Mecánica Computacional Vol. XXIV
A. Larreteguy (Editor)
Buenos Aires, Argentina, Noviembre 2005

PROGAM P: GENERACIÓN DE MALLAS DE ELEMENTOS FINITOS EN CLUSTERS TIPO BEOWULF

Emiliano López Chaparro[*†] **y Carlos García Garino**[†]

[*] Facultad de Ingeniería, Universidad de Mendoza,
Emilio Descotte 750, (5500) Mendoza, Argentina
e-mail: emilianolch@gmail.com

[†] LAPIC, Instituto Tecnológico Universitario, Universidad Nacional de Cuyo
Casilla de Correo 947 (5500), MENDOZA, Argentina
e-mail: cgarcia@itu.uncu.edu.ar

Key words: Generación de Mallas, Cálculo Paralelo, Clusters, Elementos Finitos.

Abstract. *Este trabajo presenta la version paralela, denominada PROGAM-P, de un código de generación de mallas estructuradas de elementos finitos, el cual esta diseñado para clusters tipo Beowulf. En el trabajo se presentan las caracterísiticas del código serial, se señalan algunas ventajas que surgen del empleo de Fortran 90 y se discute la paralelización del código. Los resultados obtenidos son excelentes desde el punto de vista del speed-up obtenido.*

1 INTRODUCTION

PROGAM[1,2] es un programa de computadora para la generación de mallas estructuradas de elementos finitos, que se basa en el código GRID[3], al cual se le realizaron varias mejoras, discutidas en la referencias citadas, y se lo utilizó a menudo en el diseño de mallas de elementos finitos para aplicaciones industriales procesadas por el segundo autor.

Este trabajo discute la extensión de PROGAM a entornos paralelos, tarea que constituye el trabajo final de Carrera del primer autor.

Para obtener la nueva herramienta denominada PROGAM-P, primero se actualizó y recuperó el código original con el fin de obtener un programa secuencial que se utilizó como punto de partida para desarrollar la herramienta paralela. Esta actualización consistió en una nueva implementación del PROGAM utilizando el lenguaje Fortran 90 y empleando el compilador Intel [4].

La paralelización del código actualizado se desarrolló para un Cluster tipo Beowulf [5], bajo la distribución Rocks[6] cuya descripción se indica en el apéndice 1 de este trabajo, empleando el citado compilador de Intel y la biblioteca MPICH[7].

En el trabajo se presenta una descripción del código: se muestra la estructura del mismo, y se presentan algunos detalles de la interpolación isoparamétrica con la cual se generan los nodos y elementos en cada región. Luego se discute la paralelización de la herramienta, para lo cual a cada procesador se le asigna una única región, se generan los nodos y elementos en forma local, y mediante un pase adecuado de mensajes, las regiones intercambian la información de las fronteras. La escritura de resultados se realiza en forma paralela, directamente a cargo de cada procesador hijo.

Se presentan ejemplos de aplicación, los cuales se muestran mediante el programa GID[8], y se discuten los resultados obtenidos con el cluster, mediante gráficos de rendimiento para distintas mallas y cantidad de procesadores.

2 DESCRIPCION DEL CODIGO SECUENCIAL

En este punto se resumen las características distintivas del programa, se indican las posibilidades que el mismo presenta, la técnica empleada para generar la información y finalmente se muestra un diagrama de flujo del código, para su comparación posterior con la versión paralela.

2.1 Características del Código

El programa es capaz de modelar dominios bidimensionales complejos que se obtienen como la unión de regiones más sencillas llamadas macroelementos, que se modelan como cuadriláteros generalizados, lo que incluye como caso particular una zonas triangulares.

Dentro de cada una de estas macroelementos se generan, mediante interpolación isoparamétrica, los nodos y elementos de la malla. Actualmente el código genera triángulos y

cuadriláteros lineales y se contempla la posibilidad de generar cuadriláteros en algunas regiones y triángulos en otras.

El programa genera la información de manera independiente para cada elemento y luego compatibiliza la información a partir de la conectividad de las regiones o macro elementos y de la información generada en la frontera de las mismas.

Finalmente se escribe en un archivo la información generada, de acuerdo a un formato de interés preestablecido, que puede adecuarse fácilmente a la entrada de datos de cualquier programa de Elementos Finitos.

2.2 Generación de la Información.

La entrada de datos de PROGAM es semejante a la de un código de elementos finitos: se ingresa información de control, como la cantidad de regiones o macroelementos y la cantidad nodos que las definen. También se ingresan las coordenadas de los nodos que conforman la geometría en la cual se quiere generar la malla de Elementos Finitos, y para cada macro elemento o región se define la conectividad de las mismas y la cantidad de nodos de la malla deseada en cada dirección.

El programa genera en forma local las coordenadas de los nodos y la conectividad de los elementos para cada una de las regiones y compatibiliza luego la información calculada para las diferentes regiones, con el fin de obtener una numeración global de los nodos y de los elementos.

Las zonas o macro elementos a partir de las cuales se modela la geometría de interés, se define mediante un cuadrilátero de 8 nudos, que puede utilizarse como rectángulo, cuadrilátero general o triángulo, como se muestra en la figura 1, que también muestra el sistema de coordenadas naturales característico en estos casos.

