Computación Científica Paralela mediante uso de herramientas para paso de mensajes[1]

Artículo de Revisión - Fecha de recepción: 14 de septiembre de 2012 - Fecha de aceptación: 1 de octubre de 2012

Francisco J. Fernández
Doctor en Informática, Universidad de Granada. Granada, España, jfernand@ugr.es

Mancia Anguita
Doctora en Informática, Universidad de Granada. Granada, España, manguita@ugr.es

RESUMEN

Los usuarios de Entornos de Computación Científica (SCE, por sus siglas en inglés) siempre requieren mayor potencia de cálculo para sus aplicaciones. Utilizando las herramientas propuestas, los usuarios de las conocidas plataformas Matlab® y Octave, en un cluster de computadores, pueden paralelizar sus aplicaciones interpretadas utilizando paso de mensajes, como el proporcionado por PVM (Parallel Virtual Machine) o MPI (Message Passing Interface). Para muchas aplicaciones SCE es posible encontrar un esquema de paralelización con ganancia en velocidad casi lineal. Estas herramientas son interfaces prácticamente exhaustivas a las correspondientes librerías, soportan todos los tipos de datos compatibles en el SCE base y se han diseñado teniendo en cuenta el rendimiento y la facilidad de mantenimiento. En este artículo se resumen trabajos anteriores, su repercusión, y algunos resultados obtenidos por usuarios finales. Con base en la herramienta más reciente, la Toolbox MPI para Octave, se describen brevemente sus características principales, y se presenta un estudio de caso, el conjunto de Mandelbrot.

Palabras clave

Computación científica, programación paralela, computación de altas prestaciones, computación cluster, paso de mensajes, Matlab paralelo.

1 Este trabajo ha sido financiado por el proyecto español ARC-VISION (MICINN TEC2010-15396) y el proyecto europeo TOMSY (FP7-ICT-2009-6-270436).

Parallel Scientific Computing with message-passing toolboxes

ABSTRACT

Users of Scientific Computing Environments (SCE) always demand more computing power for their CPU-intensive SCE applications. Using the proposed toolboxes, users of the well-known Matlab® and Octave platforms in a computer cluster can parallelize their interpreted applications using the native multi-computer programming paradigm of message-passing, such as that provided by PVM (Parallel Virtual Machine) and MPI (Message Passing Interface). For many SCE applications, a parallelization scheme can be found so that the resulting speedup is nearly linear on the number of computers used. The toolboxes are almost comprehensive interfaces to the corresponding libraries, they support all the compatible data types in the base SCE and they have been designed with performance and maintainability in mind. In this paper, we summarize our previous work, its repercussion, and some results obtained by end-users. Focusing on our most recent MPI Toolbox for Octave, we briefly describe its main features, and introduce a case study: the Mandelbrot set.

Keywords

Scientific computing, parallel programming, high performance computing, cluster computing, message-passing, parallel Matlab.

EDUCOSTA
EDITORIAL UNIVERSITARIA DE LA COSTA

INTRODUCCIÓN

Matlab® [1] es un "lenguaje de programación de alto nivel y un entorno interactivo de computación técnico-científica. Incluye funciones para el desarrollo de algoritmos, análisis de datos, cálculo numérico y visualización" [2]. Usando Matlab se pueden resolver problemas de computación técnica más rápidamente que con lenguajes de programación tradicionales, como C, C++ o Fortran [3].

GNU Octave [4]-[7] es un "lenguaje interpretado de alto nivel, destinado principalmente a cálculos numéricos. Proporciona recursos para la solución numérica de problemas lineales y no lineales, y para realizar otros experimentos numéricos. También proporciona capacidades gráficas extensas para visualización y manipulación de datos" [8].

Ambos entornos permiten la programación de scripts o funciones, redactados en el lenguaje interpretado (los denominados "ficheros-M"), así como de programas externos redactados en Fortran, C o C++, usando la interfaz externa Matlab o la librería Octave (los denominados "ficheros-MEX" o "funciones-DLD", respectivamente).

Esta última opción (programas externos) es la que nos permite construir un interfaz a una librería, de paso de mensajes en nuestro caso, ya sea MPI (Message Passing Interface) o PVM (Parallel Virtual Machine). La Fig. 1 muestra gráficamente cómo la aplicación del usuario se apoya en funciones de la plataforma SCE (Scientific Computing Environment) y de la herramienta de paso de mensajes (Octave y MPI ToolBox en el dibujo) pudiendo, por lo tanto, comunicarse con otras instancias del SCE, ejecutándose en otros nodos del cluster.

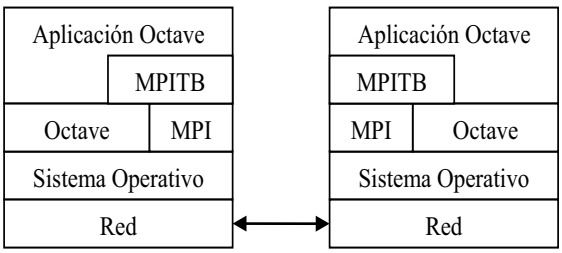

Fig. 1 Resumen gráfico del papel jugado por la herramienta. Las funciones de la *ToolBox* invocan a rutinas de la plataforma SCE y de la librería de paso de mensajes para intercomunicar procesos SCE ejecutándose en distintos nodos de un cluster.

PVM (Parallel Virtual Machine) [9] es un software que "permite usar una colección heterogénea de computadores Unix/Windows interconectados por red como si fueran un gran computador paralelo, permitiendo a los usuarios explotar su hardware de computación ya existente para resolver problemas mucho mayores a un mínimo coste adicional" [10]. Naturalmente, PVM también puede usarse como mecanismo de paso de mensajes en un cluster dedicado.

MPI (Message Passing Interface) [11]-[13] es un estándar de paso de mensajes del cual existen diversas implementaciones de alta calidad, tanto propietarias como libres.

LAM/MPI (Local Area Multicomputer) [14], [15] "implementa completamente el estándar MPI-1.2 y gran parte del MPI-2.0. Además de altas prestaciones, LAM proporciona una serie de características de usabilidad claves para desarrollar aplicaciones MPI de gran escala" [16].

Open-MPI [17] es una "implementación Open-source de MPI-2 desarrollada y mantenida por un consorcio de colaboradores de la academia, investigación e industria" [18]. Sigue en desarrollo activo, y parte del anterior equipo de LAM continúa trabajando ahora en Open-MPI.

Las herramientas PVMTB (PVM Toolbox) y MPITB son interfaces a las librerías PVM y MPI para las plataformas Matlab y Octave, permitiendo así a los usuarios de dichas plataformas paralelizar sus propias aplicaciones interpretadas por medio de paso de mensajes explícito. Se espera por tanto de los usuarios finales que modifiquen sus aplicaciones SCE (interpretadas, secuenciales, en forma de ficheros-M) para adaptarlas a la ejecución paralela, añadiéndoles rutinas de paso de mensajes (como `MPI_Send` y `MPI_Recv`, por ejemplo, para enviar y recibir variables, respectivamente), las cuales se pueden invocar desde la plataforma gracias al interfaz de nuestras herramientas, y que en último término invocarán a la librería de paso de mensajes correspondiente.

Hemos desarrollado también una serie de utilidades (ficheros-M) de más alto nivel, programadas sobre este interfaz directo a las rutinas de paso de mensajes, para facilitar al usuario la paralelización de su aplicación y reducir dichas modificaciones a un mínimo.

Se hace un resumen de trabajos previos nuestros y su repercusión, destacando algunos resultados obtenidos por otros usuarios finales. A continuación se describen brevemente las características principales de nuestra herramienta más reciente, MPITB para Octave (basada en Open-MPI). Se presenta, entonces, un estudio de caso: el Conjunto de Mandelbrot, para el cual es fácil obtener un esquema de paralelización con ganancia casi lineal, independientemente de que existan en la literatura reportes técnicos con peor escalabilidad. Luego se discuten brevemente estos resultados. Y, por último, se presentan algunas conclusiones de este trabajo.

TRABAJOS PREVIOS Y APLICACIONES DE USUARIOS

Iniciamos el desarrollo de estas herramientas en el año 1999 [19]-[23]. Incluso siendo [19] una referencia a un Congreso Matlab en castellano, ha sido la única forma de citar PVMTB durante un tiempo ([24]-[28]), aparte de mencionar la página web. Conforme [20]-[21] fueron apareciendo, los usuarios han podido usarlos como citas ([29]-[33]). Las referencias [22]-[23] han jugado un papel similar para la herramienta MPITB para Octave ([34]-[39]). Los usuarios de MPITB para Matlab, desarrollada entre medias, también han recurrido a [20]-[21] y la web de MPITB (o sencillamente su nombre) para referenciarla ([40]-[43], [29]-[30]).

Otros investigadores trabajando en herramientas paralelas para SCE han usado habitualmente [21] y la web MPITB para referirse a nuestro trabajo en sus artículos en revistas ([44]-[48]) y contribuciones a congresos ([49]-[55]). En algún caso desta-

cable ([50],[53]), excelentes resultados en prestaciones fueron referenciados simplemente con el nombre de nuestra herramienta (MPITB). Las páginas web de PVMTB y MPITB han sido citadas incluso en capítulos de libros ([55],[56]). Usar la web o el nombre de nuestras herramientas ha sido también la forma más habitual de referirse a nuestro trabajo en reportes técnicos ([57]-[59]), de nuevo con excelentes resultados en prestaciones, particularmente [57]. Algunos paquetes software ([60]-[61]) y distribuciones ([62]-[64]) funcionan con, o han incluido, algunas de nuestras herramientas.

Las herramientas también han sido provechosamente usadas para cursos y seminarios ([65]-[71]) y han sido mencionadas en diversas tesis ([72]-[76]), particularmente en [73], que incluye la revisión de Matlabs paralelos de Ron Choy [77]. La inclusión de PVMTB y MPITB en esta revisión, ampliamente difundida y reconocida desde su creación [78], ha aumentado seguramente la visibilidad de nuestro trabajo.

Evolución inicial de los SCE paralelos y el papel del paso de mensajes en ellos

El uso directo de primitivas de paso de mensajes en plataformas SCE ha sido frecuentemente cuestionado, siendo consideradas de muy bajo nivel para el alto nivel de programación de estas plataformas, pudiendo ser este el motivo de que no existiera ninguna herramienta llamada PVM o MPI ToolBox cuando iniciamos nuestros desarrollos. Las herramientas más visibles en aquel entonces[1] eran DP-TB [79]-[83], PT [84] y MultiMatlab [85], [86], esta última con apoyo activo por parte de *The MathWorks* (ver la sección de Reconocimientos en ambas referencias). DP-TB realizaba una interfaz a un subconjunto de rutinas de PVM en un paquete de bajo nivel llamado DP-LOW, que era entonces usado para construir otro de más alto nivel, DP-HIGH. PT introducía el concepto de *daemon del motor PT (PT Engine daemon)* como una capa intermedia para evitar exposición directa del usuario al mecanismo PVM subyacente. Por su parte, MultiMATLAB utilizaba MPI, e incluía comandos para evitar en la medida de lo posible el uso directo de primitivas de paso de mensajes o la programación coordinada entre maestro y esclavos, aunque algunas primitivas MPI básicas (`Send`, `Recv`, `Barrier`) tuvieron que ser incluidas para obtener la funcionalidad deseada. MultiMATLAB tuvo un sucesor (desarrollado también en Cornell, pero por distintos investigadores) llamado CMTM [87] con interfaz a muchas más rutinas MPI. Otros desarrollos como PPServer, MITMatlab, Matlab*P [88]-[92], evitaron primitivas de paso de mensajes mediante programación orientada a objetos, polimorfismo y sobrecarga de operadores para obtener paralelismo de datos, y fueron prontamente reconocidos por el propio Moler [93].

Adicionalmente, era posible descargar del sitio web de *The MathWorks* diversas herramientas orientadas a otros objetivos pare-

1 Ver, por ejemplo, URL: http://groups.google.es/group/comp.soft-sys.matlab/browse_frm/thread/1e7bd3e3b9d412cf

cidos: paralelización de bucles for, comunicación TCP/IP, etc. Tanto la proliferación de este tipo de herramientas como la de preguntas de usuarios en los diversos foros de comunicación habituales (grupos de noticias, listas de distribución de e-mail, etc.) reflejaban la creciente demanda por parte de los usuarios, conforme los clusters de computadores se iban convirtiendo en equipo de uso habitual. Esta evolución recuerda a la de Matlab incorporando LAPACK cuando los tamaños de caché requeridos se volvieron habituales [94].

El inicio de los trabajos de Pawletta y colegas sobre DP-TB puede retrotraerse documentadamente hasta un tempranísimo 1992 ([79], p. 2), y evolucionó hacia el área de HLA (High Level Architecture) y simulación basada en DEVS (Discrete EVent systems Specification). El trabajo iniciado con PPServer en 1998 eventualmente evolucionó con Choy, Edelman y colegas hacia Star-P [45], [46], [95]-[98], que fue el ganador $1K en la categoría Software de la Competición de Emprendedores del MIT en 2003. Según el Resumen en la referencia [99], Star-P comercializaba no solo los proyectos de investigación del MIT (MITMatlab, Matlab*P) sino también los de UCSB (MatlabMPI, y pMatlab, construido a partir del primero), desarrollados principalmente por Kepner, Bliss Travinin y colegas [44], [49], [100]-[106].

Durante este tiempo, *The MathWorks* desarrolló la Distributed Computing Toolbox (DCT, [107], ahora llamada Parallel CT), incluyendo funciones basadas en MPI desde la versión 2.0, y operaciones globales desde poco después [108], siguiendo por tanto una evolución parecida a la de las herramientas de Cornell (MultiMatlab-CMTM) en donde el paso de mensajes se incluye finalmente, bien como solución alternativa, o como único medio de proporcionar la funcionalidad deseada. Matlab*P también incorporaba un "modo MultiMatlab" denominado *PPEngine* ([45], Sect. VIII B).

