PARALELIZACION DE UN PROBLEMA DE APLICACION SOBRE UNA PLATAFORMA DE TRANSPUTADORES

Héctor Klíe[†] y Alejandro Teruel[‡]

Resumen. Se estudia la paralelización de un programa de frecuente uso en geofísica llamado DMO ("Dip-moveout"). El DMO posee tres niveles principales de paralelismo: por secciones sísmicas, por trazas sísmicas, por integración de los datos. Se implantan los dos últimos niveles en una malla de procesadores tipo transputadores ("transputers") aplicándose técnicas de descomposición de dominios para lograr un alto grado de paralelismo y analizándose sus comportamientos tanto teóricos como experimentales. El caso de la integración de los datos se reformula como un producto matriz-vector, en donde se presentan una serie de alternativas derivadas de los diferentes esquemas secuenciales: Fila, columna y bloque. El primero de estos aportó los mejores resultados desde el punto de vista tanto teórico como experimental.

1. Introducción

El "dip-moveout" (DMO) es una aplicación geofísica que involucra grandes cantidades de datos y adicionalmente, consume mucho tiempo de procesamiento. Entre los métodos para efectuar el DMO se destaca el de Hale [10][11], el cual se puede enunciar a través de las siguientes ecuaciones:

$$p_0(\omega_0, k, h) = \int \mathrm{A}^{-1} e^{i\omega_0 t \mathrm{A}} dt \int p(t, x, h) e^{-ikx} dx \tag{1}$$

$$p_0(t, x, h) = \frac{1}{4\pi^2} \int e^{-i\omega_0 t} d\omega_0 \int p_0(\omega_0, k, h) e^{ikx} dk \tag{2}$$

donde $A = \left(1 + \frac{k^2 h^2}{\omega_0^2 t^2}\right)^{\frac{1}{2}}$; $p(t, x, h)$ representa el sismograma original; $p_0(t, x, h)$, el sismograma corregido; t, el tiempo; x, la distancia y h el "offset".

La ecuación (1) indica una transformada de Fourier sobre x, y luego una integración sobre t para todo ω_0 y k. La ecuación (2) se traduce en una doble transformada de Fourier inversa sobre k y ω_0. Para realizar las transformadas de Fourier se utiliza el conocido algoritmo de la transformada rápida de Fourier (FFT), cuyo orden es $n \ln n$ para un conjunto de n datos.

[†] Departamento de Informática, Sección de Tecnología, INTEVEP S.A.

[‡] Departamento de Computación y Tecnología de la Información, Universidad Simón Bolívar

El cálculo del DMO es computacionalmente muy costoso, debido a la integral sobre t, cuyo orden es n^2. Se han realizado intentos para eliminar la componente computacional de dicha integral [11]. Sin embargo, los esfuerzos al respecto han conducido a simplificaciones que perjudican notablemente la precisión del método. En la Fig. 1 se muestra en detalle el fragmento correspondiente a la integración en el DMO. Nótese que la complejidad del fragmento, y por ende del algoritmo, está dado por $O(H*N*M^2)$.

```
/* N es el numero de trazas sismicas */
/* M es el numero de muestras */
/* H es el numero de secciones sismicas */
Para cada "offset" h perteneciente a {h1,..hH}
....
aux:= SQR(M / (dx * N))
/* Ciclo sobre el numero de onda (trazas transformadas) */
FOR ik:=0 TO N-1 DO
  IF ik > (N / 2) THEN k:= N - ik
                  ELSE k:= ik
/* Ciclo sobre las frecuencias (muestras transformadas) */
  FOR w:=0 TO M-1 DO
    IF w > (M / 2) THEN w0:= M - w
                   ELSE w0:= w
    Transf:=(0.0, 0.0)
/* Ciclo sobre el tiempo (muestras sin transformar)*/
    FOR t:=0 TO M-1 DO
/* Calculo del operador DMO */
      A:=SQRT(1+aux*SQR(k*h/(t*w0))
      fase:= 2*Pi*w0*t*A
 /* Calculo de la exponencial compleja */
      Parte.real:= Cos(fase)
      Parte.ima:= Sen(fase)
 /* Conversion a complejos */
      Complejo:= Complex (Parte.real, Parte.ima)
      Transf:= Transf + Complejo*P(t,ik,h) / A
    P0(w,ik,h ):= Transf
  ....
```

Fig. 1. Fragmento en pseudo-Pascal que describe el algoritmo secuencial de la integral en el DMO para corregir H secciones sísmicas.

El presente trabajo se centra en estudiar la paralelización del algoritmo de DMO propuesto por D. Hale. A lo largo del trabajo, se podrá observar la riqueza de puntos de tratamiento que posee el algoritmo. De hecho, se explorarán diferentes niveles de paralelismo pero haciendo énfasis en la descomposición de dominios que se pone de manifiesto en el costoso cálculo de la integral ya mencionada.

En la sección 2, se describen brevemente la arquitectura y las métricas utilizadas. En la sección 3, se discuten los niveles de paralelismo detectados en el algoritmo de Hale. En las secciones 4 y 5 se estudiarán los dos últimos niveles de paralelismo encontrados. En la sección 6, se presentan y discuten los resultados experimentales sobre la base de transputadores.

Finalmente, para medir los alcances de las ideas propuestas, se ofrecen una serie de conclusiones y direcciones futuras de trabajo en la sección 7.

2. Reseña de la arquitectura y de las métricas adoptadas

2.1 Topología de malla

Para el estudio de los algoritmos y las pruebas experimentales se partirá del hecho de que se cuenta con una malla de $P1 \times P2$ transputadores. Adicionalmente, el transputador $P_{1,1}$ funge como el único punto de entrada y salida de los datos.

