Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXVIII, págs. 2771-2786 (artículo completo)
Cristian García Bauza, Pablo Lotito, Lisandro Parente, Marcelo Vénere (Eds.)
Tandil, Argentina, 3-6 Noviembre 2009

OPTIMIZACIÓN DE UN MODELO HIDRODIMÁMICO BIDIMENSIONAL

Alberto Andreotti[a], **Gerardo D. Hillman**[b], **Cecilia E. Pozzi P.**[b], **Andrés Rodriguez**[b], **e Indigo SA**[c]

[a]*Facultad de Ciencias Exactas Físicas y Naturales, Universidad Nacional de Córdoba, Grupo de investigación Programación en Paralelo, Av. Filloy s/n. Ciudad Universitaria, 5000 Córdoba, Argentina, albertoandreotti@gmail.com*

[b] *Laboratorio de Hidráulica, Facultad de Ciencias Exactas Físicas y Naturales, Universidad Nacional de Córdoba, Av. Filloy s/n, Ciudad Universitaria, 5000 Córdoba, Argentina, ghillman@efn.uncor.edu, cpozzi@efn.uncor.edu, arodrig@efn.uncor.edu*

[c]*Indigo S.A.,5000 Córdoba, Argentina, indigo@indigoing.com.ar*

Palabras clave: Programación en paralelo, Hidráulica.

Resumen. El método de los elementos finitos es ampliamente utilizado en la resolución de problemas de ingeniería. Es sabido también que dicho método genera una carga computacional importante, incluso para los sistemas de cómputo actuales. En el presente trabajo estudiamos la carga computacional de un modelo hidrodinámico bidimensional. Este modelo utiliza métodos finitos y permite resolver flujos a superficie libre. Mostramos también, como aplicar técnicas de optimización para mejorar el rendimiento del mismo. Las técnicas utilizadas se componen de optimizaciones automáticas del compilador, programación paralela y librerías paralelas. Además, mostramos los resultados de estas optimizaciones para un caso de estudio en particular, utilizando una arquitectura multicore. Con estas mejoras, el programa corre más de 18 veces más rápido. Finalmente, mencionamos otras mejoras posibles en el modelo, el aumento esperado en la performance y las técnicas a ser utilizadas para implementarlas.
.

1 INTRODUCCCIÓN

Este apartado contempla la descripción de las tareas asociadas al análisis hidrodinámico del tramo en estudio a partir de la aplicación de herramientas computacionales de cálculo. El resultado esperado es la caracterización hidráulica integral de la obra proyectada a partir de la evaluación objetiva de distintas variables asociadas como velocidades máximas, profundidades, sobreelevaciones o niveles resultantes de la superficie libre.

La necesidad de caracterizar el escurrimiento bidimensional y la influencia de la implantación de una obra que permita vincular las localidades de Goya y Reconquista a través del río Paraná, ha requerido la utilización de un modelo matemático denominado RMA2 FEM del Cuerpo de Ingenieros del Ejército de los Estados Unidos (USACE, 2007).

El mismo se aplica en un entorno de unos 80km de longitud por unos 40km de ancho, centrando el dominio de cálculo en la traza propuesta para el puente. Los requerimientos necesarios para la correcta aplicación del modelo han implicado el relevamiento topo-batimétrico del área afectada al cálculo, con el detalle necesario para caracterizar en forma precisa las distintas estructuras fluviales presentes tanto en el cauce principal del río como también en la gran llanura de inundación.

Los resultados obtenidos del modelo permiten caracterizar el comportamiento hidrodinámico general del río bajo la influencia de la obra y cuantificar variables hidráulicas como niveles, sobreelevaciones, velocidades y profundidades en todo el dominio de cálculo.

Debido a las dimensiones del dominio de cálculo y al nivel de resolución requerido en los sectores próximos a la obra (inferior a 10 m de longitud de elemento), se utiliza la máxima capacidad de cálculo definida al modelo, con un promedio de 60000 nodos y 20000 elementos, lo cual impacta directamente en el tiempo de cálculo requerido, que para un caso en una estructura computacional estándar, se traduce en unas 10 horas de tiempo de cálculo.

2 DESCRIPCIÓN DEL MODELO UTILIZADO

El modelo matemático RMA2 (USACE, 2007), es un modelo numérico hidrodinámico en elementos finitos bidimensional (2DH – dos Dimensiones Horizontales -), promediado en la vertical, para simular flujos naturales a superficie libre, donde el movimiento es esencialmente horizontal y puede ser descrito por una aproximación bidimensional debido a que la aceleración vertical es pequeña en comparación con la componente horizontal.

El modelo fue desarrollado originalmente por Norton, King y Orlob (1973), del Water Resources Engineers, distrito Walla Walla del Cuerpo de Ingenieros. Hubo desarrollos posteriores que fueron llevados adelante por King y Roig en la Universidad de California y por King y Norton del Resource Management Associates (RMA) y del Laboratorio de Hidráulica de la Waterways Experiment Station (WES). Este modelo es actualizado por sus autores en forma permanente, mejorando sus potencialidades. En la versión actual del sistema se han realizado importantes mejoras en cuanto a la evaporación y precipitación.

El programa ha sido utilizado para el cálculo de niveles de agua y distribuciones de flujos alrededor de islas, en flujos bajos los puentes, en zonas de expansión y de restringimientos, en uniones de ríos, aguas arriba y aguas abajo de canales de centrales hidroeléctricas, canales de centrales de bombeo, circulación y transporte en cuerpos de agua y zonas costeras, estuarios, embalses, etc.

2.1 Características, capacidades y limitaciones

El modelo RMA2© calcula la solución por elementos finitos de las ecuaciones de Navier–Stokes, que derivan directamente de la 2ª Ley de Newton (F = ma), bajo la forma de Reynolds

para flujos turbulentos.

Este modelo calcula los niveles de la superficie libre y las componentes horizontales medias de la velocidad para flujo subcrítico en campos bidimensionales de flujo. Las pérdidas friccionales se estiman a través de la ecuación de Manning, mientras que para caracterizar la turbulencia se utilizan coeficientes de viscosidad de remolino. El modelo permite analizar tanto flujo permanente como impermanente.