Este subconjunto de Matlabs paralelos (DCT, Star-P, MatlabMPI-pMatlab y MPITB) eran los más mencionados en esta época [46], y perseveraban en popularizarse ([99], [104], [108]-[119]) coincidiendo con un auge del interés por computación cluster, probablemente debido a la amplia difusión de este tipo de plataforma paralela.

Aplicaciones de usuarios finales

La importancia que tienen la visibilidad, popularización, número de usuarios y soporte para este tipo de herramientas no puede ser más claramente resaltada que como se hizo en la referencia [120] del Libro Blanco de la TFCC [121] (Task Force on Cluster Computing, unida al TCSA o Technical Committee on Supercomputing Applications en junio de 2000 para formar el TCSC on Scalable Computing). Chapin y Worringen [120] resaltaron allí, refiriéndose a la computación cluster, que "muchos enfoques de computación superiores tanto técnica como intelectualmente fracasaron debido a la falta de soporte […] Este soporte depende fundamentalmente del número de usuarios". Atraer a nuevos usuarios era

EDUCOSTA
EDITORIAL UNIVERSITARIA DE LA COSTA

por tanto fundamental para el éxito de la computación cluster en el contexto general de [120], y para el de este tipo de Entornos de Computación Científica paralelos en particular. La estrategia de PVMTB y MPITB para atraer a usuarios es proporcionar ganancia en velocidad lineal para aplicaciones que presenten escalabilidad lineal, esto es, garantizar que la herramienta no cause una pérdida de prestaciones significativa.

En trabajos anteriores [23] ya destacamos los excelentes resultados obtenidos por Goasguen (entonces en Purdue, IN, EE.UU., [29], [30], ganancia 101 con 120 procesadores, eficiencia 84%), Dormido (UNED, España, [31]-[33], ganancia 12 con 15 procesadores, eficiencia 80%) y Creel (UAB, España, [34]-[36], ganancias 9-11 con 12 procesadores, eficiencia 75-92%). Es crédito de Creel haber obtenido ese 75% de eficiencia para un algoritmo MLE (Maximum Likelihood Estimation) usando un método de optimización quasi-Newtoniano BFGS (Broyden–Fletcher–Goldfarb–Shanno), que no es el tipo de aplicación de la que se espera que escale, ni linealmente ni de otra forma.

En esta ocasión, destacamos no solo resultados de ganancia en trabajos de investigación, sino también usos académicos y trabajos de portabilidad a otras plataformas por parte de usuarios finales.

B. Ó Nualláin (UvA, Holanda) y colegas [37]-[39] han usado MPITB para Octave en Ciencias Geológicas e Hidrometeorología. En la referencia [38] (Figs. 6 y 8) se pueden observar ganancias casi lineales para su algoritmo SCEM-UA (Shuffled Complex Evolution Metropolis) hasta un total de 25 computadores. Ó Nualláin es también Integrador de Sistemas para MPITB y otras herramientas paralelas en su distribución VL-E (Virtual Lab for E-Science) [64]. Aportó un parche para MPITB que ahora figura con agradecimientos en el código fuente.

J. August y T. Kanade (Robotics Institute, CMU, EE.UU.) [40] usaron MPITB para Matlab en Tomografía Computarizada de rayos-X, obteniendo ganancias lineales para tamaños de imagen razonables hasta el total de 20 computadores usados. Tuvieron que reducir premeditadamente el tamaño de las imágenes procesadas hasta 64x64 y 16x16 pixels para conseguir rebajar la ganancia en velocidad hasta 17 y 12 (eficiencias 85-60%).

T. Varslot (ahora en ANU, Australia) y S. E. Måsøy (NTNU, Noruega) [41] también usaron MPITB para Matlab en generación de imágenes médicas por ultrasonidos. En sus conclusiones, destacaron que su modelo pseudo-diferencial para propagación de ultrasonidos resultó particularmente fácil de distribuir sobre un gran número de procesadores. MPITB se utilizó para extender el código secuencial 2D a simulaciones paralelas 3D.

B. Skinner (UTS, Australia) [71] preparó una introducción a MPITB para los usuarios del cluster HPC en la Facultad de Ingeniería. MPITB fue preinstalado y confi-

gurado en los nodos del cluster, quedando disponible para todos los usuarios.

W. LAM ha portado MPITB de nuestra plataforma que soporta Linux-LAM a Windows-MPICH. La cantidad de ayuda que solicitó para ello fue mínima[2].

CARACTERÍSTICAS DE LAS HERRAMIENTAS

Principios de diseño: Prestaciones y mantenibilidad

Como se detalló en [23], las elecciones de diseño se realizaron considerando las prestaciones y mantenibilidad. Detalles identificativos como por ejemplo la creación de una función-DLD para cada llamada con interfaz a Octave, la clasificación de llamadas MPI según su signatura de parámetros, o siempre pasar tamaño y puntero a buffers (interfaz con 0-copias) en lugar del propio objeto buffer, son todos ellos características distintivas de MPITB. Aunque para este trabajo nos centramos en MPITB para Octave, los mismos principios se cumplen para las otras herramientas. El diseño está orientado a ahorrar tiempo tanto al ejecutar el código de interfaz (prestaciones) como al actualizar el código a nuevas versiones de la librería Octave (mantenibilidad).

Como consecuencia directa de estas elecciones, las variables Octave usadas como buffers MPI deben aparecer en la parte derecha (RHS, Right Hand Side) del signo igual en la definición de funciones MPITB, esto es, definimos `[info,stat]=MPI_Recv(buf,src,tag,comm)` en lugar de devolver una LHS Octave (Left Hand Side) como sucedería con `[buf,info,stat]=MPI_Recv(src,tag,com)`. Eso sería una elección de diseño de bajas prestaciones, ya que implica construir el objeto `buf` devuelto. Esa construcción podría ser una operación muy costosa, particularmente en el contexto de un test ping-pong o una aplicación con un patrón de comunicaciones intensivo. En lugar de eso, preferimos exigir que los buffers sean símbolos Octave (variables con nombre) pasados como parámetros de entrada, de manera que el valor de salida pueda retornarse en la propia variable de entrada. Esta característica ha sido frecuentemente criticada como de "bajo nivel", porque impide al usuario escribir código más compacto como `a=b+MPI_Recv(...)`. Otra crítica que se le puede realizar es que se opone a la semántica normal de "paso-por-valor" del lenguaje interpretado, en donde los datos RHS siempre son copiados, nunca modificados. En nuestra experiencia [21], son estas dos decisiones, 0-copia y retorno RHS, las que garantizan que las prestaciones serán superiores a las de cualquier otra herramienta que no las observe.

MPITB proporciona interfaz para prácticamente todo MPI-1.2 y algunas llamadas MPI-2.0, en total unos 160 comandos. Ninguna otra herramienta realiza una cobertura mínimamente comparable. Las llamadas que faltan del estándar MPI-1.2 son aquellas

2. Diálogo archivado en URL: http://groups.google.es/group/comp.soft-sys.matlab/browse_frm/thread/3d5bb656f09e8aea. URL original de descarga: http://www.wakun.com/download.htm. Enlace recuperable mediante http://web.archive.org/web/20070101144316/http://www.wakun.com/download.htm

que no tienen sentido bajo el entorno Octave, tales como la familia `MPI_Type_*`. Por ejemplo, con MPITB se pueden programar funciones para copia/borrado de atributos o gestores de error arbitrarios en lenguaje Octave, y se pueden registrar con la librería MPI a través del correspondiente comando MPITB (`MPI_Errhandler_{create, set}`, `MPI_Keyval_create`, `MPI_Attr_put`) de forma que sean invocados por la librería MPI cuando ésta detecte un error o necesite copiar o borrar un atributo (`MPI_Attr_delete`, `MPI_Comm_{dup, free}`). O también se pueden publicar nombres de puertos en un servidor, de forma que puedan ser consultados por clientes para establecer comunicación (`MPI_{Publish,Lookup}_name`, `MPI_Comm_{accept, connect}`).

MPITB soporta todos los tipos de datos Octave compatibles con MPI, incluyendo celdas (*cells*), rangos, booleanos, enteros con y sin signo desde `int8` hasta `uint64`, estructuras, etc. En concreto, los tipos básicos no solo pueden enviarse y recibirse empaquetados, sino que también pueden funcionar como buffer MPI para envío y recepción directa sin empaquetamiento (incluso con primitivas de intercambio como `MP_Sendrecv_replace`) y para operaciones colectivas (`MPI_{Scatter,Gather,Reduce,Alltoallv}`…). En concreto, se puede reducir con todos los operadores (`MPI_{MAX,MIN,SUM,PROD,REPLACE,LAND,LOR,LXOR}`).

Utilidades

Los usuarios finales son quienes mejor conocen su aplicación, y por tanto los más apropiados para diseñar el esquema de paso de mensajes entre las distintas instancias de su SCE ejecutándose en paralelo en un cluster. Sin embargo, esto no es motivo para dejarles solos frente a la tarea de paralelizar su aplicación. Muchos usuarios pueden beneficiarse de ejemplos sencillos de esquemas paralelos, tanto para suavizar la curva de aprendizaje inicial como para ahorrar tiempo de desarrollo mediante la personalización de código paralelo frecuentemente usado.

De esta reflexión surgieron los ficheros de arranque y las utilidades de protocolo e instrumentación incluidas en nuestras herramientas. De nuevo, aunque aquí nos concentramos en la última MPITB para Octave, la misma discusión es aplicable a nuestras otras herramientas paralelas.

Ficheros de arranque

En las aplicaciones más sencillas, múltiples instancias de Octave se arrancan desde la línea de comandos mediante `mpirun` y ellas deben invocar `MPI_Init` y `MPI_Finalize`, sin ningún tipo de soporte automatizado. Esta alternativa es la más parecida a la habitual aplicación paralela en lenguaje C o Fortran con MPI. El *benchmark* de demostración NPB-EP (NAS Parallel Benchmarks – Embarrassingly Parallel) incluido en MPITB y presentado en [23] y la aplicación de estudio Mandelbrot presentada en este trabajo, funcionan de esta forma.

Por otro lado, cuando se inician instancias esclavas interactivamente desde un proce-

so Octave maestro mediante el comando `MPI_Comm_spawn`, conviene reunir los comunicadores del maestro y los esclavos para facilitar el direccionamiento mediante rangos en un único comunicador. Los ejemplos del tutorial MPITB funcionan de esta forma. Para lograr este propósito, conviene modificar el fichero de arranque de Octave `.octaverc` para que no solamente añada el path de la herramienta, sino que también detecte si se trata de una instancia esclava y en dicho caso invoque `MPI_Init` y `MPI_Intercomm_merge`, ahorrando al usuario interactivo teclearlo en todas las instancias esclavas. En la instancia maestra sí se debe teclear `MPI_Intercomm_merge` tras `MPI_Comm_spawn`, para completar la colectiva iniciada por los esclavos.

Esta costumbre basada en la experiencia interactiva de la herramienta terminó convirtiéndose en una completa colección de ficheros de arranque y correspondientes protocolos de comunicación, cada uno personalizado a una situación particular, como se explica con mayor detalle en [23]. Además de los ficheros de arranque ya mencionados "*ajuste de path*" y "*reunión de comunicadores*", están disponibles otros esquemas como "*broadcast*" (usado por las demos *Pi, Ping-pong* y *Spawn*) y "*número de comandos*" (usado por la demo *Wavelets*). Los usuarios finales pueden también redactar sus propios ficheros de arranque y utilidades de protocolo en lenguaje Octave.

Utilidades de protocolo

En aplicaciones más complejas, los usuarios pueden desear enviar a los SCE esclavos un número fijo o indeterminado de comandos, posiblemente junto con las variables sobre las cuales operar. O tal vez el mismo comando es repetidamente ejecutado sobre datos diferentes. El esquema de arranque denominado "*número de comandos*" y los scripts de apoyo al proceso maestro proporcionados por MPITB bastan para afrontar todas estas situaciones de una forma sistemática.

El protocolo *NumCmds* funciona como se indica gráficamente en la Fig. 2. Los scripts de apoyo mencionados se muestran en la instancia SCE izquierda.

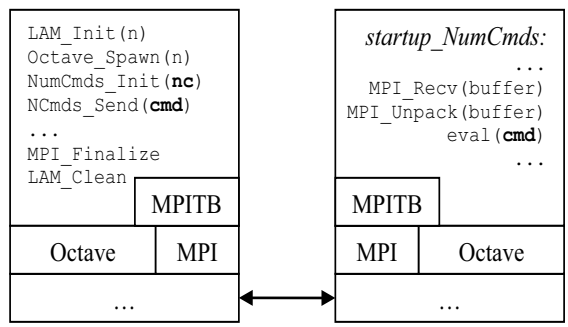

Fig. 2 Resumen simplificado del protocolo *NumCmds*, usando LAM/MPI en este caso. *LAM_Init* arranca *n* daemons LAM. *Octave_Spawn* inicia *n* instancias de Octave en esos nodos. Estas instancias terminan ejecutando un bucle *recv-unpack-eval* programado en el script *startup_NumCmds*. En este caso el bucle hará *nc* iteraciones según lo prescrito en el maestro mediante *NumCmds_Init*. En cada iteración el maestro Octave envía mediante *NumCmds_Send* un comando diferente (o vacío "" para repetir el anterior) y opcionalmente datos, a cada esclavo

El usuario se conectaría al nodo principal del cluster, arrancaría una sesión SCE maestra, y definiría la lista de nodos que desea usar como esclavos (en un *cell-array* de *strings*). Los *n* primeros son usados por las utilidades *LAM_Init* y *Octave_Spawn* para construir la sesión Octave paralela. Al

ejecutar el fichero de arranque `.octaverc` las instancias detectan que son esclavas y saltan al fichero de arranque *NumCmds*, que espera recibir del maestro un número inicial de comandos *nc*, significando 0 un número indeterminado de ellos (se seguiría iterando hasta que los esclavos recibieran el comando "*quit*"). Los esclavos entran entonces en un bucle *recv-unpack-eval*, con lo cual se ejecuta en cada iteración el comando enviado como *string* por parte del maestro.

