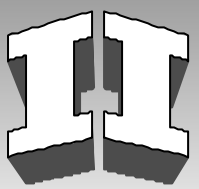

Informática

EL CLUSTER BEOWULF DEL CENTRO NACIONAL DE BIOINFORMÁTICA: DISEÑO, MONTAJE Y EVALUACIÓN PRELIMINAR

Resumen / *Abstract*

La utilización de *cluster* de computadoras en diferentes campos de investigación que requieren cálculos masivos se ha incrementado en los últimos años desde que Becker y Sterling construyeron el primer *cluster* Beowulf en 1994. En este artículo se describe el diseño -desde la selección de los componentes-, montaje y evaluación del *cluster*. Con respecto a los dos primeros aspectos, la explicación se limita a la descripción de la arquitectura de hardware y software del *cluster*. Para la evaluación del desempeño del *cluster* se utilizan varios programas *benchmark* y se comparan los resultados con los de otro *cluster* similar al tratado. Finalmente, se discuten las posibles causas de las diferencias observadas y se propone una vía para mejorar la eficiencia del *cluster*.

Computers' clusters have widely spread in later years within different research fields requiring massive calculations, since Becker and Sterling built the first Beowulf cluster in 1994. In the present paper we describe the design -beginning with component selection-, setting up and evaluation of the cluster. Regarding the two first issues, we limited the paper to the description of hardware and software cluster's architecture. To evaluate the cluster performance, we used several benchmarking programs and compared our results to the ones obtained by other cluster of similar architecture. Finally, we discuss some of the probable causes of the differences observed and propose a way of improving the cluster performance

Palabras clave / *Key words*

Bioinformática, *cluster* de computadoras, computación paralela, MPI, *cluster benchmark*

Bioinformatics, computer's cluster, parallel computing, MPI, cluster benchmarking

***David Gutiérrez Díaz**, Ingeniero en Control Automático,Especialista en Ciencia y Tecnología,Centro Nacional de Bioinformática,Ministerio de Ciencia Tecnología y Medio Ambiente, (CITMA) Ciudad de La Habana, Cuba*
e-mail:david@citma.cu

***Abel González Pérez**, Licenciado en Bioquímica, Aspirante a Investigador, Centro Nacional de Bioinformática, CITMA, Ciudad de La Habana, Cuba*
e-mail:abelito@citma.cu

***Juan Pedro Febles Rodríguez**, Licenciado en Matemática,Doctor en Ciencias Técnicas, Centro Nacional de Bioinformática, CITMA, Ciudad de La Habana, Cuba*
e-mail:febles@citma.cu

Recibido:Junio del 2003
Aprobado: Septiembre del 2003

INTRODUCCIÓN

Cluster Beowulf

Un *cluster* computacional es una agrupación de computadoras interconectadas con dispositivos de red comunes o especializados, que conforman una red privada y que actúan en conjunto usando la capacidad de cómputo de varios CPU (Unidad de Procesamiento Central) y distribuyendo los cálculos en la memoria de todas las máquinas participantes. Con este sistema se puede alcanzar un rendimiento competitivo en relación con equipos paralelos especializados (supercomputadoras) cuyos costos de operación y mantenimiento son elevados.[1-7]

En 1994 en el Centro de la Excelencia en Ciencias de los Datos y de la Información del Espacio (CESDIS), Thomas Sterling y Don Becker crearon el primer *cluster* Beowulf.[3,4]

Es un sistema de bajo costo que normalmente consta de componentes de hardware comunes en el mercado. Consiste en un nodo maestro y dos o más nodos esclavos conectados a través de una red Ethernet u otra tecnología de red. El nodo maestro es la consola del *cluster*, desde el mismo se gestiona y administra el sistema funcionando como pasarela entre el *cluster* y el exterior. Los nodos

esclavos no tienen monitores ni teclados y son accedidos vía remota desde el nodo maestro.

