# ESTADO DEL ARTE EN LAS IMPLEMENTACIONES PARALELAS DE DIVIDE Y VENCERÁS

**Resumen / *Abstract***

Divide y Vencerás, es una técnica de diseño de algoritmos ampliamente utilizada en la solución de problemas. Su implementación en paralelo resulta natural. En este trabajo se realiza una revisión bibliográfica acerca del desarrollo de algoritmos paralelos con este paradigma. En ambientes paralelos heterogéneos se mejoran las prestaciones, si en el diseño de algoritmos se tienen en cuenta las características de los procesadores utilizados y de las comunicaciones entre ellos. Los trabajos analizados desarrollan implementaciones paralelas de algoritmos basados en "Divide y Vencerás" solamente en sistemas homogéneos. Resulta imprescindible el estudio y desarrollo de este paradigma en ambientes heterogéneos.

*Divide-and-Conquer is known as a widely used algorithm design technique for problem solving. It becomes natural to implement this technique in parallel. The main objective of this work is to study the State-of-the-art of parallel Divide-and-Conquer algorithms design. In heterogeneous parallel environments the performance of the algorithms is importantly improved, if the characteristics of the used processors and communications between them are considered in the algorithm design. The works analysed, implement parallel Divide-and-Conquer algorithms only on homogeneous systems. Therefore, it results essential the study and development of such algorithms in heterogeneous systems.*

**Palabras clave / *Key words***

Divide y Vencerás, algoritmos paralelos, esqueletos computacionales, sistemas paralelos homogéneos, sistemas paralelos heterogéneos

*Divide and Conquer, parallel algorithms, computational skeletons, parallel homogeneous systems, parallel heterogeneous systems*

***Ernesto R. Carbonell Rigores,*** *Licenciado en Matemática, Centro de Estudios de Ingeniería de Sistemas (CEIS), Instituto Superior Politécnico José Antonio Echeverría Cujae, Ciudad de La Habana, Cuba*
e-mail:ernesto@ceis.cujae.edu.cu

***Antonio M. Vidal Maciá,*** *Licenciado en Ciencias Físicas, Doctor en Infomática,Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, España*
e-mail:avidad@dsic.upc.es

***Exiquio C. Leyva Pérez,*** *Ingeniero Químico, Doctor en Ciencias Técnicas, Profesor Titular, CEIS, Instituto Superior Politécnico José Antonio Echeverría Cujae, Ciudad de La Habana, Cuba*
e-mail:exiquio@ceis.cujae.edu.cu

*Recibido:octubre del 2005*
*Aprobado:diciembre del 2005*

## INTRODUCCIÓN

Divide y Vencerás (D yV) es un método utilizado en el diseño de algoritmos desde los primeros días de la programación de computadoras. Muchos de los mejores algoritmos y de los más ampliamente utilizados se basan en este método de solución de problemas, como la búsqueda binaria y los método de ordenamiento QuickSort de Hoare y MergeSort,[1,2] entre otros muchos. Esta técnica se considera un paradigma en el diseño de algoritmos y ha sido empleada para la solución de problemas simples, complejos, generales y especializados. Se reconoce como altamente apropiado para el desarrollo de implementaciones paralelas, dado que los subproblemas generados pueden ser resueltos independientemente y, por tanto, en paralelo[2] y se emplea como una filosofía general de solución de problemas, por lo que no se utiliza solo en el argot informático, sino en muchos otros ámbitos.[3]

Desde el punto de vista de la programación paralela, el paradigma DyV es reconocido como una de las llamadas **técnicas de descomposición de problemas**, específicamente una técnica de descomposición recursiva, a través de la cual los cálculos a realizar en la solución de un problema se dividen en un conjunto de tareas que serán ejecutadas de manera concurrente, según lo definido por un grafo de dependencias.[4] Por otro lado, esta técnica resulta útil para descubrir oportunidades de implementación de cálculos concurrentes en la solución de problemas complejos.[5]

