Cómputo de Altas Prestaciones. Fundamentos de Cómputo Paralelo y Distribuido. Construcción y Evaluación de Aplicaciones

Marcelo Naiouf[(1)], Armando De Giusti[(1)(2)], Laura De Giusti[(1)(3)], Franco Chichizola[(1)], Victoria Sanz[(1)(2)(3)], Adrián Pousa [(1)], Enzo Rucci[(1)(2)], Silvana Gallo[(1)(2)], Erica Montes de Oca[(1)], Emmanuel Frati[(1)], Mariano Sánchez [(1)], María José Basgall[(1)(2)], Adriana Gaudiani[(4)]

[1]Instituto de Investigación en Informática LIDI (III-LIDI)
Facultad de Informática – Universidad Nacional de La Plata
50 y 115, La Plata, Buenos Aires
Comisión de Investigaciones Científicas de la Pcia. de Buenos Aires (CIC)
526 e/ 10 y 11 La Plata Buenos Aires

[2] CONICET – Consejo Nacional de Investigaciones Científicas y Técnicas
[3]CIC – Comisión de Investigación Científica de la Provincia de Buenos Aires
[4] Universidad Nacional de General Sarmiento
{mnaiouf, degiusti, ldgiusti, francoch, vsanz, apousa, erucci, sgallo, emontesdeoca,fefrati, msanchez, mjbasgall}@lidi.info.unlp.edu.ar, agaudi@ungs.edu.ar

RESUMEN

El eje central de la línea presentada son los temas de procesamiento paralelo y distribuido para HPC (fundamentos y aplicaciones). Interesa la construcción, evaluación y optimización de soluciones con algoritmos concurrentes, paralelos y distribuidos sobre diferentes plataformas de software y arquitecturas con múltiples procesadores (multicore, clusters de multicore, cloud y aceleradores como GPU, FPGA y Xeon Phi), los lenguajes y paradigmas de programación paralela (puros e híbridos), los modelos de representación de aplicaciones paralelas, los algoritmos de mapping y scheduling, el balance de carga, las métricas de evaluación de complejidad y rendimiento (speedup, eficiencia, escalabilidad, consumo energético), y la construcción de ambientes para la enseñanza de la programación concurrente y paralela.
Se propone aplicar los conceptos en problemas numéricos y no numéricos de cómputo intensivo y/o sobre grandes volúmenes de datos (búsquedas, simulaciones, n-body, imágenes, big data, reconocimiento de patrones, bioinformática, etc), con el fin de obtener soluciones de alto rendimiento.
En la dirección de tesis de postgrado existe colaboración con el grupo HPC4EAS (High Performance Computing for Efficient Applications and Simulation) del Dpto. de Arquitectura de Computadores y Sistemas Operativos de la Universidad Autónoma de Barcelona, y con el Departamento de Arquitectura de Computadores y Automática de la Universidad Complutense de Madrid, entre otros.
Palabras clave: Cómputo paralelo y distribuido de altas prestaciones. Algoritmos paralelos y distribuidos. Clusters. Multicore. GPU. Consumo energético. Balance de carga. Aplicaciones. Evaluación de performance.

CONTEXTO

La línea de I/D que se presenta en este trabajo es parte del Proyecto 11/F017 “Cómputo Paralelo de Altas Prestaciones. Fundamentos y Evaluación de rendimiento en HPC. Aplicaciones a Sistemas Inteligentes, Simulación y Tratamiento de Imágenes” del III-LIDI acreditado por el Ministerio de Educación, y de proyectos acreditados y subsidiados por la Facultad de Informática de la UNLP. Además, existe cooperación con Universidades de Argentina, Latinoamérica y Europa a través de proyectos acreditados por AECID, CyTeD, OEI y CIC y becas de Telefónica de Argentina. Asimismo, el Instituto forma parte del Sistema Nacional de Cómputo de Alto Desempeño (SNCAD).