Esta tecnología utiliza, como sistema operativo cualquier distribución Linux, alguna biblioteca de paso de mensajes como PVM (Parallel Virtual Machine) o MPI (Message Passing Interface) y, opcionalmente, aplicaciones especializadas que permiten la gestión, administración y visualización del estado del *cluster*.[1-7]

Utilización de *clusters* de computadoras en Bioinformática

La necesidad de cálculos intensivos ha impulsado la utilización de *clusters* de computadoras en muchas áreas de la Bioinformática. Como en otros campos científicos, la utilización de *clusters* se ha presentado como una variante eficaz frente a la utilización de las supercomputadoras tradicionales. Frente a estas, sin duda ha tenido ventaja en lo que se refiere a la relación calidad-precio.

Las áreas de la Bioinformática en que se ha utilizado la supercomputación van desde la genómica funcional hasta la predicción de estructura tridimensional de proteínas y las simulaciones de dinámica molecular. En todas estas áreas han estado presente los *clusters*. Se les ha utilizado como servidores de bases de datos distribuidas, para computación distribuida y, por supuesto, para computación paralela.[8]

Un ejemplo de la utilización de *clusters* en la Bioinformática es el *cluster* Beowulf del Samuel Lunenfeld Institute de Canadá. Se trata de un *cluster* de 216 procesadores con todo el equipamiento de red de fibra óptica. Está dedicado a un sistema diseñado en el propio Instituto MoBiDick.[9]

En Cuba, hay ejemplos de la utilización de *clusters* en Bioinformática. Sin dudas el más importante es el del Centro de Ingeniería Genética y Biotecnología. Se trata de un *cluster* Beowulf de 128 procesadores, dedicado a numerosas tareas dentro de la investigación bioinformática.

Desde que se concibió la idea de un Centro de Bioinformática en el CITMA se pensó en el montaje de un *cluster* pequeño, cuyas finalidades principales sean el apoyo a otras investigaciones y la investigación sobre computación paralela.

HARDWARE Y MONTAJE

En el proceso de selección del equipamiento para el *cluster* fue muy importante la experiencia de otros grupos que habían pasado por el mismo problema. En especial, el trabajo se beneficia mucho de la experiencia del grupo del *cluster* de la división de Química-Física del Centro de Ingeniería Genética y Biotecnología (CIGB). De hecho, las opciones por las que los autores se decidieronson prácticamente idénticas a las que ellos habían experimentado poco tiempo antes. Dadas las dificultades de la compra del equipamiento requerido, por su complejidad, era importante conocer los resultados que su grupo había obtenido con su solución de hardware específica.

Tomando siempre en consideración la relación calidad-precio y el resultado de los intercambios con el grupo del CIGB, se decidieron varios parámetros importantes del equipamiento a adquirir:

1.Seleccionar un buen equipamiento de redes (*switch* y tarjetas de red).

2. Destinar una máquina con características especiales (de memoria, capacidad de disco duro y velocidad de comunicación por red) a servidor de ficheros del *cluster*.

3. Utilizar tarjetas madre ASUS, en lugar de las comunes en Cuba, AcerOpen.

4. Armar todas las máquinas del *cluster* (incluyendo el nodo maestro y el servidor de ficheros) sin tarjeta de video, teclado ni fdd.

5. Utilizar máquinas duales con procesadores Intel PIII a 933 MHz y debido al consumo energético de los dos procesadores utilizar fuentes de 250 W (como las usadas para los procesadores PIV a 1,8 GHz). En el caso del servidor de ficheros, por el consumo extra de los discos duros, se decidió utilizar una fuente de 350 W

6. Para facilitar la disipación de calor, todas las máquinas fueron equipadas con ventiladores auxiliares y se adquirió un aire acondicionado de 3 t para el local del *cluster*

La observación de estos parámetros condujo a la compra de equipamiento siguiente:

- Un nodo maestro (ZEUS) con:
  - Chasis KF-45 con fuente de 250 W
  - M/B ASUS CUV4X-D
  - 2 Procesadores PIII 933 MHz
  - 1 GB de memoria RAM (2 DIMM de 512 MB)
  - 1 HDD WDC de 120 GB
  - 2 Tarjetas de red 3Com Fast Ethernet
  - DVD-ROM 16X
  - 1 ventilador auxiliar
- Un servidor de ficheros (Atlas) con:
  - Chasis H600A SuperTower con fuente de 350 W
  - M/B ASUS CUV4X-D
  - 2 Procesadores PIII 933 MHz
  - 2 GB de memoria RAM (4 DIMM de 512 MB)
  - 5 HDD WDC de 120 GB
  - 1 Tarjeta ATA RAID IDE 100
  - 1 Tarjeta de red 3Com Fast Ethernet
  - 1 Tarjeta de red 3Com 1000 Mb/s
  - 3 ventiladores auxiliares (más uno alimentado por el M/B)
- Ocho nodos esclavos (Afrodita, Apolo, Ares, Artemisa, Atenea, Hera, Hermes, Poseidón) con:
  - Chasis KF-45 con fuente de 250 W
  - M/B ASUS CUV4X-D
  - 2 Procesadores PIII 933 MHz
  - 512 MB de memoria RAM (1 DIMM de 512 MB)
  - 1 HDD Maxtor 40 GB
  - 1 Tarjeta de red 3Com Fast Ethernet
  - 1 ventilador auxiliar
- Equipamiento de red:
  - Un switch 3Com SuperStack III 3300MM (24 puertos 100 Mb/s)
  - Un switch 3Com SuperStack III 3300SM (24 puertos 100 Mb/s con un puerto de f.o. 1000 Mb/s)
  - Router Layer 3
  - Cable de f.o. de 3 m
  - Cable UTP categoría 5

SOFTWARE E INSTALACIÓN

Instalación de Linux en el *cluster*

En todos los nodos se instaló la distribución Linux Red Hat 7.2. Esta versión contiene el Kernel 2.4.7-10 del sistema, del cual se escogió la variante smp (simmetric multiprocessing) para aprovechar la posibilidad de compartir la memoria que brindan las máquinas dual.

MPI, instalación, configuración y puesta a punto

Una de las bibliotecas de funciones utilizadas para la programación paralela es el MPI (Message Passing Interface). Esta biblioteca fue desarrollada (para C y Fortrán) por el grupo de trabajo del MPI, que incluye a unas 60 personas de alrededor de 40 instituciones. El primer borrador del estándar MPI1 se presentó en 1992. La primera implementación estuvo lista en 1994. Desde ese momento ha seguido desarrollándose y ya se ha presentado un nuevo estándar MPI2 que incluye nuevas características como I/O paralela, aunque ninguna implementación contiene todas las características nuevas de este estándar.[5,10,12]

En la actualidad, el paquete del MPI contiene no solo una biblioteca de funciones para el paso de mensajes, sino todo un entorno de programación que facilita la compilación y ejecución de programas paralelos. Las distribuciones de Linux como RedHat 7.2 y SuSe 8.0 poseen implementaciones del MPI (LAM en el caso de RedHat y LAM y MPICH en el de SuSe). Los paquetes pueden descargarse como tar.gz desde www-unix.mcs.anl.gov/mpi/. Desde ese mismo sitio puede descargarse el manual de instalación.

Lo primero en la configuración del paquete es el dispositivo. Los dispositivos son diferentes configuraciones del paquete que toman en cuenta la arquitectura de la máquina paralela. En este caso, por tratarse de un *cluster* homogéneo de nodos SMP, se escogió el dispositivo ch_p4. Por esa misma razón, al configurar se seleccionó la opción with-comm=shared, que permite la existencia de la memoria compartida en los nodos y al editar el fichero *machines*. Linux que contiene la lista de las máquinas que forman parte del cluster se declararon todos los nodos como dual.

Una de las facilidades del entorno de programación que proporciona el MPI es la existencia de *bash scripts* de compilación y ejecución de programas que ahorran mucho trabajo, sobre todo en el proceso de compilación. El mpicc, por ejemplo, que es el *script* de compilación de programas de C crea los objetos a partir de los ficheros de código fuente y enlaza con las librerías de C o Fortrán en la ruta en que están instaladas (en este caso /usr/lib y /usr/share/include) y con las librerías de mpi que se encuentran en $HOME/mpich-1.2.4/lib y $HOME/mpich-1.2.4/share/include sin necesidad de configurar una variable de entorno que guarde la arquitectura del MPI.