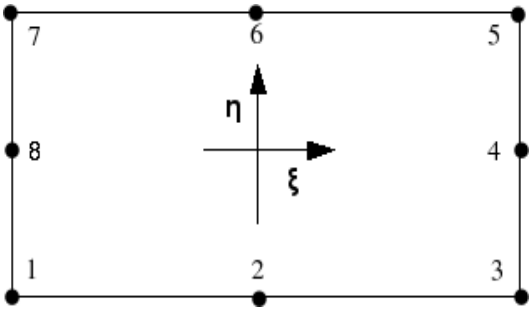

Figura 1: Región cuadrangular

Una región triangular se modela alineando dos lados del cuadrilátero general como se muestra en la figura 2.

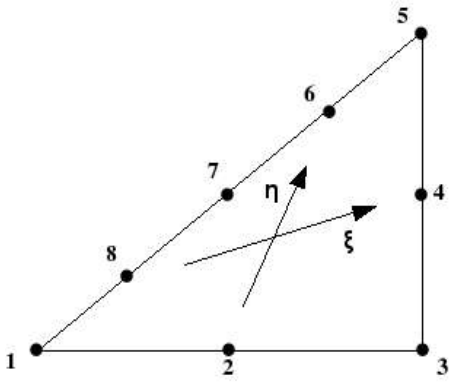

Figura 2: Región triangular

Al procesar cada región se realizan las siguientes etapas:

1. Lectura de la información de control de la región: conectividad de la misma y número de filas y columnas de nodos que formarán los elementos de la región.
2. Para generar los nodos de cada región se parte del clásico elemento patrón bidimensional de 8 nodos que definen el sistema de coordenadas naturales (ξ, η) en el centro y donde ξ y η varían entre -1 y 1. Para esta región se generan n filas por m columnas de nodos equidistantes. Finalmente se obtienen las coordenadas (x, y) de cada nodo mediante la bien conocida interpolación isoparamétrica.
3. Se realiza un control para determinar si alguno de los nodos ubicados sobre las fronteras ha sido numerado previamente por una región vecina. Si el control es positivo, los nodos sobre la frontera analizada conservan la numeración existente.
4. Los nodos se numeran secuencialmente comenzando con $\xi = -1$, $\eta = 1$ y procediendo de izquierda a derecha y de arriba hacia abajo. Todos los nodos previamente numerados se saltean.
5. La región se subdivide en elementos triangulares o cuadrangulares. En el caso de los elementos triangulares, se divide la región en cuadriláteros y luego se subdivide en triángulos utilizando la dirección de la diagonal más corta del elemento ubicado en la esquina superior izquierda.

El tamaño de los elementos puede variarse ubicando los nodos 2, 4, 6, 8 en un lugar diferente al centro del lado. El desplazamiento de estos nodos cambia el sistema de coordenadas (ξ, η) .

2.3 Estructura del Código.

La estructura del código es muy sencilla, como se muestra en la figura 3. Existe una rutina de entrada de datos, que continua con un lazo de generación de la información para cada una de los macro elementos, en los cuales se realizan todos los cálculos y operaciones discutidos en el apartado anterior. Finalmente la información se imprime a un fichero de interés.

Figura 3: Diagrama de flujo del código.

3 PARALELIZACION DEL PROGRAMA

En este apartado se discute la paralelización del código, para lo cual se presentan los criterios generales de diseño del mismo, el tratamiento de las operaciones de entrada y salida, así como la generación de la información, nodos y conectividad, en cada región.

Es importante señalar que, a priori, es candidato a ser paralelizado una gran porción del código, ya que la información de cada región se puede realizar de manera independiente. En este caso además también se logro paralelizar la escritura de resultados.

3.1 Criterios Generales

El código PROGAM-P se diseñó de manera tal que toda la información de una región esté contenida en una única estructura de datos. La Tabla 1 se muestra dicha estructura codificada en Fortran 90, junto con la descripción de cada uno de los campos.

```
TYPE región_t
    PRIVATE
    ! Número de región
    INTEGER              :: num
    ! Número de filas y columnas de nodos
    INTEGER              :: n, m
    ! Tipo de elemento (3=triangular, 4=cuadrangular)
    INTEGER              :: elem_type
    ! Datos de conectividad
    INTEGER              :: vecinas(4)
    ! Los 8 ocho nodos que definen la región
    REAL                 :: def_nodes(8,2)
    ! Coordenadas (x,y) de los nodos
    REAL, ALLOCATABLE    :: node_coords(:,:,:)
    ! Números de los nodos
    INTEGER, ALLOCATABLE :: node_nums(:,:)
    ! Cada elemento es un vector de 3 o 4 enteros
    INTEGER, ALLOCATABLE :: elements(:,:)
END TYPE region_t
```

Tabla 1: Estructura de Datos del Código

- **Número de región:** las regiones se numeran en forma secuencial a partir de 1.
- **Número de filas y colunas:** definen la cantidad de nodos en la región y determinan la cantidad de elementos: *nelem = (n-1)(m-1)* elementos cuadrangulares o *nelem = 2(n-1)(m-1)* elementos triangulares.
- **Tipo de elemento:** los únicos elementos admitidos son el triángulo de tres nodos y el cuadrado de 4 nodos.
- **Datos de conectividad:** el vector contiene los números de las regiones vecinas inferior, derecha, superior e izquierda respectivamente. Un cero en cualquiera de sus elementos indica que no hay ninguna región en la frontera correspondiente.
- **Definición de la región:** la matriz def_nodes contiene las coordenadas *(x, y)* de los ocho nodos que definen la región, comenzando por el del extremo inferior izquierdo y siguiendo en sentido antihorario.
- **Coordenadas de los nodos:** en la matriz node_coords se guardarán las coordenadas *(x, y)* de los nodos generados. El tamaño de la matriz será igual a *n.m.2* .
- **Números de los nodos:** a cada nodo generado se le asignará un número entero. Esta numeración es global, por lo tanto se requerirá comunicación entre los distintos procesos.

