Diseño e implementación de un clúster de cómputo de alto rendimiento

José de Jesús Rocha Quezada*, Salvador Botello Rionda*, José Miguel Vargas Félix*, Iván Agustín Munguía Torres*

RESUMEN

Este trabajo presenta el diseño, construcción y configuración de un sistema de computadoras agrupadas en un arreglo tipo clúster, el cual tiene un bajo costo y es comúnmente utilizado para resolver problemas que demandan grandes capacidades de cómputo, tanto de memoria como de procesamiento. En general estas tecnologías son costosas, pero en este trabajo se presenta una alternativa económica. Estos problemas no pueden ser resueltos en computadoras aisladas debido a la gran cantidad de datos que involucran y a la cantidad de operaciones numéricas que se requieren realizar. En este trabajo se muestran resultados en aplicaciones numéricas que tienen gran utilidad práctica (Mecánica de Solidos, Mecánica de Fluidos, Problemas Térmicos, etc.), en donde se refleja un buen desempeño y un adecuado uso de los recursos computacionales. El clúster mencionado en este trabajo ofrece una excelente capacidad de cómputo a un bajo costo.

ABSTRACT

This paper presents the design, construction and configuration of a computer system in a cluster type setup, which has a low cost and is commonly used to solve problems that require large computing capabilities, both memory storage and processing. In general, these technologies are expensive, but this paper presents a low-cost alternative. These problems cannot be solved on isolated computers due to the large amount of data involved and the amount of numerical operations required to perform. In this paper we show numerical results in applications that have great practical utility (Solid Mechanics, Fluid Mechanics, Thermal Problems, etc...), which reflect good performance and proper use of computational resources. The cluster mentioned in this paper provides excellent computing power at low cost.

Recibido: 1 de septiembre de 2011
Aceptado: 17 de noviembre de 2011
Artículo de Revisión

INTRODUCCIÓN

Los problemas actuales de modelación numérica requieren gran capacidad de cómputo. Si bien las computadoras personales son cada vez más rápidas y eficientes, para obtener un buen resultado en este tipo de aplicaciones se requiere mucha mayor capacidad de procesamiento y memoria. Para poder resolver problemas de modelación numérica, por ejemplo de yacimientos petroleros, mecánica de fluidos, mecánica de sólidos, solución de ecuaciones diferenciales parciales, entre otros, es necesario disponer de la capacidad de cómputo ofrecida por un clúster de computadoras. Este trabajo presenta el diseño, construcción y configuración de un sistema de computadoras agrupadas en una distribución tipo clúster, así como resultados de su aplicación a problemas reales.

Palabras clave:
Clúster; física computacional, programación en paralelo; OpenMP; MPI; Mecánica de Sólidos; problemas térmicos.
Keywords:
Cluster; computational physics; parallel programming; OpenMP; MPI; Solids Mechanics; heat diffusion.

Definición y estructura de un clúster

Un clúster de computadoras se puede definir como un conjunto o conglomerado de computadoras, construidas con componentes de hardware comunes, que se comportan como una única computadora (figura 1).

**Centro de Investigación en Matemáticas, A.C. Callejón Jalisco s/n. Valenciana, Guanajuato, Gto., México. C.P 36240. Teléfono: (473) 732-7155, Fax (473) 732-5749. Correos electrónicos: chuche@cimat.mx, botello@cimat.mx, miguelvargas@cimat.mx, munguia@cimat.mx.

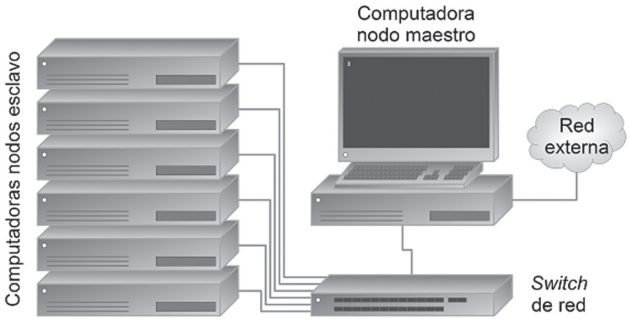

Figura 1. Clúster de cómputo de tipo Beowulf.

Un clúster es entonces un conjunto de computadoras, llamadas nodos, que están conectados entre sí a través de una red de cómputo de alta velocidad. Un ejemplo de este tipo de clústers son los clúster tipo Beowulf [8] como el que se muestra en la figura 1.

Cada nodo del clúster cuenta con varios procesadores y bancos de memoria, los cuales están interconectados ente sí. Los procesadores modernos son multinúcleo (o *multi-core*), cada núcleo es una unidad de procesamiento independiente. Esto es, cada procesador (chip, o encapsulado) contiene dos o más núcleos.

La velocidad de acceso a la memoria en el clúster se acelera por medio de memoria llamada caché, la consistencia de los datos entre los procesadores, los bancos de memoria y la memoria caché se mantiene por medio de circuitos de coherencia de caché. En la figura 2 se puede ver un diagrama esquemático de una configuración de un clúster de cuatro nodos.

