Universidad de las Ciencias Informáticas
Facultad 6

Título: “MMH en un entorno distribuido”

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas.

Autores: Yuneimy Tellez Pérez

Yudislainy Martínez Rojas

Tutores: Lic. Liesner Acevedo Martínez

Lic. Rafael Arturo Trujillo Rasúa

Ciudad de La Habana, Junio 2008.
“Año 50 de la Revolución”

“Tan sólo hace falta una pequeña idea, para hacer un gran sueño realidad."
Fidel Castro Ruz.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ___ días del mes de _____ del año _____.

Yuneimy Tellez Pérez

Yudislainy Martínez Rojas

Lic. Liesner Acevedo Martínez

Lic. Rafael Arturo Trujillo Rasúa

_____

Firma del Autor

_____

Firma del Tutor

_____

Firma del Autor

_____

Firma del Tutor

Datos del contacto

Tutor: Lic Liesner Acevedo Martínez

frodo@uci.cu

Tutor: Lic. Rafael Arturo Trujillo Rasúa

trujillo@uci.cu

Autora: Yuneimy Tellez Pérez

ytellez@estudiantes.uci.cu

Autora: Yudislainy Martínez Rojas

ymartinez@estudiantes.uci.cu

AGRADECIMIENTOS

Con el afán de agradecer a todos aquellos que me han brindado su apoyo incondicional, sería imperdonable olvidar nombres, aun así, aunque puedo omitir algunos, llegue a muchas personas mi gratitud por el empeño y confianza que depositaron en mí. Hoy, un sueño colectivo se hace realidad y una ruta más queda vencida con la ayuda de quienes en todo momento son sustento firme en mi corazón.

Llegue el agradecimiento a mi abuela Santa, a mi mamá Zoraida a mi papá Jorge y a toda mi familia en general.

A Lissy y su mamá, Tere, Jose, Yovi, Odi, Yasmany, Yasniel, Yudi, Lore, Olgui y Mari por ser parte de mi familia.

A Alex por estar a mi lado todos estos años apoyándome y ayudándome, a él por ser mi otra mitad.

A Toda mi familia por estar a mi lado dándome el apoyo que siempre necesité.

A mi mamá Maribel ¨la persona más especial en mi vida¨, quien siempre confió en mí, me apoyó y nunca dejó que me sintiera sola, a quien le agradezco su dedicación, amor, por quien siempre traté de alcanzar mis sueños, que también eran los suyos.

A mis abuelitos, Pedro e Isabel que desde pequeña fueron como mis padres, quienes me mimaron y me guiaron siempre por el mejor camino con todo el amor del mundo.

A mi querido padre por ser mi ejemplo, por enseñarme todos los días a ser una mejor persona, por ser mi fuerza, mi guía.

A mi hermanito que tanto quiero y a mi mamá Midelma por su cariño.

A mis amigas (os) Dayamis, Liyanis, Licet, Yurima, Yarmay, Yasenia, Yamirita, Alexis, Jeanlup que han compartido junto a mí inolvidables momentos.

A Lázaro un gran amigo mayor.

A mi amigo Palmero por estar dispuesto siempre a ayudarme en los momentos más difíciles.

A mis tutores por su ayuda incondicional.

YUDI.

A mis tíos Norge y Ernesto, me han hecho sentir como su hija.

A Dannier, por estar a mi lado estos años.

A todos mis amigos en especial a Kun, Isa, Yane, Aylin, Yayi, Mar, Jen, Zoili, Evy, Rosy, Ide, Lorna.

A todos aquellos que de una manera u otra estaban al tanto del trabajo de la tesis, a Roberto, Lázaro.

A mis tutores, Trujillo y Liesner por su ayuda.

Yuneimy

DEDICATORIA

A mi abuela Santa por su ternura y apoyo constante, a mis padres por sus esfuerzos que hoy son mis logros y a mi familia en general.

YUDI.

A mi familia quien me enseñó desde pequeña el sendero del mejor camino. Principalmente a mi mamá Maribel por ser la persona más especial conmigo, por ser más que madre, mi amiga. A mi papito "mi tito", y mis abuelos Pedro e Isabel que con mucho cariño me dieron los mejores consejos y las fuerzas para seguir adelante. También a mi hermanito y a mi mamá Midelma por todo su amor.

Yuneimy

RESUMEN

En el presente trabajo se propone una implementación distribuida de la metodología de Hipersuperficies de Múltiples Mínimos (MMH). Procedimiento de gran importancia en la Bioinformática para calcular, a partir de un número de agregados moleculares en un espacio multidimensional dado, las energías de asociación estadísticas y la(s) estructura(s) más favorecida(s) energéticamente en dicho espacio.

MMH demanda un elevado cómputo que hace que las implementaciones comunes en una máquina monoprocesador demoren un tiempo prolongado, en consecuencia, es poco práctico realizar los cálculos químicos-teóricos utilizando la metodología de Hipersuperficies de Múltiples Mínimos en un único ordenador. Para minimizar los tiempos de respuesta de MMH, se propone una variante distribuida que hace uso del Sistema de Cómputo Distribuido (JDSC) una plataforma para hacer cálculos distribuidos el en redes de computadoras heterogéneas.

Palabras Clave:
MMH, agregados moleculares, Java, Sistema Java de Cómputo Distribuido, energías de asociación estadísticas.

ÍNDICE

ÍNDICE DE FIGURA

INTRODUCCIÓN

Con el desarrollo de las ciencias actuales aparecen frecuentemente problemas grandes y complejos que sin el uso de la computación no sería posible tratarlos. En muchas ocasiones las soluciones a estos problemas de "gran desafío" tienen una elevada demanda de cómputo y su respuesta debe darse en un tiempo limitado o incluso real; en estos, el uso de una sola computadora no sería factible. En este sentido la tendencia del mundo actual va dirigida hacia dos direcciones fundamentales:

Existen mercados en el área de hardware en el mundo que desarrollan microprocesadores cada vez más rápidos y potentes pero a precios muy altos. Estos microprocesadores hacen uso del paralelismo transparente, como el microprocesador Xeon que funciona como si constara de dos microprocesadores virtuales, por lo que puede dividir el trabajo en procesos que se pueden planificar y ejecutar de forma independiente y el microprocesador Itanium que puede ejecutar simultáneamente diferentes instrucciones de un programa.

La otra dirección es la de conexión de varios computadores en red, no necesariamente ubicadas en el mismo lugar y que cooperan para resolver un problema común. En esta dirección se destacan los Sistemas de Computación Paralelos **(SP)** y Distribuidos **(SD)**, una vía de solución más asequible al brindar alta potencia de cálculo sin necesidad de invertir grandes sumas de dinero. Esto ha llevado a que instituciones u organizaciones creen su propia infraestructura de computación distribuida.

En la Bioinformática surgen frecuentemente problemas que requieren de gran poder de cómputo, rama surgida a finales de los años 80 y principio de 1990, cuando comenzó a estructurarse el "Proyecto Genoma Humano" y que se encarga del estudio de informaciones biológicas a partir de la teoría de la información, la computación y las matemáticas involucrando dos ciencias: la Biología y la Computación. [1]

Si se tiene en cuenta que en Cuba la salud del pueblo es prioridad, resolver estos problemas en la Bioinformática, de forma más rápida y eficiente, acelerarían las investigaciones, los resultados y por consecuencia la calidad de vida de nuestro pueblo. Se debe descartar la alternativa de adquisición de Supercomputadoras debido a la situación económica y dirigir los esfuerzos a usar infraestructuras distribuidas aprovechando más los recursos existentes.

En la facultad de Bioinformática de la Universidad de las Ciencias Informáticas (UCI) existen varios proyectos dirigidos a apoyar investigaciones en esta rama de la salud, en particular *BioGrid*, planteándose la idea de aplicar la Computación Distribuida a problemas de alta demanda de cómputo, haciéndose uso de la plataforma "Sistema Java de Cómputo Distribuido'' que representa para el país, una alternativa como solución tecnológica que se demanda.

En la actualidad, uno de los problemas que demandan gran cómputo en la Bioinformática es la Metodología de Hipersuperficies de Múltiples Mínimos, que se aplica para explorar espacios multidimensionales de agregados moleculares con el objetivo de determinar sus energías de asociación estadísticas y obtener la o las estructuras más favorecidas energéticamente en todo el espacio considerado. Para lograr esto, se establecen varios pasos y se emplean tres aplicaciones: *Granada*, *MOPAC* y *Q3*. En particular, el MOPAC recibe un gran número de datos (*conformaciones*) que analiza de forma secuencial utilizando un gran % de la CPU, como consecuencia, al aplicar la metodología en un ordenador, el tiempo de espera de los resultados es prolongado.

Por ejemplo, al utilizar el MOPAC para procesar secuencialmente el fichero SOLVATED.MOP que contiene 50 conformaciones correspondientes a la molécula de Catequina de aproximadamente 211 átomos, los cálculos con cada conformación demoran aproximadamente 32 horas siendo el tiempo de procesamiento del fichero completo aproximadamente 2 meses y 7 días.

Por esta demora en los tiempos de respuesta de la metodología MMH se plantea como **problema científico** de la investigación:

- ✓ *¿Cómo reducir los tiempos de respuesta de la metodología MMH?*

Teniendo en cuenta la demora de los cálculos y que no hay dependencias entre las conformaciones que analiza el MOPAC se plantea como **objeto de estudio**:

- ✓ *Sistemas distribuidos para el procesamiento y cálculo masivo de información.*

El objeto de estudio se enmarca en el **campo de acción:**

- ✓ *Sistema Java de Cómputo Distribuido.*

Con el fin de solucionar el problema planteado anteriormente se define como **objetivo general** del presente trabajo:

- ✓ *Desarrollar un sistema distribuido sobre la plataforma JDCS, que reduzca los tiempos de respuesta de la metodología MMH.*

Para darle cumplimiento al objetivo general se propusieron los siguientes **objetivos específicos**:

- ✓ *Levantamiento de Requisitos*.
- ✓ *Diseñar el sistema propuesto.*
- ✓ *Implementar el sistema propuesto.*
- ✓ *Evaluar la eficiencia del sistema implementado.*

Para alcanzar estos objetivos se plantearon las siguientes **tareas:**

- ✓ *Estudio de la Metodología de Hipersuperficies de Múltiples Mínimos.*
- ✓ *Análisis del estado del arte de los sistemas distribuidos.*
- ✓ *Análisis del estado del arte: distribución del MOPAC sobre entornos distribuidos.*
- ✓ *Estudio de la tecnología Java RMI (Invocación a Métodos Remotos).*
- ✓ *Estudio de la plataforma JDCS para el trabajo distribuido.*
- ✓ *Implementación del sistema.*
- ✓ *Realización de pruebas estadísticas.*

El documento está estructurado de la siguiente forma:

Capítulo 1: Estudios preliminares: En este capítulo se exponen algunas definiciones de Sistemas Distribuidos que constituyen el objeto de estudio de la investigación, presentándose algunos ejemplos de aplicaciones de estos en la Bioinformática. Finalmente se hace un acercamiento a la metodología

MMH por dar origen al problema de la investigación y a las diferentes aplicaciones que utiliza en la realización de los cálculos.

Capítulo 2: Materiales y Métodos: En este capítulo se presentan algunos elementos importantes: del lenguaje de modelado UML, de las principales metodologías, de las herramientas tanto de modelado como de programación, de algunas plataformas distribuidas de mayor tendencia en el mundo actual, destacando cuales se usaron para dar solución al problema.

Capítulo 3: Resultados y Discusión: En este capítulo se presentan los aspectos más importantes de la Ingeniería del sistema para su mejor entendimiento, los casos de usos, modelo de diseño, modelos de secuencia, despliegue, entre otros. Por último, se presentan algunos resultados experimentales del sistema desarrollado que muestran una reducción en el tiempo de respuesta de los resultados de la metodología MMH.

CAPÍTULO 1: ESTUDIOS PRELIMINARES.

En éste capítulo se describen aspectos claves para la comprensión de los Sistemas Distribuidos, los cuales constituyen una alternativa para resolver problemas científicos. Se presentan ejemplos de aplicación de estos sistemas en la Bioinformática, en particular, para reducir los tiempos de respuesta del MOPAC pues constituye la aplicación que provoca la demora en los cálculos de la metodología MMH. Se describe el funcionamiento de esta metodología y las aplicaciones Granada, MOPAC y Q3 que utiliza.

1.1 Sistemas Distribuidos.

Existen varios conceptos en el mundo de Sistemas Distribuidos, entre los que se destacan:

- ✓ Un Sistema Distribuido no es más que una colección de computadoras independientes que aparentan ser para el usuario del sistema una única computadora. [2]

- ✓ Sistemas cuyos componentes hardware y software, que están en ordenadores conectados en red, se comunican y coordinan sus acciones mediante el paso de mensajes, para el logro de un objetivo. Se establece la comunicación mediante un protocolo prefijado por un esquema cliente-servidor. [3]

Hay dos aspectos importantes que un sistema debe cumplir para que sea distribuido, uno está relacionado con el hardware, pues las máquinas deben ser autónomas, y el otro, está relacionado con el software: para los usuarios, el sistema deberá ser una única computadora. Con el fin de un mejor entendimiento de esta definición se presentan a continuación dos ejemplos:

Ejemplo 1: Se consideran varias unidades de trabajo en un departamento de una Universidad y un usuario en otro local que desde su unidad personal solicita una tarea. El sistema dinámicamente deberá seleccionar del departamento cual unidad (ociosa o trabajando) le dará solución a la tarea de acuerdo a su necesidad.

Ejemplo 2: Se considera una línea de ensamblaje dentro de una fábrica donde los trabajadores son robots, cada uno tiene asociado una computadora para el manejo de su visión, comunicación y otras

actividades. Cuando un robot que está trabajando directamente en la línea considera que una pieza a instalar está defectuosa, solicita el reemplazo de esta (la fábrica consta de almacenes de piezas y robots trabajadores en estos); si la pieza es reemplazada sin importar "quién" la suministró y el primero de los robots no alteró su estado de trabajo entonces puede verse esta fábrica como un Sistema Distribuido.

Como puede notarse, en los dos ejemplos fue transparente para el usuario que intervino todo un sistema, en el ejemplo 1, para el usuario que hizo la solicitud de la tarea fue su propio ordenador quien proporcionó la solución y en el ejemplo 2, fue el primero de los robots quien realizó todo el ensamblaje. Aunque en la actualidad no se ha establecido un acuerdo, muchos autores plantean que entre los Sistemas Distribuidos se encuentran los Sistemas de Cómputo en Paralelo (SP) y los Sistemas de Cómputo Distribuido (SD).

Los **SD** se caracterizan por presentar un grupo de elementos: unidad computacional que puede ser un proceso, un procesador, un *switch*, entre otros. Estos elementos se interconectan a través de una red de comunicación, cooperan entre sí con el propósito de resolver una tarea común, y se comunican mediante la recepción y el envío de mensajes actuando de forma espontánea y "autónoma" [4]. En estos sistemas es común encontrase un servidor que monitorea los servicios para llevar a cabo todo el proceso, por lo que el modelo del sistema responde a un Modelo Cliente-Servidor.

Los **SP** interconectan un grupo de elementos que se comunican para realizar una tarea; a diferencia de los **SD**, tienen como principio dividir las aplicaciones en subtareas que son resueltas concurrentemente. Para aplicar el principio de la Computación Paralela se debe pensar en cómputos muy largos, con el requerimiento de ser posible dividirlos en subtareas que puedan procesarse de forma independiente.

Aunque ambos están concebidos para fortalecer la capacidad de cálculo en una red de computadoras, si se tiene en cuenta que el objetivo de los **SP** es alcanzar el máximo *speedup* en la solución de un problema, seguramente, muchas organizaciones se inclinarían por esta alternativa, sin embargo, puede ser compleja su aplicación. Entre otros aspectos, para aplicar los **SP**, se necesita comprender completamente el algoritmo secuencial que da solución al problema para poderlo dividir en subtareas totalmente independientes, y se requieren formas de coordinar los accesos a recursos compartidos,

por ejemplo, MPI (Message Passing Information), que coordina los accesos mediante el modelo de paso de mensajes.

Otros de los aspectos a tener en cuenta, es que en estos sistemas no es frecuente que el número de procesadores o la forma de interconexión cambie en el transcurso del tiempo, por lo menos durante la ejecución, además, los elementos en la red de interconexión deben estar separados por pequeñas distancias.

Por otro lado, si se piensa en un **SD**, el programador no deberá conocer el código del programa secuencial que se quiera compartir, a diferencia de los **SP**, para reducir los tiempos de cálculos. Su principio es ejecutar en los clientes las mismas instrucciones con diferentes datos aprovechando los recursos ya existentes.

Aunque no suelen ser los **SD** completamente rápidos, sus elementos están diseñados para actuar de forma independiente, por lo que si falla uno, no implica que falle el sistema completo, estos pueden estar separados por centenares de metros o kilómetros.

Los **SD** permiten además, la aparición de recursos o su desaparición, incluso en tiempo de ejecución. Las instrucciones se pueden ejecutar en arquitecturas heterogéneas, y se admiten varias aplicaciones a la vez. Dentro de estos sistemas se encuentra los Sistemas de *Computación Grid,* los cuales tienen como principio no solo la compartición de datos sino de otros recursos como: *Computadores, Redes e Instrumentos.* En estos sistemas la seguridad es requerimiento básico para su funcionamiento [5].

1.1.1 Ventajas de los Sistemas Distribuidos.

Entre las ventajas de los Sistemas Distribuidos se destacan:

- ✓ **Buena relación costo-beneficio**: En la actualidad es común encontrarse procesadores que ejecuten un gran número de instrucciones, en consecuencia el precio de estos es elevado. Los sistemas distribuidos ofrecen una mejor relación costo-beneficio: ejecutan de igual modo un grupo de instrucciones y su precio los hace más asequibles.
- ✓ **Velocidad:** Con la aplicación de los sistemas distribuidos se logra mayor velocidad de cómputo que con un único computador.

- ✓ **Independencia de fallo de los recursos**: El fracaso de algún recurso lógico o físico no implica que se destruya el sistema.
- ✓ **Fiabilidad o Confiabilidad:** Después de la ocurrencia de algún fallo el sistema debe proporcionar los medios para reconfigurarlo o debe reasignar la tarea.
- ✓ **Distribución inherente:** Un grupo de personas ubicadas en lugares diferentes trabajan juntas para realizar una tarea común.
- ✓ **Compartición de datos:** Permiten el acceso de varios usuarios a datos comunes, como servidores de bases de datos, servidores de cómputo, servidores de realidad virtual, servidor de información multimedia, entre otros.
- ✓ **Compartición de dispositivos:** Permite a muchos usuarios compartir costosos periféricos como impresoras láser en colores, dispositivos de almacenamiento masivo de archivos, entre otros.
- ✓ Flexibilidad y extensibilidad: Los Sistemas Distribuidos son capaces de crecimiento incremental y proporcionan la extensión o modificación. Se adaptan a un ambiente cambiante sin desestabilizar sus operaciones.

1.1.2 Desventajas de los Sistemas Distribuidos.

Uno de los problemas está relacionado con el software: aunque en la actualidad los **SD** están tomando gran fuerza, no se tiene mucha experiencia en su diseño, implementación y su uso. Es difícil aún dar respuestas a preguntas como: ¿qué tipo de sistema operativo es el más apropiado?, ¿qué lenguaje de programación usar?, ¿cuáles aplicaciones son apropiadas para el sistema?, ¿cuánto deberían conocer los usuarios sobre la distribución?, ¿cuánto deberían hacer el sistema y los usuarios?.

El segundo problema está relacionado con la red de comunicación, se pueden perder mensajes. Con el objetivo de recuperarlos el sistema deberá incluir software especiales y la red puede llagar a recargarse, cuando esto sucede debe ser reemplazada o una segunda red debe ser agregada. Una vez que el sistema dependa de la red, su saturación puede negar muchas de las ventajas a lograr para lo cual el sistema distribuido fue construido.

Finalmente, el intercambio de datos puede convertirse en un problema ya que las personas pueden acceder a todos los datos del sistema incluyendo aquellos que no les son permitidos. Por esta razón, la seguridad es frecuentemente un problema.

1.2 Aplicación de los Sistemas Distribuidos en la Bioinformática.

La Bioinformática utiliza las tecnologías de la información para captar, organizar, analizar y distribuir informaciones biológicas con el objetivo de responder preguntas complejas en Biología. Es el resultado de la convergencia de la informática con la Bioquímica, la Genética, la Biotecnología, la Biología molecular, entre otras, posibilitándoles valorar de manera integrada los datos que aceleran cada vez más los procesos de investigación. [6]

El uso de las computadoras para resolver cuestiones biológicas comenzó con el desarrollo de algoritmos y su aplicación al estudio de las interacciones de los procesos biológicos y las relaciones filogenéticas entre diversos organismos. En los últimos años, el incremento de la cantidad de secuencias disponibles de proteínas, ADN, entre otras, la alta demanda de cómputo y la complejidad de las técnicas que emplean dichas computadoras para la adquisición y el análisis de los datos conlleva al uso de los Sistemas de Cómputo Distribuido y Paralelo con el objetivo de procesar de forma más eficiente posible las informaciones biológicas.

