EL DISEÑO Y ANÁLISIS DE ALGORITMOS PARALELOS EN LA UNIVERSIDAD DE LA LAGUNA

Rodríguez C., León C., Sande F., Roda J.L., González D., Almeida F., García F., García Forte. L.
Grupo de Paralelismo, Centro Superior de Informática, Universidad de La Laguna.
casiano@csi.ull.es

***Resumen**: Para lograr nuestros objetivos en la enseñanza de las asignaturas relacionadas con el Diseño y Análisis de Algoritmos Paralelos en la Universidad de La Laguna hemos desarrollado varias herramientas. La primera de las dos que se presentan aquí es un lenguaje de programación de alto nivel que permite la escritura y verificación de los numerosos algoritmos para el modelo PRAM. La segunda herramienta expande el modelo de librería de paso de mensajes (InmosC, PVM) con macros y funciones que permiten el anidamiento de sentencias paralelas. La eficiencia proporcionada es comparable con la obtenida usando el lenguaje nativo de la máquina paralela.*

1. INTRODUCCION.

La estructura de este trabajo es como sigue. En este párrafo introducimos los aspectos más relevantes del desarrollo de las asignaturas. Aunque las asignaturas relacionadas con el Diseño y Análisis de Algoritmos Paralelos en la Universidad de La Laguna son dos, *Programación en Paralelo I* y *Programación en Paralelo II*, nos ocuparemos con mas detalle, por razones de espacio, de la primera. Los dos siguientes párrafos se centran respectivamente en los capítulos 2 (El Modelo de Programación Paralela PRAM) y 5 (Como portar Algoritmos PRAM a Redes de Procesadores) y en las herramientas que hemos diseñado para la enseñanza de estos tópicos. El cuarto y quinto epígrafes contienen, a modo de apéndices, los programas de las dos asignaturas. Por ultimo, en las sección de referencias figuran los libros de texto, los de consulta y algunas direcciones en Internet relacionadas. Los libros utilizados como textos fundamentales en las dos asignaturas son el libro de Kumar *"Introduction to Parallel Computing. Design and Analysis of Algorithms."* [17] y el libro de Quinn. *"Parallel Computing. Theory and Practice"* [25]. La asignatura *Programación en Paralelo I* tiene 7,5 créditos, 4 teóricos y 3,5 prácticos o de laboratorio. Sus descriptores, tal como aparecen en el BOE número. 9 martes 11 Enero 1994 son:

Modelos PRAM. Técnicas Algorítmicas. Algoritmos para Sistemas Multiprocesadores. Complejidad Algorítmica Paralela.

En el segundo capítulo de *Programación en Paralelo I* se utilizan *Fork95* y *ll* como lenguajes orientados al modelo PRAM. En el capítulo 3 (Redes de Procesadores y el Modelo de

Programación por Paso de Mensajes) se introduce *occam* como primer lenguaje paralelo para programación basada en paso de mensajes [5], [14], [24], [28], [29]. Existe un compilador de *occam* para PCs debido a Michael Poole [23]. Un compilador de *occam* para estaciones de trabajo puede encontrarse en el servidor HENSA en el Computing Laboratory de la Universidad de Kent (Canterbury) [28]. En este mismo servidor pueden encontrarse manuales de *occam* [29]. Uno excelente es el desarrollado por la Universidad de Twente, del que existe una versión española debida al grupo de Manolo Capel en Granada [5]. También en el capítulo 3 se introduce la librería de paso de mensajes *InmosC*. Para los que no dispongan de una red de transputers y estén interesados en usarlo, una librería compatible con *inmosC* para estaciones de trabajo puede encontrarse en el citado servidor de HENSA. En nuestro laboratorio de computación paralela disponemos de varias redes de transputers. Las redes utilizan como máquinas anfitrión a ordenadores personales ejecutando *Linux*. Existe una versión del iserver para *Linux* de dominio público [22]. Es fácil, (¡nosotros lo hemos hecho!), portar las herramientas de inmos para transputers (*occam* Toolset, *InmosC* Toolset) bajo DOS a *Linux*. Los alumnos pueden así usar las redes de transputers desde cualquier punto. Se introduce también la que es posiblemente la más popular de las librerías de paso de mensajes: *PVM*. Esta librería, junto con *MPI*, será ampliamente utilizada en la continuación de la asignatura, Programación en Paralelo II. Para la presentación y las prácticas de los capítulos 4 (Operaciones Básicas de Comunicación) y 7 (Algoritmos Sistólicos) se utilizan *occam* e *inmosC*. En el capítulo 5 (Como Portar Algoritmos PRAM a Redes de Procesadores) se proporciona una metodología para determinar si un algoritmo PRAM es portable o no a una red de procesadores y para su traducción en caso de que lo sea. Se utilizan durante su exposición la herramienta *La Laguna C* [7] y el lenguaje funcional *NESL* [3]. También se considera en éste capítulo un lenguaje que, sin ser orientado al modelo PRAM, ofrece facilidades para la implantación de esta clase de algoritmos: *C**. *High Performance Fortran* es introducido en el capítulo 6 (Paralelismo Automático).

