Álgebra Lineal en Paralelo: Factorizaciones en Clusters Heterogéneos

Fernando G. Tinetti, Mónica Denham
III-LIDI (Instituto de Investigación en Informática LIDI)
Facultad de Informática, UNLP
50 y 115, 1900 La Plata
Argentina
fernando@lidi.info.unlp.edu.ar, mdenham@lidi.info.unlp.edu.ar
Tel.: +54 221 4227707, Fax: +54 221 4273235

Resumen

En este trabajo se presenta una solución paralela a la factorización LU de matrices para ser utilizada en clusters heterogéneos interconectadas por redes Ethernet. Primero se describe el problema a resolver y las principales características de la solución secuencial por bloques y la solución paralela propuesta. Luego se presentan distintos métodos para la distribución de trabajo entre los nodos con el objetivo de obtener balance de carga. Para cada método se describe la experimentación realizada y los resultados obtenidos. Por último, se describen las conclusiones y el trabajo futuro.

Palabras Clave: Cómputo en Clusters, Clusters Heterogéneos, Balance de Carga en Clusters, Rendimiento de Cómputo.

Workshop al que está dirigido: V Workshop de Procesamiento Distribuido y Paralelo.

1. Introducción

En el área del cómputo paralelo, uno de los principales objetivos es obtener soluciones a problemas con máximo rendimiento. Para obtener soluciones que realmente consigan buen rendimiento, es útil tener en cuenta las características propias de la arquitectura específica que se utilizará para ejecutar dicha solución.
De esta forma, se obtienen soluciones a problemas orientadas a un tipo de arquitectura en particular. Esto tiene las ventajas de poder aprovechar estas características algorítmicamente, como así también evitar penalizaciones ocasionadas por propiedades específicas de la arquitectura. Pero tiene la desventaja de perder generalidad, pues una solución orientada a una arquitectura, seguramente no obtenga buen rendimiento si se la utiliza en otras arquitecturas con características distintas a para la cual fue desarrollado.
Este trabajo es la continuación de [16] en el cual se describe una solución a la factorización LU para un tipo de cluster heterogéneo específico.
Como se menciona en [16], la factorización LU de matrices pertenece a LAPACK (Linear Álgebra PACKage), y es utilizada para la resolución de sistemas de ecuaciones lineales. Los sistemas de ecuaciones lineales se usan en diversas aplicaciones de diferentes dominios, como por ejemplo física, medicina, simulaciones, etc. También se está utilizando la factorización LU para testear sistemas y listarlos en el TOP500, en el cual se encuentran los 500 sistemas más rápidos que se usan en la actualidad [18]. Por otra parte, existen otras factorizaciones que utilizan el mismo patrón de procesamiento que la factorización LU (factorización QR, Cholesky).
En este trabajo se describen los avances obtenidos en la solución a dicha factorización para clusters heterogéneos (generalizándose dicho trabajo para clusters con nodos de cualquier tipo y capacidad).

Esta solución también se puede utilizar en un cluster homogéneo, donde todos los nodos tienen la misma potencia de cómputo.
Un factor muy importante en la solución paralela es el balance de carga. Este especto no es trivial ya que los nodos pertenecientes a un cluster heterogéneo pueden tener distintas capacidades (de memoria, cómputo, etc), por lo tanto, se debe tener especial cuidado en la cantidad de trabajo asignado a cada nodo, para que el rendimiento no se vea penalizado por el tiempo del nodo que más tarda en resolver la factorización.
Para obtener balance de carga, se tienen en cuenta las capacidades de cada uno de los nodos que componen el cluster. Dicha capacidad puede ser los Mflop/s (millones de operaciones en punto flotante por segundo) de cada uno de los nodos que componen el cluster. Para obtener los Mflop/s se debe ejecutar en cada nodo (o cada tipo de nodo) algún algoritmo que permita caracterizar el rendimiento. En este caso, se utiliza la propia factorización LU secuencial en cada uno de los nodos.
La solución propuesta es una solución paralela por bloques de la factorización LU. Esto es, la matriz se divide en bloques y estos bloques son los que se distribuyen entre los nodos y la factorización se resuelve bloque a bloque progresivamente. En [5] se propone dividir los bloques en grupos y distribuir los bloques de cada grupo según la capacidad de cómputo de cada nodo. En este trabajo, se utiliza esta idea, pero además se agrega que los bloques de cada uno de los grupos se distribuyen de forma cíclica entre los nodos, para evitar desbalance de carga a medida que avanza la factorización.
Se muestran distintos métodos de distribución de bloques, los cuales tienen las siguientes características:

- distribución por bloques enteros de la matriz.
- distribución de los bloques en función de la capacidad de cómputo de cada nodo.
- distribución cíclica de los bloques.