Utilidades de instrumentación

Prácticamente la totalidad de usuarios de estas herramientas necesitarán conocer cómo se consume el tiempo en su aplicación SCE secuencial, para escoger las partes que más tiempo consuman como objetivo a paralelizar. A este efecto, MPITB proporciona una serie de rutinas de fácil uso (`time_stamp`, `look_stamp`, etc.) para generar, manipular y mostrar gráficamente los datos de instrumentación. Mostrar estos datos en un formato directamente comprensible puede ser de enorme ayuda para paralelizar correctamente una aplicación.

La Fig. 3 muestra el aspecto de la información de instrumentación para una de las demos Wavelet.

ESTUDIO DE CASO: CONJUNTO DE MANDELBROT

La importancia de los Problemas Juguete

La estrategia de PVMTB y MPITB para atraer a usuarios es proporcionar ganancia en velocidad lineal para aplicaciones con escalabilidad lineal. Muchas aplicaciones normalmente utilizadas por usuarios SCE presentan buena escalabilidad, y bastantes

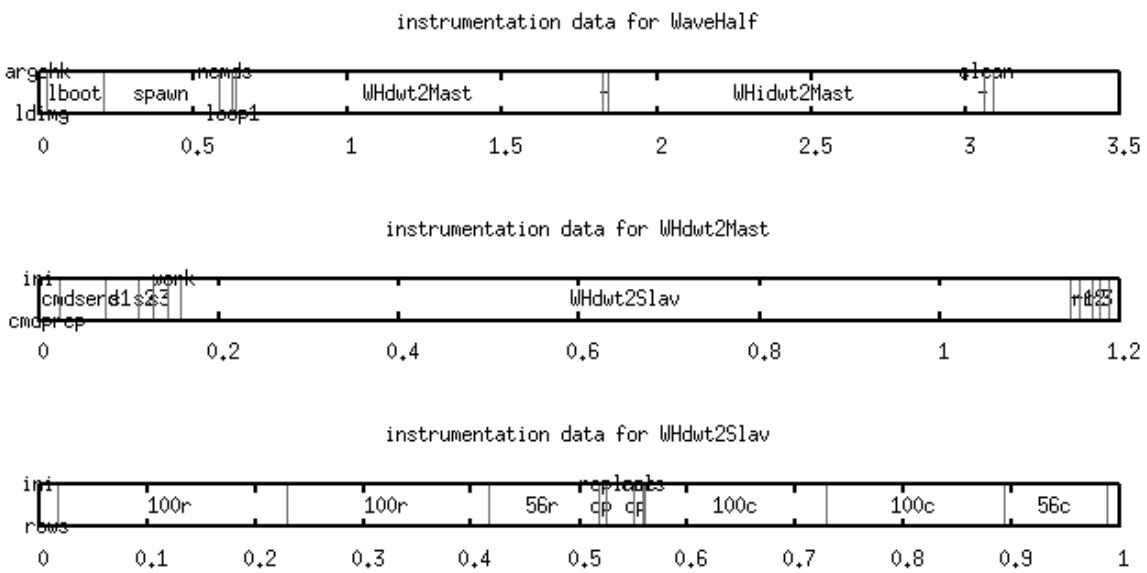

Fig. 3 Información de instrumentación para la demo *WaveHalf*. Si el usuario utiliza etiquetas *time_stamp* con prefijo común, *look_stamp* puede agruparlas, lo cual es muy conveniente para una visualización jerárquica. En este ejemplo, el usuario ha expandido la subrutina *WHdwt2Mast* (0.6s-1.8s aprox. de los 3.5s que tarda *WaveHalf*), y entonces *WHdwt2Slav* (0.15s-1.15s aprox. de los 1.2s de *WHdwt2Mast*).

de ellas escalabilidad casi lineal, esto es, que con dos procesadores se ejecute el doble de rápido, con tres casi el triple, y así sucesivamente.

La escalabilidad es una propiedad del algoritmo implementado en la aplicación paralela, no de la herramienta en sí, y es abundante la literatura (p. ej. [122]) que estudia abstractamente algoritmos paralelos por paso de mensajes sin indicar qué herramienta concreta se ha de usar. En dichos estudios las latencias, anchos de banda y otras características particulares de la herramienta se dejan indicados como parámetros, y corresponde a la herramienta cumplir o no las expectativas (latencia suficientemente pequeña, ancho de banda suficientemente grande) requeridas para que el algoritmo escale bien.

Por eso son preferibles problemas con carga computacional reducida (problemas juguete, *toy problems*) para evaluar las prestaciones de herramientas distintas. De esta forma, menores diferencias en prestaciones tienen alguna posibilidad de alterar el tiempo de ejecución paralelo. No solo se trata de que el tiempo consumido por la transmisión de datos debe ser mucho menor que el tiempo de computación de la aplicación (si es que se espera obtener ganancia en velocidad), sino que nos interesa evaluar la *diferencia* entre los costes de transmisión de las distintas herramientas; esas diferencias deberían ser una pequeña fracción del tiempo total de transmisión, y éste debería ser a su vez una pequeña fracción del tiempo de ejecución paralelo. Si esas *diferencias* han de manifestarse, será más fácil conseguirlo con problemas de juguete.

La Fig. 5 en la referencia [40] de August y Kanade es un excelente ejemplo de esta afirmación, ya que solo con tamaños tan pequeños que no eran realistas se consiguió que la ganancia en velocidad de nuestra herramienta MPITB se separara de la linealidad ideal.

Si se obtienen ganancias aceptables para el problema de tamaño juguete, sólo cabe esperar mejores resultados para problemas de tamaño real, donde la proporción entre costes de comunicación vs. computación usualmente disminuye o se mantiene constante, por difuminarse el efecto de las latencias. Si la proporción aumentara, no tendría sentido desear manejar el tamaño entero sino procesarlo por bloques, a menos que la aplicación permitiera ocultar el coste de la comunicación solapándola con la computación, en cuyo caso la diferencia de prestaciones entre herramientas quedaría como asunto de menor importancia. De nuevo, esta posibilidad de solapar sería una propiedad de la aplicación, no de la herramienta.

El Conjunto de Mandelbrot

Aunque el estudio matemático del conjunto de Mandelbrot es un tema complejo e importante en Dinámica Holomorfa (Complex Dynamics) y otros campos de estudio, generarle una aproximación gráfica por computador (Fig. 4) es un problema juguete, y además uno frecuentemente encontrado en la literatura ([122], [58], [50]).

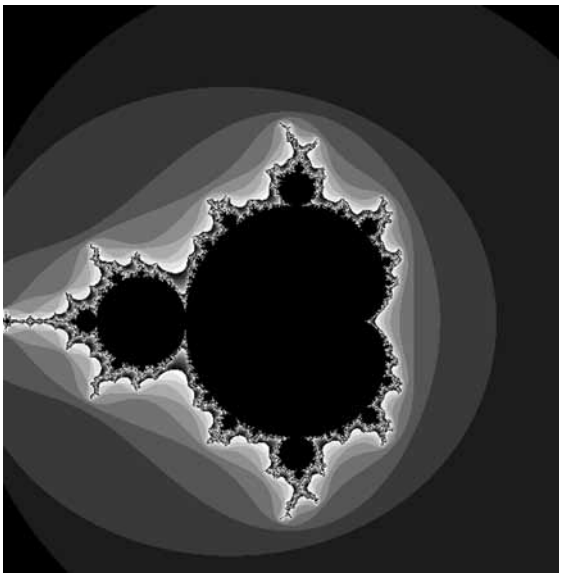

Fig. 4 Conjunto de Mandelbrot: aproximación gráfica entre los límites [-1.78-1.42i, 1.42+1.78i]. La circunferencia de radio 2 queda expuesta en tres de las esquinas. Se asignan colores más claros conforme más iteraciones requiere el pixel, reutilizando la paleta cada 10 colores. Si se alcanza el límite de iteraciones el pixel se deja sin colorear (negro). Para esta figura se usaron 64 iteraciones y 600x600 pixels. Una zona similar escogieron los autores de [58]. Más adelante se indica una propiedad de esta zona respecto al reparto de carga

Se puede encontrar una buena explicación *online* en Wikipedia [123]. Para cada punto c en el plano complejo se calcula la secuencia $z_0=0$, $z_{k+1}=z_k^2+c$. El *Conjunto de Mandelbrot* es el lugar geométrico de los puntos donde la secuencia no diverge.

Por motivos prácticos, se selecciona previamente la porción del plano complejo a visualizar (estando las características más interesantes en el rectángulo [(-2-i), (0.5+i)], o el disco cerrado de radio 2), el cálculo de la secuencia se restringe a las coordenadas correspondientes a los pixeles de la imagen resultado, y se fija un máximo de iteraciones por pixel permitidas.

La condición de divergencia se da por satisfecha cuando el módulo |z| alcance 2 (esto es, fuera del disco). Es costumbre colorear los pixeles según el número de iteraciones realizadas, lo cual revela también las *Curvas de Mandelbrot*, que convergen a la frontera del *Conjunto de Mandelbrot*.

Versión secuencial

El código de esta demo en MPITB está vectorizado, esto es, opera en *arrays* completos en lugar de iterar sobre los elementos, prefiriendo velocidad a ahorro de espacio. Una versión iterativa, más lenta, haría un bucle anidado sobre las coordenadas (fila y columna), almacenando únicamente el contador final de iteraciones requerido para colorear el pixel. La solución vectorizada, más rápida, requiere *arrays* para almacenar las coordenadas *c*, la secuencia *z*, los contadores de iteraciones *mandel*, y los índices que no divergen *mset*, como se muestra en la Fig. 5.

```
 [x,y]=meshgrid(r_min + scl_r*[0.5:1:x_siz],...
                i_min + scl_i*[0.5:1:y_siz]);
     c=      (x + i*y);          % plano complejo
     z= zeros(size(c));          % secuencia z
mandel= zeros(size(c));          % iteraciones
  mset=1:prod(size(c));          % puntos no diverg
for counter=1:max_it
    mset(abs(z(mset))>=MOD)=[]; % eliminar diverg
    z(mset)=z(mset).^2+c(mset); % iter z = z^2 + c
    mandel(mset)=counter;       % colorear
end
```

Fig. 5 Código Octave vectorizado para calcular el Conjunto de Mandelbrot. Las coordenadas de los centros de pixel en el plano complejo se almacenan en *c*. La secuencia *z* se itera simultáneamente en todos los puntos que aún no hayan divergido (instrucción central del bucle), anotándoseles que llegaron a esta iteración. Antes de iterar se eliminan de la lista de puntos aquellos que diverjan

Para paralelizar este tipo de aplicaciones se suelen usar técnicas de división de dominio, ya sean estáticas o dinámicas. La carga de trabajo se divide conceptualmente en secciones que pueden ser calculadas independientemente, y los resultados parciales son posteriormente incorporados en la solución final. Una partición estática se realiza antes de iniciar los cálculos, según el número y prestaciones de los procesadores disponibles, para que no haya procesadores ociosos y todos los resultados se obtengan a su tiempo.

Una partición dinámica se realiza durante la propia ejecución de la aplicación paralela, siendo la "bolsa de tareas" (*task-bag*, o *work-pool*) la técnica más común: la carga se divide en muchas tareas independientes pequeñas (muchas más que procesadores disponibles) y un proceso maestro reparte inicialmente una tarea a cada procesador; en cuanto hay disponible algún resultado, el maestro lo incorpora a la solución y vuelve a asignar otra tarea de la bolsa al procesador ocioso. Aunque esto implica más mensajes y mayor sobrecarga de comunicaciones, proporciona una cierta forma de equilibrio de carga automático: el reparto de carga no se realiza estáticamente a la hora de diseñar la aplicación, con fórmulas cerradas (N subtareas de carga apropiada para los N procesadores disponibles), sino dinámicamente conforme se ejecuta la aplicación. Esto es ventajoso cuando la carga asociada a una subtarea no es fácilmente predecible en función de sus dimensiones.

Versión paralela - división de dominio estática

Frecuentemente (sobre todo con procesadores de idénticas prestaciones ejecutando el mismo código paralelo) se estima que la carga crece linealmente con el tamaño del subproblema, y se asume que a igual tamaño, igual carga.

Eso no se cumple en la demo de Mandelbrot en general, y menos aún en los límites mostrados en la Fig. 4 en particular, ya que los pixeles del exterior divergen en muy pocas iteraciones (los externos al disco de radio 2 ni siquiera llegan a cumplir una iteración), los pixeles centrales no divergen y consumen las 64 iteraciones, y la carga no puede predecirse según el tamaño del subdominio, y puede resultar muy desigualmente repartida.

Con los límites de la Fig. 4, el típico reparto de dominio en franjas verticales (de igual anchura, tantas como procesadores disponibles) produciría columnas casi sin carga en el tercio derecho del dominio, quedando inmediatamente ociosos los procesadores que les fueron asignados.

Es obligado mencionar como ventaja de esta técnica que el código paralelo es inmediato de programar, mostrándose la parte relevante en la Fig. 6. Las ganancias que hemos obtenido (Fig. 7) corroboran nuestra reflexión anterior, discrepando hasta cierto punto con algún resultado en la literatura [58].

