Fundamentos y Aplicaciones de Procesamiento Distribuido y Paralelo

Marcelo Naiouf, Armando E. De Giusti, Laura C. De Giusti, Franco Chichizola, Adrian Pousa, Waldo Hasperué, Victoria Sanz, Fabiana Leibovich
Instituto de Investigación en Informática LIDI (III-LIDI) - Facultad de Informática - UNLP
{mnaiouf, degiusti, ldgiusti, francoch, apousa, whasperue, vsanz, fleibovich}@lidi.info.unlp.edu.ar

CONTEXTO

La línea de Investigación que se presenta es parte del Proyecto "Procesamiento paralelo y distribuido. Fundamentos y aplicaciones en Sistemas Inteligentes y Tratamiento de imágenes y video." del Instituto de Investigación en Informática LIDI acreditado por la UNLP, y de proyectos apoyados por Cyted, CIC, Agencia, IBM, Fundación YPF y Telefónica.
Existe cooperación con Universidades de Argentina, Latinoamérica y Europa a través de los proyectos "CyTEDGrid. Tecnología grid como motor de desarrollo regional" acreditado por CyTED, "Formación en Computación Avanzada" y "FRIVIG: Formación de Recursos Humanos e Investigación en el Area de Visión por Computador e Informática Gráfica" acreditados por AECID, e "IberoTIC. Red Iberoamericana de Ingeniería y Tecnologías de la Información" subsidiado por la OEI (Organización de Estados Iberoamericanos).
También el III-LIDI forma parte (desde la Facultad de Informática) de la iniciativa LAGrid (LatinAmerican Grid) de IBM y el proyecto EELA2 (E-infrastructure shared between Europe and Latin America).

RESUMEN

El eje central de esta línea de I/D lo constituye el estudio de los temas de procesamiento paralelo y distribuido, tanto en lo referente a los fundamentos como a las aplicaciones. Esto incluye los problemas de software asociados con la construcción, evaluación y optimización de algoritmos concurrentes, paralelos y distribuidos sobre arquitecturas multiprocesador.

Los temas de interés abarcan aspectos de fundamentos tales como el diseño y desarrollo de algoritmos paralelos sobre diferentes arquitecturas multiprocesador y plataformas de software, paradigmas paralelos, modelos de representación de aplicaciones, *mapping* de procesos a procesadores, métricas, escalabilidad, balance de carga, predicción y evaluación de performance. Las arquitecturas de soporte a utilizar pueden ser homogéneas o heterogéneas, incluyendo multicore, clusters, multiclusters y grid

Se trabaja en la concepción de aplicaciones paralelas numéricas y no numéricas sobre grandes volúmenes de datos y/o que requieren cómputo intensivo, y en el desarrollo de laboratorios remotos para el acceso transparente a recursos de cómputo paralelo.

Palabras clave: *Sistemas paralelos. Algoritmos paralelos y distribuidos. Clusters. Multicluster. Grid. Multicore. Balance de carga. Evaluación de performance.*

1. INTRODUCCION

Por numerosos motivos, el procesamiento paralelo y distribuido se ha convertido en un área de gran importancia e interés dentro de la Ciencia de la Computación, produciendo profundas transformaciones en las líneas de I/D [1][2][3][4][5][8][9][10][11].

Interesa realizar investigación en la especificación, transformación, optimización y evaluación de algoritmos distribuidos y paralelos. Esto incluye el diseño y desarrollo de sistemas paralelos, la transformación de algoritmos secuenciales en paralelos, y las métricas de evaluación de performance sobre distintas plataformas de soporte (hardware y software). Más allá de las mejoras constantes en las arquitecturas físicas de soporte, uno de los mayores desafíos se centra en cómo aprovechar al máximo la potencia de las mismas. En esta línea de I/D la mayor importancia está dada en los *algoritmos paralelos*, y en los métodos utilizados para su construcción y análisis [6][7][26].

En los últimos años, uno de los cambios de mayor impacto ha sido la utilización (de manera masiva) de procesadores con más de un núcleo (multinúcleo o multicore). Esto ha producido plataformas de soporte distribuidas híbridas, con memoria compartida y distribuida, llevando a la necesidad de desarrollar sistemas operativos, lenguajes de programación y fundamentalmente algoritmos que utilicen adecuadamente la arquitectura.