Primero se describe un método para clusters homogéneos, el cual incluye la idea de dividir los bloques en grupos y luego repartir estos bloques entre los nodos.
Luego, se describe un método para clusters heterogéneos, casi "directo". Este método tiene las características mencionadas anteriormente. Luego se describen dos métodos más, los cuales intentan solucionar los problemas encontrados en los anteriores, avanzando a soluciones que logran balance de carga y que mantienen las características mencionadas anteriormente. Esto amplia el trabajo en [16] generalizando la solución a clusters formados por nodos de cualquier capacidad.

2. Factorización LU por bloques y descripción de la arquitectura utilizada

La operación implementada es la factorización LU de matrices, la cual pertenece a la librería LAPACK y es utilizada para resolver sistemas de ecuaciones lineales. La factorización LU convierte un sistema como:

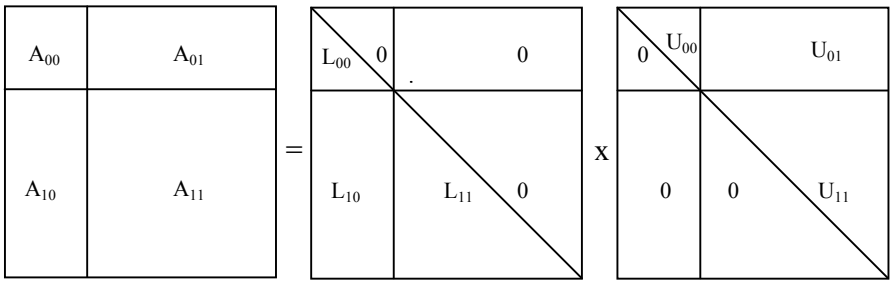
$$Ax = b \qquad (1)$$

en dos sistemas triangulares equivalentes que se utilizan para encontrar el valor de las incógnitas representadas por x en la ecuación (1).
El método consiste en encontrar las matrices L y U tales que:

$$A = LU \qquad (2)$$

Para obtener los valores de L y U tal que se cumpla la ecuación (1), se aplican sucesivos pasos de eliminación Gaussiana sobre los datos de la matriz original A.

Teniendo en cuenta los distintos niveles de memoria cache instalados en las computadoras y con el objetivo de aprovecharlos, es usual resolver operaciones de álgebra lineal (incluyendo esta factorización) por bloques. La solución paralela propuesta está basada en la solución secuencial por bloques, por lo que se explicará brevemente.
La matriz a factorizar se divide en bloques como muestra la figura 1:



$A_{00} = L_{00}U_{00}$ (7)
$A_{01} = L_{00}U_{01}$ (8)
$A_{10} = L_{10}U_{00}$ (9)
$A_{11} = L_{10}U_{01} + L_{11}U_{11}$ (10)

Figura 2.1: División por bloques de la matriz.

Entonces, se deben hallar los bloques L_{00}, L_{10}, L_{11} y U_{00}, U_{01} y U_{11}, de la Figura 2.1. Primero, se aplica LU de forma directa a A_{00}, obteniendo L_{00} y U_{00} (resolviéndose la ecuación (7)). Luego, como L_{00} es una matriz triangular inferior, se aplica el método de resolución de sistema de ecuaciones triangulares, y se resuelve la ecuación (8), obteniéndose U_{01}. De la misma forma y usando la ecuación (9) se obtiene L_{10} (U_{00} es una matriz triangular superior). Por último, para obtener las submatrices L_{11} y U_{11} se usa la ecuación (10), la cual se puede transcribir como:

$$L_{11}U_{11} = A_{11} - L_{10}U_{01} \quad (11)$$

Si se aplica el mismo método de resolución por bloques al resultado de la resta $A_{11} - L_{10}U_{01}$, se obtiene la factorización completa de la matriz A.
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.
En el algoritmo propuesto se incluye una técnica de pivoteo con el objetivo de mantener estabilidad numérica. Esto implica que antes de factorizar cada dato, se obtiene el elemento de mayor valor absoluto de toda la fila y se permutan las columnas del dato a factorizar y la columna donde se encuentra el máximo.
Por otro lado, la arquitectura utilizada es un cluster heterogéneo. Este tipo de máquina paralela pertenece a las arquitecturas MIMD (Multiple Instruction – Multiple Data stream) [15]. Esta máquina paralela está formada por múltiples nodos interconectados por una red Ethernet. Los nodos son computadoras de escritorio comunes, por lo tanto, se utiliza una red local como máquina paralela.
En la mayoría de los casos, estas redes están formadas por nodos los cuales no necesariamente son iguales entre si, pudiendo variar (en un amplio rango) las capacidades de cómputo, almacenamiento, etc. (esto es lo que hace que la arquitectura sea heterogénea). Cada computadora tiene una placa de red y la red Ethernet puede utilizar switches y/o hubs.
Este tipo de arquitectura tiene la ventaja de utilizar redes de computadoras ya instaladas y en uso, por lo tanto el costo de hardware, instalación y mantenimiento son nulos (o casi nulos). Además, si se necesita agregar un nodo (para reemplazar uno existente o para ampliar la red), este nodo puede ser cualquier computadora. Esto es una gran ventaja dado que la tecnología utilizada en computadoras de escritorio cambia constantemente, por lo tanto ajustarse a un tipo de nodo específico, haría mucho más complejo la actualización del cluster. Además, el precio de las