Sus características básicas se resumen a continuación:

- Identifica y permite corregir errores en la implementación de la malla de elementos finitos.
- Permite reiniciar (hotstart) una simulación desde una corrida previa del RMA2© y continuar con la misma.
- Permite tener en cuenta los efectos de rotación de la tierra (aceleración de Coriolis).
- Permite aplicar las tensiones originadas por el viento.
- Permite usar alternativamente distintos coeficientes de intercambio turbulento, valores del n de Manning, temperatura, etc.:
- Utilizar ecuaciones que asignan automáticamente en forma dinámica valores del n de Manning de acuerdo a la profundidad de cada nodo de la malla.
- Utilizar el número de Peclet para asignar automáticamente en forma dinámica los coeficientes de intercambio turbulento en cada nodo de la malla.
- Permite modelar incluso hasta con cinco diferentes tipos de estructuras de control de flujo.
- Permite calcular el caudal que atraviesa líneas de continuidad preseleccionadas.
- Permite considerar los efectos de corrientes secundarias en curvas.
- Permite simular el secado y vuelta a inundar de sectores del área modelada en función de las variaciones de nivel sin necesidad de generar una nueva grilla de elementos finitos.
- Acepta una gran variedad de condiciones de borde:
- Magnitud de los ángulos/velocidades por nodos.
- Componentes de velocidad por nodos.
- Elevación de la superficie libre del agua por nodos/líneas.
- Descarga por nodos/elementos/líneas.
- Niveles de mareas impermanentes por líneas.
- Descarga como función de la altura del pelo de agua por líneas.
- Velocidad del viento y dirección por nodos/elementos o tipos de material de cada elemento.

El modelo asume condiciones hidrostáticas, es decir que las principales aceleraciones en la dirección vertical son despreciables y por lo tanto, no es recomendable utilizarlo para resolver problemas donde los vórtices, vibraciones o aceleraciones verticales sean relevantes. Por lo tanto, el modelo es particularmente adecuado para simular flujos subcríticos y ondas de marea.

2.2 Ecuaciones básicas

El modelo general RMA2© resuelve las ecuaciones integradas en profundidad de conservación de masa y cantidad de movimiento en dos direcciones horizontales, cuya forma general se presenta a continuación:

Ecuaciones de la Cantidad de Movimiento:

$$h\frac{\partial v}{\partial t}+hu\frac{\partial v}{\partial x}+hu\frac{\partial v}{\partial y}-\frac{h}{p}\left(E_{yx}\frac{\partial^2 v}{\partial x^2}+E_{yy}\frac{\partial^2 v}{\partial y^2}\right)+gh\left(\frac{\partial a}{\partial y}+\frac{\partial h}{\partial y}\right)+\frac{gvn^2}{\left(h^{1/6}\right)^2}+\left(u^2+v^2\right)^{1/2} \\ -\xi V_a^2 sen\psi-2h\bar{w}vsen\phi=0 \tag{1}$$

$$h\frac{\partial u}{\partial t}+hu\frac{\partial u}{\partial x}+hu\frac{\partial u}{\partial y}-\frac{h}{p}\left(E_{xxx}\frac{\partial^2 u}{\partial x^2}+E_{xy}\frac{\partial^2 u}{\partial y^2}\right)+gh\left(\frac{\partial a}{\partial x}+\frac{\partial h}{\partial x}\right)+\frac{gun^2}{\left(h^{1/6}\right)^2}+\left(u^2+v^2\right)^{1/2} \\ -\xi V_a^2\cos\psi-2h\bar{w}vsen\phi=0 \tag{2}$$

Ecuación de Conservación de la Masa:

$$\frac{\partial h}{\partial t}+h\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)+u\frac{\partial h}{\partial x}+v\frac{\partial h}{\partial y}=0 \tag{3}$$

donde:

- h, profundidad;
- u, v, velocidades promediadas en coordenadas cartesianas;
- x, y, t, coordenadas cartesianas y tiempo;
- ρ, densidad del fluido;
- E, coeficiente de viscosidad de remolino;
- xx, dirección normal al plano vertical que contiene al eje x;
- yy, dirección normal al plano vertical que contiene al eje y;
- xy y yx, dirección tangencial a cada plano;
- g, aceleración de la gravedad;
- a, cota de fondo;
- n, coeficiente de rugosidad de Manning;
- ζ, coeficiente empírico de corte por viento;
- Va, velocidad del viento;
- ψ, dirección del viento;
- ω, velocidad angular de giro de la Tierra;
- ϕ, latitud local.

Las ecuaciones precedentes (1), (2) y (3) se resuelven por el método de los elementos finitos utilizando el método de Galerkin de residuos ponderados. Los elementos pueden ser líneas unidimensionales, cuadriláteros o triángulos bidimensionales, y pueden tener lados curvos (parabólicos). Las funciones de forma son cuadráticas para la velocidad y lineales para la profundidad. La integración espacial se realiza por integración Gaussiana. Las derivadas temporales se reemplazan por una aproximación no lineal en diferencias finitas. Se asume que las variables varían en un intervalo de tiempo en la forma:

$$f(t)=f(0)+at+bt^c \\ t_0\le t\le t_0+\Delta t \tag{4}$$

diferenciada respecto al tiempo en forma de diferencias finitas. Las letras a, b y c son constantes. Se ha encontrado por experimentación que el mejor valor de c es 1,5 (USACE, 2007).

La solución es completamente implícita y el conjunto de ecuaciones simultáneas se resuelve por la iteración no lineal de Newton–Raphson. El código ejecuta la solución por

medio de un algoritmo de solución de tipo frontal, que resuelve la matriz por partes, acoplando a la parte en cuestión la resuelta anteriormente.

2.3 Características de la simulación

Los datos de entrada al modelo están constituidos por:

a- Malla de elementos finitos, donde cada nodo queda definido por sus coordenadas (x, y) y cota (z).

b- Condiciones Iniciales, valores con los que se inicia la simulación para cada uno de los nodos.

c- Parámetros generales (rugosidades, coeficientes de dispersión turbulenta, coeficientes de vorticidad).

d- Condiciones de Borde, impuestas en nodos particulares de la malla.

2.4 Consideraciones a la hora de modelar

Es necesario especificar como condición inicial el nivel de agua para cada nodo de la región a modelar. Cuanto más apartada esté dicha condición de la solución final, más iteraciones será necesario efectuar hasta alcanzar la convergencia de la misma.