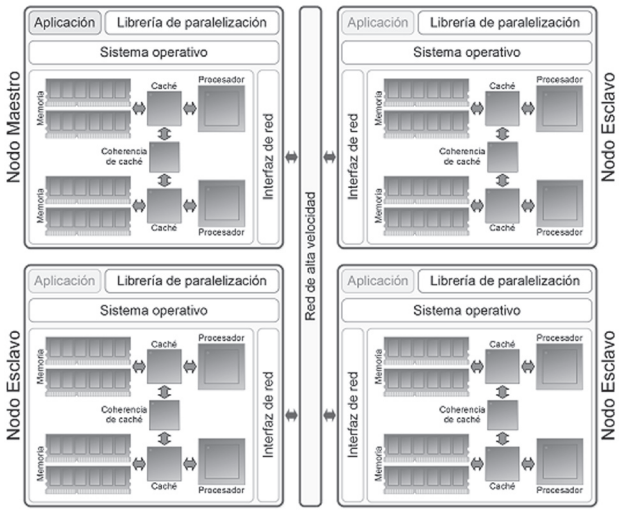

Figura 2. Diagrama esquemático de los componentes del clúster.

De manera lógica, cada nodo del clúster tiene una parte de hardware y otra de software. El hardware está compuesto por procesadores, memoria, interfaz de red y discos duros entre otros. En cuanto al software, el nivel bajo corresponde al sistema operativo, el medio consiste en las librerías de paralelización y el alto está representado por la aplicación que se desea ejecutar en el clúster.

La red de comunicaciones entre los nodos juega un papel muy importante en la eficiencia del equipo. Un *switch* de red normal no es suficiente, se debe elegir un *switch* especializado considerando el gran volumen de datos y la velocidad con la que son requeridos. Existen *switch* de alta velocidad de transmisión de datos tipo *Infiniband* pero son demasiado costos y su utilización puede limitar sustancialmente el poder de cómputo de un clúster. Es esencial realizar una buena selección del tipo de *switch* a utilizar a fin de garantizar un bajo tiempo de latencia (tiempo de espera al enviar datos de un nodo a otro) y capacidad de memoria intermedia para evitar congestionamiento y pérdida de datos.

Una aplicación o programa diseñado para ejecutarse en un clúster se ejecuta en el nodo maestro, el sistema operativo y la librería de paralelización se encargan de ejecutar copias de este programa en los nodos esclavos del clúster. Todas estas instancias de la aplicación se ejecutan en paralelo y trabajan en conjunto enviándose datos para colaborar en el cálculo de la solución final.

MÉTODOS

A continuación se presenta el diseño y la implementación del clúster que fue construido y se encuentra actualmente funcionando en el Centro de Investigación en Matemáticas CIMAT [2]. El nombre que se le dio al clúster fue "El Insurgente" en honor al Bicentenario de la independencia de México.

Figura 3. Emblema del clúster El Insurgente del CIMAT

Diseño del clúster

Consideraciones

Al diseñar el clúster se consideraron los siguientes aspectos:

Economía y mantenimiento: Utilización de componentes de bajo costo y además fácilmente reemplazables. Los componentes deben ser lo suficientemente comunes para poder ser reemplazados a un costo razonable en caso de fallo. Nosotros optamos por equipo marca DELL ya que cuenta con soporte y ofrece buena garantía en sus partes electrónicas. El consumo de energía eléctrica también debe ser considerado. Es importante incluir también el costo de mantenimiento y energía en el presupuesto destinado para el proyecto.

Adecuación de las instalaciones: Un clúster requiere de un ambiente controlado. Esto es, una habitación especial con sistema de enfriamiento, capacidad suficiente de carga eléctrica, control de humedad y un ambiente libre de polvo. Además se deben considerar el peso del equipo sobre el piso (ya sea falso o piso normal) Se recomienda colocarlo en los sótanos o en la plata baja de edificios.

En el CIMAT se contaba con un laboratorio de cómputo poco utilizado que reunía la mayoría de las características requeridas. La inversión más fuerte es el suministrar la capacidad de energía eléctrica que el conjunto de computadoras requiere.

Escalabilidad: Un clúster debe de ser escalable. Se recomienda seleccionar componentes de marcas reconocidas que soporten alta carga trabajo de forma continua. Si se cuenta con un presupuesto bajo, pero se piensa crecer en un futuro, es importante seleccionar hardware al que se le pueda incrementar la cantidad de memoria, la capacidad de los procesadores y la capacidad de almacenamiento. Se debe llegar a un punto intermedio entre comprar hardware moderno y costoso y hardware antiguo de precio bajo que se vuelva obsoleto rápidamente.

Se decidió armar dos clusters, uno con procesadores AMD Opteron en cada nodo y otro con procesadores Intel Xeon series E5500, ambos procesadores de 64 bits. Se eligieron memorias totalmente comerciales y fáciles de comprar en el mercado aunque de velocidad media, para no incrementar en demasía el costo. Hay casos como SGI, SUN, BLUE GEN de IBM, o computadores Vectoriales como las NEC SX-9 cuya arquitectura de computadoras es demasiado cara y difícil de conseguir soporte.

En la parte de redes se utilizaron switches *fast Ethernet* con una velocidad de 1000 Mbps, los cuales son más económicos que los de fibra óptica. Es muy importante que la persona que construya el clúster conozca a fondo los componentes y configuraciones físicas y lógicas que se van a utilizar tanto de red como de las computadoras.

Software: El sistema operativo debe constar con servicios dedicados para compartir y respaldar archivos, así como capacidad de acceder a los datos de trabajo rápidamente. El sistema operativo debe también proveer servicios de monitoreo y reporte de fallas en el sistema. Es posible cubrir estos requerimientos con Windows HPC, sin embargo, el costo de esta versión de Windows es alto. La alternativa es utilizar alguna distribución de Linux.

