GEOS, Unión Geofísica Mexicana, A.C., Septiembre 1998

PROGRAMACIÓN PARALELA EN ESTACIONES DE TRABAJO DE UN MODELO NUMÉRICO DE UN ACUÍFERO CONFINADO RESUELTO POR ELEMENTO FINITO

José Gómez-Valdés y Roberto Soto-Amador
CICESE, Km 107 Carretera Tijuana-Ensenada, Ensenada, B. C., México

RESUMEN

Con la finalidad de analizar el rendimiento del paquete Parallel Virtual Machine (PVM) en un red local de estaciones de trabajo con poder de cómputo heterogéneo, se presenta un ejercicio de programación paralela. Se trabaja con un modelo matemático de un acuífero confinado homogéneo isotrópico con flujo en una dimensión. Se usa el método de elemento finito para discretizar el dominio del flujo en elementos rectangulares y se resuelve numéricamente la ecuación diferencial parcial que describe el flujo en el acuífero, utilizando el método de Galerkin. Se obtiene un sistema de ecuaciones algebraicas que se resuelven con métodos iterativos. Se hace un programa para computadora en lenguaje C para cada método iterativo usado. Se diseña al algoritmo paralelo según el paradigma maestro-esclavo. En el caso del método de Jacobi, bastó con descomponer el dominio en partes iguales para obtener una eficiencia aceptable. En cambio con el método de Gauss-Seidal, además de hacer descomposición del dominio resultó necesario paralelizar el avance de las iteraciones. Aunque la red local que se utilizó en este trabajo es de uso múltiple y no fue diseñada para hacer cómputo paralelo, se encuentra lo esperado, esto es, la salida del código paralelo es igual a la salida del código secuencial y se reduce significativamente el tiempo de ejecución al incrementar el número de computadoras. Con el uso del método de Gauss-Seidal se alcanza una acelaración más alta que con el método de Jacobi.

DESCRIPCIÓN DEL PROBLEMA

Construir un sistema de ecuaciones algebraicas según el método de elemento finito apartir de un modelo matemático del flujo de agua dependiente del tiempo en un acuífero confinado, el cual consiste de un dominio rectangular de material poroso en un flujo saturado, homogéneo, isotrópico y de fondo rocoso e impermeable. Implantar en lenguaje de programación C el algoritmo secuencial. Enseguida, con el uso del paquete PVM (Parallel Virtual Machine) hacer un programa en C que se ejecute en paralelo en estaciones de trabajo en redes locales. La solución numérica del algoritmo paralelo y del algoritmo secuencial deberán de ser idénticas. Medir la eficiencia tanto del algoritmo paralelo como de la red local.

FORMULACIÓN DEL PROBLEMA

MODELO MATEMÁTICO

El modelo matemático de agua subterránea en un acuífero se obtiene a partir del principio de conservación de masa y de la ley de Darcy. Si se tiene que el agua subterránea es incompresible, conservación de volumen se obtiene de la igualdad entre el volumen de agua que sale del acuífero y el volumen de agua que entra más la razón en la cual el nivel de agua cambia con el tiempo. La ley de Darcy, por otro lado, establece que la razón de flujo de volumen por unidad de área es directamente proporcional al gradiente hidráulico. La constante de proporcionalidad es la conductividad hidráulica del medio, que depende del tipo de material que conforma el acuífero.

Aplicando la ley de Darcy y conservación de volumen en el caso de un acuífero confinado homogéneo e isotrópico con flujo bidimensional variable en el tiempo, sin fuente de carga o descarga, se obtiene

$$\frac{\partial^2 h}{\partial x^2} + \frac{\partial^2 h}{\partial y^2} = \frac{S}{T}\frac{\partial h}{\partial t}, \qquad (1)$$

donde $h(x,y,t)$ es la carga hidráulica, x, y son las coordenadas cartesianas, t el tiempo, S es el coeficiente de almacenamiento y T es la transmisividad. La transmisividad del acuífero es el producto de su conductividad hidráulica por su espesor.

La Eq. 1 es una versión de la ecuación de difusión.

