Revista Ingeniería UC, Vol. 16, No. 3, Diciembre 2009 45 - 50

Parlab: un ambiente interactivo de programación con sintaxis Matlab/Octave para ejecución eficiente de operaciones matriciales en multiprocesadores

Demetrio Rey*, Enrique Flores, Edwin Vargas

Instituto de Matemática y Cálculo Aplicado, Facultad de Ingeniería, Universidad de Carabobo, Bárbula–Naguanagua, Código Postal 2008. Estado Carabobo, Venezuela.Teléfono: (58–241) 8672710 Extensión 264.

Resumen.-

ParLab es un sistema interactivo de programación que se encuentra actualmente en desarrollo en la Facultad de Ingeniería de la Universidad de Carabobo, cuyo objetivo es facilitar la migración de aplicaciones de cálculo intensivo desarrolladas en MATLAB u Octave, para su ejecución eficiente en computadores multiprocesadores. ParLab interpreta, paraleliza, y ejecuta instrucciones con sintaxis similar a MATLAB/Octave, en forma interactiva. Este ambiente permite la ejecución de aplicaciones en un computador multiprocesador, sin necesidad que el usuario paralelice explícitamente el código. En este trabajo se discute la motivación y objetivos de diseño de ParLab, estrategias de paralelización, limitaciones y resultados preliminares.

Palabras clave: Aplicaciones de cálculo intensivo, computador multiprocesador, paralelización

ParLab: An Interactive Programming Environment with MATLAB/Octave Syntax for Efficient Matrix Operations on Multiprocessors

Abstract.-

ParLab is an interactive programming system currently in development at Facultad de Ingeniería of Universidad de Carabobo that aims to help with migration of computing intensive applications developed with MATLAB or Octave for efficient execution on multiprocessor computers. ParLab interprets, parallelizes, and executes with syntax similar to MATLAB/Octave in an interactive way. This environment allows parallel execution of applications in a multiprocessor computer without requiring the user to explicitly parallelize code. In this work, we discuss the motivations, design objectives of ParLab, as well as the parallelization strategies, limitations and preliminary results.

Keywords: Computing intensive applications, Multiprocessor computer, Parallelization

1. Introducción

MATLAB es una de las herramientas más utilizadas para el prototipo de aplicaciones de cálculo matemático. Su carácter interactivo, el alto nivel de las instrucciones y operadores, la extensa disponibilidad de funciones y subrutinas, herramientas de visualización y *toolboxes* específicos para diferentes disciplinas, han consolidado el uso de esta herramienta en la comunidad de investigadores, académicos y profesionales en ciencias e ingeniería [1]. Otro lenguaje de extensiva utilización por esta comunidad es Octave [2], el cual es un sistema similar a MATLAB en cuando a su sesión interactiva y la sintaxis del lenguaje, limitado en cuanto *toolboxes* y recursos gráficos, pero con la ventaja de ser software libre.

La experiencia en el Instituto de Matemática y Cómputo Aplicado en la Facultad de Ingeniería de la Universidad de Carabobo no es diferente a la de otros centros de cómputo: diversas aplicaciones que se solicitan para su ejecución en computadores paralelos, tienen un prototipo codificado en MATLAB u Octave. Para ejecutar dichas aplicaciones en un computador paralelo, generalmente se requiere volver a codificar estos programas con un lenguaje estándar para estos ambientes, tal como C/MPI (MPI: Message Passing Interface, interfaz de paso de mensajes), Fortran/MPI, HPF [3] o ZPL [4]. Sin embargo, esta transición suele ser un proceso no trivial. En los casos de C y Fortran con MPI, el programador debe trabajar en un lenguaje escalar, con lazos e indexación de elementos, e invertir un tiempo de desarrollo apreciable en la partición del

*Autor para correspondencia
Correos-e: drey@uc.edu.ve (Demetrio Rey), evflores@uc.edu.ve (Enrique Flores), vargase@uc.edu.ve (Edwin Vargas)

problema, en la comunicación y en la sincronización de los nodos de cómputo. En los casos de HPF y ZPL, generalmente el programador se enfrenta a un nuevo lenguaje que requiere un tiempo significativo de aprendizaje.