## DIVIDE Y VENCERÁS

La solución de un problema, de un tamaño determinado, mediante este paradigma consiste en dividir (recursivamente) dicho problema (**etapa de división**) en un conjunto de subproblemas del mismo tipo, pero de tamaño menor. Cada subproblema obtenido en el proceso de división puede ser, a su vez, subdividido siguiendo el mismo criterio. Este proceso se lleva a cabo hasta que el subproblema actual sea indivisible (**caso elemental** o **caso base**) o hasta que este pueda ser resuelto por un método directo (**etapa de resolución**). La convergencia hacia el caso base está garantizada por la exigencia de que el tamaño de los subproblemas debe ser menor que el del problema que le da origen. Las soluciones obtenidas en la etapa de resolución se combinan convenientemente (**etapa de combinación**) para construir la solución del problema original. En la referencia 3 se presenta un estudio de este paradigma y se muestra un esqueleto para la implementación de algoritmos que siguen esta técnica, así cómo se discuten aspectos relacionados con la complejidad algorítmica de estos.

## ALGUNAS APLICACIONES REPORTADAS

DyV se ha utilizado en la solución de muchos problemas de diversas ramas como las Ciencias de la Computación, el Álgebra Lineal, las Matemáticas, la Geometría, el Procesamiento de Señales. la Biología Molecular Computacional,[6,7] en el procesamiento de información del ADN.[8] en espectroscopia por resonancia magnética nuclear para la determinación estructural de proteínas,[9] en el análisis y modelado de proteínas,[10-15] en algoritmos para simulación,[16] en la implementación de algoritmos de descomposición en valores singulares de matrices,[17] en la especificación y desarrollo de programas mediante cálculos abstractos,[18] como estrategia en la paralelización de un algoritmo de votación[19] y más recientemente en los trabajos de análisis del genoma humano.[20,21]

## IMPLEMENTACIONES PARALELAS

DyV, por naturaleza, puede implementarse en máquinas paralelas. Muchos algoritmos basados en este paradigma generan un número muy grande de subproblemas en aplicaciones reales típicas y, por tanto, pueden implementarse potencialmente con un alto grado de paralelismo.[2] Tradicionalmente, el paralelismo en estos algoritmos se logra mediante la solución paralela de los subproblemas en los niveles superiores de recursividad. Sin embargo, el paralelismo obtenido por esta técnica no escala bien, debido a la composición secuencial de las etapas de división y combinación. En la referncia 22 se propone una solución a este inconveniente, mediante: La división del problema en un número variable de subproblemas, utilizando un algoritmo paralelo escalable; la solución de los subproblemas en paralelo, usando un algoritmo secuencial eficiente y la combinación de las soluciones de los subproblemas, utilizando un algoritmo paralelo escalable. Con ello, se logra que las operaciones o etapas de división y combinación se desarrollen con algoritmos paralelos escalables, que haya un solo nivel de división del problema y que el número de subproblemas pueda ser variado en dependencia del tamaño del problema y de la cantidad de procesadores que intervienen.

En referencia 23, se presenta una formulación general de paralelismo de datos para una clase de problemas basados en el paradigma DyV. Se exponen algunas ideas que sirven de base a la paralelización de los algoritmos basados en DyV para resolver dichos problemas, y se muestra un modelo para el cálculo del tiempo total de ejecución y de comunicaciones de los programas paralelos implementados.

En el campo del Álgebra Lineal, por ejemplo, se han reportado implementaciones paralelas de algoritmos DyV en la solución del problema del cálculo de los valores propios de matrices tridiagonales simétricas,[24,25] en la solución de sistemas de ecuaciones lineales banda,[26] así como en la inversión de matrices triangulares.[27]

La Ciencia de la Computación también se ha beneficiado con la implementación de algoritmos del tipo DyV. Se reporta, por ejemplo, la implementación secuencial y paralela de un algoritmo basado en este paradigma para la identificación de componentes fuertemente conexas en grafos dirigidos.[28] Estas implementaciones se introducen como alternativa al uso de la búsqueda primero en profundidad propuesta en trabajos anteriores y que presenta la dificultad de su paralelización en la solución de este problema cuando el tamaño del grafo es suficientemente grande.