1. INTRODUCCIÓN

El área de procesamiento paralelo se ha convertido en clave dentro de las Ciencias de la Computación, debido al creciente interés por el desarrollo de soluciones a problemas con muy alta demanda computacional y de almacenamiento, produciendo transformaciones profundas en las líneas de I/D [RAU10][KIR12].
El desafío se centra en cómo aprovechar las prestaciones obtenidas a partir de la evolución de las arquitecturas físicas. En esta línea de I/D la mayor importancia está en los algoritmos paralelos y en los métodos utilizados para su construcción y análisis a fin de optimizarlos.
Uno de los cambios de mayor impacto ha sido el uso de manera masiva de procesadores con más de un núcleo (*multicore*), produciendo plataformas distribuidas híbridas (memoria compartida y distribuida) y generando la necesidad de desarrollar sistemas operativos, lenguajes y algoritmos que las usen adecuadamente. También creció la incorporación de placas aceleradoras a los sistemas multicore constituyendo plataformas paralelas de memoria compartida con paradigma de programación propio asociado. Asimismo, los entornos de computación cloud introducen un nuevo foco desde el punto de vista del HPC, brindando un soporte “a medida” para la ejecución de aplicaciones sin la necesidad de adquirir el hardware.
La creación de algoritmos paralelos en arquitecturas multiprocesador no es un proceso directo [MCC12]. El costo puede ser alto en términos del esfuerzo de programación y el manejo de la concurrencia adquiere un

rol central en el desarrollo. Los pasos básicos para diseñar aplicaciones paralelas incluyen particionamiento, comunicación, aglomeración y mapeo de procesos a procesadores, y si bien en las primeras etapas el diseñador puede abstraerse de la máquina sobre la que ejecutará el algoritmo, para obtener buen rendimiento debe tenerse en cuenta la plataforma de destino. En las máquinas multiprocesador, se deben identificar las capacidades de procesamiento, interconexión, sincronización y escalabilidad [PAR09]. La caracterización y estudio de rendimiento del sistema de comunicaciones es de interés para la predicción y optimización de performance, así como la homogeneidad o heterogeneidad de los procesadores.

Muchos problemas algorítmicos se vieron impactados por las máquinas multicore y el 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, surgiendo así varios niveles de comunicación. Esto impacta sobre el desarrollo de algoritmos que aprovechen adecuadamente las arquitecturas, y motiva el estudio de performance en sistemas híbridos. 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 [MUR11].

Para algunos problemas ha crecido la utilización de placas aceleradoras, como pueden ser las unidades de procesamiento gráfico (GPU, graphic processing unit) de NVIDIA y AMD o los coprocesadores Xeon Phi de Intel [KIR12][JEF13]. Esto se debe a la capacidad que tienen estos dispositivos de alcanzar picos de rendimiento y cocientes de eficiencia energética superiores a los de la CPU a un menor costo. Por otro lado, el uso de FPGAs (Field Programmable Gate Array) se ha vuelto atractivo para HPC debido la evolución en su capacidad de cómputo, su bajo consumo energético y al desarrollo de nuevas herramientas de programación más familiares para el área [SET13].

La combinación de arquitecturas con múltiples núcleos con aceleradores dio lugar a plataformas híbridas con diferentes características [RUC16b]. Más allá del tipo de acelerador utilizado, la programación de esta clase de plataformas representa un verdadero desafío. Para lograr aplicaciones de alto rendimiento, los programadores deben enfrentar diversas dificultades como pueden ser: estudiar características específicas de cada arquitectura y aplicar técnicas de programación y optimización particulares para cada una de ellas, lograr un balance de carga adecuado entre los diferentes dispositivos de procesamiento y afrontar la ausencia de un estándares y de herramientas avanzadas para este tipo de sistemas.

Por otra parte, los avances en las tecnologías de virtualización han dado origen al paradigma de Cloud Computing, que se presenta como una alternativa a los tradicionales sistemas de cluster [EC213]. El uso de cloud para HPC presenta desafíos atractivos, brindando un entorno reconfigurable dinámicamente sin la necesidad de adquirir hardware, y es una excelente plataforma para testear escalabilidad de algoritmos aunque queda mucho por hacer en cuanto al diseño, lenguajes y programación

Métricas de evaluación del rendimiento y balance de carga

La diversidad de opciones vuelve complejo el análisis de performance de los Sistemas Paralelos, ya que los ejes sobre los cuales pueden compararse dos sistemas son varios. Existe un gran número de métricas para evaluar el rendimiento, siendo las tradicionales: tiempo de ejecución, speedup, eficiencia.

