Solución de Sistemas de Ecuaciones Ralas en Clusters de Computadoras

Fernando G. Tinetti[1,2], Walter J. Aróztegui[1], Antonio A. Quijano[1]

[(1)]Centro de Técnicas Analógico-Digitales
Facultad de Ingeniería, UNLP
48 y 116, 2do. Piso, 1900 La Plata
Argentina

[(2)]Instituto de Inv. en Informática-LIDI
Facultad de Informática, UNLP
50 y 115, 1900 La Plata
Argentina

fernando@info.unlp.edu.ar, waroz@graffiti.net, quijano@volta.ing.unlp.edu.ar

Resumen

En este artículo se presenta una estrategia de paralelización en clusters del método de Gauss-Seidel para la solución de sistemas de ecuaciones ralas. Desde el punto de vista de la solución numérica para matrices de coeficientes con poca densidad de elementos no nulos, se siguen los lineamientos estándares. Es decir: esquemas de almacenamiento especiales (solamente se almacenan los elementos no nulos) y métodos iterativos de búsqueda de solución por aproximaciones sucesivas.

Desde el punto de vista de la paralelización del procesamiento en clusters de computadoras, se siguen dos principios básicos: distribución de datos unidimensional y utilización de mensajes broadcast para toda comunicación de datos entre procesos. La distribución de datos unidimensional facilita al máximo la distribución de carga de procesamiento aún en el caso de clusters heterogéneos (que, sin embargo, no se presentan en este artículo). La utilización de mensajes broadcast para toda comunicación de datos entre procesos está directamente orientada a la optimización de rendimiento de las comunicaciones en la mayoría de los clusters instalados y que se utilizan para cómputo paralelo. La interconexión más usual de estos clusters es la provista por las redes Ethernet, y por lo tanto pueden implementar los mensajes broadcast a nivel físico con sobrecarga mínima.

Se presenta el análisis de rendimiento paralelo y además los resultados obtenidos en una red local de computadoras heterogéneas que se utilizan como si fueran homogéneas. En este caso se utiliza una implementación de la biblioteca MPI (Message Passage Interface) para la comunicación entre procesos.

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

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

1. Introducción

Muchas ramas de la ciencia y la ingeniería necesitan una considerable potencia de cálculo en el modelado y simulación de diversos fenómenos físicos. Basta citar algunos problemas, como los modelos ambientales en aplicaciones de meteorología, aerodinámica en el diseño de naves

aéreas y espaciales, hidrodinámica, circuitos electrónicos, simulación de redes eléctricas, modelado biológico y una amplia y variada gama de aplicaciones científicas típicas. Recientemente se han sumado otras áreas como las de simulación financiera y económica.

Así, muchos de estos modelos involucran un gran número de ecuaciones diferenciales parciales (EDPs) y la manera más común de resolverlas es discretizándolas, es decir, aproximándolas por ecuaciones que contienen un número finito de incógnitas. Se producen de esta manera grandes sistemas de ecuaciones lineales ralos o dispersos (sparse) [3] del tipo que se trata en este trabajo.

Para conseguir la potencia de cálculo necesaria para resolver los sistemas de ecuaciones mencionados, históricamente se han diseñado diversos tipos de supercomputadoras de funcionamiento paralelo con características variadas respecto a arquitectura, configuración, memoria, etc. [17]. Desde hace unos pocos años los avances incluídos en los microprocesadores llamados "domésticos" o de oficina y una más que aceptable relación costo-beneficio determinan la tendencia a la utilización de computadoras paralelas formadas por redes o *clusters* de PC ya sean homógeneos cuando todas las PC son del mismo tipo (que es la definición comunmente aceptada de cluster) o heterogéneos si se trata de diferentes clases [5]. Este artículo se enfoca primariamente en un grupo homogéneo conectado con una red del tipo Ethernet, que se puede encontrar en ámbitos variados y cotidianos.

Los métodos de resolución de sistemas de ecuaciones lineales pueden clasificarse en directos o indirectos, en los métodos directos las matrices que representan a los sistemas a calcular son llevados mediantes transformaciones a formas más simples y éstas calculadas directamente. Un ejemplo característico es el método de eliminación gaussiana que transforma la matriz que representa el sistema original en una de tipo triangular cuya resolución es simple y directa. Los métodos iterativos o indirectos dan lugar a una sucesión de vectores que idealmente convergen a la solución del sistema mediante la repetición de un proceso sencillo. El cálculo se detiene cuando se cuenta con una solución aproximada con cierto grado de precisión especificado de antemano o después de un determinado número de iteraciones [8] [13].