Las condiciones de borde, que dependen del tipo de contorno y de las condiciones del flujo, deben ser especificadas sobre el contorno de la malla para todo el período de simulación. Físicamente, en flujos a superficie libre hay dos tipos de contornos, uno cerrado o de flujo nulo y otro abierto. Un borde cerrado puede estar constituido por una línea de costa, un terraplén, un espigón, etc. La velocidad normal es igual a cero, por lo que no se produce flujo a través del mismo. Una condición de borde abierto define un área a lo largo de un contorno de la malla de elementos finitos en donde se produce entrada o salida de flujo. Los valores a especificar en un borde abierto dependen del tipo de contorno (entrada o salida) y del régimen del flujo (subcrítico: una condición aguas arriba y una aguas abajo; o supercrítico: dos condiciones aguas arriba).

Generalmente, para modelar un curso de agua y su valle de inundación con la hipótesis de flujo siempre subcrítico ($Fr < 1$), se especifica el flujo en las direcciones x e y en el contorno de entrada y la elevación de la superficie libre en el contorno de salida. El modelo no puede pasar de una condición de flujo subcrítico a suprcrítico debido a que debería de cambiar el esquema numérico de resolución para poder tener en cuenta el cambio en las condiciones de contorno (se necesitarían dos condiciones aguas abajo para un flujo supercrítico, en vez de una condición aguas arriba y una aguas abajo que son las necesarias para un flujo subcrítico). Si en un cierto instante, en algún nodo del dominio, el flujo pasa de ser subcrítico a supercrítico, el modelo interrumpe la simulación. En todos los puntos del dominio el flujo debe de mantenerse subcrítico durante toda la duración de la simulación.

Asimismo, en donde se planteen las condiciones de borde los contornos de la malla deben estar lo suficientemente alejados del área donde se pretende encontrar una solución, a fin de reducir la probabilidad de que los resultados se vean afectados por cualquier imprecisión en la estimación de dichas condiciones.

2.5 El caso de prueba

El caso de prueba utilizado fue la modelación hidrodinámica de un tramo del Río Paraná realizada para evaluar los efectos de la construcción de un puente sobre el cauce de dicho río. Para realizar las simulaciones se construyeron dos mallas de elementos finitos.

La primera de ellas implica resolver el escenario fluvial en estado natural, buscando

calibrar el mismo para crecientes observadas. En la Figura 1 se muestra la configuración de la malla de elementos finitos. Para la discretización del dominio se han utilizado 13204 elementos triangulares y 1631 cuadriláteros. La malla ha quedado conformada por 14835 elementos y 31550 nodos. Para la representación del dominio circundante a la traza (ubicada en la zona central de la figura) se han utilizado elementos de 300m de lado en la zona de los cauces de los ríos y de 500m de lado en la zona correspondiente a la planicie de inundación. En el resto del área modelada, el tamaño de los elementos en el cauce principal del Río Paraná es de 400m y en la planicie de 900m. Los riachos se han representado por lo menos con dos elementos. En los bordes exteriores de la malla los elementos tienen 2km de lado aproximadamente.

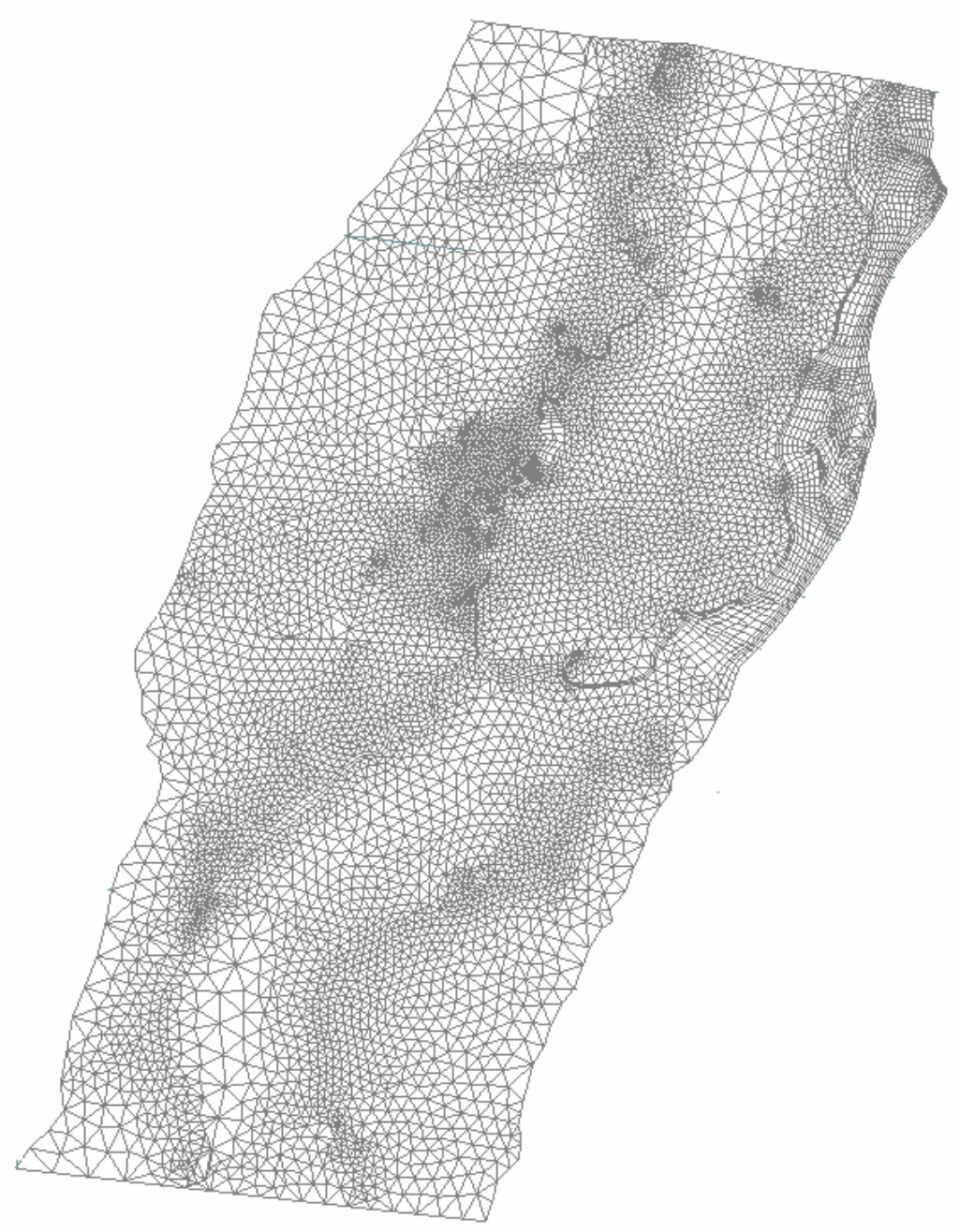

Figura 1: Malla de elementos finitos. Dominio de modelo sin puente

La segunda malla de elementos finitos se construyó incorporando la estructura diseñada, compuesta por terraplenes, puentes aliviaderos y viaducto principal, a los efectos de evaluar el comportamiento y la influencia en la hidrodinámica del tramo, brindando nuevos elementos de análisis en el diseño específico de distintos componentes de la obra.