Se puede comprender que aquellos autores no desearan mostrar las eficiencias para

```
 info      =MPI_Init;             % Iniciar MPI
   WORLD=MPI_COMM_WORLD;
[info rank]=MPI_Comm_rank(WORLD); % reparto
[info nprc]=MPI_Comm_size(WORLD); % estático franja
   lins =   fix(x_siz/nprc);      % líneas por rank
   rlns =   rem(x_siz,nprc);      % repartir resto
 nlns=lins +       (rank<rlns);  % a primeros rank
 strl=lins*rank+min(rank,rlns);  % línea de inicio
if rank!=MAST                    % ** esclavos **
   strip           =CalcStrip(strl,nlns); %calculo
   info = MPI_Send(strip,MAST,RTAG,WORLD);%y envío
else                             % ** maestro **
   mandel=zeros(y_siz,x_siz);    % sitio solución
   mandel(:,1:nlns)=CalcStrip(strl,nlns); %calculo
                                  % y recibo resto
   strip=zeros(y_siz, nlns);     % sitio recibir
   for slv=1:nprc-1              % cualquier orden
    [info stat]=MPI_Recv(strip,ANYS,RTAG,WORLD);
    task=stat.src;               % cuál esclavo?
    nlns=lins+        (task<rlns); % localizar
    strl=lins*task+min(task,rlns); % y adjuntar
    mandel(:,strl+[1:nlns])=strip(:,1:nlns);
   end                           % a la solución
end
```

Fig. 6 Código Octave paralelo para calcular el Conjunto de Mandelbrot por reparto de dominio estático. Conociendo el número de líneas a calcular (*x_siz*), el de procesadores disponibles (*nprc*) y su propio rango (*rank*), cada proceso puede determinar su línea de inicio y número de líneas a calcular, que son parámetros para la función *CalcStrip*. Los procesos esclavos devuelven el resultado al maestro, el cual incorpora las franjas verticales en la imagen final.

3 y 5 procesadores, y se puede excusar que no se muestren las ganancias, siendo eficiencia=ganancia/nº procesadores, pero sin conocer los tiempos de ejecución no se puede valorar apropiadamente el significado de que las eficiencias de una herramienta sean a veces superiores y a veces inferiores a los de otra.

En la Fig. 14 y en los Comentarios se trata con más detalle cómo un reparto entre un nº impar de procesadores causa que la franja central tenga un coste muy superior al resto.

El Conjunto de Mandelbrot es (al menos en estos límites) una de esas aplicaciones para las cuales conviene usar un esquema de paso de mensajes más complejo, aunque en principio pueda parecer más ineficiente por requerir un patrón de comunicaciones más intenso.

Versión paralela – bolsa de tareas

El paradigma de bolsa de tareas es inmediatamente adaptable a este problema juguete, dividiendo la carga total en columnas de 1 pixel de anchura, en lugar de franjas verticales. La sobrecarga causada por el gran número de pequeños mensajes no es tan grave como resultó ser (en los límites escogidos) el desequilibrio de carga con 3 franjas (u otro número impar, en general) en la Fig. 7.

Además, la carga de cada columna crece y decrece gradualmente conforme se barre el eje real (ver Fig. 4), así que al acabarse la bolsa todos los procesadores han recibido una columna con muy poca carga, con lo cual el tiempo que pasan ociosos esperando al último es despreciable.

Los resultados se muestran en la Fig. 8. Aunque la técnica es conceptualmente inmediata de aplicar, la lógica e indexado requeridos para implementarla son bastante elaborados, especialmente si se desea que el proceso que reparte la bolsa trabaje también, como hemos elegido en esta variante.

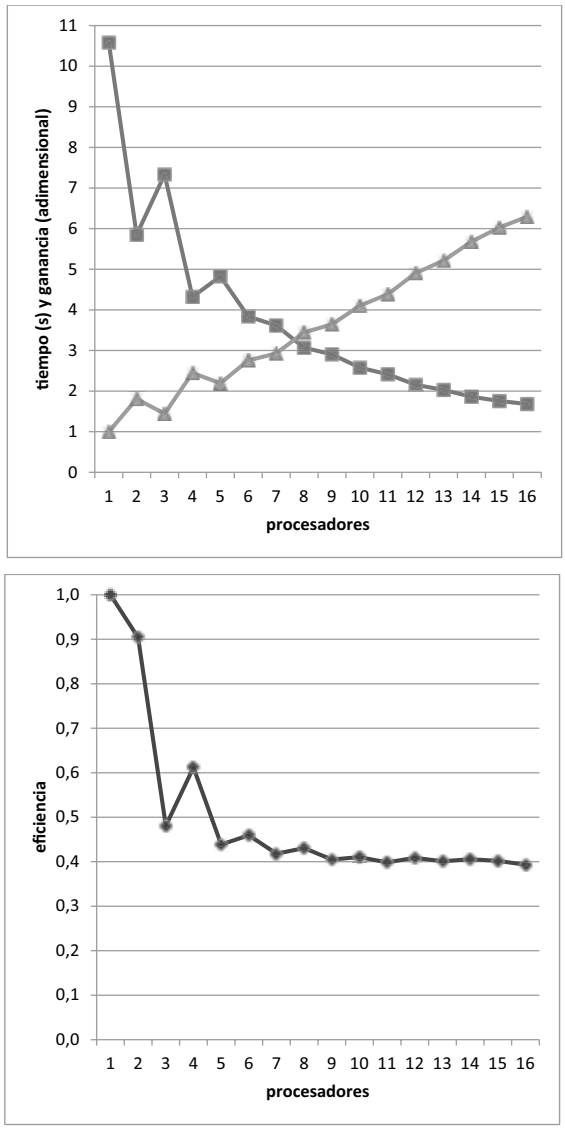

Fig. 7 Tiempo de ejecución, ganancia en velocidad y eficiencia para la versión paralela por división de dominio habitual (franjas verticales).

La otra variante de bolsa de tareas, en donde el nodo maestro ejecuta dos procesos, uno dedicado al reparto y otro trabajando como un esclavo normal, es más fácil de programar, y el S.O. suele realizar un buen trabajo conmutando entre ambos procesos, como se puede comprobar en los resultados de la Fig. 10, correspondientes al código que (ahora sí) se ha podido resumir en la Fig. 9. Es de destacar que la eficiencia no cae ahora inmediatamente por debajo de 0.9 en cuanto se usan dos procesadores, como sucedía con la primera variante (Fig. 8).

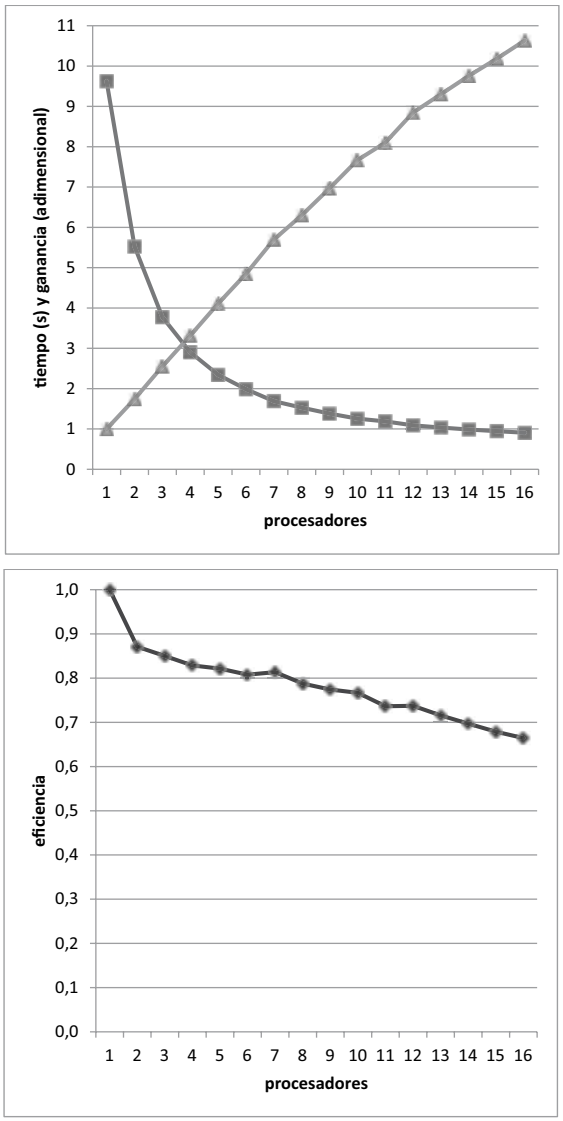

Fig. 8. Tiempo de ejecución, ganancia en velocidad y eficiencia para la versión paralela por bolsa de trabajo, 1ª variante con proceso de reparto que también trabaja

```
if rank==MAST            % reparto - no trabajo
  mandel=zeros(y_siz, x_siz);  % sitio para imagen
    line=zeros(y_siz,     1);  % sitio para colum
  for slv=1:nprc-1        % índice col a esclavos
    info=MPI_Send(slv,slv,DTAG,WORLD);
  end                     % 1ª tarea, col==slv
  TaskCast = 1:nprc-1;% anotación inicial tareas
     flght =   nprc-1; % nº de tareas en vuelo
       col =   nprc-1; % última col enviada
  while flght>0      % recibir hasta bolsa vacía
   [info stat]=MPI_Recv(line,ANYS,RTAG,WORLD);
    slv=stat.src;              % cuál esclavo?
    slvcol=TaskCast(slv);   % qué linea se envió?
    mandel(:,slvcol)=line;     % guardar result
    if col<x_siz               % bolsa vacía? no
       col  =  col+1;     % en vuelo los mismos
       info =  MPI_Send(col, slv, DTAG,WORLD);
       TaskCast (slv) = col;   % anotar sig tarea
    else
       flght=flght-1;   % bolsa vacía, en vuelo -1
       info =  MPI_Send(col, slv, TTAG,WORLD);
    end                % mensaje de terminación
  end                          % fin bucle TaskBag
 else      % rank!=MAST        % código esclavo
  col = 0;            % buffer donde recibir nº
 [info stat]  =MPI_Recv(col ,MAST,ANYT,WORLD);
  while stat.tag==DTAG  % data tag/terminate tag
        line =CalcLine(col);
    info      =MPI_Send(line,MAST,RTAG,WORLD);
   [info stat]=MPI_Recv(col ,MAST,ANYT,WORLD);
  end                % devolv.result, recv tarea
end     % rank==MAST
```

Fig. 9 Código Octave paralelo para calcular el Conjunto de Mandelbrot por bolsa de tareas. En esta segunda variante hay un proceso (rango 0) dedicado al reparto, recogiendo resultados y enviando nuevas tareas a los procesos que van quedando ociosos, o enviando un mensaje de terminación si la bolsa está vacía ya. La asignación de tareas se anota (en *TaskCast*) para facilitar la incorporación de resultados cuando se reciban. Los procesos esclavo calculan líneas (*CalcLine*) y devuelven resultados hasta que se reciba un mensaje de terminación. El nodo maestro ejecuta el proceso de reparto y un proceso esclavo.

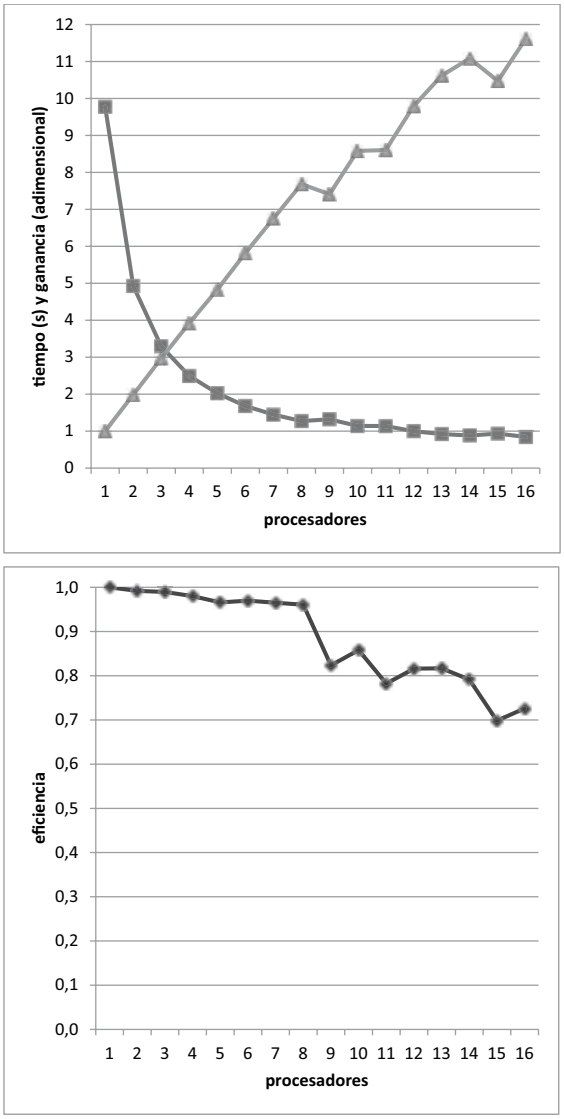

Fig. 10 Tiempo de ejecución, ganancia en velocidad y eficiencia para la versión paralela por bolsa de trabajo, 2ª variante con proceso separado para reparto de tareas

Versión paralela - conocimiento del usuario

Ambas técnicas, franjas y bolsa de tareas, son adaptables a muchas aplicaciones SCE, y pueden ofrecerse a los usuarios finales en forma de rutinas de alto nivel o como construcciones paralelas ("esqueletos"). Alternativamente los usuarios pueden reutilizar el código de demostración proporcionado,

adaptándolo a los datos de su aplicación. Pero el punto fuerte de los usuarios está en el conocimiento que poseen de su campo de estudio, y la paralelización de sus aplicaciones puede beneficiarse enormemente de dicho conocimiento, como se va a comprobar con este ejemplo de juguete.

En los apartados anteriores se vio que la división por franjas (estática) causa menos comunicaciones, pero sufre desequilibrio de carga. La división por columnas (dinámica) funciona mejor gracias a la variación gradual de la carga a lo largo del eje real del plano complejo. Ese conocimiento del usuario final sobre el problema a paralelizar puede explotarse aún más efectivamente, reflexionando que una división por columnas repartidas cíclicamente entre los N procesadores (de manera que el rango i procesa las columnas kN+i) también evita el desequilibrio, impidiendo que las columnas de mayor carga (franjas centrales) recaigan sobre unos procesadores mientras que otros (franjas laterales) quedarán ociosos en cuanto terminen la escasa carga asignada.

