Sociedad Colombiana de Matemáticas
XV Congreso Nacional de Matemáticas
2005 **Apuntes**

Lecturas Matemáticas
Volumen Especial (2006), páginas 213–224

Computación científica en paralelo con interfaz MATLAB a PETSc

Carlos D. Acosta
Universidad Nacional de Colombia, Manizales
Carlos E. Mejía
Universidad Nacional de Colombia, Medellín*

Abstract. PETSc (Portable Extensible Toolkit for Scientific Computation) is a powerful toolkit for parallel numerical solution of scientific applications modeled by partial differential equations. Likewise, MATLAB is a high-performance language for technical computing, which has a rich and easy to use environment for data acquisition, generation and visualization. Nevertheless, the computing power of MATLAB is not suitable for large scale problems and the data management in PETSc requires advanced programming knowledge. In this paper, we present an interface that uses MATLAB for Pre and Postprocessing and PETSc as computational engine. Numerical solutions of elliptic and hyperbolic problems are used for performance comparisons.

Key words and phrases. PETSc Interface, Parallel MATLAB, Parallel Computing, Scientific Computing.

2000 AMS Mathematics Subject Classification. 65-04, 65Y05, 65N30

Resumen. PETSc (Portable Extensible Toolkit for Scientific Computation) es un poderoso conjunto de herramientas para la solución en paralelo de aplicaciones científicas modeladas con ecuaciones diferenciales parciales. Por su parte MATLAB es un lenguaje de alto nivel para cálculo técnico que cuenta con un entorno rico en herramientas fáciles de usar para la adquisición, generación y visualización de datos. No obstante, el poder de cómputo de MATLAB no es apropiado para problemas de gran tamaño y la administración de los datos en PETSc requiere conocimientos en programación avanzada. En este trabajo

*C. D. Acosta, apoyado parcialmente por DIMA, proyecto número 20101002840. C. E. Mejía, apoyado parcialmente por DIME, proyecto número 30802867 y COLCIENCIAS, proyecto número 1118-11-16705.

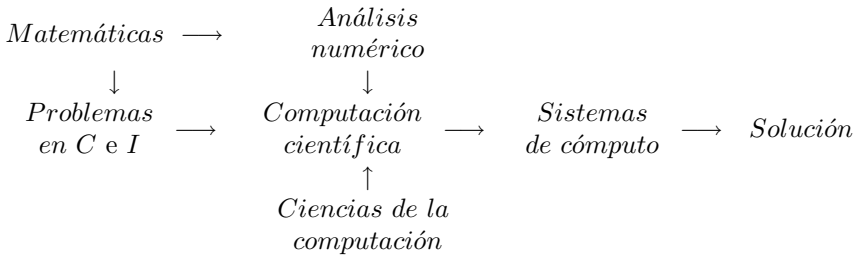
se presenta una interfaz que permitehacer el pre y posproceso de los datos en MATLAB y usar PETSc como máquina de cómputo. Soluciones numéricas a problemas elípticos e hiperbólicos son usadas como pruebas de desempeño.

1. **Introducción**

A fin de entender mejor la naturaleza e importancia del presente trabajo, se presentan en esta sección los conceptos de *computación científica* y de *computación en paralelo*. Para ello nos basaremos en textos fundamentales sobre estos temas y que clarifican el enfoque en el cual entendemos y trabajamos estas ideas.

1.1. **Computación científica.**

Como definición de computación científica emplearemos la presentada por GOLUB y ORTEGA en [4]: "La computación científica se mueve entre la matemática y las ciencias de la computación para desarrollar las mejores formas de utilizar los sistemas de cómputo para resolver problemas de ciencias e ingenierías". El siguiente esquema ilustra las interrelaciones que plantea esta definición.



El objeto de este trabajo es presentar una nueva aproximación para el uso de una de las más poderosas herramientas de las ciencias de la computación, la computación en paralelo o distribuida, en la solución de problemas de las ciencias y las ingenierías (C e I en el diagrama de arriba) conducentes a sistemas de ecuaciones lineales de gran tamaño.