En la Figura 2 se observa la malla de elementos finitos a la cual se le ha incorporado la estructura del puente. En este caso se han utilizado 18177 elementos triangulares y 1528 cuadriláteros. La malla ha quedado conformada por 19705 elementos y 41904 nodos. Para la representación del dominio en la zona de la traza se han utilizado elementos de entre 10m y 35m de lado a los efectos de representar los vanos de los puentes. La conformación de la malla para el resto del dominio se ha mantenido como en el caso sin puente.

Figura 2: Malla de elementos finitos. Dominio de modelo con puente

En la Figura 3 y Figura 4 se muestra un detalle de la malla de elementos de la zona de la traza del puente. En ellas se puede apreciar la distribución de elementos más pequeños que han permitido representar en detalle los vanos de los puentes.

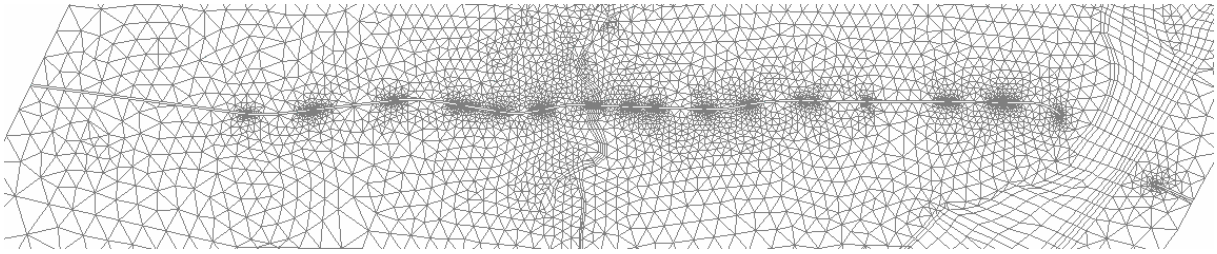

Figura 3: Detalle malla de elementos en la zona de la traza del puente

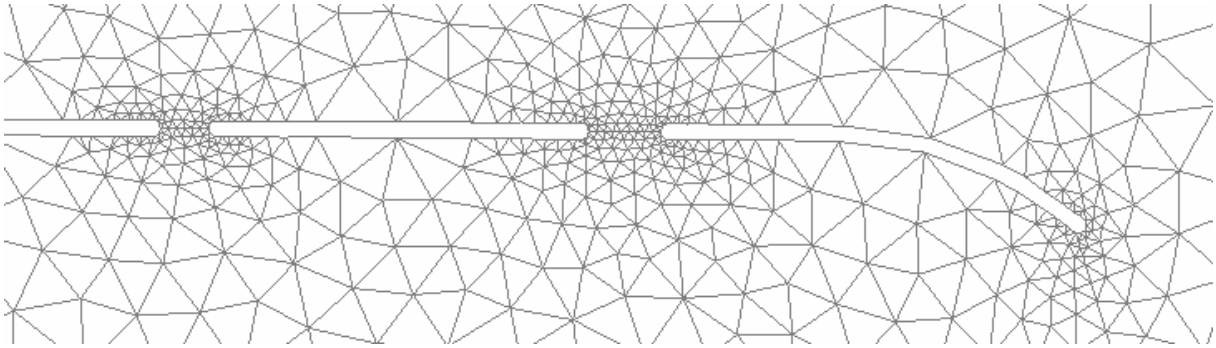

Figura 4: Detalle en el sector de vanos de puentes

Las modelaciones realizadas constan de un paso de cálculo de 6 minutos, durante 150 horas de modelación.

3 PROCESO DE OPTIMIZACIÓN

3.1 La carga del programa

Analizar la composición de la carga del programa es esencial para desarrollar una estrategia de optimización inteligente. Mediante este análisis, podemos identificar las secciones que consumen la mayor parte del tiempo de cómputo. De esta manera se puede predecir en forma cuantitativa el impacto global de una mejora realizada en una parte del programa. Para esto utilizamos la ley de Amdhal (Hennessy y Patterson, 2007), que explicamos a continuación.

$$MejoraTotal = \frac{1}{(1-P)+\frac{P}{S}} \tag{5}$$

donde (1-P) es la parte de la aplicación que no se afecta por la mejora, P es la parte de la aplicación que se afecta por la mejora y S es un número entero que expresa la cantidad de veces en que se mejora. La ley de Amdahl está graficada en la Figura 5 para distintos valores de S.

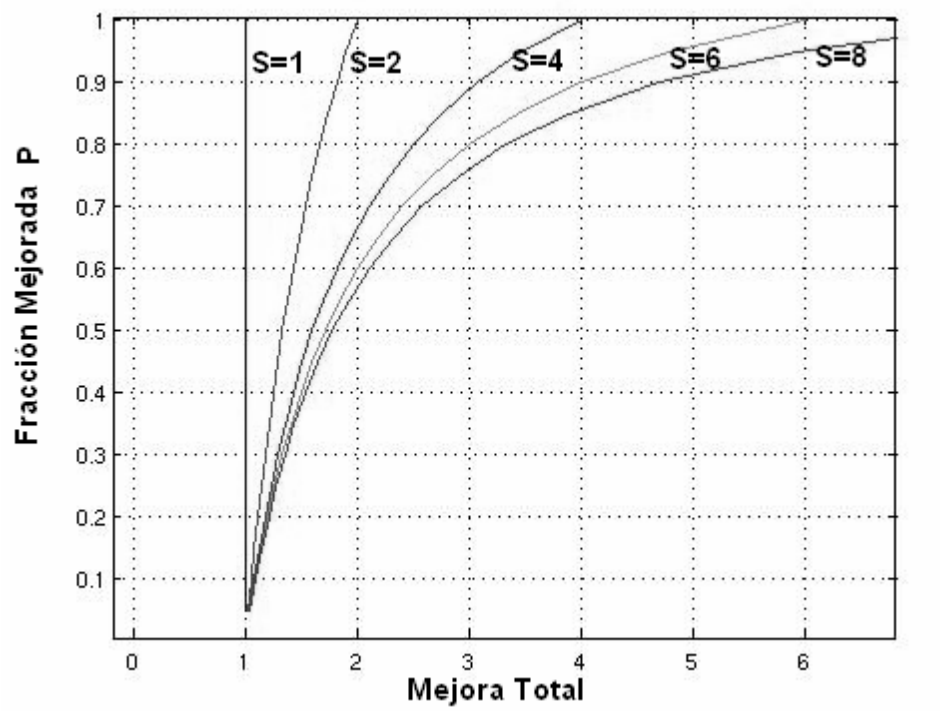

Figura 5: Representación gráfica de la ley de Amdahl para distintos valores de mejora S