Si bien utilizar múltiples procesadores para resolver problemas, en general de complejidad creciente y para obtener resultados en menor tiempo, resulta un concepto intuitivo, los fundamentos subyacentes a los mecanismos y soportes de paralelización presentan numerosas variantes. Pueden encontrarse diferentes formulaciones paralelas para un problema, y el rendimiento y la eficiencia de cada una dependerá del algoritmo y de la arquitectura utilizada.

1.1. Algoritmos Paralelos y Distribuidos y Arquitecturas Multiprocesador

La creación de algoritmos paralelos y distribuidos sobre arquitecturas multiprocesador, o la

transformación de un algoritmo secuencial en paralelo, no es un proceso directo. El "costo" del paralelismo puede ser alto en términos del esfuerzo de programación [8][9][10][11]. El manejo de la concurrencia adquiere un rol central en el desarrollo de aplicaciones paralelas. Los pasos básicos para diseñar aplicaciones paralelas incluyen el particionamiento, la comunicación, la aglomeración y el mapeo de procesos a procesadores.

Un *sistema paralelo* (SP) es la combinación de un algoritmo paralelo y la máquina sobre la cual éste se ejecuta; ambos factores poseen numerosas variantes y de un adecuado "matching" entre ellos depende el éxito de la solución [16][17].

Respecto de los algoritmos, pueden ser escritos utilizando diferentes paradigmas (cliente/servidor, pipeline, dividir y conquistar, SPMD); otra forma de clasificarlos es por la utilización de paralelismo de datos o de control.

Las arquitecturas para procesamiento paralelo y distribuido han evolucionado, y en la actualidad las redes de computadoras constituyen una plataforma de cómputo paralelo muy utilizada por sus ventajas en términos de la relación costo/rendimiento. La noción de sistema distribuido como máquina paralela es común a las denominaciones redes de computadoras, NOW, redes SMP, clusters, multiclusters y grid. En estos casos, se deben identificar las capacidades de procesamiento, interconexión, sincronización y escalabilidad.

El concepto de multicluster es una generalización que permite que redes dedicadas a una aplicación paralela se interconecten y puedan cooperar en un algoritmo, compartiendo recursos e incrementando la potencia de cómputo. Esto implica clusters dedicados interconectados, a diferencia de *grid computing* o de *cloud computing* en que cada procesador puede realizar otras tareas independientes del algoritmo, brindando alta disponibilidad de procesamiento y/o de almacenamiento [14][15][16][17].

La caracterización y estudio de rendimiento del sistema de comunicaciones es de particular interés para la predicción y optimización de performance de los algoritmos, así como la homogeneidad o heterogeneidad de los procesadores que componen la arquitectura.

Muchos de los problemas algorítmicos en arquitecturas multiprocesador se han visto fuertemente impactados por el surgimiento de las máquinas multicore (que integran dos o más núcleos computacionales dentro de un mismo chip), y la tendencia creciente al uso de clusters de multicore.

A partir de incorporar varios chips multicore dentro de un nodo, y conectar múltiples nodos vía red, se puede crear una arquitectura NUMA de multicores, de modo que los cores dentro de un chip compartan la memoria principal, y puedan acceder remotamente a la memoria dedicada de otro chip, aunque ese acceso sea más costoso. También son comunes las arquitecturas donde los chips multicore dentro de un nodo comparten la memoria principal, la cual es accedida a través de un bus compartido o dedicado. De este modo surgen varios niveles de comunicación: Intra CMP (entre 2 cores del mismo chip), Inter CMP (entre 2 cores que radican en distintos chips pero en el mismo nodo), e Inter Nodo (entre 2 core de 2 nodos distintos).

Esto obliga al desarrollo de algoritmos que aprovechen adecuadamente esas arquitecturas, y al estudio de performance en sistemas híbridos, en los que se tiene una combinación de memoria compartida y distribuida [20][28][29]. Asimismo, es necesario estudiar la utilización de los diferentes lenguajes de programación para estas arquitecturas, ya que aún no se cuenta con un establecido en cuanto a lenguaje, aunque puede mencionarse el uso de MPI, OpenMP y OpenMPI.

1.2. Métricas de evaluación

La diversidad de opciones en los SP vuelve complejo el análisis de performance, ya que los ejes sobre los cuales pueden compararse dos sistemas son varios (relacionados con el algoritmo y/o la arquitectura). La performance obtenida en el sistema paralelo está dada por una compleja relación en la que intervienen factores como el tamaño del problema, la arquitectura de soporte, la distribución de procesos en procesadores, la existencia o no de un algoritmo de balance de carga, etc.