Tanto $P1$, $P2$ y el número de las muestras M se consideraran potencias de 2, con lo cual se asegura la divisibilidad de M entre P=$P1$x$P2$ para $M \geq P$. Durante el curso de este trabajo se tuvo acceso a una plataforma de 8 transputadores con 1 Mbyte de memoria cada uno y 1,5 Mflops de desempeño pico.

2.2 Cómputo y comunicación

Se utiliza la siguiente notación:

- $T_{cal}(M, P)$, es el tiempo de cómputo sobre P procesadores.
- $T_{com}(M, P)$, es el tiempo de comunicación de datos entre los procesadores.
- $T_o(M, P)$, como el "overhead" de comunicación y sincronización.

El tiempo de envío de un dato compuesto de m bytes entre dos procesadores vecinos está dado por :

$$t_{start} + mt_{send} \tag{3}$$

donde t_{send}, es el tiempo de envío de un byte y t_{start}, es el tiempo de latencia antes de enviar efectivamente el mensaje. El lector puede observar que en la comunicación entre nodos vecinos $T_o(M, P) = t_{start}$ y $T_{com}(M, P) = mt_{send}$.

2.3 Aceleramiento

El aceleramiento ("speedup") es una función del número de procesadores P y del tamaño del problema M [1] que está expresado por :

$$A(n, p) = \frac{T(n, 1)}{T(n, p)} \tag{4}$$

2.4 Fracción serial

A. Karp y H. Flatt [14] exponen la siguiente métrica:

$$f = \frac{\frac{1}{A(n,p)} - \frac{1}{p}}{1 - \frac{1}{p}} \tag{5}$$

3. Niveles de paralelismo asociados al problema

En la Fig. 2 se ilustra la relación entre los elementos que conforman un sismograma de sirve de entrada al procesamiento DMO.

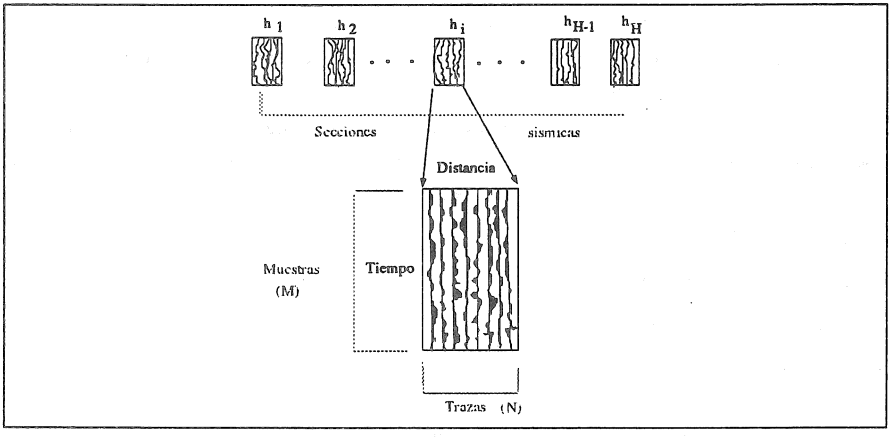

Fig. 2. La sección sísmica vista como un arreglo de datos bidimensional de tamaño MxN.

En primer término el sismograma se compone de conjuntos diferenciados de trazas que reciben el nombre de secciones sísmicas. Como en las ecuaciones (1) y (2) el parámetro h permanece constante, cada sección sísmica con un h dado puede ser procesada independientemente de las demás.

Al propio tiempo, cada sección es un arreglo bidimensional de datos que es transformado en forma independiente a lo largo de sus parámetros o ejes (tiempo y distancia), tal como se desprende de las integrales en (1) y (2). De modo que para cada traza (muestra) se pueden transformar simultáneamente todas las muestras (trazas) de una sección sísmica en particular. De esta manera, una misma sección se puede dividir en lotes de trazas o en lotes de muestras (i.e. cortes verticales u horizontales) y efectuar la correspondiente transformada de Fourier o integración de un grupo determinado de datos.

Otra descomposición del dominio del problema tiene que ver con la costosa integración sobre t en (1). Para ello, el fragmento de la integración se expresa matricialmente de la siguiente manera:

$$\begin{pmatrix} P_0(0,k,h) \\ \cdot \\ \cdot \\ P_0(M-1,k,h) \end{pmatrix} = \tag{6}$$

$$= \begin{pmatrix} W^{0,0} & \cdot & \cdot & W^{0,M-1} \\ \cdot & W^{1,1} & \cdot\cdot & \cdot \\ \cdot & \cdot & \cdot\cdot & \cdot \\ W^{M-1,0} & \cdot & \cdot & W^{M-1,M-1} \end{pmatrix} \begin{pmatrix} P(1,k,h) \\ \cdot \\ \cdot \\ P(M-1,k,h) \end{pmatrix}$$

donde $W^{wt} \equiv e^{2\pi i w_0 t \Lambda(w_0,t,k,h)} \mathrm{A}^{-1}(w_0,t,h,k)$.

La reformulación anterior ilustra como el nivel de paralelismo por integración de los datos aprovecha la descomposición del producto matriz-vector entre los procesadores.

La discusión anterior, basada sobre la Fig. 2 apunta a los diferentes niveles de paralelismo encuentrados en el problema del DMO:

- Secciones sísmicas
- Trazas sísmicas
- Integración de los datos.

El primer nivel no se pudo implantar en la plataforma disponible por falta de memoria [15]. En las secciones siguientes se analizan las ventajas y desventajas computacionales que trae consigo la exploración de los dos últimos niveles.