En el caso especial de los sistemas ralos o *dispersos*, en los que una proporción grande de los elementos de la matriz del sistema son ceros, los métodos iterativos son particularmente eficientes. Más aún cuando se emplean esquemas de almacenamiento especiales para matrices ralas, esto se puede verificar al considerar un sistema con una baja densidad de elementos no nulos. Utilizando un método directo, las transformaciones sobre ceros pueden producir elementos distintos de cero, por lo que debe trabajarse con el sistema como si fuera denso [6]. Al utilizar un método iterativo, los coeficientes nulos se mantendrán invariables y puede prescindirse de ellos, con el resultante ahorro de espacio en memoria y tiempo de procesamiento al no tener que acceder ni utilizar operandos nulos.

En las sección que sigue (sección 2), se mostrarán los métodos de almacenamiento de matrices ralas y el método iterativo Gauss-Seidel para resolución de ecuaciones lineales. La sección 3 introduce la paralelización del método iterativo directamente orientada a clusters interconectados por redes Ethernet. La sección 4 muestra los resultados obtenidos en la experimentación sobre un cluster en particular y sobre el final se dan las conclusiones y trabajos futuros (sección 5), así como la bibliografía a la que se hace referencia a lo largo del trabajo.

2. Sistemas Ralos: Almacenamiento y Método de Gauss-Seidel

En esta sección inicialmente se muestra cómo ahorrar espacio en el almacenamiento de matrices con una gran proporción de elementos nulos. Posteriormente, se muestra cómo se resuelven los sistemas de ecuaciones con el método Gauss-Seidel y en qué medida el método se ve afectado por el sistema de almacenamiento de matrices ralas.

2.1. Almacenamiento de Matrices Ralas o Dispersas

De estos esquemas de almacenamiento, el más simple es el llamado de formato coordinado, que consta de una estructura de datos con tres arreglos:

- Un arreglo real conteniendo todos los valores (reales o complejos) de los elementos distintos de cero de la matriz en cualquier orden.
- Un arreglo entero conteniendo los índices de fila.
- Un segundo arreglo entero conteniendo los índices de columna.

Los tres arreglos de largo igual a la cantidad de elementos distintos de cero de la matriz, que se denominará Nz. Como los elementos usualmente se listan por filas o por columnas, uno de los arreglos con índices mencionados antes puede contener información redundante. Teniendo esto en cuenta, se llega al formato de almacenamiento que es probablemente el más popular en matrices ralas y que se conoce como CSR (Compressed Sparse Row) o formato de fila rala comprimida [14]. En este caso se tienen también tres arreglos, pero ahora son:

- Un arreglo conteniendo los elementos no nulos de la matriz, almacenados por fila.
- Un arreglo entero conteniendo los índices de columna respectivos.
- Un arreglo entero conteniendo los punteros al inicio de cada fila.

De esta manera se tienen dos arreglos de longitud Nz y otro de longitud igual $n+1$ con n igual al número de filas (ecuaciones) y siendo el último elemento de éste el que indica la cantidad total de elementos no nulos más uno. Si, por ejemplo, se tiene el sistema de ecuaciones

$$\begin{pmatrix} 2{,}04 & 0 & 1 & 2{,}3 \\ 0 & 0 & 3{,}2 & 0 \\ 0 & -1 & -6 & 0 \\ 4 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 40{,}8 \\ 0{,}8 \\ 0{,}8 \\ 200{,}8 \end{pmatrix}$$

La matriz de coeficientes será almacenada en los tres vectores que se muestran a continuación:

$$\begin{array}{rcl} AA & = & (\ 2{,}04 \quad 1 \quad 2{,}3 \quad 3{,}2 \quad -1 \quad -6 \quad 4 \quad -1\ ) \\ JA & = & (\ 1 \quad 3 \quad 4 \quad 3 \quad 2 \quad 3 \quad 1 \quad 3\ ) \\ IA & = & (\ 1 \quad 4 \quad 5 \quad 7 \quad 9\ ) \end{array}$$

donde AA contiene los coeficientes no nulos, JA contiene los índices de columna de cada elemento no nulo, y el arreglo IA contiene los índices de comienzo de cada fila. Otras formas de almacenamiento de matrices ralas son:

- CSC (Compressed Sparse Column), que es una forma análoga a la CSR, y que utiliza la herramienta MatLab, por ejemplo [11].
- MSR (Modified Sparse Row) o fila rala modificada que se orienta a aprovechar que generalmente los elementos de la diagonal principal de la matriz son distintos de cero y que se accede a ellos con más frecuencia que al resto de elementos [14].
- Por diagonales, para las matrices estructuradas diagonalmente, es decir con sus elementos no nulos concentrados en unas pocas diagonales [14] [6].
- Ellpac-Itpack, que es un esquema general y popular en las máquinas vectoriales [14].