Por ejemplo, el grupo de Computación Distribuida Heterogéneo de la Universidad Nacional de Irlanda ha desarrollado varias aplicaciones bioinformáticas en entornos distribuidos, tales como **MultiPhyl** [7] que permite al usuario procesar cientos o miles de aminoácidos o nucleótidos alineados simultáneamente y realiza computacionalmente tareas intensivas: modelo de selección, árboles de búsqueda, e inicio de flejado (bootstrapping) de cada una de las alineaciones. Otros ejemplos son **DPRml, DSEARCH.**

DSEARCH: permite que el usuario distribuya la tarea de buscar en la bases de datos el excedente de las secuencias de un sistema de procesadores semi-ociosos. La razón de la buscar en la base de datos es identificar regiones similares del DNA, el RNA, o las secuencias de la proteína.

También se han desarrollado aplicaciones en entornos distribuidos para reducir el tiempo de respuesta del MOPAC (*sección 1.3.2*): programa para el cálculo de orbitales moleculares por métodos semi-empíricos, entre estas aplicaciones se encuentran:
"Paralización del MOPAC con Sistemas de Cómputo Distribuidos mediante el framework AVS" [8] que consiste en dividir la interfaz de usuario en dos módulos **mopacavs** y **geomcntl**. El módulo **mopacavs** es la verdadera interfaz del usuario, sirve como controlador y servidor de las tareas paralelizadas del MOPAC, también es el puente entre MOPAC y AVS; recibe la información geométrica del átomo interior y los envía al módulo **geomcntl** el cual transforma esta información en el objeto geométrico AVS. La construcción AVS de la vista geométrica del módulo toma el objeto y lo visualiza en la pantalla. Para lograr que los módulos trabajen, se necesita que estén conectados en una red de AVS.

En la Universidad de las Ciencias Informáticas se desarrolló el *"Módulo para el cálculo distribuido de mecánica molecular y mecánica cuántica"* con el que se redujo el tiempo de procesamiento del MOPAC en más de un 98% haciendo uso de la plataforma JDCS. En pruebas realizadas a tres lotes de moléculas 3000, 5000 y 20000 que disminuyeron sus tiempos de cálculo de 110, 236 y 464 horas a 1.5, 4.1 y 6.6 horas respectivamente [9].

En la actualidad, como parte del Proyecto *BioGrid* de la facultad de Bioinformática de esta Universidad se desarrollan algunas aplicaciones para reducir los tiempos de respuestas de algunas aplicaciones como *Gaussian*, *Vega* y la metodología MMH. Esta última constituye el objetivo del presente trabajo.

1.3 MMH

La metodología de Hipersuperficies de Múltiples Mínimos permite la exploración completa del espacio multidimensional de agregados moleculares con la finalidad de determinar sus energías de asociación estadísticas y la obtención de la o las estructuras más favorecidas energéticamente en todo el espacio considerado [10]. Para esto utiliza varias aplicaciones: Granada, MOPAC y Q3.

Paso #1: En la realización de los cálculos primeramente se procesan varios ficheros **.car** haciendo uso del programa Granada, estos ficheros constituyen moléculas de entrada en coordenadas cartesianas, el primero corresponde a un soluto y el resto a disolventes. Un archivo nombrado **input** (sin extensión) debe estar presente en el directorio predefinido, el cual contiene los parámetros generales para la

corrida: el número de moléculas medioambientales o del solvente deseadas, la mitad de la dimensión sobre todos los ejes cartesianos de un cubo virtual, donde las moléculas medioambientales se pondrán aleatoriamente, el número de configuraciones a ser generadas, el número de moléculas medioambientales y la fracción molar de la molécula medioambiental.

Lo que se hace en este paso es considerar un ensamble canónico compuesto por M celdas o supermoléculas de configuraciones diferentes del solvente rodeando al soluto, a temperatura y volumen constante. El programa Granada se utiliza para generar de forma aleatoria estas configuraciones. Estas M celdas conforman un solo fichero de cálculo de extensión **.mop** de entrada al programa de minimización de energía MOPAC.

Paso #2: Las M celdas son optimizadas a través de un gradiente estándar de minimización, usando en este caso un hamiltoniano semi-empírico. Este segundo paso produce un fichero **.rsm** que contiene un conjunto de celdas típicas de mínima energía en el espacio de configuraciones de la interacción soluto–solvente. Se obtiene entonces, la energía, de cada celda, en el estado del ensemble obtenido. La principal atención se centra en buscar un reducido conjunto de celdas que puedan representar las contribuciones más significativas del estado a todo el sistema en estudio.

Las optimizaciones se realizan mediante el programa MOPAC el cual analiza de forma secuencial cada una de las celdas. El tiempo de espera de los resultados es directamente proporcional al tamaño de las celdas, puede demorar horas, días e incluso meses.

Paso #3: El fichero **.rsm** y un fichero **input.q** se necesitan estar presentes en el directorio predefinido para la ejecución del programa Q3 que calcula la magnitud de la termodinámica de asociación molecular. La primera línea del **input.q** contiene un título explicativo y la segunda línea un grupo de valores separados por coma: eref, anum, temp, conv, tolind, ifile.

- ✓ **eref:** Energía de referencia (en ev) como suma de las energías totales de las moléculas integrantes aisladas y optimizadas con las mismas opciones MOPAC que los agregados moleculares.
- ✓ **anum**: Número de moléculas de referencia que se usa para poder expresar los valores por mol y por molécula de referencia.

- ✓ **temp**: Temperatura (en K) a la que se desean los cálculos termodinámicos.
- ✓ **conv**: Energía de convergencia de las entalpías molares.
- ✓ **tolind**: Nivel de discriminación para el coeficiente de semejanza de Tanimoto.
- ✓ **ifile**: Nombre del fichero del tipo .RSM.
- ✓ Contiene además tantas líneas como celdas son generadas, cuyos valores son corresponden a la energía de referencia. La salida del programa Q3 consiste en 2 ficheros cuyos nombres son **output.q3** y **output.dt3**.

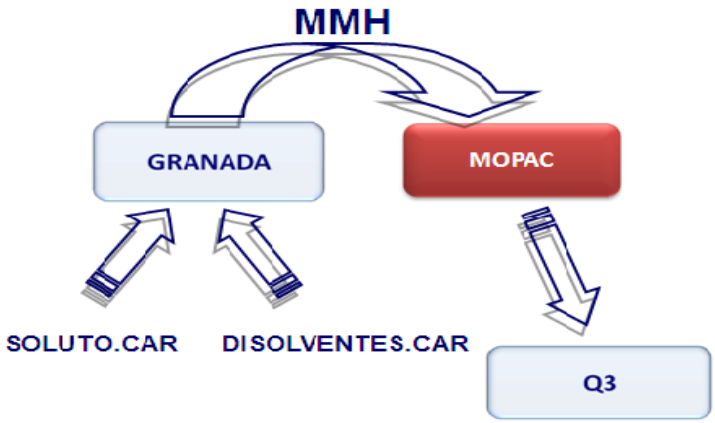


Figura 1: Fases de la metodología MMH.

Granada

Es un programa ejecutable de MS-DOS diseñado para leer archivos de coordenadas cartesianas o internas que contienen datos atómicos de moléculas con el fin de generar celdas donde las moléculas medioambientales están rodeando o están en las cavidades del sistema central de forma completamente aleatoria.

MOPAC

El MOPAC es un programa para el cálculo de orbitales moleculares por métodos semi-empíricos de propósito general, para el estudio de estructuras químicas en estados sólidos, moléculas aisladas, tanto en estado base como excitado, en medio de disolventes y reacciones químicas [11]. Las

aplicaciones hamiltonianas semi-empíricos MNDO, MINDO, AM1 y PM3 se utilizan en la parte electrónica del cálculo para obtener orbitales moleculares, el calor de formación y otros datos derivados relacionados con la geometría molecular.

Utilizando estos resultados, el MOPAC calcula el espectro de vibraciones, cantidades termodinámicas, sustituciones isotópicas, constantes de fuerza para moléculas, radicales, iones y polímeros, etc. Cuenta con rutinas de localización de estados de transición y de optimización de esos estados, para el estudio de reacciones químicas. Para que los usuarios obtengan el mayor provecho del programa, deben entender cómo trabaja, cómo entrar los datos, cómo interpretar los resultados y qué hacer cuando hay problemas (fallas, excepciones).

Aunque el MOPAC emplea muchos conceptos de la química quántica y la termodinámica, y utiliza algunos métodos matemáticos muy avanzados, el usuario no necesita estar familiarizado con estos aspectos del programa. Los datos obtenidos se guardan de una manera tan simple como es posible para que los usuarios presten su atención a la química implicada en el fenómeno estudiado y se despreocupen de los aspectos puramente matemáticos y más teóricos de la química cuántica y la termodinámica.

La forma más sencilla de describir cómo trabaja el MOPAC es que el usuario genera los datos con los cuales describe un sistema molecular en el fichero de entrada, así como los tipos de cálculos a realizar y las salidas (de resultados) que desee. Entonces, el usuario utiliza el MOPAC para llevar a cabo el cálculo utilizando estos datos. Finalmente, el usuario ordena la ejecución y obtiene los ficheros texto de salidas solicitadas a partir del fichero de entrada de datos, que le permiten interpretarlos.

Q3

Es un programa ejecutable de MS-DOS diseñado para leer archivos (denominados **input.q)** que contienen las energías de las celdas generadas después de la optimización por el programa MOPAC, y las del sistema de referencia, así como la indicación de cuál fichero contiene la información espacial para las comparaciones entre estructuras con posibilidades de ser redundantes, según los índices de Tanimoto.

1.4 Conclusiones

Después de un análisis detallado de la metodología de Hipersuperficies de Múltiples Mínimos se concluyó que era necesario implementar un Sistema Distribuido que minimizara sus tiempos de respuestas debido a que los cálculos que realiza son de gran importancia en la Bioinformática. Aunque se han desarrollados **SD** que resuelven el problema de alta demanda de cómputo del MOPAC, no existe en la actualidad una solución que resuelva el problema de gran demanda de cómputo de la metodología MMH por lo que el presente trabajo pretende brindar este tipo de solución.

CAPÍTULO 2: MATERIALES Y MÉTODOS.

En este capítulo se exponen los elementos más importantes que se tuvieron en cuenta para la selección del lenguaje y las herramientas de modelado, las herramientas de programación, así como algunos elementos de las principales metodologías de desarrollo de software. Se presentan algunas de las plataformas de cómputo distribuido que constituyen en la actualidad alternativas para resolver problemas que demanden gran potencia de cálculo. Finalmente en el capítulo, se exponen algunos aspectos de la tecnología Java RMI por ser el principio de funcionamiento de la plataforma JDSC que se utiliza para dar solución al problema de la investigación.

2.1 Lenguaje de Modelado y Metodología.

Lenguaje de modelado UML

El Lenguaje Unificado de Modelado (Unified Modeling Language, UML) es una notación estándar para representar los planos de un sistema de software. Está compuesto por diversos componentes (elementos, relaciones y diagramas) que son los que le dan vida al lenguaje unificado y se encuentran en constante relación. Cuenta con reglas para combinar los elementos las cuales pueden utilizarse para:

- ✓ *Visualizar*: Es la manera de representar gráficamente los modelos que en muchas ocasiones los programadores piensan, escriben en papel, para realizar sus análisis. Es muy útil para que los clientes puedan entenderlos. Por otra parte si el desarrollador del código no deja información escrita de lo que escribió en el papel o lo que pensó la información se pierde. Un modelo explícito facilita la comunicación.

- ✓ *Especificar*: Significa construir modelos si ambigüedades más bien precisos y completos. UML cubre la especificación de todas las decisiones de análisis, diseño e implementación que deben realizarse al desarrollar y desplegar un sistema de software.

- ✓ *Construir:* Los modelos de UML se pueden conectar con varios lenguajes de programación como, Java, C++, Visual Basic, entre otros.

- ✓ *Documentar:* Una organización de software que trabaje bien produce toda clase de artefactos además de código ejecutable. Estos artefactos incluyen: Requisitos, Arquitectura, Diseño, Código fuente, Planificación de proyectos, Prototipos, Pruebas, Versiones.

Es un lenguaje muy explícito, incluye las vistas necesarias que facilitan el desarrollo de los sistemas de software. UML no es un lenguaje de programación. Las herramientas pueden ofrecer generadores de código para varios lenguajes de programación, así como construir modelos por ingeniería inversa a partir de programas existentes. Está muy bien validado para la representación del paradigma POO, es bastante independiente del proceso, por lo que se puede aplicar a cualquier sistema de software.

Metodología Rational Unified Process (RUP)

RUP se caracteriza por tomar los mejores elementos de metodologías anteriores. Es un proceso flexible pues se le pueden hacer determinadas variaciones en dependencia del tipo de sistema que se desee desarrollar. Se encuentra preparado para desarrollar grandes y complejos proyectos. Describe detalladamente todas las actividades, roles, responsabilidades, productos de trabajo y herramientas para definir el quién, qué y cuándo y cómo en un proyecto de desarrollo de software. Está orientada a objetos y utiliza UML como notación para la representación visual. Sus principales características son:

- ✓ Dirigido por los casos de uso.
- ✓ Centrado en la arquitectura.

- ✓ Iterativo e incremental.

Dirigido por los casos de uso: Le proporcionan al proceso de desarrollo seguir una guía al avanzar a través de una serie de flujos de trabajo donde los modelos que se obtienen como resultado de estos, representan la realización de los casos de uso.

Centrado en la arquitectura: La arquitectura muestra una visión completa del sistema que se desarrolla por lo que le ofrece tanto a los clientes como a los desarrolladores una idea clara de este. Se toman decisiones concernientes a cómo debe ser construido el sistema y en qué orden. Incluye elementos de calidad, rendimiento, reutilización y capacidad de evolución. La arquitectura está asociada a propiedades o cualidades del sistema operativo, de los softwares implicados en el desarrollo, protocolos, que constituyen requisitos no funcionales del Sistema.

Iterativo e incremental: Las interacciones se refieren a pasos en cada flujo de trabajo y los incrementos, al crecimiento del producto. Se van a desarrollar un grupo de interacciones que permitirán el cumplimiento de cada flujo de trabajo y que se logre al final de cada una de las interacciones un incremento.
En RUP se agrupan las actividades en 9 flujos de trabajo principales. En la siguiente figura se representa el proceso en el que se grafican los flujos de trabajo y las fases donde intervienen dichos flujos.

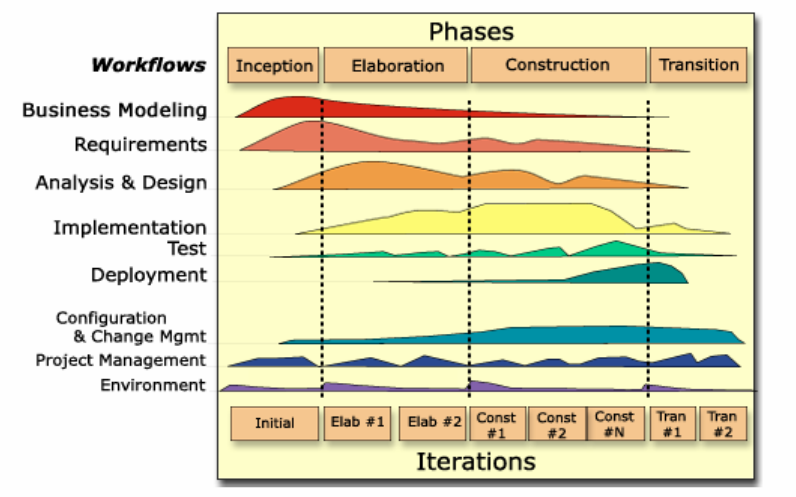


Figura 2: RUP en dos dimensiones.

Metodología Basic Unified Process (BUP)

Es una versión de la metodología RUP para proyectos pequeños y ágiles. Constituye un proceso mucho más simple pues la mayoría de las partes optativas de RUP se han excluido y se han unido muchos elementos. Sí se conservan las características esenciales: desarrollo iterativo, casos de uso, ente otras. Puede ser usado como está, ser extendido y personalizado para propósitos específicos.
Se caracteriza por cuatro principios básicos:

- ✓ Colaboración para alinear los intereses y un entendimiento compartido.
- ✓ Balance comprobatorio de las prioridades (necesidades y costos técnicos) con el objetivo de maximizar el valor para los interesados.
- ✓ Enfoca en particular la arquitectura para facilitar la colaboración técnica, reducir los riesgos y minimizar excesos y trabajo extra.

- ✓ Evolución continua para reducir riesgos, demostrar resultados y obtener retroalimentación de los clientes.
- ✓ Descarta la modelación del negocio, la planificación, el ambiente, la dirección de requisitos avanzada, la dirección de la configuración, porque estas preocupaciones son consideradas avanzadas para un proyecto pequeño o se manejan por otras áreas de la organización, no por el equipo del proyecto. Se enfoca en los requisitos, la arquitectura, el desarrollo, las pruebas, dirección del proyecto y dirección de cambio.

2.2 Herramientas de modelado y programación.

Rational Rose

Rational Rose es una herramienta software para el modelado visual mediante UML de sistemas informáticos. Permite especificar, analizar y diseñar el sistema antes de codificarlo. Rational Rose es la herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases.
Algunas características de Rational Rose:

- ✓ Posee una interface fácil de manipular.
- ✓ Permite la posibilidad de organizar por paquetes dentro de cada vista con el objetivo de organizar mejor el proyecto
- ✓ Mantiene la consistencia de los modelos del sistema software.
- ✓ Chequeo de la sintaxis UML.
- ✓ Generación de documentación automáticamente.
- ✓ Generación de código a partir de los modelos.

- ✓ Permite la ingeniería Inversa.

Visual Paradigm

Visual Paradigm es una herramienta visual de Ingeniería de Software para el modelado, brinda una colección de menús, barras de herramientas y ventanas que forman el área de trabajo, lo cual permite crear diferentes tipos de diagramas en un ambiente completamente visual. Está diseñada para una gran variedad de usuarios, incluyendo Ingenieros de Software, Analistas de Sistema, Analistas de Negocios, por nombrar algunos. Provee una manera intuitiva para llevar a cabo sistemas de análisis y diseño orientado a objetos. Soporta los últimos estándares de anotaciones de Java, UML, provee soporte para la generación de código y la ingeniería inversa para Java, se integra con algunas herramientas de este lenguaje, como: Eclipse, Netbeans, Jbuilder, Oracle, entre otras. Es una herramienta gratis y multiplataforma.

Java

Java fue diseñado e implementado por SUN MICROSYSTEMS. Al principio fue pensando como un lenguaje para programar pequeños aparatos (electrodomésticos y artefactos electrónicos) pero más tarde se convirtió en un lenguaje de programación poderoso. En la actualidad se considera como uno de los lenguajes de programación más usados.

- ✓ Es fácil de aprender y programar, sobre todo si se tienen conocimientos de lenguajes orientados a objetos. Oculta dificultades presentes en lenguajes como C y C++, entre otras características:
- ✓ No implementa punteros, aunque los maneja interna y transparentemente.
- ✓ El manejo de la memoria no es un problema, la gestiona el propio lenguaje y no el programador.
- ✓ No permite programar siguiendo otro paradigma que no sea el paradigma orientado a objetos.

- ✓ Después de compilado un programa, se crea un fichero que incluye un pseudocódigo prácticamente al nivel de código máquina (bytecodes) el cual se interpreta por una JVM (Java Virtual Machine). De esta forma los programas creados en Java se pueden ejecutar en cualquier tipo de sistema operativo y en cualquier tipo de procesador que incorpore la máquina utilizada.
- ✓ Permite desarrollar aplicaciones con pocos errores, fundamentalmente porque la gestión de memoria y de punteros no la realiza el programador. El lenguaje contiene estructuras para la detección de excepciones y obliga al programador a escribir código fiable mediante la declaración de excepciones posibles para una determinada clase reutilizable. Verifica el código al mismo tiempo que se escribe, y antes de ejecutarse.
- ✓ Permite crear diferentes tipos de aplicaciones Java Applets, Aplicaciones Standalone, Paquetes, Java Servlets, por nombrar algunas.
- ✓ Facilita la ejecución de tareas concurrentes dentro de un programa mediante los Multi-Threading.
- ✓ Permite la distribución de objetos en la red, para esto proporciona protocolos como CORBA, (Arquitectura intermediaria para solicitar objetos comunes) y RMI (Invocación de Métodos Remotos).
- ✓ Controla la seguridad frente al acceso a recursos del sistema y permite gestionar permisos y criptografía.
- ✓ Es un lenguaje público.