ELEMENTO FINITO

El método de elemento finito, que se ajusta bien a cualquier tipo de dominio, se ha usado bastante en el modelado de acuíferos, sobre todo en problemas de transporte de contaminantes (Barry, 1990). Con el fin de obtener la mejor aproximación posible a la geometría del problema físico, el dominio espacial se divide en subdominios (elementos finitos) de forma poligonal, por lo general triángulos o cuadriláteros, los cuales son especificados por un nodo en cada vértice. Estos nodos son los puntos internos del dominio del problema en donde $h(x, y, t)$ es calculada. El método permite que $h(x, y, t)$ sea definida dentro de cada elemento en términos de los valores nodales por medio de funciones de interpolación (las funciones base). Cada función base se asocia a un nodo. Después, se establece un conjunto de ecuaciones integro-diferenciales siguiendo la técnica de Galerkin. De esta manera, se genera un sistema de ecuaciones

lineales, cuya solución se puede encontrar por métodos directos o métodos iterativos.

Método de Galerkin

Se pueden usar otras técnicas de elemento finito en la búsqueda de la solución numérica de la Eq. 1 (ver por ejemplo Zienkiewicz, 1977). Nosotros empleamos la técnica de Galerkin, por su consistencia con el método variacional, ampliamente utilizado en sistemas físicos. Enseguida se detalla la aplicación de ésta técnica (más información se puede encontrar en Wang y Anderson (1982)).

Se inicia proponiendo una solución aproximante $\hat{h}(x,y,t)$ de la forma

$$\hat{h}(x, y, t) = \sum_{l=1}^{n} h_l(t) N_l(x, y), \tag{2}$$

donde $N_l(x, y)$ es la función base asociada con el nodo l y n es el número total de nodos en el dominio del problema. Para simplificar, de aquí en adelante $\hat{h}(x,y,t) = \hat{h}$ y $N_l(x, y) = N_l$.

La Eq. 2 se sustituye en la Eq. 1 y se pide que el residual pesado con las funciones base sea nulo, es decir,

$$\iint_D \left(\frac{\partial^2 \hat{h}}{\partial x^2} + \frac{\partial^2 \hat{h}}{\partial y^2} - \frac{S}{T}\frac{\partial \hat{h}}{\partial t}\right) N_l \, dx dy = 0, \tag{3}$$

donde $l = 1,..., n$ y D es el dominio del problema. La integral indica que hay una ecuación para cada nodo.

Para reducir el orden de las derivadas, y así evitar problemas de discontinuidades, se integra por partes la Eq. 3, esto es,

$$\iint_D \left(\frac{\partial \hat{h}}{\partial x}\frac{\partial N_l}{\partial x} + \frac{\partial \hat{h}}{\partial y}\frac{\partial N_l}{\partial y}\right) dx\, dy + \iint_D \frac{S}{T}\frac{\partial \hat{h}}{\partial t} N_l \, dx\, dy$$

$$= \int_\Gamma \left(\frac{\partial \hat{h}}{\partial x} n_x + \frac{\partial \hat{h}}{\partial y} n_y\right) N_l \, d\sigma, \tag{4}$$

donde Γ es la frontera del dominio del problema, n_x y n_y son las componentes del vector normal hacia fuera de la frontera y σ es la longitud de arco.

Enseguida se elijen las funciones base y se sustituyen en la Eq. 4. Debido a que el dominio es rectangular, se justifica usar elementos rectangulares, pero no se recomiendan para problemas con fronteras irregulares, en los que es más apropiado usar elementos triangulares. Las integrales son evaluadas elemento a elemento por cuadratura Gaussiana. Con notación matricial podemos expresar más claramente el resultado de estas acciones en la forma

$$G\bar{h} + P\frac{\partial \bar{h}}{\partial t} = \bar{f}, \tag{5}$$

donde G y P son las matrices de los coeficientes de las ecuaciones algebraicas que resultan de sustituir las funciones base en los integrandos del lado izquierdo de la Eq. 4, $\bar{h}$ es el vector columna solución y $\bar{f}$ es el vector columna que resulta de la integral de línea del lado derecho.

Existen varios esquemas para resolver la parte temporal de la Eq. 5. Nosotros usamos una aproximación en diferencias finitas de la derivada temporal de h, y resolvemos el sistema de ecuaciones lineales en forma implícita.

Sistemas lineales

Para sistemas de ecuaciones lineales como el de la Eq. 5 con muchas incógnitas, el método directo de eliminación Gaussiana consume mucho tiempo de CPU. Esto se puede ilustrar calculando el tiempo que se requiere al factorizar una matriz de dimensión n, que es igual a $(1/3)Cn^3$, donde C es el tiempo en que una computadora hace un ciclo en el proceso de factorización. Si $n = 10{,}000$ y $C = 0.65 \times 10^{-6}$ (de una IBM 370 3090), entonces se obtiene un tiempo de CPU de 2.5 días (Hager, 1988). Los métodos iterativos son más eficientes, consumen menos tiempo de CPU, ya que van generando una secuencia de soluciones que va convergiendo a la solución del sistema de ecuaciones. Aquí se escogen por su facilidad de implantación los métodos iterativos de Jacobi y Gauss-Seidel en donde el algoritmo no cambia de iteración a iteración.

