36%	7,42%	2,47%	2,37%
1EZO	0,19%	9,15%	2,99%	2,41%
2HYM	0,29%	16,56%	3,33%	2,36%
2JQX	0,22%	8,02%	2,21%	2,08%
APOA1	0,07%	9,67%	1,87%	0,85%
F1ATPASE	0,12%	10,27%	1,87%	0,88%
STMV	0,25%	12,15%	2,91%	2,23%

Tabla 2. Resumen de varianzas para el Rack de Ingeniería en configuración secuencial

Proteína	Varianza Mínima	Varianza Máxima	Promedio de Varianzas	Mediana de Varianzas
1A5E	0,24%	7,11%	2,34%	2,18%
1AFO	0,35%	11,56%	2,87%	2,22%
1AOY	0,13%	7,06%	2,40%	2,18%
1EZO	0,29%	12,52%	3,22%	2,70%
2HYM	0,28%	14,73%	2,79%	1,98%
2JQX	0,10%	14,50%	2,25%	1,59%
APOA1	0,11%	15,10%	2,77%	1,17%
F1ATPASE	0,08%	11,88%	2,61%	1,31%
STMV	0,18%	11,51%	4,03%	2,61%

Tabla 3. Resumen de varianzas para el Rack de Ingeniería en configuración alternada

Las varianzas en las dos configuraciones son similares (ligeramente mayor en la configuración alternada) y en promedio son bastante bajas, lo cual indica que hay baja aleatoriedad en el experimento. Se puede observar que a pesar de existir en algunos casos varianzas altas, los valores de la media y la mediana cercanos al 3% confirman que en la mayoría de ejecuciones se tuvieron tiempos de ejecución similares. Teniendo esto en cuenta, para los demás experimentos se realizó solamente una ejecución.

Respecto a los ambientes de ejecución

Como guía para un usuario que desee realizar un procesamiento como el descrito en este trabajo, es interesante tener la capacidad de estimar cuánto tardará en obtener sus resultados conociendo el ambiente en el que está trabajando. Es por esto que se han obtenido algunas ecuaciones que permiten estimar este tiempo, en segundos. Estas ecuaciones fueron obtenidas por la regresión aplicada a los resultados de 2JQX y de STMV, cuyas características pueden consultarse en la Tabla 4. Debe tenerse en cuenta que la variable *x* en las ecuaciones corresponde a la cantidad de procesadores.

En un ambiente como el de la Sala A, con procesadores Pentium IV de 1.7Ghz o similar, 1 Gb en RAM y conexión a red de 100Mbps, las ecuaciones halladas fueron:

Para 2JQX: $T(x) = 57.302x^{-1,185}$ con un R^2 igual a 0,9857.

Para STMV: $T(x) = -102{,}92x^5 + 3.698{,}6x^4 - 52.377x^3 + 365.671x^2 - 1E6x + 2E6$ con un R^2 igual a 0.9964

En un ambiente como el de la sala B, con procesadores Pentium IV de 2.66Ghz o similar, 512 Mb en RAM y conexión a red de 100Mbps, las ecuaciones halladas fueron:

Para 2JQX: $T(x) = -0{,}2274x^5 + 13{,}806x^4 - 327{,}5x^3 + 3.815{,}9x^2 - 22.098x + 55.020$ con un R^2 igual a 0,9924.

Para STMV: $T(x) = -2.604{,}7x^3 + 138.606x^2 - 2E6x + 1E7$ con un R^2 igual a 0.9481

En un ambiente como el de la sala BD, con procesadores Core 2 Duo de 2.0Ghz o similar, 2 Gb en RAM y conexión a red de 1Gbps, las ecuaciones halladas fueron:

Para 2JQX: $T(x) = 0{,}0002x^6 - 0{,}0319x^5 + 2{,}0601x^4 - 66{,}98x^3 + 1.151{,}7x^2 - 9.981x + 37.131$ con un R^2 igual a 0,9832.

Para STMV: $T(x) = 0{,}0002x^6 - 0{,}031x^5 + 2{,}0495x^4 - 69{,}275x^3 + 1.265{,}5x^2 - 12.050x + 50.861$ con un R^2 igual a 0.9963

En un ambiente como el del Rack Ingeniería, con procesadores Intel Xeon Dual Core 3Ghz o similar, 2Gb en RAM y conexión a red de 1Gbps, las ecuaciones halladas fueron:

Para 2JQX: $T(x) = -0{,}0196x^5 + 1{,}6417x^4 - 52{,}956x^3 + 814{,}33x^2 - 5.988{,}8x + 19.354$ con un R^2 igual a 0,9962

Para STMV: $T(x) = 0{,}1611x^4 - 11{,}047x^3 + 281{,}08x^2 - 3.257{,}8x + 17.542$ con un R^2 igual a 0.9876

Por otra parte, a pesar de haber ejecutado los experimentos en un ambiente altamente heterogéneo, se presenta una ecuación que puede utilizarse de guía para estimar a partir de la cantidad de átomos, cuantos procesadores pueden ser necesarios para lograr el tiempo mínimo:

$N(x) = 0{,}3577x^3 - 4{,}1043x^2 + 14{,}157x + 12{,}913$ con un R^2 igual a 0.9856

Siendo *x* la cantidad de átomos involucrados en el experimento.

Podría entonces tomarse el resultado de la ecuación N(*x*) como parámetro para la ecuación T(*x*) de acuerdo al ambiente computacional de interés y estimar así cuantos procesadores se deberían tener y cuanto tiempo en segundos se tardaría en obtener el resultado.

Respecto a las características de las proteínas

A continuación se presenta una tabla que resume la información de cada proteína:

	peso molecular (u)	#residuos	# átomos antes de hidratar	# átomos en la simulación	%hélices	# hélices	# residuos en hélice	% Hojas Beta	# cadenas	# residuos en hojas Beta
1AOY	8.722,19	78	1.239	29.595	41%	3	32	10%	2	8
1AFO	8.908,92	A: 40 B: 40	1.322	37.151	A: 67% B: 67%	A: 1 B: 1	A: 27 B: 27	0	0	0
1A5E	16.554,80	156	2.103	41.163	38%	7	60	0	0	0
1EZO	40.741,50	370	5.735	76.004	39%	16	145	9%	15	34
APOA1	30.778,00	267		92.224	70%	7	188	4%	3	11
2JQX	80.566,10	723	11.282	98.650	46%	30	339	14%	35	104
2HYM	43.587,00	A: 212 B: 165	6.063	99.516	A: 1% B: 62%	A: 1 B: 9	A: 3 B: 103	A: 42% B: 0%	A: 17 B: 0	A: 91 B: 0
F1ATPASE	353.498,53	528		327.506	34%	21	178	20%	17	108
STMV	24.218,20	A: 159 B: RNA 10 poli A C: RNA 10 poli U		1.066.628	A: 6%	A: 3	A: 10	A: 35%	A: 9	A: 57

Tabla 4. Información de las proteínas utilizadas en la simulación.

Se puede observar que hay proteínas con una sola cadena o con dos cadenas de nucleótidos. También se tiene una minimización que comprende una cadena de nucleótidos con dos cadenas de RNA. Las proteínas tienen diferente composición: todas tienen al menos dos hélices alfa, pero solo algunas tienen hojas plegadas Beta. De igual manera, porcentualmente hablando, la composición de la proteína en término de sus

estructuras es bastante diversa: APOA1 tiene 7 hélices con 188 residuos, lo cual hace que el 70% de la proteína tenga esta estructura, mientras que por ejemplo, 2JQX tiene 30 hélices con 339 residuos, pero esta cantidad solo cubre el 46% de la proteína.