4. Paralelismo por trazas sísmicas

Una sección sísmica puede subdividirse en lotes de trazas. Esto permite

- Paralelización sencilla del programa .- Solamente se requiere la paralelización del lazo más externo de la integración (i.e. el lazo sobre *ik* de la Fig. 1). Por tanto, basta tomar los datos pasados por la primera transformada de Fourier y devolver los que resultan de la integración.
- Menor requerimiento de memoria para la implantación.- Bastaría utilizar dos arreglos de complejos de dimensión $\frac{NM}{P}$ para almacenar los datos antes y después de la integración. Para datos típicos (i.e. N=128, M=2048), esto implica un requerimiento de al menos 5 transputadores para correr la aplicación.
- Manejo fácil de la entrada y salida.- Basta hacer una carga y posterior escritura de la sección sísmica a corregir, tal como se hace en la versión secuencial. De esta manera se evita el manejo concurrente de grandes volúmenes de entrada y salida sobre un sistema con pocos puntos de acceso.[3]

---

[3] Una limitación del sistema utilizado es que provee un sólo canal de comunicación con el servidor de discos.

Un punto que vale la pena mencionar con referencia a los cálculos, es que puede aprovecharse la descomposición espectral del número de onda. Es decir, los datos transformados poseen una simetría alrededor de $k=N/2+1$ tal como se puede apreciar en el algoritmo de la Fig. 1. De esta manera, basta asignar $(N/2+2)/P$ trazas sísmicas a cada procesador para efectuar la integración.

El aprovechamiento de la descomposición espectral es válido en cualesquiera de los niveles de paralelismo con el fin de reducir el cómputo. Sin embargo, sólo en el paralelismo por trazas, la descomposición espectral del número de onda contribuye a una reducción del uso de memoria local por transputador. Esto se debe a que en el paralelismo por sección los datos originales no presentan simetrías y, en el paralelismo por integración de los datos, se trabaja sobre una traza a la vez. Consecuentemente, los $4/P$ Mbytes de memoria del ejemplo se reducen a 2,16 $/P$ Mbytes. La siguiente proposición resume la reciente discusión:

Proposición 1.[4] *El orden del tiempo de cómputo en el paralelismo por trazas es el mismo al de secciones sísmicas. Sin embargo, los requerimientos de memoria de un sistema paralelo distribuido con igual capacidad, es al menos* $\frac{2}{5}P$ *veces menor en el primer caso, pero nunca* $2P$ *veces mayor en relación al segundo.*

5. Paralelismo por integración

La formulación matricial (15) puede resolverse a través de tres esquemas secuenciales: Por filas, por columnas y por bloques (Golub y van Loan [7]). El problema del producto matriz-vector en paralelo sobre diversas topologías ha sido tratado en [1], [3], [4], [6] y [8].

Cada esquema posee una misma complejidad computacional de $O(n^2)$ para una matriz de dimensión n. No obstante, la descomposición del dominio que sugiere cada uno de ellos es diferente para la paralelización del producto matriz-vector. Los dos primeros conducen a una descomposición por tiras ("stripes") unidimensionales de la matriz. El tercero en el fondo, es una combinación de los dos primeros, generándose de este modo una descomposición bidimensional (Snyder-Socha [18]).

Los requerimientos de memoria sobre un sistema paralelo varían de un esquema a otro. Sin embargo, en el peor de los casos sólo se requiere almacenar el vector, es decir, la traza sísmica a transformar. Esto se traduce en al menos 8 Mbytes de memoria global en el sistema. Como contraparte, es necesario cargar y descargar del sistema todas las trazas que conforman una sección. Más específicamente, para la ejecución del DMO sobre un sismograma completo se requieren $H*(N/2 + 2)$ cargas al sistema; por lo discutido en el paralelismo por trazas.

Explotar este nivel de paralelismo posee las siguientes implicaciones:

- Tratamiento del fragmento de la integración solamente.
- Mejor aprovechamiento de la memoria local del transputador.

---

[4] En [15] se demuestran las proposiciones y corolarios del presente trabajo.

- Diferentes posibilidades en la descomposición de la matriz.
- El manejo más sencillo de la entrada y salida en relación a los otros dos niveles de paralelismo.

En los apartados que siguen se analizan por separado cada uno de los esquemas algorítmicos planteados para la descomposición matrtiz-vector. Sin pérdida de generalidad, el análisis dejará de lado el cómputo de las entradas de la matriz . El lector podrá inferir sin mucha dificultad que el orden de este cómputo es $\frac{M^2}{P}$.

Se supone en lo que sigue, que la suma de 2 números complejos es equivalente a 2 sumas reales, y el producto de 2 números complejos es equivalente a 2 sumas y 4 productos reales. En consecuencia, se establece una relación de desempeño 1:4 entre ambas operaciones complejas, respectivamente. Esta suposición no es tan caprichosa, si se sabe que en el transputador un producto es *11/7* [12] veces más costoso que una suma de punto flotante en precisión simple (32 bits)[5]. El costo de realizar una suma o resta real en el transputador se denotará por τ.

5.1 Esquema por filas

En la Fig. 3 se muestra la descomposición de la matriz en trozos horizontales, idénticos y rectangulares. La cantidad de trozos es igual a P y cada uno es asignado a un procesador específico.

El algoritmo se puede bosquejar en tres pasos:

1. Se reparte el vector de datos de la traza y los límites verticales (i.e. el rango de filas) del trozo de fila a cada procesador.
2. En paralelo, cada nodo P_l $(l = 1, .., P)$ construye la entradas de la matriz y lo multiplica por la entrada correspondiente del vector siguiendo un esquema por filas. Si i es el índice de las filas y j el de las columnas, entonces el producto se obtiene de la siguiente manera:

$$y_i = \sum_{j=1}^{M} w_{ij} x_j \qquad i = (l-1)\frac{M}{P} + 1, .., l\frac{M}{P} \tag{7}$$

3. Se recolectan los trozos resultantes y se colocan en la posición que les corresponde dentro del vector final de resultado.