EL CASO DE UN ACUÍFERO EN DONDE EL FLUJO ES UNIDIRECCIONAL

No obstante que el modelo matemático es simple y que por ello la aplicación de la técnica de Galerkin también es sencilla, existe una rica variedad de problemas que se pueden resolver numéricamente con el material expuesto en las secciones anteriores. En este trabajo, para efectos de comparación, nosotros tomamos el mismo problema que Wang y Anderson (1982, Sec. 7.2).

El caso corresponde a un acuífero confinado homogéneo e isotrópico con flujo unidimensional en la dirección x. El espesor del acuífero es de 10 m, la forma es rectangular y tiene las siguientes características: la base va de $x = 0$ a $x = l = 100$ m, en la vertical el nivel del agua es inicialmente ($t = 0$)16 m. Repentinamente cae el nivel del agua en $x = l$, de 16 m a 11 m. Los parámetros del acuífero son $T = 0.02$ m^2 min^{-1} y $S = 0.002$. Las condiciones a la frontera son $h(0, t) = 16$ m y $h(l, t) = 11$ m para $t > 0$. Asimismo la condición inicial es $h(x, 0) = 16$ m en $0 \leq x \leq l$ (Figura 1).

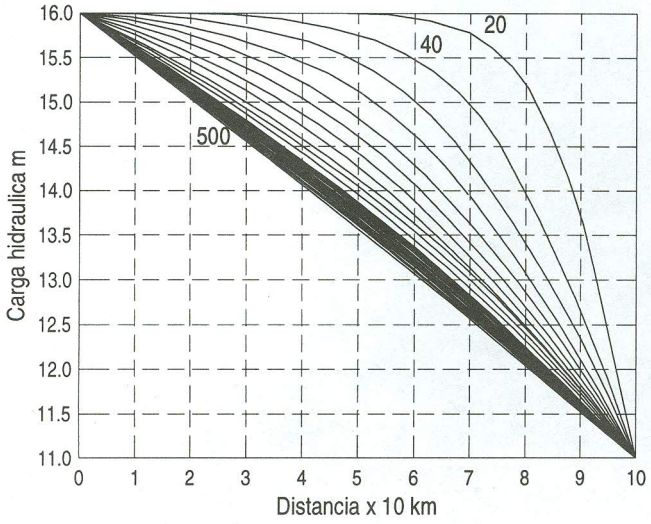

Figura 1. Dominio, elementos, condiciones iniciales y de frontera del acuífero confinado tratado en este trabajo.

ALGORITMO SECUENCIAL

Después de desarrollar la Ec. 1 por el método de elemento finito usando elementos rectangulares y siguiendo el método de Galerkin, se hace un programa secuencial en lenguaje C, estructurado en tres partes o bloques (Figura 2). Se elige C, como lenguaje de programación, por sus ventajas de portabilidad y precisión (Frez, 1997). En el primero de los bloques se definen las coordenadas nodales y se especifican los valores iniciales de los nodos interiores y de los nodos frontera con niveles de agua fijos. En el segundo bloque se construyen las matrices del sistema de ecuaciones algebraicas (G y P de la Ec. 5), se suman en secuencia las contribuciones de cada elemento y se ejecutan las integraciones por cuadratura Gaussiana. En el bloque tres se resuelve el sistema de ecuaciones algebraicas por métodos iterativos y la dependencia temporal en forma implícita. El ejecutable del programa C otorga los resultados correctos. En la Figura 3 se presenta la solución numérica para t = 20, 30,..., 300 min, usando Δt = 5 min con el método de Gauss-Seidel y se compara la solución numérica con la solución analítica para el caso estacionario; se aprecia, que a medida que t se hace más grande la solución numérica se acerca más a la solución analítica. También se hace una versión C usando el método de Jacobi, modificando sólo las declaraciones de la estructura do...while, con el cual se obtiene la misma solución numérica.

```
proc_begin main (proceso de control)
   // bloque # 1
   for_begin (geometria)
   for_end
   for_begin (condiciones iniciales)
   for_end
   for_begin (condiciones de frontera)
   for_end
   // bloque # 2
   for_begin (construcion de las matrices cuadradas G y P)
    [se genera numero de nodos del elemento k]
    [se calcula 0.5* altura y 0.5.*ancho]
     for_begin (calculo de los elementos de las matrices)
        for_begin (cuadratura gaussiana]
        for_end
     for_end
   for_end
   // bloque # 3
   for_begin (solucion del sistema lineal)
      for_begin (construccion de la matriz columna f)
      for_end
      do (metodo iterativo)
        [error = 0.0]
        [Jacobi o Gauss_Seidel]
      while [max > error]
      for_begin (actualizacion)
      for_end
      for_begin (salida del  vector solucion)
      for_end
  for_end
proc_end
```