En la siguiente tabla se muestran los tiempos máximos y mínimos obtenidos luego de ejecutar la minimización. La información corresponde a la media móvil de los datos utilizando un periodo igual a tres (los datos completos se encuentran disponibles en el CD anexo):

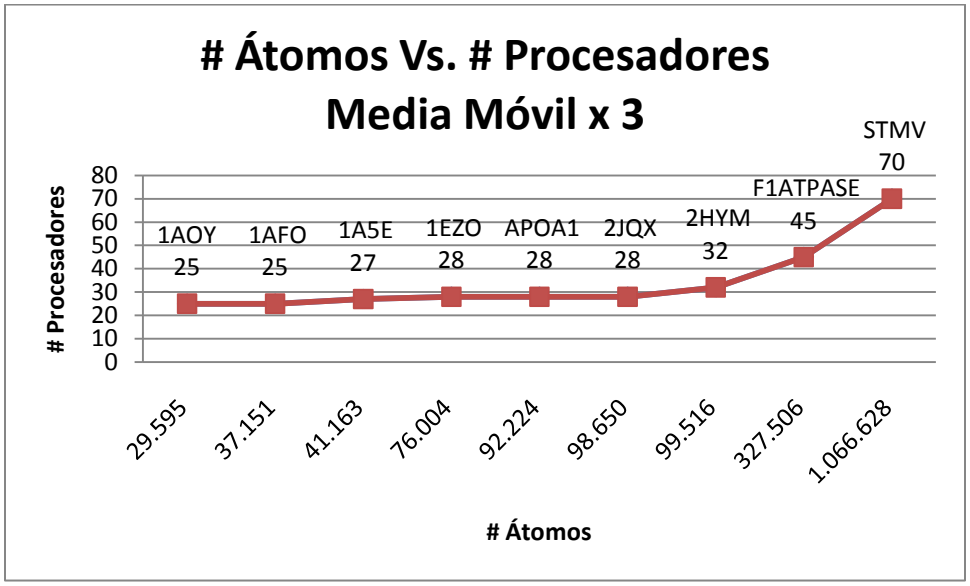
	T. Máximo (s)	T. Máximo (H:M:S)	# Procesadores	Configuración	T. Mínimo (s)	T. Mínimo (H:M:S)	# Procesadores	Configuración	T. Promedio (s)	T. Promedio (H:M:S)
1AOY	4.914,45	1:21:54,45	3	Sala A	427,07	0:07:07,07	25	Supercluster	1.066,91	0:17:46,91
1AFO	6.302,84	1:45:02,84	3	Sala A	547,95	0:09:07,95	25	Supercluster	1.342,21	0:22:22,21
1A5E	6.907,50	1:55:07,5	3	Sala A	585,51	0:09:45,51	27	Supercluster	1.537,30	0:25:37,3
1EZO	53.653,18	14:54:13,18	3	Sala B	1.011,08	0:16:51,08	28	Supercluster	2.399,11	0:39:59,11
APOA1	2.042,69	0:34:02,69	3	Sala BD	175,26	0:02:55,26	28	Supercluster	376,75	0:06:16,75
2JQX	17.230,58	4:47:10,58	3	Sala BD	1.305,62	0:21:45,62	28	Supercluster	2.863,01	0:47:43,01
2HYM	17.470,40	4:51:10,4	3	Sala A	1.224,07	0:20:24,07	32	Supercluster	2.812,41	0:46:52,41
F1ATPASE	29.655,29	8:14:15,29	3	Sala B	434,85	0:07:14,85	45	Sala BD	1.127,42	0:18:47,42
STMV	179.686,89	49:54:46,89	8	Sala B	1.086,33	0:18:06,33	70	Supercluster	5.254,29	1:27:34,29

Tabla 5. Resumen de los tiempos obtenidos luego de la ejecución de las minimizaciones.

En la Tabla 5 se muestra el tiempo máximo de ejecución en segundos y en formato H:M:S, seguido por el número de procesadores y en cual ambiente se alcanzó dicho tiempo. Posteriormente se observa el tiempo mínimo de ejecución en segundos y en formato H:M:S, seguido por el número de procesadores y con cual configuración de salas se logró dicho tiempo. Finalmente, se encuentra el tiempo promedio de ejecución de la minimización en segundos y en formato H:M:S.

De acuerdo a lo estimado con referencia a la configuración de las salas, los tiempos máximos resultaron de las ejecuciones en las salas A y B, sin embargo APOA1 y 2JQX tardaron más tiempo en ejecutarse en la sala BD que en las salas más lentas. De igual manera, se esperaba que los tiempos mínimos se obtuvieran en la configuración Supercluster; sin embargo, en la mayoría de proteínas no se necesitaron más de 28 procesadores para obtener el tiempo mínimo. Para las dos proteínas con el mayor número de átomos, se necesitaron 45 (F1ATPASE) y 70 (STMV) procesadores respectivamente.

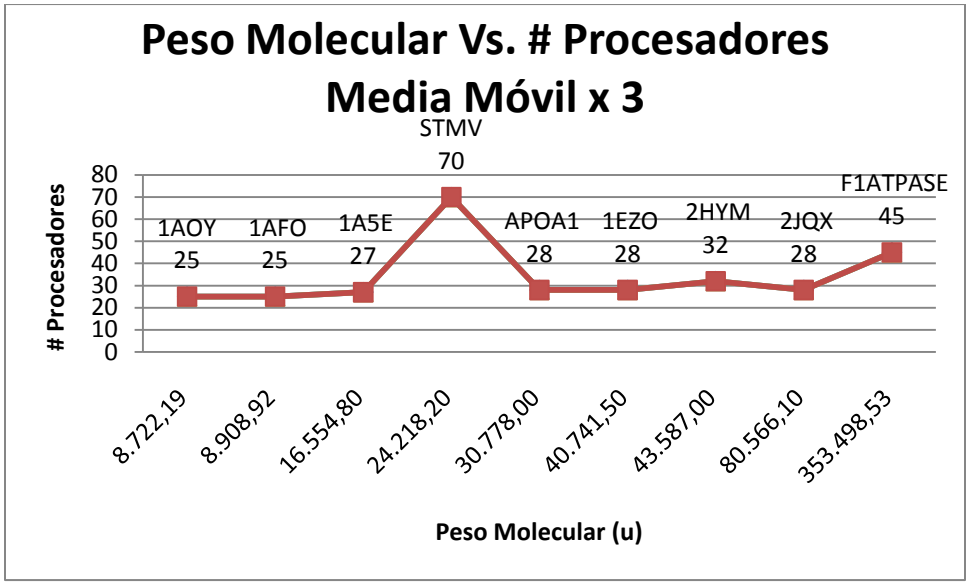
Un punto interesante es que a pesar de haberse necesitado 45 procesadores para obtener el tiempo mínimo en la proteína F1ATPASE, la configuración que dio el mejor rendimiento fue Sala BD y no Supercluster. A continuación se muestra la gráfica de cantidad de átomos Vs. Cantidad de procesadores necesarios para obtener el mínimo tiempo de ejecución, de acuerdo los datos de la Tabla 5.



Gráfica 1. Cantidad de átomos en la simulación Vs. Cantidad de procesadores necesarios para alcanzar el mínimo tiempo de ejecución. Datos tomando en cuenta la media móvil con periodo 3

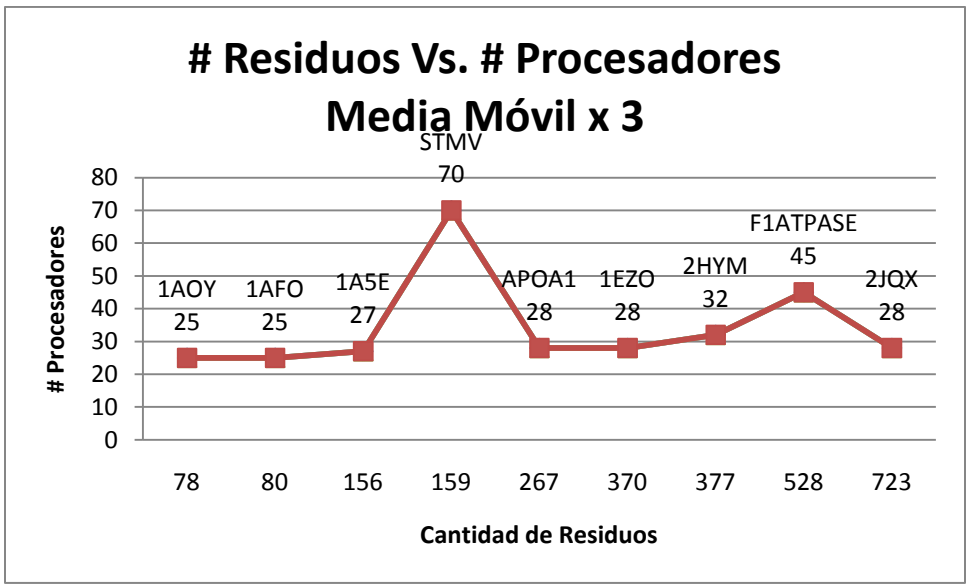
La Gráfica 1 muestra que a mayor cantidad de átomos en la simulación, más procesadores se necesitan para obtener el mínimo tiempo de ejecución. Sin embargo, no es un resultado contundente pues es importante recordar que se trata de la unión de varios sistemas de cómputo heterogéneos, y que dicha unión se hizo de tal manera que primero se aprovecharan los procesadores más rápidos y luego los más lentos. Esta puede ser una de las razones por las cuales luego de 100.000 átomos, la curva crece con un comportamiento posiblemente polinómico o exponencial. Si se tuvieran más procesadores rápidos, posiblemente el número de procesadores descendería al igual que el tiempo mínimo de ejecución.