Por su parte, la *escalabilidad* permite capturar características de un algoritmo paralelo y la arquitectura en que se lo implementa. Posibilita 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.

El objetivo principal 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 tal que cada una tenga una cantidad de trabajo que demande aproximadamente el mismo tiempo, y esto es más complejo si hay heterogeneidad. Dado que el problema general de mapping es *NP*-completo, pueden usarse enfoques que dan soluciones subóptimas aceptables. Las técnicas de planificación a nivel micro (dentro de cada procesador) y macro (en un cluster) deben ser capaces de obtener buen balance de carga. Existen técnicas estáticas y dinámicas cuyo uso depende del conocimiento que se tenga sobre las tareas que componen la aplicación [DUM08].

Un aspecto de interés que se ha sumado como métrica es el del consumo energético requerido [BAL13]. Entre las aplicaciones de interés se encuentran las numéricas y no numéricas con alta demanda de cómputo.

2. LÍNEAS DE INVESTIGACIÓN, DESARROLLO E INNOVACIÓN

- Paralelización de algoritmos secuenciales. Diseño y optimización de algoritmos.
- Comparación de lenguajes y bibliotecas para procesamiento paralelo y distribuido.
- Estudio de complejidad de algoritmos paralelos, considerando multicore y heterogeneidad.
- Modelos y paradigmas de computación paralela. Modelo Map-reduce.
- Arquitecturas multicore y many-core. Arquitecturas FPGA.
- Arquitecturas híbridas (diferentes combinaciones de clusters, multicores y GPUs) y arquitecturas heterogéneas.

- Programación sobre modelos híbridos: pasaje de mensajes y memoria compartida en cluster de multicores, clusters de GPU, clusters multicore-GPU.
- Técnicas de programación sobre arquitecturas many-core (GPU y Xeon Phi) y FPGA.
- Técnicas para soluciones de HPC en cloud.
- Lenguajes y Estructuras de Datos para nuevas arquitecturas de cómputo paralelo.
- Mapping y scheduling de aplicaciones paralelas sobre distintas arquitecturas multiprocesador. Balance de carga estático y dinámico. Técnicas.
- Desarrollo de soluciones paralelas a problemas de cómputo intensivo y/o con grandes volúmenes de datos (búsquedas, simulaciones, n-body, aplicaciones científicas, big data, bioinformática) sobre diferentes modelos de arquitectura homogéneas y heterogéneas (multicores, clusters, clusters de multicore, GPU, Xeon Phi, FPGA y cloud).
- Evaluación de rendimiento, eficiencia energética y costo de programación de las diferentes soluciones implementadas teniendo en cuenta las arquitecturas y las herramientas de programación utilizadas.
- Ambientes para la enseñanza de programación concurrente

3. RESULTADOS OBTENIDOS/ESPERADOS

- Desarrollar y optimizar algoritmos paralelos sobre diferentes modelos de arquitectura. En particular, en aplicaciones numéricas y no numéricas de cómputo intensivo y tratamiento de grandes volúmenes de datos (big data).
- Utilizar arquitecturas híbridas que combinan memoria compartida y pasaje de mensajes, evaluando performance para distintos modelos de comunicación.
- Estudiar y comparar los lenguajes sobre las plataformas multiprocesador para diferentes modelos de interacción entre procesos.
- Investigar la paralelización en plataformas que combinan clusters, multicore y aceleradores. Comparar estrategias de distribución de trabajo teniendo en cuenta las diferencias en potencias de cómputo y comunicación, dependencia de datos y memoria requerida.
- Evaluar la performance (speedup, eficiencia, escalabilidad, consumo energético) de las soluciones propuestas. Analizar el rendimiento de soluciones paralelas a problemas con diferentes características (dependencia de datos, relación cómputo / comunicación, memoria requerida).
- Mejorar y adecuar las técnicas disponibles para el balance de carga (estático y dinámico) entre procesos a las arquitecturas consideradas.