computadoras de escritorio y del cableado de una red Ethernet es mucho más bajo que el precio de cualquier nodo o red de interconexión de una máquina paralela.
Por otro lado, la red de comunicación Ethernet es lenta si se la compara con una red de interconexión de una máquina paralela. Otra desventaja es que la heterogeneidad de los nodos implica una rigurosa distribución de trabajo, para lograr que todos los nodos tarden mas o menos el mismo tiempo y no se penalice todo el cómputo por el nodo más lento.

3. Algoritmos y experimentación

El algoritmo utilizado es el mismo que en [16], por lo tanto sólo se enumerarán las principales características del mismo. Luego se pone especial atención a la distribución de trabajo entre los nodos, describiéndose cada uno de los métodos propuestos.
Las principales características del algoritmo utilizado son: resuelve la factorización LU por bloques, es de tipo SPMD (donde en todos los nodos se ejecuta el mismo código). Las operaciones utilizadas para resolver LU, la multiplicación de matrices y la resolución de sistemas de ecuaciones triangulares, están totalmente optimizadas, por lo que se tiene procesamiento secuencial optimizado en cada nodo. Por otra parte, todas las comunicaciones son de tipo broadcast, aprovechando así la capacidad de broadcast físico de la red Ethernet. Además, se utiliza una rutina de comunicación que permite solapar comunicación con cómputo (esta característica igualmente depende de la capacidad de cada nodo de poder solapar).
Como ya se mencionó, la distribución de datos es la que definirá el balance de carga. Se han desarrollado 4 métodos, los cuales se describirán y se mostrarán resultados de distintas experimentaciones.
Primero se describirá una distribución para clusters homogéneos la cual incluye la división de bloques en grupos. Luego se usa esta idea pero en clusters heterogéneos, donde se asignan bloques directamente utilizando los cientos de Mflop/s de cada máquina. Luego, se propone realizar ciertos cálculos sobre las potencias para definir un tamaño de grupo más chico pero manteniendo la relación de las potencias entre los nodos. Con este tamaño de grupo, se distribuyen los bloques de 2 formas distintas: distribución de bloques consecutivos o distribución de bloques cíclica dentro de cada grupo.

Cluster Homogéneo

Cuando el cluster es homogéneo, todos los nodos tienen la misma capacidad de procesamiento. En este caso, la distribución de datos es directa, ya que se asigna la misma cantidad de bloques a cada uno de los nodos que componen el cluster. De esta forma, todos los nodos tardarán aproximadamente el mismo tiempo. Los bloques se asignan de forma cíclica, uno a cada nodo, hasta asignar cada uno de los bloques de la matriz.
Para realizar esta distribución, se dividen los bloques en grupos [5]. El tamaño del grupo es igual a la cantidad de nodos que componen el cluster y cada nodo recibe un bloque de cada uno de los grupos.
La figura 3.1 ilustra esta distribución. La matriz se divide en 5 bloques y se distribuyen entre 3 procesadores homogéneos. En la figura 3.1 se muestra el caso de que la cantidad de bloques no sea divisible por la cantidad de nodos, quedando nodos que reciben 1 bloque menos. Entonces, el balance de carga es cercano al deseado, como máximo la diferencia es de un bloque (entre la cantidad que debe recibir y la que realmente recibe). Como la distribución es de bloques enteros, esta diferencia de a lo sumo un bloque es inevitable.

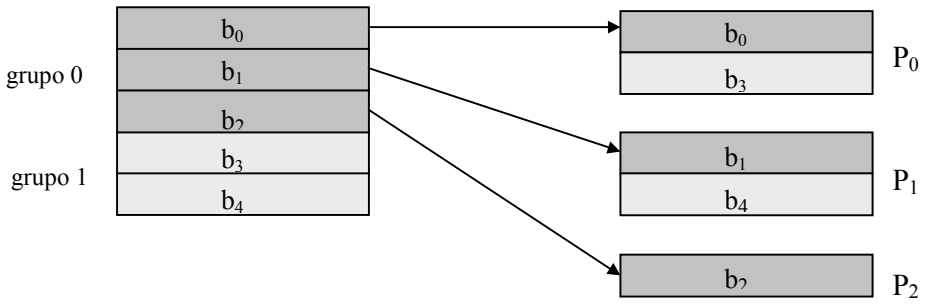


Figura 3.1: Distribución cíclica de bloques en un cluster homogéneo.

Esta distribución es correcta para clusters homogéneos, pero no para clusters heterogéneos, ya que todos los nodos reciben la misma cantidad de trabajo, por lo tanto el tiempo total de resolución estará determinado por el tiempo del nodo que más tarda (el más lento). Por esta razón se mostrarán directamente pruebas con distribuciones que tienen en cuenta la heterogeneidad del cluster.
El método para clusters homogéneos incluye la división de los bloques en grupos, la cual se utiliza en los métodos siguientes.