Este reparto se suele denominar "a saltos" (*strided*), y es también una división de dominio estática, calculada antes de empezar a trabajar los esclavos. Para el caso de procesadores de distintas prestaciones (pero conocidas de antemano), la fórmula puede alterarse de manera que un procesador que puede ir *x* veces más rápido reciba *x* veces más columnas.

En nuestro caso, todos los procesadores son idénticos, el código para realizar el reparto a saltos (Fig. 11) es incluso más sencillo que para el reparto por franjas, y los resultados de ganancia mostrados en la Fig. 12 pueden considerarse prácticamente lineales.

```
info      =MPI_Init;            % Iniciar MPI
      WORLD=MPI_COMM_WORLD;
[info rank]=MPI_Comm_rank(WORLD); % reparto
[info nprc]=MPI_Comm_size(WORLD);% estático saltos
   lins =   fix(x_siz/nprc);  % líneas por rank
   rlns =   rem(x_siz,nprc); % repartir resto
   nlns=lins + (rank<rlns);   % a primeros rank
if rank!=MAST                  % ** esclavos **
   strip= CalcStrip(nlns,rank,nprc);
   info = MPI_Send(strip,MAST,RTAG,WORLD);
 else                          % ** maestro **
   mandel=zeros(y_siz,x_siz);   % sitio solución
   mandel(:,1:nprc:x_siz)=...   % cálculo local
         CalcStrip(nlns,rank,nprc); % ver stride
                                % resto esclavos
   strip=zeros(y_siz, nlns);    % sitio recibir
   for slv=1:nprc-1            % cualquier orden
    [info stat]=MPI_Recv(strip,ANYS,RTAG,WORLD);
      task=stat.src;            % cuál esclavo?
      nlns=lins+        (task<rlns);% ver stride
    mandel(:,task+1:nprc:x_siz)=strip(:,1:nlns);
   end                          % adjuntar a sol.
end
```

Fig. 11 Código Octave paralelo con reparto de dominio estático "a saltos". No es necesario calcular la línea de inicio (*strl*), *CalcStrip* necesita conocer *rank* y *nprc*, y los resultados se incorporan con indexación a saltos (*inic:paso:fin*). Al calcular columnas esparcidas por el eje real, la carga está más equilibrada, incluso para los límites mostrados en la Fig. 4, frecuentemente usados en la literatura

COMENTARIOS

Para las mediciones se utilizó un cluster de 8 nodos biprocesador, como revelan las gráficas de escalabilidad y de eficiencia al pasar de 8 a 9 procesadores. Pueden ejecutarse hasta 8 procesos cada uno en su nodo, teniendo que compartir nodo (y por tanto acceso a memoria) el 9º con algún

EDUCOSTA
EDITORIAL UNIVERSITARIA DE LA COSTA

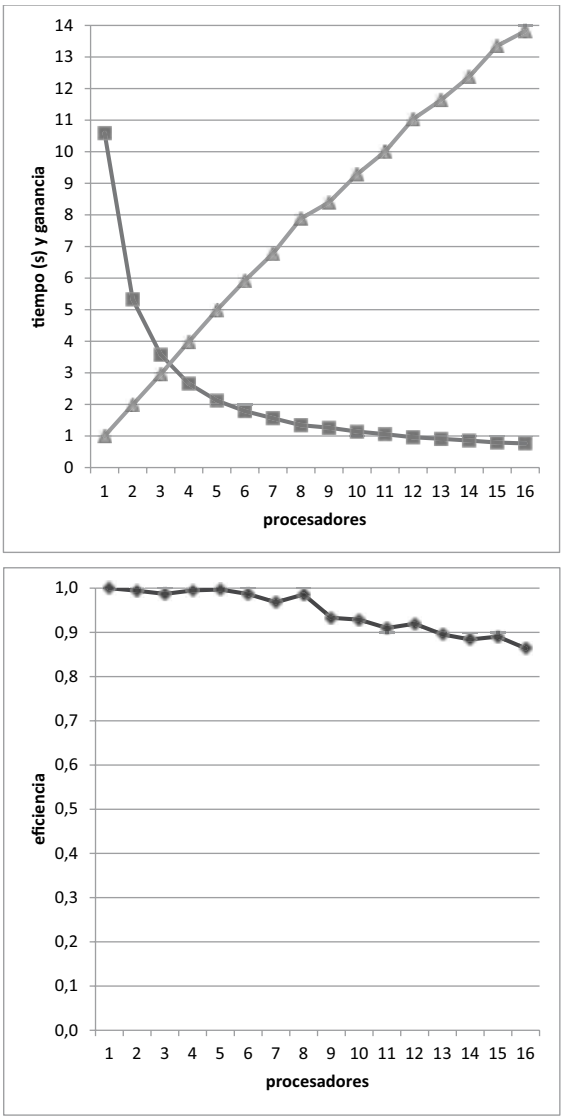

Fig. 12 Tiempo de ejecución, ganancia en velocidad y eficiencia para la versión paralela con reparto estático a saltos.

otro proceso. Esa ligera pérdida de prestaciones es comprensible, habiéndose usado código vectorizado para iterar la secuencia $z_{k+1}=z_k^2+c$, esto es, prefiriendo velocidad a ahorro de espacio.

Comparando los tiempos de ejecución para un procesador en las cuatro versiones, puede extrañar que tarden menos los algoritmos de bolsa de trabajo, cuando éstos requieren paso de mensajes (2ª variante) o al menos operaciones adicionales. Aunque no resulte intuitivamente obvio, procesar el dominio columna a columna, o en bloques en general, puede ser más eficiente que realizar la operación vectorizada de una vez sobre el dominio completo. El tamaño de bloque óptimo depende obviamente de la aplicación y datos concretos. En nuestro caso, queda claro que el dominio es lo suficientemente grande como para usar procesamiento por bloques, ya que incluso con bloques de una columna mejora el tiempo de ejecución. El tamaño de bloque óptimo podría ser una columna, varias, o el dominio entero, según las velocidades relativas del intérprete Octave y del sistema de memoria (para ese tamaño de bloque).

El código para procesamiento por bloques es ligeramente más complicado, y la granularidad óptima depende del computador usado, siendo esas las principales razones para haber preferido usar el código de la Fig. 5. Por completar el estudio, al código secuencial se le añadió procesamiento por bloques, siendo el nº de bloques un parámetro, y se ejecutó en tres computadores distintos: un nodo de nuestro cluster Athlon, otro nodo Xeon, y un computador de sobremesa de prestaciones inferiores. El tamaño de bloque óptimo varió entre 8 y 4 columnas (Fig. 13), requiriendo entre 75 y 150 iteraciones (bloques) para procesar las 600 columnas del dominio completo. En el cluster Xeon no se hubiera notado esa mejora anti-intuitiva en los tiempos de los algoritmos de bolsa de trabajo en un procesador, al

ser más rápido procesar el dominio completo que hacerlo columna a columna.

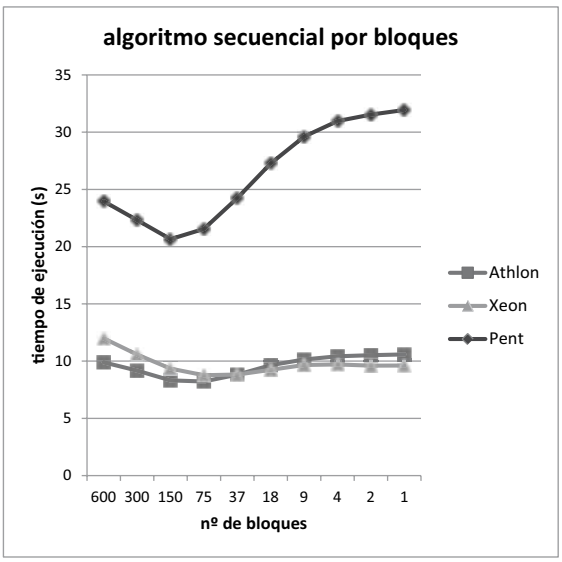

Fig. 13 Tiempo de ejecución en un solo computador, usando procesamiento por bloques. El dominio se divide en el nº de bloques indicado en la abscisa, desde 600 bloques de una columna a un solo bloque (el dominio completo). El tamaño óptimo depende de las velocidades relativas del intérprete Octave y de las prestaciones del sistema de memoria procesando vectorialmente bloques de dicho tamaño. Diferentes computadores muestran distintos óptimos. A 8B/double, un bloque de 8 columnas ocupa 38400B, sin relación inmediata con los (mayores) tamaños de caché de los procesadores.

Estrictamente hablando, la ganancia en velocidad de un programa paralelo es la razón entre el tiempo del mejor programa secuencial y el del programa paralelo. En nuestras gráficas hemos comparado los tiempos paralelos en un solo procesador con el de varios procesadores, como también es habitual en la literatura. De seguir la definición literalmente, debería demostrarse que el algoritmo usado es el óptimo, al menos en la plataforma usada para el experimento.

Estudiando el código de la Fig. 5, también puede extrañar que se coloreen en cada iteración todos los puntos que aún no divergen. En otra optimización de menor impacto, inicializamos la imagen `mandel` al nº máximo de iteraciones, coloreando en cada iteración solo los elementos divergentes. Aunque esto ahorra la mayoría de las asignaciones de color, el ahorro de tiempo es aún menor que el previamente discutido con procesamiento por bloques, así que de nuevo preferimos la más sencilla y elegante versión de la Fig. 5.

El tipo base por defecto para los *arrays* en estos SCEs es el *double* de 8B, para el cual están definidas prácticamente la totalidad de operadores y rutinas, siendo por tanto el tipo más cómodo de usar. No se ha considerado el uso de tipos enteros más pequeños para la imagen `mandel`. Se podría argumentar que incluso *unsigned byte* serviría para almacenar hasta un máximo de 256 iteraciones (más que las 64 aquí usadas), y eso sería ventajoso a la hora de recolectar imágenes parciales de los computadores esclavos (hasta 8 veces más rápido, según el tamaño transmitido). Sin embargo, operar con estos tipos de datos puede ser ineficiente, complicado, o estar sencillamente prohibido (según el entorno SCE usado y la versión concreta), y pronto nos veríamos recurriendo a costosos *typecasts* para obtener la Fig. 4, que necesita una operación `mod()`.

Unas sencillas trazas XMPI como las mostradas en la Fig. 14 para 8 y 9 procesos de la versión por franjas pueden ilustrar las totalmente dispares escalas de tiempo para comunicación y computación en este estu-

dio de caso. Esa disparidad es conveniente, volviendo a la reflexión de la Sección I.A, si se esperan buenas ganancias de la paralelización. También desalienta respecto al cambio de tipo de datos comentado, ya que el ahorro sería insignificante en comparación con el desequilibrio de carga. No conviene complicar el código para reducir a 1/8 un tiempo de comunicación que de por sí ya es reducido en comparación con el tiempo de computación, mientras que éste está claramente desequilibrado. Además, todos los tiempos de comunicación quedan solapados con computación salvo el último, ya que el proceso recolector no tiene ninguna otra tarea que realizar salvo recibir esos mensajes.

El tiempo paralelo está principalmente determinado por la carga de trabajo de la franja central, lo cual convierte a este (desequilibrado) ejemplo de juguete en un pobre *benchmark* para herramientas de programación paralela, volviendo a la reflexión de la Sección I.A. La variante a saltos sería mucho más apropiada a tal fin, ya que corrige el desequilibrio y deja de ocultar el tiempo de transmisión solapándolo con computación. La variante de bolsa de trabajo sería mejor aún, ya que ejercita mucho más el paso de mensajes, aunque no sea tan eficiente.

Este desequilibrio causa la bajada al 40% de eficiencia en la versión por franjas en la Fig. 7. Las iteraciones por columna en el área central son unas 2.5 veces superiores a la media, de ahí el límite de 0.4, alcanzado cuando la franja central cae enteramente en el área 2.5x (7…9 procesadores) y mantenido desde entonces.

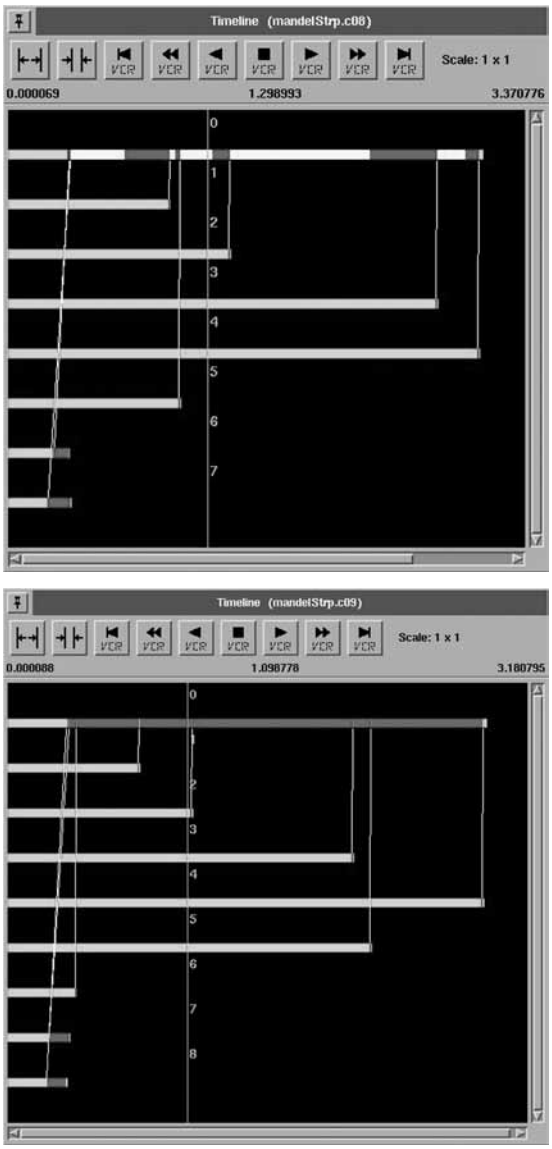

Fig. 14 Trazas XMPI para la ejecución con 8 (arriba) y 9 (abajo) procesadores de la versión por franjas. El tiempo de transmisión es despreciable en comparación con la carga de trabajo, pese a estar usándose tipos de datos *double*. Casi todo el coste de transmisión está solapado con el coste computacional de la franja central, que determina mayormente el tiempo de ejecución total. Si se usara esta variante para comparar distintas herramientas de paso de mensajes, la única transmisión que no quedaría oculta sería la última recolección de datos de la franja central, que es una fracción (1/8, 1/9) del total de comunicaciones, que a su vez es una pequeña fracción del tiempo paralelo total. Las diferencias se pueden disimular aún más incrementando la carga computacional del ejemplo, ya sea aumentando el tamaño de la imagen o el nº de iteraciones de la secuencia *z*

Se podría prefijar un límite distinto cambiando el parámetro `max_it` de la Fig. 5 (máximo nº de iteraciones para decidir divergencia) y alterando así la densidad de trabajo en las franjas centrales, como se muestra en la Fig. 15.