Existe un gran número de métricas para evaluar el rendimiento de los SP. Las tradicionales y más conocidas son el tiempo de ejecución paralelo, el speedup y la eficiencia. Otras características de los SP pueden ser analizadas por medio del costo, overhead paralelo, grado de concurrencia, etc. [19]

La *escalabilidad* de un SP permite capturar las características de un algoritmo paralelo y de la arquitectura en la que se lo implementa. Permite testear la performance de un programa paralelo sobre pocos procesadores y predecir su performance en un número mayor, y caracterizar la cantidad de paralelismo inherente en un algoritmo. Una de las principales métricas para medir la escalabilidad es la isoeficiencia.

Al tratar con multiclusters y grid, los problemas clásicos que caracterizan el análisis de los algoritmos paralelos, reaparecen potenciados por las dificultades propias de la interconexión a través de una red que en general es no dedicada. Esto se torna más complejo aún si se considera que cada uno de los nodos puede ser en si mismo una máquina multicore, en la cual además aparecen varios niveles de memoria.

Las mejoras que se obtienen al utilizar multicores se reflejan a través del paralelismo desarrollado para las

aplicaciones y su correcto mapeo en la arquitectura. El uso de multicores conlleva grandes cambios en la forma de desarrollar aplicaciones y software, y evaluar su rendimiento. Es necesario reescribir programas existentes, paralelizando el código para sacar provecho de los sistemas con múltiples núcleos. La cantidad de threads disponibles en estos sistemas también es un tema importante, debido a que la creación y administración de threads requiere del uso de recursos como memoria; además los threads deben ser cuidadosamente planificados (scheduling) e incorporados en la pila de ejecución. En este sentido, el desarrollo de técnicas de scheduling eficientes en multicores y clusters de multicores es un tema de interés.

1.3 El problema del balance de carga

El objetivo primario de la computación paralela es reducir el tiempo de ejecución haciendo uso eficiente de los recursos de cómputo. El *balance de carga* es un aspecto central y consiste en, dado un conjunto de tareas que comprenden un algoritmo y un conjunto de procesadores donde ejecutarlas, encontrar el mapeo (asignación) de tareas a procesadores que resulte en que cada una tenga una cantidad de trabajo que demande aproximadamente el mismo tiempo.

Un mapeo que balancea la carga de trabajo de los procesadores incrementa la eficiencia global y reduce el tiempo de ejecución. Este objetivo es particularmente complejo si los procesadores (y las comunicaciones entre ellos) son heterogéneos, y deben tenerse en cuenta las distintas velocidades.

El problema general de asignación es *NP*-completo para un sistema con *n* procesadores, y por lo tanto la tarea de encontrar una asignación de costo mínimo es computacionalmente intratable salvo para sistemas muy pequeños (métodos *óptimos*). Por esto pueden utilizarse enfoques que brindan soluciones subóptimas aceptables, como relajación, desarrollo de soluciones para casos particulares, optimización enumerativa, u optimización aproximada (métodos *heurísticos*) [18].

Si el tiempo de cómputo de una tarea dada puede determinarse “a priori”, se puede realizar el mapeo antes de comenzar la computación (balance de carga *estático*). En muchas aplicaciones la carga de trabajo para una tarea particular puede modificarse en el curso del cómputo, y no puede estimarse de antemano; en estos casos el mapeo debe cambiar durante el cómputo (balance de carga *dinámico*), realizando etapas de balanceo durante la ejecución de la aplicación. No puede establecerse un método efectivo y que sea eficiente en *todos* los casos. Siempre la elección depende de la aplicación y la plataforma de soporte, y en muchos casos es necesario adaptar o combinar métodos existentes para lograr buena performance [24][27].

Las técnicas de planificación o scheduling tanto a nivel micro (dentro de cada procesador) y macro (en un cluster) deben ser capaces de obtener un buen balance de carga. Si todos los cores/procesadores logran realizar aproximadamente la misma cantidad de operaciones, esto permite mejorar el tiempo de respuesta a los requerimientos, así como incrementar el rendimiento (throughput). Pero para ello es esencial un uso equitativo de los recursos por parte de todos los requerimientos. En este contexto, interesa trabajar en la investigación y desarrollo de algoritmos de scheduling que permitan ejecutar eficientemente aplicaciones paralelas sobre arquitecturas multicore y cluster de multicores, manejando la distribución de procesos en los cores desde la aplicación para obtener mayor ganancia de performance.