Vemos que cuanto más grande sean P y S, mayor será la mejora total. De esta manera, podemos decidir entre dos alternativas de optimización comparando el impacto global de cada alternativa sobre todo el sistema.

Para optimizar RMA2, se corrió el caso de prueba y se identificaron las regiones del código con más carga computacional. La medición se realizó mediante el profiler Vtune, y es la que se muestra en la Figura 6. El resultado es que la mayor carga está en la rutina Front. Front resuelve un sistema de ecuaciones mediante el método frontal (Duffi y Reid, 1983). La ley de Amdahl nos indica, que cualquier mejora en el tiempo de Front se traducirá en una mejora casi igual en el tiempo del programa completo.

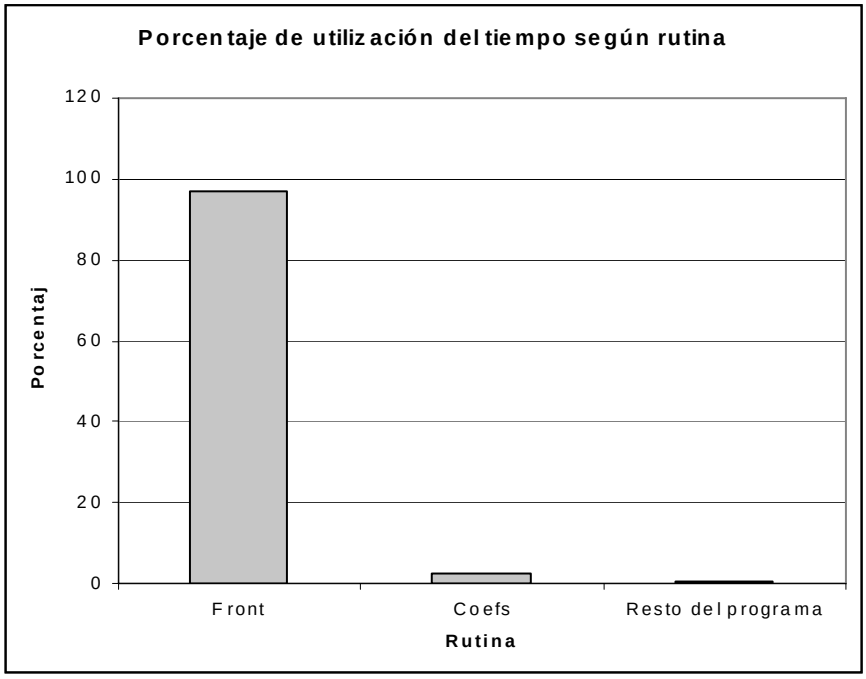

Figura 6: Tiempo consumido por las distintas rutinas del programa RMA2 cunado se ejecuta en nuestro caso de prueba

El siguiente paso es investigar las características del código involucrado en la rutina Front y plantear una estrategia de optimización. Haciendo una medición línea por línea del código involucrado obtenemos que el 86% de la carga está compuesto por líneas de la forma:

```
DO 300 L=1, LIMIT
    EQ(L,K) = EQ(L,K) – FAC * QQ(L)
300
```

Es decir, una gran cantidad de operaciones de punto flotante efectuadas sobre operandos vectoriales. Esto corresponde a la parte de la eliminación del método frontal; en la sección siguiente explicaremos el método frontal. Más adelante veremos que una forma de optimizar este tipo de cómputos es mediante vectorización.

3.2 El método Frontal

Con motivo de indagar más en las características del cómputo en Front, se hizo ingeniería inversa de la rutina. La implementación consiste en una rutina de 400 líneas. Además, se investigó la implementación del método frontal, se determinó como se representan los datos y como se realiza el cómputo. Se identificó sobrecarga (operaciones que han sido introducidas en la implementación y que podrían haber sido evitadas desde el punto de vista matemático) en las operaciones de eliminación.

El método frontal es un método utilizado para resolver sistemas de ecuaciones lineales. Es una variante del método de Gauss adaptada al caso de elementos finitos. Los problemas de elementos finitos producen por lo general, una matriz que tiene la forma:

$$A = \sum_{l} B^{(l)} \tag{6}$$

donde es la contribución de un solo elemento finito y es cero excepto en un número pequeño de filas y columnas (Duffi y Reid, 1983).

En el método frontal se saca ventaja de que las etapas de eliminación,

$$a_{lj} = a_{lj} - a_{lk} a_{kk}^{-1} a_{kj} \tag{7}$$

no tienen que esperar a las etapas de ensamblado de (3.1). A saber,

$$a_{lj} = a_{lj} + b_{ij} \tag{8}$$

para ser completadas. La operación de eliminación puede proceder tan pronto como la columna y fila del elemento pívot se hallen completamente sumadas, ya que en caso de no estar completamente sumadas, la cantidad substraída sería incorrecta. De esta manera, el algoritmo mantiene un número limitado del total de filas y columnas de la matriz global en memoria, en una estructura llamada matriz frontal. En síntesis, el algoritmo alterna entre una serie de etapas que se muestran a continuación (Figura 7).

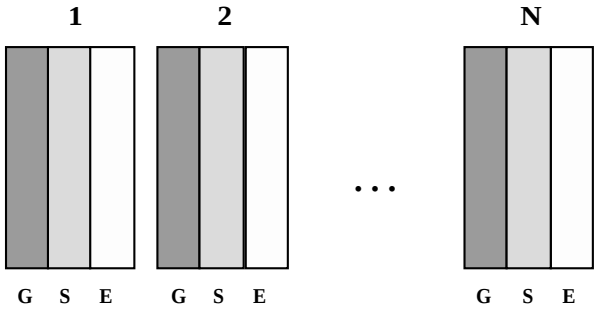

Figura 7: Etapas en las que alterna el algoritmo

En la Figura 7, G significa generación de coeficientes, S significa suma de coeficientes en la matriz frontal y E significa eliminación de variables en la matriz frontal.