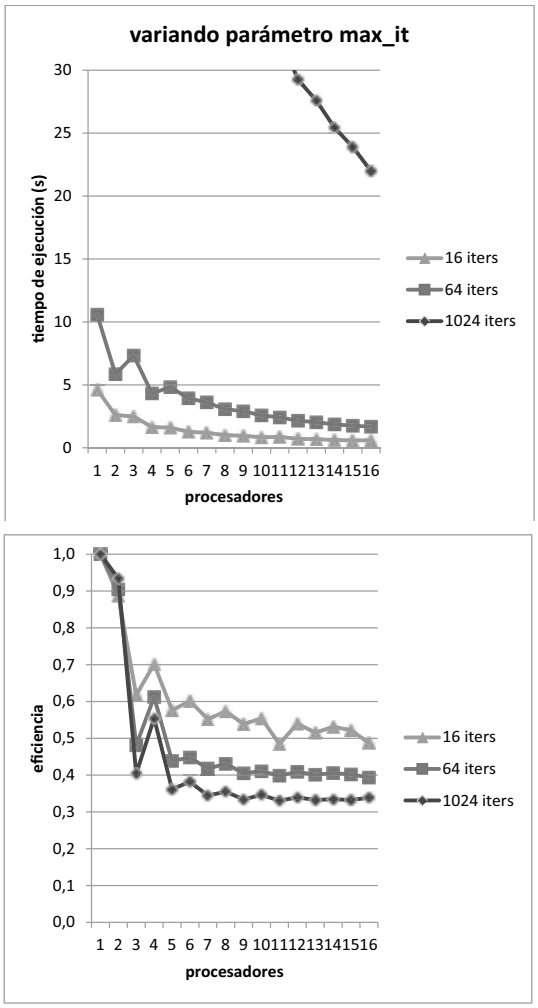

Fig. 15 Tiempos de ejecución y eficiencias de la variante por franjas para 3 valores del parámetro `max_it` = 16, 64, 1024. Al incrementarlo, aumenta la carga dentro del conjunto de Mandelbrot, pero la carga media no aumenta tanto porque el conjunto es apenas la cuarta parte del dominio escogido, lo cual prefija un límite inferior de eficiencia aún menor que el 40% correspondiente a `max_it` = 64. Los tiempos y eficiencias para 64 ya se habían dado en la Fig. 7. Para 1024, los tiempos llegan hasta 120s, fuera del gráfico, y el trazado tiene forma similar, aunque aún más pronunciada

Se espera haber clarificado con esta discusión nuestro comentario de la Sección I.D, respecto a que sin conocer los tiempos de ejecución no se puede valorar apropiadamente el hecho de que las eficiencias de una herramienta sean a veces superiores y a veces inferiores a los de otra.

CONCLUSIÓN

Con la popularización de los clusters de computadores, una potencia de cálculo previamente no disponible quedó al alcance de usuarios normales. Pero muchos usuarios programan sus aplicaciones de cálculo intensivo en Entornos de Cálculo Científico (Scientific Computing Environments, SCE). Las herramientas PVMTB y MPITB permiten a estos usuarios utilizar esta potencia de cálculo, al precio de adaptar sus aplicaciones secuenciales al paradigma de programación de paso de mensajes explícito.

En algunos casos esta adaptación es inmediata, y las herramientas proporcionan una serie de utilidades para facilitarla en casos más complicados. Para muchos ejemplos, como la *demo* Mandelbrot, es posible obtener un esquema de escalabilidad lineal. Incluso para aplicaciones inherentemente iterativas y secuenciales (como optimización MLE/BFGS), algunos usuarios finales han obtenido escalabilidades muy buenas.

El paso de mensajes ha sido considerado tradicionalmente como muy complejo y de demasiado bajo nivel para los usuarios SCE, pero las ganancias en prestaciones no pueden ser simplemente ignoradas. Ade-

más, alternativas de más alto nivel tienden a incluir un creciente número de primitivas de bajo nivel como solución alternativa, o como único medio de proporcionar la funcionalidad deseada.

Una serie de usuarios finales han utilizado con éxito nuestras herramientas para investigación, educación y desarrollo de software. A todos ellos les agradecemos su contribución a popularizar la computación en paralelo (e indirectamente, el paso de mensajes) en sus campos respectivos. También les agradecemos su realimentación, que ha mejorado enormemente las versiones originales de las herramientas y guiado el desarrollo de las utilidades de apoyo.

AGRADECIMIENTOS

Los autores agradecen al Departamento de Arquitectura y Tecnología de Computadores y a la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada por el uso de sus instalaciones cluster. También gracias a Jeff Squyres, del equipo de desarrollo de LAM/MPI (ahora en Cisco Systems) por su inestimable apoyo en el desarrollo de la primera MPITB, y a Brian W. Barret por mantener el software XMPI usado para obtener la Fig. 14. Gracias a los doctores D. Eduardo Miguel de la Hoz Correa y D. Emiro de la Hoz Franco por su interés en nuestro trabajo y por animarnos a publicarlo. Gracias también a nuestros colegas M.A. Piñar y T.E. Pérez por sus comentarios sobre la carga paralela para este estudio de caso, y a S. Dormido, S. Goasguen y M. Creel por su temprana adopción de las respectivas herramientas y sus amables consejos. Por último, pero no menos importante, gracias a J.W. Eaton y sus colaboradores por desarrollar y mantener Octave.

REFERENCIAS

[1] C. B. Moler, *Numerical Computing with MATLAB, Revised Reprint*. SIAM, 2004, 2008. Para otras referencias autorizadas ver también http://www.mathworks.com/support/books/

[2] Web de *The MathWorks,* Disponible en: http://www.mathworks.com/products/pfo/

[3] Web de *The MathWorks,* Disponible en: http://www.mathworks.com/products/matlab/

[4] J. W. Eaton, D. Bateman y S. Hauberg, *GNU Octave Manual*. Network Theory Ltd., 2008.

[5] J. W. Eaton, "GNU Octave: History and outlook for the future" in *Conference Proceedings of the 2005 AIChE Annual Meeting,* Cincinnati Ohio, November 1, 2005.

[6] J. W. Eaton and J. B. Rawlings, "Ten Years of Octave - Recent Developments and Plans for the Future" in *Proceedings of the 3rd International Workshop on Distributed Statistical Computing DSC-2003,* Vienna, Austria, 2003.

[7] J. W. Eaton, "Octave: Past, Present and Future" in *Proceedings of the 2nd International Workshop on Distributed Statistical Computing DSC-2001,* Vienna, Austria, 2001.

[8] Web de Octave, Disponible en: http://www.gnu.org/software/octave/.

[9] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderam, *PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing*. The MIT Press, 1994.

[10] Web PVM, Disponible en: http://www.csm.ornl.gov/pvm/.

[11] MPI Forum, "MPI: A Message-Passing Interface standard." *Int. J. Supercomput. Appl. High Perform. Comput.*, vol. 8, no. 3/4, pp. 159-416, 1994. Ver también los documentos del MPI Forum: MPI 2.2 standard (2009), MPI 3.0 Draft (2012), University of Tennessee, Knoxville. Disponible en: http://www.mpi-forum.org/

[12] W. Gropp, E. Lusk and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message Passing Interface*, 2nd Edition. The MIT Press, 1999.

[13] W. Gropp, E. Lusk and R. Thakur, *Using MPI-2: Advanced Features of the Message-Passing Interface*. The MIT Press, 1999.

[14] G. Burns, R. Daoud and J. Vaigl, "LAM: an open cluster environment for MPI" in *Proceedings of Supercomputing symposium*, 1994, pp. 379-386.

[15] J. Squyres and A. Lumsdaine, "A component architecture for LAM/MPI" in *Proceedings of the 10th European PVM/MPI Users' Group Meeting, Lect. Notes Comput. Sc.,* vol. 2840, pp. 379-387, 2003.

[16] Web LAM, Disponible en: http://www.lam-mpi.org/about/overview/.

[17] E. Gabriel *et al.*, Open-MPI team, "Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation" in *Proceedings, 11th European PVM/MPI Users' Group Meeting*, Budapest, Hungary, September 2004.

[18] Web Open-MPI, Disponible en: http://www.open-mpi.org/.

[19] J. Fernández, "PVMTB (Parallel Virtual Machine Toolbox)" in *III Congreso Usuarios MATLAB'99,* 17-19 Nov. 1999, UNED, Madrid, Spain. pp.523–532. Disponible en: http://atc.ugr.es/~javier/investigacion/papers/Users99.pdf

[20] J. Fernández, "Message passing under MATLAB" in *Proceedings of the Advanced Simulation Technologies Conference ASTC'01,* Seattle Washington, April 22-26, 2001, pp. 73-82.

[21] J. Fernández, A. Cañas, A. F. Díaz, J. González, J. Ortega and A. Prieto, "Performance of message-passing MATLAB toolboxes" in *Proc. of the VecPar 2002, Lect. Notes Comput. Sc.,* vol. 2565, pp. 228-241, 2003. URL de las Toolboxes http://www.ugr.es/~jfernand

[22] J. Fernández, M. Anguita, S. Mota, A. Cañas, E. Ortigosa and F.J. Rojas, "Parallel programming toolboxes for Octave (poster)" in *Proc. of the VecPar 2004,* Valencia, Spain, June 28-30 2004, pp. 797-806. Disponible en: http://www.ugr.es/~jfernand/investigacion/papers/VecPar04.pdf

[23] J. Fernández, M. Anguita, E. Ros and J.L. Bernier, "SCE toolboxes for the development of high-level parallel applications" in *Proc. of the 6th ICCS 2006, Part II,* Reading, United Kingdom, May 28-31, 2006. Lect. Notes Comput. Sc., vol. 3992, pp. 518-525.

[24] R. Pfarrhofer, P. Bachhiesl, M. Kelz, H. Stögner and A. Uhl, "MDICE - A MAT-

LAB toolbox for efficient cluster computing" in *Proc. of Parallel Computing* (Parco'03), Dresden, Germany, September 2-5, 2003, pp. 535-542.

[25] R. Pfarrhofer, M. Kelz, P. Bachhiesl, H. Stögner and A. Uhl, "Distributed optimization of fiber optic network layout using MATLAB" in *Proc. ICCSA 2004, Part III, Lect. Notes Comput. Sc.*, vol. 3045, 2004, pp. 538-547.

[26] D. Petcu, D. Dubu and M. Paprzycki, "Extending Maple to the Grid: Design and implementation" in *Proc. of the 3rd ISPDC / HeteroPar'04*, University College Cork, Ireland, July 5th - 7th 2004, pp. 209-216. DOI: 10.1109/ISPDC.2004.25.

[27] D. Petcu, M. Paprzycki and D. Dubu, "Design and implementation of a Grid extension for Maple" *Scientific Programming*, vol. 13, no. 2, 2005, pp. 137-149.

[28] D. Petcu, "Editorial: Challenges concerning symbolic computations on grids" *Scalable Computing: Practice and Experience*, vol. 6, no. 3, September 2005, pp. iii-iv.

[29] S. Goasguen, A. R. Butt, K. D. Colby and M. S. Lundstrorn, "Parallelization of the nanoscale device simulator nanoMOS-2.0 using a 100 nodes linux cluster," in *Proc. of the 2nd IEEE Conference on Nanotechnology,* pp. 409-412, 2002. DOI 10.1109/NANO.2002.1032277.

[30] S. Goasguen, R. Venugopal and M. S. Lundstrom, "Modeling transport in nanoscale silicon and molecular devices on parallel machines," in *Proc. of the 3rd IEEE Conference on Nanotechnology*, vol. 1, pp. 398-401, 2003. DOI 10.1109/NANO.2003.1231802.

[31] S. D. Canto, A. P. de Madrid and S. D. Bencomo, "Dynamic programming on clusters for solving control problems" in *Proc. of the 4th Asian Control Conference ASCC'02*, Suntec, Singapore, September 25-27, 2002.

[32] M. Parrilla, J. Aranda and S. D. Canto, "Parallel evolutionary computation: application of an EA to controller design," in *Proc. IWINAC 2005, Lect. Notes Comput. Sc.*, vol. 3562, pp. 153-162. DOI: 10.1007/11499305_16.

[33] S. D. Canto, A. P. de Madrid and S. D. Bencomo, "Parallel dynamic programming on clusters of workstations," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 9, pp. 785-798, 2005.

[34] M. Creel, "User-friendly parallel computations with Econometric examples," in *Proc. of the 11th Int. Conf. on Computing in Economics and Finance*, paper no. 445, 2005, Jun 23-25, Washington DC.

[35] M. Creel, "Creating and using a non-dedicated HPC cluster with Parallel-Knoppix," in *Proc. of the 12th International Conference on Computing in Economics and Finance*, no. 202, Cyprus, Jun 22-24, 2006.

[36] M. Creel, "User-friendly parallel computations with Econometric examples," *Computational Economics*, vol. 26, no. 2, pp. 107-128, Springer, October 2005. DOI: 10.1007/s10614-005-6868-2.