2. EL MODELO DE PROGRAMACION PARALELA PRAM.

El segundo capítulo del programa está dedicado al modelo PRAM. El modelo define una arquitectura, un método de análisis de complejidad y un modelo de programación. Aún cuando existen proyectos para su implantación como modelo de arquitectura [2], el modelo PRAM es a

```
TASK partition(left, right : INTEGER; VAR pivotPos: INTEGER);
VAR pivot, St : INTEGER;
BEGIN
  pivot := S[left];
  PARALLEL left..right DO
    IF S[name]>pivot THEN sum[name] := 1 ELSE sum[name] := 0;
  prefixSum(left,right);
  St := sum[right];
  pivotPos := right-St;
  PARALLEL (left+1)..right DO
    VAR pos : INTEGER;
    BEGIN
      IF S[name] > pivot THEN pos := right - St + sum[name]
      ELSE IF S[name] <= pivot THEN pos := name - sum[name] -1;
      S[pos] := S[name];
    END;
  S[pivotPos] := pivot;
END;
```

Figura 1: Procedimiento partition

menudo criticado de adolecer de falta de realismo. Lo mismo suele decirse de su reputación como modelo de análisis de complejidad, por considerar constante el coste de las comunicaciones. Sin embargo, la expresión de algoritmos paralelos cuando se usa como modelo de programación es de una gran elegancia. Es por eso que, en el desarrollo del capítulo, se hace especial énfasis en su aspecto de modelo de programación. Se introducen en ese momento tres lenguajes: Un lenguaje funcional, *NESL* [3], y otros dos fork95 [15] y *ll* [19] que están específicamente orientados a este modelo. El lenguaje *ll* (las dos letras "*l*" provienen de *la laguna)*, basado en Pascal, ha sido desarrollado por los autores en La Laguna como una herramienta para la enseñanza de algoritmos PRAM. Desde 1990 existe un compilador de *ll* para ordenadores personales IBM PC compatibles, lo que lo hace, hasta donde llega nuestro conocimiento, el primero en su género. Este compilador, de dominio público, puede obtenerse en la dirección que figura en la referencia [19]. Existen así mismo, versiones del compilador para redes de transputers. De acuerdo con la opinión de W. Rytter [26], [9], cualquier algoritmo PRAM puede expresarse en *ll*. La figura 2 muestra un algoritmo de ordenación paralelo tomado del libro de JáJá [13]. El algoritmo, que paraleliza el quicksort, tiene tres diferentes niveles anidados de paralelismo: el usado al nivel más exterior en el divide y vencerás, el utilizado en el procedimiento *partition* (figura 1) y el que ocurre en el procedimiento que calcula la suma de prefijos, *prefixsum*. Hemos elegido este ejemplo, no por ser el algoritmo especialmente eficiente, sino por el reto que supone su expresión de forma sencilla en un lenguaje paralelo.

```
TASK sort(left, right : INTEGER );
VAR pivotPos : INTEGER;
BEGIN
  IF left < right THEN
  BEGIN
    partition(left, right, pivotPos);
    PARALLEL DO
      sort(left, pivotPos-1) || sort(pivotPos+1, right);
  END; { IF left < right }
END; { sort }
```