Eclipse

Está integrado por un núcleo y muchos plug-ins los cuales interactúan mediante interfaces o puntos de extensión; de esta forma, las nuevas aportaciones se integran sin dificultad. Su uso más común es como un Entorno Integrado de Desarrollo (IDE, por sus siglas en inglés) para Java, pero puede adaptarse a cualquier lenguaje de programación como C, C++, soporta la Programación Orientada a Objeto (POO), su principal característica es la extensibilidad. La arquitectura de plug-ins del Eclipse permite además de integrar varios lenguajes, introducir otras aplicaciones que pueden resultar útiles durante el proceso de

desarrollo como pueden ser: herramientas UML, editores visuales de interfaces, ayuda en línea para librerías, entre otras características. Este entorno de desarrollo integrado ofrece el control del editor de código, del compilador y del depurador desde una única interfaz de usuario. Su misión consiste en evitar tareas repetitivas, facilitar la escritura de código correcto, disminuir el tiempo de depuración e incrementar la productividad del desarrollador. Estas tareas pueden realizarse de diferentes formas mediante la inclusión de asistentes para las tareas más habituales y mecánicas, además de gestores de archivos fuentes. Eclipse es una plataforma de código abierto implementado en Java que se puede ejecutar en diferentes sistemas operativos.

2.3 Plataforma distribuida.

La tecnología Grid es un tipo de **SP** y **SD** que permite compartir, seleccionar y agregar dinámicamente recursos autónomos que están dispersos geográficamente, en función de su disponibilidad, capacidad, rendimiento, coste o de las necesidades de los usuarios. "…*en una Grid los recursos no están administrados de manera centralizada, sino que funcionan de manera análoga a la red eléctrica, integrando y conectando ordenadores de alto rendimiento con distintas arquitecturas a través de la red, y reasignando los servicios en función de los picos de actividad*." [12]
Dentro de esta tecnología se encuentran varias infraestructuras, entre las cuales se destacan:

Biblioteca Java Computación Distribuida (JDCL, por sus siglas en inglés)

El sistema denominado Biblioteca Java Computación Distribuida (JDCL) se creó para facilitarle el trabajo a los desarrolladores de plataformas de cálculos distribuidos, pero este tenía grandes limitaciones como: la no existencia de una interfaz de usuario, excepciones manipuladas inadecuadamente, no se contaba con mecanismos de seguridad, la limitación de gestionar una sola aplicación distribuida a la vez. Producto

a la carencia de un planificador, el usuario está obligado a reiniciar todo el sistema para ejecutar un nuevo cómputo. [13]

Condor.

Condor es un sistema que desde 1986 se está utilizando en la programación paralela producto a su alto rendimiento en esta. [14] La calidad de muchos proyecto dependen del tiempo de respuesta de obtenido, esto se logra mediante un entorno de computación denominado HTC (High-Throughput Computing) en el que el punto clave es el uso eficiente de los recursos disponibles. Hoy en día también existen software como es el caso de Condor que se encarga de repartir el trabajo entre los distintos ordenadores interconectados a travésde una red de la siguiente manera: Un usuario envía un trabajo a Condor, Condor busca y encuentra una máquina disponible en la red y se ejecuta el trabajo en dicha máquina.

Características de Condor:

- ✓ Checkpoint y migración. Los usuarios de Condor tienen la garantía de que sus trabajos se llevarán a cabo, incluso si se modifica el entorno. Cuando una máquina que está ejecutando un trabajo enviado a Condor deja de estar disponible, se realiza un checkpoint a trabajo, y éste migra a otra máquina que esté disponible.
- ✓ Sistema de llamadas remotas. A pesar de ejecutar trabajos en máquinas remotas, el modo de ejecución del universo estándar de Condor preserva el entorno de ejecución local a través de las llamadas remotas. Los usuarios no se deben preocupar sobre los sistemas de ficheros disponibles en las estaciones remotas, ni en tener cuentas de usuario en ellas. El programa se comporta en Condor como si se estuviera ejecutando en la máquina originaria del trabajo.
- ✓ No es necesario modificar el código fuente de las aplicaciones. No se requiere una programación específica para ejecutar aplicaciones en Condor.

- ✓ Condor es capaz de ejecutar programas no interactivos. Los checkpoint y migraciones se realizan en Condor de manera transparente y automática, así como el uso de las llamadas remotas.
- ✓ Los trabajos se pueden ordenar. El orden de ejecución de los trabajos por motivos de dependencias entre ellos se puede manejar fácilmente. Los conjuntos de trabajos se especifican utilizando un grafo dirigido acíclico, donde cada trabajo viene representado por un nodo en el grafo. Los trabajos se envían a Condor en función de las dependencias dadas en el grafo.Condor permite computación Grid. Puesto que la computación Grid es ya una realidad, Condor la soporta.
- ✓ El mecanismo ClassAd de Condor proporciona un framework extremadamente flexible y expresivo para resolver peticiones de recursos según los recursos ofertados en el entorno. Los usuarios pueden pedir recursos necesarios y deseados para sus trabajos. Por ejemplo, un usuario puede pedir que un trabajo se ejecute en una máquina con 64 MB de RAM, pero sería deseable utilizar 128 MB. EL propietario de una estación puede fijar las preferencias de la estación a la hora de ejecutar trabajos para determinados grupos de usuarios.
- ✓ Condor soporta aplicaciones PVM y MPI. [15]

Sistemas Java de Cómputos Distribuido (JDSC).

El funcionamiento de la plataforma JDSC está representado por una arquitectura Cliente-Servidor, las computadoras que se comunican durante el cómputo para procesar los datos representan a los clientes, y el computador donde está montada la plataforma constituye el servidor, el que cual se encarga de monitorear todo el proceso. La meta principal de la plataforma es reducir la complejidad de programación que tienen otros **SP** y **SD**.

Para la invocación de métodos remotos se usa de forma trasparente al desarrollador la tecnología de Java RMI, de forma tal que para la programación de una aplicación distribuida sólo se necesita extender dos clases de Java e implementar sus métodos, la clase `DataManager` que funciona en los clientes y la clase `Algorithm` que funciona en el servidor.

En la clase `DataManager` se generan las unidades de trabajo para los clientes que realizan solicitudes de trabajo, se procesan todos los resultados devueltos por dichos clientes, se ajusta el particionamiento de las unidades del trabajo, se genera la información de estado del problema, y se termina el cómputo distribuido. Para poder darle solución a un problema, se debe implementar una clase que herede de la `DataManager` en la que el constructor no recibe parámetro alguno y su iniciación se realiza al leer de un fichero.

DataManager

```
public class ExtendedDataManager extends DataManager {

  //declaración de los atributos...

  public ExtendedDataManager() throws Throwable {
      // inicialización de los datos del objeto...
  }
  //resto de la clase...
}
```

Figura 3: Declaración de la clase y el constructor de un `DataManager` extendido.

Una vez que un cliente realiza una solicitud el sistema invoca al método `generateWorkUnit()` donde se generan las unidades de trabajo, estas unidades no son más que los datos que el cliente necesita para llevar a cabo los cálculos, por ejemplo: puede ser el nombre de un programa que está en el servidor y se necesita en dichos cálculos.

generateWorkUnit

```
public Vector generateWorkUnit() throws Throwable {

   /* Generar unidad de trabajo. Retornar null en caso de que
   no se disponga de trabajo */
   Vector workUnit = new Vector();
   ...
   return workUnit;
}
```

Figura 4: Declaración del método `generateWorkUnit()` de la clase `DataManager`.

El método **`processUnit()`** implementado en la clase `Algorithm` que se ejecuta en el cliente recibe una colección de datos en forma de Vector como unidades de trabajo que necesita para el procesamiento, los resultados de los cálculos se envían al servidor en otro Vector.

processUnit

```
public Vector processUnit( Vector workUnit ) throws Throwable {

   /* procesar la unidad de trabajo enviada por el servidor
      y retornar un vector que represente el conjunto de resultados */
   Vector results = new Vector();
   //...
   return results;
}
```

Figura 5: Declaración del método `processUnit()` de la clase `Algorithm`.

Otro método que se debe implementar en la clase `DataManager` es `processResults()` el cual se llama cada vez que el cliente devuelve un resultado. Recibe dos parámetros, el identificador de la unidad

del trabajo y un Vector que contiene los resultados enviados por los clientes. Si el cómputo distribuido se acaba entonces este método retorna *true*, si no *false*.

processResults

```
public boolean processResults( Long unitID, Vector results ) throws Throwable
{
  // procesar resultado
  if( <problema esta terminado> ){
      return true;
   }
   else{
      return false;
   }
}
```

Figura 6: Declaración del método `processResults()` de la clase `DataManager`.

Otros métodos también deben implementarse por el programador, estos son: `adjustGranularity()`, `getStatus()` y `closeResources()`.

El método `adjustGranularity()` se llama periódicamente por el servidor, recibe como parámetro un porcentaje negativo o positivo que representa el valor de cuánto debe ajustarse el particionado de los datos.

adjustGranularity

```
public void adjustGranularity ( int percent ) throws Throwable {
 //assuming granularity is controlled by a
 //variable called granularity
 granularity = granularity + ((int) ( granularity * percent ) / 100);
}
```

Figura 7: Declaración del método `adjustGranularity()` de la clase `DataManager`.

Se invoca `getStatus()` cada vez que el usuario verifica a través de la interfaz el estado del problema en cuestión. El retorno consiste en una cadena de caracteres que ofrece información útil de dicho estado.

getStatus

```
public String getStatus() throws Throwable {
 String s =//something meaningful about problem progress
 return s;
}
```

Figura 8: Declaración del método `getStatus()` de la clase `DataManager`.

El sistema está representado por una arquitectura MIMD sus características esenciales son:

- ✓ Transparencia.
- ✓ Eficiencia.
- ✓ Flexibilidad.
- ✓ Escalabilidad.
- ✓ Fiabilidad.
- ✓ Grupos de usuarios con privilegios según el rol que desempeñan.
- ✓ Autorización de elementos de cómputos.
- ✓ Conexión desde otro software.

Para llevar a cabo un cómputo distribuido en esta plataforma se requiere de:

- ✓ Máquina Virtual de Java: La plataforma está desarrollada con este lenguaje de programación.

- ✓ Servidor: Encargado de dirigir el proceso.
- ✓ Clientes: Son los encargados de procesar los cálculos.
- ✓ Interface: Actúa de intermediaria entre el cliente y el sistema especificándole a este último los elementos necesarios para realizar el cómputo distribuido.

2.4 Invocación de Métodos Remotos.

Es una tecnología de Java que permite invocar objetos remotos (cuando son definidos mediante interfaces). Las aplicaciones que implementan esta técnica incluyen dos programas: un programa que se ejecuta en el servidor y otro en el cliente. En el programa-servidor se definen un número de objetos remotos que se quieran compartir en la red y se incluyen en el registro de Remote Method Invocation (RMI), este es una aplicación llamada *Rmiregistry* que corre como un proceso separado y que permite: registrar y buscar objetos remotos. Luego, el programa cliente usa este registro para obtener la referencia de los que desee utilizar, para esto usa el nombre (objeto), y el servidor se encarga de buscarlo en el registro, entonces ya está en condiciones de usar los métodos que desee del objeto como si estuviera invocándolos localmente. RMI solo proporciona comunicación remota entre programas escritos en Java. Si se quiere comunicar con otras tecnologías debe usarse CORBA o SOAP. Esta tecnología es sumamente importante para el desarrollo de sistemas de gran potencia, pues hace posible distribuir los recursos y procesarlos en varias máquinas.

Teniendo en cuenta principalmente: las posibilidades y limitaciones que brindan las herramientas y metodologías analizadas a la solución distribuida que requiere el problema, el tamaño de este, así como las condiciones de nuestro país y la tendencia al software libre, se decidió para el desarrollo del problema:

- ✓ UML como lenguaje de modelado pues representa una notación estándar que facilita el desarrollo de los sistemas de software.

- ✓ La metodología BUP, ya que se adapta a condiciones específicas de proyectos ágiles y pequeños como el nuestro, permitiendo entregar un software con calidad.
- ✓ La herramienta de modelado: Visual Paradigm.
- ✓ Lenguaje de Programación Java pues es el lenguaje de programación que soporta la plataforma JDCS.
- ✓ Como IDE se escogió Eclipse.
- ✓ Plataforma JDSC como Sistema de Cómputo Distribuido, la cual ha sido montada en la Universidad y en la actualidad representa una alternativa asequible en costes y recursos.

2.5 Modelo Distribuido.

Modelo Cliente/Servidor

- ✓ ***Desde un punto de vista conceptua***l: Es un modelo para construir sistemas de información, que se sustenta en la idea de repartir el tratamiento de la información y los datos por todo el sistema informático, permitiendo mejorar el rendimiento del sistema global de información.

- ✓ ***En términos de arquitectura:*** Los distintos aspectos que caracterizan a una aplicación (proceso, almacenamiento, control y operaciones de entrada y salida de datos) en el sentido más amplio, se sitúan en más de un computador, los cuales se encuentran interconectados mediante una red de comunicaciones.

- ✓ ***IBM define al modelo Cliente/Servidor:*** Es la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de

trabajo inteligentes o "clientes", resultan en un trabajo realizado por otros computadores llamados servidores.

Arquitectura Cliente Servidor

Esta arquitectura consiste básicamente en que un programa -el Cliente informático- realiza peticiones a otro programa -el servidor- que le da respuesta. Esta idea resulta muy ventajosa si se aplica a un sistema operativo multiusuario distribuido a través de una red de computadoras. En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico.

Ventajas del Modelo Cliente Servidor

- ✓ Existencia de plataformas de hardware cada vez más baratas. Permite utilizar máquinas más baratas que las requeridas por una solución centralizada, basada en sistemas grandes. Además, se pueden utilizar componentes, tanto de hardware como de software, de varios fabricantes, lo cual contribuye a la reducción de costos y favorece la flexibilidad en la implantación y actualización de soluciones.

- ✓ Facilita la integración entre sistemas diferentes y comparte información permitiendo, por ejemplo que las máquinas ya existentes puedan utilizarse pero empleando interfaces más amigables al usuario. De esta manera, podemos integrar PCs con sistemas medianos y grandes, sin necesidad de que todos tengan que utilizar el mismo sistema operacional.
- ✓ Los sistemas tienen mayor interacción con el usuario producto al uso de interfaces gráficas interactivas, pues que no siempre es necesario transmitir información gráfica por la red, esta puede residir en el cliente, impidiendo aprovechar mejor el ancho de banda de la red.
- ✓ La estructura inherentemente modular facilita además la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional, favoreciendo así la escalabilidad de las soluciones.
- ✓ Permite la integración de la información relevante a nivel global.

Desventajas del Modelo Cliente Servidor

Se cuenta con muy escasas herramientas para la administración y ajuste del desempeño de los sistemas.

- ✓ Es importante que los clientes y los servidores utilicen el mismo mecanismo lo cual implica que se deben tener mecanismos generales que existan en diferentes plataformas.
- ✓ Hay que tener estrategias pare el manejo de errores y para mantener la consistencia de los datos. Se deben hacer verificaciones en el cliente y en el servidor. También se puede recurrir a otras técnicas como la encriptación.
- ✓ Un aspecto importante es el de cómo distribuir los datos en la red. En el caso de una organización, por ejemplo, éste puede ser hecho por departamentos, geográficamente, o de otras maneras. Hay que tener en cuenta que en algunos casos, por razones de confiabilidad o eficiencia, se pueden tener datos replicados, y que puede haber actualizaciones simultáneas. [16]

2.6 Conclusiones

Teniendo en cuenta principalmente las posibilidades y limitaciones que brindan las herramientas y metodologías analizadas a la solución distribuida que requiere el problema de la investigación, el tamaño de este, así como las condiciones de nuestro país y la tendencia al software libre, se decidió utilizar para el desarrollo:

- ✓ La metodología BUP, pues se adapta a condiciones específicas de proyectos ágiles y pequeños, permitiendo entregar un software con calidad.
- ✓ Plataforma JDSC como Sistema de Cómputo Distribuido, la cual ha sido montada en la Universidad y en la actualidad representa una alternativa asequible en costes y recursos.
- ✓ UML como lenguaje de modelado.
- ✓ Visual Paradigm como herramienta case.
- ✓ Lenguaje de Programación Java.
- ✓ Como IDE se escogió Eclipse.

CAPÍTULO 3: DESARROLLO Y RESULTADOS

En el presente capítulo se presentan los elementos más importantes de la ingeniería del sistema, se responden preguntas tales como ¿qué debe hacer el sistema y cómo lo debe hacer? en término de requisitos funcionales (***RF***) y no funcionales, se describen los casos de usos correspondientes a los ***RF*** para un mejor entendimiento, así como la solución propuesta. Se presenta el modelo de clases del diseño y de interacción que permitirán modelar dicha solución, el modelo de componentes y de despliegue. Finalmente se muestran algunos resultados experimentales que demuestran una reducción en el tiempo de respuesta de los cálculos de la metodología MMH utilizando el sistema desarrollado.

3.1 Solución propuesta:

A pesar que en varios lugares del mundo se han desarrollado soluciones que reducen el tiempo de espera de los resultados del MOPAC, haciendo uso de Supercomputadoras y de los Sistemas distribuidos, es necesario brindar una solución de acuerdo a las características de la organización en la que se quiere implantar la solución, para una institución puede ser mejor alternativa el uso de Supercomputadoras o **SP**, pero para otras pueden ser los **SD**.

Se debe tener presente también las características del MOPAC, su código contiene más de 30.000 líneas escritas por varios autores, y analizarlo no es una tarea fácil, en consecuencia, el uso de los SP puede ser una solución compleja. En particular, lo que se desea lograr con el presente trabajo, no es solo distribuir el MOPAC por provocar el cómputo elevado de la Metodología de Múltiples Mínimos, sino lograr un sistema que integre tanto la distribución de sus cálculos como los demás pasos de la metodología para reducir el tiempo de espera de los resultados del proceso completo.

Teniendo presente que uno de los retos de la Universidad de Ciencias Informáticas es integrar los más de 6000 ordenadores con los que cuenta en una gran red para computar datos, se debe pensar en una infraestructura que soporte arquitecturas heterogéneas, elementos ubicados en lugares distantes y que se

adapte a un entorno cambiante; la plataforma “Sistema Java de Cómputo Distribuido” representa en estos momentos para la Universidad y el país una infraestructura de este tipo.

Como solución a la problemática de la investigación y teniendo presente los elementos antes mencionados se decidió implementar un sistema distribuido de la metodología MMH sobre la plataforma JDCS.

3.2 Ingeniería del sistema.

Se hace necesario para la comprensión de un sistema informático la realización de una ingeniería del software. Unas de los elementos más importante en el desarrollo de la ingeniería es la especificación de los requerimientos del sistema, donde el propósito primordial de este flujo de trabajo de requisitos es el de guiar el desarrollo del sistema [17].

3.2.1 Requisitos Funcionales:

Los requisitos funcionales son los requisitos que debe cumplir el sistema para cubrir las necesidades del cliente. En particular, el sistema desarrollado debe ser capaz de:

R1. Gestionar Cálculos.

R.1.1. Realizar cálculos químicos teóricos.

R.1.2. Buscar problemas en ejecución.

1.2.1. Interrumpir la ejecución.

1.2.2. Mostrar estado de la ejecución.

R.1.3. Buscar resultados obtenidos.

1.3.1. Descargar solución.

1.3.2. Eliminar solución.

3.2.2 Requerimientos no Funcionales.

Los requisitos no funcionales son los requisitos o características con la que debe cumplir el sistema para que este sea confiable, usable y seguro.

- ✓ ***Usabilidad:***

El sistema lo puede utilizar cualquier individuo con conocimientos básicos de computación. No se requiere basta experiencia haciendo uso de él para explotar todas sus funcionalidades.

- ✓ ***Rendimiento:***

El sistema está concebido para lograr una mayor velocidad en los cálculos que se realizan en la metodología MMH.

- ✓ ***Portabilidad:***

Los clientes que procesan las unidades del MOPAC podrán contar con sistema operativo tanto Windows como Linux.

- ✓ ***Soporte:***

El sistema deberá ser flexible a la incorporación de nuevas funcionalidades y debe proporcionar su mejoramiento.

- ✓ ***Software:***

Para el correcto funcionamiento del sistema se requiere la instalación de una máquina virtual de Java y se necesita que en los clientes exista Sistema Operativo Linux o Windows.

- ✓ ***Hardware:***

Se requiere que las computadoras sean: Procesador Pentium 3 o superior, con 256 MB de RAM y 50 MB de capacidad del disco duro, con Sistema Operativo Windows o Linux y que tenga conexión a una red local.

- ✓ ***Seguridad:***

El usuario debe identificarse antes de acceder a cualquier acción del sistema. Solo podrán utilizar el sistema aquellos usuarios a los que se les asignaron determinados permisos en el servidor. El administrador de servidor que es el encargado de asignar estos permisos.