Ahora, la computación en paralelo es una herramienta tan poderosa como compleja y exigente. Sus dominios incluyen aspectos como manejo de redes de computadores, administración de los mensajes y comandos que circulan entre los equipos de la red, protocolos apropiados de almacenamiento de datos, estructuras diferentes para manejo de datos, algoritmos de cómputo específicos para la estructura de la red, etc. Afortunadamente, existen proyectos como PETSc ([8], [9], [10]) que le ofrecen al usuario un paquete completo que, una vez instalado, le permite manejar los programas en paralelo desde un nivel más alto, pensando sólo en estructuras matriciales y vectoriales.

De otro lado, tenemos a MATLAB ([7]), que cuenta con gran aceptación en la comunidad científica por su gran versatilidad y su capacidad para la adquisición, generación y visualización de datos. Desafortunadamente, como

MATLAB se usa por medio de un lenguaje interpretado, los tiempos de cómputo son mayores a los empleados por lenguajes compilados como FORTRAN o C. Por tal razón MATLAB no es un entorno apropiado para resolver problemas de gran tamaño típicos de la computación en paralelo.

La idea que se desarrolla aquí es utilizar MATLAB para generar los datos del problema de gran tamaño, enviar la información a PETSc para que haga el trabajo computacional y recibir los resultados para visualización y análisis posterior. Si bien la razón de este trabajo es claramente práctica, tiene la segunda intención de motivar a los usuarios de MATLAB a que tengan un acercamiento a la computación en paralelo como alternativa real para realizar sus cálculos científicos.

1.2. **Computación en paralelo.** Este concepto lo presentamos basados en el trabajo de Foster en [2]. Según estas ideas la computación en paralelo (CP) puede pensarse como *la ejecución simultánea de dos o más procesos sobre una misma base computacional*. Existen tres modos básicos de realizar cómputo en paralelo, a saber: CP sobre arquitecturas paralelas, CP sobre arquitecturas distribuidas y CP sobre arquitecturas híbridas. Para entender mejor cada una de ellas empezamos por presentar modelos que simplifican las estructuras de cómputo para facilitar el análisis.

1.2.1. *Modelo de von Neumann.* Este modelo consiste en pensar en el computador como una CPU (Unidad de Procesamiento Central) conectada a una memoria de almacenamiento, véase la figura 1. La CPU ejecuta el programa guardado que consta de conjuntos de operaciones secuenciales de lectura y escritura sobre datos en la memoria. Este modelo se adapta muy bien a los computadores de escritorio.

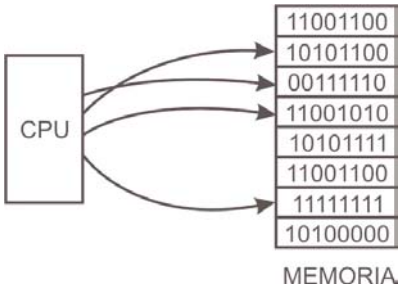


Figure 1. Modelo von Neumann

1.2.2. *Modelo multicomputador.* El modelo de multicomputador considera una máquina paralela como un conjunto de computadores tipo von Neumann, llamados nodos, que están enlazados por una red de interconexión, véase la figura

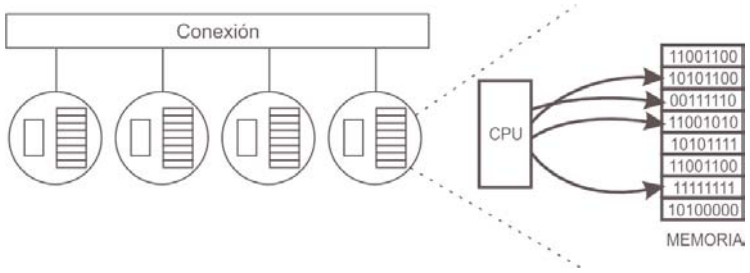


FIGURE 2. Modelo multicomputador

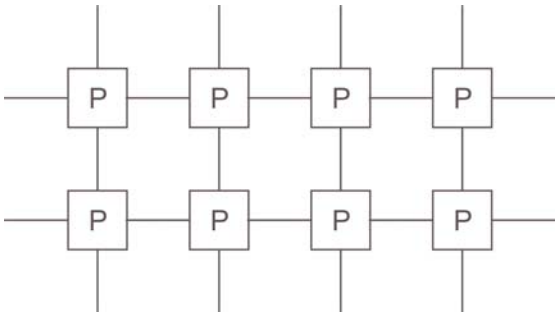


FIGURE 3. Memoria distribuida

2. Cada nodo ejecuta su programa, manejando los datos en su memoria local. Adicionalmente, cada nodo puede enviar y recibir mensajes a otros nodos, esto le da acceso a datos remotos. Existen dos tipos grandes de computadores que se acerca a este modelo ideal, el computador paralelo de memoria distribuida y el computador multiprocesador de memoria compartida.