Otra opción alternativa sería la paralelización automática del prototipo de MATLAB. En este caso ideal no se requeriría volver a desarrollar la aplicación, sino modificarla ligeramente para que pudiera aprovechar automáticamente la presencia de más de un procesador. Si se observase una aceleración significativa del cómputo con un prototipo en particular, implicaría que la aplicación en cuestión tiene potencial de aprovechar eficientemente un sistema de alto rendimiento. Esto sería de gran valor para justificar una posterior recodificación con herramientas estándar como C/MPI o HPF. Aún más interesante sería el caso en el que un rendimiento bastante satisfactorio del prototipo en MATLAB/Octave evite la necesidad de recodificación.

Para responder a esta motivación, en este trabajo presentamos ParLab, un sistema interactivo de programación paralela actualmente en desarrollo en la Facultad de Ingeniería de la Universidad de Carabobo, cuyo objetivo es la ejecución de eficiente de aplicaciones de cálculo intensivo a partir de prototipos con sintaxis similar a la de MATLAB/Octave en sesiones interactivas. En este trabajo presentamos los aspectos fundamentales ParLab y resultados preliminares de experimentación.

2. Factibilidad de paralelización del lenguaje MATLAB

Desde los inicios de MATLAB se han hecho esfuerzos para paralelizar este ambiente de trabajo en forma transparente al usuario, y se han identificado dificultades importantes para su paralización óptima [5]. A pesar de estos inconvenientes, ciertos aspectos fundamentales del lenguaje MATLAB facilitan su paralelización automática:

Operaciones matriciales de alto nivel y funciones matemáticas de alta complejidad que abarcan una cantidad de operaciones individuales considerable. Con un costo computacional alto para cada instrucción, la relación comunicación/cómputo para su ejecución paralela se hace más favorable. Algunos casos, donde la cantidad de operaciones es $O(n^3)$, se muestran en la Figura 1:

```
A = B * C              % multiplicación de matrices
[L,U,P] = lu (A)       % factorización lu
E = eig(A)             % cálculo de autovalores
```

Figura 1: Algunas operaciones matriciales en sintaxis de MATLAB/Octave

Código vectorizado: similar al caso anterior, las instrucciones vectorizadas aumentan el costo computacional por instrucción, lo cual favorece su paralelización (Figura 2).

```
A =  B(:,k)  *  C(k,:)    % multiplicación columna por fila
S = sum(1:100)            % sumatoria progresión aritmética
```

Figura 2: Algunas operaciones vectorizadas en sintaxis de MATLAB/Octave

Programas compactos (pocas instrucciones). Lo cual minimiza el costo de interpretación de instrucciones respecto al tiempo total de ejecución.

En este sentido, si nos restringimos únicamente a la sintaxis del lenguaje, y evaluando las características mencionadas, se concluye que es posible crear subrutinas optimizadas para la ejecución en paralelo de varios de los operadores y funciones. Por ejemplo, la multiplicación de matrices (denotada en la Figura 1 y en la Figura 2 con un asterisco, *) podría ser implementada con una subrutina paralela de multiplicación de alto rendimiento. De igual forma podrían ser implementadas las funciones `lu()`, `eig()`, `sum()`, etc.

2.1. ParLab y su estrategia de programación

ParLab es un ambiente interactivo de programación cuyo objetivo es ejecutar cálculos en aplicaciones de carácter intensivo de manera eficiente en multiprocesadores a partir de prototipos con sintaxis similar a la de MATLAB/Octave en forma interactiva. La interactividad de ParLab radica en que el usuario da inicio a una sesión donde a través de una línea de comandos introduce y modifica los datos a procesar; y una vez efectuado el cálculo de la data procesada, el usuario puede observar los resultados obtenidos en forma directa sobre la línea de comandos y dichos resultados pueden almacenarse en un espacio de trabajo de fácil acceso para su posterior uso, de requerirse. Así mismo, es posible que el usuario escriba y ejecute sus propias rutinas empleando constructos y operaciones disponibles en ParLab.

