Procesamiento Paralelo y Distribuido. Fundamentos, Modelos y Aplicaciones.

Marcelo Naiouf, Armando De Giusti, Laura De Giusti, Franco Chichizola, Adrian Pousa,
Victoria Sanz, Fabiana Leibovich, Emmanuel Frati, Enzo Rucci, Silvana Gallo, Franco Ronchetti, Diego Encinas
Instituto de Investigación en Informática LIDI (III-LIDI) - Facultad de Informática - UNLP
{mnaiouf, degiusti, ldgiusti, francoch, apousa, vsanz, fleibovich, fefrati, erucci, sgallo, fronchetti, dencinas}@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 III-LIDI acreditado por la UNLP, y de proyectos apoyados por Cyted, CIC, IBM, Fundación YPF y Telefónica.
Existe cooperación con Universidades de Argentina, Latinoamérica y Europa a través de los proyectos "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 LAGrid (LatinAmerican Grid) de IBM y 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, en lo referente a los fundamentos y a las aplicaciones. 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 en 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, grid y cloud.

Se trabaja principalmente en la concepción de aplicaciones paralelas numéricas y no numéricas sobre grandes volúmenes de datos y/o que requieren cómputo intensivo.

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 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. Más allá de las mejoras constantes en las arquitecturas físicas, uno de los mayores desafíos se centra en cómo aprovechar al máximo su potencia. En esta línea de I/D la mayor importancia está 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 el uso de manera masiva de procesadores con más de un núcleo (multinúcleo o multicore). Esto ha producido plataformas distribuidas híbridas (memoria compartida y distribuida), llevando a la necesidad de desarrollar sistemas operativos, lenguajes y algoritmos que las usen adecuadamente.

Pueden encontrarse diferentes formulaciones paralelas para un problema, y el rendimiento y la eficiencia de cada una depende del algoritmo y de la arquitectura.

1.1. Algoritmos Paralelos y Distribuidos y Arquitecturas Multiprocesador

La creación de algoritmos paralelos y distribuidos en arquitecturas multiprocesador, o la transformación de un algoritmo secuencial en paralelo, no es un proceso directo. El "costo" 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 particionamiento, comunicación, aglomeración y mapeo de procesos a procesadores.

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

Los algoritmos, pueden ser escritos utilizando diferentes paradigmas (C/S, 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 la noción de sistema distribuido como máquina paralela es común a

denominaciones como redes, NOW, SMP, clusters, multiclusters, *grid* y *cloud*. En estos casos, se deben identificar las capacidades de procesamiento, interconexión, sincronización y escalabilidad [14][15][16][17]. La caracterización y estudio de rendimiento del sistema de comunicaciones es de interés para la predicción y optimización de performance de los algoritmos, así como la homogeneidad o heterogeneidad de los procesadores.

Muchos de los problemas algorítmicos 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 modo que los cores en un chip compartan memoria principal, y puedan acceder remotamente a la memoria dedicada de otro chip, aunque ese acceso sea más costoso. 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 [20][28][29]. Además, es necesario estudiar la utilización de diferentes lenguajes de programación, ya que aún no se cuenta con un establecido en cuanto a lenguaje, aunque puede mencionarse el uso de MPI, OpenMP y Pthreads.

1.2. Métricas de evaluación

La diversidad de opciones vuelve complejo el análisis de performance de los SP, ya que los ejes sobre los cuales pueden compararse dos sistemas son varios. La performance obtenida está dada por una compleja relación en que intervienen numerosos factores. Existe un gran número de métricas para evaluar el rendimiento, siendo las tradicionales tiempo de ejecución, speedup y eficiencia. Otras características pueden ser analizadas por medio del costo, overhead, grado de concurrencia, etc. [19]

La *escalabilidad* permite capturar características de un algoritmo paralelo y de la arquitectura en que se lo implementa. Permite testear la performance de un programa sobre pocos procesadores y predecirla en un número mayor, así como caracterizar la cantidad de paralelismo inherente en un algoritmo.

En arquitecturas distribuidas, los problemas que caracterizan el análisis de los algoritmos paralelos aparecen potenciados por las dificultades propias de la interconexión en una red en general no dedicada. Esto se torna más complejo aún si cada nodo puede ser un multicore con 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. Esto conlleva grandes cambios en la forma de desarrollar aplicaciones y software, y evaluar su rendimiento. La cantidad de threads disponibles en estos sistemas también es importante, ya que la creación y administración de los mismos 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 es un tema de interés.