1.2.3. *Computador paralelo de memoria distribuida.* En este caso cada procesador ejecuta una serie de instrucciones en su porción local de datos, véase la figura 3. La memoria distribuida implica la distribución de los datos sobre todos los nodos interconectados, en lugar de estar en una ubicación centralizada. La forma más generalizada es que la información del problema se particiona entre los nodos para su análisis. Por supuesto, los procesadores pueden usar la interconexión para enviar y recibir mensajes, accediendo de esta manera a información en equipos remotos. En este caso se habla de CP sobre arquitectura distribuida.

1.2.4. *Computador multiprocesador de memoria compartida.* En este modelo todos los procesadores comparten el acceso a una memoria común, usualmente a través de un canal común (*common bus*), véase la figura 4. Cada procesador

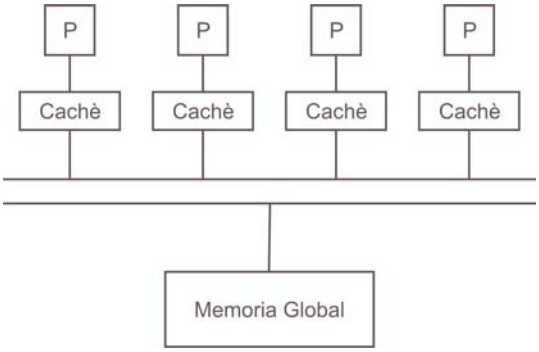


Figure 4. Memoria compartida

puede acceder cualquier posición de memoria en un momento determinado. En este modelo, generalmente, se usa una memoria caché para cada procesador, que guarda copias de los datos usados más frecuentemente, y así se optimiza el acceso a memoria. Esto para aprovechar que la velocidad de acceso a la memoria caché es mayor que el acceso a la memoria compartida. En este caso se habla de CP sobre arquitectura paralela.

Ahora, es posible configurar una red de memoria distribuida donde alguno o todos los nodos son de arquitectura paralela, en este caso es que se habla de CP sobre arquitectura híbrida.

2. **PETSc**

PETSc es la sigla inglesa para *Portable Extensible Toolkit for Scientific Computation.* El proyecto PETSc empezó en septiembre de 1991 en el seno de la División de Matemáticas y Ciencias de la Computación del *Argonne National Laboratory*, apoyado por recursos del Departamento de Energía y la Fundación Nacional para la Ciencia de los Estados Unidos. La idea original es utilizar paradigmas modernos de programación en la solución de problemas científicos de gran tamaño usando código C, $C++$ y $Fortran$. PETSc es un poderoso conjunto de herramientas para la solución numérica de ecuaciones en derivadas parciales (PDE) y problemas asociados [8], [10], [9].

El desarrollo de programas en paralelo para la solución de PDE´s es una tarea difícil. PETSc es una colección de herramientas que pueden facilitar y agilizar enormemente esta tarea. El modelo de programación con PETSc tiene como metas la portabilidad, el alto desempeño y el paralelismo escalable. La primera se entiende como la capacidad de ser instalada y ejecutada en virtualmente cualquier arquitectura computacional bajo cualquier sistema operativo. Sigue el alto desempeño, entendido como la capacidad de usar de modo eficiente los recursos disponibles buscando un balance entre los tiempos de cómputo y

los tiempos de comunicación. Por último, la escalabilidad esta relacionada con la capacidad deseable de trabajar con objetos dinámicos adaptables al tamaño de los datos y al número de procesadores en uso.