Lo anterior no es correspondiente con el peso molecular; la siguiente gráfica relaciona el peso molecular con la cantidad de procesadores:



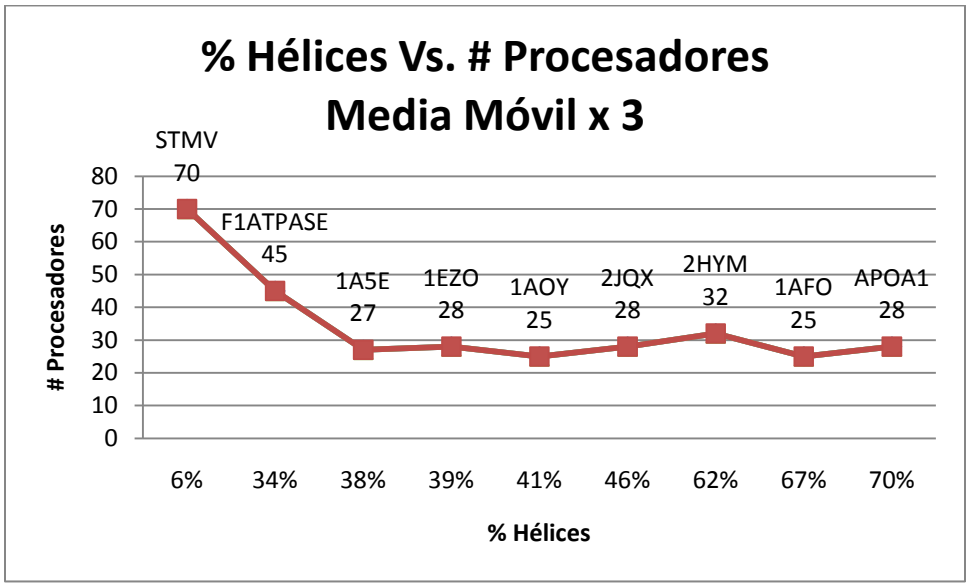
Gráfica 2. Peso Molecular Vs. Cantidad de procesadores necesarios para alcanzar el mínimo tiempo de ejecución. Datos tomando en cuenta la media móvil con periodo 3

La Gráfica 2 muestra que es muy baja la tendencia a incrementar el número de procesadores mientras se incrementa el peso molecular de la proteína a minimizar. De hecho, la proteína con mayor peso molecular no es la que necesita la mayor cantidad de procesadores; dicho requerimiento es de la cuarta proteína en orden de peso molecular, la cual necesitó de 45 procesadores.



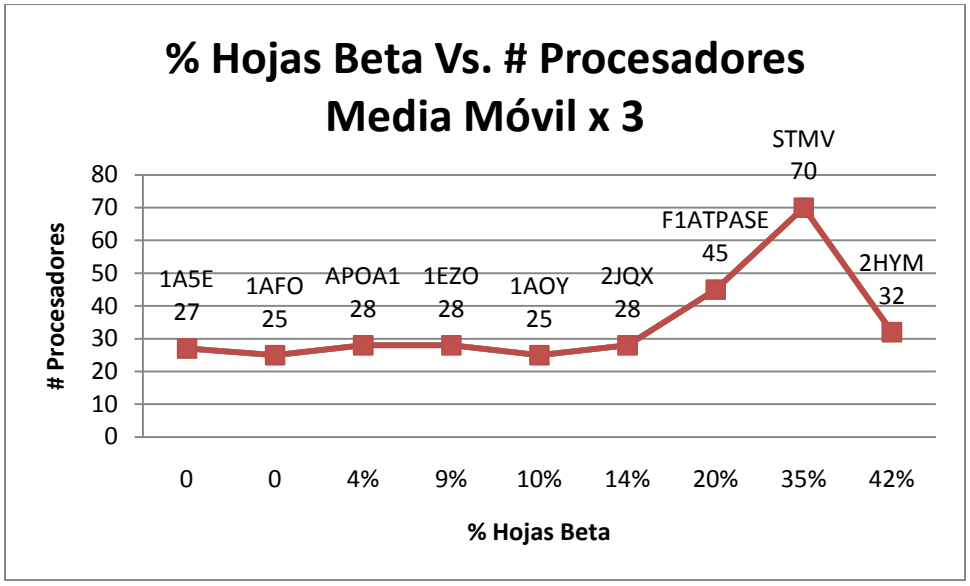
Gráfica 3. Cantidad de residuos Vs. Cantidad de procesadores necesarios para alcanzar el mínimo tiempo de ejecución. Datos tomando en cuenta la media móvil con periodo 3

Igual sucede con la Gráfica 3: no hay suficiente evidencia para afirmar que entre más residuos tenga la proteína, más procesadores se necesitarán para alcanzar el mínimo tiempo de ejecución. Nuevamente la proteína con más átomos en su minimización es la que requiere mayor cantidad de procesadores.



Gráfica 4. Porcentaje de Hélices Alfa Vs. Cantidad de procesadores necesarios para alcanzar el mínimo tiempo de ejecución. Datos tomando en cuenta la media móvil con periodo 3

Sin embargo, la Gráfica 4 muestra una tendencia que indica que entre más composición de hélices alfa tenga la proteína, menos procesadores se necesitan para alcanzar el mínimo tiempo de ejecución. Se observa que la proteína que menos composición de hélices alfa tiene (STMV, con 6% de hélices) requiere de 70 procesadores para su minimización, seguida de la F1ATPASE con 34% de composición, la cual requiere de 45 procesadores. Como se sabe, las hélices alfa son estructuras con poca flexibilidad ya que están altamente estabilizadas por los puentes de hidrógeno. Al tener alta composición de hélices en la proteína, se restringe su movimiento y por ende, los cálculos intensivos se centran en los elementos menos estructurados.



Gráfica 5. Porcentaje de hojas plegadas Beta Vs. Cantidad de procesadores necesarios para alcanzar el mínimo tiempo de ejecución. Datos tomando en cuenta la media móvil con periodo 3

Por su parte, la Gráfica 5 muestra que entre menos composición de hojas plegadas Beta tenga la proteína, menos procesadores se requieren en su minimización. Sin embargo, también se observa que la proteína con mayor composición de hojas plegadas en Beta (2HYM, con el 42%) requiere solo de 32 procesadores, mientras que la STMV, con 35% de hojas plegadas en Beta, requiere de 70 procesadores. A pesar de ser regiones con alta estructuración, las hojas plegadas en Beta no son tan estables como las hélices alfa.

Respecto a la estabilización

En este trabajo se definió el punto de estabilización de la simulación como aquel en el cual el hecho de añadir un procesador mas al sistema no produce una mejora superior al 10% con una tolerancia del 1%, es decir, 10% ± 1%. Se tomaron los datos obtenidos de la ejecución aplicando una media móvil con periodo 3.

	T. Mínimo (H:M:S)	No. Procesadores T. Mínimo	% Máxima Mejora Respecto a un procesador	T. Ejecución en punto estabilización (H:M:S)	% Estabilización del rendimiento	No. Procesadores Estabilización	Diferencia entre estabilización y mínimo (H:M:S)
1A5E	0:09:45,51	27	85,64%	0:20:28,87	69,85%	5	0:10:43,37
1AFO	0:09:07,95	25	85,07%	0:18:49,88	69,21%	5	0:09:41,93
1AOY	0:07:07,07	25	85,14%	0:14:38,82	69,43%	5	0:07:31,74

	T. Mínimo (H:M:S)	No. Procesadores T. Mínimo	% Máxima Mejora Respecto a un procesador	T. Ejecución en punto estabilización (H:M:S)	% Estabilización del rendimiento	No. Procesadores Estabilización	Diferencia entre estabilización y mínimo (H:M:S)
1EZO	0:16:51,08	28	87,06%	0:50:41,03	61,07%	4	0:33:49,94
2HYM	0:20:24,07	32	88,25%	0:52:58,98	69,48%	5	0:32:34,91
2JQX	0:21:45,62	28	87,41%	0:50:47,09	70,62%	5	0:29:01,47
APOA1	0:02:55,26	28	85,65%	0:07:46,45	61,80%	4	0:04:51,19
F1ATPASE	0:07:20,25	45	88,46%	0:25:36,97	59,72%	4	0:18:16,71
STMV	0:18:06,33	70	81,12%	1:21:10,94	15,35%	5	1:03:04,62

Tabla 6. Comparación entre el tiempo mínimo obtenido y el tiempo de estabilización de la simulación.