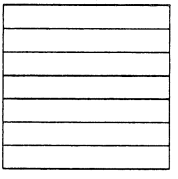

Fig. 3. Descomposición por filas

[5] Algunos autores [5], [16], han logrado maneras mas eficientes de realizar el producto de complejos sobre máquinas cuyos productos y sumas en punto flotante poseen una relación mayor de 3:1. El transputador no cae en este caso.

Proposición 2. *El tiempo T(M,P) del algoritmo paralelo por filas está dado por:*

$$\frac{5M^2}{P}\tau - \frac{M}{P}\tau + 2(P1 + P2 - 2)t_{start} + \frac{M}{2P1^2}(3P1^3 - 5P1^2 - P1 + 2PP1 + P)t_{send} \tag{8}$$

El tiempo de cómputo del algoritmo secuencial *T(M,1)* está dado por $5M^2\tau - M\tau$. Por lo que si se desprecia el tiempo de comunicación, el aceleramiento será lineal con respecto al número de procesadores.

5.2 Esquema por columnas

El algoritmo de columnas difiere levemente del de filas en los siguientes puntos:

1. Cada procesador $P_l \quad (l = 1, .., P)$ recibe el trozo del vector columna que va desde la entrada con índice $(l-1)\frac{M}{P} + 1$ hasta la entrada $l\frac{M}{P}$.
2. El producto en cada procesador se efectúa como sigue:

$$y_i^l = \sum_{i=1}^{M} w_{ij}x_j \qquad j = (l-1)\frac{M}{P} + 1, .., l\frac{M}{P} \tag{9}$$

   Esto es, y^l se escribe como combinación lineal de las $\frac{M}{P}$ columnas que le fueron asignadas, por lo que su longitud es M.
3. Se recolectan y suman entre sí los vectores y^l resultantes de cada procesador P_l

A pesar de que la comunicación es la misma en ambos algoritmos, el cómputo no lo es. Esto se debe primordialmente al paso 3. Nótese que los vectores obtenidos en cada procesador son resultados parciales del vector final de resultado. Estos son sumados a medida que se recolectan los datos. Más específicamente:

$$y = \sum_{j=1}^{M} y_j^l \qquad l = 1, .., P \tag{10}$$

Esta suma ocurre en forma encadenada y sólo en paralelo a nivel de las filas o columnas de la malla del procesador. La siguiente proposición resume esta discusión:

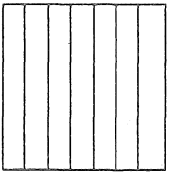

Fig. 4. Descomposición por columnas

Proposición 3. *El tiempo T(M,P) del algoritmo paralelo por columnas está dado por:*

$$\frac{5M^2}{P}\tau + (P1 + P2 - 3)M\tau + 2(P1 + P2 - 2)t_{start} + \frac{M}{2P1^2}(3P1^3 - 5P1^2 - P1 + 2PP1 + P)Mt_{send} \tag{11}$$

A diferencia del algoritmo por filas, el de columnas no realiza todas las operaciones (i.e. sumas) en paralelo. En el algoritmo anterior cada trozo resultante es asignado a la posición que le corresponde en el vector resultante. El identificador de cada procesador y el número de filas repartidas es suficiente para lograr dicha asignación.

Corolario 1. *Si la malla de procesadores es cuadrada (i.e. $P1 = P2 = \sqrt{P}$) entonces el tiempo óptimo de cómputo del algoritmo por columnas ocurre cuando:*

$$P^* = \sqrt[3]{25M^2} \approx 2,92M^{\frac{2}{3}} \tag{12}$$

Más aún, el tiempo óptimo de cómputo del algoritmo es de orden $O\left(M^{\frac{4}{3}}\right)$

Para tener una mejor idea de lo que significa el resultado del corolario, se puede tomar por ejemplo una matriz de dimensión 512. Entonces el requerimiento óptimo de procesadores es

$$P^* = \sqrt[3]{25 * (512)^2} \approx 187,14 \tag{13}$$

En el mejor de los casos, esto implica contar con una malla cuadrada de 14x14.

Si se sustituye (12) en las ecuaciones (8) y (11), se realiza el cociente y se toma el límite cuando M crece al infinito, resulta:

$$\lim_{M\to\infty} \frac{T(M,P) \text{ por filas}}{T(M,P) \text{ por columnas}} = \frac{5}{5 + 2 * 5^{\frac{1}{6}}} \approx 0,6565 \tag{14}$$

Asintóticamente, el algoritmo por filas es aproximadamente uno y medio veces *(1,52)* más rápido que el de columna. En este cálculo simplificado se está suponiendo una cantidad infinita de memoria por procesador y un ancho de banda infinita por canal.

5.3 Esquema por bloques

La idea básica del algoritmo por bloques es descomponer la matriz en $P1 \times P2$ submatrices de tamaño $\frac{M}{P1} \times \frac{M}{P2}$. De manera formal, el producto matriz-vector se replantea de la siguiente forma:

$$y_i^\mu = \sum_{\nu,j} W_{i,j}^{\mu,\nu} x_j^\nu \qquad 1 \le \mu \le P1\,, \quad 1 \le \nu \le P2 \tag{15}$$

donde μ y ν son los subíndices de los bloques.