Después de probar diferentes distribuciones, se eligió Rocks Clusters System. Ésta distribución es fácil de instalar y configurar, además de tener la ventaja de ser un software libre con licencia "open source". Rocks Clusters System está basado en RedHat Enterprise Server 5. Se debe poner especial atención en configurar el nodo maestro que sirve como nodo de autenticación para red exterior. En cuanto a compiladores, se instaló GCC de GNU el cual también es software libre.

Hardware: Los nodos deben ser altamente escalables y dedicados a trabajar por largos periodos de tiempo, deben de contar con más de un procesador y tener una muy eficiente velocidad de procesamiento. Por lo tanto se seleccionó una arquitectura multi-núcleo de 64 bits compatible con x86_64, la cual cuenta con mayor capacidad de direccionamiento de memoria que las computadoras de 32 bits (el límite de estas últimas es de 4GB).

Interconexión: Se debe considerar también el tipo de cableado a utilizar para conectar los nodos. Estos van desde cable UTP de cobre hasta fibra óptica. Por cuestión económica seleccionamos el estándar UTP de cobre con la norma IEEE 802.11 con la categoría de cableado 5e. Los switches que se adquirieron fueron de una velocidad de 1000 Gbps. El primero que se probó era de poca eficiencia en el ciclo interno y sin soporte de cargas masivas de intercambio de datos entre los puertos. Cuando se ejecutaron los procesos en el clúster, el switch se colapsaba debido a que sus recursos internos de memoria no eran suficientes. Se tomó la decisión de remplazar el switch por uno administrable, eficaz y rápido en el intercambio de información entre los puertos, con un ancho de banda interno alto y con una arquitectura sin bloqueos.

Almacenamiento: En caso de que los dispositivos de almacenamiento no basten para contener los datos de trabajo, se debe considerar la instalación de sistemas de discos de red. En nuestro caso utilizamos un

disco de gran capacidad en el nodo maestro y discos de poca capacidad en los nodos esclavos. Esto debido a que las aplicaciones desarrolladas para El Insurgente no requieren almacenamiento en disco en los nodos esclavos.

Recursos humanos especializados: Para garantizar el buen funcionamiento de un clúster se requiere contar con técnicos altamente especializados y actualizados en tecnologías de la información que puedan estar disponibles en la construcción y operación del mismo, a fin de sintonizar de forma efectiva el funcionamiento del equipo, así como modificaciones, actualizaciones y cuidado del equipo.

Construcción

El Insurgente fue diseñado y construido específicamente para aplicaciones de cómputo científico que requieren grandes volúmenes de datos en memoria RAM, programación distribuida con un híbrido de MPI [10] y OpenMP [11], usando redes de bajo costo. Se colocó en un espacio libre de polvo de aproximadamente veinte metros cuadrados, con humedad controlada, sistema de enfriamiento y con el suministro de energía eléctrica adecuado para su funcionamiento. Adicionalmente se instaló un sistema de respaldo de energía de alta capacidad. Las características generales del clúster El Insurgente del CIMAT son las siguientes:

Software

El software instalado en cada uno de los nodos es el siguiente:

- Sistema operativo: Rocks Clúster Linux 5.3 (64 bits)
- Compiladores: GCC 4.5.2. Soporte para C, C++, Fortran y OpenMP
- Librería de manejo de mensajes: Open-MPI 1.5.3
- Software de pre y post procesamiento: GiD 9.0.6

Hardware

El clúster está compuesto por 32 nodos con las siguientes características:

- Nodo Maestro:
  - Procesador: AMD Quad Core Opteron 2350 HE (8 núcleos), Memoria: 12 GB, Disco Duros: SATA 250 GB, 1 TB. Ambos de 7200 revoluciones por minuto.
- 16 Nodos Esclavos:
  - Procesador: 2 x AMD Quad Core Opteron 2350 HE (8 núcleos), Disco Duro: SATA 160 GB, Memoria: 12 GB con velocidad de 667Mhz.
- 15 Nodos Esclavos:
  - Procesador: Intel(R) Xeon(R) CPU E5502 (4 núcleos), Disco Duro: SATA 160 GB, Memoria: 16 GB con velocidad de 1333 MHz.

El hardware adicional es el siguiente:

- Interconexión de Red
- Switch Linksys: 48 puertos, 1Gbps, modelo SLM2048. Para administración interna.
- Switch SMC: 48 puertos. 1 Gbps (red interna). Modelo SMC8150L2. Capa 2.
- Switch Linksys: 24 puertos. 1 Gbps, modelo SRW2024 (red externa).
- Sistema de energía ininterrumpida (UPS) de 30 KVA, salida senoidal, tres salidas 120 v y entrada de 220 v. El equipo trabaja a 120 v.

Recursos totales del clúster:

- Núcleos de unidades de procesamiento (cores): 216
- Capacidad en memoria: 455.3 GB
- Capacidad en disco: 5.7 TB

Operación

El consumo eléctrico del equipo es de 6.3 kWh cuando está en estado latente y de 7.2 kWh en plena operación. El calor se logró disipar mediante el empleo de un aire acondicionado de 2 toneladas, de 220 volts, configurado a 24 °C. Este aire está colocado en la parte superior de tal manera que se tiene un pasillo frío al frente del clúster y un pasillo caliente en la parte trasera de los racks del clúster.

Los rendimientos del equipo en general presentan eficiencias de al menos 80 % en el total de ejecuciones realizadas, siempre y cuando la aplicación se encuentre en la memoria RAM del equipo y la comunicación entre maquinas se realice de forma moderada.