Una vez configurado y compilado el MPICH en ZEUS se editó el fichero *machines*. Linux que guarda las máquinas accesibles al *cluster*

Otra de las facilidades del paquete de MPI son los programas de *benchmark* que tiene incorporados, que permiten explorar la capacidad de procesamiento de la computadora paralela y probar la eficiencia de las funciones del MPI en la máquina paralela. Para ello solo es necesario compilarlos, copiarlos en los nodos esclavos en la misma ruta en que se encuentran en el nodo maestro y ejecutarlos lanzando el *script* mpirun. Este *script* permite definir el número de nodos en que se quiere correr el programa, los nombre de los nodos e incluso si se quiere ejecutar utilizando los dos procesadores de un nodo o no.

RESULTADOS Y DISCUSIÓN DE LA EVALUACIÓN DEL CLUSTER

Cálculo de Pi

El algoritmo del cálculo de pi determina la integral definida entre 0 y 1 de la función f(x)=4/(1+x2). El resultado de esa integral es, obviamente, pi. El algoritmo en serie recibe como entrada un número de intervalos de integración (número de rectángulos en los que dividir el área de integración) y calcula el área de los intervalos como el producto del valor de la función en el punto medio de cada rectángulo y la base del rectángulo que, por supuesto, es igual al inverso del número de intervalos.

El algoritmo en paralelo, divide el cálculo entre un número de procesos definido en el mpirun y pasado al programa a través de la función MPI_Comm_Size. Cada proceso calcula un área parcial y la devuelve al proceso maestro que las suma y devuelve el resultado final al usuario. La implementación del algoritmo utiliza solo dos funciones de comunicación del MPI: MPI_Broadcast que pasa el valor del número de intervalos de integración de una localización de memoria en el nodo maestro a la misma localización de memoria en los nodos esclavos y MPI_Reduce (SUM) que toma los valores de las áreas parciales almacenados en la misma localización de memoria en todos los nodos, los suma y deposita el resultado en una localización de memoria definida del nodo maestro.

Antes de presentar y discutir los resultados de la evaluación del *cluster* con el cálculo de pi es necesario hacer énfasis en el hecho de que todo algoritmo de cálculo implementado en paralelo va a consumir tiempo por dos causas: el cálculo y la comunicación entre los procesos. Podría enunciarse que:

$$t_{\text{total}} = t_{\text{cálculo}} + t_{\text{comunicación}}$$

Aunque, en realidad, hay otros procesos que también consumen tiempo computacional.

En esta simple suma se refleja uno de los principios de la programación paralela: un algoritmo podrá ser paralelizado eficientemente solo si el tiempo que toman los cálculos es superior al tiempo de comunicación que emplearán los procesos. Solo en ese caso la ganancia de velocidad en el cálculo influirá en la disminución del tiempo total. Si el orden del tiempo de cálculo y el tiempo de comunicación son semejantes, la ganancia en tiempo de cálculo no repercutirá positivamente en el tiempo computacional total, porque la comunicación será el factor determinante del orden del tiempo total.[2,7]

En la figura 1 se presenta el resultado de la ejecución del programa que implementa el cálculo de pi en paralelo en un número creciente de procesadores y con diferente número de intervalos de integración. En el caso de 1 000 intervalos de integración, el

aumento en el número de procesadores no resulta en una disminución del tiempo de computación. En ese caso, el orden del tiempo de comunicación es similar al del tiempo de cálculo. (Esto se comprobó ejecutando un programa de *benchmark* que prueba el ancho de banda del *cluster* para las funciones de MPI utilizadas por el programa de pi o sea, MPI_Broadcast y MPI_Reduce, obsérvese la figura 2.)[10-12]

Otro análisis interesante que se deriva del gráfico es cómo la pendiente de las rectas ficticias de incremento de la velocidad del *cluster* al aumentar el número de procesadores se va acercando a la pendiente ideal 1 a medida que aumenta el número de intervalos de integración y, por tanto, se hace más complejo el cálculo. La pendiente de estas rectas representa un análogo de la aceleración que experimentan los cuerpos al actuar sobre ellos una fuerza. En este caso, se trata del aumento en la velocidad del cálculo al añadir nuevos procesadores a la tarea. Con esta analogía, muchas veces se llama **aceleración del *cluster*** a esta variable, aunque no siempre se determina de esta manera.[2, 7]

Otro fenómeno interesante se observa en la curva de n = 10000000 en la cual la adición de un procesador de orden impar (que implica la incorporación al cálculo de una nueva máquina) reporta menos ganancia en velocidad que la incorporación de uno de orden par.