Figura 2. Seudocódigo del problema del flujo dependiente del tiempo.

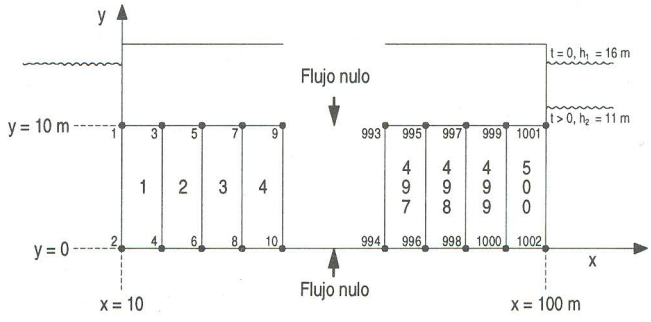

Figura 3. Solución numérica para diferentes tiempos de del problema del flujo dependiente del tiempo en un acuífero rectangular, donde Δt = 5 min.

SOFTWARE Y HARDWARE

SOFTWARE

El software PVM es un paquete de rutinas de programación paralela para usarse en computadoras en red y en computadoras paralelas. Se compone de dos partes: 1) un proceso Unix llamado **pvmd**, el que se ejecuta en todas las estaciones de trabajo, y mantiene en funcionamiento paralelo a las computadoras, y 2) una librería de subrutinas llamada **libpvm**. **pvmd,** que funciona como un controlador y enrutador, y provee el punto de contacto, el proceso de autentificación, el control de procesos y la detección de fallas. Existen dos tipos de procesos **pvmd**: a) el que corre en la computadora de control (que puede ser cualquiera de las computadoras) y el que corre en cada una de las otras. Se considera que las computadoras tienen un funcionamiento equivalente entre sí, pero sólo la computadora de control puede inicializar a las otras y agregarlas a la configuración de la red. **libpvm** contiene las subrutinas que se usan para pasar mensajes, crear procesos, sincronizar tareas y modificar la configuración. El software PVM tiene versiones para los lenguajes Fortran, C

y C++. (Más detalles se pueden encontrar en Geist et al. (1994) y en la página internet http://www.epm.ornl.gov/pvm).

Debido a que cada problema físico tiene su propio grado de paralelismo, no hay una metodología para elaborar esquemas paralelos óptimos. Sin embargo, una serie de elementos básicos de programación paralela para el modelo de pasar mensajes han sido bien identificados (véase por ejemplo Ragsdale, 1991). Entre estos elementos, uno básico es la elección de la forma de manejar el dominio del problema. Hacer una partición del mismo, en entidades iguales, ha dado muy buenos resultados en la paralelización de diversos programas Fortran en física e ingeniería. Existen dos técnicas de programación paralela en el modelo de pasar mensajes que han dominado en los últimos cinco años. Una de ellas consiste en hacer un sólo programa y cargar su ejecutable en cada CPU, y la otra en hacer dos programas: uno de los ejecutables se carga en la computadora control y el otro en las demás computadoras. Gómez-Valdés y Wang (1995) probaron que la primera técnica opera eficientemente en las computadoras paralelas de memoria distribuida Intel. Geist *et al.* (1994) recomiendan a la segunda técnica para ambientes PVM.

En la técnica de programación de dos programas, también conocida como paradigma maestro(a)-esclavo(a), el código maestro corre en la computadora de control y el código esclavo corre en las otras computadoras. El código maestro controla la inicialización y la terminación de los esclavos y recolecta resultados parciales (también puede hacer las veces de esclavo). El código esclavo hace los cálculos y las llamadas de las subrutinas de comunicación.