Figura 2: Quicksort en ll

3. CÓMO PORTAR ALGORITMOS PRAM A REDES DE PROCESADORES

Muchos algoritmos PRAM pueden ser eficientemente ejecutados por máquinas paralelas distribuidas porque se pueden expresar en términos de un conjunto de primitivas de reducción/prefijos y primitivas de comunicación. Si un lenguaje o una librería provee de manera versátil y eficiente estas primitivas y además la facilidad de anidar paralelismo, dispondremos de una herramienta apropiada para portar algoritmos PRAM a máquinas paralelas distribuidas. Este es el hilo conductor en el que se basa el capítulo quinto del curso. En La Laguna hemos desarrollado un conjunto de librerías denominado *La Laguna C* o más abreviadamente, *llc*, que extiende una librería de paso de mensajes como *inmosC* o *PVM*. El entorno *La Laguna C* [7] dota al programador de los recursos descritos y en consecuencia de la capacidad de mezclar con libertad paralelismo y recursividad. Esta característica facilita la escritura de algoritmos paralelos del tipo Divide y Vencerás como el de la Figura 2. Para ver la eficiencia obtenida con esta herramienta, veamos los resultados obtenidos al ejecutar la codificación en *La Laguna C* del algoritmo de la figura 2, en el que, se susituido el procedimiento *partition* por el procedimiento *find* de partición equilibrada debido a Hoare. Para más detalles, consúltense la dirección URL en [7] y la referencia [10].

	2		4		8	
	BH	Llc	BH	llc	BH	llc
256 K	0.79	1.16	1.04	1.78	1.21	2.23
512 K	0.74	1.06	0.97	1.61	1.13	2.03
768 K	0.81	1.13	1.08	1.73	1.27	2.17

Tabla I: Aceleraciones: Algoritmos Quicksort paralelos de Hansen y *llc*.

Los resultados que se presentan en la tabla I se obtuvieron ejecutando los algoritmos en una máquina Parsys SN-1000 con transputers T-800 a 20Mhz y links operando a 10Mb/s. Se llevaron a cabo experimentos con hipercubos de 2, 4 y 8 transputers. Para cada tamaño de hipercubo y para cada algoritmo, la entrada de la tabla muestra la aceleración (cociente entre el tiempo secuencial y el tiempo paralelo) obtenida contra el mejor algoritmo secuencial. Todos los algoritmos fueron codificados utilizando Inmos C. Como entrada a los algoritmos se utilizaron vectores de tamaños 256K, 512K y 768K. La Tabla I compara la eficiencia del algoritmo *llc* de la figura 1 (columnas etiquetadas *llc*) con la obtenida programando en Inmos C un algoritmo paralelo quicksort específicamente diseñado para hipercubos por un experto en paralelismo, el inventor de los monitores y del Concurrent Pascal, profesor P. B. Hansen [10] (columnas etiquetadas BH).

Al interpretar los resultados, téngase en cuenta que, dado que el ancho de entrada/salida para el problema en una red de transputers es igual al tamaño n del vector a ordenar (los n datos deben salir y entrar por el procesador raíz), cualquier algoritmo de ordenación paralelo tiene su aceleración acotada superiormente por el logaritmo de n. Dado el alto valor de la constante de cómputo, para alcanzar eficiencias próximas a la unidad habría que trabajar con vectores de gran tamaño. En cualquier caso, los resultados prueban claramente la superioridad de la implementación en *La Laguna C*, a pesar de ser más sencilla, mas elegante y más fácil de programar.

4. APENDICE I: PROGRAMA DE PROGRAMACION EN PARALELO I

1. *INTRODUCCION A LA PROGRAMACION EN PARALELO*
   1. *EVOLUCION DE LOS COMPUTADORES DE ALTO RENDIMIENTO*
   2. *APLICACIONES QUE REQUIEREN PROCESAMIENTO PARALELO*
   3. *DEFINICIONES. TERMINOLOGIA BASICA*
   4. *CLASIFICACIONES DE ARQUITECTURAS PARALELAS*