La eficiencia de ParLab se fundamenta en que las operaciones a realizar en la ejecución se apoyan en librerías de cálculo científico de libre acceso tales como ScaLAPACK [6], BLAS [7], LAPACK [8], BLACS [9] las cuales cuentan con rutinas especializadas y que

son llamadas de acuerdo a su disponibilidad y a la naturaleza de cada operación (secuencial o susceptible de paralelización). De no contarse con las rutinas apropiadas en las librerías ya nombradas, se hace el llamado a rutinas propias de ParLab.

Dado que se quiere que las operaciones tengan sintaxis similares a las de MATLAB/Octave, es deseable reducir al mínimo los cambios a las correspondientes instrucciones de MATLAB/Octave. Bajo esta premisa, se diseñó una estrategia de paralelización fundamentada en la distribución de datos sobre la red de procesadores en aquellas operaciones susceptibles de ser paralelizadas.

La estrategia de programación paralela de ParLab consiste en indicar cuales matrices están distribuidas sobre la red de nodos de cómputo, mediante la función `dist()`. Esta función se encarga de distribuir datos entre los procesadores: `dA = dist(A)`. En esta instrucción, la función `dist(A)`, distribuye los datos de la matriz *A* en toda la red de procesadores, siguiendo un esquema de bloque cíclico [6]. El arreglo resultante, *dA*, contiene los mismos elementos de *A*, pero distribuidos de manera que a cada procesador le corresponde una parte del arreglo total. El arreglo *dA* puede servir de operando o parámetro, al igual que *A*. Sin embargo, la característica fundamental de *dA*, es que todas las operaciones en las cuales forme parte, se realizarán en paralelo. En la Figura 3 se muestra un segmento de código donde se ejemplifica lo descrito:

```
1: A = rand(1000,1000);
2: B = rand(1000,1000);
3: dA = dist(A);
4: C = A * B;                    # multiplicación secuencial
5: dC = dA * B;                  # multiplicación paralela
```

Figura 3: Ejemplo de uso de la función `dist(A)` en ParLab

Todas las operaciones subsecuentes que involucren a *dA* en el código mostrado en la Figura 3, serán efectuadas en paralelo; $dC = dA * B$, se ejecutará en paralelo, dado que uno de los operandos (dA) está distribuido, mientras que $C = A * B$ será una operación secuencial.

El resultado de una operación paralela es generalmente una matriz paralela. Por ejemplo, el arreglo resultante *dC*, también será distribuido. Por lo tanto, el paralelismo se va propagando a las subsecuentes operaciones posteriores que contengan a *dC* como operando.

De acuerdo a los objetivos de diseño, no es necesario hacer llamadas a funciones MPI o sincronización. Las operaciones y funciones serán paralelas automáticamente cuando alguno o varios de los operandos o argumentos sean distribuidos, e igualmente cuando la variable a asignar sea distribuida. Las operaciones matriciales se encargarán de la comunicación y sincronización implícitamente.

2.2. Implementación

ParLab está desarrollado en C/MPI y hace uso de librerías de cómputo matemático secuenciales BLAS [7], LAPACK [8], y paralelas tal como BLACS [9] y ScaLAPACK [6] para alcanzar alto rendimiento en las operaciones matriciales secuenciales o paralelas.

```
1: function dist (  A: array  ): array
2:    if A not distributed then
3:       dA = block-cyclic distribute A over processors
4:    else
5:       dA = copy of A;
6: return dA
```

Figura 4: Implementación de la función `dist(A)` en ParLab

Algoritmo de la Función `dist()`. Esta función puede ser invocada por el usuario para cualquier matriz, o por las funciones de ParLab cuando debe distribuir un parámetro de entrada. Su implementación en ParLab se muestra en la Figura 4.