En este marco, pueden mencionarse los siguientes resultados:

- Para la experimentación se han utilizado y analizado diferentes arquitecturas homogéneas o heterogéneas, incluyendo multicores, cluster de multicores (con 128 núcleos), GPU y cluster de GPU, Xeon Phi y FPGA.
- Se experimentó la paralelización en arquitecturas híbridas, con el objetivo de estudiar el impacto del mapeo de datos y procesos, así como de los lenguajes y librerías utilizadas.
- Respecto de las aplicaciones estudiadas y algoritmos implementados, se trabajó fundamentalmente con los siguientes problemas:

➢ **Best-first search (BFS) paralelo sobre multicore y cluster de multicore.** El algoritmo de búsqueda BFS es utilizado para resolver problemas combinatorios, para los cuales se requiere encontrar una secuencia de acciones que permita transformar una configuración inicial (problema) en una configuración final (solución). En especial, A* es una variante de BFS la cual permite encontrar soluciones de costo óptimo. Estos algoritmos requieren una alta capacidad de cómputo y gran cantidad de memoria, por esto su paralización es imprescindible. En los últimos años se ha re-impulsado el desarrollo de algoritmos paralelos BFS con el objetivo de aprovechar: (a) la potencia de cómputo de los procesadores *multicore* (b) la gran cantidad de RAM y potencia de cómputo de los *clusters de muticore.* En este sentido, HDA* [KIS13] El algoritmo de búsqueda BFS es utilizado para resolver problemas combinatorios, para los cuales se requiere encontrar una secuencia de acciones que permita transformar una configuración inicial (problema) en una configuración final (solución). En especial, A* es una variante de BFS la cual permite encontrar soluciones de costo óptimo. Estos algoritmos requieren una alta capacidad de cómputo y gran cantidad de memoria, por esto su paralización es imprescindible. En los últimos años se ha re-impulsado el desarrollo de algoritmos paralelos BFS con el objetivo de aprovechar: (a) la potencia de cómputo de los procesadores *multicore* (b) la gran cantidad de RAM y potencia de cómputo de los *clusters de muticore.* En este sentido, HDA* [KIS13] paraleliza A* sobre clusters utilizando MPI: cada procesador realiza una búsqueda cuasi-independiente utilizando un esquema de distribución de nodos basado en una función hash estándar. Otros autores [BUR10] adaptaron HDA* para máquinas multicore utilizando Pthreads: esta versión elimina el overhead extra del paso de mensajes entre procesadores (threads) en una arquitectura de memoria compartida y utiliza menor cantidad de memoria, ya que los threads comparten estructuras de datos comunes. Los mismos autores notaron que utilizar una función hash estándar para asignar nodos a procesadores genera alto overhead de comunicación entre threads. Por ende, propusieron AHDA*, una versión de HDA* para máquinas multicore que utiliza una función especial (función de abstracción) para asignar bloques de nodos a procesadores, en vez de nodos individuales. En una primera etapa de desarrollo, implementamos versiones propias de HDA* (HDA* MPI) [SAN16a] y HDA* para máquinas multicore (HDA* Pthreads) [SAN14]

[SAN15]. Ambas versiones incluyen técnicas para mejorar el rendimiento, respecto a las versiones originales. Luego presentamos Optimized AHDA* [SAN16b], una versión propia de AHDA* que incorpora una técnica para optimizar las transferencias de nodos entre threads, la cual permitió reducir la comunicación y contención. Con el objetivo de explotar eficientemente los recursos de un *cluster de multicore*, desarrollamos Hybrid HDA* (HHDA*), una versión híbrida de HDA* programada con MPI+Pthreads. El trabajo experimental demostró que HHDA* alcanza un rendimiento superior y consume menor cantidad de memoria, comparado con HDA* MPI. Estas mejoras permitieron a HHDA* resolver instancias más complejas del caso de estudio. Como línea de trabajo futuro se plantea adaptar los algoritmos antes mencionados para encontrar soluciones en menor tiempo comprometiendo la calidad de la solución.