En la Tabla 6 se muestra nuevamente el tiempo mínimo obtenido luego de la ejecución, en formato H:M:S, la cantidad de procesadores que se necesitaron para alcanzar dicho mínimo, el porcentaje de mejora en el tiempo que hubo respecto a la ejecución utilizando un solo procesador, el tiempo en que se alcanzó la estabilidad en la simulación (en formato H:M:S), la cantidad de procesadores necesarios para alcanzar dicho tiempo de estabilidad, y la diferencia entre el mínimo tiempo de ejecución y el tiempo de estabilidad.

Se observa que en la mayoría de ejecuciones se alcanza la estabilización con cinco procesadores. Sin embargo, los tiempos si presentan variación entre los 5 y los 63 minutos aproximadamente. No se observan tendencias relacionadas con los datos respecto a la estabilización de la simulación.

Lo anterior muestra que para una simulación con características similares a las presentadas en este trabajo, se requiere de mínimo cinco procesadores para tener un tiempo de ejecución aceptable, sabiendo que añadir un procesador más no va a dar una mejora superior al 10%. Pero también se debe tener en cuenta que las pequeñas mejoras obtenidas después de este número de procesadores pueden hacer que al final el tiempo de ejecución sí disminuya considerablemente.

También es posible observar que con el máximo número de procesadores necesarios para alcanzar el tiempo mínimo, se logra una mejora de entre el 81% y el 88%, mientras que usando los procesadores necesarios para la estabilización se logra una mejora de entre el 15% y el 70%.

Speed Up y Eficiencia

El rendimiento de un programa está relacionado con su tiempo total de ejecución. Si T_1 es el tiempo de ejecución para resolver algún problema usando un programa secuencial en un solo procesador, y T_p es el tiempo de ejecución para resolver el mismo problema usando un programa paralelo ejecutado en p procesadores, entonces el Speed Up (SU) se define como: $SU = \frac{T_1}{T_p}$.

Si al añadir p procesadores el programa se ejecuta p veces más rápido, se dice que el Speed Up es lineal o perfecto, pero el caso general es que el Speed Up del programa ejecutado en p procesadores sea menor que p.

Por otra parte, la eficiencia es una medida de qué tan bien un programa paralelo utiliza procesadores extra. La eficiencia Ef se define como $Ef = \frac{SU}{p} = \frac{T_1}{(p \cdot T_p)}$. Si un programa tiene Speed Up lineal, su eficiencia es 1 (o 100%). Tanto el Speed Up como la eficiencia son medidas relativas que dependen del número de procesadores, el tamaño del problema y el algoritmo usado[91]

	T. Mínimo (H:M:S)	No. Procesadores T. Mínimo	SpeedUp T.Mínimo	% Eficiencia T.Mínimo	No. Procesadores Estabilización	SpeedUp en estabilización	Eficiencia en estabilización
1A5E	0:09:45,51	27	6,96	26,81%	5	3,39	85,96%
1AFO	0:09:07,95	25	6,71	27,96%	5	3,34	84,24%
1AOY	0:07:07,07	25	6,73	28,07%	5	3,36	84,87%
1EZO	0:16:51,08	28	7,75	28,78%	4	2,74	92,03%
2HYM	0:20:24,07	32	8,52	27,47%	5	3,34	84,70%
2JQX	0:21:45,62	28	7,95	29,47%	5	3,51	88,38%
APOA1	0:02:55,26	28	6,97	25,83%	4	2,81	94,16%
F1ATPASE	0:07:20,25	45	8,67	19,71%	4	2,65	89,10%
STMV	0:18:06,33	70	5,30	7,91%	5	1,21	69,06%

Tabla 7. Eficiencia y Speed Up en tiempo mínimo y en tiempo de estabilización

En la Tabla 7 se muestra el tiempo mínimo de simulación alcanzado junto con el número de procesadores que se utilizaron. Seguidamente se encuentra el Speed Up logrado y su eficiencia. Luego se encuentra la cantidad de procesadores necesarios para alcanzar la estabilización de la simulación, el Speed Up alcanzado y su correspondiente eficiencia.

Se observa que en el tiempo mínimo el Speed Up es muy bajo y alejado del Speed Up lineal, y la eficiencia no supera el 29%, mientras que en los tiempos de estabilización, el Speed Up es más cercano y la eficiencia del sistema es más alta. Lo anterior concuerda con lo encontrado en la literatura, donde se afirma que frecuentemente la eficiencia de un programa paralelo disminuye a medida que el número de procesadores aumenta[91].

Conclusiones y Trabajo Futuro

Con este trabajo se muestra la posibilidad de hacer cómputo de alto desempeño utilizando los recursos fácilmente accesibles en una universidad, con especial énfasis pero no restringido a la dinámica molecular.

De esta forma, se aprovecha la infraestructura computacional instalada en la universidad, sin sobrecostos para los investigadores y sin tener que buscar dinero para la adquisición de equipos, con los tiempos que estos trámites implican, junto con los tiempos de instalación y configuración.

Adicional a lo anterior, el adquirir un supercomputador para un solo proyecto interno en un departamento es, financieramente hablando, una alternativa inviable. Una posibilidad consistiría en aplicar a una convocatoria de financiación en una entidad como Colciencias o internacional como el National Science Foundation. Sin embargo, los tiempos y los trámites para aplicar son bastante engorrosos y la investigación debe tener un impacto realmente alto para poder siquiera ser considerada.

En la Universidad existen muchos grupos de investigación con necesidades de cómputo de alto desempeño como las Facultades de Ingeniería, Ciencias, Estudios Ambientales y Rurales, Derecho, Arquitectura y Diseño, Medicina, solo por mencionar algunas. En este sentido, es mucho más provechoso para todas las Facultades, el unir esfuerzos para comprar servidores modestos, que unidos a los ya existentes y con el apoyo técnico de la DTI, provean un "supercomputador virtual" a todo aquel que lo requiera, sin importar la magnitud de sus investigaciones. De hecho, como lo muestra este trabajo, esta alternativa ya existe al poner en funcionamiento común un conjunto de computadores como lo pueden ser los de una sala de estudiantes en periodos ociosos o no utilizados sin necesidad de invertir mucho: solo basta con tener personal en capacidad de dar mantenimiento o soporte a los usuarios, y dicha dependencia ya existe. Se debe investigar en alternativas existentes o el desarrollo de herramientas nuevas que permitan la tolerancia a fallos y la replicación en ambientes poco controlados como las salas de cómputo y en general los sistemas oportunistas (o no dedicados).

Adicionalmente, con el grupo de investigación SIDRe se ha iniciado la tarea de crear conciencia en la comunidad javeriana para unir esfuerzos y crear la denominada Grilla Javeriana, es decir, un sistema de computo de alto desempeño disponible para todos. Este sistema a futuro se uniría a la iniciativa de Grid Colombia, que pretende crear un supercomputador a nivel nacional, y a mayor plazo, a nivel mundial.

Es importante como trabajo adicional, calcular cuántos FLOPs es capaz de desarrollar un sistema como este y eventualmente, publicar los resultados en TOP 500. Adicionalmente, ejecutar trabajos más grandes cuyos tiempos estén reportados y que no pertenezcan necesariamente a las ciencias

biológicas. En el campo de la simulación de sistemas complejos existen muchos ejemplos, como simulaciones de supernovas, nanomáquinas, dinámica de muchedumbre, etc.

Se hicieron pruebas sobre un cluster Clustermatic, el cual optimiza las comunicaciones al modificar el protocolo de red. Sin embargo no era viable su implementación en las salas de cómputo disponibles pues requería una modificación importante en su configuración. Se sugiere hacer el análisis sobre este cluster en un ambiente controlado más grande y trabajar en convertirlo en una alternativa viable para las condiciones restringidas de una institución académica.

Los problemas de dinámica molecular por su naturaleza son altamente acoplados, lo cual conlleva a que su paralelización no sea trivial. Esto es un impedimento para que un sistema de computación en grilla pueda aprovecharse plenamente, aunque para los sistemas de cómputo en cluster son problemas ideales. Lo anterior no quiere decir que los sistemas de computación en grilla no se puedan utilizar en dinámica molecular, pues la grilla se encarga automáticamente de configurar un sistema en cluster para luego enviar allí el trabajo.