La aproximación a estas metas se hace a través de varias premisas de trabajo, a saber:

1. El modelo de máquina paralela a considerar es el de memoria distribuida.

2. No se programa para un tipo especial de compilador sino que se trabaja con base en las librerías BLAS (*Basic Linear Algebra Subprograms*), LAPACK (*Linear Algebra Package*) y MPI (*Message Passing Interface*) de la arquitectura particular.

3. El acceso a los datos en los equipos remotos se hace por comunicación por MPI; los detalles de la comunicación se ocultan e integran en la estructura misma de los objetos de trabajo (vectores, matrices, etc.) permitiendo así que el usuario los maneje desde el más alto nivel de abstracción.

2.1. **Estructura.** Revisamos enseguida la estructura de PETSc no sólo para conocer mejor la herramienta sino para especificar qué componentes numéricas vienen ya implementadas. Empezaremos desde los niveles más básicos e iremos subiendo el nivel de abstracción.

(1) Núcleos de cómputo y comunicación. Este es el nivel básico dependiente de arquitectura en el cual se implementa PETSc, esencialmente se trata de las librerías de álgebra lineal BLAS y LAPACK y de la librería de comunicación MPI.
(2) Interfaz de Reporte. Se trata de la interfaz intermedia entre los objetos PETSc y el nivel básico. Es utilizada para generar reportes de desempeño que permiten revisar la calidad de las aplicaciones. Dichos informes usan el tiempo de ejecución, el número de operaciones de punto flotante, la cantidad de mensajes y la longitud de los mismos, como indicadores de desempeño. Muestra también los objetos PETSc generados en la ejecución del aplicativo y la cantidad de recursos empleados por estos.
(3) Nivel Orientado a Objetos. Este nivel se basa en el montaje de las estructura de índices, de los vectores y las matrices. Las matrices se ofrecen de distintos tipos, Fila Esparcida Comprimida, Fila Esparcida Comprimida por Bloques, Bloques Diagonales, Densa y Matriz no almacenada (solo se conoce su acción).
(4) SLES. A este nivel se implementan los métodos de solución a sistemas lineales divididos en precondicionadores y métodos de subespacios de Krylov. Se incluyen los precondicionadores Jacobi, SOR, LU, Cholesky, LU incompleta y Cholesky Incompleta. Además, los métodos de subespacios de Krylov Gradiente Conjugado (CG), Bi-CG, LSQR y GMRES entre otros.

(5) SNES. Métodos para solución de problemas no lineales. Se incluyen procedimientos de solución por Métodos Tipo Newton de Búsqueda Lineal y Región Verdadera.
(6) TS. Métodos de avance en tiempo para Ecuaciones Diferenciales Ordinarias. Se incluyen métodos de Euler, Euler hacia atrás y seudo-avance.

Cuando se desarrolla un aplicativo basado en SLES, por ejemplo, todos los niveles anteriores quedan automáticamente incluidos y omiten los subsiguientes posibilitando usar las herramientas anteriores sin necesidad de cargar todo el paquete.

2.2. **Interfaz MATLAB - PETSc.** Adicionalmente a las funcionalidades incluidas ya en PETSc, este permite hacer uso de otras librerías y paquetes con el fin de tomar ventaja de desarrollos alcanzados por otros proyectos de computación de alto desempeño. Por ejemplo, SuperLU que es una librería de propósito general para la solución directa de problemas lineales de gran tamaño [11]. Otro ejemplo es ParMetis, que es parte del proyecto Metis y que implementa una amplia variedad de algoritmos para particionar grafos y mallas no estructuradas y el manejo óptimo de matrices esparcidas [5].

Sin embargo, nuestro principal interés es la capacidad de PETSc para interactuar con el paquete MATLAB, esto debido a la alta aceptación y preferencia que tiene éste entre la comunidad académica. Lo que se pretende es aprovechar la alta difusión de MATLAB y su amplia base de usuarios para acercar una mayor población a la computación en paralelo a través de una interfaz apropiada entre los dos paquetes.