- **Criptografía de grandes volúmenes de datos.** El encriptado y desencriptado de datos requiere un tiempo de cómputo adicional, que dependiendo de su tamaño puede ser muy alto. AES (Advanced Encryption Standard), es un algoritmo de cifrado simétrico por bloques que se ha convertido en estándar en 2002, y actualmente es el más ampliamente usado para codificar información. Este algoritmo se caracteriza por ser simple, rápido y por consumir pocos recursos. Sin embargo el tiempo de cifrar y descifrar grandes cantidades de datos es importante por lo que es oportuno aprovechar las posibilidades que brindan las arquitecturas multicores para reducir este tiempo. Para este propósito, las arquitecturas multicore actuales, como clusters de multicore y GPUs, proporcionan una forma de acelerar el cómputo de encriptar y desencriptar información logrando un excelente rendimiento [POU15].

- **Alineamiento de secuencias biológicas.** Esta operación consiste en comparar dos o más secuencias biológicas, como pueden ser las de ADN o las de proteínas, y resulta fundamental en investigaciones de la bioinformática y la biología molecular. El algoritmo de Smith-Waterman es considerado el método de alineamiento más preciso. Desafortunadamente, este algoritmo resulta costoso debido a su complejidad computacional cuadrática mientras que la situación se agrava aún más a causa del crecimiento exponencial de datos biológicos en los últimos años [RUC16a]. El reciente surgimiento de aceleradores en HPC (GPU, Xeon Phi, FPGA, entre otros) da la oportunidad de acelerar los alineamientos sobre hardware comúnmente disponible a un costo accesible. En primer lugar, se continuó el estudio preliminar del empleo del modelo de programación OpenCL sobre FPGAs para acelerar el alineamiento de secuencias de proteínas [RUC15]. Se desarrollaron diferentes soluciones paralelas para arquitecturas heterogéneas basadas en FPGA de forma de encontrar la configuración más beneficiosa. Se analizó el rendimiento, la eficiencia energética y el costo de programación de cada solución y se los comparó con el de otras implementaciones basadas en multicores, Xeon Phi y GPUs [RUC16b]. Aprovechando el conocimiento adquirido, se adaptó el código desarrollado para computar alineamientos de secuencias largas de ADN [RUC17]. A futuro, interesa explorar el uso de nuevas generaciones de procesadores y aceleradores para esta aplicación, como pueden ser los procesadores Xeon Phi Knights Landing de Intel.

- **Simulación distribuida de modelos orientados al individuo.** El modelado orientado al individuo resulta de gran interés ya que permite analizar y extraer conclusiones acerca de un sistema a través de la simulación de la interacción de sus individuos. No obstante, a medida que los modelos incorporan más características del sistema se vuelven más complejos y en consecuencia se necesita mayor cantidad de cómputo y comunicación para lograr resultados significativos. La aparición de arquitecturas distribuidas y procesadores con varios núcleos ha favorecido el desarrollo de dichos modelos a gran escala utilizando técnicas de simulación distribuida. Lograr mejoras en los tiempos de ejecución y en la eficiencia de dichas aplicaciones es esencial. Con el fin de mejorar la distribución de la carga entre los procesos lógicos de la simulación se pretende analizar el desempeño del simulador en cuanto al balance de cómputo, ya que al trabajar con grandes cantidades de individuos, y debido a la cantidad de cómputo asociada a la selección de vecinos para realizar los desplazamientos, se producen desbalances en cuanto a la cantidad de trabajo de cada uno de los diferentes procesos lógicos. Dichos desbalances obstaculizan el aprovechamiento de la arquitectura disponible, por lo que se deben estudiar diferentes técnicas de balance de cómputo y desarrollar las soluciones que aporten dichas mejoras.