Los vectores y^{μ} y x^{ν} son de longitud $\frac{M}{P1}$ y $\frac{M}{P2}$ que se obtienen de dividir y y x entre $P1$ y $P2$, respectivamente. Es decir,

$$x_{\frac{M}{P2}(\nu-1)+j} = x_j^{\nu}$$

$$y_{\frac{M}{P1}(\mu-1)+i} = y_i^{\mu} \tag{16}$$

Suponiendo que cada procesador de la malla está etiquetado por el par (μ, ν), el algoritmo se presenta como sigue:

1. Se reparten los elementos de x^{ν} y los límites verticales y horizontales que definen la matriz-bloque $W^{\mu\nu}$ a cada procesador $P_{\mu,\nu}$.
2. Se efectúa en paralclo, el producto matriz-vector para los bloques. Cada bloque (μ, ν) utiliza un esquema secuencial por filas[6]:

$$y^{\mu(\nu)} = \sum_{j=1}^{M/P2} W_{i,j}^{\mu,\nu} x_j^{\nu} \tag{17}$$

   sin realizar sumas sobre el índice ν.
3. Se recolectan y suman los resultados parciales. Esto se puede efectuar de dos maneras:

   a. Recolectar y sumar horizontalmente los resultados $y^{\mu(\nu)}$. Es decir:

$$y^{\mu} = \sum_{\nu=1}^{M/P2} y^{\mu(\nu)} \tag{18}$$

   Luego en sentido vertical, se colocan los trozos y^{μ} que están distribuidos en la primera columna de la malla en un sólo vector final resultado en el procesador raíz (i.e. el procesador que está conectado con el servidor o anfitrión). Este criterio se denominará suma-asignación.

   b. Recolectar los resultados parciales en un vector de longitud M. De esta manera, los resultados quedan localizados sobre la primera fila de la malla de procesadores. Luego, estos se suman horizontalmente en dirección del nodo raíz que al final poseerá el vector final resultado. Esta suma se describe analíticamente como:

$$y^{\mu} = \sum_{\nu=1}^{M} y^{\mu(\nu)} \tag{19}$$

   El criterio se denominará asignación-suma.

---

[6] También es igualmente válido tomar el algoritmo por columnas.

Proposición 4. *El tiempo T(M,P) del algoritmo paralelo por bloques usando el criterio suma-asignación está dado por:*

$$\frac{5M^2}{P}\tau + (P2-2)\frac{M}{P1}\tau + 2(P1+P2-2)t_{start} + \frac{M}{2PP1^2}(P1^5 + P1^4 - 4P1^3 - 2P1P + 2PP1^2 + 2P^2)t_{send} \tag{20}$$

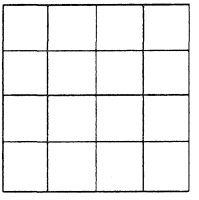

Fig. 5. Descomposición por bloques

Proposición 5. *El tiempo T(M,P) del algoritmo paralelo por bloques usando el criterio asignación-suma está dado por:*

$$\frac{5M^2}{P}\tau + (P2-1)M\tau - \frac{M}{P1}\tau + 2(P1+P2-2)t_{start} + (2P^2 - PP1 + 2P1^3 + PP1^2 - 4P1^2)\frac{M}{PP1}t_{send} \tag{21}$$

De las dos proposiciones anteriores, se ve claramente que el criterio suma-asignación reduce tanto el tiempo de cómputo como el de comunicación cuando $P1 > 1$. La explicación es la misma que para el caso columna: El criterio suma-asignación aprovecha mejor el paralelismo para la suma de vectores e implica el manejo de vectores más pequeños en la transmisión de datos.

Comparando las expresiones (8), (11) y (20) se infiere que los tiempos de cómputo ordenados de menor a mayor son:

$$T_{fila} < T_{bloque} < T_{columna} \tag{22}$$

Ahora, ¿hasta qué punto es bueno el algoritmo por bloques?. La inquietud se traduce en analizar cuál es la configuración de bloques y malla de procesadores más idónea que arroja los menores tiempos de cómputo. Para ello, se reduce la componente $(P2-2)\frac{M}{P1}$ de la expresión (20). Es fácil entrever que $P2$ debe disminuir y $P1$ aumentar para lograr ese objetivo. De

esta manera, el cálculo por bloques se reduce a una descomposición por bloques de filas de la matriz de DMO.

6. Resultados experimentales

La implantación de los programas se realizó sobre una plataforma de transputadores usando el lenguaje de programación Occam [17] y el ambiente Toolset [13].

6.1 Filas

La Tabla 1 condensa los resultados obtenidos para la descomposición homogénea por filas de la matriz de DMO.

Dimension de la matriz	8 transputadores	4 transputadores	2 transputadores	1 transputador
256	0,4768	0,9424	1,8798	3,7557
512	1,8995	3,7789	7,5535	15,0847
1024	7,5868	15,1383	30,2594	60,4737
2048	30,3281	60,5794	121,1067	242,8271
4096	121,2524	242,3346	484,5697	970,4310

Tabla 1 Tiempos de ejecución (en segundos) para el algoritmo paralelo de filas.

Si se analiza la Tabla 1 en sentido vertical, se observa que los tiempos se cuadruplican aproximadamente al duplicar la dimensión de los datos. Esto era de esperarse por la misma ecuación (8) pues si se duplica el tamaño del problema, esto es, se pasa de M a $\tilde{M} = 2M$, se obtiene que :

$$T\left(\tilde{M}, P\right) \simeq \frac{5\tilde{M}^2}{P} = \frac{5(2M)^2}{P} = \frac{5\left(4M^2\right)}{P} \simeq 4T(M, P) \tag{23}$$

Nótese que el tiempo de comunicación no incidió decisivamente en el tiempo global del algoritmo paralelo.Een la Tabla 2 se aprecian los correspondientes valores de aceleramiento.

Dimensión de la matriz	8 transputadores	4 transputadores	2 transputadores	1 transputador
256	7,8768	3,9852	1,9979	1,0
512	7,9414	3,9918	1,9970	1,0
1024	7,9709	3,9947	1,9985	1,0
2048	8,0066	4,0084	2,0050	1,0
4096	8,0033	4,0045	2,0026	1,0

Tabla 2 Aceleramientos del algoritmo paralelo por filas

No obstante, de dicha tabla se pueden extraer muchas ideas interesantes. Primero, la tabla muestra en sentido horizontal los cocientes de tiempos para un tamaño de problema fijo y usando diferente número de procesadores. Para problemas más pequeños se observa un descenso más acentuado del aceleramiento a medida que se aumentan el número de procesadores. Esto se debe a la incidencia de dos factores fundamentalmente: El tiempo de comunicación y el tiempo de carga del vector.