2.2. Método Iterativo de Gauss-Seidel

Los métodos iterativos intentan encontrar la solución del sistema a partir de un vector solución inicial $x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}$ que generalmente se toma como cero, de manera que se producen paso a paso valores mejorados de x_i. Este proceso se repite hasta llegar a lo que se considera una buena aproximación. Asumiendo que los coeficientes del sistema de ecuaciones son denotados como a_{ij} y b_i es el resultado buscado de cada ecuación del sistema, una expresión más general para este procedimiento es:

$$x_i^{(k)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k-1)} \right) \tag{1}$$

que como mínimo requiere que $a_{ii} \neq 0$ y constituye el método de Jacobi, en el que en cada aproximación se calcula con los datos de la iteración anterior.

En el método de Gauss-Seidel este proceso iterativo se modifica de manera que las ecuaciones utilicen los valores de la incógnitas calculados en las ecuaciones anteriores, correspondientes a la actual iteración. Esto produce una considerable mejora en la convergencia y se expresa de la forma:

$$x_i^{(k)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x_j^{(k)} - \sum_{j>i} a_{ij} x_j^{(k-1)} \right) \tag{2}$$

Tanto el método de Jacobi como el de Gauss-Seidel tienen condiciones similares para la convergencia para cualquier solución inicial, aunque normalmente el método de Gauss-Seidel requiere significativamente menor cantidad de iteraciones, es decir valores de k en la Ec. (2). También para los dos métodos iterativos, el cálculo del error suele utilizar el cálculo de las diferencias entre valores sucesivos de las variables, es decir:

$$\Delta x_i = |x_i^{(k)} - x_i^{(k-1)}| \tag{3}$$

Se debe hacer notar que ninguno de estos métodos se ve alterado por la forma de almacenamiento de la matriz de coeficientes. Por supuesto que el acceso a los datos en el caso del almacenamiento CSR para matrices ralas debe ser realizado utilizando los vectores mencionados antes, pero esto no cambia las características numéricas del algoritmo (cantidad de iteraciones y valores del vector solución) sino la cantidad de operaciones realizadas dado que no se utilizan los valores nulos.

En este artículo se presenta la paralelización del método de Gauss-Seidel por varias razones:

- Es uno de los más representativos entre los algoritmos iterativos más simples.

- Es una evolución del método de Jacobi, con mejor tiempo de convergencia, y puede tomarse como un caso particular de otro procedimiento muy conocido como es el de sobrerrelajación SOR (Successive Overrelaxation)

- La dependencia de datos en los cálculos dada en la Ec. (2) es suficientemente fuerte como para no estar en el mejor de los casos para los requerimientos de cómputo, como es el caso del método de Jacobi, donde una vez que se tiene una solución los cálculos para la siguiente solución son todos independientes entre sí y por lo tanto su paralelización es inmediata.

3. Paralelización de Gauss-Seidel

Como se menciona antes, la paralelización del método de Gauss-Seidel no es inmediata dada la dependencia de datos que existe para calcular los valores de cada iteración. En general, para matrices ralas los esfuerzos se centran no tanto en la paralelización como en el pre-ordenamiento de las matrices para llevarlas a formas que puedan ser implementadas en paralelo de una manera eficiente. De esta manera se aprovechan las condiciones inherentes a la velocidad de convergencia de Gauss-Seidel y los elementos nulos que no se utilizan. Algunos casos que se pueden mencionar son el ordenamiento en bloques multicolores independientes entre sí [14] o los que producen las matrices llamadas flecha (Block-Diagonal-Bordered-Sparse) [10] [9].

Además de utilizar almacenamiento específicamente orientado a los sistemas de ecuaciones ralos, la paralelización que se propone en este artículo sigue los principios delineados en [15], que se pueden resumir en:

- Modelo de programación por pasaje de mensajes, que es el considerado más apropiado para el procesamiento paralelo en clusters.

- Modelo de procesamiento SPMD (Single Program - Multiple Data), que es sencillo y escalable. Por otro lado, la gran mayoría de los algoritmos paralelos para problemas de álgebra lineal siguen este modelo de procesamiento.

- Distribución de datos unidimensional, es decir que todos los procesos/procesadores se consideran interconectados en un arreglo lineal o por un bus de comunicaciones. Se tiende a pensar en un bus siguiendo la definición misma del estándar Ethernet de interconexión de computadoras en una red local.