La matriz frontal permite ahorrar memoria, porque una vez que la columna del pívot se ha eliminado, esta no se debe guardar más en memoria. La fila del pívot tampoco se guarda en memoria una vez que ha sido utilizada en el proceso de eliminación, pero a diferencia de la columna, la fila si será necesaria en el futuro para obtener la solución por lo que es guardada en almacenamiento secundario.

El problema con este algoritmo, es que ejecuta el doble de las operaciones estrictamente necesarias. Si recordamos el método de Gauss (en este razonamiento se consideraun n impar), la primera fila debía restarse a las (n - 1) restantes, la fila (n + 1)/2, debía restarse a las (n - 1)/2 filas restantes, y la fila n, no debe restarse a ninguna. En promedio, existen (n - 1)/2 operaciones de eliminación en el método de Gauss. El método frontal por su lado, al necesitar eliminar todas las variables de la columna, en cada paso, ejecuta (n - 1) eliminaciones. Por supuesto, el promedio es (n - 1).

Por lo tanto, el algoritmo del método frontal, ejecuta el doble de las operaciones de suma y multiplicación que son estrictamente necesarias. Las operaciones de suma y multiplicación, son operaciones de punto flotante, que consumen mucho tiempo. La carga de operaciones de punto flotante es el 87% de la carga total. Por lo tanto, eliminar esta redundancia supondría eliminar el 43,5% de la carga computacional del programa.

Este método está implementado de esta manera con el objetivo de ahorrar memoria, si tomamos un caso típico para una matriz del programa de dimensión 32000x32000, vemos que el tamaño total de la misma para elementos de punto flotante de 4 bytes es de,

$$32.000x32.000x4bytes/1048576 = 3906MB \tag{9}$$

Este número es cercano a 4 gigabytes, y es prohibitivo incluso para los tamaño de memoria actuales. Además, típicamente alrededor del 0.05% de la matriz tiene elementos distintos de

cero. Por lo tanto, utilizando una matriz frontal de 850x850 (ocupa aproximadamente 2.75MB) que contenga solo una parte la matriz global, los cálculos pueden ser llevados a cabo, incluso para los tamaños de memoria de los años 70, época en la que el programa fue creado.

3.3 Vectorización

La vectorización de las operaciones consiste en utilizar instrucciones especiales del procesador que permiten operar sobre vectores en forma eficiente. Los procesadores que soportan instrucciones vectoriales cuentan con unidades funcionales de vectorización especializadas tal como la que se muestra en la Figura 8.

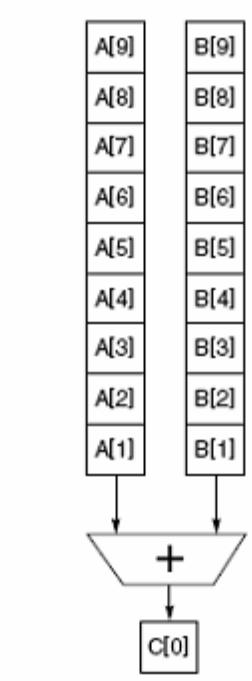

Figura 8: Implementación de una suma vectorial en un procesador con soporte de operaciones vectoriales, en el ejemplo se calcula la suma vectorial de C=A+B

Para sacar ventaja de las unidades de vectorización hacen falta dos cosas, instrucciones vectoriales y un acceso inteligente a la memoria. Las primeras son por lo general generadas por el compilador mediante directivas, mientras que la segunda depende del programador.

Para mantener el flujo de datos requerido por las unidades vectoriales, es necesario un acceso eficiente a la memoria. Los accesos a memoria deberán mostrar localidad para lograr un correcto aprovechamiento de la memoria cachè. El principio de localidad en los datos establece que los datos que se acceden en puntos en el tiempo próximos entre sí, estarán almacenados en posiciones cercanas en memoria. La menmoria cachè es una memoria intermedia entre el procesador y la memoria principal que se basa en el principio de localidad para predecir los datos que serán accedidos por el procesador en el futuro y buscarlos anticipadamente desde la memoria principal. Por lo tanto, para obtener un acceso eficiente a memoria y un buen uso de las instucciones vectoriales, debemos acceder los datos de la matriz de manera de generar accesos consecutivos en la memoria. En Fortran, las matrices son guardadas en memoria con los elementos de las columnas en forma consecutiva (este formato

se llama Column Major-Order). Es decir, los datos deben ser accedidos por columnas

Si volvemos al fragmento de código de la sección 3.1 vemos que los accesos a memoria son a direcciones consecutivas, por lo tanto el bucle es un buen candidato a ser optimizado mediante vectorización. Podemos calcular cuál será la mejora total mediante la ecuación (5) tomando un S=6 que es típico para estos problemas en las arquitecturas actuales, el valor de S alcanzado depende además del tamaño de los vectores (Hennessy y Patterson, 2007), a partir de cierto tamaño de los vectores, S tiende a estabilizarse. Reemplazando:

$$MejoraTotal = \frac{1}{(1-P)+\frac{P}{S}} = \frac{1}{(1-0{,}86)+\frac{0{,}86}{6}} = 2{,}6 \tag{10}$$

En la sección 4, cuando expliquemos los resultados, se verá que la mejora obtenida es muy cercana a nuestra predicción.

3.4 La nueva rutina Front

Ahora cabe la pregunta, de si es posible obtener una mejora más allá de la obtenida mediante vectorización. Se investigó la posibilidad de utilizar un lenguaje con directivas explícitas para la paralelización tal como OpenMP. La estrategia consistía en paralelizar la etapa de eliminación partiendo cada iteración de eliminación en varias partes a ejecutarse en paralelo. Debido a las características del algoritmo esto no llevo a ninguna mejora. Tal como se señaló en el punto anterior, el algoritmo intercala tres etapas que se repiten para cada elemento:

- Generación de coeficientes.
- Suma en la matriz frontal.
- Eliminación.

Cada una de estas etapas es pequeña y dependiente de la anterior, haciendo que el tiempo de la sincronización involucrado en cada iteración sea comparable al tiempo empleado en cada iteración, y por lo tanto prohibitivo. Se concluyó que para obtener una mejora superior, es necesario cambiar el algoritmo.