Obsérvese un aumento del aceleramiento en el sentido vertical de la Tabla 2. Es más, el aceleramiento alcanza valores supralineales pues excede el número de procesadores empleados.

Gustavson [9] menciona una de las causas fundamentales para que esto ocurra: el aprovechamiento de la memoria. En el transputador existen dos niveles de memoria [2]:

- Dentro del "chip", con una capacidad de 4 Kbytes (i.e. 1024 valores de punto flotante).
- Fuera del "chip", que reside en la tarjeta que soporta el transputador y posee una capacidad de 1 Mbyte (i.e. 524.288 valores de punto flotante).

En el algoritmo por fila (y similarmente en los demás algoritmos), se usan dos vectores[7] de longitud M y dos de M/P para guardar el resultado. Como el programa reside en la parte más alta de la memoria (i.e. luego de todas la variables declaradas), todos los vectores permanecen dentro del "chip" si:

$$M < 512\frac{P}{P+1} \tag{24}$$

Una particularidad del compilador de Occam es que coloca las últimas variables declaradas en las localidades más bajas de la memoria [2]. Esto quiere decir, que las variables declaradas de primero corren el riesgo de quedar fuera de la memoria del "chip" ya que ésta ha sido ocupada las variables declaradas posteriormente. En la codificación de los programas, se declararon de último los vectores resultados.

A efectos prácticos, se puede suponer el caso borde que para $M = 512\frac{P}{P+1}$ no se sale ningún valor de la memoria dentro del "chip". Haciendo esta salvedad y usando como base la desigualdad (24), se puede construir la Tabla 3.

La Tabla 3 muestra que si la matriz posee una dimensión menor de 512, los tiempos de acceso a memoria son los mismos, pero al aumentar de esa cantidad comienzan a aparecer irregularidades en el tiempo de ejecución sobre el sistema de memoria distribuido. Esto se evidencia, si por ejemplo se estima el tiempo de ejecución para la dimensión del problema igual a 1024. Como ilustración se puede considerar que la operación que gobierna el cómputo es el acceso a memoria. Así, el tiempo de ejecución secuencial se desglosa como 1024 accesos a la memoria del "chip" y 3072 fuera de él. Por tanto, si τ_m es el costo de un acceso a memoria:

$$T(M,1) \approx 10 * 1024^2\tau_m + 6 * 1024\tau_m \tag{25}$$

[7] Uno para representar la parte real y el otro la parte imaginaria del complejo.

Procesa-dores	Dimensión 256	Dimension 512	Dimensión 1024	Dimensión 2048	Dimensión 4096
1	0	1024	3072	7168	15360
2	0	512	2048	5120	11240
4	0	256	1536	4096	9216
8	0	128	1280	3584	8192

Tabla 3. Cantidad de valores que quedan fuera de la memoria del "chip".

El término elevado al cuadrado se refiere al acceso del vector columna dentro del ciclo más interno de la integración, y el lineal al acceso del vector resultado en el lazo más externo. Esto se debe a que la suma de los términos que resultan del producto interno fila de la matriz con el vector columna se acumulan en una variable para luego ser asignadas en el lazo más inmediatamente exterior en la entrada correspondiente del vector resultado. De allí que la intensidad de los accesos a memoria estén regidas por el vector columna.

Si se usan dos transputadores la cifra se convierte en:

$$T(M,2) \approx 10 * \frac{1024^2}{2}\tau_m + 2 * 512\tau_m \tag{26}$$

Entonces, el correspondiente aceleramiento para dos procesadores viene dado por:

$$A(M,2) = \frac{T(M,1)}{T(M,2)} = \frac{20 * 1024 + 12}{10 * 1024 + 2} \approx 2,0008 \tag{27}$$

De esta manera se ilustra la procedencia de este acceleramiento supralineal. En general, la expresión de aceleramiento para P procesadores y un problema de tamaño $M \geq 1024$ es:

$$A(M,P) = \frac{10M^2 + 10M - 4096}{10M^2 + 10M - 4096P}P \tag{28}$$

Claramente, al aumentar el número de procesadores el coeficiente que acompaña a P aumenta también.

Aparentemente, la contribución de las diferencias de tiempo para el acceso a memoria se suman a un incremento del tiempo de comunicación, aunque sigue predominando el cómputo asociado a la integración (i.e. divisiones, productos, seno, coseno, etc.). De hecho si se incrementa el tamaño del problema, la expresión (28) disminuye para un número fijo de procesadores P, confirmándose la atenuación de los efectos de memoria.

El concepto de fracción serial aporta aún más información sobre el aprovechamiento de los recursos computacionales por parte del algoritmo. La Tabla 4, indica que la máxima fracción serial está asociada con el problema de dimensión 32: 0,0227, esto es, 2,27 % del algoritmo paralelo. Ahora bien, se presentan irregularidades que valen destacar de la tabla. La columna asociada al uso de 4 transputadores presenta valores más pequeños que en las columnas restantes. Esto levanta la sospecha que al contar con una malla cuadrada, el "overhead" de comunicación resulta menor que en las mallas rectangulares de 2x1 y 2x4 procesadores. Si bien por la falta de procesadores esta situación no es confirmable, es claro que el tiempo de t_{start} en la ecuación (8) alcanza su menor valor cuando la malla es cuadrada. Lo mismo no se desprende de manera tan evidente para t_{send} [15].

Dimensión de la matriz	8 transputadores	4 transputadores	2 transputadores	1 transputador
256	0,0022	0,0012	0,0011	1,0
512	0,0010	0,0006	0,0015	1,0
1024	0,0005	0,0004	0,0007	1,0
2048	0.0000	0.0000	0.0000	1,0
4096	0.0000	0.0000	0.0000	1,0

Tabla 4 Fracción serial ∫ del algoritmo por filas.