PETSc cuenta con tres formas predeterminadas de interactuar con MATLAB: La primera consiste en poder imprimir en pantalla las salidas de programas PETSc en formato ASCII reconocible por MATLAB, opción esta que puede resultar incómoda cuando el tamaño del problema es muy grande. La segunda forma es enviando salidas de programas PETSc a una sesión activa de MATLAB pero esto requiere que el usuario destine espacios en memoria para ello y la ejecución del programa PETSc se suspende hasta que el usuario acepta, de forma manual, la transferencia del dato. La tercera opción consiste en la posibilidad de que una rutina PETSc invoque MATLAB para que realice cálculos para ella, lo cual es lo contrario de lo que nosotros proponemos, que es utilizar a PETSc para que realice cómputos para MATLAB.

La interfaz que proponemos se encarga de: generar la matriz A y el vector b del sistema lineal, activar y configurar el cluster de la plataforma de cómputo, configurar e invocar la rutina PETSc encargada de la solución del sistema, enviar los resultados directamente al espacio de trabajo de MATLAB y desactivar el cluster. La interfaz es gráfica, sencilla, intuitiva y completa.

La idea es que para resolver un problema específico el usuario use un esquema del tipo

$$\text{Preproceso} \rightarrow \text{Proceso} \rightarrow \text{Posproceso}.$$

En el preproceso el usuario debe desarrollar una rutina MATLAB sujeta a su problema específico, que reduzca su problema a un problema lineal esparcido y almacene la matriz de coeficientes y el lado derecho en un archivo".mat". Para esto el usuario debe hacer uso de sus conocimientos previos de MATLAB y de las facilidades de éste para adquisición, generación y almacenamiento de datos vectoriales y matriciales. La etapa de proceso en sí está a cargo de PETSc. En este punto el usuario ejecuta una interfaz gráfica de MATLAB en la que debe indicar el archivo que contiene los datos, el método de solución que quiere usar para el sistema, el número de procesos simultáneos que desea utilizar, etc. Luego ordena la solución del problema y PETSc hace su trabajo. Los resultados obtenidos son cargados al entorno de trabajo de MATLAB o almacenados a un archivo, segun escogencia del usuario. Una vez resuelto el sistema lineal y con la solución disponible, el usuario puede hacer uso de las excelentes herramientas de visualización y manejo de datos de MATLAB para analizar sus resultados, este es el llamado Posproceso. Es importante notar que un usuario sin mucho conocimiento de programación avanzada puede ejecutar con éxito una interfaz de este estilo.

Ahora, aunque el esquema propuesto de Preproceso -> Proceso -> Posproceso es por naturaleza secuencial, el hecho de utilizar cómputo paralelo con PETSc para la etapa de Proceso, permite mejorar grandemente los tiempos de ejecución. Adicionalmente, para las etapas de Preproceso y Posproceso, usualmente menos exigentes computacionalmente, se pueden utilizar estrategias de vectorización para las que MATLAB está bien preparado. De esta manera también se mejora el desempeño.

3. **Resultados experimentales**

En esta sección se presentan dos ejemplos utilizados para explorar la calidad de la interfaz desarrollada. El primero es un problema estacionario de tipo elíptico y el segundo es un problema dinámico de tipo hiperbólico. Como plataforma computacional se utilizó un cluster bajo Linux Fedora Core 4, conformado por dos equipos Intel Pentium 4 de 3.0GHz y 512Mb de RAM. Para el nivel básico de PETSc se uso LAM-MPI ([1]) para administrar los mensajes entre los nodos y las versiones de LAPACK y Blas incluidas en la distribución Linux como núcleo de cómputo. La versión MATLAB usada fue la 7.0 R14 SP1.

Ejemplo 01. Tomado de [6]. Aplique gradiente conjugado precondicionado para resolver por una discretización de diferencias finitas de cinco puntos el problema

$$-u_{xx} - u_{yy} + \exp(x+y)u = 1, 0 < x, y < 1,$$

sujeto a las condiciones de frontera no homogéneas

$$u(x,0) = u(x,1) = u(1,y) = 0, u(0,y) = 1, 0 < x, y < 1.$$

Solución. Para un entero positivo n fijo, discretizamos $[0,1] \times [0,1]$ con tamaño de paso $h = 1/(n+1)$. Obtenemos entonces un sistema de n^2 incógnitas y ecuaciones, el cual ha de ser muy grande si n es suficientemente grande. Note además que la matriz no es Toeplitz ni el lado derecho tiene todas sus componentes iguales. Esto por la presencia de $\exp(x+y)u$ en la ecuación y la no homogeneidad de la condición Dirichlet. Concretamente, se obtiene un sistema de la forma

$$v_i^{j-1} + v_{i-1}^j - (4 + h^2 \exp(x_i + y_j))v_i^j + v_{i+1}^j + v_i^{j+1} = h^2,$$

donde v_i^j es nuestra aproximación de $u(x_i, y_j)$, $x_i = ih$ y $y_j = jh$.