Algoritmo genérico de las operaciones matriciales. Las operaciones matriciales de ParLab determinan si uno de los operandos está distribuido, o si la variable que se va asignar al efectuar la operación está distribuida. En caso positivo, la operación matricial invocará a la rutina de ScaLAPACK correspondiente, o en caso contrario a una rutina en paralelo desarrollada específicamente en ParLab. Si ninguno de los operandos ni la variable asignada están distribuidos, entonces se ejecutará la rutina secuencial correspondiente, ya sea de LAPACK o propiamente de ParLab. En la figura 5, se muestra una implementación en ParLab del algoritmo genérico descrito.

3. Ejecución en paralelo de ParLab

La ejecución en paralelo de ParLab consiste en la corrida de una copia del interpretador en cada nodo que participa en el cómputo. Los P nodos se organizarán en un nodo principal (nodo 0) y los nodos secundarios (nodo 1 a nodo P–1). Cuando se abre la sesión interactiva, el nodo principal espera por el comando del usuario y los nodos secundarios esperan por la información que proviene del nodo principal. Una vez que el usuario introduce el comando, éste es propagado a todos los nodos secundarios con una llamada a *MPI_Broadcast*. Seguidamente,

```
1 : function matop ( A, B: array ): array
2 :   if A is distributed or B is distributed or
3 :       assigned variable is distributed;
4 :       dA = dist ( A )
5 :       dB = dist ( B )
6 :       if ScaLAPACK routine is available
7 :             dR := ScaLAPACK matop( dA, dB );
8 :       else
9 :             dR := parallel matop ( dA, dB );
10:       return dR;
11:   else
12:       if LAPACK routine is available
13:             R := LAPACK matop( A, B );
14:       else
15:             R := seq matop ( A, B );
16:       return R;
```

Figura 5: Implementación del algoritmo genérico de las operaciones matriciales en ParLab

cada nodo interpreta la instrucción, construyendo su árbol sintáctico (*Abstract Syntax Tree*, AST). Una vez construido el árbol, éste se evalúa para producir un valor, que puede ser impreso en pantalla y/o almacenado en la tabla de símbolos (*Workspace*) del programa (en caso de una asignación). Existe un árbol para cada tipo de instrucción: expresión, asignación, selección (*if*), lazos, funciones, etc. En caso de aquellas estructuras que se ejecutan repetidamente (lazos y subrutinas), la aceptación de la instrucción, *broadcast* y construcción del AST ocurre una sola vez. Cada vez que se ejecute una iteración o se llame a una función, simplemente se ejecuta el AST almacenado en memoria que fue construido al momento de la definición del lazo o función. La evaluación recursiva del AST abarca valores constantes, variables, operaciones aritméticas, lógicas, funciones, etc. La eficiencia de ParLab depende críticamente de la evaluación del AST, por lo que desde el inicio del desarrollo, se decidió que la función que implementa esta operación llame a rutinas de matemáticas optimizadas, cuando sea posible.

3.1. Replicación y distribución

Todos los nodos construyen y evalúan el AST de una instrucción, por lo cual todas las operaciones se ejecutan redundantemente. Los valores escalares y matrices no distribuidas de ParLab, son replicados en cada procesador. Esta redundancia ayuda a minimizar los mensajes que de otra forma se tendrían que efectuar para cada instrucción secuencial. Como cada nodo efectúa las mismas operaciones, las instrucciones secuenciales no generan ningún costo de comunicación y en promedio éstas se ejecutan en el mismo tiempo que si se ejecutaran en un solo nodo. En contraste, el paralelismo de ParLab reside en aquellas operaciones que se realizan sobre operandos distribuidos. En cuyo caso la instrucción se ejecuta en menor tiempo, dado el trabajo simultáneo de varios procesadores, cada uno sobre una parte de la data total.

El modelo de paralelismo descrito de ParLab corresponde estrictamente al de paralelismo de datos [10].

3.2. Limitaciones

La distribución de los datos en ParLab es fija para todas las matrices distribuidas. Todos los arreglos distribuidos están alineados sobre una red bidimensional de procesadores y exhiben distribución bloque–cíclica con un tamaño de bloque de 64×64 elementos.