Los valores de ∫ reflejan sin embargo, que a medida que crece el problema en el procesador, disminuye la fracción serial. Esto se debe a que crece la fracción paralela asociada al cómputo.

6.2 Columnas

En el algoritmo por columnas se obtuvieron los siguientes tiempos:

Dimensión de la matriz	8 transputadores	4 transputadores	2 transputadores	1 transputador
256	0,4926	0,9667	1,9250	3,8077
512	1,9568	3,8814	7,7477	15,4189
1024	7,8085	15,5360	31,1084	62,2270
2048	31,1464	62,2894	124,6848	248,4312
4096	124,9169	249,9302	500,0774	997,1921

Tabla 5 Tiempos de ejecución (en segundos) para el algoritmo paralelo por columnas.

Los valores de tiempo son mayores a los del algoritmo por filas, tal como era de esperarse según el desarrollo analítico expuesto en el capítulo anterior. Igualmente se encontró la presencia de valores supralineales que pueden ser explicados de manera análoga al caso de filas [15].

Además, se obtuvo el mismo comportamiento en cuanto a la fracción serial, un decremento relativo para el uso de 4 transputadores con respecto al uso de 2 y 8 transputadores.

6.3 Bloques

En el algoritmo por bloques, cada procesador se encargó de ejecutar en forma local el algoritmo por filas. El esquema utilizado fue de suma-asignación. De esta manera, para el uso 1 y 2 transputadores se llega al caso particular del algoritmo paralelo por filas. Por lo tanto, las siguientes tablas hacen sólo mención a las mallas de 2x2 y 2x4 procesadores.

Dimension de la matriz	8 transputadores	4 transputadores
256	0,4815	0,9501
512	1,9109	3,7956
1024	7,6152	15,2104
2048	30,4692	60,9308
4096	121,2524	243,8778

Tabla 6 Tiempos de ejecución (en segundos) para el algoritmo paralelo por bloques.

Se obtuvieron tiempos menores a los del algoritmo por columnas pero mayores a los logrados con el algoritmo por filas tal como lo sugiere (22). Las diferencias se hacen más notables a medida que aumenta la dimensión del problema. Al observar los aceleramientos se encuentró el mismo patrón de comportamiento. Las fracciones seriales dejan entrever la misma situación anterior.

6.4 Comparación de los diferentes esquemas paralelos

La Tabla 7 resume los tiempos de ejecución usando el máximo potencial del sistema para los algoritmos propuestos. Es evidente que los tiempos del algoritmo por bloques se aproximan más a los del algoritmo por filas que al de columnas. Esto también se puede concluir mediante los tiempos de cómputo de la ecuaciones (8), (11) y (20).

Si se efectúan los cocientes columna-bloque y bloque-fila se obtienen respectivamente:

$$\frac{1+\frac{24}{5M}}{1+\frac{8}{5M}} \quad \text{y} \quad \frac{1+\frac{8}{5M}}{1-\frac{1}{5M}} \tag{29}$$

Comparando ambos cocientes, se obtiene que $M > 2{,}5$ lo que se cumple trivialmente.

Dimension de la matriz	Fila	Bloque	Columna
256	0,4768	0,4815	0,4926
512	1,8995	1,9109	1,9568
1024	7,5868	7,6152	7,8085
2048	30,3281	30,4692	31,1464
4096	121,2524	121,9490	124,9169

Tabla 7 Comparación entre los tiempos (en segundos) de las diferentes descomposiciones homogéneas sobre 8 transputadores..

Adicionalmente, se puede determinar cuán cerca están los tiempos del algoritmo por bloques con respecto a los otros dos. Para ello, se escribe éste como una combinación lineal de los tiempos de cómputo del algoritmo por filas y por columnas. En otras palabras:

$$\rho\left(\frac{5M^2}{P} - \frac{M}{P}\right)\tau + (1-\rho)\left(\frac{5M^2}{P} - (P1+P2-3)M\right)\tau = \frac{5M^2}{P} + M \tag{30}$$

donde $0 < \rho < 1$. Por lo que:

$$\rho = \frac{P(P1+P2-4)}{P(P1+P2-3)+1} \tag{31}$$

Para $P1$=2 y $P2$=4, esto indica que el tiempo del algoritmo por bloques está a 25/16 del tiempo del algoritmo por filas y en consecuencia, a 25/9 del algoritmo por columnas.

Si se hacen los cálculos en base a los valores de la tabla, como por ejemplo los tiempos asociados al problema 512 se tiene que:

$$1,8995\rho + (1-\rho)1,9568 = 1,9109 \quad \Rightarrow \quad \rho = \frac{0,0459}{0,0573} \approx 0,8 \tag{32}$$

Este valor se aleja en un 16% del valor estimado por (31), probablemente como resultado de omitir la influencia del tiempo de comunicación y la memoria.

6.5 Desempeño de otras arquitecturas

Con el fin de medir el alcance de los tiempos obtenidos, se comparó el desempeño de los mejores algoritmos con otros computadores. Para efectos de esta comparación se usó el computador IBM 3090 con y sin facilidad vectorial, la máquina RS/6000 modelo 530 de IBM

que es conceptuada como la estación más rápida que hay en el mercado actualmente y dos modelos diferentes de las estaciones de trabajo SUN. Los resultados se muestran en la Tabla 8 De ella se puede deducir la siguiente relación jerárquica a partir de los tiempos:

Dimen-sión	IBM 3090/VF	8T por filas	IBM 3090	RS6000-530	SUN 4/ 260	SUN SPARC-1
512	0,59	1,89	1,90	3,58	6,20	12,50
1024	2,21	7,58	7,50	14,28	24,80	51,00
2048	8,66	30,33	29,90	56,94	98,80	202,50
4096	34,46	121,25	119,26	228,23	395,60	822,30

Tabla 8 Tiempos (en segundos) para la integración de una traza particular sobre diferentes arquitecturas.