2. *EL MODELO DE PROGRAMACION PARALELA PRAM*
   1. *EL MODELO PRAM DEFINICION Y VARIANTES*
   2. *REDUCCIONES Y PREFIJOS EN EL MODELO PRAM*
   3. *MEDIDAS DE RENDIMIENTO:*
      1. *TIEMPO, ACELERACION, EFICIENCIA, COSTE, ESCALABILIDAD*
      2. *EL CONCEPTO DE ALGORITMO OPTIMO*
      3. *ISOEFICIENCIA.*
   4. *LENGUAJES ORIENTADOS AL MODELO PRAM:*
      1. *EL LENGUAJE ll*
      2. *LOS LENGUAJES FORK y fork95*
   5. *ALGORITMOS PRAM*

6. *FUENTES DE INEFICIENCIA*
    1. *EL COSTE DE LA COMUNICACIÓN Y SINCRONIZACION*
    2. *EL EFECTO DE LA GRANULARIDAD Y LA ASIGNACION DE DATOS EN EL RENDIMIENTO. EL TEOREMA DE BRENT*
    3. *EL COSTE DEL DESEQULIBRIO DE LA CARGA DE TRABAJO*
    4. *COMPUTACION ADICIONAL INTRODUCIDA POR EL PARALELISMO*
7. *EL MODELO PRAM DE ANALISIS DE COMPLEJIDAD*
    1. *LA TESIS DE LA COMPUTACION PARALELA*
    2. *LA CLASE NC Y LA CLASE P-COMPLETA*

3. *REDES DE PROCESADORES Y EL MODELO DE PROGRAMACION POR PASO DE MENSAJES*

1. *REDES DE INTERCONEXION ESTATICAS Y DINAMICAS*
2. *TOPOLOGIAS:*
    HIPERCUBOS, MALLAS, ÁRBOLES, N-GRAFOS, MARIPOSAS, CUBOS CONECTADOS EN CICLOS, SHUFFLES, DE BRUIJIN...
3. *PARAMETROS QUE CARACTERIZAN LAS REDES:*
    DIAMETRO, GRADO,CONECTIVIDAD, ANCHO DE BISECCION.
4. *PARAMETROS DE RENDIMIENTO EN LAS COMUNICACIONES*
    1. *ANCHO DE BANDA, TASA DE TRANSFERENCIA.*
    2. *LATENCIA O TIEMPO DE ARRANQUE*
    3. *TIEMPO POR "HOP"*
5. *EL MODELO CSP*
    6. *PROCESADORES TRANSPUTER*
    7. *EL LENGUAJE occam*
6. *EL MODELO DE PROGRAMACION DE LIBRERÍAS CON PASO DE MENSAJES*
    1. *LA LIBRERÍA DE PASO DE MENSAJES InmosC*
    2. *LA LIBRERÍA PVM*
    3. *LA LIBRERÍA MPI*

4. *OPERACIONES BASICAS DE COMUNICACIÓN*

1. *MECANISMOS DE ENCAMINAMIENTO*
    1. *ENCAMINAMIENTO "STORE-AND-FORWARD"*
    2. *ENCAMINAMIENTO "CUT-THROUGH"*
2. *EMISIONES (BROADCASTING) UNO A TODOS*
3. *EMISIONES (BROADCASTING) TODOS A TODOS*
4. *REDUCCIONES*
5. *PREFIJOS*
6. *EMISIONES (BROADCASTING) UNO A TODOS PERSONALIZADO*
7. *EMISIONES (BROADCASTING) TODOS A TODOS PERSONALIZADO*
8. *PATRONES DE COMUNICACIÓN DE DESPLAZAMIENTO (SHIFTS)*
9. *PATRONES DE COMUNICACIÓN EN ARBOL (TREES)*
10. *PATRONES DE COMUNICACIÓN GENERAL*
11. *EMPOTRADOS (EMBEDDINGS) ENTRE REDES*

5. *COMO PORTAR ALGORITMOS PRAM A REDES DE PROCESADORES*

1. *GRANJAS Y "EMBARRASINGLY PARALLEL PROBLEMS"*
2. *ALGORITMOS ASCENDENTES*

3. *ALGORITMOS DIVIDE Y VENCERAS:*
    1. *EL LENGUAJE llc*
    2. *EL LENGUAJE FUNCIONAL NESL*
4. *REDES PARA LA IMPLANTACIÓN DE MEMORIA COMPARTIDA*
5. *EL LENGUAJE C**

