Paralelización de la Factorización LU de Matrices en Clusters Heterogéneos

Fernando G. Tinetti[*], Mónica Denham[†]

Instituto de Investigación en Informática LIDI [‡]
Facultad de Informática. Universidad Nacional de La Plata.

Resumen

Este trabajo presenta las ideas básicas de paralelización de la factorización LU de matrices en clusters de computadoras interconectadas por redes Ethernet y las adaptaciones necesarias cuando las computadoras son heterogéneas. Inicialmente se describen las principales características de la arquitectura paralela, que se tienen en cuenta para la paralelización del algoritmo. Luego se describe el problema a resolver, dando una solución secuencial por bloques. A continuación se describe el algoritmo paralelo propuesto, presentándose primero cómo se distribuyen los datos entre las máquinas utilizadas y luego se muestra un pseudocódigo del mismo.

Se describe la experimentación realizada con el objetivo de identificar el rendimiento obtenido y/o los posibles problemas de rendimiento en un cluster heterogéneo. Luego se muestran los resultados obtenidos, siendo éstos los tiempos tomados en las distintas ejecuciones y se hace un análisis de los mismos. Por último se describen las conclusiones y se presentan los pasos posibles pasos futuros.

Palabras Clave: Cómputo en Clusters, Clusters Heterogéneos, Problemas de Algebra Lineal en Paralelo, Algoritmos Paralelos, Rendimiento de Cómputo y Comunicaciones.

1 Introducción

Entre los principales objetivos del cómputo paralelo se incluye sin lugar a dudas la obtención del mayor rendimiento posible en la resolución de problemas. Sobre este objetivo se ha trabajado arduamente, obteniendo cada vez soluciones más precisas y ventajosas para cada problema, pero también más orientadas a una arquitectura en particular.

Resolver un problema desde el punto de vista de la arquitectura a utilizar tiene la ventaja de aprovechar algorítmicamente todas las características presentes en dicha arquitectura, por lo que se obtienen soluciones más eficientes. Sin embargo, también hay desventajas dado

[*]Prof. Adjunto DE. III-LIDI. Facultad de Informática. UNLP. e-mail: fernando@lidi.info.unlp.edu.ar
[†]Auxiliar Docente. III-LIDI. Facultad de Informática. UNLP. e-mail: mdenham@lidi.info.unlp.edu.ar
[‡]III-LIDI miembro del Instituto de Investigación en Ciencia y Tecnología Informática (IICyTI). Facultad de Informática. UNLP. Calle 50 y 115 1er Piso, (19000) La Plata, Argentina. TE/Fax +(54) (221) 422-7707. http://lidi.info.unlp.edu.ar

que un algoritmo orientado a una arquitectura en particular que haga uso optimizado de sus características, también lleva aparejada la posibilidad de no tener el mismo rendimiento (o rendimiento *aceptable*) en otras arquitecturas.

Los clusters de computadoras han probado ser las plataformas de cómputo paralelo más ventajosas en términos de la relación costo-rendimiento [3] [4]. Aunque la tendencia hasta el momento ha sido la utilización de clusters homogéneos (con computadoras idénticas) para cómputo paralelo, los clusters heterogéneos (con computadoras diferentes entre sí) tienen una gran importancia por varias razones [13] [14]:

- Utilización de redes locales para cómputo paralelo. Las redes locales, aunque sin disponibilidad completa, se pueden considerar como plataformas de cómputo paralelo de costo cero en hardware. Las redes locales no solamente ya han sido instaladas y están funcionando, sino que además se mantienen independientemente del cómputo paralelo dado que tienen sus usuarios y administradores *están*

- A medida que pasa el tiempo es cada vez más difícil conseguir *exactamente* los mismos componentes de las computadoras de escritorio de bajo costo, básicamente PCs. Es así que aunque se haya instalado un cluster homogéneo para cómputo paralelo normalmente a partir de los seis meses posteriores es cada vez más difícil conseguir, por ejemplo, el mismo tipo de memoria o los mismos procesadores y la dificultad es mayor a medida que transcurre más tiempo. En este punto, si se quiere mantener la capacidad de cómputo de un cluster y alguna/s computadora/s falla/n se pasará a un cluster heterogéneo o se debería instalar un cluster nuevo con la cantidad de computadoras idénticas que se necesiten.

- Un caso similar al anterior pero no necesariamente el mismo se da cuando se necesitan resolver aplicaciones *mayores* en términos de potencia de cálculo necesaria o en términos de capacidad de almacenamiento. Una vez más, si se necesita una mayor cantidad de computadoras y no se consiguen *exactamente* las mismas que ya tiene un cluster instalado se debe pasar a un cluster heterogéneo o comprar otro cluster *completo* (en cuanto a que se compran todas las máquinas necesarias y no solamente las que se necesitan sumar al cluster existente).