De acuerdo con la derivación analítica del algoritmo por filas, y a lo que se puede observar en la tabla, se podría alcanzar el desempeño en la IBM3090/VF usando aproximadamente el triple del número de transputadores y triplicando el tamaño del problema para mantener la eficiencia [15].

6.6 Comparación entre el paralelismo por trazas y paralelismo por integración

Se puede observar en la Tabla 9 que los resultados son muy similares, aunque son levemente menores en el caso del paralelismo por integración de los datos. Esta diferencia puede atribuirse a las diferencia de tiempo de acceso a memoria, ya que el paralelismo por trazas involucra más accesos de memoria fuera del "chip".

Dimensión de la sección	Integración de los datos usando el esquema por filas	Paralelización por trazas
64x128	7,7568	7,8985
64x256	30,5270	31,4800
64x512	121,6204	125,8772
62x1024	486,6712	503,8840

Tabla 9 Tiempos (en segundos) del paralelismo por trazas e integración de los datos sobre 8 transputadores.

7. Conclusiones y perspectivas

Se concluyen los siguientes aspectos en torno al trabajo:

1. El DMO es una aplicación en la cual es factible explotar los tres niveles de paralelismo estudiados.
2. La demora por tiempo de comunicación es realmente ínfima en relación a los tiempo de cómputo.
3. Se mostró una gran semblanza entre los resultados teóricos y empíricos.
4. De los resultados obtenidos, se puede construir una jerarquía en base a los tiempos de respuesta de cada algoritmo implantado para el nivel de integración de los datos:
   a. Por filas
   b. Por bloques
   c. Por columnas

   Si se tuviese que ubicar el paralelismo por trazas, con el fin de obtener una jerarquía más general, éste posiblemente ocuparía la última posición, según se desprende de la Tabla 9.
5. A través de los resultados empíricos, se pudo observar la enorme incidencia de los accesos a memoria dentro o fuera del "chip" del transputador.
6. La fracción serial ciertamente aporta una información práctica que no es detectada con las demás métricas. En los experimentos realizados, mostró indicios de las posibles anomalías que se pueden presentar al trabajar sobre una malla cuadrada o rectangular de procesadores.
7. Existen ciertas limitaciones de memoria y capacidad de comunicación en la plataforma la arquitectura utilizada para explotar cada uno de los niveles de paralelismo encontrados en el DMO.

Se recomiendan las siguientes acciones:

1. Estudiar con mayor detenimiento el problema de entradas repetidas en la matriz que se derivan del operador para reducir la cantidad de cálculo.(El estudio de este problema se está tratando actualmente y su publicación está en curso).
2. Explorar el paralelismo por secciones y por trazas sísmicas en mayor detalle.
3. Conjugar los tres niveles de paralelismo con el fin de explotar un mayor grado de paralelismo.
4. Llevar las experiencias computacionales a un mayor número de procesadores y tamaño del problema.

Referencias

[1] Akl, S. *The Design and Analysis of Parallel Algorithms.* Englewood Cliffs, N. J., Prentice Hall, 1989.

[2] Atkin, P. *Performance Maximisation.* INMOS Technical Note. Bristol, N. 17: 1-26.

[3] Codenotti, B. y Puglisi, C. Matrix-Vector Multiplication: Parallel Algorithms and Arquitectures. *Computer Mathematical Applications.* 12(16): 1057-1063, 1988.

[4] Codenotti, B., Lotti, G. y Romani, F. Area-Time Trade-Offs for Matrix-Vector Multiplication. *Journal of Parallel and Distributed Computing.* 1(8): 52-59, 1990.

[5] Fam, A. Efficient Complex Matrix Multiplication. *IEEE Transactions on Computers.* 7(C-37): 877-879, 1988.

[6] Fox, G. et al. *Solving Problems on Concurrent Computers.* Englewood Clifffs, N.J. Prentice Hall, 1988.

[7] Golub, G. y van Loan, Ch. *Matrix Computations* . Johns Hopkins University Press, 2nd edition, 1989.

[8] Grigg, K., Miguet, S. y Robert, Y. Complexity of the Symmetric Matriz-Vector Product on a Ring of Processors. *1990, Fifth Distributed Memory Computing Conference.* South Carolina, Vol II: 1324-1333.

[9] Gustavson, J. Fixed Time, Tiered Memory and Superlinear Speedup *1990, Fifth Distributed Memory Computing Conference.* South Carolina, Vol II: 1255–1260.

[10] Hale, D., Dip-moveout by Fourier Transform. *Geophysics*, 6(49): 741–757, 1984

[11] Hale, D., *Dip-moveout Processing : Course Notes.* . Colorado School of Mines, Tulsa, 1988.

[12] INMOS. *The Transputer Databook.* 2nd Edition, 1989.

[13] INMOS. *Occam 2 Toolset.* User manual, 1989.

[14] Karp A., y Flatt, P. Measuring Parallel Processor Performance. *Communications of the ACM.* 5(33): 539-543, 1990.

[15] Klíe, H. *Descomposición de Dominios: Caso de Estudio.* Tesis de Maestría, Universidad Simón Bolívar, Febrero 1991.

[16] Moharir, P. Extending the Scope of Golub's Method Beyond Complex Multiplication. *IEEE Transaction on Computers.* 5(C-34): 484-487, 1985.

[17] Pountain, D. y May D. *A Tutorial Introduction to Occam Programming.* INMOS, 1988.

[18] Snyder, L. y Socha D. An Algorithm Producing Balanced Partitionings of Data Arrays. *1990, Fifth Distributed Memory Computing Conference.* Vol II: 867-874.