Otro trabajo propone[29] un esquema de implementación paralelo SPMD (del inglés *Single Program Multiple Data*) para especificaciones de tipo DyV en la programación funcional, basado en transformaciones formales de programas. Partiendo de una determinada clase de especificaciones, representadas mediante expresiones funcionales, se construye formalmente un esquema común de implementación paralela que se aplica a cada miembro de esa clase y transformando, mediante reglas de derivación, esas especificaciones a expresiones con mejores propiedades. En trabajos posteriores,[30,31] se propone un método que explota el paralelismo inherente a DyV en funciones sobre listas. El método se desarrolla en dos etapas: la extracción del paralelismo y la implementación del paralelismo. La primera se

logra encontrando una representación homomórfica de una función dada. La segunda se logra derivando un programa paralelo eficiente que compute dicha función, haciendo uso del esquema común de implementación paralela presentado.[29] En la referencia 32 se presenta la **transformada rápida de Fourier** (FFT) multidimensional, como caso de estudio de la derivación formal de programas propuesta en la referencia 29 y partiendo de las especificaciones de algoritmos DyV.

DyV ha servido, a su vez, como base para el diseño, implementación y verificación de nuevos lenguajes de programación. Un ejemplo de ello lo constituye Multilisp, que extiende el lenguaje de programación Scheme, dialecto de Lisp, con un paradigma de programación paralela. En la referencia 33 se propone un método de distribución de tareas basado en un protocolo de pases de mensajes como alternativa que mejora la creación "perezosa" de tareas en una implementación anterior de Multilisp, posibilitando la ejecución paralela eficiente de programas del tipo DyV. Un trabajo posterior[34] propone la, denominada llamada "perezosa" a **procedimiento remoto**, especialmente diseñada para la implementación de programas paralelos del tipo DyV que permitió la implementación del lenguaje ParSubC (del inglés *Parallel Subset of C*), variante paralela de C que posibilita el procesamiento simbólico paralelo.

En otro trabajo[35] se describe el diseño e implementación del lenguaje de programación MPP Haskell, que constituye una extensión del lenguaje Haskell para el desarrollo de programas paralelos, y se utiliza una versión paralela de DyV para verificar el desempeño de este lenguaje.

Asimismo, se propone un modelo paralelo[36] para grupos de procesadores, denominado *team-parallel model*, enfocado al desarrollo de algoritmos DyV irregulares, o sea, aquellos que operan sobre estructuras de datos no uniformes, como los grafos y las matrices escasas, en arquitecturas paralelas. En este trabajo se tratan los algoritmos del tipo DyV en atención a varios criterios. Se desarrolla el sistema *Machiavelli* y una extensión del lenguaje C, que posibilita el paralelismo de datos y la recursividad paralela, haciendo uso de MPI (del inglés *Message Passing Interface*) para las comunicaciones de mensajes entre procesadores.

Otros trabajos[37,38] en el campo de la programación funcional describen la obtención de programas en C a partir de un esqueleto general DyV especificado en Haskell. Los programas obtenidos contienen ciclos anidados con paralelismo de datos y la traducción, del esqueleto general DyV en Haskell a dichos programas C, se lleva a cabo a través de una serie de refinamientos sucesivos de ese esqueleto. La especificación en Haskell se define en el lenguaje HDC (del inglés *higher-order divide-and-conquer*), que es un subconjunto del Haskell, originalmente diseñado para la paralelización de recursividad del tipo DyV. El compilador propuesto traduce un programa fuente en HDC a código objeto en C.

En la referencia 39 se presenta el lenguaje de programación Fork95, que es un rediseño del lenguaje Fork, basado en ANSI C y proporciona herramientas adicionales para crear procesos paralelos, dividir jerárquicamente grupos de procesadores en subgrupos y manipular subespacios compartidos y privados de direcciones de memoria. Entre los paradigmas de programación que soporta este lenguaje se encuentra DyV.

Asimismo, en la referencia 40, se refiere un algoritmo basado en DyV para la redistribución de datos irregulares, como pueden ser los bloques de arreglos de tamaño irregular, en un mecanismo de pases de mensajes entre procesadores.

Otro trabajo [41] presenta un esqueleto paralelo distribuido para el paradigma DyV en C++. El programador debe especificar el tipo del problema, del subproblema y de la solución y programar la aplicación y el esqueleto se encarga de dar solución al problema planteado haciendo uso de estas especificaciones.