- **Elementos:** un elemento está representado por un vector que contiene los números de los nodos que lo definen, por lo tanto el tamaño de la matriz elements será igual a *nelem.3* o *nlem.4* .

La figura 4 ilustra el diagrama de flujo de PROGAM-P para un ejemplo genérico de tres regiones. Los bloques en gris representan funciones que requieren comunicación entre procesos. Los bloques en blanco utilizan datos de una única región y su funcionalidad es independiente de los otros procesos, por lo que su implementación no difiere de la versión secuencial.

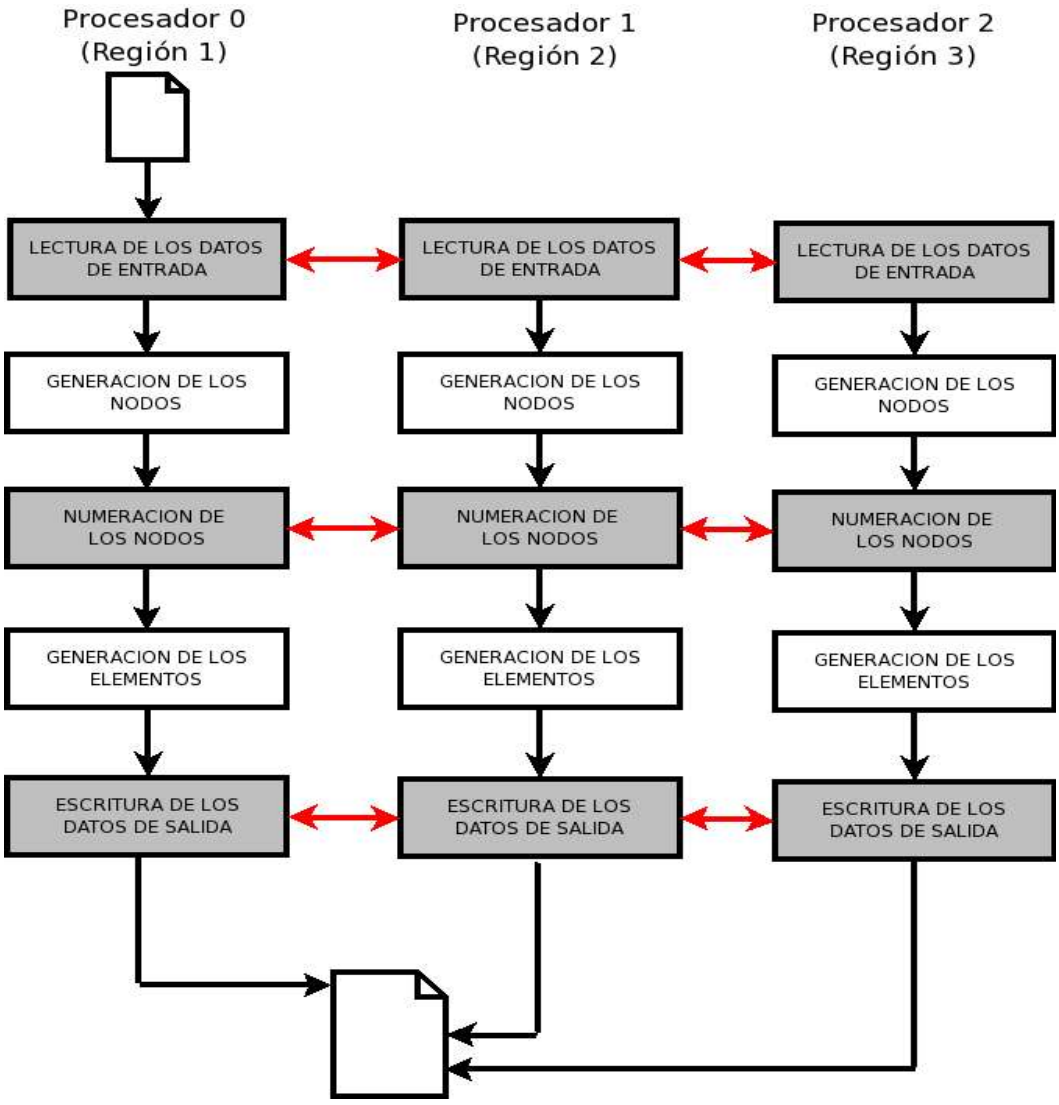

Figura 4: Diagrama de flujo del código paralelo

3.2 Operaciones de entrada y salida

Los datos de entrada se leen por el canal de entrada estándar. El procesador 0 es el único que tiene habilitado este canal, por lo tanto debe leer la totalidad de los datos y distribuirlos entre los procesadores correspondientes.