HARDWARE

La Figura 4 muestra la red utilizada en este estudio, la cual está instalada en el edificio de Oceanología del Centro de Investigación Científica y de Educación Superior de Ensenada (CICESE). La red cuenta con 3 procesadores SPARC SLC, 1 procesador SPARC 1+ y 2 SPARC IPC. El conducto de comunicación es ethernet, topología tubular con control CSMA/CD. Esta red fue adquirida a principios de la decada de los 90, tiene Unix como sistema operativo, los usarios son estudiantes del Departamento de Oceanografía Física, quienes usan las computadoras para procesar datos y correr modelos oceanográficos. Durante el día la carga computacional en la red es elevada, siendo mayor por la mañana. Esta red está conectada a la red principal del CICESE, lo cual implica un potencial de cómputo mayor, ya que PVM está cargado en uno de los servidores de la red principal, y hay más computadoras disponibles en otros departamentos.

ALGORITMO PARALELO

Acto seguido se pasa a construir el algoritmo paralelo PVM en lenguaje C, usando el paradigma maestro(a)-esclavo(a). También se puede usar Fortran o C++, pero la ventaja de C consiste en tener más portabilidad. En el primer bloque de

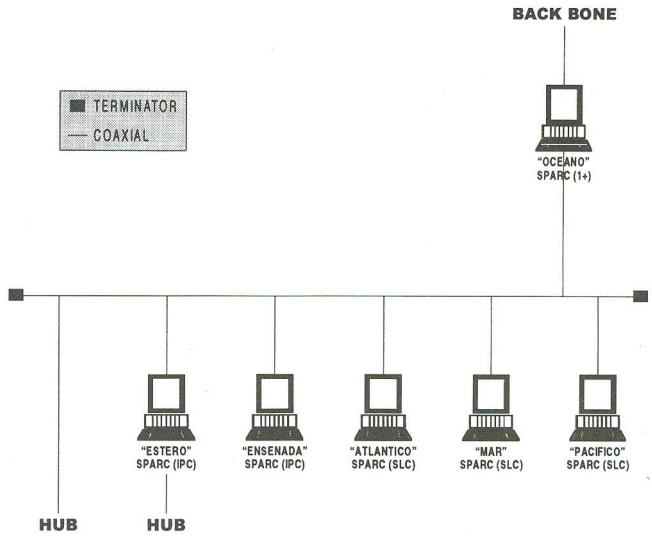

Figura 4. Red de estaciones SUN del Departamento de Oceanografía Física del CICESE, conectada a la red principal a través del servidor Sparc 1 + OCEANO.

programación actúan la computadora maestra y las computadoras esclavas. En el código maestro se inicializan las variables nnode (número de nodos) y nproc (número de computadoras), y en el código esclavo se calcula nlem = ((nnode/2)-1), número de elementos y se efectúa la asociación entre los subdominios y las computadoras. Aquí se asigna a cada computadora esclava nlem/nproc elementos; si nlem no es divisible por nproc entonces se asigna un elemento de más a una o varias computadoras. El bloque de programación número dos se desarrolla completamente en el programa esclavo, donde cada esclavo trabaja con su correspondiente subdominio. Al hacer la descomposición del dominio resulta que cada computadora esclava comparte dos nodos con las esclavas vecinas, por lo que los valores en las matrices locales en cada computadora son parciales en al menos dos renglones. Para que cada esclava trabaje con las matrices locales completas se establece, en esta porción del programa, un bloque de comunicación entre esclavas, el cual se resume como sigue:

1. La esclava r envía a la esclava r+1 los renglones que comparte con ella; al mismo tiempo la esclava r recibe los dos renglones de la esclava r-1.

2. Se completan los valores de los renglones recibidos en todas las esclavas intermedias y en la última.

3. La esclava r envía los valores actualizados a la esclava r-1; al mismo tiempo la esclava r recibe de la esclava r+1 los valores actualizados correspondientes al paso 1.

El grado de paralelización del sistema de ecuaciones lineales dependiente del tiempo que se encuentra en el bloque tres de programación (Figura 2), es función del método iterativo y del esquema para manejar el incremento en Δt. Como se había dicho anteriormente, en este problema usamos Jacobi y Gauss-Seidel

y el tiempo se resuelve implícitamente. (Más detalles se pueden encontrar en Soto-Amador (1996)).

MÉTODO DE JACOBI

Un sistema lineal como el de la Eq. 5 se puede expresar como

$$Ax = b, \tag{6}$$

donde A es una matriz cuadrada, x es el vector de incógnitas y b es el vector de los terminos independientes.