Cluster heterogéneo

Método "Directo"

Cuando el cluster está formado por nodos con diferentes potencias de procesamiento, la distribución de datos no es directa como en el caso homogéneo. La distribución de carga de trabajo debe ser tal que se evite que el tiempo total de procesamiento esté determinado por el nodo más lento y que queden nodos ociosos durante la factorización. Esto significa que se debe repartir el trabajo de tal forma que todos los nodos tengan tiempo de procesamiento similar. Esto se logra repartiendo los bloques en función de la capacidad de procesamiento de cada nodo.
Sean $p_0..p_{n-1}$ los nodos que componen el cluster y sea Pp_i la potencia del nodo p_i. En este caso, se toma solo la parte más representativa de los Mflop/s, o sea, cada p_i son los cientos de Mflop/s de p_i. Además, los nodos están ordenados de potencia mayor a menor ($Pp_0 > Pp_1 > ... > Pp_{n-1}$).
Cada grupo se compone por la sumatoria de todas las potencias, esto es

$$\textbf{dimensión grupo} = \sum Pp_i \qquad (13)$$

Luego, se distribuyen los bloques del grupo teniendo en cuenta directamente cada potencia Pp_i, o sea, a p_0 se le asignan los primeros Pp_0 bloques del grupo, a p_1 se le asignan los siguientes Pp_1 bloques, y así con todos los nodos y con cada uno de los grupos
De esta forma, se obtiene que cada nodo recibe una cantidad de bloques proporcional a su capacidad de cómputo, como era el objetivo.
Por ejemplo, si se tiene una matriz dividida en 14 bloques y tenemos 3 nodos, p_0, p_1 y p_2, con potencias $Pp_0 = 3$, $Pp_1 = 2$ y $Pp_2 = 1$ respectivamente. Realizando la distribución descrita anteriormente, cada grupo tiene 6 bloques, y se distribuyen 3 bloques para p_0, 2 bloques para p_1 y 1 bloque para p_2 de cada uno de los grupos en que se dividen los bloques de la matriz. La figura 3.2 muestra esta distribución.

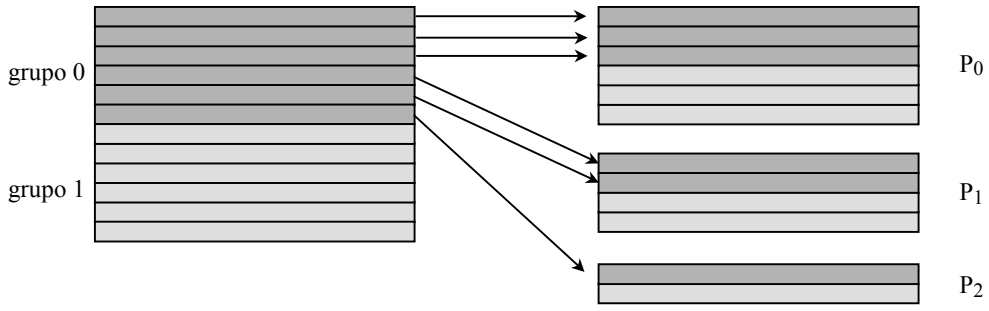


Figura 3.2: Distribución cíclica de bloques en un cluster heterogéneo.

Se ha experimentado con un cluster formado de la siguiente forma:

Cantidad	Nombre	CPU	Frec. Reloj	Memoria	Mflop/s LU por bloques
4	Lidipar73..76	AMD-Duron	850 MHz	256 MB	11
4	Lidipar65..68	Pentium III	700 MHz	64 MB	9

El tamaño de matriz es de 7936x7936 flotantes y el tamaño de bloque es de 64 filas cada uno. Se obtuvo la siguiente distribución y tiempos:

	filas asignadas	% ideal	% recibido	Diferencia
Lidipar73..76	1408	13.75%	17.74%	+3.99
Lidipar65..68	576	11.25%	7.25%	-4

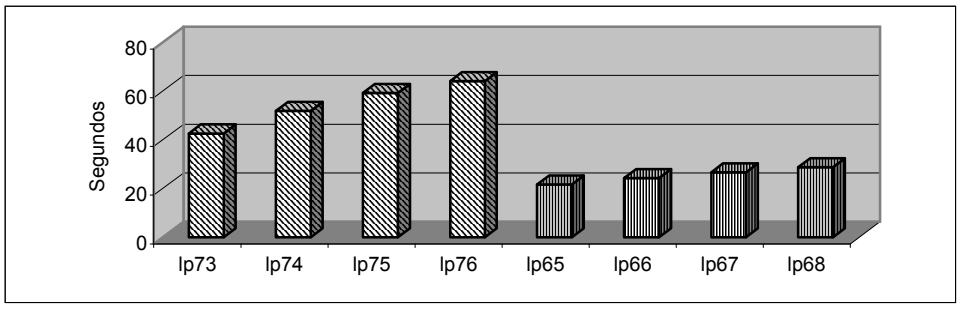


Figura 3.3: Tiempos obtenidos utilizando el método “directo”.

