Algoritmos Paralelos y Distribuidos. Fundamentos, Modelos y Aplicaciones.

Marcelo Naiouf, Armando De Giusti[1], Laura De Giusti, Franco Chichizola, Victoria Sanz[2], Adrian Pousa, Fabiana Leibovich, Enzo Rucci, Emmanuel Frati[2], Diego Encinas, Silvana Gallo, Erica Montes de Oca, Javier Balladini
Instituto de Investigación en Informática LIDI (III-LIDI) - Facultad de Informática - UNLP
{mnaiouf, degiusti, ldgiusti, francoch, vsanz, apousa, fleibovich, erucci, fefrati, dencinas, sgallo, emontesdeoca}@lidi.info.unlp.edu.ar, javier.balladini@gmail.com

CONTEXTO

Se presenta una línea de Investigación que 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 el Ministerio de Educación, y de proyectos específicos apoyados por organismos nacionales e internacionales.

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) para el intercambio de Docentes y Alumnos de Doctorado.

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 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, evaluación de performance. Las arquitecturas pueden ser homogéneas o heterogéneas.

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

Actualmente se han incorporado temáticas como el uso de GPGPU para el desarrollo de soluciones, y el análisis del consumo y la eficiencia energética en algoritmos paralelos.

Este proyecto se coordina con otros dos en curso en el III-LIDI, relacionados con Arquitecturas Distribuidas y Paralelas y Sistemas de Software Distribuido.

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

1. INTRODUCCION

Por numerosos motivos, el procesamiento paralelo y distribuido se ha convertido en un área de gran interés dentro de la Ciencia de la Computación, produciendo transformaciones en las líneas de I/D [BEN06] [BIS08][GRA03][RAU10][SHA08][BEC08].

Interesa realizar I/D en la especificación, transformación, optimización y evaluación de algoritmos distribuidos y paralelos, ya que más allá de las mejoras en las arquitecturas físicas, el mayor desafío se centra en cómo aprovechar al máximo su potencia. En esta línea la mayor importancia está en los *algoritmos paralelos* y en los métodos utilizados para su construcción y análisis [WIL04] [QIU08].

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.

1.1. Algoritmos Paralelos y Distribuidos y Arquitecturas Multiprocesador

La creación de algoritmos paralelos/distribuidos en arquitecturas multiprocesador, o la paralelización de un algoritmo secuencial, no es un proceso directo. El "costo" puede ser alto en términos del esfuerzo de programación [SHA08][BEC08][LEO01], y 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 (que puede escribirse usando diferentes paradigmas) 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.

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 [FOS03][PAR09]. La caracterización y estudio de rendimiento del sistema de comunicaciones es de interés para la predicción y optimización de

[1] Investigador CONICET [2] Becario CONICET

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 multicores. 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 [CHA07][SID07]. Además, es necesario estudiar la utilización de diferentes lenguajes ya que aún no se cuenta con un standard, 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.

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.

El uso de procesadores con múltiples núcleos conlleva 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 su creación y administración 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. Esto es particularmente complejo si los procesadores (y las comunicaciones) 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 [OLI08].

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* [LIU07].

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. Interesa realizar I/D de algoritmos de scheduling que permitan ejecutar eficientemente las aplicaciones paralelas, manejando la distribución de procesos en los cores desde la aplicación para obtener mayor ganancia de performance [DUM08].

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. Las variantes existentes en estos factores impiden que los modelos existentes puedan usarse para *todas* las máquinas paralelas.

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. Un aspecto de interés que se ha sumado es el del consumo requerido [FEN05].

[1] Investigador CONICET [2] Becario CONICET

Entre las aplicaciones de interés se encuentran las numéricas y no numéricas con alta demanda de cómputo.

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 multicores.
- Modelos de representación y predicción de performance de algoritmos paralelos.
- Mapping y scheduling de aplicaciones paralelas sobre distintas arquitecturas multiprocesador.
- Análisis de los problemas de migración y asignación de procesos y datos a procesadores.
- Evaluación de performance. Speedup, eficiencia, escalabilidad.
- Balance de carga estático y dinámico. Técnicas.
- Análisis de consumo y eficiencia energética en 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, cluster de multicores (en principio de 64, 128 y 256 núcleos) utilizando modelos de programación híbridos, y GPGPU
- 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, y con distintas arquitecturas de soporte.
- Estudiar el impacto producido por los modelos de programación, lenguajes y algoritmos sobre el consumo y la eficiencia energética.

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, infraestructura grid experimental, cluster de multicores con 128 núcleos y GPGPU.

- En cuanto a modelos de representación y predicción de performance, se trabajó sobre posibles extensiones del algoritmo de scheduling AMTHA para realizar la asignación de múltiples aplicaciones paralelas a una misma arquitectura distribuida heterogénea. Asimismo, se estudiaron las modificaciones necesarias a los modelos y algoritmos de scheduling para adecuar su uso a clusters de multicores y grid [LIU07].