3.2. 3 Actores del Sistema a Automatizar.

Un actor del sistema es una entidad externa representada por un ser humano, un software o una máquina que interactúa con este. El autor, generalmente le transmite al sistema algunos eventos y en otro caso simplemente obtiene algo de él.

Actores	Justificación
Especialista	Representa al usuario que interactúa con el sistema para realizar los cálculos químico-teóricos.
wgranada	Representa el software que utiliza la metodología MMH en su primer paso para generar las n celdas de configuraciones diferentes.
wmopac	Representa el software de optimización de energía (para SO Windows) que utiliza la metodología MMH en el segundo de sus pasos.
lmopac	Representa el software de optimización de energía (para SO Linux) que utiliza la metodología MMH en el segundo de sus pasos.

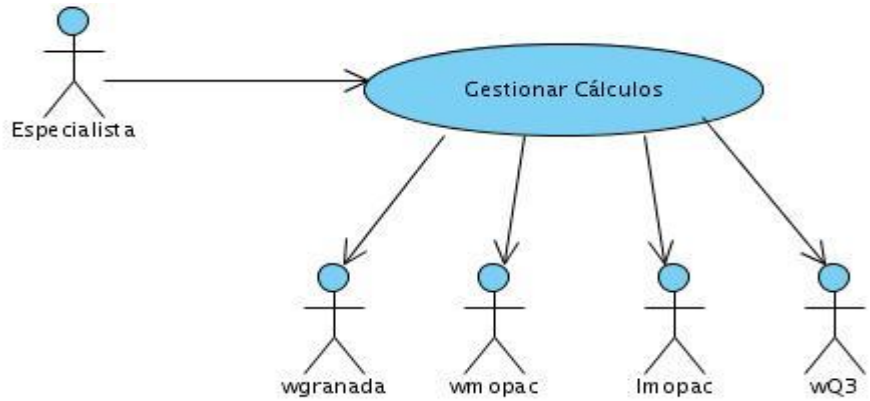
wQ3	Representa el software que utiliza la metodología MMH en el tercer paso.

3.2.4 Casos de Usos del Sistema

Los casos de uso son descripciones de las funcionalidades con las que contará el sistema independiente de la implementación.

Cod	Nombre del caso de uso	Justificación
1	Gestionar cálculos.	Constituye el objetivo fundamental del sistema e incluye todos los cálculos de la Metodología MMH.

3.2.5 Diagrama de Casos de Uso del Sistema



.

Figura 9: Diagrama de Casos de Uso del Sistema.

3.2.6 Descripción de los Casos de Uso del Sistema.

Caso de Uso:	Gestionar cálculos.
Actores:	Especialista [Inicia], wgranada, wmopac, lmopac, wQ3.
Propósito:	Realizar los cálculos de la Metodología MMH y ofrecer al especialista diferentes operaciones sobre las ejecuciones y soluciones de dichos cálculos.
Resumen:	El caso de uso inicia cuando el especialista desea realizar los cálculos químicos-teóricos o efectuar alguna operación sobre las ejecuciones y soluciones que se encuentran en el servidor. El sistema muestra una interfaz con las opciones: Autenticarse. Realizar cálculos químicos-teóricos. Buscar problemas en ejecución. Buscar soluciones obtenidas. El sistema muestra habilitada solo la opción: Autenticarse. Si la conexión es válida, se habilitan el resto de las operaciones. Posteriormente si el usuario escoge la opción: Ejecutar un problema, el sistema muestra una interfaz que posibilita que se introduzcan los datos para la corrida así como la selección de los ficheros necesarios. Al escoger la opción: Buscar problemas en ejecución, el sistema muestra otra interfaz con algunos datos de todos los problemas que se encuentran ejecutando en el servidor y facilita que el usuario termine la ejecución que desee. Si el usuario escoge la opción: Buscar soluciones obtenidas, el sistema muestra una interfaz con algunos datos de las soluciones obtenidas y permite al especialista descargar una solución para su ordenador o

	eliminar cualquiera de estas. El caso de uso termina cuando el especialista obtiene los resultados de los cálculos.
Referencias:	R1.
Precondiciones:	PC cliente con SO Windows o Linux, usuario con permisos en la plataforma para realizar las diferentes operaciones.
Poscondiciones:	Devolver los resultados a los especialistas.

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El usuario se autentica en el sistema. 3. Si el usuario selecciona la opción: Realizar cálculos químicos teóricos: (Ir a la sección 1). Si escoge la opción: Buscar problemas en ejecución (Ir a la sección 2), Si escoge la opción Buscar resultados obtenidos (Ir a la sección 3).	2. Si es un usuario con permiso para acceder al sistema, este permitirá realizar las operaciones: Realizar cálculos químicos teóricos, Buscar problemas en ejecución, Buscar resultados obtenidos.

Flujos Alternos	
Acción del Actor	Respuesta del Sistema
	2.1. El sistema muestra Mensaje de error y no permite realizar ninguna operación.

Sección 1	
Acción del Actor	Respuesta del Sistema
2. El usuario introduce los datos y selecciona los ficheros.	1. El sistema muestra una interfaz para que el usuario introduzca los datos y seleccione los ficheros necesarios para la corrida.

5. Ir a la actividad 2 del Flujo normal de eventos.	3. El sistema conforma con los datos introducidos algunos ficheros necesarios para la ejecución. 4. El sistema inicia el cálculo químico teórico para lo cual envía al servidor los ficheros conformados y los seleccionados por el usuario.
Flujos Alternos	
Acción del Actor	Respuesta del Sistema
Sección 2	
Acción del Actor	Respuesta del Sistema
2. El usuario escoge el problema que desea interrumpir y presiona el botón: Terminar. 4. Ir a la actividad 2 del Flujo normal de eventos.	1. El sistema muestra una interfaz con algunos datos de los problemas que se muestran en ejecución en el servidor. 3. El sistema interrumpe la ejecución del problema seleccionado y actualiza las ejecuciones.
Flujos Alternos	
Acción del Actor	Respuesta del Sistema
Sección 3	
Acción del Actor	Respuesta del Sistema
2. El usuario escoge una solución. 3. Si el usuario desea descargar la solución	1. El sistema muestra una interfaz con las soluciones que se encuentran en el servidor.

presiona el botón: Descargar. 5. Si el usuario desea eliminar la solución presiona el botón: Eliminar.	4. El sistema muestra un explorador para que el usuario seleccione el directorio donde desea descargar la solución. 6. El sistema elimina del servidor la solución seleccionada y actualiza las soluciones.
Flujos Alternos	
Acción del Actor	Respuesta del Sistema
Prioridad	Crítico

3.2.7 Patrón de diseño.

Un patrón de diseño es:

- ✓ Una solución estándar para un problema común de programación.
- ✓ Una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios.
- ✓ Un proyecto o estructura de implementación que logra una finalidad determinada.
- ✓ Un lenguaje de programación de alto nivel.
- ✓ Una manera más práctica de describir ciertos aspectos de la organización de un programa.
- ✓ Conexiones entre componentes de programas.
- ✓ La forma de un diagrama de objeto o de un modelo de objeto. [18]

3.2.7.1 Master-Slave(Maestro-Esclavo).

"El Maestro-Esclavo es un patrón de diseño que apoya la tolerancia de fallas, computación paralela y precisión computacional. Un componente maestro distribuye el trabajo a componentes esclavos idénticos y calcula un resultado final.

La idea del patrón Maestro-Esclavo es introducir un ejemplo de coordinación entre los clientes del servicio y la tramitación de las distintas tareas idénticas. El Maestro divide el trabajo en sub-tareas y el componente esclavo calcula un resultado final. Este principio general se encuentra en diferentes áreas de aplicación:
Tolerancia a fallos: La ejecución de un servicio se delega a repetirse varias implementaciones. La falta de servicio de ejecuciones puede ser detectada y tratada.

Computación paralela: Una tarea compleja se divide en un número determinado de sub-tareas que se ejecutan en paralelo. El resultado final se construye con la ayuda de los resultados procedentes de la transformación de estas sub-tareas.

Precisión Computacional. La ejecución de un servicio se delega en diversas implementaciones.
Resultados inexactos pueden detectarse y manipularse, se les ofrece a todos los esclavos una interfaz común y los servicios generales se comunican sólo con el maestro. [19]

3.2.7.2 Patrón GRASP.

Los patrones GRASP (Patrones de Software para la Asignación General de Responsabilidades) se encargan de definir normas o principios fundamentales en el diseño orientado a objetos y las responsabilidades de estos de acuerdo a su comportamiento. Estos patrones ayudan a la compresión de las soluciones implementadas.

Dentro de los patrones GRASP, por su gran utilidad se destacan (Experto, Creador, Alta cohesión, Bajo acoplamiento y Controlador).

- ✓ **Experto (Expert)**

Consiste en asignar una responsabilidad a una clase que tiene la información necesaria para la realización de la asignación.

Beneficios:

Se mantiene el encapsulamiento de la información y el bajo acoplamiento.

- ✓ **Creador (Creator)**

Consiste en asignarle a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos:

- B agrega los objetos A.
- B contiene los objetos A.
- B registra las instancias de los objetos A.
- B utiliza específicamente los objetos A.
- B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A).
- B es un creador de los objetos A.
- Si existe más de una opción, prefiera la clase B que agregue o contenga la clase A.

Problema:

¿Quién debería ser responsable de crear una nueva instancia de alguna clase?

Beneficios:

Favorece el bajo acoplamiento.

- ✓ **Bajo Acoplamiento (Low Coupling)**

Problema:

¿Cómo soportar bajas dependencias, bajo impacto del cambio e incremento de la reutilización?

Solución:

Asignar una responsabilidad de manera que el acoplamiento permanezca bajo.

Beneficios:

Disminuye el impacto de los cambios.

Se facilita el entendimiento y la reutilización.

- ✓ **Alta cohesión (High Cohesion)**

Problema:

¿Cómo mantener la complejidad manejable?

Solución:

Asignar una responsabilidad de manera que la cohesión permanezca alta.

Beneficios:

Se facilita la comprensión y mantenimiento.

Favorece el bajo acoplamiento y la reutilización.

- ✓ **Controlador (Controller)**

Problema:

¿Quién debería ser el responsable de gestionar un evento de entrada al sistema?

Solución:

Asignar una responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase.

Beneficios:

Posibilita la organización y claridad en el diseño.

Favorece el bajo acoplamiento.

3.2.7.3 Patrón CRUD (Creating, Reading, Updating and Deleting).

El patrón CRUD propone identificar un Caso de Uso(CU), llamado “Información CRUD” o “Administrar Información”, que modela todas las operaciones que se pueden realizar sobre una parte de información de cierto tipo (o sea en una misma entidad), tal como crearla, leerla, actualizarla y eliminarla.

Aplicación

Este patrón debe ser usado cuando todos los flujos contribuyen al mismo valor de negocio, son cortos y sencillos.

3.2.8 Modelo de clases del Diseño.

El Modelo de Diseño es un modelo de objeto que describe la realización física de los casos de uso, centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar.

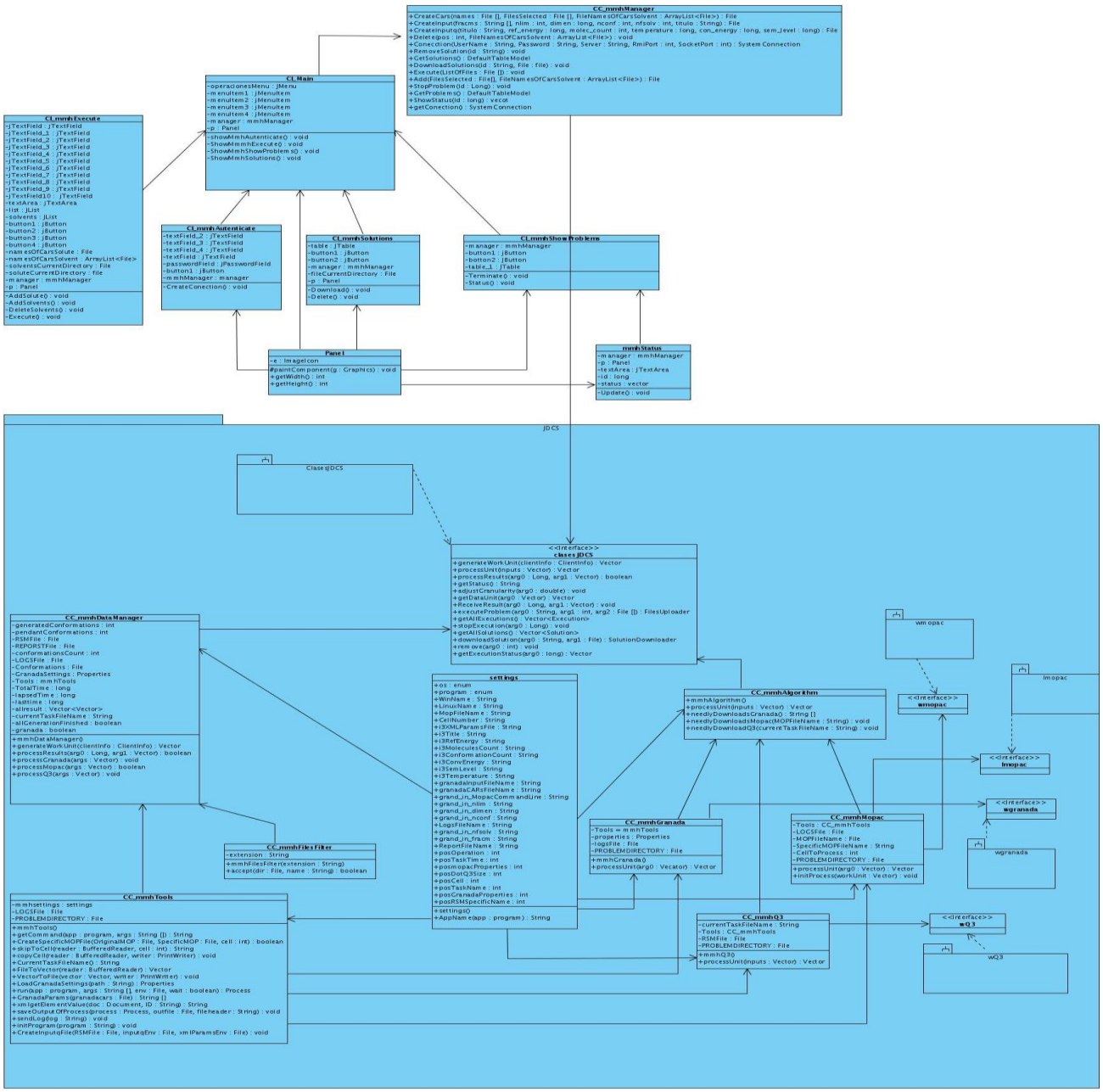


Figura 10: Diagrama de clases del Diseño.

3.2.9 Descripción de las clases más importantes.

Nombre: mmhDataManager	
Tipo: mmhDataManager (Es la clase que genera las unidades de trabajo para los clientes que realizan solicitudes de trabajo, procesa todos los resultados devueltos por dichos clientes y controla el estado del problema).	
Atributos	**Tipo**
conformationsCount	Int
generatedConformations	Int
pendantConformations	Int
RSMFile	File
LOGSFile	File
GranadaSettings	Properties
Tools	mmhTools
TotalTime	Long
lapsedTime	Long
lasttime	Long
allresult	Vector<Vector>
currentTaskFileName	String
allGenerationFinished	Boolean
hacerQ3	Boolean
granada	Boolean
Responsabilidades	

Nombre:	**Descripción:**
generateWorkUnit(ClientInfo clientinfo)	Método que genera unidades de trabajo para los clientes.
getStatus()	Método que muestra el estado del problema en ejecución.
processResults(Long arg0, Vector arg1)	Método que integra los resultados recibidos por los clientes en dependencia del programa que se ejecutó.
processGranada(Vector args)	Método que integra los datos procedentes del cliente que realiza la ejecución del programa Granada.
processMopac(Vector args)	Método que integra los datos procedentes del cliente que realiza la ejecución del programa MOPAC tanto para Linux como Windows.
processQ3(Vector args)	Método que integra los datos procedentes del cliente que realiza la ejecución del programa Q3.

Nombre: mmhAlgorithm	
Tipo: mmhAlgorithm(Es la clase que incluye todas las operaciones que el cliente realizará.)	
Atributos	**Tipo**
Tools	mmhTools
Log	File
mmhSettings	Settings

Responsabilidades	
Nombre:	**Descripción:**
processUnit(Vector inputs)	Método que contiene las operaciones que realizará el cliente en dependencia del programa que le corresponda ejecutar en correspondencia con la metodología MMH.
needlyDownloadsGranada()	Método que descarga para el cliente los ficheros y programas necesarios para ejecutar el programa Granada.
needlyDownloadsWMopac(String MOPFileName)	Método que descarga para el cliente los ficheros y programas necesarios para ejecutar el programa MOPAC para Windows.
needlyDownloadsLMopac(String MOPFileName)	Método que descarga para el cliente los ficheros y programas necesarios para ejecutar el programa MOPAC para Linux.
needlyDownloadQ3(String currentTaskFileName)	Método que descarga para el cliente los ficheros y programas necesarios para ejecutar el programa Q3.

Nombre: mmhGranada	
Tipo: mmhGranada(Es la clase que contiene las operaciones que realizará el cliente al ejecutar el programa Granada.)	
Atributos	**Tipo**
Tools	mmhTools
properties	Properties

PROBLEMDIRECTORY	File
Responsabilidades	
Nombre:	**Descripción:**
processUnit(Vector arg0, String[] params)	Método que contiene las operaciones que realizará el cliente al ejecutar el programa Granada.

Nombre: mmhMopac	
Tipo: mmhMopac(Es la clase que contiene las operaciones que realizarán los clientes al ejecutar el programa MOPAC (Windows o Linux.))	
Atributos	**Tipo**
Tools	mmhTools
MOPFileName	String
SpecificMOPFileName	String
CellToProcess	Int
PROBLEMDIRECTOR	File
Responsabilidades	
Nombre:	**Descripción:**
processUnit(Vector arg0)	Método que contiene las operaciones que realizarán los clientes al ejecutar el programa MOPAC para Windows.
initProcess(Vector workUnit)	Método que conforma el fichero .mop específico con la celda que corresponde procesar en los clientes.

Nombre: mmhQ3	
Tipo: mmhQ3(Es la clase que contiene las operaciones que realizará el cliente al ejecutar el programa Granada.)	
Atributos	**Tipo**
currentTaskFileName	String
Tools	mmhTools
RSMFile	File
PROBLEMDIRECTORY	File
Responsabilidades	
Nombre:	**Descripción:**
processUnit(Vector inputs)	Método que contiene las operaciones que realizará el cliente al ejecutar el programa Q3.

3.2.10 Diagramas de Secuencia.

Los Diagramas de Secuencia son unos de los diagramas más efectivos para modelar la interacción entre los objetos de un sistema de manera secuencial y en forma de mensajes. Son modelados para cada caso de uso y contienen detalles de la implementación, incluyendo clases que se usan en esta. Un diagrama de secuencia muestra los objetos que intervienen en el escenario y los mensajes entre estos.