Para razones comparativas el problema fue resuelto de tres modos distintos, todos usando los mismos datos, métodos y criterios de parada. El primero fue mediante una rutina MATLAB, el segundo fue mediante la interfaz MATLAB-PETSc recién descrita y la tercera mediante una rutina PETSc que incluía preproceso y proceso, sin usar MATLAB. Los resultados para distintos valores de n se presentan en las siguientes tablas. El parámetro np indica el número de procesos en que se dividió la solución del problema, debe tenerse en cuenta que un procesador puede administrar varios procesos al tiempo.

Tiempos para MATLAB

$n = 128$	$n = 256$	$n = 512$	$n = 1024$
2.76s	22.06s	4m30s	++30m

Tiempos para la interfaz sobre dos computadores

	$n = 128$	$n = 256$	$n = 512$	$n = 1024$
$np = 1$	1.01s	3.53s	22.3s	3m25s
$np = 2$	1.04s	2.86s	15.5s	1m53s
$np = 4$	1.46s	3.71s	19.4s	2m44s

PETSc sin Interfaz sobre dos computadores

	$n = 128$	$n = 256$	$n = 512$	$n = 1024$
$np = 1$	0.90s	2.64s	25.85s	3m12s
$np = 2$	0.88s	2.50s	15.40s	1m48s
$np = 4$	1.17s	3.34s	19.40s	1m35s

Este simple ejemplo muestra que la interfaz desarrollada permite obtener tiempos de solución ostensiblemente mejores a los de MATLAB. Ahora, aunque los tiempos de PETSc sin interfaz siguen siendo inferiores a los tiempos con interfaz, es importante tener en cuenta que la rutina PETSc que trabaja asociada a la interfaz debe ser una rutina de tipo bastante general que debe estar preparada para trabajar con una amplia gama de matrices y de opciones, en tanto que la rutina PETSc libre de interfaz fue desarrollada para el problema específico, depende del ejemplo y por tanto se ajusta mejor a él. En todo caso

se justifica este tipo de interfaz como punto intermedio válido en el camino de búsqueda de equilibrio entre el costo de desarrollo y la calidad del desempeño.

Ejemplo 02. Se pretende resolver para u, en

$$\begin{array}{ll} u_{tt} - \nabla \circ (a \nabla u) = f, & (x,y) \in Int(I \times I) \\ u(x,y,t) = g(x,y,t), & (x,y) \in \partial(I \times I) \\ u(x,y,0) = u_0(x,y), & (x,y) \in Int(I \times I). \end{array}$$

Los datos del problema son generados a partir de la solución deseada

$$u(x,y,t) = \exp(-20(y - t - 1/4)^2)$$

y tomando $a(x,y) = 1 + x^2$. Esto ha de generar un pulso gaussiano moviéndose en un dominio rectangular.

Solución. Se utilizan elementos finitos en espacio para reducir el problema a uno semidiscreto de la forma

$$\left\{ \begin{array}{c} Mv'' + Kv = F. \\ v(0) = p, \\ v'(0) = q. \end{array} \right. ,$$

donde v depende sólo del tiempo y almacena los coeficientes requeridos para recuperar u en el instante de tiempo actual. Usamos ahora diferencias finitas centradas y llamamos w la aproximación a encontrar, tenemos el sistema:

$$Mw_{n+1} = \left(2M - k^2 K\right) w_n - Mw_{n-1} + k^2 F_n.$$

Para preprocesar este problema se realizaron unas modificaciones al Toolbox que acompaña el libro [3], a fin de poder trabajar problemas multidimensionales y con notación vectorial. Ahora, dado que el vector F_n depende sólo de la condición de contorno, estos se calcularon de manera simultánea y la interfaz generada entonces toma como datos las matrices de masa M, la matriz de rigidez K y una colección de los F_n. Se quiere experimentar en este ejemplo la dificultad de manejar el tráfico de datos cuando se trabaja con varios lados derechos (RHS). Las tablas siguientes ilustran los resultados obtenidos para varias cantidades de los F_n pasados al tiempo a la interfaz.