Los autores de este trabajo han desarrollado un esqueleto paralelo para DyV basado en una arquitectura maestro/esclavo, en la que la etapa de división del problema original se lleva a cabo en el procesador maestro, que distribuye los subproblemas generados en esta etapa a los esclavos. Estos, a su vez, los resuelven y envían al maestro las soluciones obtenidas para su combinación en el propio maestro.

## IMPLEMENTACIÓN PARALELA EN SISTEMAS HETEROGÉNEOS

Explotar al máximo la potencia de un sistema paralelo heterogéneo es una tarea compleja que, con frecuencia, requiere de estrategias sofisticadas de administración de recursos. En la práctica es difícil predecir el desempeño de una estrategia particular de administración de recursos en un sistema compuesto por componentes que difieren en cuanto a funcionalidades y desempeño. Para maximizar el desempeño total de una aplicación en tales sistemas se hace necesario el uso de un mecanismo más estructurado que permita especificar, de manera explícita, los requerimientos de recursos, así como cuantificar y calcular el desempeño de una estrategia de administración de recursos.[42]

Ninguno de los trabajos revisados sobre el tema hace referencia a implementaciones de DyV en sistemas paralelos heterogéneos. Resultaría, por tanto, imprescindible y novedoso el estudio y desarrollo de implementaciones paralelas de DyV en este tipo de sistemas.

Actualmente, los autores trabajan en el desarrollo de un esqueleto computacional para DyV en sistemas paralelos heterogéneos.

## CONCLUSIONES

Se realiza un estudio bibliográfico de la técnica de diseño de algoritmos Divide y Vencerás. Se reportan innumerables aplicaciones secuenciales y paralelas de este paradigma en la solución de muchos problemas de diversas ramas de la ciencia y la tecnología.

Por su naturaleza, los algoritmos basados en DyV pueden ser paralelizables, dado que la solución de los subproblemas

generados en el proceso recursivo de división de esta técnica, puede llevarse a cabo de manera independiente y, por lo tanto, en diferentes procesadores.

Se han reportado muchas implementaciones paralelas de DyV en la solución de problemas, pero ninguno de los trabajos revisados refiere implementaciones paralelas heterogéneas. Resultaría imprescindible y novedoso implementar DyV en sistemas paralelos heterogéneos.