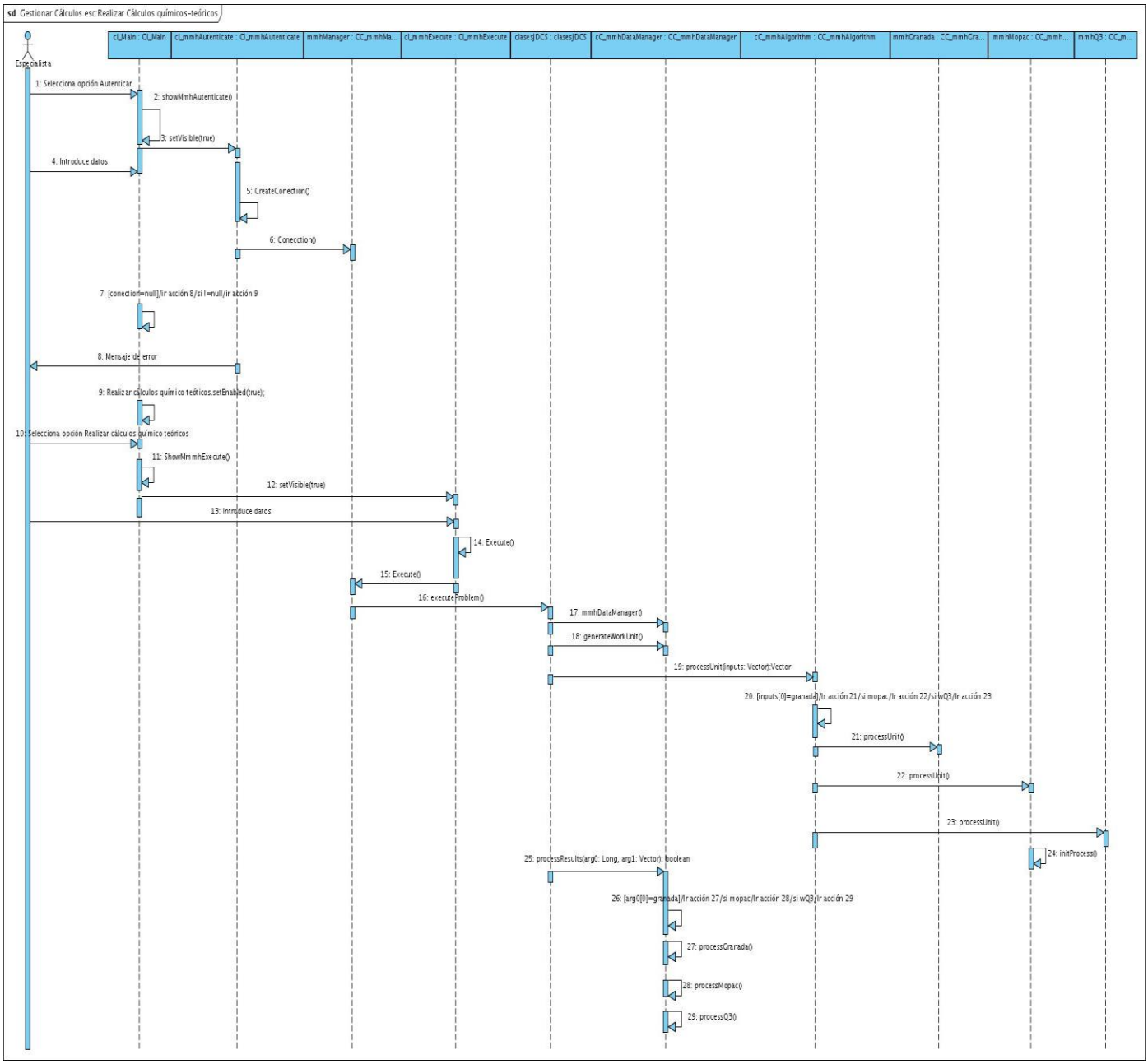


Figura 11: Diagrama de Secuencia: Gestionar Cálculos esc_Realizar cálculos químicos-teóricos.

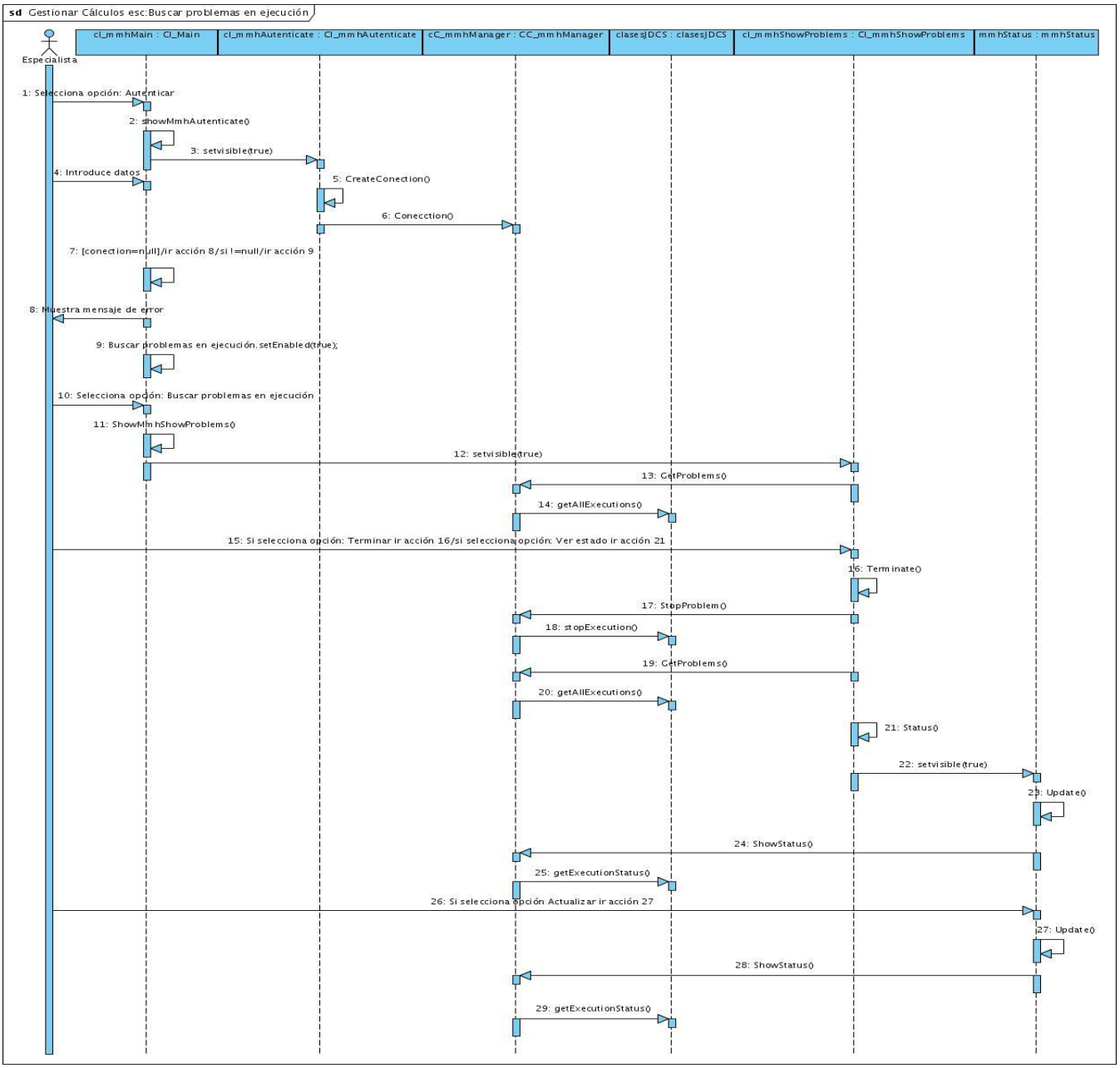


Figura 12: Diagrama de Secuencia: Gestionar Cálculos esc_Buscar problemas en ejecución.

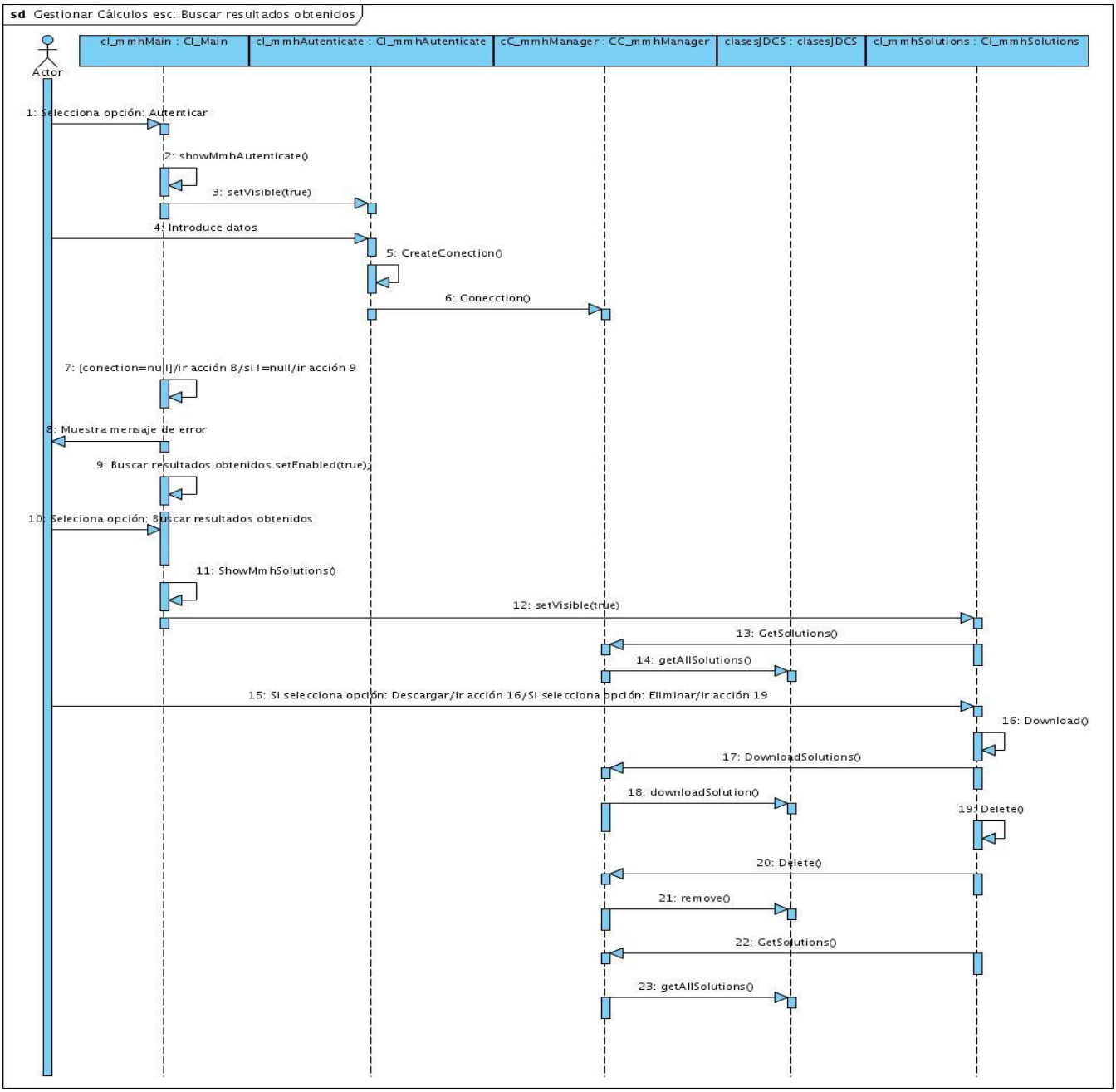


Figura 13: Diagrama de Secuencia: Gestionar Cálculos esc_Buscar resultados obtenidos.

3.2.11 Modelo de despliegue.

El diagrama de despliegue es un modelo de objetos que describe la distribución física del sistema, muestra la distribución en términos de cómo se distribuye la funcionalidad del mismo entre los nodos de cómputo. Los nodos son utilizados para crear la topología del hardware sobre el cual se ejecuta el sistema.

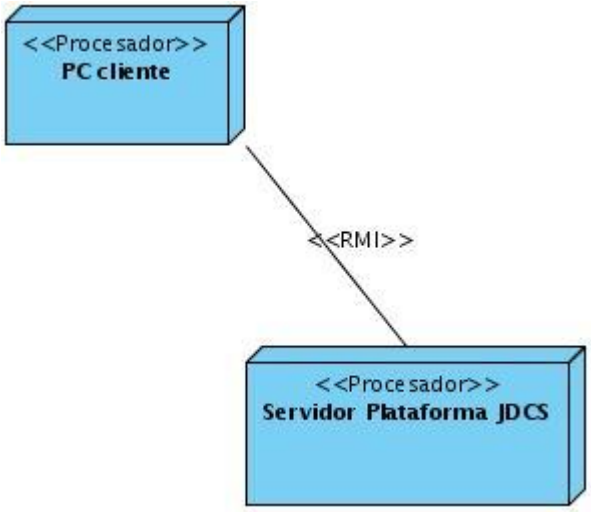


Figura 14: Diagrama de Despliegue.

3.2.12 Diagrama de Componentes.

Los diagramas de componentes expresan las dependencias entre los componentes del sistema los cuales representan, códigos, scripts, ejecutables, entre otros.

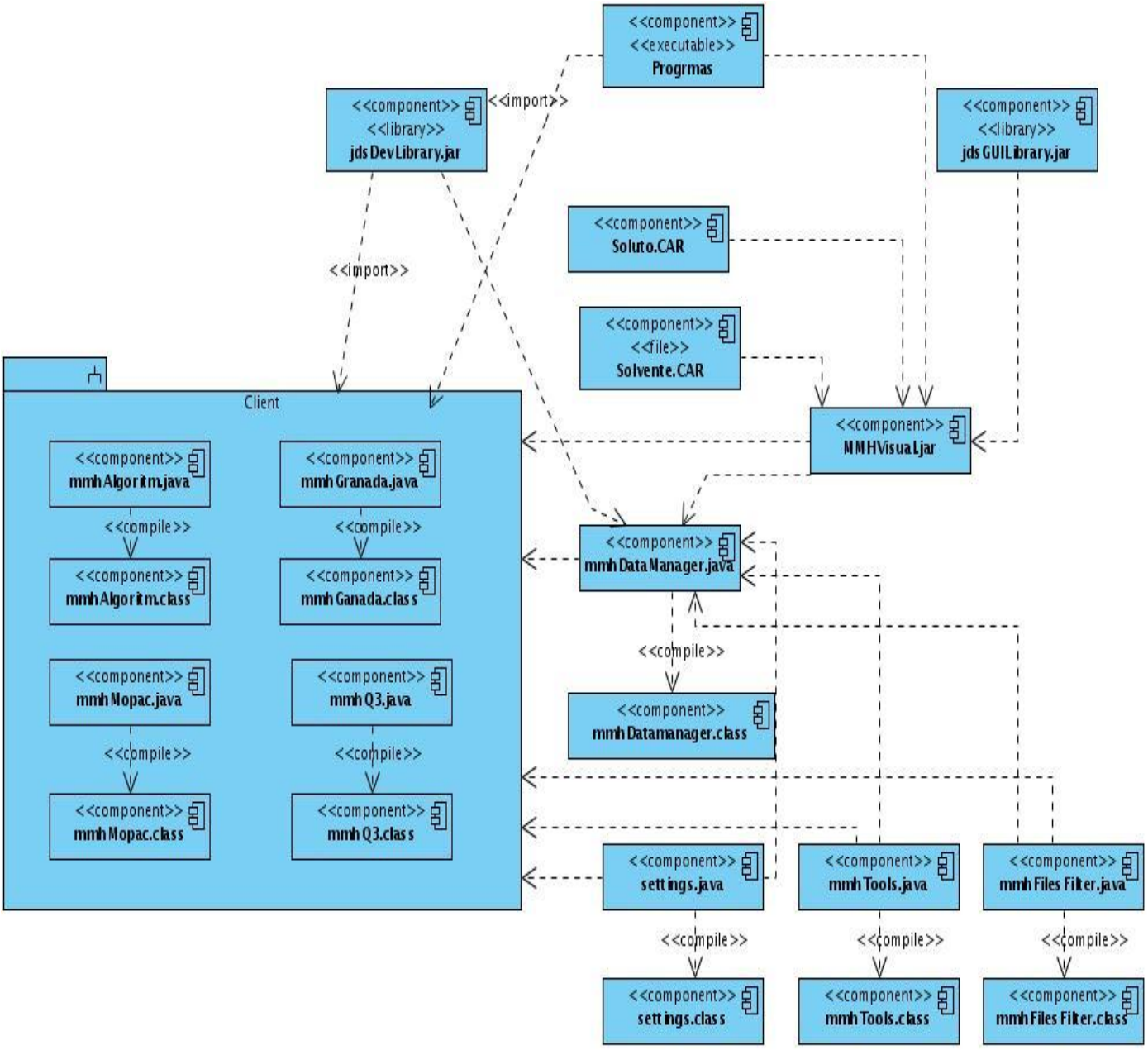


Figura 15: Diagrama de Componentes.

3.2.13 Fragmentos de Código

Siguiendo los pasos de la metodología MMH, la solución desarrollada para realizar los cálculos consiste en:

Ante las solicitudes de los clientes se envían las unidades de trabajo correspondientes a través del método `generateWorkUnit()` de la clase `mmhDataManager`. La primera posición del Vector retornado contiene el nombre del programa que corresponda ejecutar y en el resto de las posiciones otros datos de importancia para la realización de los cálculos.

```
public Vector generateWorkUnit(ClientInfo clientinfo) throws Throwable
{
      if (allGenerationFinished == true) return null;
      if (granada == false)
      {
         // Generando unidades del Granada
        if (clientinfo.getOS().toLowerCase().contains(settings.WinName)==true){
                      Vector vGranada = new Vector();
                      vGranada.add(settings.program.granada);
                      granada = true;
                      return vGranada;
        }
        else
            return null;

      }
      //Mientras no finalice la ejecución Granada
      if (conformationsCount == 0 && generatedConformations == 0)
            return null;
      if (generatedConformations != conformationsCount){
            //Grandana's Unit was Processed
            Vector vector = new Vector();
            Properties property= new Properties();
            vector.add(settings.program.mopac);
            property.setProperty(settings.MopFileName, currentTaskFileName);
            property.setProperty(settings.CellNumber,Integer.toString(++generate
            dConformations));
            vector.add(property);
```

```
            return vector;
      }
      //Generando unidades del Q3
      if(!hacerQ3)
      {
            Tools.sendLog("No se puede ejecutar el Q3");
            return null;
      }
      Vector vector = new Vector();
      vector.add(settings.program.q3);
      vector.add(currentTaskFileName);
      allGenerationFinished = true;
      return vector;
}
```

Estos datos son recibidos en el vector `inputs` del método `processUnit()` de la clase `mmhAlgoritm` del que los clientes leen en la primera posición el nombre del programa, y según esta información procesa la unidad que le corresponde. Si corresponde ejecutar el programa Granada se invoca al método `processUnit()` de la clase `mmhGranada`, este recibe como parámetro el mismo vector `inputs`. Si corresponde la ejecución del MOPAC se invocará al método `processUnit()` de la clase `mmhMopac`.

```
public Vector processUnit(Vector arg0)throws Throwable
{
      try
      {
            initProcess(arg0);    //Crea fichero específico a procesar

            long timeIni = System.currentTimeMillis();
            settings mmhsettings= new settings();
            if(mmhsettings.OperatingSystem == settings.os.windows)
            {
                  String cmd = " cmd /c "+
                  Tools.getCommand(settings.program.mopac, new String[]{
                  SpecificMOPFileName});
                  Process  p = null;
                  try {
```

```
                    p = Runtime.getRuntime().exec(cmd, null,
                    PROBLEMDIRECTORY);
            }
            catch (Throwable e){
                  throw new Throwable("Error "+ e.getMessage());
            }
      }
      else
      {
            Process process = null;
            try
            {
                  Process procs = Runtime.getRuntime().exec("chmod +x -R
                  ./",null,PROBLEMDIRECTORY);
                  process =
            Runtime.getRuntime().exec(PROBLEMDIRECTORY.getAbsolutePath()+F
            ile.separator+settings.MOPACscriptName+"
            PROBLEMDIRECTORY.getAbsolutePath()+File.separator+settings.MOP
            AClName+" "+ PROBLEMDIRECTORY.getAbsolutePath()+
            File.separator+SpecificMOPFileName.substring(0,SpecificMOPFile
            Name.lastIndexOf(".")), null,PROBLEMDIRECTORY );
            }
            catch(Exception e)
            {
                  throw new Throwable("Error "+ e.getMessage());
            }
      }
      long timeEnd = System.currentTimeMillis();

      Vector result = new Vector();

      result.add(settings.program.mopac);   // nombre del Mopac

      result.add(Long.valueOf((timeEnd-timeIni) // Tiempo
      result.add(Long.valueOf(CellToProcess));   // Celda procesada

      String SpecificRSMFileName = SpecificMOPFileName.substring(0,
      SpecificMOPFileName.indexOf(".")   )+".rsm";
            result.add(SpecificRSMFileName);

      mmhFilesFilter filt = new mmhFilesFilter("rsm");
```

```
            while(PROBLEMDIRECTORY.listFiles(filt).length == 0)
                  Thread.sleep(10000);

            Vector aa = Tools.FileToVector(new BufferedReader(new
            FileReader( new File(PROBLEMDIRECTORY,SpecificRSMFileName)))) ;

            result.addAll(aa);
            return result;
      }
      catch(Throwable as){
            throw new Throwable(as.getMessage());
      }
}
```

Si corresponde al programa Q3 se invocará al método `processUnit` de la clase `mmhQ3`. En particular, los clientes que procesaran el MOPAC entre otros elementos importantes, leerán del vector el nombre del fichero .mop que corresponda procesar, para luego descargarlos del servidor y formar un fichero .mop específico (método `initProcess()` de la clase `mmhMopac`) con la celda que corresponda procesar, dato que también es enviado en el vector. Para la ejecución del Q3, un elemento importante es el nombre del fichero .rsm que de igual modo es enviado en el vector y descargado hacia el cliente para la ejecución.