El método de Jacobi consiste en resolver localmente cada incognita de la Eq. 5. Esto significa que en cada iteración se barre todo el dominio del problema en la forma

$$x_i^{n+1} = \frac{b_i \sum_{j=1}^{i-1} a_{ij} x_j^n - \sum_{j=i+1}^{n} a_{ij} x_n^n}{a_{ii}} \tag{7}$$

donde i va de 1 a k, y n es el número de la iteración, x_j es el vector de incógnitas, a_{ij} es la matriz de coeficientes y b_i es el vector de los términos independientes. Este método se puede paralelizar fácilmente, ya que se puede hacer una partición en x_i, tal que cada computadora calcula los x_i^{n+1} asignados, usando los x_i^n propios y hace su propia actualización: $x_i^n \leftarrow x_i^{n+1}$. Después de calcular en paralelo los x_i^{n+1}, las computadoras comunican los valores nuevos a sus vecinos para poder avanzar. El cómputo y la comunicación se hacen en el programa esclavo. Como la convergencia hacia la solución es lenta con este método, resulta que al emplear más iteraciones habrá más comunicación, por lo tanto decrecerá la eficiencia del programa paralelo.

MÉTODO DE GAUSS-SEIDEL

El método de Gauss-Seidel es otra manera de resolver iterativamente la Eq. 5. En este método se usan valores actualizados tan pronto están disponibles, en la forma

$$x_i^{n+1} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{n+1} - \sum_{j=i+1}^{n} a_{ij} x_j^n}{a_{ii}}, \tag{8}$$

donde la nomenclatura es idéntica a la usada en la Eq. 7. Su grado de paralelización depende de la estructura de la matriz de coeficientes. En el caso de una matriz densa, es decir, donde la mayor parte de los elementos de a_{ij} son diferentes de cero, el algoritmo tendrá un bajo grado de paralelización, ya que no se puede hacer una partición paralela eficaz en x_j, puesto que una porción de x_j es evaluada al tiempo n y otra al $n+1$. Sin embargo, si la mayoría de los coeficientes son ceros y si los coeficientes de la matriz diferentes de cero se encuentran concentrados alrededor de la diagonal principal, en forma simétrica, entonces se puede obtener cierto grado de paralelización.

Para el caso del problema del acuífero que nos ocupa, la matriz es simétrica y en banda, con un ancho de banda de 7. Como los renglones de la matriz se pueden distribuir entre las esclavas en forma continua, la paralelización se realiza en función del número de iteraciones haciendo una descomposición del dominio. Como los métodos iterativos parten de una solución inicial, el cómputo paralelo empieza por el cálculo de la iteración 1 en la esclava 0 (E_0), la cual una vez que avanza en el proceso, estará en condiciones de enviar los valores que la esclava E_1 requiere para trabajar en la iteración 2 al mismo tiempo que E_0 continua con la iteración 1. E_1 envía los valores que la esclava E_2 requiere para trabajar en la iteración 3, al mismo tiempo que E_1 esta trabajando en la iteración 2 y E_1 en la iteración 1 y así sucesivamente. Nótese que el algoritmo tiende a ser más paralelo a medida que se incrementa el número de iteraciones. En cada una de las iteraciones hay comunicación entre las computadoras, tarea que se realiza en el programa esclavo.

EFICIENCIA

Alto rendimiento en computación en paralelo implica al menos dos cosas: reducción del tiempo total de computación e incremento del trabajo de computación en un determinado tiempo. El aumento de la velocidad de procesamiento y la eficiencia paralela son dos parámetros que regularmente se evalúan en el desarrollo de los algoritmos paralelos. En Gómez-Valdés (1996) se discute ampliamente su definición. Sin embargo, su evaluacion en redes locales heterogéneas (conformadas de computadoras diferentes) es compleja (Horie y Kuramae, 1996), debido a que alguno, varios o inclusive todos los factores siguientes pueden entrar en juego (Fox et al.,1989):

- El problema no es paralelizable.
- El aumento de la velocidad está limitado por un imbalance en el método.
- Los costos de comunicación en el algoritmo son muy altos.
- El aumento de la velocidad está limitado por la rapidez de la computadora más lenta (imbalance en la configuración).
- La velocidad de comunicación en la red es baja.
- El número de computadoras es muy grande para el problema.

La Tabla 1 muestra el tiempo de ejecución de los programas paralelos, usando 500 elementos. Los experimentos fueron realizados cuidando que la carga computacional extra sobre la red fuera mínima y usando procesadores con poder de cómputo semejante. El programa paralelo con el método de Jacobi resultó más rápido para 3 y 4 procesadores que el programa paralelo usando el método de Gauss-Seidel, pero con 5 procesadores resultó más rápido el método de Gauss-Seidel. Con este último método el programa paralelo se acerca a a la escalabilidad computacional (a medida que se agregan más computadoras la eficiencia crece).