El arreglo que contiene las coordenadas de los nodos que definen las regiones es enviado a todos los procesos mediante la rutina MPI_BCAST. El proceso receptor determina cuáles nodos corresponden a la región que le toca procesar, y descarta el resto.

Los datos propios de cada región, como número de filas, número de columnas, tipo de elemento, etc, son transmitidos al procesador correspondiente mediante llamadas a MPI_SEND y MPI_RECV. Estos datos incluyen los índices de los ocho nodos que definen la región en el arreglo transmitido anteriormente mediante MPI_BCAST.

La información de salida se escribe en un único archivo de texto. Para esto deben resolverse dos problemas fundamentales:

1. Acceso al archivo: todos los procesadores deben escribir en el mismo archivo, por lo tanto éste deberá ubicarse en un sistema de archivos NFS.
2. Sincronización: para que la salida sea ordenada el procesador *n* no podrá comezar a escribir hasta que el procesador *n-1* haya terminado. Este problema se resuelve mediante la transmisión de un turno, el cual está inicialmente en posesión del procesador 0.

La Tabla 2 ilustra el procedimiento propuesto. En la misma se ha simplificado la sintaxis de las llamadas a las funciones MPI_SEND y MPI_RECV. Es importante señalar, que como es práctica usual en la programación MPI se ha asignado el número del procesador local a la variable *my_rank*.

```
IF (my_rank > 0)
        CALL MPI_RECV(my_rank-1, turno)

Abrir archivo.
Escribir información de salida.
Cerrar archivo.

IF (my_rank < max_rank)
        CALL MPI_SEND(my_rank+1, turno)
```

Tabla 2: Esquema de escritura de datos en paralelo.

Nótese que la llamada a MPI_RECV debe ser bloqueante. Esto garantiza que el procesador *n* se detendrá en la segunda línea hasta recibir el turno del procesador *n-1*.

3.3 Generación de la información

La generación de la información se realiza siguiendo las mismas etapas del código secuencial y como ya se ha señalado, es una etapa del código que posee un alto grado de paralelismo, ya que se realiza la operación de manera independiente para cada región.

Sin embargo el proceso de numeración de los nodos debe asignar un número entero único y global a cada nodo. Para ello las regiones limítrofes deben asignar los mismos números a los nodos ubicados en la frontera en común, esta tarea que es muy sencilla en un código secuencial obliga a intercambiar información entre regiones, y en este caso, entre procesadores.

El procedimiento aplicado en PROGAM-P se realiza en tres etapas:

1. Se determina la cantidad de nodos a numerar, teniendo en cuenta que los nodos ubicados sobre una frontera común a dos regiones son numerados por la que tenga menor número de región. El proceso *n* recibe del proceso *n-1*, mediante MPI_RECV, un entero que registra el último nodo global asignado. Esta variable entera se toma como número inicial, se le suma la cantidad de nodos que tiene que numerar este procesador, y envía este valor al proceso *n+1* usando la rutina MPI_SEND.
2. Se numeran los nodos de la región local, comenzando por el número 1 si se trata de la región 1, o por el número recibido del procesador *n-1* si se trata de otra región. Se procede de izquierda a derecha y de arriba hacia abajo, salteando los nodos ubicados en las fronteras que son numeradas por una región vecina.
3. Finalmente, se envían a las regiones vecinas los números de los nodos correspondientes a las fronteras que han sido numeradas por el proceso local, haciendo uso de la rutina MPI_SEND. Los numeros de los nodos que fueron salteados en la etapa anterior se reciben mediante MPI_RECV.

4 EJEMPLOS DE APLICACION

En este apartado se presentan diversas aplicaciones para mostrar las capacidades de la herramienta. En primer lugar se muestra un ejemplo sencillo para ilustrar el funcionamiento del programa y la preparación de los datos de entrada. Luego se presentan dos aplicaciones procesadas en entorno paralelo que permiten comparar el rendimiento del programa paralelo con el del programa serial.

4.1 Ejemplo simple de generación de datos

Se ha escogido un problema muy sencillo para mostrar el funcionamiento del programa. Para ello se emplea un dominio triangular, el cual se subdivide en una región cuadrilátero y dos triángulos, como se muestra en la figura 5.

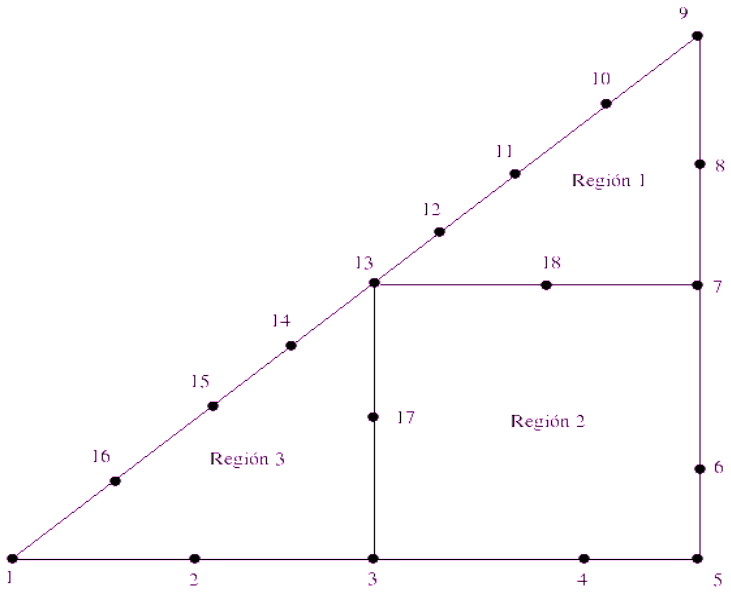

Figura 5: Ejemplo de generación de mallas con PROGAM-P. Definición de la geometría

La numeración de los nodos utilizados para definir las regiones es arbitraria. Los nodos 4 y 6 se desplazan hacia el nodo 5, para obtener los menores elementos cerca de la esquina.