## REFERENCIAS


1. **AHO, A.V.; J. E. HOPCROFT AND J. D. ULLMAN:** *Data Structures and Algorithms*, Addison-Wesley, Reading, Massachusetts, 1983.
2. **KRONSJO, L.:** *Advances in Parallel Algorithms*, Edited by Lydia Kronsjo and Dean Shumsheruddin, Oxford, Blackwell Scientific, 1992.
3. **GUEREQUETA, R. Y A. VALLECILLO:** *Técnicas de diseño de algoritmos*, Servicio de Publicaciones de la Universidad de Málaga, 1998.
4. **GRAMA, A.; A. GUPTA; G. KARYPIS AND V. KUMAR:** *Introduction to Parallel Computing*, Second Edition, Addison-Wessley, 2003,
5. **FOSTER, IAN:** *Designing and Building Parallel Programs*, Addison-Wesley, 1995.
6. **GRICE, J. ALICIA; RICHARD HUGHEY AND DON SPECK:** "Reduced Space Sequence Alignment" in *Comput. Appl. Biosci*, 13: 45 - 53. Oxford University Press, February 1997.
7. **LU, CHIN LUNG AND YEN PIN HUANG:** "A Memory-Efficient Algorithm for Multiple Sequence Alignment with Constraints", in *Bioinformatics*, 21: 20 - 30, Oxford University Press, Jan 2005.
8. **OTU, HASAN H. AND KHALID SAYOOD:** "A Divide-and-Conquer Approach to Fragment Assembly" in *Bioinformatics*, 19: 22 - 29, Oxford University Press, Jan., 2003.
9. **RIGDEN, DANIEL J.:** "Use of Covariance Analysis for the Prediction of Structural Domain Boundaries from Multiple Protein Sequence Alignments", in *Protein Eng*., 15: 65 - 77, Oxford University Press, Feb., 2002.
10. **CALLAND, PIERRE-YVES:** "On the Structural Complexity of a Protein" in *Protein Eng*., 16: 79 - 86. Oxford University Press, Feb 2003.
11. **HICKSON, ROBERT E.; CHRIS SIMON AND SOREN W. PERREY:** "The Performance of Several Multiple-Sequence Alignment Programs in Relation to Secondary-Structure Features for an rRNA Sequence" in *Mol. Biol. Evol*. 17: 530 - 539. Oxford University Press. Apr., 2000.
12. **LI, HAIQUAN AND JINYAN LI:** "Discovery of Stable and Significant Binding Motif Pairs from PDB Complexes and Protein Interaction Datasets" in *Bioinformatics*, 21:314-324, Oxford University Press, Feb., 2005.
13. **SMEJKAL, GARY B. AND ALEXANDER LAZAREV:** "Solution Phase Isoelectric Fractionation in the Multi-Compartment Electrolyser: A Divide and Conquer Strategy for the Analysis of Complex Proteomes" in *Brief Funct Genomic Proteomic*, 4: 76 - 81. Oxford University Press, 2005.
14. **STOYE, JENS; VINCENT MOULTON AND ANDREAS W. M. DRESS:** "DCA: An Efficient Implementation of the Divide-and-Conquer Approach to Simultaneous Multiple Sequence Alignment" in *Comput. Appl. Biosci*., 13: 625 - 626, Oxford University Press, December, 1997.
15. **TOSATTO, SILVIO C.E.** *et al.*: "A Divide and Conquer Approach to Fast loop Modelling" in *Protein Engineering*, Oxford University Press. Vol. 15, No. 4, pp. 279-286, April, 2002.
16. **MATUSUMAE, SUSUMU:** "An Efficient Scaling-Simulation Algorithm of Reconfigurable Meshes by Meshes with Statically Partitioned Buses" in *IEICE Trans D: Information*, E88-D: 82-88. Oxford University Press, January, 2005.
17. **DRMAC, Z.:** "A Posteriori Computation of the Singular Vectors in a Preconditioned Jacobi SVD Algorithm", in *IMA J Numer Anal*, 19: 191 - 213. Oxford University Press, April, 1999.
18. **FRIAS, M. F.; A. M. HAEBERER AND G. A. BAUM:** "Representability and Program Construction within Fork Algebras" in *Logic Jnl IGPL*, 6: 227 - 257, Oxford University Press, March,1998.
19. **PARHAMI, B.:** "Parallel Threshold Voting" in *The Computer Journal*, 39: 692 - 700, Oxford University Press. 1996.
20. **KIM, D.** *et al.***:** "PROSPECT II: Protein Structure Prediction Program for Genome-Scale Applications" in *Protein Eng*., 16: 641 - 650, Sep., 2003.
21. **ZHANG, KUI; FENGZHU SUN AND HONGYU ZHAO:** "HAPLORE: a Program for Haplotype Reconstruction in General Pedigrees Without Recombination," in *Bioinformatics*, Oxford University Press, 21: 90 - 103, Jan, 2005.
22. **THORNLEY, J.:** *Performance of a Class of Highly-Parallel Divide-and-Conquer Algorithms, Technical Report*, California Institute of Technology, [CaltechCSTR:1995.cs-tr-95-10].
23. **AMOR, M.** *et al*.**:** "A Data-Parallel Formulation for Divide and Conquer Algorithms" in *The Computer Journal*; 44: 303 - 320, Oxford University Press, 2001.
24. **TISSEUR, FRANÇOISE AND JACK DONGARRA:** " A Parallel Divide and Conquer Algorithm For The Symmetric Eigenvalue Problem On Distributed Memory Architectures" in *SIAM Journal on Scientific Computing*, Vol. 20, No.6, pp. 2223-2236, 1999.

**25. VIDAL MACIÁ, ANTONIO M. Y J. M. BADÍA CONTELLES:** "Cálculo de los valores propios de matrices tridiagonales simétricas mediante la iteración de Laguerre" en *Métodos numéricos para cálculo y diseño en ingeniería, Revista Internacional*, Vol. 16, No. 2, pp. 227-250, 2000.

**26. ARBENZ, P. AND W. GANDER:** "A Survey of Direct Parallel Algorithms" for Banded Linear Systems" in *Technical Report 221*, Departement Informatik, ETH Zurich, 1994.