Una ves terminadas las ejecuciones en los clientes estos envían los resultados obtenidos en un vector por medio de la clase `mmhAlgoritm` y otros datos como el tiempo transcurrido, además de enviar en la primera posición el nombre del programa que se ejecutó, pues en dependencia de esto se integrarán los resultados en el servidor. Si los resultados son procedentes del cliente que ejecutó Granada se invocará al método `processGranada()` de la clase `mmhDataManager`, si son procedentes de los clientes que procesan las unidades del MOPAC se invoca al método `processMopac()` de esta misma clase.

```
private boolean processMopac(Vector args) throws Throwable
{
      long actualtime = System.currentTimeMillis()/1000;
      TotalTime += Long.valueOf(args.get(settings.posTaskTime).toString());

      lapsedTime +=  ( actualtime - lasttime );
```

```
      lasttime = actualtime;
      int CellNumber = Integer.parseInt(args.get(settings.posCell).toString());

      PrintWriter confwriter = new PrintWriter(new File(PROBLEMDIRECTORY,
                           args.get(settings.posRSMSpecificName).toString()));

      args.remove(0); // rmv Operation
      args.remove(0); // rmv Time
      args.remove(0); // rmv Cell Number
      args.remove(0); // rmv RSM Name

      Tools.VectorToFile(args,confwriter);
      confwriter.close();
      allresult.setElementAt(args,CellNumber-1);
      pendantConformations--;
      if(pendantConformations!=0)
            return false;

      RSMFile = new File(PROBLEMDIRECTORY,currentTaskFileName+".rsm");
      PrintWriter writer = new PrintWriter(new FileWriter(RSMFile));
      for (int i = 0; i < conformationsCount; i++)
            //Create .RSM file from client results
            if (allresult.get(i)!=null)
                  Tools.VectorToFile(allresult.get(i), writer);
      writer.close();
      Tools.sendLog("Creando fichero .RSM ... OK \n");
      hacerQ3 = true;
      return true;
}
```

Finalmente se integrarán los resultados de la ejecución del Q3 en el método `processQ3()`.

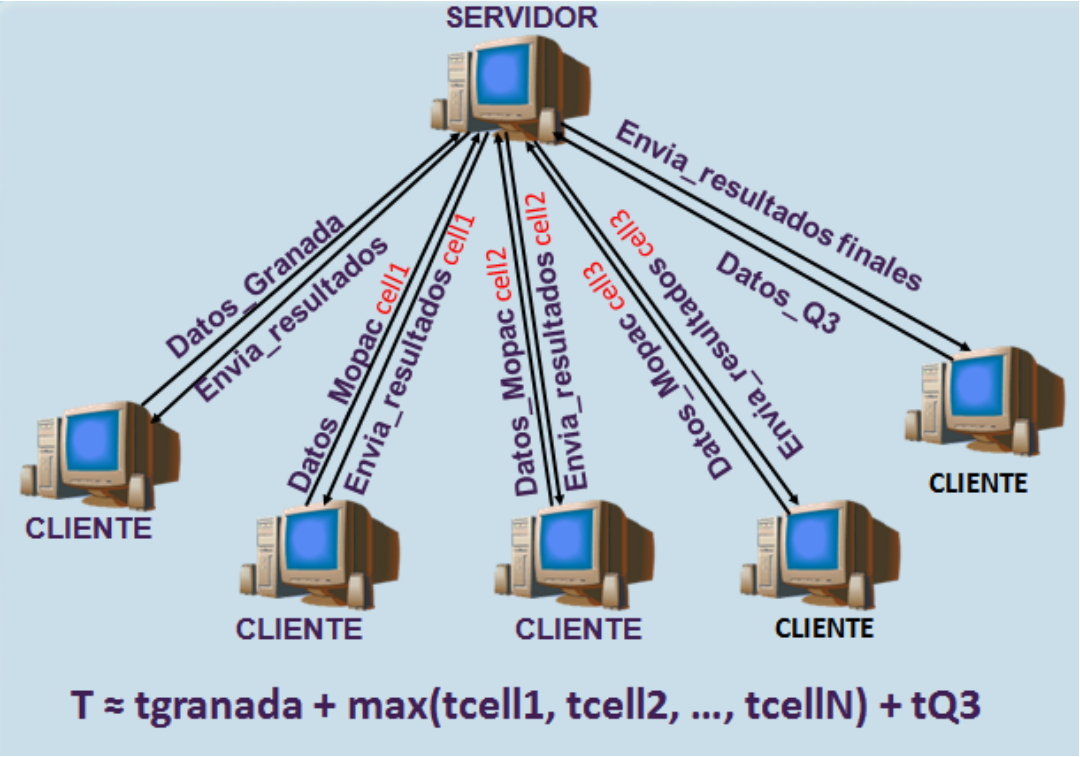


Figura 16: Solución propuesta.

3.3 Resultados experimentales.

Con el objetivo de demostrar la reducción de los tiempos de respuestas de la metodología MMH aplicando el sistema implementado, se realizaron cálculos de forma secuencial y utilizando el sistema propuesto sobre la plataforma JDCS. Teniendo ambos tiempos se pudo comparar y demostrar la reducción de los mismos utilizando el sistema desarrollado.

Los cálculos se realizaron a partir de una molécula de soluto y otra molécula de solvente cuyo fichero **.mop** resultante posterior a la ejecución del programa Granada contenía 50 conformaciones de 44 átomos cada una; (el tiempo de procesamiento de aplicar la metodología MMH con estos datos en un ordenador

fue aproximadamente igual a **154** minutos). Para la realización de estos cálculos se usaron 582 clientes Pentium IV conectados por una red de velocidad igual a 100 Mbps, 553 clientes con sistema operativo Windows de 256 MB de memoria RAM y 22 clientes con sistema operativo Linux de 256 MB, 512 MB y 712 MB de memoria RAM. Finalmente, se hizo un ajuste lineal con un polinomio de grado 4 en función del tiempo y las horas de procesamiento a partir de varias pruebas realizadas en diferentes días y horarios aplicando la metodología MMH a los ficheros soluto y solvente anteriores. El resultado concluyó en la siguiente gráfica:

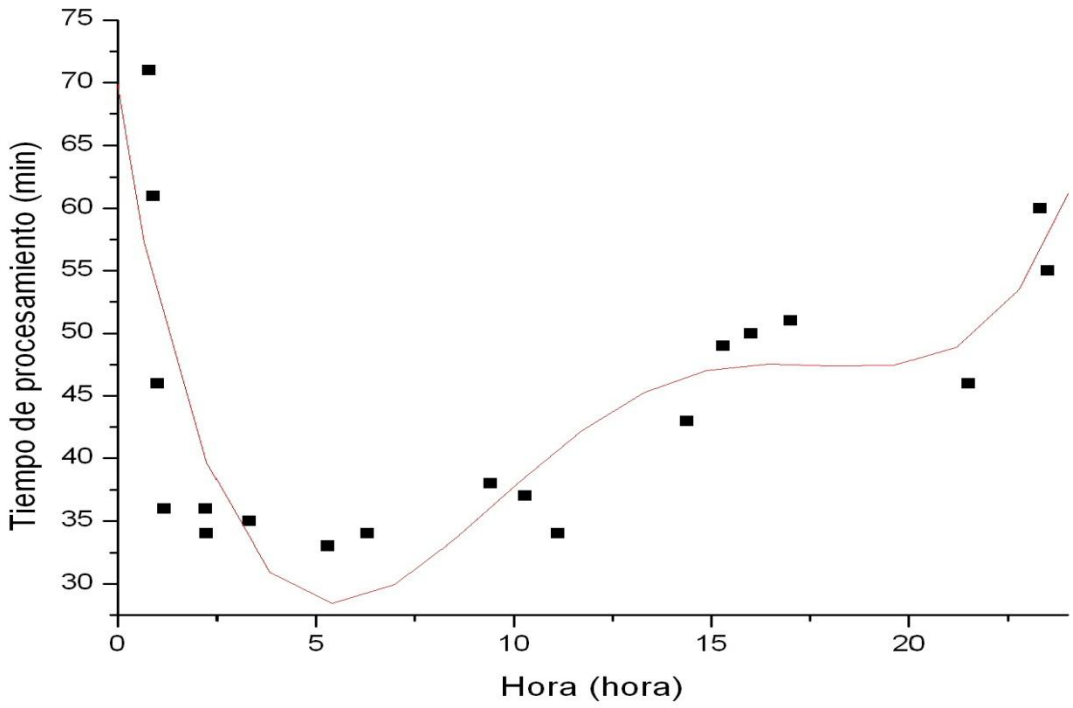


Figura 17: Comportamiento del sistema en distintas horas del día.

Este análisis permitió principalmente estimar el tiempo máximo de procesamiento y las horas más favorecidas para la realización de los cálculos. La ecuación que representa la gráfica anterior resultó ser la siguiente: **Y = A + B1*X + B2*X^2 + B3*X^3 + B4*X^4.**

Parámetros	Valor	Error
A	67.95633	6.75779
B1	-17.89132	5.09581
B2	2.65246	0.93544
B3	-0.14268	0.06105
B4	0.00261	0.00129

Desviación estándar	# Nodos
7.3811	19

Tanto la gráfica de la figura 17 como los coeficientes de la ecuación de la curva de interpolación se obtuvieron usando la herramienta Origin 6.0 (http://www.originlab.com/).

Los datos usados para obtener la gráfica de la figura 17 se muestran en la siguiente tabla:

Tiempo Sistema Implementado	
Hora de comienzo	Tiempo de procesamiento (minutos)
0:09	61
1:00	46

2:23	36
2:25	34
3:30	35
5:30	33
6:30	34
15:30	49
16:00	50
17:00	51
23:05	55
23:30	60

Tabla 1: Tiempo de ejecución distribuido de cada prueba realizada.

En el Anexo 2 se pueden ver las tablas de las pruebas con un nivel de detalle mayor, pues por cada prueba realizada se muestra el tiempo del procesamiento de cada celda o conformación, el número IP del cliente que procesó dicha celda, la hora de inicio y hora de culminación del procesamiento en cada cliente. A partir de los datos de la tabla 1 y del anexo 2 podemos extraer información acerca del rendimiento del sistema distribuido.

La siguiente gráfica muestra la ganancia de velocidad (*speed-up*) [20] obtenida en las distintas pruebas realizadas:

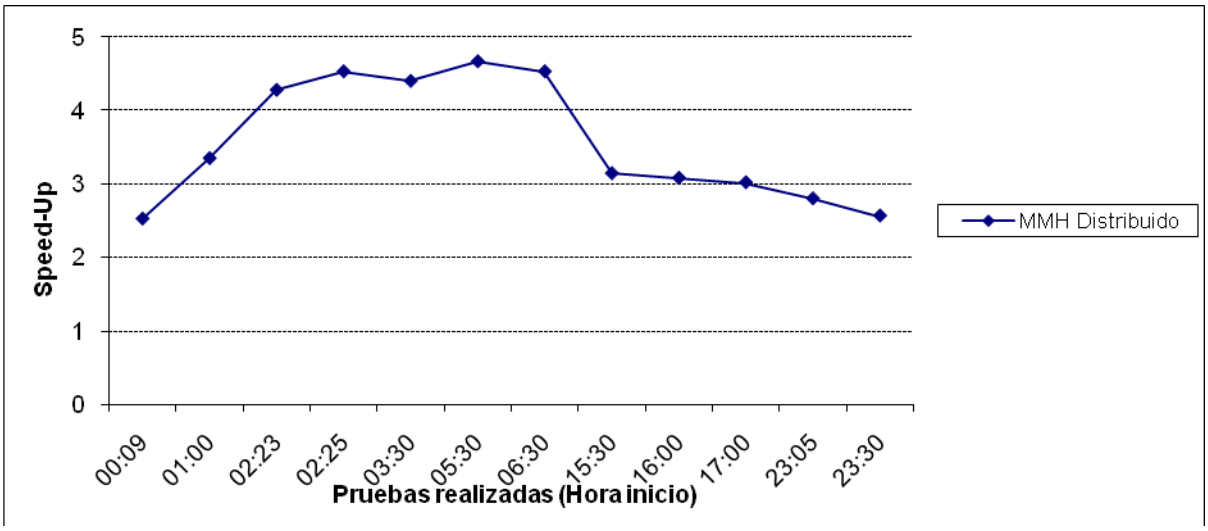


Figura 18: Speed-Up obtenido en las pruebas realizadas.

Como se puede apreciar, el *speed-up* siempre se comportó por encima de 2.5, es decir, el sistema distribuido obtiene una ganancia de velocidad al menos 2.5 veces mayor y en la mayoría de los casos la velocidad del sistema distribuido es 3 veces mayor comparado con la velocidad del sistema ejecutado de forma secuencial. También se puede notar que las horas donde se obtuvo mejor rendimiento (tal y como se mostró también en la figura 17) fueron las horas de la madrugada, donde se supone hay un menor tráfico en la red y las máquinas están menos ocupadas.

La ganancia de velocidad hubiera sido mucho mayor de haber existido un completo solapamiento en el trabajo de todos los clientes, es decir, si todas las celdas o conformaciones se hubieran procesado al mismo tiempo. En todas las pruebas realizadas ocurrió que algunas celdas se comenzaron a procesar mucho después de la culminación del procesamiento al resto de las celdas. La siguiente tabla muestra en 8 de las pruebas realizadas, la cantidad de celdas que se comenzaron a procesar muy tarde respecto al resto.

Prueba (hora inicio - hora fin)	Celdas demoradas en procesar
9:41 - 10:19	2
10:28 - 11:05	3
11:12 - 11:46	4
14:38 - 15:21	2
21:05 - 21:51	12
00:08 - 1:19	10
1:17 - 1:53	4
2:23 - 2:59	7

Tabla 2: Cantidad de celdas que se procesaron mucho después al resto.

Las causas de la demora en el procesamiento de algunas celdas puede estar dada por varias razones, que en el tiempo de inicio de procesamiento solo hicieron peticiones de trabajo algunos clientes y el resto lo hicieron mucho tiempo después. También puede ocurrir que en un primer momento estas unidades de trabajo se enviaron a los clientes y luego de un tiempo de no reportarse al servidor se consideraron expiradas y en consecuencia enviadas nuevamente a otros clientes.

Otra posible razón es que en algunas estaciones de trabajo la prioridad de ejecutar este proceso sea muy baja, o que tenga muchos procesos corriendo y como consecuencia sea más lenta la ejecución. En las pruebas realizadas algunos clientes se reportaron al servidor y no empezaron a procesar las unidades de trabajo hasta después de un tiempo prolongado luego de haber realizado la petición, incluso más de 20 minutos. Se presentan a continuación datos de algunos de los clientes que más demoraron en comenzar a procesar las unidades de trabajo.

IP	Hora-Petición	Hora-Inicio
10.34.6.34	0:07:15	0:39:03
10.34.6.34	9:41:00	9:45:45
10.34.12.32	0:07:42	0:45:38
10.34.14.14	21:04:18	21:45:28
10.35.17.6	1:17:00	1:49:11
10.35.17.6	2:22:24	2:54:00

Este análisis es muy importante pues puede brindar a los administradores de la plataforma JDCS información acerca de aquellos clientes con las características anteriores y rechazar en un futuro sus peticiones de trabajo.

En la siguiente gráfica se muestra la eficiencia del sistema [20], que no es más que la división del *speed-up* entre la cantidad de procesadores que intervinieron en la solución:

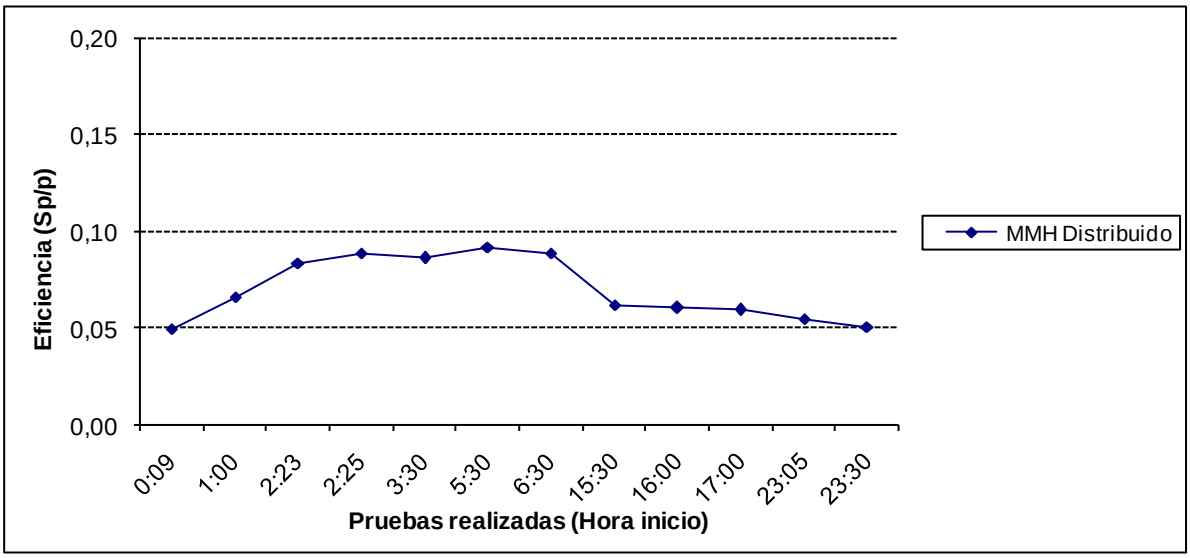


Figura 19: Eficiencia del sistema en las pruebas realizadas.

La eficiencia está aún muy lejos de llegar al valor de 1 (que significaría un 100% de eficiencia del sistema), y las causas de esto son las mismas mencionadas anteriormente.

3.4 Conclusiones.

En este capítulo se presentaron artefactos importantes de la ingeniería del sistema desarrollado, los cuales permitirán entender con mayor claridad el mismo. A pesar de que el análisis realizado no fue lo suficientemente profundo por la cantidad de datos que se analizaron y aunque no se alcanzó una mayor eficiencia, se consideraron las posibles consecuencias, se estimó la ganancia de velocidad utilizando el sistema desarrollado la cual resultó ser como mínimo 2.5 veces mayor que la ejecución del sistema de forma secuencial. El análisis permitió además estimar elementos importantes como horarios más favorables para realizar los cálculos. Se demostró de esta forma que la solución propuesta representa una buena alternativa en coste de tiempo y esfuerzos.

CONCLUSIONES GENERALES

En el transcurso de la realización de esta investigación se fueron cumpliendo con todos los objetivos planteados:

- ✓ Se realizó un estudio detallado del funcionamiento de la metodología MMH y de algunos sistemas distribuidos que facilitaron el desarrollo de la aplicación.
- ✓ Se valoraron las herramientas Visual Paradigm, Eclipse, el lenguaje de programación Java, la plataforma JDCS, el lenguaje de modelado UML y la metodología OpenUp que se utilizaron para el desarrollo de la aplicación.
- ✓ Se identificaron las funcionalidades que debe cumplir el sistema mediante la captura de los requisitos funcionales y no funcionales.
- ✓ Se diseñó el sistema que facilitó la planificación de la etapa de la implementación.
- ✓ Se implementó un sistema que reduce considerablemente el tiempo de respuesta de la metodología MMH. En pruebas realizadas su velocidad demostró ser 2.5 veces superior que las ejecuciones normales en un único procesador y aunque no se alcanzó la eficiencia deseada se estimaron las posibles consecuencias que de no existir hubiesen permitido alcanzar resultados más satisfactorios.

RECOMENDACIONES

Se recomienda:

- ✓ Realizar diferentes pruebas durante un tiempo para comprobar el correcto funcionamiento de los requisitos solicitados por el cliente.

- ✓ Incorporar a la aplicación el uso de los logs que genera el MOPAC durante sus ejecuciones para recomenzar las tareas en caso de que las unidades de trabajo expiren.

REFERENCIAS

1. *Bioinformática: en busca de los secretos moleculares de la vida.* **Cañedo Andalia, Rubén y Arencibia Jorge, Ricardo.** La Habana : s.n., 2004.

2. *Impacto de la Bioinformática en las ciencias biomédicas.* **Perezleo Solórzano, Ligeya, y otros.** La Habana : Ciencias Médicas, 2003.

3. **S. Tanenbaum, Andrew.** *Distributed Operating Systems.*

4. **Coulouris, George.** *Sistemas Distribuidos.* Madrid : s.n., 2001.

5. **Santoro, Nicola.** *DESIGN AND ANALYSIS OF DISTRIBUTED ALGORITHMS.* Ottawa, Canada : s.n., 2007.

6. **Fernández Casaní, Álvaro.** *ARQUITECTURAS GRID orientadas a la gestión de recursos.* 2004.

7. A Java based heterogeneous distributed computing system . [En línea] [Citado el: 15 de diciembre de 2007.] http://distributed.cs.nuim.ie/.

8. **Hui Lin, Tseng, Haupt, Tomasz y C. Fox, Geoffrey.** *Parallelizing MOPAC on distributed computing.* Syracuse, NY : s.n.