- **Problemas de tipo N-body**. Se utilizaron las plataformas de memoria compartida GPU y cluster de multicore para la resolución de problemas con alta demanda computacional del tipo N-body. Se emplearon diferentes modelos de comunicación: memoria compartida (Pthreads en CPU y CUDA en GPU), pasaje de mensajes (MPI) y soluciones híbridas (MPI-Pthreads). Se han mostrado los beneficios del uso de la GPU en problemas con características similares al caso planteado. El trabajo experimental ha dado como resultado una buena aceleración obtenida utilizando cluster de GPU. Además, se observó claramente que el uso del cluster de GPU logró una aceleración proporcional al speedup conseguido con el cluster de CPU pero con tiempos de ejecución significativamente menores [MON14]. También se han desarrollado diferentes alternativas de distribución de trabajado usando un Cluster de GPUs heterogéneas [MON16].

Actualmente, se ha puesto énfasis en el análisis del consumo energético de esta experimentación.

- **Problemas de simulación relacionados con fenómenos naturales (inundaciones)**. Análisis de diferentes soluciones para la paralelización de este tipo de aplicaciones que son intensivas en cómputo; y el tiempo de ejecución y la performance alcanzable son críticas dado que los resultados que se esperan determinarán alertas y toma de decisiones. La utilización de escenarios de simulación en entornos donde interesa estudiar el comportamiento en situaciones de desastres producidos por fenómenos naturales como las inundaciones. En este ámbito se avanza en dos temas: (1) La implementación de un método de sintonización de un simulador de inundaciones en ríos de llanura, mediante la técnica de simulación paramétrica. El proceso requiere lanzar miles de escenarios de simulación hasta encontrar un conjunto ajustado de parámetros de entrada del simulador. La experimentación se lleva a cabo con un modelo master-worker sobre un cluster [GAU15]. (2) En colaboración con el Laboratorio de Hidrología de la UNLP se comenzó con la paralelización de la simulación de inundaciones producidas por lluvias (en particular en el ámbito de la ciudad de La Plata, donde una corrida "standard" es del orden de las 8 hs), a fin de reducir el tiempo de ejecución a pocos minutos y permitir establecer un sistema de alertas [GAU16].

- **Ambientes para la enseñanza de concurrencia.** Se desarrolló el entorno R-INFO para la enseñanza de programación concurrente y paralela a partir de cursos iniciales en carreras de Informática. Incluye un entorno visual que representa una ciudad en la que pueden definirse varios robots que interactúan. Combina aspectos de memoria compartida y distribuida mediante instrucciones para bloquear y liberar esquinas de la ciudad y el concepto de pasaje de mensajes a través de primitivas de envío y recepción. Además, se incluyen los conceptos de heterogeneidad (diferentes velocidades de los robots) y consumo energético [DEG16a]. Se ha integrado con el uso de robots físicos (Lego Mindstorm 3.0) que ejecutan en tiempo real las mismas instrucciones que los robots virtuales y se comunican con el entorno mediante bluetooth [DEG14]. Se ha ampliado para incorporar conceptos básicos de computación en la nube (Cloud Computing) [DEG16b].

- **Aplicaciones en Big Data.** En esta línea se está trabajando en la aplicación de técnicas de Machine Learning sobre grandes volúmenes de datos utilizando el framework para el procesamiento paralelo y distribuido de grandes cantidades de datos, llamado MapReduce. Se está llevando a cabo la evaluación de rendimiento de los diferentes algoritmos implementados para las soluciones de procesamiento en batch y en streaming aplicados a problemas de Big Data [BAS16a][BAS16b]. Se trata de una línea incipiente en el grupo.

4. FORMACIÓN DE RECURSOS HUMANOS

Dentro de la temática de la línea de I/D se concluyó 1 tesis doctorales, 3 Trabajos Finales de Especialización y 2 Tesinas de Grado de Licenciatura. Se encuentran en curso en el marco del proyecto 7 tesis doctorales, 2 de maestría, 3 trabajos de Especialización y 2 Tesinas.

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

Hay cooperación con grupos de otras Universidades del país y del exterior, y tesistas de diferentes Universidades realizan su trabajo con el equipo del proyecto.