Las funciones de graficación propias de MATLAB y Octave no están implementadas actualmente para ParLab. Puesto que éstas funciones son de utilidad para los usuarios de MATLAB y Octave, se tiene planeado implementarlas. Sin embargo, dichas funciones son inherentemente secuenciales y generan una gran concentración de datos hacia el nodo principal, por lo que su uso en programas de alto rendimiento debería ser limitado. Dado que ParLab es un proyecto de desarrollo propio, y no depende del código de otros sistemas como MATLAB u Octave, el número de funciones y operaciones implementadas es por los momentos limitado. Para optimizar los recursos invertidos en el proyecto, el progreso de implementación de funcionalidades dependerá de la demanda de la comunidad de usuarios.

4. Experimentación

Para comprobar el rendimiento y escalamiento de ParLab, se ejecutó un programa de solución de sistemas de ecuaciones lineales en un clúster de computadores, variando el tamaño del sistema (N). Se estableció la comparación de Octave (secuencial), y ParLab corriendo en $P = 1, 2$ y 4 procesadores.

Cada nodo del clúster de experimentación consta de un CPU Pentium IV 2,4 Ghz y 1 Gb de RAM. En cada nodo se tiene instalado el sistema operativo Linux (Kernel 2,6,18,6). Los nodos del clúster están conectados mediante una red de 1 Gigabit Ethernet. En pruebas anteriores [11], se determinó que la latencia de comunicación MPI internodo de este clúster es de 38,2 μs, y los mensajes cortos (4096 bytes) exhiben un tiempo de transferencia de 120 μs. El programa experimental se muestra en la Figura 6.

```
1: n = k;
2: a = dist([ ]);
3: b = dist([ ]);
4: a = rand(n,n);
5: b = rand(n,1);
6: tic;
7: x = a\b;
8: toc;
```

Figura 6: Implementación en ParLab de la solución de un sistema de ecuaciones lineales

N se varió de 2000 a 10000 elementos y se midió el tiempo de ejecución de la instrucción $x = a \setminus b$ únicamente (Tabla 1).

Tabla 1: Tiempos de solución, aceleración y eficiencia para un sistema de ecuaciones lineales de tamaño N con P procesadores. T: tiempo, S: Aceleración, E: Eficiencia

N	Octave, P=1	Parlab, P=1	Parlab, P=2	S, P=2	E, P=2	Parlab, P=4	S, P=4	E, P=4
	T(s)	T(s)	T(s)			T(s)		
2.00	4,16	3,84	3,5	1,10	55 %	2,54	1,51	38 %
3.00	13,33	12,45	10,59	1,18	59 %	7,04	1,77	44 %
4.00	30,65	29,07	24,91	1,17	58 %	15,42	1,89	47 %
5.00	57,85	55,83	46,21	1,21	60 %	27,65	2,02	50 %
6.00	98,82	95,71	78,85	1,21	61 %	45,81	2,09	52 %
7.00	167,08	150,780	121,88	1,24	62 %	75,23	2,00	50 %
8.00	–	227,65	182,93	1,24	62 %	104,15	2,19	55 %
9.00	–	400,3	253,74	1,58	79 %	142,04	2,82	70 %
10.00	–	–	349,43	–		193,85	–	

Los resultados mostrados en la Tabla 1, sugieren que el tiempo de ejecución de ParLab es similar al de Octave para un solo procesador, siendo el primero ligeramente más rápido. Para 1GB RAM por nodo, el máximo tamaño obtenido con Octave fue N=7000, mientras que ParLab pudo resolver un sistema de N = 9,000 en un procesador y N = 10,000 usando 2 ó 4 procesadores.