- Comunicación entre procesos con mensajes *broadcast* únicamente siempre que sea posible, es decir que todo intercambio de datos se piensa en función de un proceso que envía y que todos los demás reciben los datos enviados. También se tiende a la optimización de la implementación de esta primitiva de comunicaciones sobre el hardware proporcionado por el estándar Ethernet [7], que es inmediato.

El método de almacenamiento elegido es el CSR (Compressed Sparse Row) mencionado antes. La partición o distribución de los datos a procesadores entre los procesos o procesadores se realiza por bloques de filas en partes iguales (en este caso, el cluster se toma como homogéneo). La Fig. 1 muestra esquemáticamente la partición de un sistema de ecuaciones entre tres procesos P_0, P_1 y P_2. El proceso P_0 tendrá asignado el primer tercio de ecuaciones (filas de la matriz de coeficientes), el proceso P_1 el segundo tercio y el proceso P_2 tendrá el tercer y último tercio de filas de la matriz de coeficientes del sistema de ecuaciones. Debe recordarse que, por un lado,

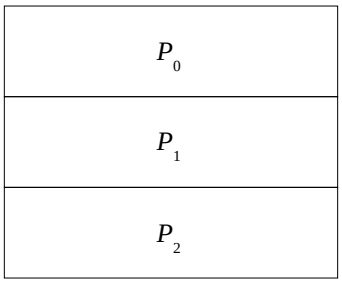


Figura 1: Distribución de Bloques de Filas en Tres Procesos.

esta partición es independiente del almacenamiento CSR de una matriz rala, y por otro lado que, dado un bloque de filas, queda automáticamente determinado el conjunto de variables que el proceso puede calcular. Más específicamente, cada proceso puede calcular el valor de las variables que están asociadas a la diagonal principal de la matriz (rala) de coeficientes, tal como puede derivarse de la Ec. (2). Una vez identificadas las filas de cada proceso, cada uno se encarga de dividir las filas asignadas por el elemento correspondiente a la diagonal principal respectiva. El sistema quedará:

$$x_i = b_i^* - \sum_{j \neq i} a_{ij}^* x_j; \qquad b_i^* = \frac{b_i}{a_{ii}}; \quad a_{ij}^* = \frac{a_{ij}}{a_{ii}}$$

A partir de ahora empieza el proceso iterativo, que se puede separar en varias etapas, y que conviene describir desde la primera iteración:

1.- Inicialmente, es decir con un valor inicial para las incógnitas, el proceso con el primer bloque de filas puede comenzar con el cálculo de las variables asociadas. Es decir que el proceso P_0 con las primeras bs filas puede determinar el valor de las variables $x_1^{(1)}, \ldots, x_{bs}^{(1)}$ directamente a partir de $x_1^{(0)}, \ldots, x_n^{(0)}$. Los demás procesos solamente pueden llevar a cabo los resultados intermedios que corresponden a la segunda sumatoria que aparece en la Ec. (2), es decir que dependen de los valores $x_i^{(0)}$.

2.- Una vez que se tienen calculados los valores $x_1^{(1)}, \ldots, x_{bs}^{(1)}$, todos los demás procesos pueden utilizar estos valores para el cálculo de la primera sumatoria de la Ec. (2). Dado que el proceso P_0 es el que tiene los valores de estas variables y todos los demás los necesitan, es *natural* pensar en un broadcast desde el proceso P_0 a todos los demás. De hecho, en este segundo paso se completan todos los cálculos necesarios para los valores de $x_{bs+1}^{(1)}, \ldots, x_{2bs}^{(1)}$ en el proceso P_1 que, a su vez, pueden ser enviados a todos los demás procesos para los cálculos que dependen de ellos de acuerdo con la Ec. (2). Se debe notar que estos valores $x_{bs+1}^{(1)}, \ldots, x_{2bs}^{(1)}$ se pueden utilizar también en el proceso P_0, pero para el cálculo de las variables de la siguiente iteración de Gauss-Seidel, es decir para el cálculo parcial de los valores $x_1^{(2)}, \ldots, x_{bs}^{(2)}$.

3.- Los dos pasos anteriores se pueden continuar (cálculos parciales llegando a los valores de cada conjunto de variables en los procesos correspondientes) tomando como referencia los procesos P_i hasta el último proceso con el último bloque de filas/variables, donde:

- Se tienen los valores para todas las variables de una iteración de Gauss-Seidel (la primera, siguiendo con ejemplo con $k = 1$).

- Si se calculan los errores junto con las variables de acuerdo con la Ec. (3) se puede tener el error total correspondiente a la iteración *actual* de Gauss-Seidel y por lo tanto se puede determinar si seguir o no, de acuerdo con el valor de este error.