Benchmark HPL

El HPL es una implementación gratis y de código abierto de *Benchmark Linpack* para computación de alto desempeño. Este *benchmark* resuelve un sistema de ecuaciones lineales con números de doble precisión (64 bits). Es el *benchmark* más usado para determinar el desempeño real de las supercomputadoras por lo que se le utiliza para ordenar a las computadoras más potentes del mundo en la lista de las cien computadoras más rápidas. Debido a su popularidad, se dispone de puntos de comparación bastante parecidos a cualquier sistema paralelo.[13]

En el caso del *cluster* tratado se obtuvo un máximo desempeño para una matriz de tamaño 2000 x 2000 (tabla 1).

El desempeño del *Cluster*1 supera al cluster en cuestión en casi diez veces. La razón para está diferencia se encuentra en el equipamiento de la conexión de red. En el caso del *Cluster* 1 se utilizó la tecnología Myrinet, que permite alcanzar anchos de banda de alrededor de 1 Gb/s, mientras que en el otro *cluster* utiliza la tecnología corriente Fast Ethernet, con anchos de banda diez veces menores, pero mucho más económica.

Otro detalle tiende a apoyar este razonamiento. Por ejemplo los diseñadores del HPL sugieren que el tamaño de matriz óptimo para probar el desempeño de un *cluster* se acerca a aquel que ocupe toda la memoria RAM disponible de los nodos.