9. **Camejo Isaac, Andrés Yanier y Turro Rodríguez, Adnier.** *Módulo para el cálculo distribuido de mecánica molecular y mecánica cuántica.* La Habana : s.n., 2007.

10. LABORATORIO DE QUÍMICA COMPUTACIONAL Y TEÓRICA. [En línea] [Citado el: 8 de noviembre de 2007.] http://karin.fq.uh.cu/mmh/.

11. **P. Stewart, James J.** *MOPAC 93 MANUAL.* 1994.

12. **E.M, Virgós.** MATEMATICALIA . [En línea] [Citado el: 20 de enero de 2008.] http://www.matematicalia.net/index.php?option=com_content&task=view&id=293&Itemid=191..

13. **Keane B.Sc., Thomas.** *A General-Purpose Heterogeneous.* 2004.

14. Condor High Throughput Computing. [En línea] [Citado el: 25 de enero de 2008.] http://www.cs.wisc.edu/condor/.

15. Centro Informatico Cientifico de Andalucia. *Centro Informatico Cientifico de Andalucia.* [En línea] [Citado el: 20 de 03 de 2008.] www:cica.es/condoe.html..

16. **Informatica, Instituto Nacinal de Estadisticas e.** Tecnologia cliente servidor. *Tecnologia cliente servidor.* [En línea] [Citado el: 28 de 02 de 2008.]

17. **Larman, C**. (1999). UML y Patrones. Introducción al análisis y diseño orientado a objetos.
18. (octubre de 2001). Recuperado el 6 de mayo de 2008, de
http: //mit.ocw.universia.net /6.170/6.170 /f01 /pdf/lecture-12.pdf

19. *Vico.org.* (s.f.). Recuperado el 7 de mayo de 2008, de Vico.org:
http://www.vico. org/pages /PatronsDisseny/attern%20Master%20Slave/index.html

20. **Kumar, Vipin, et al.** *Introduction to Parallel Computing, Second Edition.* s.l. : Addison Wesley, 2003.

BIBLIOGRAFÍA

1. *Bioinformática: en busca de los secretos moleculares de la vida.* **Cañedo Andalia, Rubén y Arencibia Jorge, Ricardo.** La Habana : s.n., 2004.

2. *Impacto de la Bioinformática en las ciencias biomédicas.* **Perezleo Solórzano, Ligeya, y otros.** La Habana : Ciencias Médicas, 2003.

3. **S. Tanenbaum, Andrew.** *Distributed Operating Systems.*

4. **Coulouris, George.** *Sistemas Distribuidos.* Madrid : s.n., 2001.

5. **Santoro, Nicola.** *DESIGN AND ANALYSIS OF DISTRIBUTED ALGORITHMS.* Ottawa, Canada : s.n., 2007.

6. **Fernández Casaní, Álvaro.** *ARQUITECTURAS GRID orientadas a la gestión de recursos.* 2004.

7. A Java based heterogeneous distributed computing system . [En línea] [Citado el: 15 de diciembre de 2007.]

8. **Hui Lin, Tseng, Haupt, Tomasz y C. Fox, Geoffrey.** *Parallelizing MOPAC on distributed computing.* Syracuse, NY : s.n.

9. **Camejo Isaac, Andrés Yanier y Turro Rodríguez, Adnier.** *Módulo para el cálculo distribuido de mecánica molecular y mecánica cuántica.* La Habana : s.n., 2007.

10. LABORATORIO DE QUÍMICA COMPUTACIONAL Y TEÓRICA. [En línea] [Citado el: 8 de noviembre de 2007.]

11. **P. Stewart, James J.** *MOPAC 93 MANUAL.* 1994.

12. **E.M, Virgós.** MATEMATICALIA . [En línea] [Citado el: 20 de enero de 2008.]

132**.** *A General-Purpose Heterogeneous.* 2004.

14. Condor High Throughput Computing. [En línea] [Citado el: 25 de enero de 2008.]

15. Centro Informatico Cientifico de Andalucia. *Centro Informatico Cientifico de Andalucia.* [En línea] [Citado el: 20 de 03 de 2008.] www:cica.es/condoe.html..

16. **Informatica, Instituto Nacinal de Estadisticas e.** Tecnologia cliente servidor. *Tecnologia cliente servidor.* [En línea] [Citado el: 28 de 02 de 2008.]

17. **Larman, C**. (1999). UML y Patrones. Introducción al análisis y diseño orientado a objetos.

18.(octubre de 2001). Recuperado el 6 de mayo de 2008, de http: //mit.ocw.universia.net /6.170/6.170 /f01 /pdf/lecture-12.pdf

19. *Vico.org.*(s.f.).Recuperado el 7 de mayo de 2008, de Vico.org: http://www.vico. org/pages /PatronsDisseny/attern%20Master%20Slave/index.html

20. **Kumar, Vipin, et al.** *Introduction to Parallel Computing, Second Edition.* s.l. : Addison Wesley, 2003.

21. ***Kacsuk, P., Fahringer, T., & Németh, Z.*** (2007). *Distributed and Parallel Systems From Cluster to Grid Computing.*

22. ***Abrahamsson, P., Salo, O., & Ronkainen, J***. (2002). *Agile software development methods Review and analysis.* Oulu.

23**. *Pressman, Roger.*** *Ingenieria del Software. La Habana : Felix Varela, 2005.*

24. **Balduino, Ricardo.** *Basic Unified Process: A Process for Small and Agile Projects.*

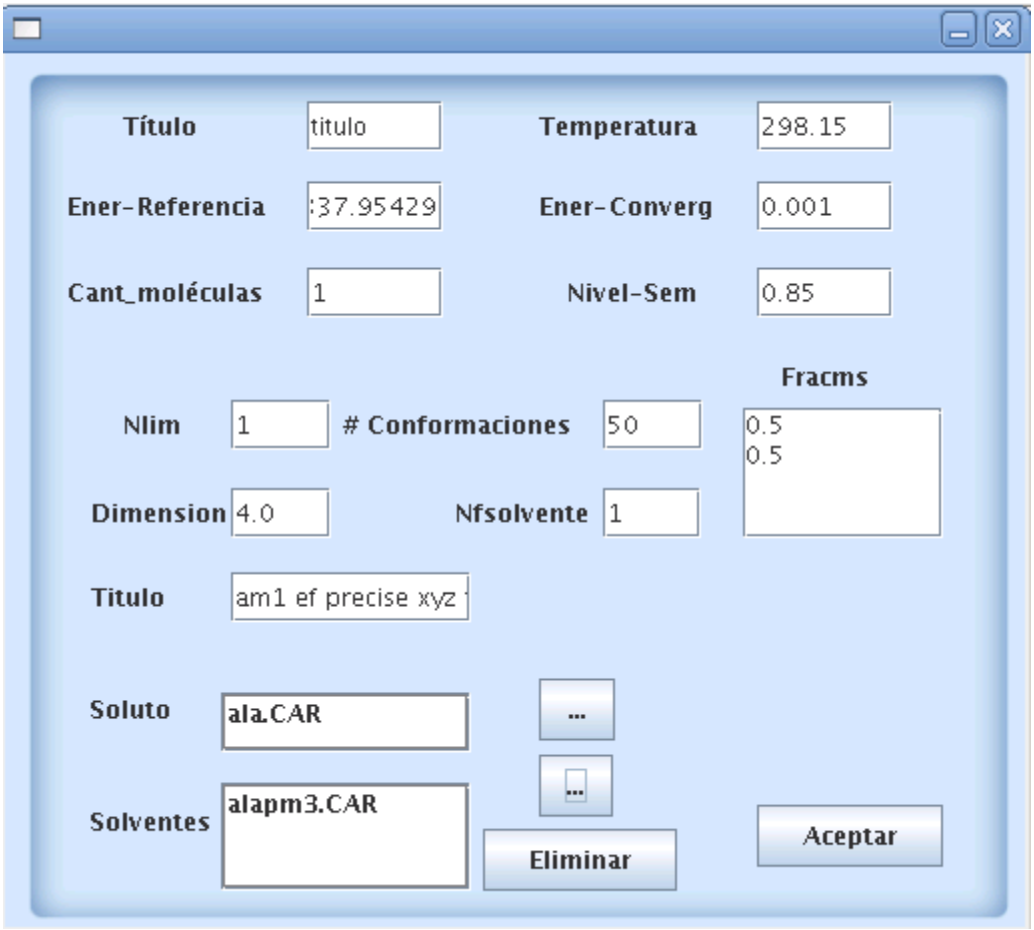
25. **KeaneT.A.** *General-Purpose Heterogeneous Distributed Computing System.* sl. : National University of Iriland Maynooth., 2004.

26. **Batista Pérez, Javier.** *MODELOS TEÓRICOS MECANOCUÁNTICOS DE LAS ESTRUCTURAS DE LOS COMPLEJOS NO-(H2O)n.* Ciudad de la Habana, 2007.

ANEXOS

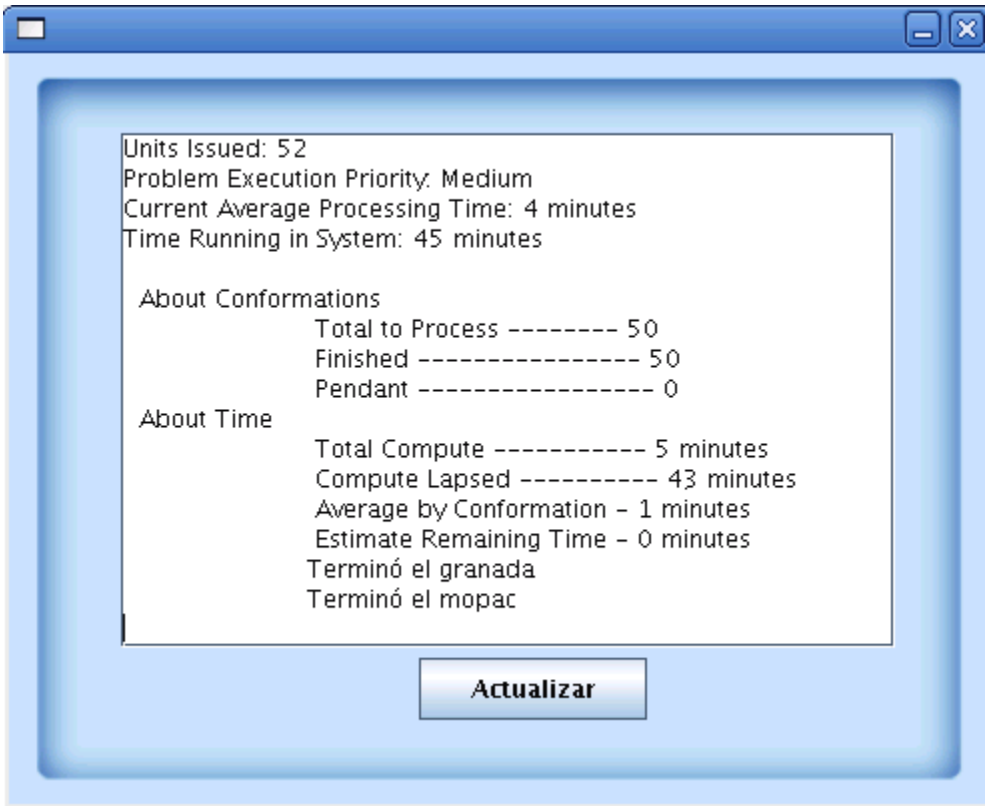
Anexo 1: Prototipos Funcionales.

Realizar cálculos químicos teóricos.

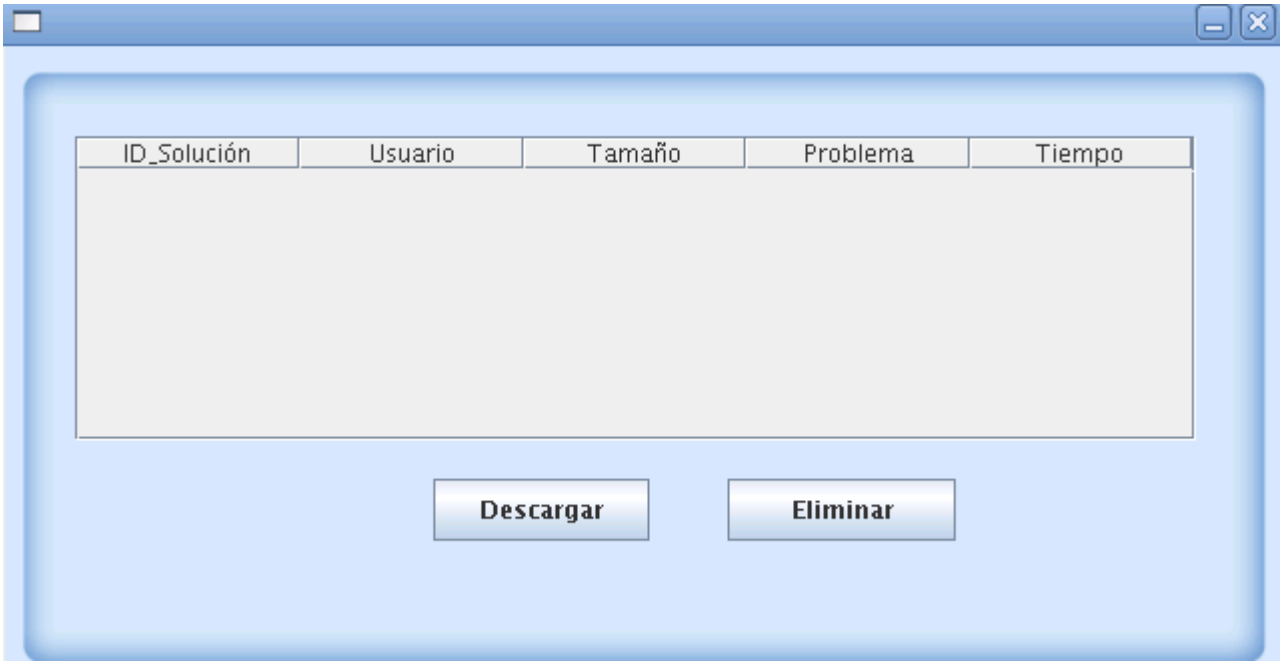


Buscar problemas en ejecución.

Mostrar estado de la ejecución.



Buscar resultados obtenidos.



Anexo 2: Pruebas de tiempo de ejecución.

Cálculos realizados en Horarios del día.

Celda	IP- Cliente	Tiempo - Celda	Hora Inicio	Hora Fin
1	10.35.17.33	260	9:41:39	9:45:59
2	10.35.17.15	250	9:41:46	9:45:56
3	10.34.12.5	277	9:41:49	9:46:26
4	10.35.17.16	431	10:11:57	10:19:08
5	10.34.16.26	181	9:42:04	9:45:05
6	10.34.16.10	291	9:42:08	9:46:59
7	10.33.5.33	470	9:42:11	9:50:01
8	10.33.6.27	281	9:42:17	9:46:58
9	10.34.9.28	213	9:42:24	9:45:57
10	10.34.9.16	241	9:42:32	9:46:33
11	10.34.5.11	251	9:42:25	9:46:36
12	10.34.15.23	270	9:42:27	9:46:57
13	10.33.8.52	211	9:42:25	9:45:56
14	10.34.9.29	261	9:42:26	9:46:47
15	10.34.13.16	183	9:42:27	9:45:30
16	10.34.8.19	351	9:42:26	9:48:17
17	10.35.18.37	240	9:42:28	9:46:28
18	10.34.5.5	211	9:42:29	9:46:00
19	10.34.12.31	150	9:42:28	9:44:58
20	10.34.5.43	111	9:42:31	9:44:22
21	10.34.5.35	397	9:42:32	9:49:09

22	10.34.9.38	191	9:42:33	9:45:34
23	10.35.17.23	286	9:42:34	9:47:20
24	10.34.16.11	232	9:42:40	9:46:32
25	10.34.6.34	342	9:45:45	9:51:27
26	10.34.9.34	183	9:42:38	9:45:41
27	10.34.5.4	261	9:42:37	9:46:58
28	10.34.5.33	130	9:42:41	9:44:51
29	10.34.5.13	251	9:42:44	9:46:55
30	10.34.16.27	170	9:42:47	9:45:37
31	10.34.5.30	170	9:42:46	9:45:36
32	10.34.5.18	190	9:42:55	9:46:05
33	10.35.18.33	363	9:42:48	9:48:51
34	10.35.18.1	181	9:42:47	9:45:48
35	10.34.8.14	180	9:42:49	9:45:49
36	10.34.9.27	191	9:42:51	9:46:02
37	10.34.5.21	251	9:42:54	9:47:05
38	10.33.9.32	340	9:42:57	9:49:37
39	10.34.5.23	290	9:42:57	9:47:47
40	10.34.9.12	250	9:43:01	9:47:11
41	10.35.17.21	302	9:43:01	9:48:03
42	10.34.15.29	280	9:43:02	9:47:42
43	10.33.5.14	237	9:43:01	9:46:58
44	10.34.8.33	404	9:43:29	9:50:13
45	10.33.9.37	232	9:43:01	9:45:53
46	10.34.14.17	283	9:43:05	9:47:48
47	10.35.19.9	200	9:43:03	9:46:23
48	10.33.9.10	301	9:43:03	9:48:04

49	10.34.15.7	190	9:43:06	9:46:16
50	10.34.14.7	301	9:43:02	9:48:03
Granada	10.35.17.19	4	**9:41:37**	9:41:43
Q3	10.34.5.37	24	10:19:16	**10:19:40**
Tiempo Total	**38 minutos**			

Tabla 1: Cálculos realizados (hora **9:41** AM - **10:19** AM)

Celda	IP- Cliente	Tiempo - Celda	Hora Inicio	Hora Fin
1	10.34.8.19	261	10:29:16	10:33:37
2	10.35.17.251	311	10:29:06	10:34:17
3	10.35.20.10	222	10:29:11	10:32:53
4	10.34.14.3	226	10:29:12	10:32:58
5	10.35.19.16	217	10:29:14	10:32:51
6	10.35.17.15	261	10:29:14	10:33:35
7	10.34.6.20	253	10:29:21	10:33:34
8	10.33.16.17	271	10:29:19	10:33:50
9	10.35.17.10	366	10:29:19	10:35:25
10	10.35.19.251	221	10:29:20	10:33:01
11	10.35.19.18	180	10:29:09	10:32:09
12	10.35.18.35	210	10:29:22	10:32:52
13	10.35.20.30	190	10:29:22	10:32:32
14	10.34.7.10	201	10:29:26	10:32:47
15	10.34.5.37	291	10:32:16	10:37:07
16	10.34.12.28	140	10:29:29	10:31:49

17	10.34.7.46	251	10:29:30	10:33:41
18	10.33.5.18	200	10:29:31	10:32:51
19	10.34.19.21	180	10:59:37	11:02:37
20	10.33.6.32	213	10:29:37	10:33:10
21	10.34.7.27	161	10:29:38	10:32:19
22	10.34.8.32	346	10:40:15	10:46:01
23	10.34.19.46	146	10:29:28	10:31:54
24	10.33.7.32	130	10:29:42	10:31:52
25	10.34.9.23	191	10:29:54	10:33:05
26	10.34.12.3	282	10:29:05	10:34:36
27	10.33.10.33	131	10:29:53	10:32:04
28	10.33.6.29	180	10:29:54	10:32:54
29	10.34.8.17	217	10:30:00	10:33:37
30	10.35.18.58	235	10:30:09	10:34:04
31	10.35.16.54	203	10:30:02	10:33:25
32	10.34.5.35	250	10:30:14	10:34:24
33	10.33.10.9	240	10:30:06	10:34:06
34	10.34.5.38	262	10:30:06	10:34:28
35	10.35.20.18	190	10:30:08	10:33:18
36	10.33.5.14	244	10:30:10	10:34:14
37	10.33.6.34	359	10:30:10	10:36:09
38	10.34.13.4	363	10:30:19	10:36:22
39	10.34.6.18	130	10:32:55	10:35:05
40	10.35.20.59	241	10:30:11	10:34:12
41	10.35.18.32	240	10:30:07	10:34:07
42	10.33.7.16	210	10:30:20	10:33:50
43	10.33.5.3	240	10:30:11	10:34:11

44	10.35.20.9	220	10:30:21	10:34:01
45	10.33.8.6	231	10:30:12	10:34:03
46	10.33.8.55	250	10:30:14	10:34:24
47	10.33.7.5	202	10:30:13	10:33:35
48	10.35.20.250	321	10:30:14	10:35:35
49	10.34.8.155	248	11:00:28	11:04:36
50	10.34.7.23	372	10:30:13	10:36:25
Granada	10.35.19.3	21	**10:28:42**	10:29:03
Q3	10.34.12.48	30	11:04:52	**11:05:22**
Tiempo total	**37 minutos**			

Tabla 3: Cálculos realizados (hora **10:28** AM - **11:05** AM)