1.4 Modelos de representación, predicción y análisis de performance

Es importante referirse a un algoritmo paralelo mencionando el modelo de computación para el que fue diseñado. Uno de los objetivos en la definición de un modelo de computación es la posibilidad de *predicción de performance* que brinde el mismo, teniendo en cuenta conceptos tales como comunicación, sincronización y arquitectura física. Estos factores impiden que alguno de los modelos existentes pueda ser usarse para *todas* las máquinas paralelas.

Respecto de la representación de las aplicaciones paralelas en arquitecturas distribuidas, existen diferentes modelos basados en grafos para caracterizar el comportamiento de las mismas [21]. Entre los modelos se pueden mencionar TIG (Grafo de Interacción de Tareas), TPG (Grafo de Precedencia de Tareas) y TTIG (Grafo de Interacción Temporal de Tareas) [22]. Sin embargo, estos modelos consideran que la arquitectura es homogénea, situación que en general no se da en cluster, multicluster y grid, lo que hace necesaria la investigación en esta área. El desarrollo de nuevos modelos de predicción y análisis de performance para estas arquitecturas requiere caracterizar el contexto de comunicaciones entre los procesadores y la asociación entre los algoritmos de aplicación, el paradigma de cómputo paralelo elegido y la arquitectura de soporte.

1.5 Evaluación de performance. Aplicaciones

Es de interés la evaluación de performance de distintas clases de aplicaciones sobre las arquitecturas disponibles. Muchos sistemas paralelos no alcanzan su capacidad teórica, y las causas de esta degradación son muchas y no siempre fáciles de determinar. El análisis permite estudiar el impacto que tienen algunos de estos factores sobre las implementaciones, y adecuar las métricas a las mismas. Interesa estudiar la influencia de las estrategias de distribución de procesos y datos, y la carga (estática o dinámica) asignada a cada procesador sobre el speedup, la eficiencia y la escalabilidad. Aunque existen numerosas posibilidades estudiadas y aplicaciones

resueltas, es necesario continuar con la investigación en esta área, fundamentalmente por la aparición de nuevas características en las arquitecturas [24].

Entre las aplicaciones de interés se encuentran las numéricas y no numéricas, el tratamiento de imágenes y video, reconocimiento de patrones en secuencias de ADN, bases de datos distribuidas, sistemas inteligentes, data mining, etc.

Por otro lado, interesa el desarrollo de laboratorios remotos para el acceso a recursos de cómputo paralelo. Esto implica software de administración de recursos físicos, comunicaciones y software disponible en clusters, multiclusters y grid, con el objetivo de permitir acceso transparente.

2. LINEAS DE INVESTIGACION y DESARROLLO

- Paralelización de algoritmos secuenciales. Diseño y optimización de algoritmos.
- Lenguajes y bibliotecas para procesamiento paralelo y distribuido. Comparación de performance.
- Arquitecturas multicore y multithreading en multicore. Arquiecturas multiprocesador distribuidas.
- Sistemas paralelos como combinación de software y arquitectura.
- Estudio de la complejidad de algoritmos paralelos, en particular considerando múltiples núcleos y heterogeneidad.
- Modelos y paradigmas de computación paralela.
- Combinación de pasaje de mensajes y memoria compartida en cluster de multicore (programación sobre un modelo híbrido)
- Modelos de representación y predicción de performance de algoritmos paralelos.
- Mapping y scheduling de aplicaciones paralelas sobre distintas arquitecturas multiprocesador [13].
- Métricas del paralelismo. Speedup, eficiencia, rendimiento, granularidad, superlinealidad.
- Balance de carga estático y dinámico. Técnicas de balanceo de carga.
- Análisis de los problemas de migración y asignación óptima de procesos y datos a procesadores. Migración dinámica.
- Patrones de diseño de algoritmos paralelos.
- Escalabilidad de algoritmos paralelos en arquitecturas multiprocesador distribuidas.
- Implementación de soluciones sobre diferentes modelos de arquitectura homogéneas y heterogéneas (multicores, clusters, multiclusters y grid). Ajuste del modelo de software al modelo de hardware, a fin de optimizar el sistema paralelo. Evaluación de performance.
- Laboratorios remotos para el acceso transparente a recursos de cómputo paralelo