Por otro lado, existe un conjunto de problemas que por naturaleza son altamentente paralelizables, y que son muy necesarios, por lo que su optimización es muy importante. Una clase de estos problemas son, en general, las operaciones y los métodos provenientes del área de álgebra lineal, y se utilizan en aplicaciones del ámbito de la medicina, astronomía, física, simulaciones y otros. Desde hace varios años, existe una biblioteca de funciones definidas para estas aplicaciones, denominada LAPACK (Linear Algebra PACKage) [1] [2].

A un subconjunto de las operaciones más básicas de álgebra lineal se las conoce como BLAS (Basic Linear Algebra Subroutines) [5] [11][7], y la optimización de este conjunto de operaciones es un objetivo planteado por muchos investigadores. Los resultados de dichos esfuerzos se reflejan en los numerosos trabajos y avances que hay sobre la optimización de dichas rutinas.

Dentro de BLAS se definen distintos niveles, de acuerdo al tipo de operandos que participan en cada una de ellas y por la determinación de los requerimientos de cómputo y memoria. Los niveles definidos son: BLAS Level 1 (operaciones cuyos operandos son vectores), BLAS Level 2 (operaciones entre vectores y matrices) y BLAS Level 3 (donde ambos operandos son matrices).

El gran esfuerzo de optimización recae sobre operaciones de BLAS Level 3 dado que es el nivel que mayor requerimiento de cómputo y por lo tanto su optimización es la más útil y ventajosa. Ejemplos de operaciones de este nivel son: multiplicación de matrices, soluciones a sistemas de ecuaciones *triangulares*, etc.

2 Arquitectura de los Clusters Heterogéneos

Desde el punto de vista de procesamiento paralelo, este trabajo sigue la idea tradicional de utilización de una máquina paralela en particular. Más específicamente, los clusters considerados como máquinas paralelas pertenecen a las arquitecturas MIMD (Multiple Instruction - Multiple Data stream) dentro de la clasificación de Flynn [8] [9], y está formada por múltiples nodos interconectados por medio de una red Ethernet. Cada uno de los nodos es una estación de trabajo o una PC. En definitiva, se usa una red local como máquina paralela.

Cada computadora de escritorio interconectada por la red puede ser de cualquier tipo y con cualquier capacidad de cómputo, no necesariamente idéntica a la de las demás del cluster (esto la hace una arquitectura heterogénea). Cada máquina necesita una placa de red para conectarse a la red Ethernet, y el cableado de la red puede incluir *switches* y/o *hubs*. El uso de esta arquitectura como máquina paralela, tiene ventajas y desventajas.

La principal ventaja de la utilización de clusters heterogéneos como máquinas paralelas es su excelente relación costo-rendimiento que ya se ha mencionado. Sin embargo, se tienen varias desventajas:

- Tanto las máquinas como la red de interconexión no fueron creadas originalmente para ser usadas como máquinas paralelas. Solamente como ejemplo, no se tienen herramientas de desarrollo y depuración de programas paralelos estándares para cómputo paralelo en clusters. Esta falta de estandarización hace que coexistan múltiples herramientas (algunas comerciales y otras de uso libre) para la/s misma/s tarea/s.

- La red de interconexión es lenta comparada con una red de interconexión de una máquina paralela. Esto es una gran desventaja, pues la comunicación entre procesos es uno de los factores que penalizan el cómputo paralelo.

- A menos que la red Ethernet tenga *switching* completo (un único switch que interconecta a todas las computadoras), el rendimiento de las comunicaciones es muy dependiente del tráfico de datos en general (en toda la red). Se debe recordar que si bien existe la posibilidad de *switching* completo para una gran cantidad de computadoras, el costo de *switching* completo crece más que linealmente con la cantidad de computadoras conectadas.

- La posibilidad de heterogeneidad en cuanto a capacidad de cómputo en particular complica notablemente las aplicaciones paralelas desde el punto de vista del balance de carga. En el ámbito de las aplicaciones de álgebra lineal, donde el procesamiento de datos es muy regular y predecible, es muy sencillo lograr balance de carga en procesadores homogéneos, y de hecho todos los algoritmos paralelos desarrollados tienen este problema resuelto de manera muy sencilla. Sin embargo, no son utilizables de manera directa en los ambientes heterogéneos, donde se deben tomar recaudos especiales para mantener el balance de carga de cómputo de los diferentes procesadores (computadoras).