Desde el punto de vista de cada proceso, a partir de la inicialización se sigue una secuencia de recepción de datos provenientes de los demás procesos (vía broadcast), cálculos parciales con estos datos, envío de los valores de las variables que se manejan localmente una vez que se tienen completamente calculados (vía broadcast) y determinación de continuar o no de acuerdo al error o a una señal del último proceso que lo indica.

3.1. Características de Rendimiento del Algoritmo Paralelo

Solamente se resumirán los puntos más relevantes con respecto al rendimiento paralelo del algoritmo, para evitar demasiados detalles que se pueden aclarar directamente en los experimentos de la sección siguiente. Una posiblidad de análisis de rendimiento del algoritmo propuesto puede ser la de asumir que las comunicaciones no tienen ningún peso de tiempo de ejecución. Este sería el mejor caso posible y, de hecho, no es posible en una implementación real (los datos estarían *instantáneamente* disponibles en todas las computadoras). Sin embargo, da como resultado el óptimo absoluto o cota máxima de rendimiento que es de esperar en una implementación real.

Teniendo en cuenta la descripción anterior, la simultaneidad de los cálculos se da al utilizar los valores obtenidos para un subconjunto de variables en las dependencias determinadas para las demás. Siguiendo el punto **1.-** del ejemplo anterior, una vez que el proceso P_0 calcula los valores $x_1^{(1)}, \ldots, x_{bs}^{(1)}$ y los comunica a todos los demás, $P_1, \ldots, P_{p-1}$ (asumiendo que son p procesadores/computadoras en total), todos éstos pueden computar simultáneamente con los datos obtenidos de P_0. A partir de este punto el algoritmo es básicamente igual: un proceso envía los valores de bs variables y todos los demás ($p-1$ procesos) los utilizan para sus cálculos parciales. Esto significa que en todo tiempo hay a lo sumo $p-1$ procesadores computando simultáneamente. Siguiendo con la idea de considerar que los mensajes broadcast no tienen ningún peso de tiempo de ejecución, el speedup máximo del algoritmo es, claramente, $p-1$ para p procesadores.

Aunque tan claro como idealizado, este speedup óptimo puede llegar a superarse en algunas circunstancias. En [4] se comentan algunos obstáculos significativos para la obtención de rendimiento óptimo en el contexto de los sistemas de ecuaciones ralos directamente relacionados con el rendimiento secuencial:

- Falta de regularidad y localidad en el acceso a los datos. El mayor problema radica en la utilización de los esquemas de almacenamiento de matrices ralas que, como se mostró específicamente para el caso del esquema CSR, tiene tres arreglos de datos y requiere el acceso indirecto a los datos de la matriz de coeficientes.

- La operación básica a optimizar es la de la multiplicación de una matriz por un vector, dado que es la principal operación a utilizar en los métodos iterativos de resolución de sistemas de ecuaciones. La diferencia de rendimiento con los sistemas densos suele ser muy grande, dado que en los sistemas densos la operación utilizada con mayor frecuencia es la de multiplicación de matrices.

Por lo tanto, en el contexto de los sistemas ralos la localidad juega un papel preponderante en el rendimiento obtenido ¿Cómo puede afectar esto al rendimiento paralelo? En cierta forma está relacionado con lo que se comenta más extensamente en [16]: cuando el sistema de ecuaciones se mantiene constante, a mayor distribución de los datos (entre mayor cantidad de computadoras), la cantidad de datos que maneja cada una de ellas es proporcionalmente menor

y, por lo tanto, la localidad crece junto con la cantidad de computadoras utilizadas. Esto significa que si no se escala el problema junto con la cantidad de datos se obtiene ganancia de rendimiento no solamente por el cómputo simultáneo sino también por el cómputo local (secuencial) más veloz. Expresado de otra forma, se tienen dos fuentes de ganancia de rendimiento: cómputo local más rápido y cómputo simultáneo en varias computadoras. De esta manera se puede llegar a superar el speedup óptimo absoluto, pero se debe recordar que esto sucede por considerar el rendimiento en términos de speedup con tamaño fijo del problema, quizás adoptando la misma visión que la llamada "Ley de Amdhal" [1].

4. Experimentación y Resultados Obtenidos

En términos de hardware, la implementación se llevó a cabo en una red de PCs interconectadas con Ethernet de 10Mb/s. Todas las PCs se utilizaron con el sistema operativo Linux y la implementación LAM (Local Area Multicomputer) [2] de la biblioteca estándar MPI (Message Passage Interface) [12] para pasaje de mensajes entre procesos.

De la biblioteca MPI solamente se usó la función broadcast, dado que de hecho el algoritmo fue concebido para que esto sea posible, prescindiendo eventualmente de la utilización de un switch en la red, que es el único recurso con que otros métodos pueden eludir las colisiones que aumentan y se hacen importantes al crecer la dimensión de los sistemas a calcular.