5. BIBLIOGRAFÍA

[BAL13] Balladini J., Rucci E., De Giusti A., Naiouf M., Suppi R., Rexachs D., Luque E. "Power Characterisation of Shared-Memory HPC Systems". Computer Science & Technology Series – XVIII Argentine Congress of Computer Science Selected Papers. ISBN 978-987-1985-20-3. Pp. 53-65. EDULP, La Plata (Argentina), 2013

[BAS16a] M. J. Basgall, W. Hasperué, M. Naiouf. "Data stream treatment using sliding windows with MapReduce". Journal of Computer Science & Technology; vol. 16, no. 2. Noviembre 2016. Pp. 76-83. ISSN: 1666-6038.

[BAS16b] M. J. Basgall, W. Hasperué, C. A. Estrebou, M. Naiouf. "Clustering de un flujo de datos usando MapReduce". Proceedings del XXII Congreso Argentino de Ciencias de la Computación (CACIC 2016) – XIII Workshop Bases de datos y Minería de Datos. Octubre 2016. Pp 682-691.

[BUR10] Burns E, Lemons S, Ruml W, Zhou R. "Best First Heuristic Search for Multicore Machines". Journal of Artificial Intelligence Research, Vol.39, No.1, pp. 689-743, 2010.

[DEG14] De Giusti L., Leibovich F., Sanchez M., Chichizola F., Naiouf M., De Giusti A. "Herramienta interactiva para la enseñanza temprana de Concurrencia y Paralelismo: un caso de estudio", Procs XX Congreso Argentino de Ciencias de la Computación – Workshop de Innovación en Educación. Octubre 2014. Pp 133-140. ISBN: 978-987-3806-05-6

[DEG16a] L. C. De Giusti, F. Leibovich, F. Chichizola, M. Naiouf. "Teaching Concurrency and Parallelism Concepts with CMRE". Journal of Computer Science & Technology; vol. 16, no. 2. Noviembre 2016. Pp 95-100. ISSN: 1666-6038.

[DEG16b] L. C. De Giusti, F. Chichizola, S. Rodriguez Eguren, M. Sanchez, J. M. Paniego, A. E. De Giusti. "Introduciendo conceptos de Cloud Computing utilizando el entorno CMRE". Proceedings del XXII Congreso Argentino de Ciencias de la Computación (CACIC 2016)