Este método tiene las ventajas de tener una implementacion fácil y directa, y cuando la cantidad de bloques y el tamaño de grupo son apropiados se obtiene un balance de carga aceptable. De todas maneras, el buen funcionamiento del método es totalmente dependiente de la cantidad de bloques y del tamaño de grupo. La mayoría de las veces, se puede observar que:

- la cantidad de bloques por grupo es muy grande.
- el tamaño del grupo no divide a la cantidad de bloques total de la matriz (quedando un último grupo incompleto).

La primera observación provoca pocos grupos con muchos bloques cada uno. Por otro lado, este método reparte los bloques de todos los grupos de la misma forma (incluyendo el último grupo que posiblemente, pueda tener menos bloques). Esto hace que haya nodos que reciben bloques del último grupo y otros que no. Si se tiene en cuenta que p_i recibe Pp_i bloques consecutivos del grupo, y los bloques están compuestos por múltiples filas, esto puede llegar a producir una diferencia importante de trabajo asignado y el trabajo que realmente debería recibir.

Se han realizado distintas pruebas, y se incluye una prueba donde sucede lo anteriormente explicado: los bloques del último grupo (incompleto) se reparten entre los primeros 4 nodos, distribuyéndose mal la carga de trabajo entre los distintos nodos.
Además, se observa una gran diferencia de tiempos entre los distintos tipos de nodos, la cual no se esperaba ya que se esperaba un desbalance menor, correspondiente al desbalance producido por los bloques del último grupo. Este comportamiento se observa en todas las pruebas realizadas (las cuales no se incluyeron en este trabajo) y en pruebas donde el porcentaje de trabajo recibido por cada nodo es cercano al óptimo. Esta diferencia no debería ser tan grande, por lo que se puede pensar que la caracterización de los Mflop/s no es la correcta (se utilizó la factorización LU secuencial para determinar esta cantidad). Por esta razón, se recalcularon los Mflop/s de los nodos pero esta vez utilizando la factorización LU secuencial por bloques. Lo que se pudo observar es que este hecho (la utilización de bloques) afecta en gran forma a los nodos Duron y no así a los nodos Pentium III. Entonces, se utiliza la factorización secuencial pero por bloques ya que es la que mejor caracteriza a todos los nodos.

Método propuesto inicial

El método propuesto se basa en armar grupos de pocos bloques, y distribuir de forma especial los bloques del último grupo, en el caso de que éste quede incompleto. De esta forma, intenta obtener un balance de carga independientemente de factores tales como tamaño de grupo, tamaño de bloque, tamaño de la matriz, etc, que determinan la distribución final de los bloques.
Para esto, el método propuesto realiza cálculos sobre las potencias, de tal forma que se mantenga la relación de potencia de cómputo entre los nodos, pero reduciéndolas y así obtener grupos con menos bloques. Una vez que se obtienen las nuevas potencias, éstas se usan de igual forma que en el método anterior para repartir los bloques entre los nodos.
Además, si el último grupo queda incompleto, se realizan nuevos cálculos sobre las potencias para resolver la distribución de estos bloques restantes.
Los cálculos realizados sobre las potencias son:
1) se obtiene el nodo con potencia mínima ($p_{\min}$). Este valor será usado en los próximos pasos.
2) se dividen todas las potencias por Pp_{min} y se multiplican por 10. Al dividir cada potencia por la potencia menor, se obtienen a todas las potencias en relación a la potencia mínima. O sea, se expresan todas las potencias como el trabajo que realizará cuando p_{min} realiza una unidad del trabajo total (pues p_{min} va a quedar como 1). Como estas divisiones pueden no ser exactas, se multiplican por 10 para incluir el dígito más significativo de la parte decimal, de esta forma se obtiene mayor precisión al intentar expresar las potencias en relación a la potencia menor.
3) se obtiene el máximo común divisor (mcd) de estas potencias y se dividen todas las potencias por el mcd. Al dividir todas las potencias por el mismo número, se obtienen potencias más chicas pero que mantienen la relación de las distintas capacidades de cómputo que representan.
Luego, estas potencias se utilizan para formar cada grupo y repartir los bloques de cada grupo entre los nodos, de la misma forma que se realizaba en el método directo.
Por los cálculos realizados sobre las potencias originales, se sigue manteniendo la relación entre las potencias originales y la cantidad de trabajo que recibe cada nodo.
Cuando el tamaño del grupo no divide exactamente a la cantidad de bloques, el último grupo queda incompleto. Para evitar un posible desbalance de carga ocasionado por la distribución de estos bloques, se "recalculan" las potencias, para ver cuánto trabajo realiza cada nodo de este grupo incompleto. Como se conoce el tamaño de grupo y para cada nodo se conoce cuántos bloques recibe del grupo completo, se utilizan estos valores para obtener cuántos bloques debe recibir del grupo incompleto (por regla de 3 simple).