A medida que los sistemas crecen en tamaño, el costo de comunicaciones también aumenta, y uno de los propósitos de la experimentación es, justamente, determinar el peso relativo de las comunicaciones con respecto al cómputo para este algoritmo. Más específicamente, se debe determinar el rendimiento para distintos tamaños del sistema de ecuaciones con diferentes cantidades de computadoras. Siempre el objetivo es obtener mayor simultaneidad en los cálculos cuando se utiliza mayor cantidad de computadoras. Más específicamente, se espera *a priori* obtener mayor rendimiento (*speedup* y eficiencia) con mayor cantidad de máquinas utilizadas y tamaños relativamente grandes de sistemas de ecuaciones.

Se utilizaron diferentes cantidades de computadoras y diferentes tamaños de sistemas de ecuaciones con un porcentaje de densidad (valores no nulos) de la matriz de coeficientes, y el rendimiento se determina por la métrica conocida de *speedup*. Resumiendo los experimentos:

- Se utilizaron entre una (procesamiento secuencial) y cinco computadoras en paralelo que es la máxima cantidad de PCs disponibles para este trabajo de investigación. Aunque todas las computadoras son diferentes entre sí (el cluster es heterogéneo) se utilizaron como si fueran homogéneas.

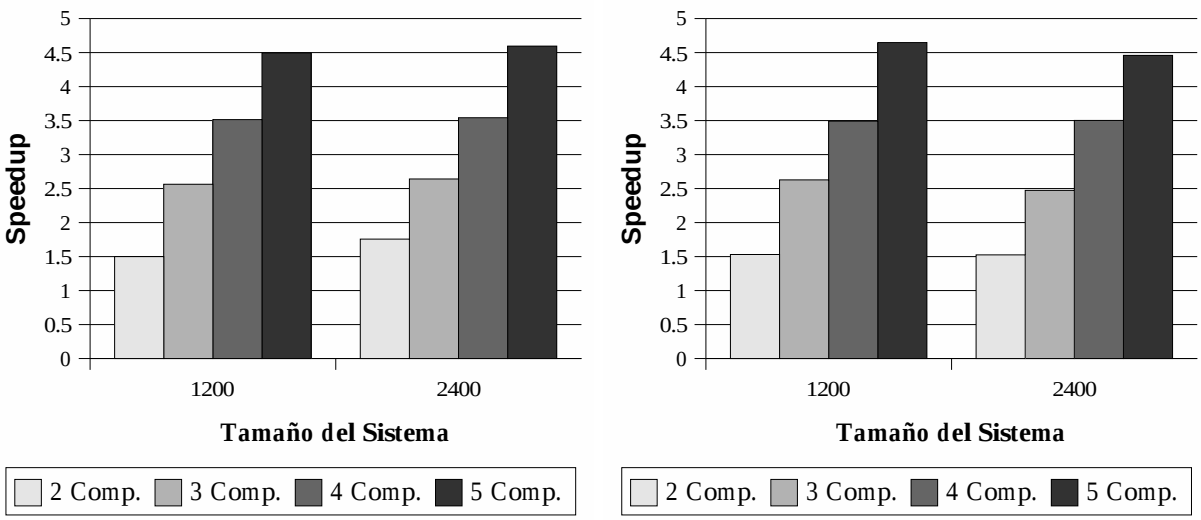
- Los tamaños de sistemas de ecuaciones fueron: 1200 y 2400 ecuaciones con las respectivas incógnitas. El mínimo está dado por la cantidad mínima de ecuaciones que tiene sentido paralelizar (el tiempo secuencial es suficientemente grande). El máximo está relacionado con la cantidad máxima de datos que una computadora puede contener en memoria principal sin hacer uso de la memoria *swap* y también con el tiempo de los experimentos (para que no sea excesivo).

- El speedup se calcula con respecto al procesamiento secuencial sin hacer uso de ninguna primitiva de pasaje de mensajes. Es decir que no es afectado por ninguna sobercarga que tenga relación con la paralelización del algoritmo propuesta. Además, el valor de referencia de tiempo de cómputo secuencial se toma en la más lenta de todas las computadoras

utilizadas, es decir que se consideran todas las computadoras como la de menor capacidad de cómputo.

- Las densidades se fijaron en 20 % y 50 %, es decir que la matriz de coeficientes tiene el 80 % y el 50 % de los valores iguales a cero respectivamente.