Por lo tanto, en el momento de desarrollar un algoritmo, es muy importante tener en cuenta todas las características de este tipo de arquitectura, en particular el tipo de red de interconexión

y el tipo de máquinas que componen la arquitectura, con el objetivo de poder sacar el mayor provecho posible y evitar penalizaciones de rendimiento.

En particular, las redes Ethernet utilizan el protocolo 802.3, donde las máquinas están conectadas a un único medio físico, al cual se accede por CSMA/CD (Carrier Sense, Multiple Access/Collision Detect) [10]. Cuando se transmite un mensaje, éste ocupa el único medio de transmisión y las máquinas conectadas a la red compiten por él. Uno de los mensajes más naturales a utilizar en las redes Ethernet es el *broadcast*, en el cual una computadora envía un mensaje y éste es transmitido por toda la red, llegando a todas las PCs conectadas. Se tiene un único emisor y múltiples receptores a nivel físico, lo cual se puede aprovechar satisfactoriamente desde el punto de vista del rendimiento y de la escalabilidad de los programas paralelos.

Por otro lado, las máquinas que conforman la máquina paralela normalmente son PCs comunes, cuyas capacidades de cómputo y almacenamiento pueden variar, teniendo así heterogeneidad de cómputo. Además, cada una de las máquinas tiene su propia jerarquía de memoria, dada por sus distintos tipos de memoria instaladas (caches, memoria principal). Dejar de lado estas características también implica normalmente pérdida de rendimiento de las aplicaciones paralelas.

Habiendo analizado el dominio de aplicación y las principales características de la arquitectura a utilizar, los objetivos son:

- Desarrollar algoritmos que implementen soluciones a problemas de álgebra lineal dada la gran importancia que tendría obtener optimizaciones de éstas.
- Aprovechar las redes locales instaladas para cómputo paralelo, dada su amplia disponibilidad y costo mínimos de hardware, instalación y mantenimiento.

Por lo tanto, se deben desarrollar algoritmos orientados a estas arquitecturas, teniendo en cuenta la heterogeneidad de cómputo y almacenamiento y la capacidad de broadcast físico que presentan. En particular, en este trabajo se presentan las características más importantes de la factorización LU de matrices, la cual fue desarrollada teniendo en cuenta lo explicado antes.

3 Factorización LU por Bloques

La operación implementada es la factorización LU, la cual pertenece a LAPACK y sirve para resolver sistemas de ecuaciones lineales. En particular, la factorización LU convierte un sistema del tipo:

$$Ax = b \tag{1}$$

en dos sistemas triangulares equivalentes para encontrar los valores de las incógnitas representadas por x en la Ec. (1). El método para solucionar el problema problema consiste en encontrar las matrices triangulares L y U tales que:

$$A = LU \tag{2}$$

Donde L es una matriz triangular inferior (con 1s en la diagonal principal) y U es una matriz triangular superior. Entonces, a partir de la Ec. (1) y la Ec. (2)

$$LUx = b \tag{3}$$
$$Ux = z \tag{4}$$

donde $z = L^{-1}b$, en toncesse puede determinar z:

$$Lz = b \tag{5}$$

y a continuación se resuelve

$$Ux = z \tag{6}$$

obteni éndose x.

Las matrices triangulares L y U se obtienen aplicando pasos de eliminación Gaussiana sobre los datos de A. Una forma muy común de resolver operaciones de álgebra lineal, tambi én aplicable a la factorización LU, se basa en la partición por bloques [7]. De esta forma, la matriz a factorizar se divide en bloques para aprovechar los niveles de memoria cache provistos por las computadoras. Antes de explicar el algoritmo paralelo propuesto, se describe brevemente la factorización secuencial por bloques, en el cual se basa la soluci´onpropuesta. La matriz a factorizar se divide entonces de la forma que se muestra la Fig. 1. P orlo tanto, se deben

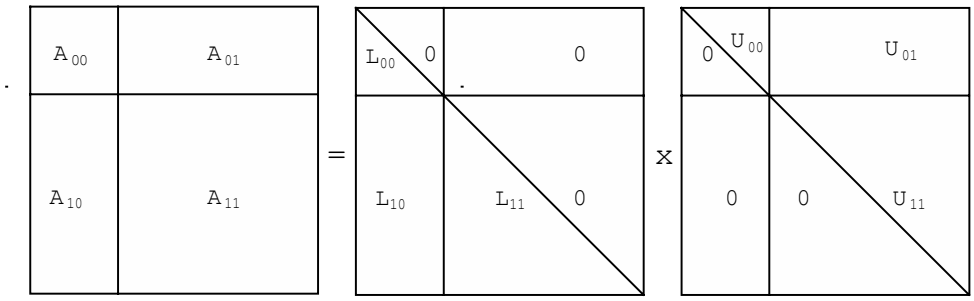

Figura 1: P artición de las Matrices.