Si luego del recálculo de las potencias y la distribución de estos bloques del grupo incompleto, siguen quedando bloques sin asignar, estos se asignan cíclicamente uno por nodo.
Se han realizado distintas pruebas, donde en todas se obtuvieron resultados semejantes. Como ejemplo, se incluye una prueba en un cluster formado por 16 nodos, estos nodos son de dos tipos distintos:

Cantidad	Nombre	CPU	Frec. Reloj	Memoria	Mflop/s LU por bloques
8	Watson26,29,22, 24,14,19,17,18	Pentium 4	2.4 GHz	1GB	5994
8	Lidipar65..72	Pentium III	700 MHz	64 MB	1418

El tamaño de matriz es de 20160 x 20160 flotantes (con este tamaño no se utiliza memoria swap en los nodos). El tamaño de bloque es de 64 filas. A continuación se resume el porcentaje de trabajo recibido por cada nodo y se grafican los tiempos obtenidos.

	filas asignadas	% ideal	% recibido	Diferencia
Watson26,29,22, 24,14,19,17,18	2048	10.10%	10.15%	0.05
Lidipar 65, 66 y 67	512	2.39%	2.53%	0.14
Lidipar 68..72	448	2.39%	2.22%	-0.17

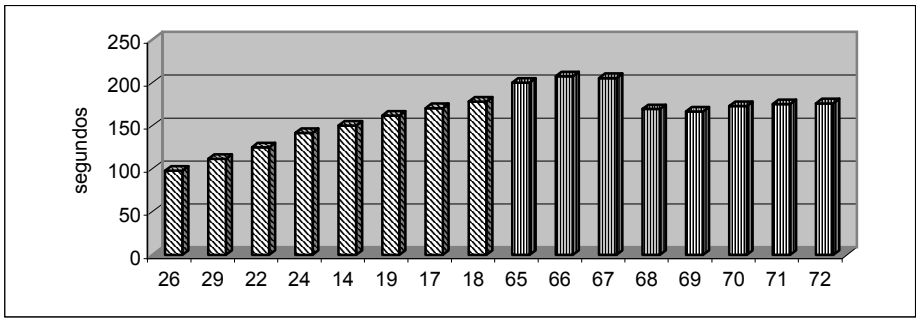


Figura 3.4: Tiempos obtenidos utilizando el método propuesto inicial.

En esta prueba se observa que los primeros 3 nodos Pentium III reciben un bloque más que el resto de las Pentium III y reciben más trabajo del que deberían recibir.
Este caso es interesante porque quedan bloques restantes que se distribuyen entre los primeros 11 nodos, entre los cuales hay 3 Pentium III (con menos capacidad). Estos nodos son los que determinan el tiempo total de la factorización. Esta diferencia se nota con 1 bloque (diferencia mínima), entonces, si la diferencia hubiese sido de más bloques, el tiempo se hubiese visto más penalizado por estos últimos nodos más lentos.
Un comportamiento no esperado es el que se observa para los primeros 8 nodos (Pentium 4). Se puede observar una especie de "escalera" en los mismos. Este comportamiento no debería ocurrir ya que estos nodos tienen la misma capacidad y reciben la misma cantidad de filas (2048), por lo tanto se espera que tarden tiempos similares. Esto se debe a que los bloques de cada grupo se asignan de forma consecutiva a cada nodo. En esta prueba, cada uno de los primeros 8 nodos reciben 21 bloques consecutivos del grupo y cada uno de los nodos restantes (los 8 nodos Pentium III) reciben 5 bloques consecutivos. Entonces, a medida que avanza la factorización, cada uno de los primeros nodos va reduciendo la parte activa de la matriz de a 21 bloques (de 64 bloques cada uno). Esto hace que los primeros nodos vayan reduciendo su parte activa de la matriz de una forma significativa y esto influya en los tiempos. Para los nodos Pentium III esto también sucede, pero de

a 5 bloques cada vez (por lo tanto no es tan significativo). Este comportamiento es el que determina la diferencia con el próximo método de distribución.