**27. NASRI, WAHID AND ZAHER MAHJOUB:** "Generalizing the Implementation of an Optimal Parallel Recursive Algorithm for Triangular Matrix Inversion", Cited by CiteSeer.IST Scientific Literature Digital Library, 2000.

**28. FLEISCHER, LISA K.; BRUCE, HENDRICKSON AND ALI PINAR:** "A Divide-And-Conquer Algorithm for Identifying Strongly Connected Components", Submitted to SIAM J. Discrete Math.

**29. GORLATCH, S. Y C. LENGAUER:** "Parallelization of Divide-and-Conquer in the Bird-Meertens Formalism", in *Formal Aspects of Computing*, 3, 1995.

**30. GORLATCH, S.:** "Systematic extraction and Implementation of Divide-and-Conquer Parallelism" en H. Kuchen and S.D. Swierstra, editors, Eighth International Symposium on Programming Languages, Implementations, Logics, and Programs, PLILP'96, Vol. 1140 of LNCS, pp. 274-288, September 1996.

**31. GORLATCH, S.:** "Extracting and Implementing List Homomorphisms in Parallel Programming Development" in *Science of Computer Programming*, Vol. 33, No. 1, pp. 1-27, 1999.

**32. GORLATCH, S. Y H. BISCHOF:** "Formal Derivation of Divide-and-Conquer Programs: A Case Study in the Multidimensional FFT's" in *Formal Methods for Parallel Programming: Theory and Applications. Workshop at IPPS'97*, D. Mery editor, pp. 80 - 94, 1997.

**33. FEELEY, MARC:** "A Message Passing Implementation of Lazy Task Creation" in *Parallel Symbolic Computing: Languages, Systems, and Applications*, Vol. 748 of LNCS, pp. 94-107, Springer-Verlag, 1993.

**34. FEELEY, MARC:** "Lazy Remote Procedure Call and Its implementation in a Parallel Variant of C", in *Proceedings of International Workshop on Parallel Symbolic Languages and Systems*, No. 1068 in Lecture Notes in Computer Science, pp. 3-21, Springer-Verlag, 1995.

**35. DAVIS K.:** "MPP Parallel Haskell" in *IFL'96*, International Workshop on the Implementation of Functional Languages, pp. 49-54, Bad Godesberg, Germany, September 1996. Draft Proceedings.

**36. HARDWICK, JONATHAN C.:** "Practical Parallel Divide-and-Conquer Algorithms", Ph.D. Thesis, School of Computer Science, 154 pp., Carnegie Mellon University, December, 1997.

**37. HERRMANN, C. A. AND C. LENGAUER:** "Parallelization of Divide-and-Cconquer by Translation to Nested Loops" in *J. of Functional Programming*, 9 (3), pp. 279 -310, 1999.

**38. ———:** "HDC: A Higher-Order Language for Divide-and-Conquer" in *Parallel Processing Letters* 10 (2/3), pp. 239-250, 2000.

**39. KELER, CHRISTOPH W. AND H. SEIDL:** *Integrating Synchronous and Asynchronous Paradigms: The Fork95 Parallel Programming Language, Technical Report 95-05*, FB IV Informatik der Universitat Trier, 1995.

**40. WANG, HUI; MINYI GUO AND DAMING WEI:** "Message Scheduling for Irregular Data Redistribution in Parallelizing Compilers", in *IEICE Trans. D: Information*, E89-D: 418 - 424. Oxford University Press, February, 2006.

**41. GONZÁLEZ GONZÁLEZ, JUAN R. Y COROMOTO LEÓN HERNÁNDEZ:** "Esqueletos paralelos distribuidos. Paradigmas de ramificación y acotación y divide y vencerás", en *Documento de Trabajo Interno DT-2003-07*, Esc. Técnica Superior de Ingeniería Informática, Universidad de La Laguna, España, 2003.

**42. AU, P. *et al*.:** *Co-ordinating Heterogeneous Parallel Computatio,*L. Bouge, P. Fraigniaud, A. Mignotte, and Y. Robert, editors, Europar 96, pp. 601- 614. Springer-Verlag, 1996.