3. RESULTADOS OBTENIDOS/ESPERADOS

- Formar recursos humanos en los temas del Subproyecto, incluyendo tesinas de grado y tesis de postgrado.
- Desarrollar y optimizar algoritmos paralelos sobre los diferentes modelos de arquitectura multiprocesador. En particular, en aplicaciones numéricas y no numéricas de cómputo intensivo y tratamiento de grandes volúmenes de datos.
- Estudiar y desarrollar modelos de representación de aplicaciones paralelas y distribuidas y los algoritmos de mapeo (estático y dinámico) de procesos en procesadores asociados
- Realizar la migración de aplicaciones paralelas conocidas a esquemas multicore y cluster de multicore (en principio de 64, 128 y 256 núcleos), utilizando modelos de programación híbridos.
- Evaluar la performance (eficiencia, rendimiento, speedup, escalabilidad) de las soluciones propuestas.
- Mejorar y adecuar las técnicas disponibles para el balance de carga (estático y dinámico).
- Estudiar y proponer las adecuaciones necesarias para los modelos de predicción y evaluación de performance con diferentes paradigmas de interacción entre procesos, en esquemas multicore, cluster, multicluster y grid.
- Generalizar los laboratorios remotos ya desarrollados para el acceso transparente a recursos de cómputo paralelo.

En este marco, pueden mencionarse los siguientes resultados:

- Se han utilizado diferentes tipos de arquitecturas homogéneas o heterogéneas, incluyendo clusters conectados en la misma o diferentes LAN, en WAN por Internet 2, infraestructura grid experimental en el marco del Proyecto CyTEDGrid, y cluster de multicore con 64 núcleos.

- En cuanto a modelos de representación y predicción de performance:

  - Desarrollo de dos modelos para representar aplicaciones paralelas de manera más realista considerando la heterogeneidad de procesadores y red de interconexión [23].
  - Desarrollo de tres algoritmos de scheduling [23], que permiten obtener una mejor distribución de procesos en procesadores.
  - 2 extensiones del algoritmo de scheduling AMTHA para realizar la asignación de múltiples aplicaciones paralelas a una misma arquitectura.
  - Se trabajó sobre las modificaciones necesarias a los modelos y algoritmos de scheduling para adecuar sus usos sobre clusters de multicore y grid [27].

- Respecto de los algoritmos implementados, básicamente se trabajó con soluciones paralelas previamente tratadas en clusters:
  - ***N-Puzzle:*** es un problema de optimización discreta en el cual se debe llegar a un tablero "final" a partir de uno "inicial". Para esto deben realizarse intercambios entre una pieza y el lugar libre en el tablero (hueco) [25]. Se utilizó una variante de la heurística clásica de predicción de trabajo a realizar para llegar a una solución (tablero "final") y se demostró que su empleo mejora fuertemente el tiempo del algoritmo secuencial A*. Se realizó una solución paralela sobre una arquitectura distribuida para analizar el speedup en función del número de procesadores, la eficiencia, el balance de carga y la superlinealidad al escalar el problema. Este es un problema de interés por su aplicación principalmente en el campo de la robótica. Se trabajó sobre la generalización del problema a multiobjetivos, y se está analizando la migración a cluster de multicore.
  - ***Análisis de Secuencias de ADN:*** el análisis de secuencias de ADN tiene múltiples aplicaciones, una de ellas es la búsqueda de similitud entre una secuencia *test* y las almacenadas en grandes bases de datos. El gran tamaño de las bases de datos, y la complejidad computacional para comparar dos secuencias (Smith-Waterman) hacen necesaria la paralelización del algoritmo [12]. Se diseñaron dos algoritmos de acuerdo a la centralización y/o replicación de la Base de Datos. Estos algoritmos se ejecutaron sobre arquitecturas de tipo cluster y grid, estudiando el speedup y la eficiencia alcanzable, la escalabilidad de las soluciones y el overhead introducido en grid por las comunicaciones y el middleware requerido. Se pretende extender las experiencias sobre GRID utilizando procesadores geográficamente distribuidos sobre diferentes redes ubicadas en Argentina, Brasil y España.
  - ***Extracción de conocimiento en grandes bases de datos utilizando estrategias adaptativas:*** la tecnología actual posibilita el almacenamiento de enormes volúmenes de información. Es importante encontrar métodos y técnicas de minería de datos que sean capaces de generar conocimiento útil, produciendo resultados que sean de provecho al usuario. Para esto, se estudian métodos de clustering y clasificación de patrones para lograr asociar respuestas dinámicas con los datos de entrada obtenidos. Dado el gran volumen de información a procesar (que puede estar físicamente distribuida), se estudian la arquitectura y los paradigmas de programación paralela utilizables de modo de minimizar el tiempo de cálculo.
- Se avanzó sobre la paralelización de aplicaciones en cluster de multicore, utilizando paralelismo de datos, de control e híbrido, utilizando como caso de estudio BASIZ (algoritmo de procesamiento de imágenes). Se desarrolló una técnica de asignación de procesos a núcleos (mapping) desde las aplicaciones y se la comparó con el mapeo automático realizado por el sistema operativo obteniendo muy buenos resultados.
- En cuanto a los laboratorios para acceso remoto a recursos de cómputo paralelo, se ha desarrollado una aplicación que permite el acceso a los clusters de la Facultad de Informática vía WEB (para investigadores y alumnos). Esto implica una capa de software de administración de los recursos y acceso transparente. Se está trabajando en la generalización de la herramienta a clusters que no se encuentran en la misma red y a cluster de multicore.