Método propuesto mejorado

Este método es similar al anterior en cuanto que se recalculan las potencias de la misma forma a como se explicó anteriormente, pero no se toman bloques consecutivos dentro de cada grupo, sino que se asignan cíclicamente entre los nodos.
Con el mismo ejemplo de la Figura 3.2, utilizando 3 nodos de potencia 3, 2 y 1, se reparten los 12 bloques en que se divide la matriz original

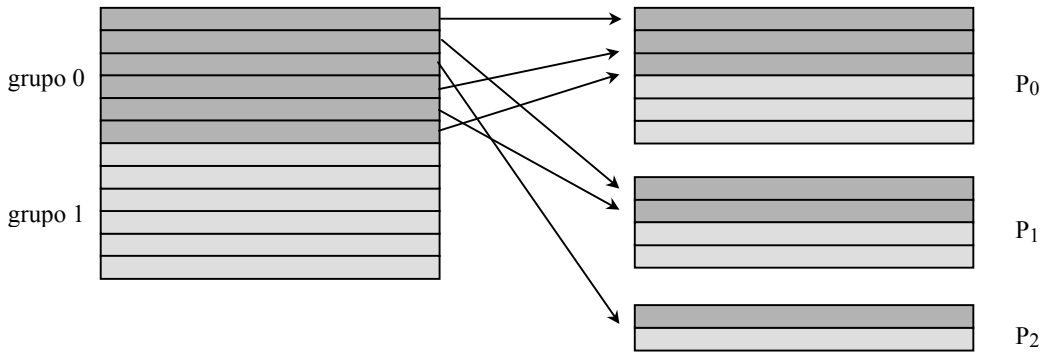


Figura 3.5: Distribución cíclica de bloques dentro de cada grupo en un cluster heterogéneo.

Se muestran sólo las flechas del primer grupo para no perder claridad. En la figura se observa que los bloques dentro de cada grupo se asignan cíclicamente entre todos los nodos, con esto se evita el desbalance producido por la progresión de la factorización.
Se han realizado las mismas pruebas que con el método propuesto inicial, utilizando el mismo cluster. La figura 3.6 muestra los resultados obtenidos.

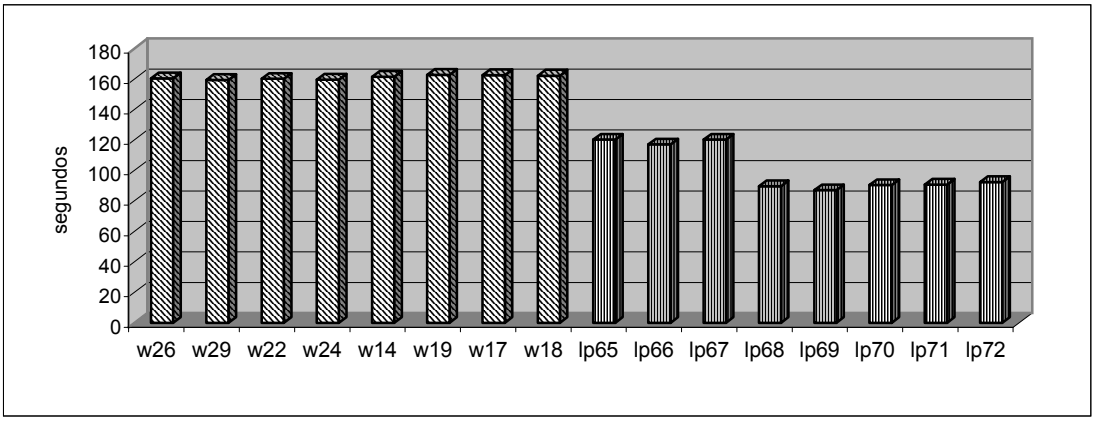


Figura 3.6: Tiempos obtenidos utilizando el método propuesto mejorado.

El gráfico muestra cómo, al distribuir los bloques de cada grupo de forma cíclica, todos los nodos que tienen la misma capacidad reciben la misma cantidad de bloques y tardan un tiempo similar en realizar el trabajo.

Por otro lado, se observa diferencia entre los tiempos de los distintos tipos de nodos, esta diferencia está dada por la diferencia entre el porcentaje de trabajo que debe realizar un nodo y el que realmente se le asigna. Estas diferencias dependen del tamaño de bloque, tamaño de la matriz y de las potencias de los nodos. Lo que se intenta en este método es que estas diferencias sean lo más chicas posibles para cualquier tipo de cluster (homogéneo o heterogéneo), y que sirva para cualquier tipo de "heterogeneidad".