resolver

$$A_{00} = L_{00}U_{00} \tag{7}$$
$$A_{01} = L_{00}U_{01} \tag{8}$$
$$A_{10} = L_{10}U_{00} \tag{9}$$
$$A_{11} = L_{10}U_{01} + L_{11}U_{11} \tag{10}$$

De acuerdo con la definición de multiplicación de matrices, ahora el problema consiste en encontrar los bloques L_{00}, L_{10}, L_{11} y U_{00}, U_{01} y U_{11} de la Fig. 1. Primero, se aplica LU de forma directa a A_{00}, obteniendo L_{00} y U_{00}, resolvi éndose así la Ec. (7). Luego, dado que L_{00} es una matriz triangular inferior, se puede aplicar el m étodo de resolución de sistema de ecuaciones triangulares, y se resuelve la Ec. (8), obteniéndose U_{01}. De la misma forma, y usando la Ec. (9) se obtiene L_{10} (U_{00} es una matriz triangular superior). P or ´ultimo, para obtener las submatrices L_{11} y U_{11} se usa la Ec. (10), la cual se puede transcribir como:

$$L_{11}U_{11} = A_{11} - L_{10}U_{01} \tag{11}$$

Y por lo tanto, encontrar las submatrices L_{11} y U_{11} *correspondientes* a la submatriz A_{11} se "reduce" a aplicar el mismo m étodo a lasubmatriz A_{11} *actualizada*, es decir a $A_{11} - L_{11}U_{11}$.

En este punto se puede v er que la resolución de la factorización LU se basa en distintas operaciones: eliminación gaussiana (LU), resolución de sistemas de ecuaciones triangulares, multiplicación y resta de matrices, etc. P ara obtenermay or estabilidad num érica en el c´alculo de las matrices L y U normalmente se utiliza una técnica de pivoteo, en la cual antes de

factorizar cada dato, se busca el elemento de mayor valor absoluto en toda la fila y se permutan las columnas respectivas. En el algoritmo paralelo se incluye esta técnica a pesar de que normalmente implica penalización de rendimiento, pero se considera esencial para mantener estabilidad numérica como en el método *original* (secuencial).

4 Algoritmo Paralelo *para* Clusters Heterogéneos

Para describir el algoritmo, primero se explicará cómo se distribuyen los datos entre las computadoras participantes y luego se darán detalles sobre el código a ejecutar en cada una de las computadoras que participa en la factorización de la matriz.

El algoritmo utilizado se basa en la solución por bloques del problema. Se tiene en cuenta, además, las diferencias de cómputo de las distintas máquinas que componen un cluster heterogéneo, como así también las características de la red de interconexión del mismo. El algoritmo propuesto trata de obtener el mayor beneficio de estas características como así también evitar las penalizaciones que éstas podrían ocasionar.

Teniendo en cuenta el primer punto, o sea, que los nodos que componen el cluster podrían tener distintas capacidades de procesamiento, se distribuirán los datos de forma tal que los nodos con mayor capacidad obtengan más bloques a procesar que las de menor capacidad. De esta forma, se evita que el tiempo total de cómputo esté determinado por el tiempo que necesitan para procesar las computadoras más lentas.

El primer problema que surge es el de definir la capacidad de cómputo de cada nodo que compone el cluster y del cluster como máquina paralela. Se puede resolver este problema de varias formas, y se ha elegido la más sencilla:

1. Se determina la capacidad de cada máquina, en términos de Mflop/s (millones de operaciones de punto flotante por segundo). Esta ya es una medida de capacidad relativa, que es útil para la comparación de potencias de cálculo de cada computadora en el ámbito de las aplicaciones numéricas.

2. Se determina la capacidad total de la máquina paralela como la suma de las capacidades de cada una de las computadoras del cluster.

Con estas definiciones se puede saber cuánto trabajo del trabajo total puede (o debe) realizar cada nodo dado que se tiene de manera inmediata la capacidad de cálculo de cada máquina con respecto a la capacidad de cálculo de la máquina paralela. La capacidad de cada máquina en términos de Mflop/s puede obtenerse directamente por experimentación sobre un problema similar (multiplicación de matrices, por ejemplo) o del mismo problema (factorización LU), en ambos casos tomando como referencia un tamaño reducido de matrices.