1.3 El problema del balance de carga

El objetivo primario del cómputo paralelo es reducir el tiempo de ejecución haciendo uso eficiente de los recursos. 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, 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. Dado que el problema general de mapping es *NP*-completo, pueden usarse enfoques que brindan soluciones subóptimas aceptables [18].

Si el tiempo de las tareas puede determinarse "a priori", es posible realizar el mapeo para lograr balance de carga antes de comenzar la computación (*estático*). En muchas aplicaciones la carga de trabajo de las tareas puede modificarse en el curso del cómputo, y deben usarse técnicas *dinámicas.* No puede establecerse un método efectivo y eficiente en *todos* los casos. Siempre la elección depende de la aplicación y la plataforma de soporte [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 buen balance de carga. Si todos los cores/procesadores logran realizar aproximadamente la misma cantidad de operaciones, puede mejorarse el tiempo de respuesta así como incrementar el rendimiento (throughput). Interesa realizar I/D 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 los modelos existentes pueda ser usarse para *todas* las máquinas paralelas.

Respecto de la representación de las aplicaciones paralelas en arquitecturas distribuidas, existen modelos basados en grafos para caracterizar el comportamiento de las mismas [21]; se pueden mencionar TIG, TPG y TTIG [22], pero consideran una arquitectura homogénea, lo que hace necesaria la investigación en esta área. El desarrollo de nuevos modelos 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. 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 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, sistemas inteligentes, data mining, etc.

2. LINEAS DE INVESTIGACION y DESARROLLO

- Paralelización de algoritmos secuenciales. Diseño y optimización de algoritmos.
- Comparación de lenguajes y bibliotecas para procesamiento paralelo y distribuido.
- Arquitecturas multicore. Multithreading en multicore. Multiprocesadores distribuidos.
- Estudio de complejidad de algoritmos paralelos, en particular considerando multicore y heterogeneidad.
- Modelos y paradigmas de computación paralela.
- Programación sobre un modelo híbrido (pasaje de mensajes y memoria compartida) en cluster de multicore.
- Modelos de representación y predicción de performance de algoritmos paralelos.
- Mapping y scheduling de aplicaciones paralelas sobre distintas arquitecturas multiprocesador [13].
- Análisis de los problemas de migración y asignación óptima de procesos y datos a procesadores. Migración dinámica.
- Evaluación de performance. Speedup, eficiencia, rendimiento, granularidad.
- Escalabilidad de algoritmos paralelos en arquitecturas multiprocesador distribuidas.
- Balance de carga estático y dinámico. Técnicas.
- Patrones de diseño de algoritmos paralelos.
- Implementación de soluciones sobre diferentes modelos de arquitecturas homogéneas y heterogéneas.

3. RESULTADOS OBTENIDOS/ESPERADOS

- Formar RRHH en los temas del Subproyecto, incluyendo tesis de postgrado y tesinas de grado.
- Desarrollar y optimizar algoritmos paralelos sobre los diferentes modelos de arquitectura. 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.

En este marco, pueden mencionarse los siguientes resultados:

- Se han utilizado y analizado diferentes tipos de arquitecturas homogéneas o heterogéneas, incluyendo clusters conectados en la misma o diferentes LAN, en WAN por Internet 2, infraestructura grid experimental, y cluster de multicore con 128 núcleos.
- En cuanto a modelos de representación y predicción de performance:
  - 2 extensiones del algoritmo de scheduling AMTHA para realizar la asignación de múltiples aplicaciones paralelas a una misma arquitecturadistribuida heterogénea.
  - Se trabajó sobre las modificaciones necesarias a los modelos y algoritmos de scheduling para adecuar su uso a clusters de multicore y grid [27].
- Se estudiaron las técnicas de mapping y scheduling utilizadas en forma estándar por el sistema operativo y se desarrollaron técnicas propias, comparando los resultados obtenidos.
- 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]. Este es un problema de interés por su aplicación principalmente en el campo de la robótica y su complejidad, haciendo indispensable el desarrollo de algoritmos paralelos para su resolución. Se desarrolló un algoritmo secuencial y un algoritmo paralelo basados en el algoritmo de