4. Conclusiones y trabajo futuro

Se han desarrollado distintos métodos para la distribución de carga de trabajo para la factorización de matrices en un cluster heterogéneo.
Primero se describe un método para clusters homogéneos. Luego, se describe el método "directo", el cual es muy simple y fácil para implementar. Sin embargo, el balance de carga obtenido puede no ser bueno, ya que puede existir mucha diferencia entre el porcentaje de trabajo asignado a un nodo y el porcentaje de trabajo que debería recibir en función de su capacidad. Esto sucede principalmente cuando se asignan muchos bloques seguidos dentro de un grupo, y queda el último grupo incompleto. De esta forma, hay nodos que reciben bloques de este último grupo y otros que no, por lo tanto, pueden quedar nodos consecutivos con potencias "similares" o iguales (los nodos están ordenados desde la máxima potencia a la mínima) pero con mucha diferencia de bloques asignados, produciéndose un desbalance de carga que afecta en el tiempo total de resolución. El objetivo es encontrar un método que obtenga balance de carga para cualquier cluster, sin depender de factores tales como tamaño de matriz, tamaño de bloque, potencias de los nodos, etc.
Luego se propone un método que intenta armar grupos con menor cantidad de bloques. El porcentaje de trabajo asignado a cada nodo es cercano al ideal, pero se pudo observar un desbalance de carga producido a medida que avanza la factorización. Esto es producido porque se asignan bloques consecutivos de cada grupo a cada nodo, y la parte activa de la matriz se va reduciendo con mucha rapidez cuando los nodos tienen muchos bloques consecutivos.
Para solucionar este problema, se propone distribuir de forma cíclica los bloques que pertenecen a un grupo. De esta forma, se evita el desbalance de carga ocasionado por la progresión misma de la factorización.
Para cada uno de los métodos se presentó el resultado de una de las experimentaciones realizadas. Se puede observar una mejora en los tiempos obtenidos al utilizar el método propuesto. Particularmente, si se comparan los tiempos obtenidos con el método propuesto inicial y el mejorado, se observa una reducción de aproximadamente el 20% del tiempo total.
Luego, se obtiene un método para la distribución de bloques para la factorización LU de matrices en paralelo, orientado a clusters heterogéneos. Este método sirve para clusters heterogéneos donde no importa la "heterogeneidad", o sea, no importa la diferencia de potencias de los nodos que componen el cluster. De esta forma, también puede ser utilizado para clusters homogéneos, ya que distribuiría a todos los nodos la misma cantidad de bloques.
Este método puede también utilizarse para otras factorizaciones en paralelo, además de la factorización LU (por ejemplo, las factorizaciones QR, RQ, Cholesky, todas incluidas en LAPACK).
Por otro lado, para determinar el rendimiento de cada uno de los nodos (para obtener la potencia expresada en Mflop/s), se han utilizado diferentes algoritmos, los cuales mostraron distintos comportamientos en los distintos nodos.
Primero se utilizó la factorización LU secuencial y se pudo observar que el balance de trabajo obtenido era el esperado en función de las potencias. Sin embargo, los tiempos no eran como se esperaban, y eso se debe a que el rendimiento obtenido con la factorización LU secuencial no es

igual al rendimiento de la factorización LU secuencial por bloques. Además, esto afecta de forma distinta a los distintos tipos de nodos. Por lo tanto, se utilizó la factorización LU por bloques, la cual representa de forma más real el rendimiento de la factorización LU por bloques de forma paralela. En un futuro, se intentará formalizar las características del método de distribución de forma tal que se pueda calcular cuánto trabajo se asignará a cada nodo, la diferencia entre el trabajo asignado y el porcentaje del trabajo que debería recibir, antes de utilizar el algoritmo.

Referencias

[1] Anderson E., Z. Bai, C.Bischof, J. Demmel, J. Dongarra, J. DuCroz, 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, 1900

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

[3] Anderson T., D. Culler, D. Patterson, and de 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 Perfomormance Cluster Computing: Architectures and Systems, Vol. 1, Prentice-Hall, Upper Saddle River, NJ, USA, pp. 3-47, 1999.

[5] Barbosa J., J. Tavares and A. J. Padilha. "Linear Algebra Algorithms in a Heterogeneous Cluster of Personal Computers". Proceedings 9th Heterogeneous Computing Workshop (HCW 2000) pp: 147-159

[6] Dongarra J., J Du Croz, S. Hammarling, R. Hanson, "An extended Set of Fortran Basic Linear Subroutines", ACM Trans. Math. Soft., 14 (1), pp 1-17, 1988.

[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 Organization 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 Phisical Layer Specifications ANSI/IEEE 802.3 – IEEE Computer Society, 1985.

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

[12] Kumar V., A. Grama, A. Gupta, G. Karypis, Introducción to Parallel Computing. Desing and Analisys of Algorithms, The Benjamin/Cummings Publishing 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), Facultad 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 Parallel Matrix Multiplication”, Y. Cotronis and J. Dongarra (Eds.): EuroPVM/MPI 2001, LNCS 2131, pp 296-303, Springer-Verlag, 2001.

[15] Tinetti F. G. “Cómputo Paralelo en Redes Locales de Computadoras”, Tesis Doctoral.Universidad Autónoma de Barcelona, Facultad de Ciencias. Marzo 2004.

[16] Tinetti F. G., M. Denham, “Paralelización de la Factorización LU de Matrices en Clusters Heterogéneos”, Proceedings IX Congreso Argentino de Ciencias de la Computación (CACIC 2003), Facultad de Informática de la Universidad Nacional de La Plata, La Plata, Argentina, p 385-396. Oct. 2003.

[17] Wilkinson B., Allen M., Parallel Programing: Techniques and Applications Using Networked Workstations, Prentice-Hall, Inc., 1999.

[18] TOP500 SUPERCOMPUTER SITES disponible en http://www.top500.org/