Para la región 2 se elige una subdivisión de cinco filas y cinco columnas. Esta selección fija el número de filas en la región 3 en cinco y el número de columnas en la región 1 en cinco, porque el número de nodos ubicados en un borde en común debe ser idéntico.

En las regiones 1 y 3 se obtienen elementos más grandes colocando sólo tres filas de nodos en la región 1, y tres columnas en la región 3. La región 3 será subdividida en elementos cuadrangulares, mientras que para las restantes se utilizarán elementos triangulares.

En cada región, los nodos se numeran comenzando por el extremo superior izquierdo y procediendo de izquierda a derecha y de arriba hacia abajo. Los nodos ubicados en los bordes comunes son numerados por la región que tenga el menor número, es decir que la región que tenga el número mayor debe tomar los valores asignados por la región vecina.

En la figura 6 se muestra la numeración de los nodos de la región 3 del ejemplo. La misma se ha generado a partir de los nodos de la entrada de datos que se muestran en la figura 5 (siguiendo una secuencia antihoraria son los nodos 1, 2, 3, 17, 13, 11, 15, 16 y 1). Cabe destacar que la región 3 al ser un triángulo presenta de manera alineada los lados tres y cuatro del cuadrilátero generalizado (definidos por los nodos 13-14-15 y 15-16-1, respectivamente).

La numeración de la malla generada para la región 3 (3 columnas y 5 filas) se realiza de arriba hacia abajo y de izquierda a derecha, comenzando por los nodos 36, 37 y 11, continua en la segunda fila con los nodos 38, 39 y 6 y así sucesivamente hasta los nodos 44, 45 y 31 en la última fila. Los nodos 11, 16, 21, 26 y 31 que corresponden al lado derecho de la figura se generaron previamente en la región 2. De esta manera se ha ejemplificado el procedimiento enunciado en la sección 2.2 del trabajo, punto 4.

Cada elemento queda definido mediante un vector con los números de los nodos que lo

forman. Si para una determinada región se van a generar elementos triangulares, primero debe dividirse en elementos cuadrangulares, para luego dividir cada uno en dos, siguiendo la dirección de la diagonal más corta del elemento ubicado en el extremo superior izquierdo.

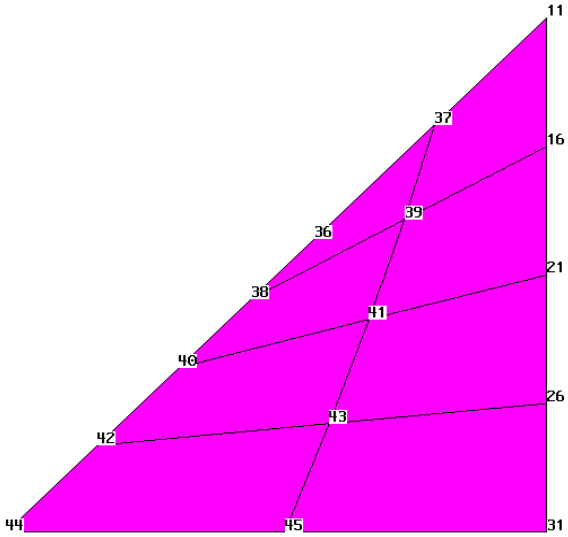

Figura 6: Numeración de la región 3.

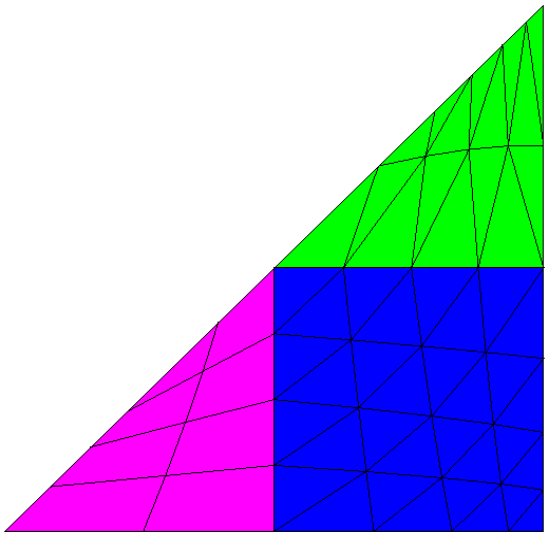

Figura 7: Malla generada

4.2 Rendimiento de la versión paralela

Para comparar el rendimiento de la versión paralela con la versión secuencial se utilizó el dominio compuesto por cuatro regiones que se muestra en la Figura 8, que se procesó con diferente número de nodos.