búsqueda A*.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*. La solución paralela se implementó para hacer uso de una arquitectura distribuida, y se analizó el speedup en función del número de procesadores, la eficiencia, el balance de carga y la superlinealidad al escalar el problema. También se implementó un algoritmo secuencial y un algoritmo paralelo basado en la técnica de búsqueda de soluciones subóptimas Weighted A* y se evaluó la calidad de la solución encontrada respecto de la óptima, así como también la ganancia en tiempo y nodos procesados. 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 semejanzas entre dos secuencias. El gran tamaño que pueden alcanzar las secuencias (hasta 10^9 nucleótidos) y la complejidad computacional para compararlas por medio del algoritmo de Smith-Waterman (orden N^2) hacen necesaria la paralelización del algoritmo [12]. Se diseñaron soluciones paralelas utilizando diferentes modelos de comunicación (memoria compartida, mensajes y una combinación de ambos). Estos algoritmos se ejecutaron sobre arquitecturas de tipo cluster estándares homogéneos y heterogéneos, y por otra parte combinando memoria compartida y distribuida en un cluster de multicores. En todos los casos interesa analizar el speedup y la eficiencia alcanzable, además de la escalabilidad de las soluciones implementadas.
- ***Controlador robótico obtenido a través una metaheurística de población variable:*** la robótica evolutiva muestra un fuerte interés en las redes neuronales (RN) por considerarlas un modelo artificial de la manera en que los humanos aprenden y por poseer la capacidad de representar el conocimiento adquirido a través de una estructura que, una vez entrenada, puede operar en tiempo real. A una RN capaz de comandar un robot autónomo se la denomina controlador neuronal. Se propuso una nueva estrategia evolutiva (metaheurística poblacional) donde cada individuo es un controlador neuronal completo [30]. Esta estrategia tiene una complejidad computacional importante en el entrenamiento ya que debe evaluar constantemente a los individuos de la población por medio de simulaciones. Se realizó una solución paralela de la etapa de entrenamiento para reducir el tiempo de aprendizaje y/o mejorar el comportamiento del robot al poder trabajar con poblaciones de mayor tamaño. Esta solución se evaluó sobre una arquitectura distribuida homogénea (cluster homogéneo) [31].
- ***Simulación de eventos discretos:*** se trata de sistemas con grandes necesidades de cómputo; en particular, se trabajó con modelos discretos, dinámicos y estocásticos, Se estudió la paralelización con OpenMPI sobre un cluster de multicore con diferentes estrategias, entre ellas: replicación de instancias, descomposición del dominio y distribución temporal.

- Se avanzó en la paralelización de aplicaciones en cluster de multicore con paralelismo de datos, tomando como caso de estudio una fase del algoritmo BASIZ (procesamiento de imágenes). Se desarrollaron soluciones con memoria compartida, pasaje de mensajes y otra híbrida, que permite aprovechar las características de la arquitectura. Se estudió la mejora introducida por el uso de una estrategia híbrida en dos sentidos: por una parte, al crecer el tamaño del problema (escalabilidad), y por la otra comparando la solución con otras puras de memoria compartida o de pasaje de mensajes, obteniendo buenos resultados que favorecen a la estrategia híbrida.

4. FORMACION DE RECURSOS HUMANOS

Dentro de la temática de la línea de I/D se concluyó una Tesis Doctoral, un Trabajo Final de Especialización y 3 Tesinas de Grado de Licenciatura. Se espera concluir este año otras 2 tesis doctorales que se encuentran en curso, 2 tesis de maestría y 2 Tesinas de Grado de Licenciatura.

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] Rucci Enzo, De Giusti Armando E., Chichizola Franco, Naiouf R. Marcelo, De Giusti Laura C.. "Comparación de modelos de comunicación/sincronización en Programación Paralela sobre Cluster de Multicores". Proceedings del XVI Congreso Argentino de Ciencias de la Computación. Buenos Aires, Argentina. Págs. 201-210. ISBN 978-950-9474-49-9. Octubre 2010.
[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
[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 Argentino 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.
[30] Ronchetti F., Lanzarini L. "Controlador Robótico obtenido a través de una metaheurística de población variable". Proceedings del XVI Congreso Argentino de Ciencias de la Computación (CACIC'10). Buenos Aires, Argentina. Págs. 92-101. ISBN 978-950-9474-49-9. Octubre 2010.
[31] Ronchetti F.. "Controlador robótico obtenido a través una metaheurística de población variable". Tesina de grado de Lic. en Informática., UNLP. 2011.