Aplicaciones

El principal uso del clúster El Insurgente ha sido resolver problemas de modelación de fenómenos físicos, como deformación de sólidos [1] y difusión de calor [2]. Éstos problemas son modelados con el método

de elementos finitos, utilizando programas que se han desarrollado en el CIMAT [13] [14]. Este método consiste en dividir la geometría del problema en una malla constituida de pequeñas figuras geométricas [5] [6] [7]. La malla generada con esta geometría tiene una equivalencia con un sistema de ecuaciones lineales. La mayor parte del tiempo de procesamiento del clúster es utilizado en resolver grandes sistemas de ecuaciones.

Estos sistemas de ecuaciones de gran tamaño (decenas o cientos de millones de ecuaciones) se requieren para modelar problemas con geometrías complejas que demandan una alta precisión en la solución. Afortunadamente la modelación con el método de elementos finitos produce matrices ralas (o dispersas) [3], en donde la gran mayoría de las entradas son cero. Utilizando un esquema inteligente de almacenamiento en memoria se pueden llegar a resolver problemas del orden de centenas de millones de variables en un tiempo razonable. Si se tuvieran que almacenar todos los valores de la matriz, se requerirían varios TB de memoria RAM.

Para aprovechar todos los procesadores del clúster, las mallas son seccionadas de tal manera que cada procesador resuelve de forma independiente e iterativa una parte de la geometría. Una vez resuelta, cada nodo comunica la solución a los nodos adyacentes hasta converger al resultado final. Este esquema de paralelización, conocido como descomposición de dominios, repercute en una disminución de los tiempos de solución. Es decir, un problema que tardaría días en una computadora personal, ahora puede resolverse en cuestión de minutos u horas utilizando todos los nodos del clúster.

RESULTADOS

A continuación se presentan resultados de ejemplos ejecutados en el clúster que muestran su buen comportamiento. Es conocido que cuando se resuelven sistemas de ecuaciones muy grandes, los tiempos requeridos para resolverlos aumentan exponencialmente. En los ejemplos de este trabajo se presentan resultados que crecen casi linealmente en tiempo de ejecución cuando se incrementa el numero de ecuaciones a resolver, lo cual muestra el buen desempeño de nuestro sistema utilizando MPI, la técnica de descomposición de dominios y el método de Elemento Finito para hacer la discretización numérica. La eficiencia probada de nuestro sistema se encuentra alrededor de un 80 %. El cálculo de esa eficiencia se efectuó tomando el tiempo teórico de solución (el tiempo de que tardaría en correr en una maquina, dividido entre el número de maquinas) entre el tiempo real de ejecución en el clúster. El tiempo de ejecución en el clúster siempre es mayor que el tiempo teórico, puesto que siempre se requerirá tiempo para las comunicaciones entre los diferentes procesos. No existe en el mundo un clúster como El Insurgente por lo cual no pueden existir comparaciones directas. Las comparaciones se realizan contra la eficiencia del mismo equipo. Es de resaltar que el costo del clúster El Insurgente es pequeño en comparación con equipos comerciales y su desempeño es muy similar en las aplicaciones que se presentan.

Los siguientes resultados no buscan manifestar un aceleramiento al utilizar cómputo en paralelo. Eso es muy difícil de cuantificar en problemas grandes, donde entran en juego factores muy dependientes de la arquitectura utilizada y que están más allá del diseño del algoritmo, como son: el tamaño del cache de los núcleos, la velocidad de la memoria RAM, la asignación de tiempo de procesamiento y memoria a cada core por parte del sistema operativo, las capacidad de memoria temporal de los *switch* de red (hablaremos en la sección siguiente de éste factor que resultó ser muy importante). La intención es mostrar que es posible trabajar problemas con alto número de grados de libertad utilizando equipo de bajo costo que puede estar al alcance de grupos de investigación pequeños.

Difusión de calor de un dominio simple en 2D

En este ejemplo se presenta una geometría sencilla, una lámina cuadrada de metal en la cual las temperaturas de sus cuatro bordes están fijas. La modelación consiste en visualizar cómo, a partir de estas temperaturas fijas, el calor se distribuye en la superficie de la lámina, generando un patrón de temperatura sobre toda la superficie [2].

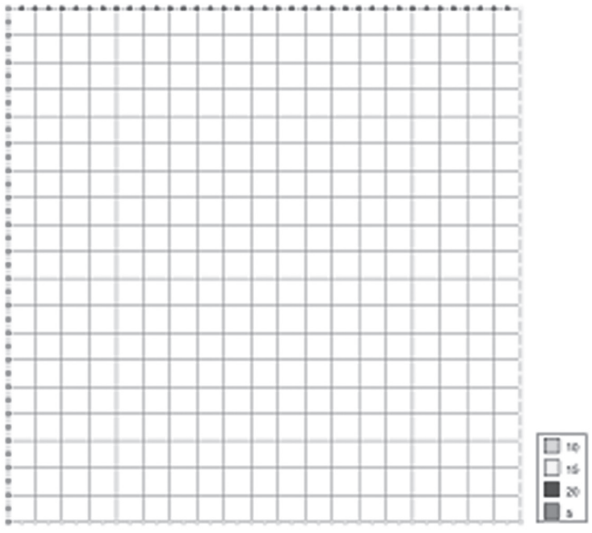

Figura 4. Lámina cuadrada de metal discretizada con una malla de elementos finitos cuadrangulares.

Se utilizaron mallas de cuadrícula regular con diferentes grados de refinamiento, esto con la finalidad de experimentar con sistemas de ecuaciones de diferente número de variables; desde diez millones hasta ciento cincuenta millones de ecuaciones.