Orden de cada sistema	1024
Cantidad de sistemas	1024
Tiempo para MATLAB	32s

Tiempo para la interfaz con dos computadores

RHS	16	32	64	128
np = 1,	59.88s	58.10s	88.03s	3m29s
np = 2,	68.95s	43.83s	37.14s	51.26s
np = 4,	1m42s	74.33s	63.56s	61.10s

Orden de cada sistema	4096
Cantidad de sistemas	1024
Tiempo para Matlab	$4m46s$

Tiempo para la Interfaz con dos computadores

RHS	16	32	64
np = 2,	5m21s	2m57s	5m05s
np = 4,	2m34s	3m50s	5m24s
np = 8,	3m30s	6m01s	3m29s

En este ejemplo se aprecian dos cosas, la primera es que se debe tener en cuenta la cantidad de información que se pasa como dato a fin de optimizar la relación entre los tiempos de carga de los mismos y de los programas. La segunda es que al usar una herramienta como PETSc se cuente con un problema de un tamaño suficientemente grande debido a que en problemas de tamaño pequeño es más lo que se tarda en cargar el paquete que en resolver el problema, esto es lo que explica las cifras de la primera tabla. En la segunda ya se empieza a notar un mejor desempeño de PETSc, esencialmente debido al incremento en el tamaño del sistema.

Bibliografía

[1] Burns G., Daoud R. y J. Vailg. *LAM : An Open Cluster Environment for MPI*, Proceedings of Supercomputing Symposium, 379–386, 1994. `http://www.lam-mpi.org`

[2] Foster I. *Designing and Building Parallel Programs*, Addison-Wesley, 1995.

[3] Gockenbach M. *Partial Differential Equations, Analytical and Numerical Methods*, SIAM, 2004.

[4] Golub G. y J. Ortega. *Scientific Computing : An Introduction with Parallel Computing*, Academic Press, 1993.

[5] Karypis G., *Multi-Constraint Mesh Partitioning for Contact/Impact Computations*, Supercomputing, 2003.

[6] Kelley C.T. *Iterative Methods for Linear and Nonlinear Equations*, SIAM, 1995.

[7] MathWorks, *Matlab*, The MathWorks Inc., Natick, MA, 2006.

[8] Satish B., Buschelman K., Gropp W., Kaushik D., Knepley M., Curfman L., Smith B. y H. Zhang. *PETSc Web Home*, 2001. `http://www.mcs.anl.gov/petsc`

[9] Satish B., Buschelman K., Gropp W., Kaushik D., Knepley M., Curfman L., Smith B. y H. Zhang. *PETSc User Manual, ANL-95/11 - Revision 2.1.5*, Argonne National Laboratory, 2004.

[10] Satish B., Buschelman K., Gropp W., Kaushik D., Knepley M., Curfman L., Smith B. y H. Zhang. *Efficient Management of Parallelism in Object Oriented Numerical Software Libraries*, Modern Software Tools in Scientific Computing (E. Arge, A. M. Bruaset y H. P. Langtangen, Ed), Birkhäuser Press, pp 163–202, 1997.

[11] Xiaoye S. L. y J. W. Demmel. *SuperLU_DIST: A Scalable Distributed-Memory Sparse Direct Solver for Unsymmetric Linear Systems, ACM Trans.* Mathematical Software, **29**(2003), pp. 110-140.

(Recibido en abril de 2006. Aceptado para publicación en abril de 2007)

DEPARTAMENTO DE MATEMÁTICAS Y ESTADÍSTICA
UNIVERSIDAD NACIONAL DE COLOMBIA
MANIZALES, COLOMBIA
e-mail: cdacostam@unal.edu.co

ESCUELA DE MATEMÁTICAS
UNIVERSIDAD NACIONAL DE COLOMBIA
MEDELLÍN, COLOMBIA
e-mail: cemejia@unal.edu.co