Inicialmente se calculó una malla con 100 nodos (5 filas x 5 columnas x 4 regiones), y se fue incrementando este valor hasta llegar a 1.000.000 de nodos (500 filas x 500 columnas x 4 regiones). El algoritmo paralelo fue ejecutado en cuatro procesadores, y la figura 9 muestra los resultados obtenidos.

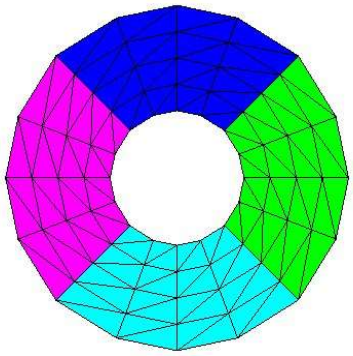

Figura 8: Dominio empleado para estimación de speed-up

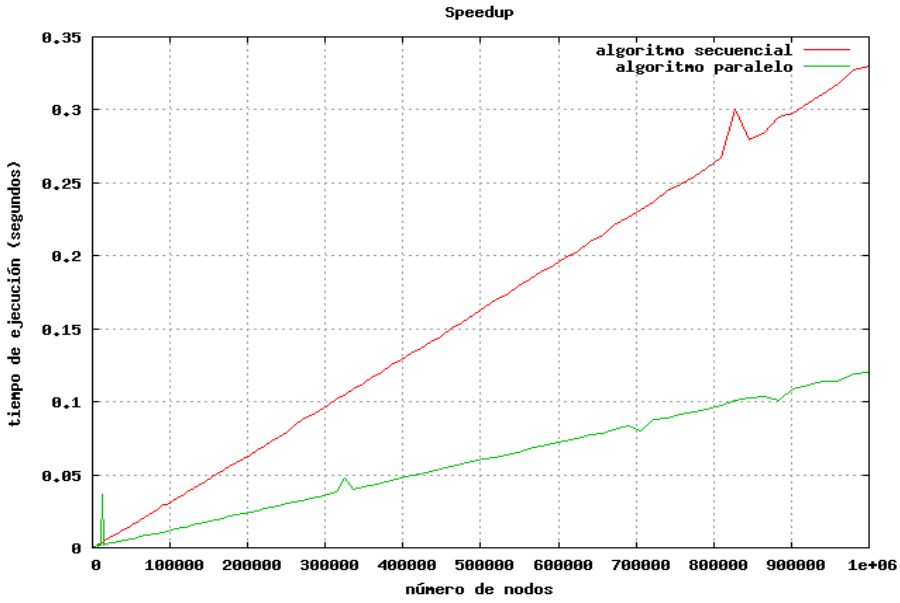

Figura 9: Comparación de los resultados obtenidos con los programas secuencial y paralelo.

Como surge de la figura 9 se obtiene un *speed-up* creciente con el tamaño de la malla, que refleja la naturaleza intrínsecamente paralela del algoritmo utilizado, que se ha traducido en las prestaciones del código. Para la malla más grande, de un millón de nodos, se alcanza una mejora cercana a 8 con la versión paralela. En conjunto con hardware actual, cual es el caso del Cluster empleado para resolver las aplicaciones de este trabajo, la herramienta discutida sugiere, a falta de pruebas en este sentido, que los límites del código pueden estar dados por la memoria RAM disponible, más que por los tiempos de cálculo.

4.3 Generación de la malla de elementos finitos de un perfil NACA

Con el fin de ilustrar una aplicación más compleja, que suele ser de interés en problemas de fluidos, se ha escogido el perfil NACA 23012 para mostrar el funcionamiento y las prestaciones de PROGAM-P.

Como es bien conocido en el caso de fluidos es importante modelar adecuadamente la capa límite, para lo cual se debe refinar convenientemente la malla alrededor del perfil. En estos casos en general se prefieren mallas no estructuradas, ya que son más versátiles para transiciones adecuadas en el tamaño de los elementos entre los cercanos al perfil y los alejados de la perturbación.

La figura 10 muestra una malla simple, más bien grosera, de 1066 elementos modelada con el código, para lo cual se han utilizado 20 procesadores de los disponibles en el cluster.