La Fig. 2 muestra los resultados obtenidos en la experimentación para todas las variantes que se describen antes. Sobre el eje x se tienen los dos tamaños de sistemas de ecuaciones con los que se experimentó (1200 y 2400 ecuaciones e incógnitas), sobre el eje y se muestra el speedup obtenido en los experimentos, y se muestran con barras cuatro series de valores, cada una de ellas correspondientes a la utilización de 2, 3, 4 y 5 computadoras respectivamente.



a) Densidad de 20 %

b) Densidad de 50 %

Figura 2: Rendimiento con Diferentes Tamaños y Cantidades de Computadoras.

Los resultados obtenidos son casi idénticos para las dos densidades con las que se experimentó: 20 % y 50 %, con lo que se puede hacer el análisis de resultados sin hacer referencia a la densidad el sistema de ecuaciones. En todos los casos se puede ver que se supera el speedup óptimo esperable por el algoritmo. De acuerdo con el análisis realizado en la sección 3.1, el óptimo para este algoritmo está dado por $p - 1$, asumiendo que p es la cantidad total de computadoras. Sin embargo, independientemente del tamaño del sistema como de la cantidad de computadoras este óptimo se supera. Esto significa que la reducción en la cantidad de datos que cada computadora procesa mejora la localidad y por lo tanto el rendimiento secuencial en cada una de ellas. Además, se debe recordar que las computadoras utilizadas son heterogéneas y que el tiempo de cómputo secuencial de referencia para el cálculo del speedup es el de la más lenta. Esto produce que los tiempos de cómputo local en cada computadora será menor que el tiempo de la más lenta. Una de las consecuencias es que, por ejemplo, cuando la computadora más lenta no interviene en los cálculos (es la que envía los valores de las variables a utilizar en todas las demás), el tiempo de cómputo de estos cálculos parciales puede ser mucho menor que si se hicieran en la más lenta.

Más específicamente, en el caso de utilizar dos computadoras, con una que es dos veces más veloz que la otra, cada una de ellas hará el 50 % del total de procesamiento, dado que se utilizan como si fueran iguales. Esto significa que la computadora con mayor capacidad de cómputo hará sus cálculos en la mitad del tiempo de referencia que es el de la más lenta y,

por lo tanto, el speedup será mayor a 1 aunque debería ser menor o igual a uno. De alguna manera, se aprovecha *automáticamente* la mayor velocidad de cómputo de las computadoras que se utilizan. Sin embargo, desde el punto de vista más general de cómputo paralelo, sin tener en cuenta el algoritmo, en ningún caso se llega a tener speedup igual a la cantidad total de computadoras usadas que es, en general, p.

Otra de las características de rendimiento interesantes para observar en la Fig. 2 es el peso de las comunicaciones en el tiempo total de cómputo. A pesar de que el análisis previo asumió que no tienen peso en tiempo de ejecución, también se advirtió que esto no sucede en realidad. En particular, es de esperar que dependiendo de la implementación de los mensajes broadcast a mayor cantidad de computadoras el tiempo de comunicaciones sea mayor, dado que puede haber mayor cantidad de retransmisiones o mayor tiempo de sincronización para las comunicaciones. En la Fig. 2 se puede notar que aún en el sistema más pequeño (1200) y con la cantidad de computadoras mayor (5) el speedup obtenido es aceptable. Sin embargo, también en la Fig. 2 se puede observar que a igual cantidad de computadoras en paralelo el speedup obtenido es un poco mayor para el sistema de ecuaciones mayor (2400). De todas maneras, aún es de esperar que a mayor cantidad de computadoras el tiempo de comunicaciones produzca mayor penalización de rendimiento, pero se deberían verificar los resultados de experimentación con mayor cantidad de computadoras que las utilizadas para este trabajo.

5. Conclusiones y Trabajo Futuro

Se ha presentado un algoritmo paralelo del método de Gauss-Seidel para matrices ralas sobre una red local, utilizando las ventajas de un almacenamiento comprimido para tratar con este tipo de sistemas de ecuaciones de baja densidad de elementos no nulos. Se muestra vía experimentación que la implementación obtiene buenos resultados en paralelo utilizando hasta cinco máquinas, aún en una red de PCs de utilización cotidiana, con una red de muy bajo rendimiento (Ethernet 10Mb/s). Un detalle relativamente importante es que se notan las diferencias de capacidad de cómputo relativa de las computadoras en el rendimiento obtenido calculado con el speedup. Aunque no necesariamente es esencial, se nota cierto aprovechamiento de las computadoras que tienen mayor capacidad de cómputo.