Tabla 1. Tiempo de ejecución (en segundos (s)) del programa paralelo para el problema del flujo dependiente del tiempo usando los métodos iterativos Gauss-Seidel y Jacobi.

Procesadores	Gauss-Seidel (s)	Jacobi (s)
3	1809	1142
4	1438	1073
5	1301	1389

DISCUSIÓN

En el presente estudio se desarrollan algoritmos paralelos del problema del flujo de agua dependiente del tiempo. Se usa el paradigma maestro-esclavo. Las soluciones numéricas de los algoritmos paralelos son idénticas a las soluciones numéricas de los algoritmos secuenciales correspondientes. Dougherty (1991), por otro lado, discute las ventajes de usar computadoras paralelas en el estudio de fenómenos hidrólogicos.

El desarrollo de algoritmos paralelos y su evaluación en una red local heterogénea de estaciones de trabajo es actualmente una línea de investigación de mucho interés en computación científica. Debido a que las redes locales cuentan por lo general con procesadores diferentes, es común encontrar en ellas el problema del balanceo de la carga de los procesadores. Además cuando el número de estaciones de trabajo se incrementa, también se incrementa el número de mensajes, lo que da lugar al problema de escalabilidad. Cabe mencionar que si el algoritmo paralelo es escalable, entonces es posible adherir más computadoras, al incrementar el tamaño del dominio, sin deteriorar la eficiencia. En este estudio, el problema del balanceo de la carga de procesadores fue resuelto en parte usando procesadores con aproximadamente el mismo poder de cómputo. Aprovechando que las computadoras SPARC SLC son las más comunes de la red, fueron estas las utilizadas en los experimentos de toma de tiempos realizados. Otro punto de interés a considerar en la computación paralela en redes de computadoras es que la red no está cien por ciento disponible para este fin, sobre todo en la academia, sino que son redes de uso múltiple. De forma tal que en el proceso de desarrollo de programas en paralelo se deben compartir los recursos computacionales existentes. En este estudio, para hacer el análisis de rendimiento, se eligieron horas de trabajo de mínimo uso de computadoras. El problema de escalabilidad es más complejo, pues involucra todos los desbalances posibles: de los procesadores, de la carga computacional de cada estación de trabajo, de la carga de la red y de los problemas de paralelización de los algoritmos. En computadoras paralelas de memoria distribuida, e.g., la familia Intel, algunos problemas pueden alcanzar alta eficiencia (Gómez-Valdés y Wang, 1995), sin embargo, en redes heterogéneas es díficil alcanzar alto rendimiento (Horie y Kuramae, 1996). Es interesante señalar que en un trabajo reciente (Tapia-Armenta, 1997) se obtuvo una aceleración de 1.3 en una computadora paralela de memoria compartida usando únicamente la opción de paralelización automática de un modelo numérico del océano costero. No obstante que PVM trabaja bien en la red local de estaciones de trabajo del Departamento de Oceanografía Física del CICESE, la eficiencia es baja. Para que PVM sea de utilidad a nivel de producción es necesario conectar computadoras del mismo CPU usando otro tipo de topología y tecnología, como por ejemplo cableado con tecnología FDDI (Fiber Distributed Data Interface).

Los métodos iterativos son muy comunes en la solución de sistemas lineales. El método de Jacobi, por su sencillez, es uno de los métodos preferidos en la enseñanza (Hager, 1988). Una de las desventajas de utilizarlo radica en que converge lentamente, implicando una baja eficiencia. Pero es fácilmente paralelizable, dado que los elementos del vector solución, x_j, pueden ser calculados por partes. Sin embargo, la lenta convergencia implica mayor número de mensajes, por ello se obtiene el mayor deterioro de la eficiencia al aumentar el número de procesadores. Por lo tanto, éste método se recomienda sólo en la fase de aprendizaje del proceso de paralelización. El método de Gauss-Seidel, base de otros métodos iterativos estacionarios, e.g., la familia de los esquemas de relajación, converge más rápido que el método de Jacobi, aunque para algunos problemas su razón de convergencia es aún lenta (Barrett *et al.*, 1994).