Por ejemplo, en el cluster, dependiendo de la configuración, todas las maquinas deben tener replicados previamente todos los archivos y programas involucrados en la simulación, razón por la cual se necesitó desarrollar un programa que hiciera esta tarea automáticamente en todas las maquinas. La grilla computacional construiría el cluster de acuerdo a los recursos que su gestor vea disponibles, y en el momento de ejecución se encargaría de distribuir los archivos entre los nodos del mismo, ahorrando trabajo y tiempo al investigador.

En el caso específico de NAMD, su ejecución distribuida requiere de la construcción inicial de un archivo con las direcciones IP de las máquinas involucradas en la simulación. Un gestor de grilla, al momento de construir el cluster también construiría dicho archivo automáticamente.

El mostrar que el experimento no es aleatorio, sino que las variables están correlacionadas y la varianza entre cada experimento es mínima, permitió ejecutar una sola vez las simulaciones en los demás sistemas; de otra manera, el haberlos repetido varias veces hubiera añadido ruido a los resultados, pues los ambientes son menos controlados (lograr que tengan condiciones constantes depende de la administración de las salas y de la Dirección de Tecnologías de Información de la Universidad) y aspectos como el tráfico en la red están prácticamente fuera de la administración del Departamento. En este aspecto, el promedio de las varianzas obtenido está alrededor del 2.75%.

Se pudo observar que a medida que se incrementa el poder de cómputo mediante la adición de nodos o procesadores, se disminuye el tiempo para realizar un trabajo hasta un determinado punto, que depende de las características del problema. En ocasiones el hecho de añadir más poder computacional hace que se empeore el tiempo total de simulación. En problemas de tamaño pequeño puede deberse a que es menor el tiempo de procesamiento que el tiempo de coordinación y comunicación entre procesadores. En problemas cuyo tamaño es grande, puede deberse a saturación de los canales de comunicación.

En este sentido, se presenta una ecuación que permite estimar la cantidad de procesadores necesaria para llevar a cabo un experimento como el descrito en este trabajo. A su vez, se presenta una serie de ecuaciones que permiten estimar el tiempo total de simulación de acuerdo a las características del ambiente computacional con que se cuente, dando como parámetro la cantidad de procesadores.

También se observa que a medida que se añaden procesadores, el porcentaje de mejora entre cada procesador adicional disminuye hasta un punto que en este trabajo se denominó "zona de estabilidad" en donde un procesador adicional no aporta más del 10%±1% de ganancia en el tiempo. Sin embargo, en ciertos problemas, la suma de estas ganancias aunque sean muy pequeñas, al final si aportan mucho en el tiempo total de solución del problema.

Se observó que a mayor cantidad de átomos involucrados en la simulación, mayor es el tiempo requerido para su finalización. Adicionalmente, el peso molecular no está directamente relacionado con el tiempo, al igual que la cantidad de residuos. Sin embargo la cantidad de residuos y la cantidad de átomos si están relacionados, por ende como trabajo futuro es interesante diseñar un experimento que permita observar el tipo de residuos (por ejemplo si son hidrofóbicos o hidrofílicos), la cantidad de átomos, etc, respecto al tiempo que tarda la simulación en finalizar.

Con referencia a la composición de la proteína, se observó que a mayor cantidad de hélices alfa, menor es el tiempo requerido para la simulación. Esto puede deberse a que las hélices alfa son altamente estructuradas y rígidas gracias a los puentes de hidrógeno que en ellas se forman. Curiosamente, entre menor sea la composición de hojas beta, menos procesadores se requieren para llegar al tiempo mínimo de simulación. Puede deberse a que a pesar de que las hojas beta también son regiones de alta estructuración, dependiendo de si son hojas paralelas o antiparalelas, los enlaces de hidrógeno no son tan estables como en las hélices alfa. En este sentido es interesante hacer más ejecuciones con proteínas que contengan diferentes composiciones y diferentes tipos de zonas de alta estructuración, todo en un ambiente con elementos de procesamiento homogéneos para verificar este hecho, reduciendo la cantidad de variables intervinientes.

A pesar de tratarse de simulaciones relativamente grandes, fueron necesarios menos procesadores de los disponibles. Probablemente con sistemas más complejos se aproveche mejor la totalidad del sistema. En la mayoría de los casos solo se necesitaron entre 25 y 45 procesadores, y solo en un caso se utilizaron 70 de ellos. Si no se hubiera utilizado el Rack de servidores, sino únicamente las máquinas disponibles en las salas de cómputo, seguramente la cantidad de máquinas requeridas aumentaría (no se deben dejar a un lado las consideraciones de la red y dispositivos de comunicación, pues también influyen enormemente en el procesamiento). Esta es la situación general que se pretende aprovechar en la Universidad, pues su mayor composición en computadores la aportan los equipos de salas disponibles para estudiantes y para personal administrativo. Además, en una universidad con pocos recursos económicos resulta más fácil unir

los computadores de escritorio disponibles, en vez de tener que adquirir un rack de servidores con los costos, adecuación, mantenimiento y capacitación que esto implica.

Se encontró que la curva de mejora en el tiempo se estabilizaba cuando se utilizaba un promedio de cinco procesadores, brindando un porcentaje de mejora de entre el 15 y el 70 por ciento respecto al tiempo de ejecución en un solo procesador. Por otra parte, el utilizar más procesadores hasta llegar al tiempo mínimo encontrado mostró diferencias en el tiempo total de hasta 63 minutos, y mejoras que oscilaban entre el 81 y el 88 por ciento respecto a un procesador.

El Speed Up y por consiguiente la eficiencia es muy baja en general. Para 24 procesadores se tiene un Speed Up de alrededor de 7 con eficiencia menor al 29 por ciento. Entre menos procesadores se utilicen, el Speed Up es más cercano al lineal, y es mayor la eficiencia. De todas maneras es importante realizar otro tipo de medidas, como el ejecutar dos o más simulaciones al tiempo, pues la teoría dice que el sistema tiene una eficiencia del 100% cuando usa un solo procesador pues se asume que todo el poder de cómputo está enfocado en resolver el problema, entonces menor porcentaje de eficiencia indica menor uso del mismo. Al ejecutar más simulaciones al tiempo, el resto de sistema que está subutilizándose por una simulación, estaría siendo aprovechado por las demás simulaciones.

Dado que todos los sistemas se encontraban en Linux, un siguiente trabajo a desarrollar es establecer los mecanismos para que los usuarios de Windows aporten poder de cómputo a este tipo de simulaciones. Una alternativa es el uso de máquinas virtuales ejecutando Linux sobre windows[90]. Esto permitiría el aprovechamiento de las máquinas no solo en las noches o en los tiempos ociosos, sino también en los periodos de subutilización, es decir, cuando no se usa el 100% del procesador, pues se puede estar ejecutando un trabajo en segundo plano haciendo que el procesador se aproveche mucho más. Dado lo anterior, se pretende además que a futuro en la Pontificia Universidad Javeriana se cuente con mayor poder computacional mediante el uso del tiempo ocioso de los computadores del campus, asegurando total transparencia tanto para los usuarios de los computadores que están aportando procesamiento, como para los usuarios del sistema de cómputo de alto desempeño que están sometiendo trabajos para ser procesados.

Los resultados de este trabajo demuestran que utilizar los recursos computacionales ya existentes en la universidad permitirán el procesamiento de grandes problemas que en este momento no se han podido procesar por falta de suficiente poder de cómputo. Es importante destacar que si bien este trabajo se enfoca en las ciencias biológicas, la capacidad de cómputo explorada puede ser utilizada en cualquier clase de problema que lo requiera, sin importar si es académico, de investigación o de soporte a alguno de los sistemas de información existentes en la Universidad.

A nivel institucional se iniciará un trabajo conjunto entre diferentes facultades, departamentos y la Dirección de Tecnologías de Información para emprender el proyecto “Grilla Javeriana” que busca impulsar el cambio de políticas y la unión de esfuerzos para conseguir un sistema de cómputo de alto desempeño para la comunidad javeriana en general.

Con la experiencia adquirida en el desarrollo de este trabajo, con el Departamento de Ingeniería de Sistemas se inició la construcción de la asignatura Computación de Alto Desempeño para pregrado y postgrado, la cual busca que los estudiantes se familiaricen con este importante tema.

No se debe dejar a un lado la búsqueda de mecanismos que aseguren tanto la información de los trabajos sometidos al sistema, como la seguridad de la información de los computadores donde los trabajos están siendo procesados.