También la experimentación muestra, en concordancia con trabajos anteriores [4] [16], la importancia que tiene la localidad de refernencia en el rendimiento paralelo cuando los tamaños de problemas no escalan con la cantidad de computadoras. Es importante recordar que estos casos se dan en el contexto de la reducción del tiempo de ejecución de un problema de tamaño específico. Sin embargo, también es importante recordar que no siempre teniendo mayor cantidad de computadoras se resuelve el *mismo* problema (del mismo tamaño) sino que se tiende a resolver problemas mayores.

Queda por analizar el rendimiento del algoritmo con mayor cantidad de computadoras. En este caso se puede identificar con mayor precisión los alcances en cuanto a escalabilidad, por ejemplo. Más específicamente, se debe analizar la penalización que imponen los mensajes broadcast en el rendimiento obtenido cuando se utilizan decenas o centenares de computadoras. Por supuesto, en esto caso se depende de la cantidad total de computadoras disponibles para realizar la experimentación.

También queda por analizar cómo se comporta el algoritmo presentado trabajando con varios bloques de filas por cada procesador. Por un lado, esto aumentaría la cantidad de comunicaciones, pero por el otro disminuiría el peso de cada una de éstas y además se podría contar con una concurrencia del total de los procesadores.

Por otro lado también es posible estudiar el algoritmo en el contexto de redes locales de computadoras heterogéneas. En este caso se debería balancear la carga de procesamiento de cada computadora de acuerdo con su capacidad de cómputo relativa a las demás. Esto produciría que el tiempo de ejecución para el procesamiento de los datos sea similar en todas las computadoras. De hecho, en este contexto se vería con mayor claridad que la distribución de datos unidimensional no solamente es atractiva por su simplicidad sino también porque produce una fácil adaptación del algoritmo paralelo al contexto heterogéneo.

Referencias

[1] Amdhal G., "Validity of the Single-Processor Approach to Achieving Large-Scale Computing Capabilities", Proc. 1967 AFIPS Conference, Vo. 30, p. 483, 1967.

[2] Burns G., R. Daoud, and J. Vaigl, LAM: An Open Cluster Environment for MPI. Ohio Supercomputer Center, May 1994. LAM/MPI is available at University of Notre Dame (http://www.mpi.nd.edu/lam) - 1998-2001.

[3] Chapra-Canale, Métodos Numéricos para Ingenieros, McGraw-Hill, 1995.

[4] Dongarra J., "High Performance Computing Trends and Self Adapting Numerical Software", High Performance Computing. 5th International Symposium, ISHPC 2003, Tokyo-Odaiba, Japan, October 20-22, 2003, Proceedings. Series: Lecture Notes in Computer Science, Vol. 2858. Veidenbaum, A.; Joe, K.; Amano, H.; Aiso, H. (Eds.), 2003, XV, pp. 1-9, ISBN: 3-540-20359-1.

[5] Foster, I., Designing and Building Parallel Programs, Addison-Wesley, Reading, Massachusetts, 1995.

[6] Golub G., C. Van Loan, Matrix Computations, 2nd Edition, The John Hopkins University Press, 1989.

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

[8] Kinkaid D., W. Cheney, Análisis Numérico (Las Matemáticas del Cálculo Científico), Addison-Wesley Iberoamericana, 1994.

[9] Koester D. P., "Parallel Block-Diagonal-Bordered Sparse Linear Solvers for Power Systems Applications", NPAC Technical Report SCCS-745, 1995.

[10] Koester D. P., S. Ranka, and G. C. Fox. "A Parallel Gauss-Seidel Algorithm for Sparse Power System Matrices", NPAC Technical Report SCCS 630, Northeast Parallel Architectures Center (NPAC), Syracuse University. Proceedings of the Scalable Parallel Libraries Conference. IEEE Press, 1994.

[11] MatLab 6.0 release 12, Guide, 2000.

[12] Message Passing Interface Forum, MPI: A Message Passing Interface Standard, International Journal of Supercomputer Applications, Volume 8 (3/4), 1994.

[13] Nakamura S., Métodos Numéricos aplicados con software, Prentice Hall, 1998.

[14] Saad Y., Iterative Methods for Sparse Linear Systems, 2nd edition, Society for Industrial and Applied Mathematic, 2003.

[15] Tinetti F. G., Cómputo Paralelo en Redes Locales de Computadoras, Tesis Doctoral (Doctorado en Informática) de la Universidad Autónoma de Barcelona, Marzo de 2004.

[16] Tinetti F. G. , M. Denham, "Paralelización y Speedup Superlineal en Supercomputadoras. Ejemplo con Multiplicación de Matrices", Proceedings VII Congreso Argentino de Ciencias de la Computación (CACIC), El Calafate, Santa Cruz, Argentina, 16 al 20 de Octubre de 2001, Tomo II, pp. 765-774.

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