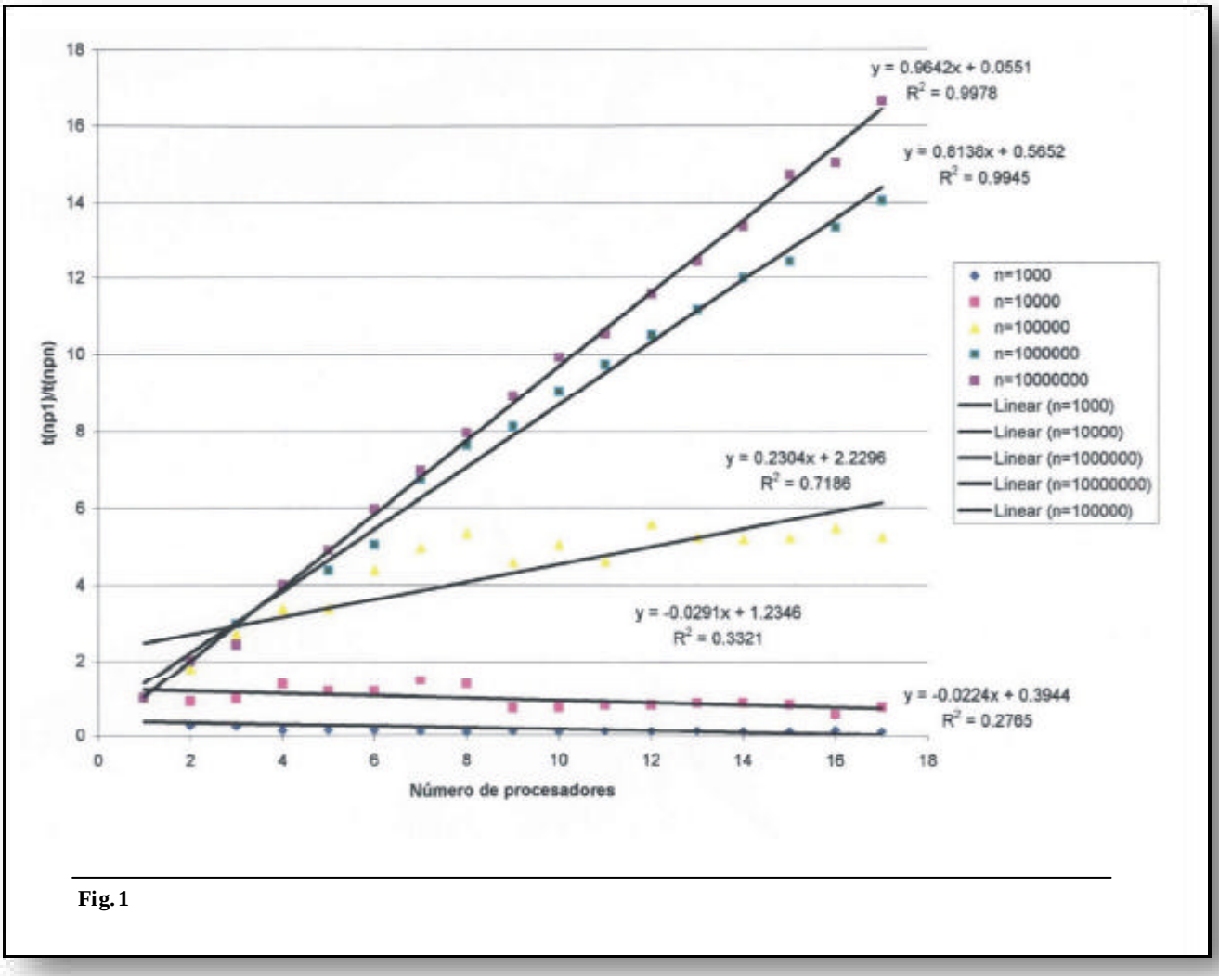

Fig. 1

En el caso del *Cluster* 1, este valor se encuentra alrededor de 20000, teniendo en cuenta que el sistema ocupa alrededor de 50 MB de memoria. En la tabla 1 se observa que el desempeño de *Cluster* 1 crece sin interrupción al aumentar el tamaño de matriz en los valores ensayados. En el *cluster* estudiado el valor óptimo de matriz es del mismo orden (la memoria de cada nodo y el número de nodos son idénticos), sin embargo, el máximo desempeño se alcanza con una matriz mucho más pequeña (2000 X 2000). (Además, tomando en cuenta los primeros resultados y experiencias anteriores,[1,13] se habían eliminado en los nodos esclavos todos los servicios del sistema operativo que consumían parte de la memoria de los nodos, reduciéndose en total la memoria ocupada por el sistema a menos de 50 MB.) Una inspección de la memoria mientras se ejecuta el programa con tamaños de matriz superiores a 2000 X 2000 (ejecutando un comando como top) revela que inclusive matrices tan grandes como 15000 X 15000 no ocupan toda la memoria disponible en los nodos. Por tanto, la única razón que puede explicar la disminución del desempeño es la limitación de la comunicación entre los nodos.

A tenor con esta conclusión, se decidió medir el ancho de banda real en la red del *cluster*, para corroborar que este se encontrase en los valores considerados normales en una red Fast Ethernet. Se utilizó uno de los programas que se encuentran dentro de la distribución del MPI, el mpptest, que mide el ancho de banda del *cluster* utilizando las funciones de comunicación del MPI. Este programa se ejecuta con el comando mpirun, como cualquier programa que incorpore las funciones de la biblioteca del MPI y mide el tiempo que le toma a paquetes de diferentes tamaños cumplir el viaje de ida y vuelta entre dos o más máquinas, en dependencia de la función MPI que se esté probando.

El programa, por supuesto, acepta como argumento el nombre de la función que se quiera probar. En el caso del *cluster* estudiado, se probó la función MPI_broadcast, que recibe una variable colocada en un sitio de la memoria buffer del proceso maestro y la envía a todos los procesos esclavos. Esta función es la más utilizada en el algoritmo del cálculo de pi descrito en el epígrafe anterior y en el HPL.