Para evitar lo señalado en el punto anterior, decidimos buscar un algoritmo que agrupe cada una de las pequeñas etapas de eliminación en una sola etapa. De esta manera, la etapa de eliminación será más grande, la comunicación será menor y el algoritmo paralelizable.

Dado que la etapa más intensa computacionalmente es la eliminación, nos centramos en elegir una manera óptima de llevarla a cabo y luego adaptar las demás etapas para que trabajen con ella. Existen en el mercado varias librerías que implementan sparse solvers que se adecuan al tipo de matrices que se generan mediante algoritmos de métodos finitos. Se eligió el solver PARDISO. PARDISO tiene un buen desempeño en sistemas smp y multicore (Gould y Hu, 2005).

Para las otras etapas, se escribieron rutinas que llevan los datos desde la representación original a la representación utilizada por PARDISO. La representación utilizada por PARDISO es la row major format (Intel® Math Kernel Library). Esta es una representación utilizada para matrices ralas, dónde se guardan únicamente los elementos no nulos de cada fila en un array.

La nueva versión de la rutina Front está compuesta como la muestra la Figura 9.

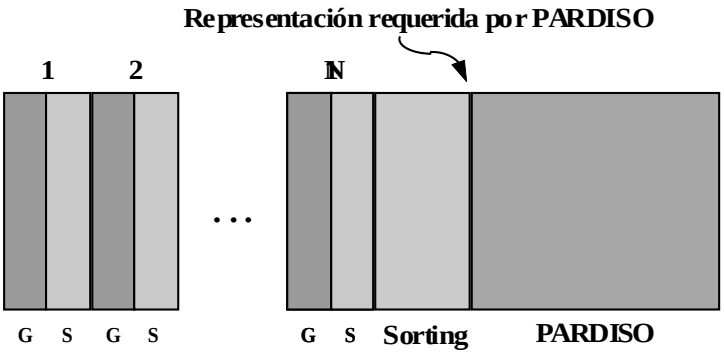

Figura 9: Composición de la nueva versión de la rutina Front

Así, la eliminación se aborda con PARDISO y el resto(Generación y Suma) mediante vectorización. Cabe señalar, que la etapa actual de suma es diferente a la original. La diferencia consiste en que la nueva etapa de suma se encarga de guardar y sumar los elementos en una representación de matriz rala. De esta manera, al final de la última etapa de suma, se encuentra en memoria una representación de la matriz completa en row major format. Esa representación es la que necesita PARDISO, excepto porque está desordenada. Finalmente, la etapa de Sorting que sigue a continuación completa el trabajo. En la sección siguiente discutimos los resultados.

4 RESULTADOS

En esta sección explicamos los resultados y algunas consideraciones prácticas. Para empezar, la versión utilizada de PARDISO es la que se incluye en la Kernel Math Library que es una versión de PARDISO optimizada por Intel. Se midió el programa con un caso de prueba y se lo comparó con la versión sin optimizar, los resultados se muestran en la Figura 10.

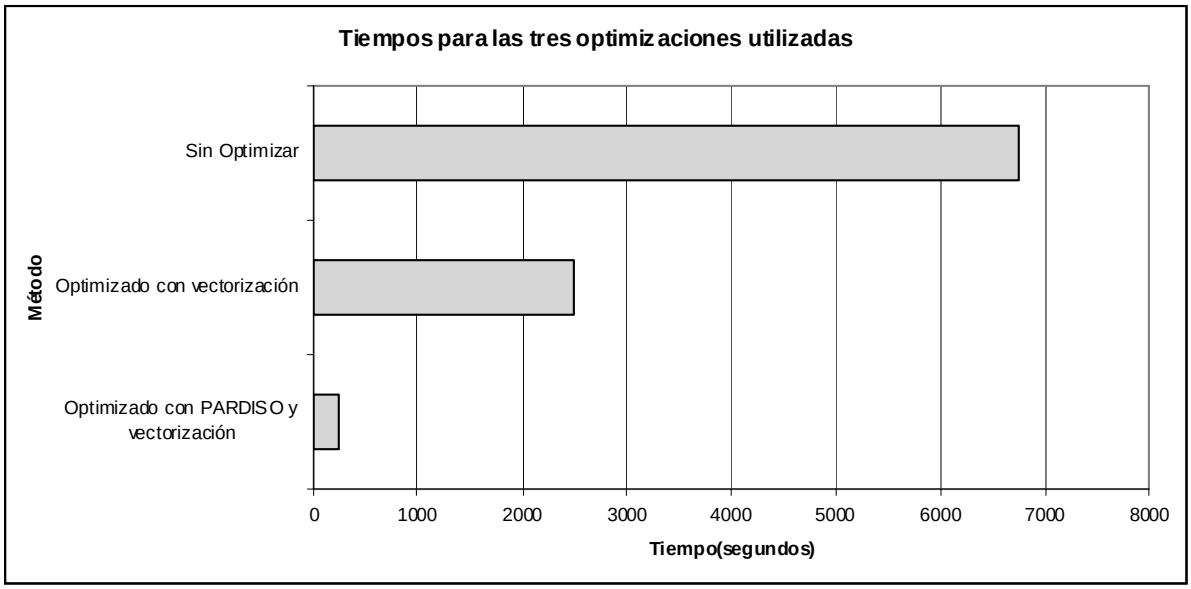

Figura 10: Resultados para las distintas optimizaciones. Las pruebas fueron realizadas sobre un Quad-Core Intel.® Xeon.® y utilizando el compilador Intel Fortran

De esta manera vemos que la optimización de vectorización da como resultado una mejora igual a:

$$\frac{SinOptimizar}{OptimizadoConVectorización} = 2{,}7 \tag{11}$$

que es semejante a la predicción de la sección 3.3, ecuación (10). A su vez, para la combinación PARDISO/vectorización tenemos

$$\frac{SinOptimizar}{OptimizadoConPARDISOyVectorización} = 27 \tag{12}$$

Por último, la mejora de la versión con PARDISO respecto a la versión optimizada con vectorización:

$$\frac{OptimizadoConVectorización}{OptimizadoConPARDISOyVectorización} = 10{,}06 \tag{13}$$

La distribución de la carga resultante en la nueva versión es la que se muestra en la Figura 11. En esta figura se puede ver que la pequeña carga que representaba coefs en la versión original es ahora mayor. Esta nueva distribución de la carga deberá ser tenida en cuenta en futuras optimizaciones. Exploramos las posibilidades de otras optimizaciones en la sección siguiente.