Desarrollo de software especializado para problemas específicos

Script para descarga de proteínas

downloadPDBs es un programa sencillo hecho en C++ estándar que recibe en su llamado un archivo con identificadores de proteínas disponibles en el Protein Data Bank. Luego de utilizar el programa se genera un nuevo archivo que al ser ejecutado empieza la descarga de aquellos identificadores especificados inicialmente.

Scripts para preparación de proteínas

Para eliminar el agua de las proteínas, se utilizó el siguiente script llamado generatePSF.png:

```
mol load pdb *.pdb
set MyProtein [atomselect top protein]
$MyProtein writepdb *-wt.pdb
mol delete top
```

y luego se utilizó el script llamado SolvateWaterBox.tcl para hidratar a la proteína en una caja de agua:

```
### Script to immerse molecule in a sphere of water just large
enough
### to cover it

set molname *-wt_autopsf

mol new ${molname}.psf
mol addfile ${ molname }.pdb

### Determine the center of mass of the molecule and store the
coordinates
set cen [measure center [atomselect top all] weight mass]
set x1 [lindex $cen 0]
set y1 [lindex $cen 1]
set z1 [lindex $cen 2]
set max 0

### Determine the distance of the farthest atom from the center of
mass
foreach atom [[atomselect top all] get index] {
  set pos [lindex [[atomselect top "index $atom"] get {x y z}] 0]
  set x2 [lindex $pos 0]
  set y2 [lindex $pos 1]
  set z2 [lindex $pos 2]
```

```
  set dist [expr pow(($x2-$x1)*($x2-$x1) + ($y2-$y1)*($y2-$y1) +
($z2-$z1)*($z2-$z1),0.5)]
  if {$dist > $max} {set max $dist}
  }

mol delete top

### Solvate the molecule in a water box with enough padding (15
A).
### One could alternatively align the molecule such that the
vector
### from the center of mass to the farthest atom is aligned with
an axis,
### and then use no padding
package require solvate
solvate ${molname}.psf ${molname}.pdb -t 15 -o ${molname}_wb

resetpsf
package require psfgen
mol new ${molname}_wb.psf
mol addfile ${molname}_wb.pdb
readpsf ${molname}_wb.psf
coordpdb ${molname}_wb.pdb
```

Es importante aclarar que estos scripts fueron adaptados de aquellos disponibles en los tutoriales de NAMD.

Generador de estructura de directorios para simulaciones

Se trata de un programa hecho en C++ estándar llamado CreateProteinStructure. Al ejecutarlo, se le deben enviar tres parámetros: el nombre del archivo que contiene los parámetros de la simulación a configurar, la cantidad de procesadores a utilizar y el nombre del archivo con la lista de nodos disponibles para la simulación.

El archivo de parámetros tiene la siguiente estructura:

- Cadena de caracteres con la ruta del ejecutable de charmrun.
- Cadena de caracteres con la ruta del ejecutable de namd2.
- Cadena de caracteres con la ruta de la carpeta raíz donde se generará la estructura de directorios.
- Número entero que indica la cantidad **C** de archivos de topología a utilizar.
- **C** cadenas de caracteres, una por línea, cada una con la ruta de cada archivo de topología.
- Número entero que indica la cantidad **T** de tamaños de proteínas a trabajar. Para cada uno de los **T** tamaños:

- Cadena de caracteres con el nombre del tamaño de proteína.
- Número entero que indica en número **N** de proteínas para el primer tamaño. Para cada una de las **N** proteínas:
- Cadena de caracteres con el nombre de la primera proteína para este tamaño.
- Dos cadenas de caracteres, una por línea, con las rutas del archivo de coordenadas (.pdb) y el archivo de estructura (.psf) en ese orden.
- Número entero que indica la cantidad **M** de minimizaciones que se van a realizar.
- Número entero que indica la cantidad **A** de ambientes o arquitecturas sobre las cuales se ejecutarán las minimizaciones. Para cada uno de los **A** ambientes:
- Cadena de caracteres con el nombre del ambiente.
- Número entero que indica la cantidad **P** de procesadores que conforman el ambiente.
- Las siguientes líneas contienen el texto que conforma el archivo de configuración de la simulación.

Por su parte, el archivo con la lista de nodos disponibles para la simulación tiene tantas líneas como nodos existan. Cada línea tiene la palabra "host" seguida de la dirección IP del nodo.

Al final de la ejecución, el programa genera un archivo de ejecución por lotes llamado run.sh, con los comandos para iniciar la minimización.

Por ejemplo, si se fuera a trabajar únicamente en el sistema con todos los procesadores disponibles, se debería generar un archivo de parámetros con la anterior estructura, que involucra una sola ejecución en un solo ambiente de simulación llamado, por ejemplo, SUPERCLUSTER, conformado por 101 procesadores, tres tamaños de proteínas (2k-20k con tres proteínas, 40k-60k con dos proteínas y 60k-1000k con una proteína).

El programa crea entonces una carpeta llamada SUPERCLUSTER y en su interior otra carpeta llamada EXECUTION_1. Dentro de esta última se generan 101 carpetas, cuyo nombre va desde 1_processor hasta 101_processor. Dentro de cada una de ellas a su vez, se crean tres carpetas, cada una con el nombre de los tamaños de las proteínas; dentro de cada carpeta correspondiente a cada tamaño de proteína, se crea una carpeta con el identificador PDB de cada una de ellas. Finalmente, dentro de la carpeta de cada proteína se encuentra el archivo de configuración de la minimización. Como los archivos con las coordenadas de los átomos y el archivo con la estructura no cambian en las ejecuciones, se crea una carpeta adicional llamada BaseFiles con los archivos de estructura, de coordenadas y de parámetros.

Para el caso del experimento, este archivo tiene 606 líneas, pues para cada proteína (seis en total) se ejecuta la simulación en uno, dos, tres hasta 101 procesadores.

Consolidador de wallclock

Teniendo todos los archivos de registro generados por la simulación en una sola carpeta, el programa getResults se encarga de abrir cada uno de ellos y de crear una tabla donde en las filas

se encuentran los datos de cada proteína y en las columnas, la cantidad de procesadores. Un ejemplo del archivo generado, a continuación:

	1 Processor	2 Processors	3 Processors	4 Processors
1A5E	10437,9922	6087,06093	4197,45448	3111,59392
1AFO	9426,83705	5544,69136	3936,9788	3078,27478
1AOY	7299,92125	4312,63535	3130,80794	2291,7112
1EZO	19742,1519	10577,9524	7934,84917	6103,81709
2HYM	26635,1272	15389,0067	10387,0691	8454,53362
2JQX	26517,7951	14167,3465	10050,3337	8081,73504

Tabla 8. Ejemplo de archivo generado por el consolidador de resultados

Dado que los datos quedan separados por tabuladores, el archivo es fácilmente interpretable por cualquier hoja de cálculo. Este programa se encuentra hecho en C++ estándar.

Consolidador de graficas en Excel

Una vez obtenidos los resultados consolidados, es necesario realizar cálculos sobre ellos, como calcular máximos, mínimos, desviaciónes, Speed Up, Eficiencia, porcentaje de mejora, etc. Luego se deben calcular las medias móviles y finalmente realizar las gráficas respectivas con el fin de observar mejor el comportamiento de las simulaciones. Se trata de una tarea tediosa dada la gran cantidad de datos y tablas resultantes. Para este trabajo por ejemplo, se tienen nueve proteínas, cada una ejecutada en seis sistemas de cómputo diferentes. A cada proteína se le calculó la eficiencia, el Speed Up, el porcentaje de mejora al añadir un procesador, respecto al procesador anterior y respecto al primer procesador. Para cada uno de los cálculos anteriores se generan las medias móviles con periodo dos, tres y cinco para observar mejor el comportamiento de los datos luego de suavizarlos. Finalmente a cada tabla generada anteriormente, se le genera su respectiva grafica, para un total de 648 gráficas, sin contar las gráficas de resumen y consolidados, a las cuales se les aplica el mismo proceso.

Por esta razón se tomó la decisión de programar una macro en Visual Basic for Applications desde Excel, que tomara los datos consolidados y ejecutara automáticamente el proceso de cálculo y generación de gráficas descrito anteriormente.