TABLA 1
Resultados de la ejecución del *benchmark* hpl para diferentes tamaños de matriz y diferentes divisiones de la matriz

Desempeño (Gflop/s)	División matriz (PxQ)	Tamaño de matriz	2000	5000	10000	15000
Cluster 1	2 x 4		1,76	2,32	2,58	2,72
	4 x 4		2,27	3,94	4,68	5
Cluster estudiado	4 x 4		0,31	0,42	0,4	0,37
	2 x 8		0.43	0,54	0,53	0,52

Nota:
Esto sucede en un *cluster* de 8 nodos esclavos dual PIII a 550 Mhz, 512 MB de memoria RAM e interconectados por Myrinet (*Cluster* 1) y en el *cluster* estudiado. El desempeño de cada *cluster* en cada una de las pruebas se presenta como miles de millones de operaciones flotantes realizadas por segundo (Gflops). La columna División matriz, especifica el número de submatrices en que se divide la matriz y es igual al número de procesos esclavos. P reprenta la cantidad de divisiones verticales y Q la cantidad de divisiones horizontales.

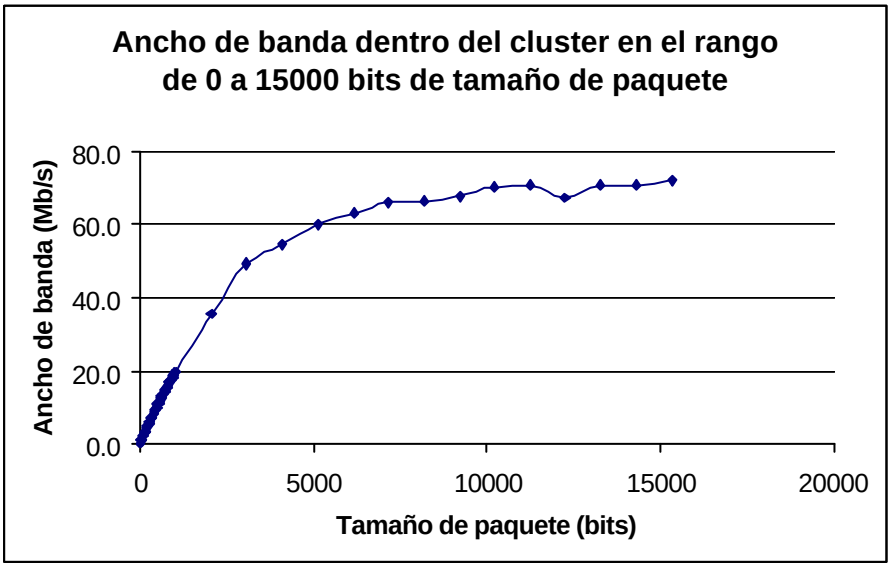

Fig. 2 *Ancho de banda de la red del cluster vs tamaño de paquete empleado por el mpptest (explicación en el texto). La saturación ocurre a un ancho de banda de aproximadamente 70 Mbits/s.*

Como pudo observarse en la figura 2, el ancho de banda máximo que se alcanza en el *cluster* (para paquetes de aproximadamente 1 Mbit) se corresponde con lo encontrado en la generalidad de las redes Fast Ethernet y, específicamente en *clusters* de computadoras,[2] o sea, unos 70Mbits/s. Como corolario de este resultado, se tiene que la comunicación de red es el factor que limita de modo más drástico el desempeño del *cluster* estudiado. De hecho, si se toma en cuenta que el valor teórico de este desempeño se encuentra alrededor de 14.9 Gflop/s (que corresponde al caso ideal en que los 16 procesadores PIII a 933MHz se encontrarán en un solo Motherboard) y el valor real máximo apenas sobrepasa 0.5 Gflop/s, es necesario rendirse a la evidencia de que el desempeño de esta computadora paralela solo alcanza un 3,4 % del que tendría una supercomputadora con 16 procesadores de este tipo y velocidad y, aproximadamente, un 10 % de la computadora presentada en la tabla 1 con el nombre de *Cluster* 1. Por supuesto, esta conclusión conduce a una discusión sobre el tema desempeño/precio y sobre el límite del rendimiento de los *cluster* de computadoras y su relación con el equipamiento existente y por existir. Esta discusión es el objeto de un artículo más profundo que se encuentra en proceso de elaboración en este momento. De hecho, se está experimentando con la tecnología de Channel Bonding (sin traducción satisfactoria al español) que mediante el uso de dos tarjetas de red en los nodos debe permitir ampliar el ancho de banda del *cluster* al doble de lo presentado en la figura 2.