La figura 5 muestra los resultados de distribución de la temperatura, así como la partición de la geometría. Cada sección de la placa fue resuelta por un núcleo diferente del clúster. Estas áreas están traslapadas para acelerar la convergencia del algoritmo.

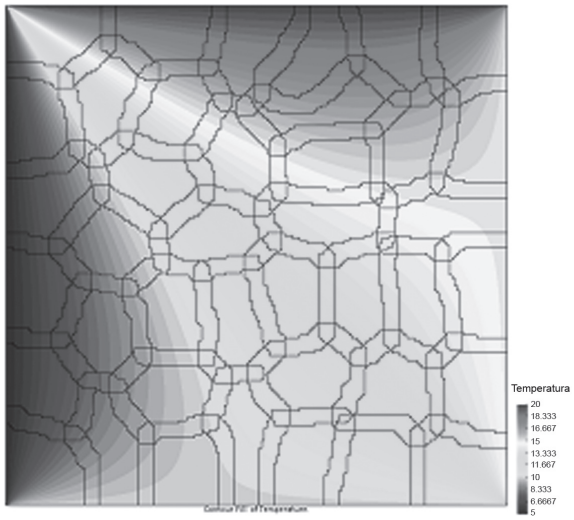

Figura 5. Resultados de la simulación numérica.
Isocontornos de distribuciones de temperaturas.
Notar en el fondo la discretización en subdominios.

La figura 6 muestra en el eje horizontal el número de ecuaciones, y en el eje vertical el tiempo requerido para llegar a la solución del sistema.

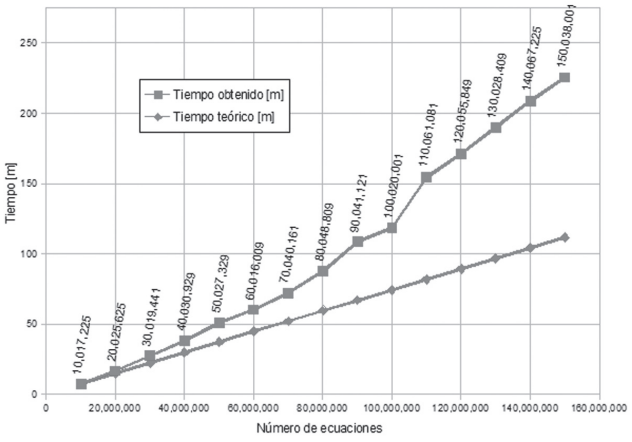

Figura 6. Tiempos requeridos para resolver el problema con diferentes sistemas de ecuaciones.

En la figura 7 se grafica el número de ecuaciones contra la cantidad de memoria requerida.

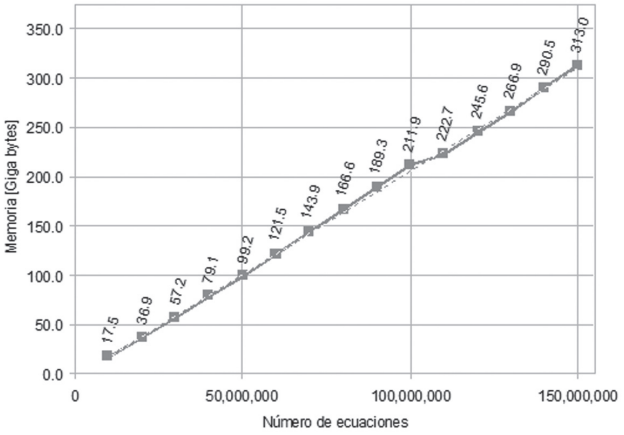

Figura 7. Memoria requerida para resolver el problema.

La Tabla 1 resume los resultados para los diferentes tipos de configuraciones.

Tabla 1.
Comparación resultados para la placa metálica.

Ecuaciones	Tiempo [m]	Memoria [bytes]	Núcleos
10 017 225	7,5	17 467 525 032	56
20 025 625	16,8	36 909 221 494	56
30 019 441	27,5	57 213 078 948	56
40 030 929	38,4	79 072 999 934	56
50 027 329	51,1	99 220 216 594	56
60 016 009	60,4	121 465 187 002	56
70 040 161	72,2	143 941 390 260	56
80 048 809	87,9	166 611 390 986	56
90 041 121	108,7	189 278 942 680	56
100 020 001	118,6	211 854 225 872	56
110 061 081	154,7	222 745 388 922	104
120 055 849	171,3	245 613 992 474	104
130 028 409	189,8	266 911 410 778	104
140 067 225	208,8	290 502 613 204	104
150 038 001	225,5	313 020 135 544	104

Deformación de estructuras en 3D

En este problema se analiza la deformación de una estructura de concreto sujeta a la acción del viento en alguna parte de su superficie. La estructura corresponde a uno de los edificios del Departamento de Ciencias Económico Administrativas de la Universidad de Guanajuato. La figura 8 muestra la geometría del problema.

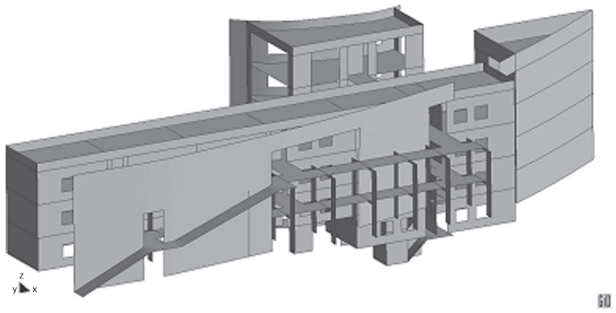

Figura 8. Geometría del edificio.