[37] J. A. Vrugt, H. V. Gupta, B. Ó Nualláin and W. Bouten, "Real-Time data assimilation for operational ensemble streamflow forecasting," *Journal of Hydrometeorology*, vol. 7, no. 3, pp. 548-565, June 2006. DOI: 10.1175/JHM504.1

[38] J. A. Vrugt, B. Ó Nualláin, B. A. Robinson, W. Bouten, S. C. Dekker and P. M. A. Sloot, "Application of parallel computing to stochastic parameter estimation in environmental models," *Computers & Geosciences*, vol. 32, iss. 8, October 2006, pp. 1139-1155. DOI: 10.1016/j.cageo.2005.10.015. Ver pies de página en p. 1140, Sect. 4, Figs. 6, 8, Sect. 6.

[39] J. A. Vrugt, H. V. Gupta, S. C. Dekker, S. Sorooshian, T. Wagener and W. Bouten, "Application of stochastic parameter optimization to the Sacramento Soil Moisture Accounting model," *Journal of Hydrology*, vol. 325, 2006, pp. 288–307. DOI: 10.1016/j.jhydrol.2005.10. 041. pp. 291, 305.

[40] J. August and T. Kanade, "Scalable regularized tomography without repeated projections," in *Proc. 18th Int. Parallel and Distributed Processing Symposium (IPDPS'04)*, pp. 232-239, 26-30 April 2004, Santa Fe, New Mexico. DOI: 10.1109/IPDPS.2004.1303277. Sects. 4/5, p. 237, Fig. 5, p. 238.

[41] T. Varslot and S.-E. Måsøy, "Forward propagation of acoustic pressure pulses in 3D soft biological tissue," *Modelling, Identification and Control*, vol. 27, no. 3, pp. 181-190. Ver Sect. 5, p. 196, y último párrafo en las conclusiones.

[42] M.Zhao, V. Chadha and R. J. Figueiredo, "Supporting application-tailored Grid File System sessions with WSRF-based services," in *Proc. of the 14th IEEE Int. Symp. on High Perf. Distributed Computing HPDC-14*, pp. 24-33, 2005. DOI 10.1109/HPDC.2005.1520930.

[43] M. Zhao and R. J. Figueiredo, "Application-tailored cache consistency for Wide-Area File Systems," in *Proc. of the 26th International Conference on Distributed Computing Systems (ICDCS 2006)*, pp. 41-50, July 4-7 2006, Lisboa, Portugal. DOI: 10.1109/ICDCS.2006.17.

[44] J. Kepner and S. Ahalt, "MatlabMPI," *Journal of Parallel and Distributed Computing*, vol. 64, iss. 8, pp. 997-1005, Elsevier, August 2004. DOI: 10.1016/j.jpdc.2004.03.018.

[45] R. Choy and A. Edelman, "Parallel MATLAB: doing it right," *Proceedings of the IEEE*, vol. 93, iss. 2, Feb. 2005, pp. 331-341. DOI: 10.1109/JPROC.2004.840490.

[46] S. Raghunathan, "Making a supercomputer do what you want: High-level tools for parallel programming," *Computing in Science & Engineering*, vol. 8, no. 5, Sept.-Oct. 2006, pp. 70-80. DOI: 10.1109/MCSE.2006.93.

[47] R. Soganci, F. Gürgen and H. Topcuoglu, "Parallel Implementation of a VQ-based text-independent speaker identification," in *Proc. 3rd ADVIS 2004, Lect. Notes Comput. Sc*., vol. 3261, pp. 291-300, 2004.

[48] C. Bekas, E. Kokiopoulou and E. Gallopoulos, "The design of a distributed MATLAB-based environment for computing pseudospectra," *Future Generation Computing Systems*, vol. 21, iss. 6, pp. 930-941, Elsevier, Jun 2005. DOI: 10.1016/j.future.2003.12.017.

[49] J. Kepner, "Parallel Programming with MatlabMPI," in *Agenda 5th Annual Workshop on High Performance Embedded Computing HPEC'01*, MIT Lincoln Laboratory, Lexington, MA, 27-29 Nov. 2001.

EDUCOSTA
EDITORIAL UNIVERSITARIA DE LA COSTA

Disponible en: http://arxiv.org/abs/astro-ph/0107406.

[50] E. Manolakos, "Rapid Prototyping of Matlab/Java Distributed Applications using the JavaPorts components," in *Proc. 6th Annual Workshop on High Performance Embedded Computing HPEC'02*, MIT Lincoln Laboratory, Lexington, MA, 24-26 Sept. 2002. Disponible en: http://www.ll.mit.edu/HPEC/agendas/proc02/presentations/pdfs/4.4-manolakos.PDF

[51] S. Gallopoulos, "PSEs in Computational Science and Engineering education & training," in *Advanced Environments and Tools for High Performance Computing*, EuroConference on Problem Solving Environments and the Information Society, University of Thessaly, Greece, 14-19 June 2003. pp. 50, 77.

[52] G. Landi, E. L. Piccolomini and F. Zama, "A parallel software for the reconstruction of dynamic MRI sequences," in *Proc. 10th EuroPVM/MPI, Lect. Notes Comput. Sc.*, vol. 2840, pp. 511-519, Springer, 2003.

[53] T. Andersen, A. Enmark, D. Moraru, C. Fan, M. Owner-Petersen, H. Riewaldt, M. Browne and A. Shearer, "A parallel integrated model of the Euro50," in *Proc. of the SPIE*, vol. 5497, paper-ID [5497-25], Europe International Symposium on Astronomical Telescopes, 21-25 June 2004, Glasgow, Scotland, United Kingdom. Ver Table 1.

[54] M. Browne, T. Andersen, A. Enmark, D. Moraru and A. Shearer, "Parallelization of MATLAB for Euro50 integrated modeling," in *Proc. of the SPIE*, vol. 5497, paper-ID [5497-71], Europe International Symposium on Astronomical Telescopes, 21-25 June 2004, Glasgow, Scotland, United Kingdom. Ver última página.

[55] D. Petcu, D. Tepeneu, M. Paprzycki, T. Mizutani and T. Ida, "Survey of symbolic computations on the Grid," in *Proc. of the 3rd Int. Conference Sciences of Electronic, Technologies of Information and Telecommunications SETIT 2005*, Susa, Tunisia, March 27-31, 2005. Ver p. 4/11.

[56] D. Petcu, D. Tepeneu, M. Paprzycki, and T. Ida, "Symbolic computations on the Grid," in B. Di Martino *et al.* (eds.), *Engineering the Grid; Status and Perspective,* America Scientific Publishers, Los Angeles, CA, Jan. 2006, Ch. 27, pp. 91-107.

[57] S. Goasguen, "High performance computing for nanoscale device simulation," in *The Army Research Office (ARO) FY2001 Defense University Research Initiative on Nanotechnology (DURINT), Kick-Off Meeting,* Third-Year Review, July 24-25, 2003. Disponible en: http://nanolab.phy.stevens-tech.edu/DURINT2003/PDF/Goasquen.pdf. pp. 9-22, 28/34. Enlace recuperable mediante Internet Archive Way Back Machine http://liveweb.archive.org/http://nanolab.phy.stevens-tech.edu/DURINT2003/PDF/Goasquen.pdf.

[58] C. Shue, J. Hursey and A. Chauhan, "MPI over scripting languages: usability and performance tradeoffs," *IUCS Technical Report TR631*, University of Indiana, Feb. 2006. Ver Fig. 5, pp. 11/13.

[59] M. Collette, B. Corey and J. Johnson, "High performace tools and technologies," *Technical Report UCRL-TR-209289,* Lawrence Livermore National Laboratory (LLNL), US. Dept. of Energy, December 2004. pp. 67-68/79.

[60] R. Serban, *LLNL: SundialsTB, a Matlab Interface to SUNDIALS*. Disponible en: http://www.llnl.gov/CASC/sundials/documentation/stb_guide/sundialsTB.html

[61] M. Creel, *OctaveForge: Econometrics package for Octave*. Disponible en: http://octave.sourceforge.net/econometrics/index.html. Ver por ejemplo en Function Reference la función gmm_estimate cuyo último parámetro requiere MPITB.

[62] M. Creel, Universidad Autónoma de Barcelona, Spain: *Parallel-Knoppix Linux,* Disponible en: http://pareto.uab.es/mcreel/ParallelKnoppix/. Ver Sección Thanks. Enlace recuperable mediante Internet WayBack Machine http://web.archive.org/web/20080819061319/http://pareto.uab.es/mcreel/ParallelKnoppix/

[63] M. Creel, Universidad Autónoma de Barcelona, Spain: *PelicanHPC Linux,* Disponible en: http://pareto.uab.es/mcreel/PelicanHPC/. Ver MPITB en Academic Work.

[64] VL-E Project, *Virtual Laboratory for e-Science, Dutch EZ ICT innovation program*. Disponible en: http://poc.vl-e.nl/. Ver Release Content.

[65] S. Goasguen, "A guided tour of nanoMOS code and some tips on how to parallelize it," summer course "Electron Devices at the Nano/Molecular Scale," Summer School at UIUC, University of Illinois, Urbana-Champaign, May, 21-22, 2002. Disponible en: http://www.mcc.uiuc.edu/summerschool/2002/Mark%20Lundstrom/Lundstrom_files/COURSE19.pdf, pp. 10-11, 13, 17/17.

[66] S. Goasguen, "On the use of MPI and PVM in Matlab," *Computing Research Institute Seminars,* CS 111, March 26th, 2003. Disponible en: http://www.cs.purdue.edu/calendar/webevent.cgi?cmd=showevent&id=426

[67] M. Law, "MATLAB laboratory for MPI Toolbox (MPITB)," *MATH2160 Mathematical & Statistical Software Laboratory*, Nov. 2003, Hong-Kong Baptist University HKBU. Disponible en: http://www.sci.hkbu.edu.hk/tdgc/TeachingMaterial/MPITB/MPITB.pdf

[68] M. Law, "Guest Lecture on Cluster Computing," *COMP3320 Distributed Systems, Cluster Computing 2006 Lecture, HKBU*. Disponible en: http://www.comp.hkbu.edu.hk/~jng/comp3320/3320-Cluster2006.ppt, pp. 16/22.

[69] M. Law, "Experiencing cluster computing, Class 2: Overview," *Learning Computational Science on a Parallel Architecture,* tutorial 5, HKBU, 2002-2012. Disponible en: http://www.sci.hkbu.edu.hk/tdgc/tutorial/ExpClusterComp/ExpClusterComp02.ppt, pp. 6, 33/46.

[70] M. Law, "Recurring HPC course, Syllabus V: MPI Toolbox (2 hrs)," *Learning Computational Science on a Parallel Architecture*, tutorial 3, HKBU, 2002-2012. Disponible en: http://www.sci.hkbu.edu.hk/tdgc/tutorial/RHPCC/syllabus.php

[71] B. Skinner, "Introduction to MPITB," *Users' presentations, Matlab support,* Research Computing website, University of Technology Sydney (UTS). Disponible en: http://services.eng.uts.edu.au/ResearchComputing/support/matlab.html#Presentations. Recuperable con Internet Archive WayBack Machine http://web.archive.org/web/20060904035940/

EDUCOSTA
EDITORIAL UNIVERSITARIA DE LA COSTA

http://services.eng.uts.edu.au/ResearchComputing/support/matlab.html

[72] A. H. Davis, "Structuring collaborative visualization using facet trees: Design and Implementation of a Prototype System," Honours Thesis, School of CIT, Griffith University, Australia, October 2000. Disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.133.1715&rep=rep1&type=pdf, pp. 10/163

[73] L. Y. Choy, "MATLAB*P 2.0: Interactive supercomputing made practical," M. Sc. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Sept. 2002. Disponible en: http://people.csail.mit.edu/cly/thesis.ps.gz, pp. 12, 14/67.

[74] H. Singh, "Parallel programming in MATLAB," M. Sc. Thesis, University of East Anglia, 2002. Disponible en: http://www.uea.ac.uk/~a207713/ thesis/report.pdf, pp. 26-27, 65, 69/108, Fig. 4.1 p. 33. Enlace recuperable mediante Internet Archive WayBack Machine http://web.archive.org/web/20060614222026/http://www.uea.ac.uk/~a207713/thesis/report.pdf

[75] S. Merchant, "Approaches for MATLAB applications acceleration using HPRC," M. Sc. Thesis, ECE Department, University of Tennessee, Knoxville, August 2003. Disponible en: http://web.eecs.utk.edu/~gdp/pdf/merchant-ms-thesis.pdf, pp. 10/163.

[76] A. Rajeev, "Peer-to-peer support for Matlab-style computing," M. Sc Thesis, Texas A&M University, May 2004. Disponible en: http://repository.tamu.edu//handle/1969.1/503, pp. 10-12, 49/53.

[77] R. Choy, "Parallel Matlab survey," Disponible en: http://people.csail.mit.edu/cly/survey.html. Internet WayBack Machine http://web.archive.org/web/20070705000531/http://people.csail.mit.edu/cly/survey.html

[78] C. Moler, *The Mathworks,* Natick, MA, personal communication, comp.soft-sys.matlab newsgroup, Nov. 21, 2001. Disponible en: http://groups.google.com/group/comp.soft-sys.matlab/browse_frm/thread/cb42ee88cd8af834/8b08b6005b61be54. Ver también S. Pawletta, University of Rostock, Germany, comp.soft-sys.matlab newsgroup, May 25, 1999. Disponible en: https://groups.google.com/group/comp.soft-sys.matlab/browse_thread/thread/38c19ab989fd56f7/a6a990b2b60f3e9b?q=pawletta

[79] S. Pawletta, T. Pawletta, W. Drewelow, P. Duenow and M. Suesse, "A Matlab toolbox for distributed and parallel processing," Int. Matlab Conference 95, Cambridge, MA, 8 pages.