6. ***PARALELISMO AUTOMATICO***
    1. *TIPOS DE DEPENDENCIAS EN BUCLES*
    2. *TECNICAS DE PARALELIZACION DE BUCLES*
    3. *EL LENGUAJE HPF*

7. ***ALGORITMOS SISTOLICOS***
    1. *EJEMPLOS*
    2. *EL PROBLEMA DE LA ASIGNACION DE ALGORITMOS SISTOLICOS A COMPUTADORES PARALELOS*

5. APENDICE II: PROGRAMA DE PROGRAMACION EN PARALELO II

1. ***ORDENACION***
    1. *REDES DE ORDENACION*
    2. *VARIANTES DEL QUICKSORT*
    3. *OTROS ALGORITMOS DE ORDENACION*

8. ***ALGORITMOS PARA MATRICES DENSAS***
    1. *OPERACIONES CON MATRICES*
    2. *RESOLUCION DE SISTEMAS*

9. ***GRAFOS***
    1. *ENVOLVENTE CONVEXA*
    2. *ARBOLES GENERADORES MINIMOS*
    3. *CAMINO MAS CORTO: ALGORITMO DE DIJKSTRA*
    4. *TODOS LOS CAMINOS MAS CORTOS*
    5. *CLAUSURA TRANSITIVA*
    6. *COMPONENTES CONEXAS*
    7. *ALGORITMOS PARA GRAFOS DISPERSOS*

10. ***TRANSFORMADA RAPIDA DE FOURIER***
    1. *ALGORITMO DE INTERCAMBIO BINARIO*
    2. *ALGORITMO TRASPUESTO*

11. ***RESOLUCION DE SISTEMAS DE ECUACIONES LINEALES CON MATRICES DISPERSAS***
    1. *ESQUEMAS DE ALMACENAMIENTO PARA MATRICES DISPERSAS*
    2. *OPERACIONES CON MATRICES DISPERSAS*
    3. *METODOS ITERATIVOS PARA SISTEMAS LINEALES DISPERSOS*
    4. *METODO DE ELEMENTOS FINITOS*
    5. *METODOS DIRECTOS PARA SISTEMAS LINEALES DISPERSOS*
    6. *METODOS MULTIMALLA*

6. PARALELIZACION DE LAS TECNICAS DE ENUMERACION Y BUSQUEDA

1. *ALGORITMOS DE BUSQUEDA Y RECORRIDO*
2. *PARALELIZACION DE LA BUSQUEDA PRIMERO PROFUNDO*
3. *PARALELIZACION DE LA BUSQUEDA PRIMERO MEJOR*
4. *PARALELIZACION DE LAS BUSQUEDAS ALFA-BETA*
5. *PROBLEMAS DE EQUILIBRADO DE LA CARGA*
6. *ANOMALIAS DE LA ACELERACION EN LOS ALGORITMOS DE BUSQUEDA*

7. LA PARALELIZACION DE LA PROGRAMACION DINAMICA

1. *FORMULACION DE LA PROGRAMACION DINAMICA A TRAVES DE LA TEORIA DE AUTOMATAS*
2. *MONADICOS MULTIETAPA: PARALELIZACIONES*
3. *MONADICOS LIBRES DE BUCLE: PARALELIZACIONES*
4. *POLIADICOS MULTIETAPA: PARALELIZACIONES*
5. *POLIADICOS LIBRES DE BUCLE: PARALELIZACIONES*

8. LA PARALELIZACION DE HEURISTICAS

1. *PARALELIZACION DE ALGORITMOS GENETICOS*
2. *PARALELIZACION DE LA TECNICA DE ENFRIAMIENTO SIMULADO*
3. *PARALELIZACION DE LA BUSQUEDA TABU*
4. *OTRAS TECNICAS HEURISTICAS EN PARALELO*

6. REFERENCIAS

[1] Akl. S. G. Diseño y análisis de Algoritmos Paralelos. Editorial Rama. 1992.