Para la modelación de este problema se utilizó la Teoría de Deformación Lineal y Elástica de Sólidos con la técnica numérica del Método de los Elementos Finitos [1]. Nuevamente se utiliza el esquema de descomposición de dominios, en la figura 9 se muestra en colores diferentes el particionamiento.

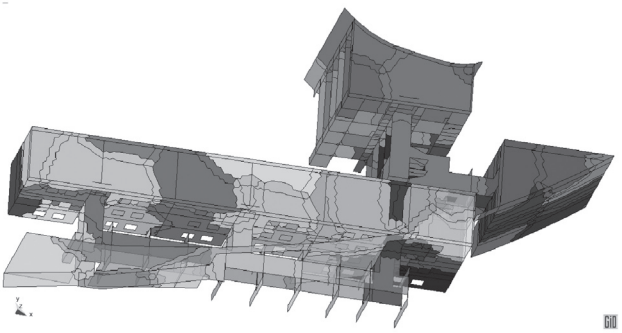

Figura 9. Descomposición de dominios del edificio.

Se realizaron varias discretizaciones de la geometría con diferentes grados de refinamiento. Los sistemas de ecuaciones resultantes de estas discretizaciones van desde 10 millones, hasta 40 millones de variables o grados de libertad. El método de descomposición de dominios utilizado es el método alternante de Schwarz [12], particionado en 60 subdominios. Para resolver el sistema de ecuaciones en cada subdominio se utilizó el método de factorización Cholesky para matrices ralas.

Para el cálculo numérico se utilizaron 15 computadoras del clúster, cada computadora cuenta con dos procesadores de 2 núcleos cada uno, lo que nos da un total de 60 núcleos. Cada subdominio entonces es resuelto por un núcleo. Se considera que el problema ha encontrado la solución cuando el error es menor a $1x10^{-5}$.

La figura 10 muestra el resultado final después de conjuntar los resultados de cada dominio. Se muestran los desplazamientos en las paredes del edificio. El color oscuro indica un mayor desplazamiento. El blanco corresponde a las superficies que permanecieron sin desplazamiento.

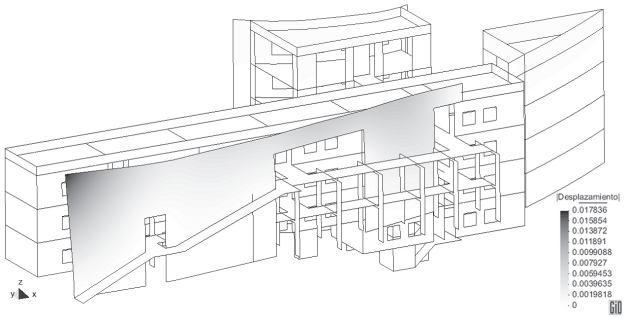

Figura 10. Isocontornos de deslazamientos totales del edificio.

La tabla 2 muestra los tiempos requeridos para la solución y la memoria total requerida.

Tabla 2.
Comparación resultados para la deformación del edificio.

Ecuaciones	Tiempo [horas]	Memoria [Giga bytes]
10 195 986	1,06	42,6
20 786 520	5,02	92,7
30 350 046	9,54	142,2
40 707 684	13,83	194,5

En las figuras 11 y 12 se grafican los resultados de la tabla 2.

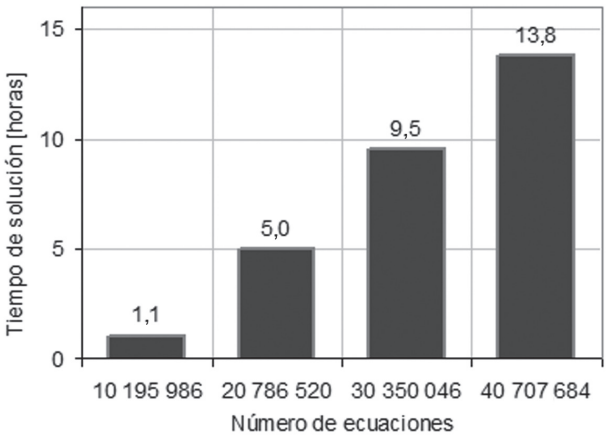

Figura 11. Tiempo consumido.

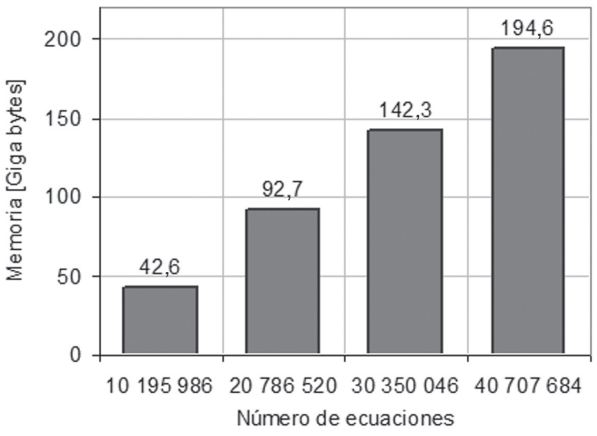

Figura 12. Memoria requerida en El Insurgente.

Difusión de calor en 3D

Presentamos ahora un ejemplo de difusión de calor [2] con una geometría compleja. El modelo corresponde a un disipador de calor utilizado para enfriar procesadores. El disipador consta de una serie de placas de aluminio interconectadas por conductores de cobre. Los conductores llegan a una base la cual está en contacto directo con el procesador. La figura 13 muestra la geometría. La figura 14 muestra el mallado del modelo.