Bibliografía

1. Deyu, L., et al., *The role of molecular modeling in bionanotechnology.* Physical Biology, 2006(1): p. S40.
2. Morrison, R., *Cluster Computing*, in *Architectures, Operating Systems, Parallel Processing & Programming Languages*. 2003.
3. Liu, H., et al., *Quantum mechanics simulation of protein dynamics on long timescale.* Proteins: Structure, Function, and Genetics, 2001. **44**(4): p. 484-489.
4. Aksimentiev, A. and K. Schulten, *Imaging {alpha}-Hemolysin with Molecular Dynamics: Ionic Conductance, Osmotic Permeability, and the Electrostatic Potential Map.* Biophysical Journal, 2005. **88**(6): p. 3745-3761.
5. Sun, J., Q. Zhang, and T. Schlick, *Electrostatic mechanism of nucleosomal array folding revealed by computer simulation.* Proceedings of the National Academy of Sciences of the United States of America, 2005. **102**(23): p. 8180-8185.
6. Gregory, R., et al., *Parallelising a model of bacterial interaction and evolution.* Biosystems, 2004. **76**(1-3): p. 121-131.
7. Sterling, T.L., et al., *How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters*. Scientific and Engineering Computation. 1998: The MIT Press.
8. El-Rewini, H. and M. Abd-El-Barr, *Advanced Computer Architecture and Parallel Processing*. 2005: John Wiley & sons, Inc. 288.
9. Andrews, G.R., *Foundations of Multithreaded, Parallel, and Distributed Programming*. 2000: Addison-Wesley. 664.
10. SIDRe, G. [cited; Available from: http://sophia.javeriana.edu.co/sidre, http://scienti.colciencias.gov.co:8081/ciencia.war/search/EnGrupoInvestigacion/xmlInfo.do?nro_id_grupo=0009103P5MW6V5.
11. BCEB. [cited; Available from: http://scienti.colciencias.gov.co:8081/ciencia.war/search/EnGrupoInvestigacion/xmlInfo.do?nro_id_grupo=0009208EILS9PO.
12. Kahn, H.J. and R.B.E. Napper, *The birth of the baby [early digital computer history].* Computer Design, 2000. Proceedings. 2000 International Conference on, 2000: p. 481-484.
13. Miller, E.K., *The computer revolution.* Potentials, IEEE 1989. **8**(2): p. 27-31.
14. Dowd, K. and C. Severance, *High Performance Computing*, ed. M. Loukides and G. Estabrook. 1998: O'Reilly.
15. Miller, E.K., *Development of the digital computer. Part 3.* Potentials, IEEE. **22**(3): p. 42-45.
16. *Grid Computing: Making the Global Infrastructure a Reality* ed. F. Berman, G. Fox, and T. Hey. 2003: Wiley
17. Grama, A., et al., *Inroduction to Parallel Computing*. 2nd ed. 2003: Pearson Ed.
18. Sanbonmatsu, K.Y. and C.S. Tung, *High performance computing in biology: multimillion atom simulations of nanoscale systems.* Journal Of Structural Biology, 2007. **157**(3): p. 470-480.
19. Shave, S.R., et al., *Ligand discovery on massively parallel systems.* IBM Journal of Research & Development, 2008. **52**(1/2): p. 57-67.
20. Shaw, D.M., M. Odelius, and J.S. Tse, *Utility of High Performance Computing Facilities for the Calculation of the Theoretical X-ray Absorption Spectra of Solids.* Performance Computing Systems and Applications, 2007. HPCS 2007. 21st International Symposium on 2007: p. 4.

21. Sahu, J., *Time-Accurate Calculations of Free-Flight Aerodynamics of Maneuvering Projectiles.* DoD High Performance Computing Modernization Program Users Group Conference, 2007: p. 64-69.
22. Davis, D.M., et al. *High-performance computing enables simulations to transform education*. in *Simulation Conference, 2007 Winter*. 2007.
23. Owens, J.D., et al., *A Survey of General-Purpose Computation on Graphics Hardware.* Computer Graphics Forum, 2007. **26**(1): p. 80-113.
24. Scott, R.K., *Assessing the impact of high-performance computing on the drug discovery and development process.* Drug Discovery Today: BIOSILICO, 2004. **2**(5): p. 175-179.
25. Atkinson, J.S., D.H.R. Spenneman, and D. Cornforth, *Redirecting under-utilised computer laboratories into cluster computing facilities.* Campus-Wide Information Systems, 2005. **22**(4): p. 201-209.
26. Pang, Y.P., et al., *EUDOC on the IBM Blue Gene/L system: Accelerating the transfer of drug discoveries from laboratory to patient.* IBM Journal of Research & Development, 2008. **52**(1/2): p. 69-81.
27. *High Performance Cluster Computing*, ed. R. Buyya. Vol. Volume 1: Architectures & Systems; Volume 2: Programming & applications. 1999: Prentice Hall PTR.
28. Karakasidis, T.E., N.S. Cholevas, and A.B. Liakopoulos, *Parallel short range molecular dynamics simulations on computer clusters: Performance evaluation and modeling.* Mathematical and Computer Modelling, 2005. **42**(7-8): p. 783-798.
29. *The Grid: Blueprint for a New Computing Infrastructure* ed. I. Foster and C. Kesselman. 1998: Morgan Kaufmann Publishers.
30. Krishnan, A., *A survey of life sciences applications on the grid.* New Generation Computing, 2004. **22**(2): p. 111-126.
31. Konagaya, A., et al., *The superstructure toward open bioinformatics grid.* New Generation Computing, 2004. **22**(2): p. 167-176.
32. Dongarra, J. and A. Lastovetsky, *An Overview of Heterogeneous High Performance and Grid Computing.* Engineering the Grid: Status and Perspective, 2006.
33. Bartocci, E., et al., *An Agent-Based Multilayer Architecture for Bioinformatics Grids.* NanoBioscience, IEEE Transactions on, 2007. **6**(2): p. 142-148.
34. Konagaya, A. *OBIGrid: towards the 'Ba' for sharing resources, services and knowledge for bioinformatics*. in *Cluster Computing and the Grid Workshops, 2006. Sixth IEEE International Symposium on*. 2006.
35. Nebro, A.J., E. Alba, and F. Luna, *Multi-Objective Optimization Using Grid Computing.* Soft Computing, 2007. **11**(6): p. 531-540.
36. Wipat, A., et al., *Developing Grid-based Systems for Microbial Genome Comparisons: The Microbase Project.* UK e-Science 2008 All Hands Meeting 2008.
37. *Overview of the IBM Blue Gene/P project.* IBM Journal of Research & Development, 2008. **52**(1/2): p. 199-220.
38. Moreira, J., et al., *The Blue Gene/L Supercomputer: A Hardware and Software Story.* International Journal of Parallel Programming, 2007. **35**(3): p. 181-206.
39. Bohm, E., et al., *Fine-grained parallelization of the Car-Parrinello ab initio molecular dynamics method on the IBM Blue Gene/L supercomputer.* IBM Journal of Research & Development, 2008. **52**(1/2): p. 159-175.
40. Meuer, H., et al. *Top 500 Supercomputer Sites*. 1993 November, 2007 [cited 2008 March]; Available from: www.top500.org.