CONCLUSIONES

• El *cluster* del Centro Nacional de Bioinformática realiza cálculos sencillos en paralelo utilizando la biblioteca de funciones paralelas del MPI. En el ejemplo del cálculo de pi, al aumentar el número de intervalos de integración (o sea, la complejidad del problema) la aceleración del *cluster* (el aumento de la velocidad al añadir nuevos procesadores al cálculo) se acerca al valor máximo teórico de 1.

• La resolución de un sistema de ecuaciones lineales mediante el *benchmark* HPL muestra que el desempeño real del *cluster* sobrepasa ligeramente los 0.5 Gflop/s, o sea, un 3.4 % del valor esperado teóricamente. La limitación para el crecimiento de este valor es la comunicación entre procesos a través de la red del *cluster*. La comunicación impone un límite a la reducción que se puede alcanzar en el tiempo de resolución de un problema al paralelizar los cálculos en razón de la expresión simplificada

$$t_{total} = t_{cálculo} + t_{comunicación}.$$

• El ancho de banda de la red del *cluster* tiene el valor máximo que se puede alcanzar con el equipamiento de red de que dispone el *cluster*. La única manera, entonces, de mejorar el ancho de banda de la red y con él el desempeño del *cluster* es modificar este equipamiento. Una recomendación directa de estos resultados es, por tanto, la aplicación de la tecnología de Channel Bonding que, sin un incremento significativo de los costos, permitiría incrementa el ancho de banda al doble del valor actual.

REFERENCIAS

1. **ADAMS, J. AND D. VOS:** *Small-College Supercomputing Building aBeowulf Cluster at a Comprehensive College* in: www.calvin.edu/~adams/html/interests/professional/publications/SmallCollegeSupercomputing.pdf, 2001.
2. **ANDERSSON, K. J.; D. ARONSSON AND P. KARLSSON:** *An Evaluation of the System Performance of a Beowul Cluster,* Department of Scientific Computing, Universidad de Upsala, Informe Interno, 2001.
3. **BECKER, D.** *et al*: "Beowulf, A Parallel Workstation fo Scientific Computation" in *Proceedings, International Conference on Parallel Processing,* Oconomowoc Wisconsin, 1995.
4. *Beowulf: Introduction and Overview,* CESDIS er www.beowulf.org/intro.html, 2000.
5. **GEIST, G. A. J. A. KOHL AND P. M. PAPADOPOULOS** *PVM and MPI: a Comparison of Features* in www.csm.ornl.gov/pvm/PVMvsMPI. ps, 1996.
6. **LINDH, B.:** *Sun Based Beowulf Cluster* in Sun BluePrints OnLine: www.sun.com/blueprints, 2002.
7. **MCGARVEY, B.** *et al.*: *Beowulf Cluster Design fo Scientific PDE Models* in www.athena-em.gatech.edu Beowulf/index.html, 2001.
8. **PHILLIPS, J. C.; G. ZHENG; S. KUMAR AND L. V. KALÉ:** *NAMD: Biomolecular Simulation on Thousands o Processors* in , 2002.
9. **MICHALICKOVA, K.; M. DHARSEE AND C.W.V .HOGUE,:** *Sequence Analysis on a 216-Processor Beowul Cluster* in www.usenix.org/publications/library/proceedings/als2000/full_papers/michalickova/michalickova.ps, 2000.
10. **GROPP, W. AND E. LUSK:** *Installation and User's Guide to MPICH, a Portable Implementation of MPI. Versio 1.2.4*, Technical Report CS-94-230, Argonne Nationa Laboratory, 2001.
11. **GROPP, W.; E. LUSK; N. DOSS AND ANTHONY SKJELLUM:** *A High-Performance, Pportable Implementation of the MPI Message-Passing Interface standard* in Parallel Computing, Vol. 22(6), pp. 789-828, 1996.
12. Message Passing Interface Forum: "MPI: A Message Passing Interface Standard en International" *Journal o Supercomputer Applications*, Vol. 8 (3- 4): pp. 165-414 1994.
13. **PETITET, A.; R. C. WHALEY; J. DONGARRA AND A CLEARY:** *HPL - A Portable Implementation of the High Performance Linpack Benchmark for Distributed-Memory Computers* in www.netlib.org/benchmark/hpl/, 2000.