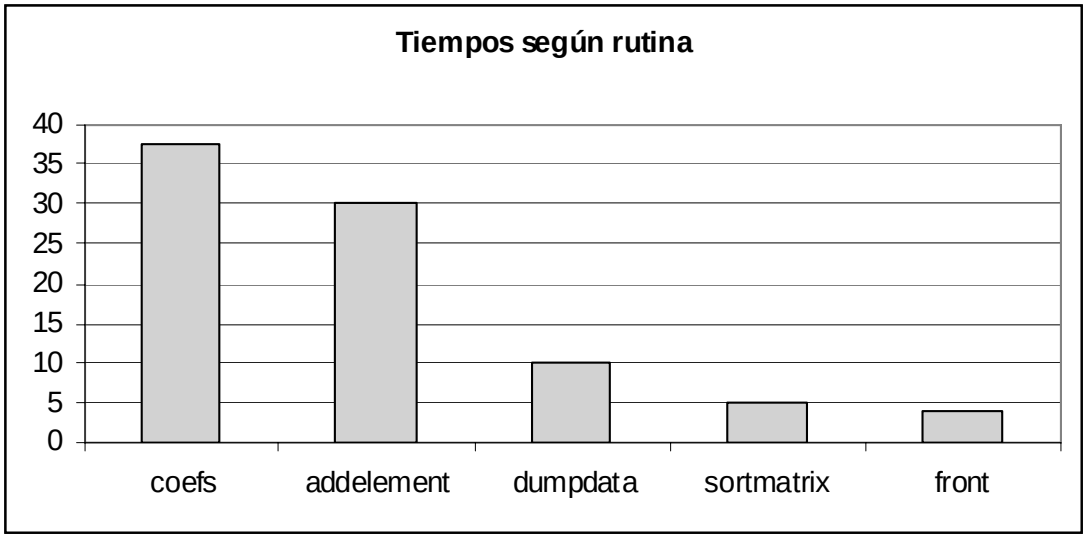

Figura 11: Tiempo en segundos para las rutinas de RMA2 optimizado

Las rutinas addelement, dumpdata y sortmatrix han sido introducidas en este trabajo para realizar la transformación en los datos.

Finalmente, para validar los resultados del programa optimizado, estos se estudiaron utilizando el visualizador SMS y se verificó que sean semejantes a los obtenidos con la versión original del programa.

5 MEJORAS FUTURAS

Se advierte que la versión resultante en este trabajo es un prototipo y por lo tanto debe ser sometida a testing. Como es sabido, el testing en un proyecto de software puede llegar a llevar hasta 40% del tiempo total del desarrollo. Las pruebas realizadas han sido escasas y sobre un solo caso. Esto puede traer problemas en el futuro tales como la dificultad de portar el programa a otras arquitecturas o incluso mantener la mejora en otras arquitecturas.

Por empezar, las rutinas que implementan el cambio en la representación de los datos que

se introduce en la nueva versión de la rutina Front puede aún optimizarse. Esto es interesante desde el punto de vista de la performance, tal como se ve en la Figura 11.

Por último, se puede paralelizar la generación de coeficientes y el ensamblado de los elementos: esto comprende investigar la dependencia de las operaciones de generación de coeficientes y plantear una estrategia de paralelización. Considerando lo expuesto sobre la ley de Amdahl y las mediciones de la Figura 11, esto es atractivo desde el punto de vista de la performance. La configuración propuesta es la mostrada en la Figura 12.

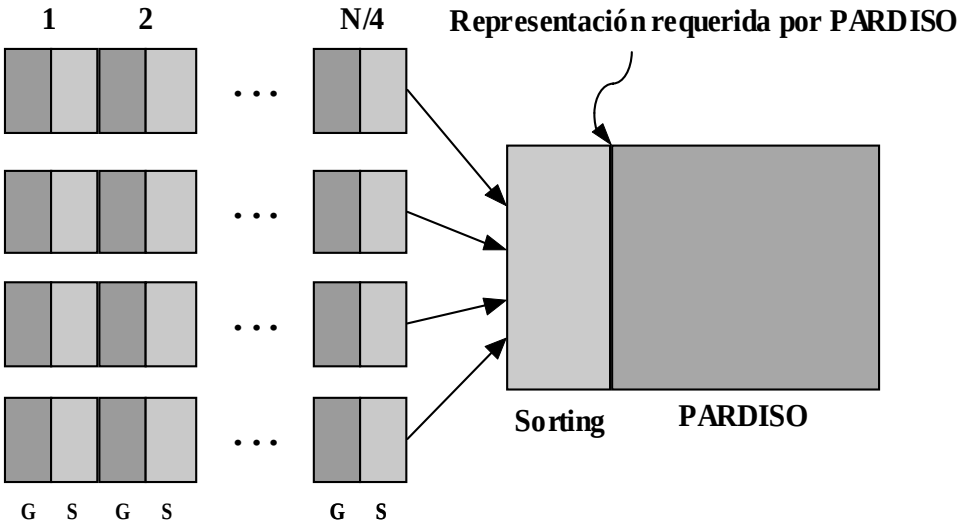

Figura 12: Nueva configuración propuesta

REFERENCIAS

USACE (1997). User's guide to RMA2-WES, Version 4.3, U.S. Army Corps of Engineers – Water Ways Experimental Station Hydraulics Laboratory, Viksburg, Miss., EEUU.

John L. Hennessy & Davida A. Patterson (2007) .Computer Architecture, a quantitative approach.

Duffi I. S. and J.K REID(1983). The Multifrontal solution of indefinite sparse Symmetric Linear Equations. ACM Transactions on Mathematical Software (TOMS).

Gould N I M, Y Hu, J A Scott (Mayo de 2005). A numerical evaluation of sparse direct solvers for the solution of large sparse, symmetric linear systems of equations.

Intel® Math Kernel Library, Reference Manual.

Rodriguez, A.; Brea, D.; Farías, D.; Bravo, H.; Castelló, E.; Hillman, G.; Pagot, G.; Weber, J. y Spalletti, P. (2002): Análisis hidro-morfodinámico del tramo medio del río Paraná para la interconexión vial Santa Fe-Corrientes. XIX Congreso Nacional del Agua, Cba.