[2] Albohasan, J. Keller, J., Paul W.J. *On Physical Realizations of the Theoretical PRAM model.* Techn. Report 21/1990, Universität des Saarlandes. http://www-wjp.cs.uni-sb.de/sbpram/

[3] Blelloch. A nested Parallel Language: *NESL*. http://www.cs.cmu.edu/~scandal/*NESL*.html

[4] Cosnard M., Trystram D. *Parallel Algorithms and Architectures.* International Thompson Computer Press. 1995.

[5] Departamento de Lenguajes y Sistemas Informáticos. Universidad de Granada. Programación Paralela Avanzada Utilizando Transputers. Curso basado en un curso de la Universidad de Twente.

[6] Foster I., Foster I., Designing and Building Parallel Programs. Concepts and Tools for Parallel Software Engineering. Addison-Wesley. 1995.

[7] García Forte, L. *La Laguna C.* Proyecto de Ingeniería, Departamento de Estadística Investigación Operativa y Computación. http://ocre.csi.ull.es/pcgull/html/casiano/llc/llc.html

[8] Geist A., Benguelin, A., Dongarra J., Jiang W., Manchek R., Sunderan V., *PVM: Parallel Virtual Machine. A User's guide and Tutorial for networked Parallel Computing.* The MIT Press.

[9] Gibbons A., Rytter W. *Efficient Parallel Algorithms*. Cambridge University press. 1988.

[10] Hansen P. B., *Studies in Computational Science. Parallel Programming Paradigms*. Prentice Hall. 1995.

[11] Inmos Limited. *ANSI C toolset reference manual.* Inmos Corp.

[12] Inmos Limited. *ANSI C user's guide reference manual.* Inmos Corp.Kumar V., Grama A., Gupta A.

[13] JáJá J. *An Introduction to Parallel Algorithms.* Addison-Wesley. 1992.

[14] Jones Geraint, Goldsmith M. *Programming in occam 2.* Prentice Hall. 1988.
[15] Keßler C.W., C., Seidl,H. *Integrating Synchronous and Asynchronous Paradigms. The fork95 Parallel Programming Language.* Proc. Of the MPPM-95 Conference on Massively Parallel Programming Models. IEEE CS Press. http://www.informatik.uni-trier.de/~kessler/fork95.html
[16] Keßler C.W., *Language and Library Support for Practical PRAM Programming.* Proc. Fifth Euromicro Workshop on Parallel and Distributed Processing. IEEE Computer Society Press. 1997.
[17] Kumar V., Grama A., Gupta A., Karpys G. *Introduction to Parallel Computing. Design and Analysis of Algorithms.* Benjamin/Cummings Pub. Co.
[18] Leighton, F.T. *Introduction to Parallel Algorithms and Architectures. Arrays. Trees. Hypercubes.* Morgan Kaufman Pub. 1992.
[19] León, C., Sande F., Rodríguez C., García F., *A PRAM Oriented Language.* Proc. Of the EuroMicro Workshop on Parallel and Distributed Processing. IEEE CS Press, 1995, 182-191. http://ocre.csi.ull.es/pcgull/html/casiano/ll/ll.html
[20] Lewis, T.G. *Introduction to Parallel Computing.* Prentice Hall. 1992.
[21] Moldovan, D.I. *Parallel Processing. From Applications to Systems.* Morgan Kaufman Pub. 1993.
[22] Niemann, Christoph. Device-driver for using transputers in a *LINUX* box. (niemann@rubdv15.etdv.ruhr-uni-bochum.de)
[23] Poole, M. *occam compiler for PCs.* http://www.hensa.ac.uk/parallel/occam/compilers/inmos/oc/msdos/index.html
[24] Pountain D. *A Tutorial Introduction to occam Programming.* Inmos Corp. England. 1986. http://www.comlab.ox.ac.uk/archive/occam.html
[25] Quinn, M.J. *Parallel Computing. Theory and Practice.* McGraw-Hill. 1994.
[26] Rytter W. Conversación Personal. La Laguna, 1991.
[27] Thinking Machines Co. *C* Programming Guide.* Cambridge, MA. 1990.
[28] Welch. Peter. *occam* for All Project. http://www.hensa.ac.uk/parallel/occam/projects/occam-for-all/index.html.
[29] Welch, Peter, *occam* documentation and manuals. http://www.hensa.ac.uk/parallel/occam/compilers/inmos/oc/docs/index.html