- 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 aplicaciones estudiadas y algoritmos implementados, se trabajó fundamentalmente con soluciones paralelas previamente tratadas en clusters tradicionales:

  - **Búsqueda de soluciones óptimas y subóptimas para problemas de optimización discreta:** los problemas de optimización discreta *(DOP)* son de interés debido a sus aplicaciones, en particular en robótica, y en general se resuelven mediante algoritmos de búsqueda heurística (variantes de *Best First Search*), que exploran el grafo que representa el espacio de estados del problema, procesando primero aquellos estados que se estiman más prometedores. La solución encontrada debe minimizar una función objetivo. Habitualmente estas técnicas de búsqueda son costosas, dado que el grafo crece en orden exponencial, haciendo imprescindible la aplicación de paralelismo. Tomando como caso de estudio el problema del N-Puzzle [SAN11], se desarrolló un algoritmo secuencial y un algoritmo paralelo (basados en el algoritmo A*) para resolverlo. 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, para distintas instancias del N-Puzzle.
  Dependiendo de la instancia a resolver, la búsqueda de una solución óptima puede requerir excesivo tiempo de cómputo, aun aplicando paralelismo. En consecuencia, se desarrolló un algoritmo secuencial y un algoritmo paralelo basados en la técnica de búsqueda de soluciones subóptimas Weighted A*. Se analizó la calidad de las soluciones encontradas por el algoritmo secuencial subóptimo, para distintas instancias del problema, a medida que se incrementa el peso usado en la función de costo.

[1] Investigador CONICET [2] Becario CONICET

Luego se analizaron el speedup y eficiencia obtenidos por el algoritmo paralelo subóptimo, y se comparó para cada instancia inicial la calidad de la solución alcanzada contra aquella encontrada por el algoritmo secuencial subóptimo, a medida que se incrementa el peso usado en la función de costo.
Actualmente se está analizando la migración del algoritmo paralelo A* a cluster de multicores.

- **Aplicaciones con paralelismo de datos**. Se avanzó en la paralelización en cluster de multicores, tomando como caso de estudio una aplicación base como la multiplicación de matrices. Se implementaron dos soluciones diferentes, una tradicional y otra obteniendo la matriz resultante por bloques. Las mismas se desarrollaron tanto con pasaje de mensajes como con un esquema híbrido, aprovechando de esta manera las características de la arquitectura. Se estudió la mejora introducida por el uso de la estrategia híbrida en dos sentidos: por una parte, al crecer el tamaño del problema (escalabilidad), y por la otra comparando la solución con otra pura de pasaje de mensajes, obteniendo buenos resultados que favorecen a la solución híbrida. Por otro lado, se realizó un análisis comparativo entre la combinación de las librerías de programación paralela que permiten la programación híbrida. Por este motivo se implementaron las soluciones híbridas mencionadas anteriormente utilizando MPI+Pthreads y MPI+OpenMP [LEI11]

- ***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 [RUC11]. 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.

- **Construcción de árboles filogenéticos:** se llama filogenia a la relación entre los diferentes conjuntos de especies del planeta, la cual puede representarse mediante un árbol, y su tarea consiste en inferir dicho árbol a partir de las observaciones realizadas sobre los organismos existentes. Los avances en las tecnologías de secuenciación, las cuales permiten obtener conjuntos de datos cada vez más grandes, y la complejidad computacional para construir los árboles filogenéticos por medio del método Neighbor-Joining (orden N^3) hacen necesaria su paralelización [STU88]. Se diseñaron soluciones paralelas utilizando diferentes modelos de comunicación (memoria compartida, mensajes y una combinación de ambos). Estos algoritmos se ejecutaron sobre una arquitectura cluster de multicores, la cual combina memoria compartida y distribuida. Interesa analizar el speedup y la eficiencia alcanzable, además de la escalabilidad de las soluciones implementadas.

- ***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 sobre un cluster de multicores con tres modelos diferentes de comunicación: memoria compartida (utilizando OpenMP y Pthreads), memoria distribuida (usando MPI) y soluciones híbridas, y variando la distribución de datos y procesos.

- ***Algoritmos de cifrado.*** El volumen de datos que se trasmiten en las redes se ha incrementado considerablemente, y en ocasiones suelen tratarse de información sensible. Por este motivo, es importante codificar los datos para enviarlos por una red pública como lo es Internet de manera segura. El encriptado y desencriptado de datos requiere un tiempo de cómputo adicional, que dependiendo de su tamaño puede ser considerable. Es importante aprovechar las posibilidades que brindan las arquitecturas multicores para reducir este tiempo. Se realizó un análisis del rendimiento del algoritmo de cifrado simétrico AES (Advanced Encryption Standard) sobre distintas arquitecturas multicore. Para ello se realizaron tres implementaciones, basadas en el lenguaje C, que utilizan herramientas de programación paralela OpenMP, MPI y CUDA, para ser ejecutadas sobre procesadores multicore, cluster de multicores y GPU respectivamente. Los resultados obtenidos mostraron la eficiencia de la implementación CUDA sobre la GPU a medida que aumenta el tamaño de los datos de entrada [POU11]. Actualmente se ha extendido la investigación a otro tipo de problemas con alta demanda computacional, como los del tipo N-body [KIR10].