[80] S. Pawletta, T. Pawletta and W. Drewelow, "Distributed and parallel simulation in an interactive environment," in *Proc. of the 1995 EUROSIM Conference,* EUROSIM '95, Vienna, Austria, Elsevier Science Publisher B.V., September 1995, pp. 345-350.

[81] S. Pawletta, T. Pawletta and W. Drewelow, "HLA-based Simulation within an Interactive Engineering Environment," In *Proc. 4th IEEE Int. Workshop on Distributed Simulation and Real-Time Applications,* DS-RT'2000, San Francisco, CA, USA, August 2000, pp. 97-102. DOI: 10.1109/DISRTA.2000.874068.

[82] S. Pawletta, W. Drewelow and T. Pawletta, "Distributed and Parallel Processing with Matlab -- The DP Toolbox," *Simulation News Europe (SNE)*, Hrsg. ARGESIM/ ASIM, Wien, 2001, no. 31, pp. 13-14.

[83] T. Pawletta, C. Deatcu, O. Hagendorf, S. Pawletta and G. Colquhoun, "DEVS-Based Modeling and Simulation in Scientific and Technical Computing Environments," in *Proc. of DEVS Integrative M&S Symposium (DEVS'06)* - 2006 Spring Simulation Multiconference (SpringSim'06), Huntsville/AL, USA, April 2-6, 2006, pp. 151-158.

[84] J. Hollingsworth, K. Liu and P. Pauca, "PT v. 1.00: Manual and Reference Pages," *Technical Report, Mathematics and Computer Science Department,* Wake Forest University, 1996. Disponible en: http://www.math.wfu.edu/pt/pt.html. Enlace Internet WayBack Machine http://web.archive.org/web/20060620113750/http://www.math.wfu.edu/pt/pt.html

[85] A. E. Trefethen, V. S. Menon, C. Chang, G. Czajkowski, C. Myers and L. N. Trefethen, "MultiMATLAB: MATLAB on Multiple Processors," *Technical Report: TR96-1586*, Cornell University, NY, USA, 1996. Disponible en: http://portal.acm.org/citation.cfm?id=866863. See also URL: http://www.cs.cornell.edu/Info/People/lnt/multimatlab.html

[86] V. Menon and A. E. Trefethen, "MultiMATLAB: integrating MATLAB with high-performance parallel computing," in *Proceedings of the 1997 ACM/IEEE Conference on Supercomputing* (San Jose, CA, November 15-21, 1997). Supercomputing'97. 18 pages. DOI: http://doi.acm.org/10.1145/509593.509623.

[87] J. Zollweg, *CMTM (Cornell Multitask Toolbox for MATLAB) web*, Disponible en: http://www.tc.cornell.edu/Services/Support/Forms/cmtm. Enlace recuperable mediante Internet WayBack Machine http://web.archive.org/web/20081121190121/http://www.tc.cornell.edu/Services/Support/Forms/cmtm

[88] P. Husbands and C. Isbell, *PPserver (Parallel Problems Server) web*, Disponible en: http://crd.lbl.gov/~parry/text/ppserver/

[89] P. Husbands and C. Isbell, "The Parallel Problems Server: A Client-Server Model for Large Scale Scientific Computation," in *Proceedings of the Third International Conference on Vector and Parallel Processing, VecPar'98*. Portugal, 1998, pp. 156-169.

[90] P. Husbands, C. Isbell and A. Edelman, "Interactive Supercomputing with MIT-Matlab," *MIT AI Memo 1642*. Presented at: the Second IMA Conference on Parallel Computation. Oxford, 1998. 10 pages.

[91] P. Husbands and C. Isbell, "MATLAB*P: A Tool for Interactive Supercomputing," in *Proc. 9th SIAM Conference on Parallel Processing for Scientific Computing*, 1999.

[92] C. Isbell and P. Husbands, "The Parallel Problems Server: An Interactive Tool for Large Scale Machine Learning," in *Advances in Neural Information Processing Systems*, vol. 12, 2000.

[93] C. Moler, "Objectively speaking," *Cleve's Corner, MATLAB News&Notes*, Winter 1999. The Mathworks. Disponible en: http://www.mathworks.com/company/newsletters/news_notes/clevescorner/

EDUCOSTA
EDITORIAL UNIVERSITARIA DE LA COSTA

[94] C. Moler, "MATLAB incorporates LAPACK," *Cleve's Corner, MATLAB News&Notes*, Winter 2000. The Mathworks. Disponible en: http://www.mathworks.com/company/newsletters/news_notes/clevescorner/

[95] R. Choy, D. Cheng, A. Edelman, J. Gilbert and V. Shah, "Star-P: High Productivity Parallel Computing," in *Proc. 8th HPEC 2004, MIT Lincoln Laboratory*, 28-30 Sept. 2004. Disponible en: http://www.ll.mit.edu/HPEC/agendas/proc04/abstracts/choy_ron.pdf. Ver también diploma ganador MIT $1K en p. 2, Disponible en: http://www.ll.mit.edu/HPEC/agendas/proc04/Talks/Wed/Focus2/choy.pdf

[96] V. Shah and J. R. Gilbert, "Sparse Matrices in Matlab*P: Design and Implementation," in *Proc. 11th International Conference High Performance Computing - HiPC 2004*, Bangalore, India, December 19-22, 2004, pp. 144-155. DOI: 10.1007/b104576.

[97] J. Gilbert, V. Shah, T. Letsche, S. Reinhardt and A. Edelman, "An Interactive Approach to Parallel Combinatorial Algorithms with Star-P", in *Proc. 9th HPEC 2005, MIT Lincoln Laboratory,* 20-22 Sept. 2005. Disponible en: http://www.ll.mit.edu/HPEC/agendas/proc05/Day_2/Abstracts/ 1030_Edelman_A.PDF

[98] A. Edelman, P. Husbands and S. Leibman, "Interactive Supercomputing's Star-P platform: Parallel MATLAB and MPI Homework Classroom Study on High Level Language Productivity," in *Proc. 10th High Performance Embedded Computing Workshop (HPEC 2006)*, MIT Lincoln Lab., 2006.

[99] A. Edelman, "Parallel MATLAB(R) doing it right," in *6th Annual Workshop on Linux Clusters for Super Computing, LCSC'05,* October 17-19, 2005. National Supercomputer Centre (NSC). Linköping University, Sweden. Disponible en: http://www.nsc.liu.se/lcsc2005/programme.html#LCSC, see abstract at URL: http://www.nsc.liu.se/lcsc2005/programme.html#edelman

[100] J. Kepner and N. Travinin, "Parallel Matlab: The Next Generation," in *Proc. 7th HPEC 2003,* MIT Lincoln Laboratory, 23-25 Sept. 2003.

[101] R. Haney, A. Funk, J. Kepner, H. Kim, C. Rader, A. Reuther and N. Travinin, "pMatlab takes the HPCchallenge," in *Proc. 8th HPEC 2004,* MIT Lincoln Laboratory, 28-30 Sept. 2004.

[102] J. Kepner, T. Currie, H. Kim, B. Mathew, A. McCabe, M. Moore, D. Rabinkin, A. Reuther, A. Rhoades, L. Tella and N. Travinin, "Deployment of SAR and GMTI Signal Processing on a Boeing 707 Aircraft using pMatlab and a Bladed Linux Cluster," in *Proc. 8th HPEC 2004,* MIT Lincoln Laboratory, 28-30 Sept. 2004.

[103] J. Kepner, "HPC Productivity: An Overarching View," *International Journal of High Performance Computing Applications, Special Issue on HPC Productivity,* J. Kepner (editor), vol. 18, no. 4, pp. 393-397, Nov. 2004.

[104] N. Travinin, R. Bond, J. Kepner and H. Kim, "pMatlab: High Productivity, High Performance Scientific Computing," 2005 SIAM Conference on Computational Science and Engineering, February 12, Orlando, FL. USA.

[105] N. Travinin, H. Hoffmann, R. Bond, H. Chan, J. Kepner and E. Wong, "pMapper: Automated Mapping of Parallel Matlab Programs," in *Proc. 9th HPEC 2005*, MIT Lincoln Laboratory, 20-22 Sept. 2005. Disponible en: http://www.ll.mit.edu/HPEC/agendas/proc05/Day_2/Abstracts/1345_ Travinin_A.PDF

[106] N. Bliss and J. Kepner, "pMatlab parallel matlab library," *International Journal of High Performance Computing Applications, Special Issue on High Level Programming Languages and Models,* J. Kepner and H. Zima (editors), 2006.

[107] *The MathWorks, Distributed Computing Toolbox,* Enlaces de 2007 recuperables mediante Internet WayBack Machine, URL: http://web.archive.org/web/20071217040037/http://www.mathworks.com/products/distribtb/, características de la versión 3.2, Sep. 2007, URL: http://web.archive.org/web/20071214225739/http://www.mathworks.com/products/distribtb/whatsnew.html, versión 2.0, Nov. 2005, URL: http://web.archive.org/web/20070817181458/http://www.mathworks.com/company/pressroom/articles/article11320.html, 1.0, Nov. 2004, URL: http://web.archive.org/web/20051124162240/http://www.mathworks.com/company/pressroom/articles/article8192.html

[108] C. Moler, "Parallel MATLAB," invited presentation in *Workshop on State-of-the-art in Scientific and Parallel Computing, PARA'06,* Umeå, Sweden, June 18-21, 2006. Session IP4, Disponible en: http://web.archive.org/web/20070126024524/http://www.hpc2n.umu.se/para06/index.php?content=ip4, http://www.hpc2n.umu.se/node/647

[109] J. Gilbert, S. Reinhardt and V. Shah, "High-performance graph algorithms from parallel sparse matrices," in *Workshop on State-of-the-art in Scientific and Parallel Computing, PARA'06,* Umeå, Sweden, June 18-21, 2006. Session MS5, Disponible en: http://www.hpc2n.umu.se/para06/ index.php?content=ms5

[110] The MathWorks, Media Newsletter April 2006, article "Distributed Computing Speeds Application Development for Technical Computing" Disponible en: http://www.mathworks.com/company/pressroom/newsletter/may06/distrib.html, Enlace recuperable mediante Internet WayBack Machine http://web.archive.org/web/20080907183646/http://www.mathworks.co.uk/company/pressroom/newsletter/may06/distrib.html

[111] Workshop on distributed computing with MATLAB, March 20-22, 2006, Arctic Region Supercomputing Center (ARSC), University of Alaska Fairbanks campus. Disponible en: http://www.arsc.edu/support/training/MATLAB_2006.html. Enlace recuperable mediante Internet WayBack Machine http://web.archive.org/web/20080528163420/http://www.arsc.edu/support/training/MATLAB_2006.html

[112] C. Moler, "Is it Finally the Time for a Parallel Matlab?," *Householder Symposium XVI, 2005, Householder Meeting on Numerical Linear Algebra,* May 23-27, 2005, Champion, Pennsylvania, USA. Resumen Disponible en: http://www.

EDUCOSTA
EDITORIAL UNIVERSITARIA DE LA COSTA

cse.psu.edu/~hh2005/H05Abstracts.pdf, p. 170. Enlace recuperable mediante Internet WayBack Machine http://web.archive.org/web/20060909160929/http://www.cse.psu.edu/~hh2005/H05Abstracts.pdf. Ver también C. Moler, "Why there isn't a parallel MATLAB," *Cleve's Corner, MATLAB News&Notes*, Spring 1995. The Mathworks. Disponible en: http://www.mathworks.com/company/newsletters/news_notes/pdf/spr95cleve.pdf

[113] A. Edelman *et al.*, Computational Research in Boston (CRiB) seminar series. Disponible en: http://www-math.mit.edu/crib/

[114] A. Edelman, R. Choy, J. Gilbert and V. Shah, "Parallel Computing Made Easy with STAR-P," *short course in SIAM Conference on Parallel Processing for Scientific Computing (PP04)*, San Francisco, CA, Feb. 24, 2004. Disponible en: http://www.siam.org/meetings/pp04/edelman.htm

[115] A. Edelman, "Interactive Parallel MATLAB® with Star-p," *Seminar in Umeå Center for Interaction Technology,* Oct. 20th 2005. Disponible en: http://www.ucit.umu.se/main.php?view=sem/9. Enlace recuperable mediante Internet WayBack Machine http://web.archive.org/web/20070520201742/http://www.ucit.umu.se/main.php?view=sem/9

[116] *The MathWorks,* SC|05 & SC|06 booth press releases, Disponible en: http://web.archive.org/web/20080905111539/http://www.mathworks.com/company/pressroom/articles/article11350.html, http://web.archive.org/web/20071012121615/http://www.mathworks.com/company/events/tradeshows/tradeshow12352.html

[117] J. Kepner, A. Reuther, L. Dean and S. Grad-Freilich, "Parallel and Distributed Computing with MATLAB," *Tutorial S07 in SuperComputing 2005, SC|05.* Disponible en: http://sc05.supercomputing.org/schedule/event_detail.php?evid=5115

[118] A. Edelman, J. Nehrbass, "Interactive Supercomputing with Star-P and MATLAB," Tutorial S11 in SuperComputing 2005, SC|05. Disponible en: http://sc05.supercomputing.org/schedule/event_detail.php?evid=5104

[119] J. Kepner, "HPCS (Workshop on High Productivity Computing Systems)," *Workshop in SuperComputing 2005, SC|05.* Disponible en: http://sc05.supercomputing.org/schedule/event_detail.php?evid=5284

[120] S. Chapin and J. Worringen, "Operating Systems," *International Journal of High Performance Computing Applications*, vol. 15, no. 2, pp. 115-123. SAGE publications, 2001.

[121] M. Baker, "Preface," *International Journal of High Performance Computing Applications*, vol. 15, no. 2, p. 91. SAGE publications, 2001.

[122] B. Wilkinson and M. Allen, *Parallel Programming: Techniques and Applications using Networked Workstations and Parallel Computers,* 2nd Ed., Pearson - Prentice Hall, NJ, 1999.

[123] *Wikipedia*, Mandelbrot set. Disponible en: http://en.wikipedia.org/wiki/Mandelbrot_set