El caso más sencillo con respecto a los casos de velocidades relativas es el que se tiene con dos computadoras P_0 y P_1 tales que la capacidad de cómputo de P_0 es el doble de la capacidad de P_1. En este caso, queda claro que P_0 debería computar 2/3 y P_1 1/3 del total de procesamiento a realizar. En general, teniendo p computadoras $P_0, \ldots, P_{p-1}$ con sus correspondientes capacidades de cómputo (por ejemplo, en Mflop/s) $mf_0, \ldots, mf_{p-1}$, se tiene la capacidad de cómputo de la máquina paralela como

$$Totmf = \sum_{i=0}^{p-1} mf_i \tag{12}$$

Y la cantidad de trabajo a realizar en la computadora P_i, wl_i, está dada por su *aporte* de cómputo con respecto al total,

$$wl_i = \frac{mf_i}{Totmf} \tag{13}$$

es decir que P_i debe computar wl_i del total de procesamiento a llevar a cabo. Siguiendo el ejemplo anterior, se tendrá que $mf_0 = 2k$, $mf_1 = k$, $Totmf = 3k$, $wl_0 = 2/3$, y $wl_1 = 1/3$. Una vez que se determina cuánto trabajo realizará cada nodo dependiendo de su capacidad, se pueden distribuir los datos.

4.1 Distribución de los Datos

La matriz a factorizar se divide en bloques de filas (todos los bloques tienen el mismo tamaño). Estos bloques se deben distribuir entre las computadoras que componen el cluster. A causa de la heterogeneidad de procesamiento y por la naturaleza propia del procesamiento de la factorización LU, la forma de distribuir los bloques tiene dos características fundamentales:

- Distribución cíclica: los bloques se distribuyen de forma tal que se evita que queden nodos inactivos a medida que el método de factorización progresa [12]. Así se logra prevenir el desbalance de carga que se produce por el avance mismo de la factorización [7].
- Distribución de los bloques dependiendo de la capacidad de procesamiento de cada máquina. Las máquinas con mayor capacidad, tendrán asignadas mayor cantidad de bloques que las que tienen menor capacidad.

Para realizar esta distribución, los bloques se dividen en grupos (denominados "chunks"), estos chunks se distribuyen en forma secuencial (uno por vez), y los bloques dentro de un chunk se distribuyen de manera cíclica entre los distintos nodos. Retomando el ejemplo anterior, donde P_0 y P_1 son tales que $mf_0 = 2k$, $mf_1 = k$, $Totmf = 3k$, $wl_0 = 2/3$, y $wl_1 = 1/3$, y teniendo en cuenta que la matriz a factorizar se divide en nueve bloques de filas, se considera que los chunks están constituidos por tres bloques de filas, tal como lo muestra la Fig. 2. Y en general,

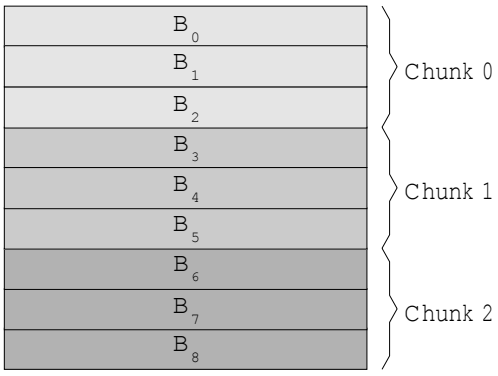

Figura 2: Bloques y Chunks de la Matrix a Factorizar.

el tamaño de los chunks en cantidad de bloques estará dado por

$$ch\_blks = \sum_{i=0}^{p-1} \frac{mf_i}{\min_{j=0}^{p-1}(mf_j)} \tag{14}$$

Luego, y siguiendo con el ejemplo, se distribuyen los tres bloques de cada chunk entre las dos computadoras que componen el cluster, según la capacidad de cada una de ellas, quedando distribuidos de la forma que muestra la Fig. 3

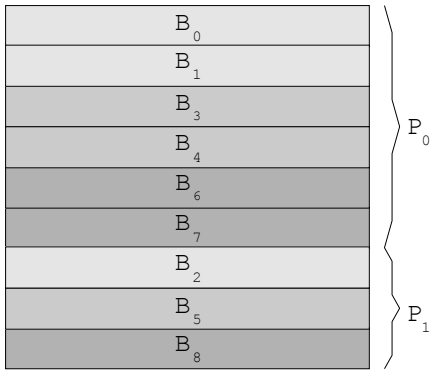

Figura 3: Distribución de Bloques de la Matriz a Factorizar.