En el cómputo de la solución de ecuaciones lineales, cuando se ejecuta en dos procesadores ($P = 2$), la aceleración de ParLab está en un rango entre 1,10−1,58 respecto a ParLab con un solo procesador (P=1); y cuando se ejecuta con cuatro procesadores (P=4), la aceleración está en un rango entre 1,51 − 2,82. Estos valores corresponden a una eficiencia de paralelización en un rango de 55 % − 79 % para dos procesadores y de 38 % − 70 % para cuatro procesadores. Estos valores mejoran ligeramente si se considerara a Octave como el caso base. Se puede concluir entonces que aún en un clúster con computadores personales, ParLab puede incrementar significativamente el rendimiento de cómputo en sistemas de ecuaciones lineales de gran tamaño (Figura 7).

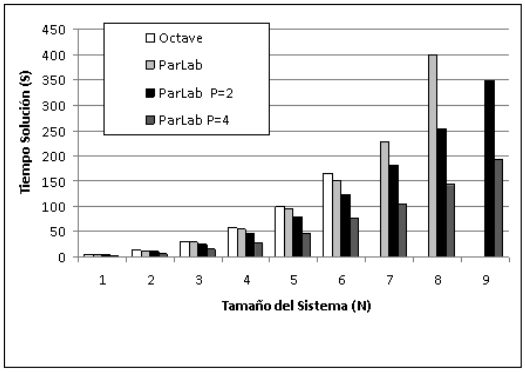

Figura 7: Tiempos de solución de sistema de ecuaciones tamaño N con P procesadores

Considerando el tiempo de envío nodo a nodo para mensajes pequeños (2K) medido en 120 μs., estimamos que el tiempo requerido en propagar las 8 instrucciones del programa sobre 4 nodos del cluster es de aproximadamente 4 ms. Este tiempo es bastante pequeño en relación a los tiempos de ejecución del programa ($2 - 200\,s$). Igualmente podemos extrapolar que el costo por interpretación de sólo 8 instrucciones es relativamente bajo respecto al tiempo total. Hemos efectuado experimentos midiendo el tiempo de ejecución de operaciones valores escalares, que corroboran esta afirmación. Todo esto indica que aún cuando el lenguaje sea interpretado y no compilado, se pueden alcanzar un rendimiento satisfactorio, dado el alto nivel de las instrucciones del lenguaje.

5. Conclusiones

ParLab permite abrir una sección interactiva en un computador de memoria distribuida o clúster, de forma similar a una sesión de MATLAB u Octave.

La sintaxis de las instrucciones de ParLab es similar a las de MATLAB y Octave, pero con la diferencia que pueden ejecutarse en paralelo y así aprovechar los recursos del clúster o supercomputador. ParLab paraleliza el código MATLAB/Octave implícitamente mediante la instrucción `dist()`, la cual distribuye los arreglos sobre la red de procesadores. Todas las instrucciones con operandos distribuidos serán paralelizadas automáticamente, y los resultados de operaciones paralelas resultan a su vez en matrices distribuidas. Dado que ParLab llama a funciones matemáticas de alto rendimiento secuenciales y paralelas, su rendimiento

es comparable al de Octave en un solo procesador; y para problemas de gran tamaño, el cómputo distribuido sobre una red de procesadores permite incrementar significativamente el rendimiento.

6. Trabajo futuro

Mediante este trabajo hemos presentado el sistema de cómputo interactivo paralelo ParLab y comprobado el rendimiento en un clúster. Ahora planeamos continuar la implementación de otras funciones y operaciones matriciales, así como estructuras de control de ejecución, lazos, subrutinas, tipos de datos complejos, y otras características que sean de interés para la comunidad de cómputo de alto rendimiento.

Igualmente, planeamos habilitar la ejecución multihilo para aprovechar eficientemente los procesadores multinúcleo y posteriormente lo sistemas mixtos (memoria distribuida con nodos multinúcleo).

7. Trabajos relacionados

La paralelización de códigos MATLAB u Octave, es el objetivo de varios proyectos de investigación académicos y comerciales. Mencionamos a continuación algunos de los que se encuentran activos actualmente. Matlab Parallel Toolbox [12] es el producto desarrollado por MathWorks para programación paralela en MATLAB. pMatlab[13] y MatlabMPI [14] son toolboxes paralelos para MATLAB del MIT Lincon Laboratory. Star*P (anteriormente MATLAB*p [15]) es un sistema interactivo con sintaxis de MATLAB, mientras que MPITB [16] es un toolbox para Octave.