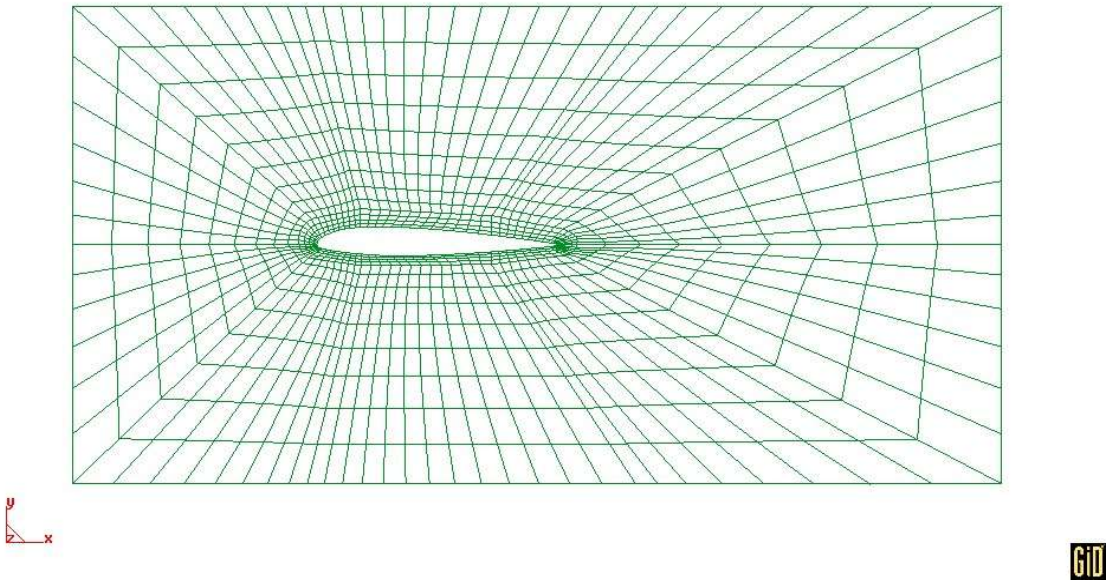

Figura 10: Malla compuesta por 1066 elementos.

La figura 11 muestra la malla obtenida para el caso de 11438 elementos. No se incluyen los resultados para las mallas más grandes ya que, dado el tamaño de los elementos más pequeños, los mismos son indistinguibles. Se alcanzaron, para esta geometría, del orden de 14

millones de elementos. Debe señalarse que no se han realizado esfuerzos en el sentido de mejorar la calidad de la malla ya que no se pretende utilizar la misma en problemas de producción. La misma puede mejorarse en gran medida definiendo y ajustando las coordenadas de los macro elementos y, si fuese necesario, aumentando el número de regiones.

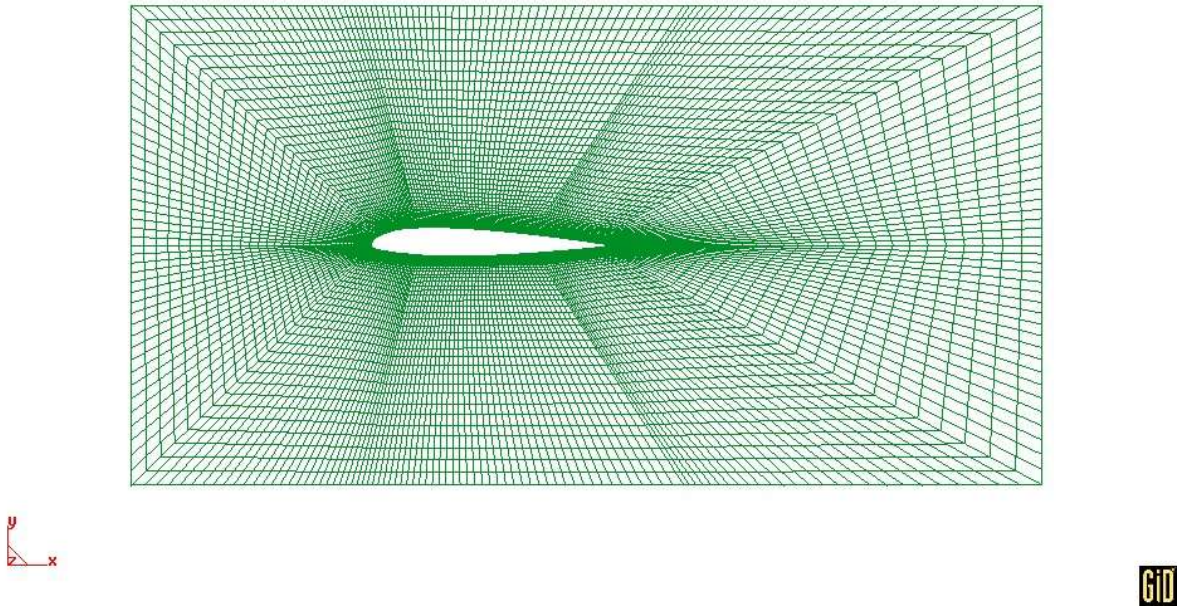

Figura 11: Malla compuesta por 11438 elementos.

Como se muestra en la figura 12 el rendimiento aumenta en este caso con el tamaño de la malla generada, al igual que en el problema del disco hueco, y se alcanza para la malla de mayor tamaño una mejora del orden de 12 veces.