Celda	IP- Cliente	Tiempo - Celda	Hora Inicio	Hora Fin
1	10.33.5.14	400	11:12:10	11:18:50
2	10.33.7.16	270	11:12:09	11:16:39
3	10.34.8.33	232	11:12:40	11:16:32
4	10.34.19.46	240	11:42:24	11:46:24
5	10.34.8.28	301	11:12:16	11:17:17
6	10.34.13.9	220	11:12:16	11:15:56
7	10.34.9.2	231	11:12:16	11:16:07
8	10.34.5.37	181	11:12:17	11:15:18
9	10.33.5.19	231	11:12:17	11:16:08
10	10.33.10.20	276	11:12:27	11:17:03
11	10.34.15.20	241	11:12:20	11:16:21

12	10.34.13.16	221	11:12:27	11:16:08
13	10.34.13.28	349	11:12:20	11:18:09
14	10.34.15.28	351	11:12:21	11:18:12
15	10.33.9.9	189	11:12:19	11:15:28
16	10.34.5.16	163	11:12:21	11:15:04
17	10.35.17.251	321	11:15:04	11:20:25
18	10.34.9.38	201	11:12:30	11:15:51
19	10.34.8.39	241	11:12:21	11:16:22
20	10.35.20.222	191	11:20:28	11:23:39
21	10.34.9.4	151	11:12:22	11:14:53
22	10.34.14.34	358	11:12:26	11:18:24
23	10.34.5.40	240	11:12:23	11:16:23
24	10.35.18.14	262	11:12:32	11:16:54
25	10.34.9.16	260	11:12:32	11:16:52
26	10.33.5.12	189	11:12:23	11:15:34
27	10.35.19.10	211	11:12:23	11:15:54
28	10.33.9.6	230	11:12:27	11:16:17
29	10.35.17.8	174	11:22:38	11:25:32
30	10.34.6.39	200	11:12:28	11:15:48
31	10.34.9.29	272	11:12:40	11:17:12
32	15:12:39	190	11:12:39	11:15:49
33	10.34.6.21	152	11:12:30	11:15:01
34	10.33.6.18	230	11:12:36	11:16:26
35	10.33.8.52	220	11:12:41	11:16:21
36	10.33.8.18	402	11:12:31	11:19:13
37	10.33.8.16	230	11:12:32	11:16:22
38	10.34.9.34	241	11:12:41	11:16:42

39	10.35.17.35	526	11:13:23	11:22:09
40	10.33.10.30	230	11:12:32	11:16:22
41	10.34.14.2	254	11:12:37	11:16:51
42	10.35.20.17	296	11:12:57	11:17:53
43	10.34.5.39	302	11:17:06	11:22:08
44	10.33.9.2	281	11:12:39	11:17:20
45	10.35.18.12	211	11:12:43	11:16:14
46	10.33.9.35	274	11:12:40	11:17:14
47	10.34.16.16	200	11:12:40	11:16:00
48	10.34.5.30	276	11:12:45	11:17:20
49	10.34.8.9	220	11:12:52	11:16:32
50	10.33.6.9	229	11:12:41	11:16:32
Granada	10.34.7.14	5	**11:12:00**	11:12:05
Q3	10.34.5.45	5	11:46:30	**11:46:36**
Tiempo Total	**34 minutos**			

Tabla 4: Cálculos realizados (hora **11:12** AM - **11:46** AM)

Celda	IP- Cliente	Tiempo - Celda	Hora Inicio	Hora Fin
1	10.35.19.230	252	14:38:47	14:42:59
2	10.33.18.27	291	14:38:46	14:43:37
3	10.33.5.11	171	14:39:00	14:41:51
4	10.33.7.32	250	14:38:59	14:43:09
5	10.33.6.20	250	14:38:53	14:43:03

6	10.33.7.16	283	14:38:51	14:43:34
7	10.35.18.14	210	14:38:57	14:42:27
8	10.33.10.28	221	14:38:52	14:41:32
9	10.34.12.27	224	14:38:52	14:42:36
10	10.34.16.26	221	14:38:55	14:42:36
11	10.35.19.15	191	14:38:56	14:42:07
12	10.33.7.1	286	14:38:54	14:43:14
13	10.34.8.2	172	14:38:57	14:41:49
14	10.34.7.8	281	14:38:56	14:43:37
15	10.35.19.112	240	14:38:56	14:42:56
16	10.33.5.26	320	14:39:08	14:44:28
17	10.33.7.27	371	14:38:58	14:45:09
18	10.34.9.12	161	14:38:57	14:41:38
19	10.34.5.2	230	14:39:05	14:42:55
20	10.34.8.29	234	14:39:03	14:42:57
21	10.34.16.16	529	14:39:23	14:47:52
22	10.33.6.34	240	14:39:04	14:43:04
23	10.34.5.14	170	14:40:13	14:43:03
24	10.34.15.19	200	14:39:07	14:42:27
25	10.34.8.21	278	14:39:05	14:43:49
26	10.33.5.1	221	14:39:20	14:43:01
27	10.34.13.9	261	14:39:19	14:43:40
28	10.33.9.39	333	14:39:09	14:44:42
29	10.34.7.16	211	14:39:11	14:42:42
30	10.33.6.15	171	14:39:21	14:42:12
31	10.34.13.1	141	14:39:11	14:41:32
32	10.35.17.34	140	14:39:13	14:41:33

33	10.34.7.9	151	14:39:13	14:41:44
34	10.35.19.10	195	14:39:15	14:42:30
35	10.33.7.30	201	14:39:13	14:42:34
36	10.33.18.23	252	14:39:29	14:43:41
37	10.33.5.4	200	14:39:25	14:42:45
38	10.33.9.35	477	14:39:16	14:47:13
39	10.34.13.20	471	14:39:20	14:47:11
40	10.33.10.8	160	14:39:25	14:42:05
41	10.33.7.19	238	14:39:20	14:43:18
42	10.35.17.35	679	14:40:05	14:51:24
43	10.35.18.35	181	14:39:26	14:42:27
44	10.35.17.13	252	14:39:22	14:43:34
45	10.33.10.44	150	14:39:23	14:41:53
46	10.35.19.24	161	15:16:07	15:18:48
47	10.34.7.10	271	14:39:24	14:43:55
48	10.33.5.33	310	15:16:07	15:21:17
49	10.34.15.28	291	14:39:24	14:44:15
50	10.34.8.8	182	14:39:29	14:42:31
Granada	10.34.12.3	5	**14:38:37**	14:38:42
Q3	10.34.8.11	3	15:21:27	**15:21:30**
Tiempo Total	**43 minutos**			

Tabla 5: Cálculos realizados (hora **14:38** PM - **15:21** PM)

Cálculos realizados en horarios de la noche.

	IP- Cliente	Tiempo - Celda	Hora Inicio	Hora Fin
1	10.34.8.18	261	21:05:19	21:09:40
2	10.34.14.14	211	21:45:28	21:48:59
3	10.35.19.15	280	21:05:23	21:10:03
4	10.34.19.148	180	21:35:23	21:38:23
5	10.35.17.14	243	21:05:23	21:09:26
6	10.34.13.13	238	21:05:27	21:09:25
7	10.35.20.24	96	21:35:00	21:36:36
8	10.34.7.3	221	21:05:26	21:09:07
9	10.33.6.2	240	21:05:26	21:09:56
10	10.33.7.32	200	21:05:34	21:08:54
11	10.34.19.24	252	21:35:31	21:39:43
12	10.34.13.14	305	21:05:30	21:10:35
13	10.34.14.13	350	21:05:33	21:11:43
14	10.34.8.33	212	21:05:59	21:09:11
15	10.33.6.9	287	21:45:58	21:50:45
16	10.33.6.45	320	21:05:36	21:10:56
17	10.35.19.16	183	21:05:36	21:08:39
18	10.35.18.31	150	21:05:36	21:08:06
19	10.34.9.17	331	21:05:37	21:11:08
20	10.33.18.18	251	21:05:38	21:09:49
21	10.34.6.29	242	21:05:40	21:09:42
22	10.34.13.20	191	21:05:40	21:08:51
23	10.33.7.10	162	21:05:44	21:08:26
24	10.35.18.37	201	21:05:42	21:09:03

25	10.33.18.23	234	21:05:50	21:09:44
26	10.33.9.7	389	21:06:22	21:12:51
27	10.35.17.35	275	21:05:46	21:10:21
28	10.33.18.27	352	21:05:51	21:11:43
29	10.34.14.14	211	21:45:28	21:48:59
30	10.35.17.21	231	21:05:47	21:09:38
31	10.33.9.18	312	21:45:34	21:50:46
32	10.35.20.20	364	21:05:49	21:11:53
33	10.34.6.24	160	21:05:52	21:08:32
34	10.33.9.6	330	21:05:48	21:11:18
35	10.34.6.14	180	21:05:49	21:08:49
36	10.34.6.20	201	21:05:50	21:09:11
37	10.33.9.32	257	21:05:51	21:10:08
38	10.34.19.34	100	21:12:09	21:13:49
39	10.35.17.240	501	21:05:53	21:14:14
40	10.34.9.20	190	21:05:53	21:09:03
41	10.33.18.10	243	21:05:53	21:09:56
42	10.33.18.15	231	21:06:01	21:09:52
43	10.34.20.24	281	21:36:00	21:40:41
44	10.34.20.12	230	21:35:57	21:39:47
45	10.34.9.1	385	21:05:54	21:12:19
46	10.33.18.28	279	21:45:27	21:50:06
47	10.34.12.48	263	21:05:53	21:10:26
48	10.35.17.6	218	21:06:06	21:09:44
49	10.33.7.33	200	21:05:57	21:09:17
50	10.34.13.7	210	21:45:48	21:49:18
Granada	10.34.8.18	5	**21:5:11**	21:5:16

Q3	10.35.20.250	14	21:50:59	**21:51:13**
Tiempo Total	**46 minutos**			

Tabla 6: Cálculos realizados (hora **21:05** PM - **21:51** PM)

Celda	IP- Cliente	Tiempo - Celda	Hora Inicio	Hora Fin
1	10.34.9.22	230	00:08:43	00:12:33
2	10.34.6.34	436	00:39:03	00:46:19
3	10.34.5.2	250	00:08:45	00:12:55
4	10.34.8.10	220	00:09:00	00:12:40
5	10.33.7.24	308	00:08:48	00:13:56
6	10.34.12.33	311	00:08:55	00:14:06
7	10.33.9.26	150	00:12:02	00:14:32
8	10.34.6.13	240	00:08:59	00:12:59
9	10.34.7.27	280	00:08:44	00:13:24
10	10.33.7.16	231	00:08:50	00:12:41
11	10.34.15.13	182	00:09:11	00:12:13
12	10.34.6.11	191	00:09:03	00:12:14
13	10.34.5.39	302	00:08:52	00:13:54
14	10.33.5.13	172	00:08:52	00:11:44
15	10.34.6.12	181	00:24:33	00:27:34
16	10.34.14.2	231	00:08:57	00:12:48
17	10.34.7.69	300	00:45:37	00: 50:37
18	10.34.9.3	222	00:08:57	00:12:39
19	10.34.9.23	200	1:15:41	1:19:01
20	10.35.19.194	267	00:08:59	00:13:26

21	10.33.7.2	186	00:09:00	00: 12:06
22	10.34.13.10	579	00:09:29	00:19:08
23	10.34.14.6	265	00:10:36	00:15:01
24	10.34.5.9	200	00:09:02	00:12:22
25	10.34.9.14	180	00:09:02	00:12:02
26	10.34.15.9	180	00:45:38	00:48:38
27	10.34.5.43	180	00:09:03	00:12:03
28	10.34.15.38	331	00:09:17	00:14:48
29	10.34.9.30	161	00: 09:04	00:11:55
30	10.34.9.6	171	00:09:07	00:11:58
31	10.34.15.17	160	00:09:07	00:11:47
32	10.33.9.15	190	00:09:08	00:12:18
33	10.34.19.14	121	00:45:37	00:47:38
34	10.33.6.13	310	00:09:10	00:14:20
35	10.34.7.23	211	00:09:10	00:12:41
36	10.34.12.32	169	00:45:38	00:48:27
37	10.35.19.9	191	00:09:11	00:12:22
38	10.33.5.18	201	00:09:20	00:12:41
39	10.35.17.16	361	00:09:11	00:15:12
40	10.34.14.230	283	00:09:28	00:14:11
41	10.34.7.19	282	00:09:12	00:13:54
42	10.35.20.27	251	00:09:14	00:13:25
43	10.34.6.42	232	00:09:14	00:13:06
44	10.33.7.33	211	00:09:23	00:12:54
45	10.34.15.14	160	00:09:23	00:12:13
46	10.34.13.23	376	00:46:51	00:53:07
47	10.33.7.13	170	00:9:17	00:12:07

48	10.34.13.26	252	00:9:17	00:13:29
49	10.33.7.32	210	00:9:26	00:12:56
50	10.34.14.27	381	00:9:36	00:15:57
Granada	10.34.9.22	8	**00:8:31**	00:8:39
Q3	10.33.7.36	11	1:19:11	**1:19:22**
Tiempo Total	**71 minutos**			

Tabla 7: Cálculos realizados (hora **00:08** AM - **1:19** AM)

Celda	IP- Cliente	Tiempo - Celda	Hora Inicio	Hora Fin
1	10.33.10.5	220	1:18:15	1:21:55
2	10.34.13.3	281	1:18:5	1:22:46
3	10.33.5.14	180	1:18:7	1:21:7
4	10.35.20.11	250	1:48:12	1:52:22
5	10.33.8.18	253	1:18:17	1:22:30
6	10.34.14.8	381	1:15:33	1:21:54
7	10.34.15.29	251	1:18:24	1:22:35
8	10.33.18.20	281	1:18:12	1:22:53
9	10.35.20.33	314	1:18:41	1:23:55
10	10.33.7.20	361	1:18:11	1:24:12
11	10.34.15.21	201	1:18:10	1:21:31
12	10.34.9.30	180	1:18:23	1:21:23
13	10.34.9.14	290	1:18:23	1:23:13
14	10.34.12.47	260	1:18:14	1:22:34
15	10.33.9.37	260	1:18:14	1:22:34
16	10.35.20.30	291	1:18:16	1:23:7

17	10.34.16.14	231	1:18:16	12:22:7
18	10.34.7.29	151	1:18:17	1:20:48
19	10.33.7.11	291	1:18:17	1:23:8
20	10.34.5.38	150	1:18:27	1:20:57
21	10.35.18.20	202	1:18:37	1:21:59
22	10.33.7.35	190	1:18:28	1:21:38
23	10.34.5.43	180	1:18:28	1:21:28
24	10.34.5.35	320	1:18:23	1:23:43
25	10.34.16.34	356	1:18:27	1:24:31
26	10.35.17.9	297	1:18:20	1:23:17
27	10.34.5.13	132	1:18:27	1:20:39
28	10.34.14.26	214	1:18:27	1:22:1
29	10.33.8.42	280	1:18:25	1:22:5
30	10.34.8.8	162	1:18:25	1:21:7
31	10.35.17.13	280	1:36:24	1:41:4
32	10.33.10.3	280	1:18:25	1:21:5
33	10.33.7.13	301	1:18:33	1:21:34
34	10.33.7.2	201	1:21:55	1:25:16
35	10.34.7.8	171	1:18:28	1:21:19
36	10.35.19.222	235	1:18:31	1:22:26
37	10.34.15.14	180	1:18:37	1:21:37
38	10.35.17.6	198	1:49:11	1:52:29
39	10.34.6.11	210	1:18:37	1:21:57
40	10.34.8.32	180	1:18:39	1:21:39
41	10.35.17.26	150	1:18:32	1:21:2
42	10.34.9.9	290	1:18:32	1:22:22
43	10.34.8.21	270	1:18:29	1:22:59

44	10.33.18.16	200	1:22:27	1:25:47
45	10.33.9.15	220	1:18:42	1:22:22
46	10.33.9.9	274	1:19:0	1:23:34
47	10.34.8.7	280	1:18:36	1:23:16
48	10.33.10.32	270	1:18:35	1:23:5
49	10.33.5.26	180	1:18:36	1:21:36
50	10.34.8.5	749	1:21:35	1:34:4
Granada	10.33.9.8	21	**1:17:40**	1:18:1
Q3	10.34.8.4	30	1:52:53	**1:53:23**
Tiempo Total	**36 minutos**			

Tabla 8: Cálculos realizados (hora **1:17** AM - **1:53** AM)

Celda	IP- Cliente	Tiempo - Celda	Hora Inicio	Hora Fin
1	10.33.7.2	150	2:23:48	2:26:18
2	10.33.18.9	190	2:23:50	2:27:0
3	10.33.8.53	251	2:23:53	4:28:4
4	10.35.17.6	240	2:54:0	2:58:0
5	10.33.9.11	191	2:23:54	2:27:5
6	10.34.13.7	262	2:23:55	2:29:17
7	10.34.12.32	270	2:23:55	2:29:25
8	10.33.9.22	264	2:23:55	2:28:19
9	10.33.10.14	195	2:55:44	2:58:59
10	10.35.17.25	191	2:54:5	2:58:16
11	10.35.20.15	241	2:23:58	2:27:59
12	10.33.9.35	290	2:24:8	2:28:58

13	10.35.19.194	260	2:24:4	2:28:24
14	10.34.12.53	230	2:54:35	2:58:25
15	10.33.18.20	160	2:24:10	2:26:50
16	10.33.10.23	211	2:24:3	2:27:34
17	10.33.18.6	250	2:24:5	2:28:15
18	10.34.13.3	170	2:24:14	2:27:4
19	10.33.8.52	140	2:24:8	2:26:28
20	10.33.18.16	171	2:24:9	2:27:0
21	10.35.18.24	169	2:24:9	2:27:20
22	10.35.19.21	250	2:24:1	2:28:11
23	10.35.17.21	211	2:24:20	2:27:51
24	10.34.13.12	180	2:24:20	2:27:20
25	10.35.20.59	252	2:24:16	2:28:28
26	10.33.7.10	191	2:24:13	2:27:24
27	10.34.13.14	209	2:24:22	2:27:51
28	10.35.20.27	212	2:24:29	2:28:1
29	10.35.17.251	220	2:24:25	2:28:5
30	10.34.13.8	171	2:24:16	2:27:7
31	10.35.20.10	181	2:54:43	2:57:44
32	10.33.7.11	300	2:24:30	2:29:30
33	10.33.18.8	81	2:24:24	2:25:45
34	10.33.7.27	276	2:24:35	2:29:11
35	10.33.9.24	252	2:24:33	2:28:45
36	10.33.7.23	164	2:24:49	2:27:33
37	10.35.19.112	160	2:24:26	2:27:6
38	10.34.19.44	170	2:38:54	2:41:44
39	10.34.13.19	370	2:24:27	2:30:37

40	10.34.12.2	243	2:24:27	2:28:30
41	10.35.17.15	194	2:55:15	2:58:29
42	10.33.7.4	351	2:24:33	2:30:24
43	10.33.6.2	151	2:24:42	2:27:13
44	10.33.7.16	210	2:24:35	2:28:5
45	10.35.17.17	311	2:24:31	2:29:42
46	10.34.14.27	221	2: 24:45	2: 28:26
47	10.35.20.33	201	2:25:14	2:28:35
48	10.33.6.15	180	2:24:36	2:27:36
49	10.35.17.9	525	2:24:49	2:33:34
50	10.33.8.43	201	2:24:41	2:28:2
Granada	10.33.9.15	6	**2:23:41**	2:23:47
Q3	10.33.9.32	4	2:59:8	**2:59:12**
Tiempo Total	**36 minutos**			

Tabla 9: Cálculos realizados (hora **2:23** AM - **2:59** AM)

GLOSARIO DE TÉRMINOS Y SIGLAS

AVS: Sistemas visuales avanzados.

Catequina: flavoloides (pigmentos vegetales no nitrogenados) compuestos fitoquímicos.

Conformación: Posición de una molécula en un momento dado.

Checkpoint: están diseñadas para acelerar el proceso de producción y para aumentar la eficacia de la cadena de suministro, ofrece aplicaciones de identificación, trazabilidad y protección.

MIMD (Multiple Instruction, Multiple Data): Varios procesadores ejecutan diferentes instrucciones simultáneamente con diversos datos.

MPI: Message Passing Interface.

Plug-ins: Los **plug-ins** permiten a tu navegador realizar funciones específicas como ver gráficos en formatos especiales o reproducir archivos multimedia.

PVM: Parallel Virtual Machine.

sockets o RPC: Los sockets no son más que puntos o mecanismos de comunicación entre procesos que permiten que un proceso emita o reciba información con otro.

Speedup: Diferencia entre velocidad secuencia y velocidad en paralelo V(S)/V(P).

Switch: llamado Conmutador es un dispositivo de la subred que realiza una labor principal.

Tanimoto: **coeficiente** de Jaccard(coeficiente de semejanza).

1