4.2 Algoritmo en cada Máquina

Una vez distribuidos los datos, cada procesador se ocupa de factorizar los bloques que le pertenecen y actualizar todos los bloques a medida que avanza la factorización. El modelo de procesamiento es SPMD (Single Program-Multiple Data) [15], es decir que en todos los nodos se ejecuta el mismo algoritmo. Además se utiliza pasaje de mensajes para la comunicación en tre los nodos. En la Fig. 4 se muestra el algoritmo en pseudocódigo que se ejecuta en el nodo P_i. donde `Lij` hace referencia al bloque L_{ij} y `Uij` hace referencia al bloque U_{ij}. Se pueden en umerarlas características más importantes del algoritmo propuesto como:

- T odaslas comunicaciones son de tipo broadcast, tanto del vector de pivotes como del bloque factorizado.
- Las operaciones utilizadas para resolver LU, los sistemas de ecuaciones triangulares y la multiplicación de matrices en cada iteración, corresponden a las operaciones lapack_xgetrf, blas_xtrsm y blas_xgemm respectivamente. Estas operaciones están totalmente optimizadas, por lo que obtenemos procesamiento secuencial optimizado en cada nodo.
- En cada iteración, se solapa la comunicación colectiva (send de pivotes y bloque factorizado) con la actualización del resto de la matriz.

5 Experimentación y Resultados Obtenidos

Se ha implementado el algoritmo para determinar tiempos de ejecución, rendimiento y posibles penalizaciones o *comportamiento* no esperado. P ara realizar las pruebas se ha utilizado un cluster heterogéneo con las siguientes características:

- 8 computadoras, de las cuales 4 son Pen tium III con 64 MB RAM y 4 son Duron con 128 MB RAM.
- Red de in terconexión Ethernet de 100Mb/s con *switching* completo.

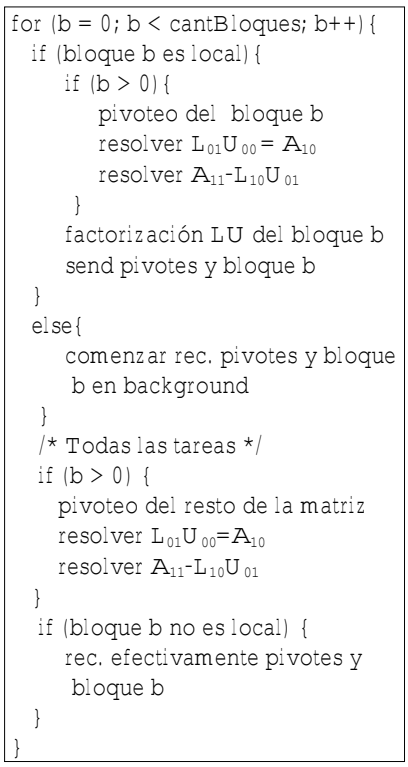

Figura 4: Pseudocódigo del Algoritmo en la Computadora P_i.

De acuerdo con la experimentación con código secuencial optimizado, tanto para la multiplicación de matrices como para la factorización LU de matrices las computadoras con procesadores Duron tienen el doble de capacidad de cómputo que las computadoras con procesadores Pentium III. Específicamente, los valores de los distintos índices definidos antes son:

- Capacidad de las computadoras. Duron: 1.2 Gflop/s, Pentium III: 600 Mflop/s.
- Capacidad total del cluster: (4x1200 + 4x600) Mflop/s.
- $wl_{Duron} = 1/6$, $wl_{PentuimIII} = 1/12$, cantidad de bloques por chunk: 12.

De esta forma, las computadoras Duron recibirán el doble de bloques que las Pentium III, logrando así el balance de carga deseado para este tipo de arquitectura heterogénea.

En la aplicación paralela se utiliza PVM (Parallel Virtual Mahine) [6] para la creación y manejo de procesos, mientras que para la comunicación entre los procesos (específicamente, broadcast de los pivotes y del bloque factorizado) se ha utilizado una biblioteca de comunicaciones que tiene, entre otras, dos características fundamentales:

- implementa el broadcast de los datos a nivel físico, aprovechando la capacidad que proporciona la red Ethernet.
- permite el solapamiento de comunicación y cómputo. Esto depende igualmente de la capacidad de cada máquina de realizar o no este tipo de solapamiento.

La razón principal para no usar PVM como biblioteca de comunicación para realizar los mensajes de tipo broadcast, es que los implementa como múltiples comunicaciones punto a punto. Esto hace que el tiempo de comunicación sea dependiente de la cantidad de máquinas que estén involucradas en dicha comunicación.

Las experimentación realizada además cuenta con las siguientes características:

- En todas las ejecuciones, la dimensión de la matriz es de 12000x12000 elementos. Esta es la máxima dimensión en la que no se necesita memoria swap en las máquinas para el procesamiento de la matriz.
- Se probaron distintas dimensiones de los bloques: 1, 4, 16, 32, 64 y 256 filas por bloque, que son los tamaños usuales en experimentos de este tipo.