- Se comenzaron a estudiar los aspectos de consumo y eficiencia energética en las aplicaciones mencionadas anteriormente, con el objetivo de medir el eventual impacto producido por los diferentes modelos de comunicación, paradigmas paralelos y lenguajes.

4. FORMACION DE RECURSOS HUMANOS

Dentro de la temática de la línea de I/D se concluyó una Tesis Doctoral, tres Trabajos Finales de Especialización y 3 Tesinas de Grado de Licenciatura. Se encuentran en curso en el marco del proyecto 8 tesis doctorales, 4 tesis de maestría, 2 trabajos de Especialización y 3 Tesinas de Licenciatura.

[1] Investigador CONICET [2] Becario CONICET

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

[BEC08] Becker Alexander (Editor), "Concurrent and Parallel Computing: Theory, Implementation and Applications", Nova Science Pub Inc, 2008, ISBN-10: 1604562749, ISBN-13: 9781604562743

[BEN06] Ben-Ari, M. "Principles of Concurrent and Distributed Programming, 2/E". Addison-Wesley. 2006.

[BIS08] 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.

[CHA07] 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.

[DUM08] 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.

[FEN05] Feng, W.C., "The importance of being low power in high-performance computing". Cyberinfrastructure Technology Watch Quarterly. 2005.

[FOS03] 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.

[GRA03] Grama A., Gupta A., Karypis G., Kumar V. "Introduction to Parallel Computing", Pearson Addison Wesley, 2nd Edition, 2003.

[KIR10] Kirk D., Hwu W. "Programming Massively Parallel Processors: A Hands-on Approach (Applications of GPU Computing Series). Morgan-Kaufmann. 2010.

[LEI11] Leibovich F., Gallo S., De Giusti L., Chichizola F., Naiouf M., De Giusti A. "Comparación de paradigmas de programación paralela en cluster de multicores: Pasaje de mensajes e híbrido. Un caso de estudio". Proceedings del XVII Congreso Argentino de Ciencias de la Computación (CACIC 2011). Págs. 241-250. ISBN 978-950-34-0756-1.

[LIU07] 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.

[OLI08] 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.

[PAR09] Parashar M., Li Xiaolin, Chandra Sumir, "Advanced Computational Infrastructures for Parallel and Distributed Applications", Wiley-Interscience, 2009 ISBN-10: 0470072946

[POU11] Pousa A., Sanz V., De Giusti A. "Análisis de rendimiento de un algoritmo de criptografía simétrica Sobre arquitecturas multicore". Procs CACIC 2011. Pp 231-240. ISBN: 978-950-34-0756-1.

[QIU08] 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.

[RAU10] Rauber T., Rünger G. "Parallel programming for multicore and cluster systems". Springer. 2010.

[RUC11] Rucci E., Chichizola F., De Giusti L., Naiouf M., De Giusti A. "DNA sequence alignment: hybrid parallel programming on a multicore cluster". Procs. of the 2011 Int'l Conference on Computers, Digital Communications and Computing (ICDCC'11). España, 2011. ISBN 978-1-61804-030-5, pp 183-190.

[SAN11] Victoria Sanz, Marcelo Naiouf, Armando De Giusti, Marcelo Naiouf. "Parallel Suboptimal Heuristic Search for finding a w-admissible solution. Performance analysis". International Conference on Computers, Digital Communications and Computing (ICDCC'11). Barcelona, España. 978-1-61804-030-5

[SHA08] Nir Shavit, Maurice Herlihy, "The Art of Multiprocessor Programming", Morgan Kaufmann Pub, 2008, ISBN-10: 0123705916, ISBN-13: 9780123705914

[SID07] Suresh Siddha, Venkatesh Pallipadi, Asit Mallick. "Process Scheduling Challenges in the Era of Multicore Processors"Intel Technology Journal, Vol. 11, Issue 04, November 2007.

[STU88] J. Studier and K. Keppler, "A note on the neighbor-joining algorithm of Saitou and Nei," Mol. BioI.Evol., vol. 5, no. 6, 1988, pp. 729-731.

[TEL08] Teller J., Ozguner F., Ewing R., Scheduling Task Graphs on Heterogeneous Multiprocessors with Reconfigurable Hardware, Proc. 2008 Int´l Conf. on Parallel Processing (37th ICPP'08) CD-ROM, 4th Int'l Workshop on Scheduling and Resource Management for Parallel and Distributed Systems, IEEE CS, Sept. 2008.

[WIL04] Wilkinson B., Allen M. "Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers (2nd Edition)". Prentice Hall, 2004.

[1] Investigador CONICET [2] Becario CONICET