41. *AMSUD – GBRAMS. South-American Grid for the Brazilian Regional Atmospheric Modeling System* [cited 2008 10/10/2008]; Available from: http://www.sticamsud.org/Proyectos/ProyectosAprobados.
42. *AMSUD*. [cited 2008 10/10/2008]; Available from: http://www.sticamsud.org/.
43. *Caso: Fundación Ciencia para la Vida*. [cited 2008 10/10/2008]; Available from: http://www.microsoft.com/conosur/hechos/studies/fcv.aspx.
44. *Centro de Modelamiento Matemático*. [cited 2008 10/10/2008]; Available from: http://www.cmm.uchile.cl/index.php?option=detalle&lang=es&CodigoNoticia=387.
45. *High Performance Computing Lab. Chile*. [cited 2008 10/10/2008]; Available from: http://www.cmm.uchile.cl/?lang=es&option=laboratorie&parent=projectarea.
46. *Universidad de San Francisco de Quito*. [cited 2008 10/10/2008]; Available from: http://latolita.usfq.edu.ec/.
47. *Red Universitaria Metropolitana de Bogotá*. [cited 2008 10/10/2008]; Available from: http://www.rumbo.edu.co/.
48. *Red Nacional Académica de Alta Tecnología*. [cited 2008 10/10/2008]; Available from: http://www.renata.edu.co.
49. *Semillero de Investigación Especializado en Computación de Alto Rendimiento*. [cited 2008 10/10/2008]; Available from: http://siecar.upbbga.edu.co/index.html.
50. *Supercomputador Distribuido de la Universidad Nacional*. [cited; Available from: http://ungrid.unal.edu.co.
51. *Grupo de Investigación SIDRe - Pontificia Universidad Javeriana*. [cited; Available from: http://ingenierias.javeriana.edu.co/portal/page?_pageid=233,831051,233_831173&_dad=portal&_schema=PORTAL&tab=inicio.
52. Chao-Tung, Y., K. Yu-Lun, and L. Chuan-Lin. *Design and implementation of a computational grid for bioinformatics*. in *e-Technology, e-Commerce and e-Service, 2004. EEE '04. 2004 IEEE International Conference on*. 2004.
53. *Just the Facts: A Basic Introduction to the Science Underlying NCBI Resources*. [cited 2008 15/09/2008]; Available from: http://www.ncbi.nlm.nih.gov/About/primer/bioinformatics.html.
54. Shih-Nung, C., et al. *Using distributed computing platform to solve high computing and data processing problems in bioinformatics*. in *Bioinformatics and Bioengineering, 2004. BIBE 2004. Proceedings. Fourth IEEE Symposium on*. 2004.
55. Jie, C., et al., *How will bioinformatics impact signal processing research?* Signal Processing Magazine, IEEE, 2003. **20**(6): p. 106-206.
56. Albayraktaroglu, K., et al. *BioBench: A Benchmark Suite of Bioinformatics Applications*. in *Performance Analysis of Systems and Software, 2005. ISPASS 2005. IEEE International Symposium on*. 2005.
57. Li, Y., et al. *Workload characterization of bioinformatics applications*. in *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005. 13th IEEE International Symposium on*. 2005.
58. Chaplot, S.L., *Parallelization in classical molecular dynamics simulation and applications.* Computational Materials Science. **37**(1-2): p. 146-151.
59. Iftimie, R., P. Minary, and M.E. Tuckerman, *Ab initio molecular dynamics: Concepts, recent developments, and future trends.* Proceedings of the National Academy of Sciences of the United States of America, 2005. **102**(19): p. 6654-6659.
60. Hayashi, R., et al., *A classical molecular dynamics simulation of the carbon cluster formation process on a parallel computer.* Diamond and Related Materials, 2001. **10**(3-7): p. 1224-1227.

61. Wu, J.-S., Y.-L. Hsu, and Y.-M. Lee, *Parallel implementation of molecular dynamics simulation for short-ranged interaction.* Computer Physics Communications. **170**(2): p. 175-185.
62. Merelli, I., G. Morra, and L. Milanesi, *Evaluation of a Grid Based Molecular Dynamics Approach for Polypeptide Simulations.* NanoBioscience, IEEE Transactions on, 2007. **6**(3): p. 229-234.
63. Kumar, S., et al., *Scalable molecular dynamics with NAMD on the IBM Blue Gene/L system.* IBM Journal of Research & Development, 2008. **52**(1/2): p. 177-188.
64. Avellaneda, F., et al. *Implementation of a Molecular Simulator Based on a MultiAgent System*. in *Intelligent Agent Technology, 2006. IAT '06. IEEE/WIC/ACM International Conference on*. 2006.
65. Gonzalez, E., et al. *ASMA: Arquitectura para agentes móviles en un entorno distribuido*. 2005 [cited 2008 November 05-th]; Available from: http://portal2.javeriana.edu.co/psp/eppro/OFI/PSFT_EP/c/UJG_CONSULTAS_INFORMES.UJG_PROY_INVESTIG.GBL?FolderPath=PORTAL_ROOT_OBJECT.UJ_OFIWEB_ACTIVIDADES&IsFolder=false&IgnoreParamTempl=FolderPath%2cIsFolder.
66. Pang, Y., et al., *EUDOC: a computer program for identification of drug interaction sites in macromolecules and drug leads from chemical databases.* Journal Of Computational Chemistry, 2001. **22**(15): p. 1750-1771.
67. Car, R. and M. Parrinello, *Unified Approach for Molecular Dynamics and Density-Functional Theory.* Physic Review Letters, 1985. **55**(22): p. 2471-2474.
68. Phillips, J.C., et al., *Scalable molecular dynamics with NAMD.* Journal Of Computational Chemistry, 2005. **26**(16): p. 1781-1802.
69. Poghosyan, A.H., H.H. Gharabekyan, and A.A. Shahinyan, *MOLECULAR DYNAMICS SIMULATIONS OF DMPC/DPPC MIXED BILAYERS.* International Journal of Modern Physics C: Computational Physics & Physical Computation, 2007. **18**(1): p. 73-89.
70. Lynch, D.L. and P.H. Reggio, *Cannabinoid CB1 receptor recognition of endocannabinoids via the lipid bilayer: molecular dynamics simulations of CB1 transmembrane helix 6 and anandamide in a phospholipid bilayer.* Journal of Computer-Aided Molecular Design, 2006. **20**(7/8): p. 495-509.
71. Benz, R.W., et al., *Experimental validation of molecular dynamics simulations of lipid bilayers: a new approach.* Biophysical Journal, 2005. **88**(2): p. 805-817.
72. Chai, L., et al. *Understanding the Impact of Multi-Core Architecture in Cluster Computing: A Case Study with Intel Dual-Core System*. in *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on*. 2007.
73. Kale, L., et al., *NAMD2: Greater Scalability for Parallel Molecular Dynamics.* Journal of Computational Physics, 1999. **151**(1): p. 283-312.
74. Condor Team, U.o.W.-M., *Condor Version 7.2.0 Manual*. 2008. p. 940.
75. Humphrey, W., A. Dalke, and K. Schulten, *VMD - Visual Molecular Dynamics.* Journal of Molecular Graphics, 1996. **14**: p. 33-38.
76. Berman, H.M., et al., *The Protein Data Bank.* Nucleic Acids Research, 2000. **28**: p. 235-242.
77. Byeon, I.-J.L., et al., *Tumor Suppressor p16INK4A: Determination of Solution Structure and Analyses of Its Interaction with Cyclin-Dependent Kinase 4.* Molecular Cell, 1998. **1**(3): p. 421-431.
78. MacKenzie, K.R., J.H. Prestegard, and D.M. Engelman, *A Transmembrane Helix Dimer: Structure and Implications*. 1997. p. 131-133.

79. Sunnerhagen, M., et al., *Solution structure of the DNA-binding domain and model for the complex of multifunctional hexameric arginine repressor with DNA.* Nature Structural Biology, 1997. **4**(10): p. 819-826.
80. Mueller, G.A., et al., *Global folds of proteins with low densities of NOEs using residual dipolar couplings: application to the 370-residue maltodextrin-binding protein.* Journal of Molecular Biology, 2000. **300**(1): p. 197-212.
81. Quadt-Akabayov, S.R., et al., *Determination of the human type I interferon receptor binding site on human interferon-{alpha}2 by cross saturation and an NMR-based model of the complex*. 2006. p. 2656-2668.
82. Grishaev, A., et al., *Refined solution structure of the 82-kDa enzyme malate synthase G from joint NMR and synchrotron SAXS restraints* Journal of Biomolecular NMR, 2008. **40**(2): p. 95-106.
83. Wang, G., J.T. Sparrow, and R.J. Cushley, *The Helix-Hinge-Helix Structural Motif in Human Apolipoprotein A-I Determined by NMR Spectroscopy.* Biochemistry, 1997. **36**(44): p. 13657-13666.
84. Phillips, J.C., et al., *Predicting the structure of apolipoprotein A-I in reconstituted high-density lipoprotein disks.* Biophys. J., 1997. **73**(5): p. 2337-2346.
85. Larson, S.B., et al., *Refined structure of satellite tobacco mosaic virus at 1.8 Å resolution.* Journal of Molecular Biology, 1998. **277**(1): p. 37-59.
86. Cabezón, E., et al., *The structure of bovine F1-ATPase in complex with its regulatory protein IF1.* Nature Structural Biology, 2003. **10**: p. 744 - 750.
87. Clustermatic. [cited July 2007; Available from: www.clustermatic.org.
88. Steinhauser, O., et al., *Parallel biomolecular simulation: Theory, algorithms and implementation.* Simulation Practice and Theory - Selected papers of the 5th European Simulation Congress, 1997. **5**(7-8): p. 573-603.
89. Ylonen, T., *The Secure Shell (SSH) Protocol Architecture.* Network Working Group, 2006.
90. Tanaka, K., M. Uehara, and H. Mori. *A Case Study of a Linux Grid on Windows Using Virtual Machines*. in *Advanced Information Networking and Applications - Workshops, 2008. AINAW 2008. 22nd International Conference on*. 2008.
91. Andrews, G.R., *Foundations of Multithreaded, parallel and distributed programming*. 2000: Addison-Wesley.