La Fig. 5 muestra los resultados obtenidos. En el eje **x** se representan los distintos tamaños de bloques. En el eje **y** se tiene el tiempo de ejecución que corresponde a cómputo numérico local en cada tipo de máquina: P entium III y DuronEs importante notar que este tiempo no incluye comunicación ni esperas por sincronización en trecomputadoras, el código se instrumentó de forma tal que se contabiliza el tiempo de procesamiento de datos sin incluir comunicaciones ni sincronizaciones. Los tiempos de los procesos ejecutados en las máquinasPentium III son

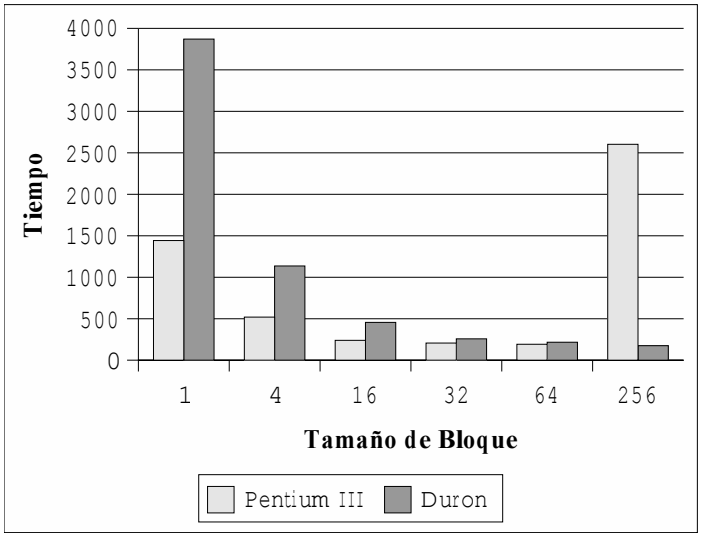

Figura 5: Resultados en un Cluster Heterogéneo.

muy similares entre sí y esto mismo sucede con las computadoras Duron. Por esta razónsólo se representa un solo tiempo de cada tipo de máquina encada ejecución.

En el gráficose observa cómopara los tamaños de bloque 1, 4, 16 y 256 los tiempos de cómputolocal en cada máquinano están balanceados. Si bien era esperable que todos los tiempos de cómputo local sean iguales, es evidente que los tamaños de bloques afectan "heterogéneamente" el rendimiento de las computadoras heterogéneas. Expresado de otra manera, el tamaño de bloque afecta el rendimiento de cada máquina (en este caso a las computadoras con procesador Duron y a las computadoras con procesador Pen tium III) de manera diferente, dado que tienen procesadores (jerarquías de memoria, etc.) diferentes. Sin embargo, para los tamaños de bloque 32 y 64 sucede lo que era de esperar (por la asignación de datos *correspondiente*): el tiempo de cómputo local de cada computadora es similar a pesar de que las computadoras con procesador Duron tienen el doble de carga de cómputoque las computadoras con procesador Pentium III dado que tienen el doble de capacidad de cómputo.

Otra forma de interpretarlos resultados obtenidos se relaciona directamente con el rendimiento paralelo en general y secuencial de cada computadora en particular. Se debe notar

que para los tamaños de bloque 32 y 64 no solamente el tiempo de cómputo local de cada tipo de máquina es similar (demostrando que la carga de trabajo está balanceada), sino que es mínimo. Esto significa que tanto el rendimiento secuencial como el paralelo es máximo. Desde el punto de vista del rendimiento secuencial, todas las computadoras procesan al máximo de su capacidad (notar que, además los tiempos de cómputo local son similares para ambos tamaños de bloque: 32 y 64). Por otro lado, el rendimiento paralelo no solamente se *beneficia* por el cómputo local maximizado sino que además también se evitan penalizaciones de rendimiento por esperas entre procesos que tienen distintos tiempos de ejecución.