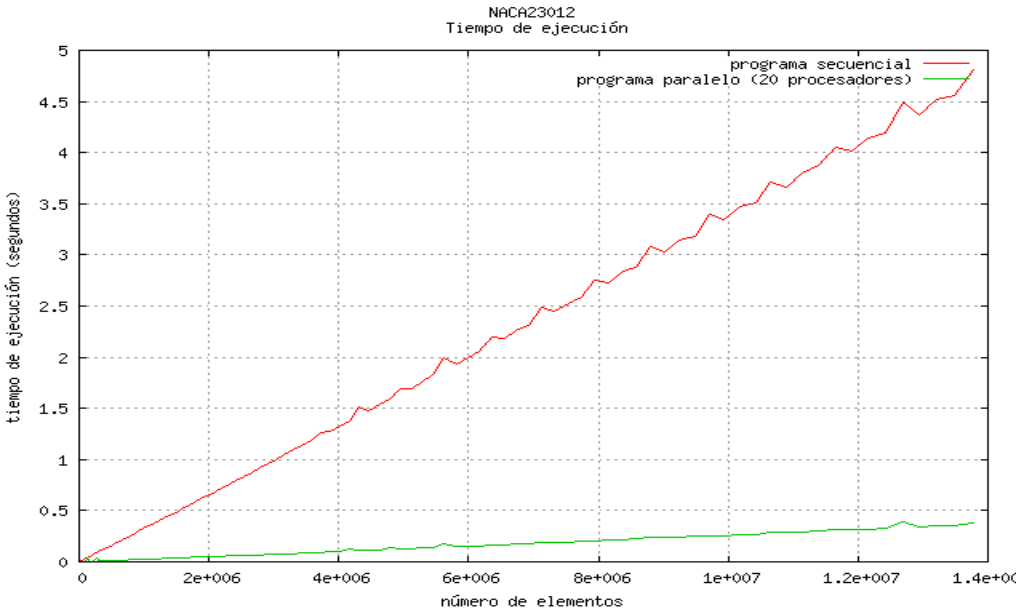

Figura 11: Comparación de los resultados obtenidos con los programas secuencial y paralelo.

5 CONCLUSIONES

El trabajo discute la versión paralela de un programa de generación de mallas estructuradas, PROGAM, cuya versión secuencial fue convenientemente mejorada y actualizada, ya que se ha programado con el lenguaje FORTRAN 90.

El diseño de la versión paralela, P-PROGAM, se ha realizado a partir de la estructura del código secuencial. La generación de la información en cada región (calculo de coordenadas de nodos y conectividad de elementos) presenta una concurrencia muy alta, que puede encuadrarse en lo que usualmente se denomina, problemas naturalmente paralelo.

Las comunicaciones son mínimas ya que solo se necesitan pasar variables enteras entre los diferentes procesadores para compatibilizar la numeración global de los nodos y la numeración de los nodos en las fronteras comunes.

También se ha podido paralelizar la escritura de resultados aspecto que evita pasar la información de toda la malla de cada región desde los hijos al master, lo que permite aumentar las prestaciones del código.

Los herramienta obtenida ha permitido materializar la característica de la aplicación naturalmente paralela, ya que el speed-up obtenido se incrementa con el tamaño de la malla generada.

Finalmente, a falta de pruebas realizadas en este sentido, los resultados obtenidos parecen sugerir que si se cuenta con hardware de buenas prestaciones los límites de la herramienta parecen estar en la memoria RAM de los nodos y no en la velocidad de proceso.

6 AGRADECIMIENTOS

Los autores agradecen a la Agencia Nacional de Promoción Científica y Tecnológica, ANPCyT, la ayuda económica recibida, a través del proyecto PICTR 184, del FONCyT.

7 REFERENCIAS

[1] Descripción del Programa de Generación Automática de Mallas (PROGAM), C. García Garino y J. Ortiz Andino, Nota Técnica Nº 21, IMPSA, 1984

[2] C. García Garino y J. Ortiz Andino: Biblioteca de Códigos PROGAM: Descripción y Aplicaciones para la generación automática de redes de elementos finitos, Anales del Quinto Congreso Latinoamericano de Métodos Computacionales en Ingeniería, Asociación Latinoamericana de Métodos Computacionales (AMC), Salvador de Bahía, Brasil, 1984.

[3] Applied Finite Element Analysis, Segerlind, Wiley, 1976

[4] Intel Fortran Compiler 8.1 for Linux, http://www.intel.com/cd/software/products/asmo-na/eng/compilers/flin/index.htm

[5] The Beowulf Cluster Site, http://www.beowulf.org.

[6] Rocks Cluster Distribution, http://www.rocksclusters.org.

[7] MPICH – A Portable Implementation of MPI, http://www-unix.mcs.anl.gov/mpi/mpich/

[8] GiD – The personal pre and postprocessor, http://www.gid.cmne.upc.es, 2005

8 APENDICE 1: DESCRIPCION DEL CLUSTER

El cluster Reloaded <reloaded.uncu.edu.ar> pertenece al Laboratorio de Producción Integrada por Computadora (LAPIC), Instituto Tecnológico Universitario, Universidad Nacional de Cuyo. Consta de 13 nodos iguales, conectados mediante un switch Gigabit Ethernet Trendware. A continuación se detalla el hardware de los nodos:

- Procesador: Intel Pentium 4 – 3.0 Ghz – Hyper Threading
- Cache L2: 1 MB
- FSB: 800 MHz
- Motherboard: Intel P4875PBZ
- Memoria RAM: 1 GB, 400MHz
- Placa de red: Gigabit Ethernet (on board)
- Disco Rígido: 80 GB Serial ATA

El nodo denominado *frontend* cumple la función de consola de administración, atiende las conexiones externas de los usuarios y posee herramientas de monitoreo del sistema. Además puede participar de los cálculos paralelos. Posee dos interfaces de red: una se conecta con el exterior y la otra con el switch interno. Los 12 nodos restantes, denominados *hijos* son exclusivamente para cálculo.

Todos los nodos tienen procesador Intel Pentium 4 con tecnología Hyper Threading. Esta tecnología permite a un procesador ejecutar dos procesos en forma paralela, otorgando al cluster la capacidad de ejecutar aplicaciones de cálculo que requieran hasta 26 procesadores.