4. FORMACION DE RECURSOS HUMANOS

Dentro de la temática de la línea de I/D se esperan concluir este año 2 tesis doctorales que se encuentran en curso y 2 tesis de maestría. Se concluyeron dos Tesinas de Grado de Licenciatura y se espera finalizar otras 2 en este año.

Además, se participa en el dictado de las carreras de Doctorado en Ciencias Informáticas y Magíster y Especialización en Cómputo de Altas Prestaciones de la Facultad de Informática de la UNLP, por lo que potencialmente pueden generarse Tesis de Doctorado y Maestría y Trabajos Finales de Especialización.

Existe cooperación con grupos de otras Universidades del país y del exterior, y hay tesistas de diferentes Universidades realizando su Tesis con el equipo del proyecto.

5. BIBLIOGRAFIA

[1] Ben-Ari, M. "Principles of Concurrent and Distributed Programming, 2/E". Addison-Wesley. 2006.

[2] Vijay K. Garg. "Elements of Distributed Computing". Wiley-IEEE Press, 2002.

[3] Bischof C., Bucker M., Gibbon P., Joubert G., Lippert T., Mohr B., Peters F. (eds.), Parallel Computing: Architectures, Algorithms and Applications, Advances in Parallel Computing, Vol. 15, IOS Press, February 2008.

[4] Grama A., Gupta A., Karypis G., Kumar V. "Introduction to Parallel Computing", Pearson Addison Wesley, 2nd Edition, 2003.

[5] Attiya H., Welch J. "Distributed Computing: Fundamentals, Simulations, and Advanced Topics", (Wiley Series on Parallel and Distributed Computing). Wiley-Interscience; 2nd edition, 2004.

[6] Berman K. A., Pau J. L. "Algorithms:

Sequential, Parallel, and Distributed". Course Technology; 1st edition, 2004.
[7] Wilkinson B., Allen M. "Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers (2nd Edition)". Prentice Hall, 2004.
[8] Nir Shavit, Maurice Herlihy, "The Art of Multiprocessor Programming", Morgan Kaufmann Pub, 2008, ISBN-10: 0123705916, ISBN-13: 9780123705914
[9] Becker Alexander (Editor), "Concurrent and Parallel Computing: Theory, Implementation and Applications", Nova Science Pub Inc, 2008, ISBN-10: 1604562749, ISBN-13: 9781604562743
[10] Leopold C. "Parallel and Distributed Computing. A survey of Models, Paradigms, and Approaches", Wiley Series on Parallel and Distributed Computing. Albert Zomaya Series Editor, 2001
[11] Heroux M. A., Raghavan P., Simon H. D., "Frontiers of Scientific Computing: An Overview . Parallel Processing for Scientific Computing, Software, Environments, and Tools", Vol. 20, pp. 1-5, SIAM, November 2006.
[12] Chichizola F., Naiouf R. M., De Giusti L. C., Rodríguez I., De Giusti A. E. "Parallel Processing ADN Sequences on Multicluter and Grid Architectures. Software Overhead". Proceedings del IX Workshop de Procesamiento Distribuido y Paralelo. La Rioja, Argentina. ISBN: 978-987-24611-0-2. Octubre de 2008.
[13] Dummler J., Ruaber T., Runger G., Mapping Algorithms for Multiprocessor Tasks on Multi-Core Clusters, Proc. 2008 International Conference on Parallel Processing (37th ICPP'08) CD-ROM, IEEE Computer Society, September 2008.
[14] Grid Computing and Distributed Systems (GRIDS) Laboratory - Department of Computer Science and Software Engineering (University of Melbourne). "Cluster and Grid Computing". 2007. http://www.cs.mu. oz.au/678/.
[15] Foster I., Kesselman C. "The Grid 2: Blueprint for a New Computing Infrastructure". (The Morgan Kaufmann Series in Computer Architecture and Design). Morgan Kaufmann; 2nd edition, 2003.
[16] Parashar M., Li Xiaolin, Chandra Sumir, "Advanced Computational Infrastructures for Parallel and Distributed Applications", Wiley-Interscience, 2009 ISBN-10: 0470072946 ISBN-13: 9780470072943.
[17] Silva V. "Grid Computing For Developers" (Programming Series). Charles River Media; 1st edition, 2005.
[18] Olivier S., Prins S., Scalable Dynamic Load Balancing Using UPC. Proc. 2008 International Conference on Parallel Processing (37th ICPP'08) CD-ROM, IEEE Computer Society, September 2008.
[19] Sun X-H . "Scalability versus Execution Time in Scalable Systems". Journal of Parallel and Distributed Computing, Number 12, 2002, pp 173-192
[20] Chapman B., The Multicore Programming Challenge, Advanced Parallel Processing Technologies; 7th International Symposium, (7th APPT'07), Lecture Notes in Computer Science (LNCS), Vol. 4847, p. 3, Springer-Verlag (New York), November 2007.
[21] Teller J., Ozguner F., Ewing R., Scheduling Task Graphs on Heterogeneous Multiprocessors with Reconfigurable Hardware, Proc. 2008 International Conference on Parallel Processing (37th ICPP'08) CD-ROM, Fourth International Workshop on Scheduling and Resource Management for Parallel and Distributed Systems, IEEE Computer Society, Sept. 2008.
[22] Roig C., Ripoll A. Guirado F. "A New Task Graph Model for Mapping Message Passing Applications". IEEE Transactions on Parallel and Distr. Systems, vol 18 (12), pp.740-1753. 2007.
[23] De Giusti L. "Mapping sobre Arquitecturas Heterogéneas". Tesis Doctoral, Universidad Nacional de La Plata (2008).
[24] Naiouf R. M., De Giusti L. C., Chichizola F., De Giusti A. E. "Dynamic Load Balancing on Non-homogeneous Clusters". G.Min et al. (Eds.): ISPA 2006 Ws, LNCS 4331, pags. 65-73, 2006. Springer – Verlag. Berlin Heidelber 2006.
[25] Victoria Sanz, Armando De Giusti, Marcelo Naiouf. "4-(n^2-1) Puzzle: parallelization and performance on clusters". Anales del XV Congreso Argentono de Ciencias de la Computación CACIC2009. Octubre 2009, Jujuy Argentina.
[26] Qiu X., Fox G. G., Yuan H., Bae S., Chrysanthakopoulos G., Nielsen H. F. "Performance of Multicore Systems on Parallel Data Clustering with Deterministic Annealing". LNCS 4331, pags. 407-416. ISBN 978-3-540-69383-3. Springer Berlin / Heidelber 2008.
[27] Yi Liu, Xin Zhang, He Li, Depei Qian. "Allocating Tasks in Multi-core Processor based Parallel Systems". Proceedings of the 2007 IFIP International Conference on Network and Parallel Computing – Workshops. IEEE Computer Society. 2007.
[28] Lei Chai, Qi Gao, Dhabaleswar K. Panda. "Understanding the Impact of Multi-Core Architecture in Cluster Computing: A Case Study with Intel Dual-Core System". IEEE Intl Symposium on Cluster Computing and the Grid 2007 (CCGRID 2007), pp. 471-478 (2007).
[29] Suresh Siddha, Venkatesh Pallipadi, Asit Mallick. "Process Scheduling Challenges in the Era of Multicore Processors"Intel Technology Journal, Vol. 11, Issue 04, November 2007.