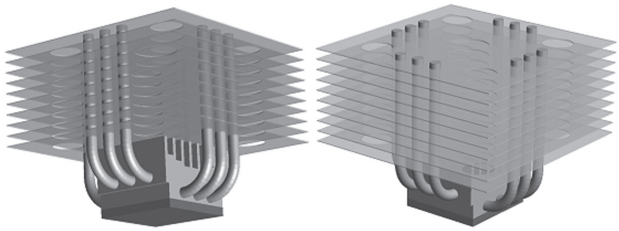

Figura 13. Disipador de calor.

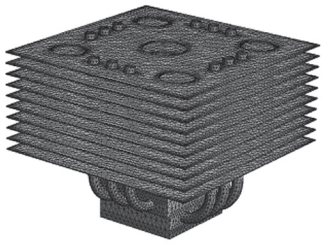

Figura 14. Malla de elementos finitos utilizada en le discretización del disipador de calor.

El mallado de este problema generó un sistema de 2 267 539 ecuaciones, y fue resuelto utilizando 14 computadoras del clúster. La figura 15 muestra la distribución de temperatura resultante en el disipador.

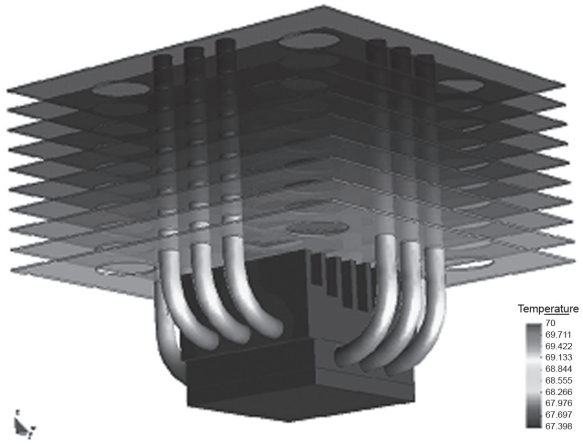

Figura 15. Isocontornos de temperatura en el Disipador.

En la figura 16 se muestran los tiempos requeridos para diferentes discretizaciones del dominio en estudio. En la figura 17 se presentan los requerimientos de memoria para poder resolver el problema con diferentes discretizaciones de la geometría.

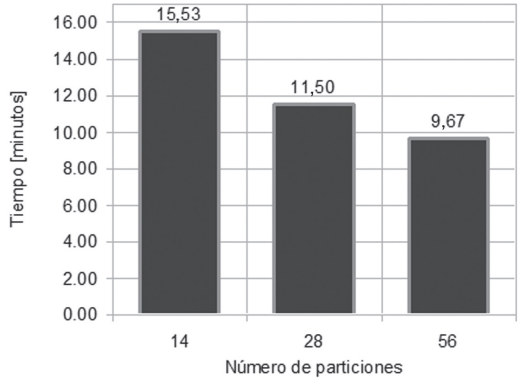

Figura 16. Tiempo requerido para resolver el problema del disipador de calor con un número diferente de particiones

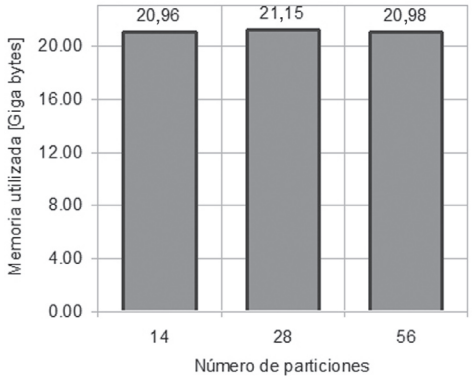

Figura 17. Cantidad de memoria requerida para resolver el problema del disipador.

La tabla 3 resume los resultados de los tiempos y memoria utilizados para modelar numéricamente el disipador de calor.

Tabla 3.
Resumen de resultados de la modelación numérica del disipador de calor.

Particiones	Núcleos por partición	Tiempo [minutos]	Memoria [Giga bytes]
14	4	15,53	20,96
28	2	11,50	21,15
56	1	9,67	20,98

DISCUSIÓN

El diseño de los programas con los cuales se obtuvieron los resultados anteriores se hizo considerando la arquitectura del cluster del CIMAT, tratando de sacar el máximo provecho de sus recursos. Por ejemplo, como la red no es de alta velocidad, se eligió y diseñó el algoritmo de descomposición de dominios para que transmitiera menor cantidad de información por la red, a costo de utilizar más memoria RAM. A continuación haremos una discusión de las consideraciones que se toman en cuenta al diseñar programas que buscan sacar el máximo provecho de clústers con características similares a El Insurgente.

Resolver problemas con decenas de millones de grados de libertad implica tener consideraciones que no entran al resolver problemas de escalas menores, uno de los más importantes es la velocidad de transmisión de datos. En el caso de un clúster conformado con computadoras multi núcleo, este problema se da en varias escalas.

En la escala mas pequeña está la memoria *cache* del núcleo, ésta es de entre 2 MB y 16 MB. Cuando se trabajan con pocos datos es posible que todas las operaciones se realicen en la memoria *cache*, que es entre 10 y 20 veces más veloz que la memoria RAM [15]. Cuando el problema sale de la capacidad del *cache* los tiempos de solución se incrementan en más de un orden de magnitud. Para sacar provecho del *cache*, nuestros programas buscan trabajar los datos primeramente en una escala pequeña, por ejemplo trabajar unos cuantos renglones del sistema de ecuaciones a la vez, para tratar realizar estas operaciones dentro del espacio del *cache*.