Aquellos sistemas que trabajan como toolboxes generalmente están diseñados para una versión específica de MATLAB u Octave. ParLab no depende de MATLAB u Octave para funcionar dado que es un ambiente interactivo independiente. Una comparación más detallada con proyectos similares va más allá de los objetivos de este artículo, pero es de nuestro interés efectuar estudios cualitativos y cuantitativos en futuros trabajos.

Referencias

[1] The MathWorks, Inc. MATLAB – The Language Of Technical Computing. disponible en: `http://www.mathworks.com/products/matlab/`.

[2] Eaton J., Bateman D., Hauberg Soren. GNU Octave Manual Version 3. A high–level interactive language for numerical computations. Network Theory Ltd. Edition 3 for Octave version 3.0.2 August 2008.

[3] High Performance Fortran Forum. High Performance Fortran language specification, version 2.0. Technical report, Center for Research on Parallel Computation, Rice University, January 1997. `http://dacnet.rice.edu/Depts/CRPC/HPFF/versions/hpf2/hpfv20/hpf--report.html`.

[4] L. Snyder. A Programmer's Guide to ZPL. The MIT Press, 1999.

[5] Moler, C.: Why isn't There a Parallel MATLAB. The MathWorks Newsletter, Spring 1995

[6] Blackford, L. S., Choi, J., Cleary, A., Petitet, A., Whaley, R. C., Demmel, J., Dhillon, I., Stanley, K., Dongarra, J., Hammarling, S., Henry, G., and Walker, D. 1996. ScaLAPACK: a portable linear algebra library for distributed memory computers – design issues and performance. In Proceedings of the 1996 ACM/IEEE Conference on Supercomputing (Cdrom) (Pittsburgh, Pennsylvania, United States, January 01 – 01, 1996). Conference on High Performance Networking and Computing. IEEE Computer Society, Washington, DC, 5.

[7] Dongarra, J., Du Croz, J., Duff,I.S., Hammarling, S. A set of Level 3 Basic Linear Algebra Subprograms, ACM Trans. Math. Soft., 16 (1990), pp. 1–17

[8] Anderson, E., Bai, Z., Dongarra, J., Greenbaum, A., McKenney, A., Du Croz, J., Hammerling, S., Demmel, J., Bischof, C., and Sorensen, D. 1990. LAPACK: a portable linear algebra library for high–performance computers. In Proceedings of the 1990 Conference on Supercomputing (New York, New York, United States). Conference on High Performance Networking and Computing. IEEE Computer Society Press, Los Alamitos, CA, 2–11.

[9] Dongarra J., Whaley, R C. A User's Guide to the BLACS 1.1 (1995)

[10] Pacheco, P. Parallel Programming with MPI (1996). Morgan Kaufmann Publishers Inc, pag. 217.

[11] Castellanos J. (2006). Determinación del tiempo de latencia Cluster Nimbus. Reporte Técnico PP2006–T2. Instituto de Matemáticas y Cálculo Aplicado, Facultad de Ingeniería. Universidad de Carabobo.

[12] Sharma G., Martin J. MATLAB®: A Language for Parallel Computing. Int J Parallel Prog (2009) 37:3–36

[13] Travinin Bliss, N., Kepner, J.: pMatlab parallel Matlab library. Int. J. High Perform. Comput. Appl. 21(3), 336–359 (2007).

[14] Kepner, J.:MatlabMPI. J. Parallel Distrib. Comput. 64(8), 997–1005 (2004).

[15] Husbands, P., Isbell, C.: Matlab*p: a tool for interactive supercomputing. In: The Ninth SIAM Conference on Parallel Processing for Scientific Computing (1999)

[16] Fernández J., Anguita M., Ros E. , Bernier J.L. SCE Toolboxes for the development of high–level parallel applications. In proceedings of the 6th International Conference Computational Science – ICCS 2006, Reading, United Kingdom, May 28–31, 2006.