Su paralelización no es inmediata, pero como su convergencia es más rápida, el método ofrece una ventaja al disminuir el número de mensajes. En el proceso de paralelización, se toma ventaja de la estructura de la matriz de coeficientes. Hay cierto grado de paralelización cuando la matriz, sin importar su tamaño, tiene muchos ceros, es simétrica y sus elementos distintos de cero se encuentran agrupados alrededor de la diagonal principal. Además, se hace una paralelización sobre el avance en el número de iteraciones. Aquí, usando Gauss-Seidel se encuentra la más alta aceleración de tres a cuatro estaciones de trabajo. Debido a que se trata de un ejercicio de progamación paralela, sólo se reporta la aceleración de tres a cinco procesadores.

Hoy en día resulta díficil justificar la adquisición de supercomputadoras con un sólo procesador, cuyo costo es de decenas de millones de dólares, pues con decenas de estaciones de trabajo, cuyo costo puede ser menor que un millón de dólares, se puede obtener el mismo poder de cómputo, al trabajarlas en un ambiente paralelo, evitando entre otra cosas los problemas de mantenimiento y calentamiento. Sin embargo, es necesario aprender a programar en paralelo. Por ello, se recomienda la enseñanza de los elementos básicos de la computación paralela a nivel de licenciatura y de posgrado usando una red de estaciones de trabajo, homogénea de preferencia, en un ambiente PVM o cualquier otro ambiente de paso de mensajes.

La Secretaría de Educación Pública de México (SEP), desde principios de 1998, ha proporcionado a instituciones de nivel medio y superior una gran cantidad de computadoras personales (cientos), cada una con un poder de cómputo equivalente a una estación de trabajo. Por otra parte, la propia SEP está construyendo edificios en donde se agrupa a las computadoras, ofreciendo así una buena oportunidad para introducir tecnologías de vanguardia en el proceso educativo de los jóvenes mexicanos.

AGRADECIMIENTOS

Los autores desean agradecer a Marisa Echevarría su ayuda en el proceso de revisión ortográfica del manuscrito. Rogelio Vázquez y Marco Antonio Pérez Flores hicieron un revisión crítica al manuscrito que mejoró la calidad del mismo y a Víctor M. Frías Camacho por su ayuda en la edición de las figuras. El trabajo fue financiado por CONACyT, bajo el contrato 225008-5-3523T.

REFERENCIAS

Barrett, R., M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. van der Vorst., 1994. Templates for the solution of linear systems: Building Blocks for Iterative Methods. SIAM, Philadelphia, 112 pp.

Barry, D.A. 1985. Supercomputers and their use in modeling subsurface solute transport. Reviews of Geophysics, 28, 277-295.

Dougherty, D.E., 1991. Hydrologic application of the connection machine CM-2. Water Resources Research, 27, 3137-3147.

Fox, G.M., Johnson, G., Lyzenga, S., Otto, J. Salmon and D. Walker., 1989. Solving problems on concurrent processors. Volume I: General Techniques and Regular Problems. Prentice Hall, New Jersey, 592 pp.

Frez, J., 1997. Geofísica computacional y enseñanza en el posgrado. GEOS. 17, 60-66.

Geist *et al.*, 1994. PVM: Parallel virtual machine, a users guide and tutorial for networked parallel computing. The MIT Press, Cambridge, MA, 279 pp.

Gómez-Valdés, J. and D-P. Wang, 1995. Massively parallel processing in coastal ocean circulation model. Journal of Scientific Computing, 10, 305-323.

Gómez-Valdés, J. 1996. Computación paralela en las geociencias. GEOS. 16, 137-141.

Hager, W.W., 1988. Applied numerical linear algebra. Prentice Hall, Englewood Clifs, NJ, 424 pp.

Horie, T. and H. Kuramae, 1996. Evaluation of parallel performance of large scale computing using workstation network. Computational Mechanics, 17, 234-241.

Ragsdale, S., 1991. Parallel Programming. McGraw-Hill, Inc. New York, 123 pp.

Soto-Amador, R., 1996. Computación paralela con PVM en una red local heterogénea. Tesis de Licenciatura, Universidad Autónoma de Baja California, Facultad de Ciencias, Ensenada, B.C. México, 49 pp.

Tapia-Armenta, J.J., 1997. Actualización, optimización y paralelización automática de un modelo tridimensional de la circulación del océano costero. Tesis de Maestría, Centro de Investigación y Desarrollo de Tecnología Digital, IPN, México, 78 pp.

Wang, H.F. and M.P. Anderson, 1982. Introduction to groundwater modeling: finite difference and finite element methods. W.H. Freeman and Company, NY, 237 pp.

Zienkiewicz, O.C., 1977. The finite element method. Mc Graw-Hill, London, 787 pp.