La escala siguiente es la transmisión de datos entre procesos que se ejecutan en distintos núcleos. Las computadoras que soportan varios procesadores utilizan una organización de memoria Non Uniform Memory Access (NUMA), con la cual cada procesador tiene asignado un banco de memoria. Cuando un proceso que se ejecuta en un procesador trata de leer datos que están en el banco de memoria de otro procesador el tiempo de lectura es mayor que tratar de leer datos del banco de memoria propio. En el caso de que se trate de escribir datos, el sistema tiene que realizar una sincronización de los caches de cada núcleo, lo que implica un tiempo aún mayor. Como se ve en los resultados de la sección anterior, los mejores tiempos se obtuvieron al resolver una partición en cada núcleo (es decir utilizar más particiones), esto disminuye la necesidad de mover datos entre núcleos, procesadores y sus respectivos bancos de memoria.

La escala mayor es la transmisión de datos por la red. Un factor a notar es la latencia o tiempo de espera a una respuesta. Al diseñar nuestro programa para reducir la cantidad de información transmitida no hubo problema con la latencia. En nuestro programa, los resultados entre particiones se intercambian casi al mismo tiempo. El problema es que la cantidad de paquetes de información que puede procesar el *switch* de red a la vez está limitado por la capacidad de memoria de éste. Se trata de una comunicación de casi todos los nodos con todos los nodos. Para problemas con más de 50 particiones el *switch* de red comienza a perder paquetes, lo que ocasionaba que el programa fallará de forma inesperada. La solución fue adquirir un *switch* con más memoria intermedia. De esta manera se pudo trabajar con más de 100 particiones a la vez, sin tener que cambiar la infraestructura de red a fibra óptica o Infiniband, lo que hubiese costado casi lo mismo que todas las computadoras del clúster.

CONCLUSIONES

En la actualidad, los clústers más rápidos (la lista Top500 muestra algunos de éstos [9]) utilizan redes de alta velocidad e infraestructura de refrigeración que, por su alto costo, no es una opción para grupos de investigación pequeños o empresas que quieren adentrarse en el cómputo de alto rendimiento. En este artículo se ha mostrado que es posible con equipos de bajo costo resolver problemas de modelación física de tamaño considerable en tiempos razonables. El uso de herramientas de programación y sistemas operativos con licencia "open source" facilita esta tarea al tener rendimientos comparables a sus contrapartes comerciales.

Para que los grandes clústers (de varias decenas de miles de núcleos) sean rentables, es necesario tener muchos usuarios utilizando los recursos la mayor parte del tiempo. Para un desarrollador o un investigador, el tener que competir por tiempo libre de procesamiento en un clúster quita versatilidad y rapidez de desarrollo. El tener un clúster propio, como El Insurgente, reduce considerablemente los tiempos de desarrollo y ejecución de pruebas.

REFERENCIAS

[1] Botello S., Esqueda H., Gómez F., Moreles M. A. y Oñate E., (2004). *Módulo de Aplicaciones del Método de los Elementos Finitos: MEFI.* Guanajuato México, CIMAT.

[2] Botello S., Moreles M.A. y Oñate E., (2010). *Módulo de Aplicaciones del Método de los Elementos Finitos para resolver la Ecuación de Poisson: MEFI-POISS.* Guanajuato, México, CIMAT.

[3] Hernández Velázquez, H. A., Botello Rionda, S. y Vargas Félix, J.M., (2011). *BAK-Pack: Un paquete para la solución de sistemas lineales con Métodos del Subespacio de Krylov.* ENOAN 2011.

[4] Rocha Quezada, J. J., Vargas Félix, J. M. y Botello Rionda, S., (2011). *Design, installation and operation of a mid-size computer cluster for simulations with the finite element method.* ISUM 2011.

[5] Vargas Félix, J. M., y Botello Rionda, S., (2010). *Parallel Direct Solvers for Finite Element Problems*. Publicación interna CIMAT.

[6] Vargas Félix, J. M., Botello Rionda, S., Hernández Velázquez, H. A. y Valdez, S. I., (2010). *Comparison of different solution strategies for structure deformation using very fine discretization.* NAMIAN2010,

[7] Vargas Félix, J. M. y Botello Rionda, S., (2011). *Comparison of different solution strategies for structure deformation using hybrid OpenMP-MPI methods.* ISUM 2011.

[8] Sterling, T., Becker, D. J., Savarese, D., Dorband, J. E., Ranawake, U. A., Packer, C. V., (1995). *BEOWULF: A Parallel Workstation for Scientific Computation.* Proceedings of the 24th International Conference on Parallel Processing.

[9] http://www.top500.org. Consultado el 1 de agosto de 2011.

[10] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard,* Version 2.1. University of Tennessee, 2008.

[11] Chapman, B., Jost, G., van der Pas, R., (2008). *Using OpenMP: Portable Shared Memory Parallel Programming*. The MIT Press.

[12] Smith, B. F., Bjorstad, P. E., Gropp, W. D., (1996). *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge Univeristy Press.

[13] Botello, S., (2011). *CALSEF 2010: Programa para Cálculo de Sólidos y Estructuras por el Método de los Elementos Finitos.* CIMAT, 2011

[14] Botello, S., (2011). *MEFI Modulo de Aplicaciones del Método de los Elementos Finitos: MEFI.* CIMAT.

[15] Drepper, U., (2007). *What Every Programmer Should Know About Memory.* Red Hat, Inc.