6 Conclusiones y Trabajo Futuro

Se han presentado los principios para balancear la carga de procesamiento en clusters heterogéneos y se ha aplicado al problema de factorización LU de matrices. Es interesante que a pesar de que la descripción se ha enfocado a la factorización LU, la mayoría de las otras factorizaciones de matrices (QR, RQ, Cholesky, etc.) incluidas en LAPACK pueden aplicar los mismos principios para obtener balance de carga en paralelo. La comprobación del balance de carga se ha dado por la experimentación en un cluster con dos tipos de máquinas y ha mostrado ser satisfactoria siempre y cuando los tamaños de bloques básicos a procesar sean *correctos*. Expresado de otra manera:

- Si las computadoras procesan al máximo de su capacidad, entonces se tiene balance de carga y rendimiento secuencial y paralelo optimizado.
- Si las computadoras no procesan al máximo de su capacidad (en el caso de la experimentación por los tamaños de bloque *inapropiados*), entonces no solamente se tiene penalizado el cómputo local sino que además produce que los tiempos de cómputo local muestren desbalance de carga de procesamiento. Resulta interesante notar que esto se puede utilizar como una forma *automática* para la detección de problemas de cómputo local que penalizan severamente el rendimiento paralelo (se debe recordar que siempre se termina procesando a la velocidad de la computadora con menor capacidad).

En términos de trabajo futuro, queda todavía utilizar y demostrar la efectividad del balance de carga propuesto en ambientes donde la heterogeneidad es mayor. En el caso extremo, quizás haya casos en los que las diferencias de capacidades relativas sean tan grandes que sean mejor dejar de utilizar computadoras en vez de intentar balancear la carga incluyéndolas. Se deben identificar y cuantificar claramente cuáles son los casos en los que esto sucede. Por otro lado, también es necesario probar el método propuesto en este trabajo en clusters con mayores cantidades de máquinas, donde no solamente se puede tener mayor heterogeneidad sino que también se puede identificar con mayor precisión los alcances en cuanto a escalabilidad, por ejemplo.

Referencias

[1] Anderson E., Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, "LAPACK: A Portable Linear Algebra Library for High-Performance Computers", Proceedings of Supercomputing '90, pages 1-10, IEEE Press, 1990.

[2] Anderson E., Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, D. Sorensen, LAPACK Users' Guide (Second Edition), SIAM Philadelphia, 1995.

[3] Anderson T., D. Culler, D. P atterson, and the NOW Team, "A Case for Networks of Workstations: NOW", IEEE Micro, Feb. 1995.

[4] Baker M., R. Buyya, "Cluster Computing at a Glance", in R. Buyya Ed., High Performance Cluster Computing: Architectures and Systems, Vol. 1, Prentice-Hall, Upper Saddle River, NJ, USA, pp. 3-47, 1999.

[5] Dongarra J., J. Du Croz, S. Hammarling, R. Hanson, "An extended Set of F ortranBasic Linear Subroutines", ACM T rans.Math. Soft., 14 (1), pp. 1-17, 1988.

[6] Dongarra J., A. Geist, R. Manchek, V. Sunderam, "Integrated p vm framework supports heterogeneous network computing", Computers in Physics, (7)2, pp. 166-175, April 1993.

[7] Dongarra J., D. Walker, "Libraries for Linear Algebra", in Sabot G. W. (Ed.), High Performance Computing: Problem Solving with Parallel and Vector Architectures, Addison-Wesley Publishing Company , Inc., pp. 93-134, 1995.

[8] Flynn M., "Very High Speed Computing Systems", Proc. IEEE, Vol. 54, 1966.

[9] Flynn M., "Some Computer Organizations and Their Effectiveness", IEEE Trans. on Computers, 21 (9), 1972.

[10] Institute of Electrical and Electronics Engineers, Local Area Network - CSMA/CD Access Method and Physical Lay er Specifications ANSI/IEEE 802.3 - IEEE Computer Society , 1985.

[11] Lawson C., R. Hanson, D. Kincaid, F. Krogh, "Basic Linear Algebra Subprograms for F ortran Usage", ACM T ransactionson Mathematical Software 5, pp. 308-323, 1979.

[12] Kumar V., A. Grama, A. Gupta, G. Karypis, Introduction to Parallel Computing. Design and Analysis of Algorithms, The Benjamin/Cummings P ublishing Company , Inc., 1994.

[13] Tinetti F. G., E. Luque, "Parallel Matrix Multiplication on Heterogeneous Networks of Workstations", Proceedings VIII Congreso Argentino de Ciencias de la Computación (CACIC), F ac.de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Buenos Aires, Argentina, p. 122, Oct. 2002.

[14] Tinetti F. G., A. Quijano, A. De Giusti, E. Luque, "Heterogeneous Networks of Workstations and the P arallel Matrix Multiplication", Y. Cotronis and J. Dongarra (Eds.): EuroPVM/MPI 2001, LNCS 2131, pp. 296-303, Springer-Verlag, 2001.

[15] Wilkinson B., Allen M., P arallelProgramming: T ec hniquesand Applications Using Networked Workstations, Prentice-Hall, Inc., 1999.