– Workshop de Innovación en Educación en Informática. Octubre 2016. Pp 1357-1365.
[DUM08] Dummler J., Ruaber T., Runger G., Mapping Algorithms for Multiprocessor Tasks on Multi-Core Clusters, Proc. 2008 International Conference on Parallel Processing IEEE CS 2008.
[EC213] Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/es/ec2/. Febrero 2013.
[GAU15] Adriana Gaudiani. "Simulación y optimización como metodología para mejorar la calidad de la predicción en un entorno de simulación hidrográfica". Tesis de Doctorado en Ciencias Informáticas (Facultad de Informática – UNLP). 2015.
[GAU16] Adriana Gaudiani, Emilio Luque, Pablo García, Mariano Re, Marcelo Naiouf, Armando De Giusti. "How a Computational Method Can Help to Improve the Quality of River Flood Prediction by Simulation". Advances and New Trends in Environmental and Energy Informatics (part V). ISBN 978-3-319-23455-7. Pp337- 351. 2016.
[JEF13] Jeffers, James; Reinders, James. "Intel Xeon Phi Coprocessor High Performance Programming", Morgan Kaufmann, 2013.
[KIR12] Kirk D., Hwu W. "Programming Massively Parallel Processors, second edition: A Hands-on Approach. Morgan-Kaufmann. 2012.
[KIS13] A. Kishimoto, A. Fukunaga, and A. Botea, "Evaluation of a simple, scalable, parallel best-first search strategy," Artificial Intelligence, vol. 195, pp. 222–248, 2013.
[MCC12] McCool, Michael. "Structured Parallel Programming: Patterns for Efficient Computation", Morgan Kaufmann, 2012
[MON14] E. Montes de Oca, L. De Giusti, F. Chichizola, A. De Giusti, M. Naiouf. "Utilización de Cluster de GPU en HPC. Un caso de estudio". Proceedings del XX Congreso Argentino de Ciencias de la Computación (CACIC 2014), pp. 1220-1227, 2014.
[MON16] E. Montes de Oca, L. C. De Giusti, F. Chichizola, A. E. De Giusti, M. Naiouf. "Análisis de uso de un algoritmo de balanceo de carga estático en un Cluster Multi-GPU Heterogéneo". Proceedings del XXII Congreso Argentino de Ciencias de la Computación (CACIC 2016), pp. 159-168, 2016.
[MUR11] Muresano Cáceres R. "Metodología para la aplicación eficiente de aplicaciones SPMD en clústers con procesadores multicore" Ph.D. Thesis, UAB, Barcelona, España, Julio 2011.
[PAR09] Parashar M., Li Xiaolin, Chandra Sumir, "Advanced Computational Infrastructures for Parallel and Distributed Applications", Wiley-Interscience, 2009 ISBN-10: 0470072946.
[POU15] Adrian Pousa, Victoria Sanz, Armando De Giusti. "Comparación de rendimiento de algoritmos de cómputo intensivo y de acceso intensivo a memoria sobre arquitecturas multicore. Aplicación al algoritmo de criptografía AES". Actas del XXI Congreso Argentino de Ciencias de la Computación, pp. 163-172, 2015.
[RAU10] Rauber T., Rünger G. "Parallel programming for multicore and cluster systems". Springer. 2010. [RUC15] "Smith-Waterman Protein Search with OpenCL on FPGA". Enzo Rucci, Armando De Giusti, Marcelo Naiouf, Carlos García Sanchez, Guillermo Botella Juan, Manuel Prieto-Matías. Proceedings of 2014 IEEE Symposyum on Parallel and Distributed Processing with Applications (ISPA). 20 al 22 de Agosto de 2015.
Helsinki, Finlandia. ISBN: 978-1-4673-7952-6. Pp. 208-213. DOI: 10.1109/Trustcom.2015.634.
[RUC16a] "State-of-the-art in Smith-Waterman Protein Database Search". Enzo Rucci, Armando De Giusti, Marcelo Naiouf, Carlos García Sanchez, Guillermo Botella Juan, Manuel Prieto-Matías. Big Data Analytics in Genomics. Springer, pp. 197-223, 2016.
[RUC16b] "OSWALD: OpenCL Smith–Waterman on Altera's FPGA for Large Protein Databases". Enzo Rucci, Armando De Giusti, Marcelo Naiouf, Carlos García Sanchez, Guillermo Botella Juan, Manuel Prieto-Matías. International Journal of High Performance Computing Applications, Online first, 2016.
[RUC16c] Rucci, Enzo. "Evaluación de rendimiento y eficiencia energética de sistemas heterogéneos para bioinformática". Tesis de Doctorado en Ciencias Informáticas, Facultad de Informática, UNLP. 2016.
[RUC17] "Accelerating Smith-Waterman Alignment of Long DNA Sequences with OpenCL on FPGA". Enzo Rucci, Armando De Giusti, Marcelo Naiouf, Carlos García Sanchez, Guillermo Botella Juan, Manuel Prieto- Matías. Lecture Notes in Computer Science, En prensa, 2017.
[SAN14] V. Sanz, A. De Giusti, and M. Naiouf, "On the Optimization of HDA* for Multicore Machines. Performance Analysis," in PDPTA'14 (The 2014 International Conference on Parallel and Distributed Processing Techniques and Applications), Las Vegas, 2014, pp. 625-631.
[SAN15] V. Sanz, A. De Giusti, and M. Naiouf, "Performance tuning of the HDA* algorithm for multicore machines." in Computer Science & Technology Series - XX Argentine Congress of Computer Science - Selected Papers. Argentina: EDULP, 2015, pp. 51-62. [SAN16b] V. Sanz, A. De Giusti, and M. Naiouf, "Improving Hash Distributed A* for Shared Memory Architectures Using Abstraction," in ICA3PP-2016 - The 16th International Conference on Algorithms and Architectures for Parallel Processing, Granada, 2016, pp. 431 - 439.
[SAN16a] V. Sanz, A. De Giusti, and M. Naiouf, "Scalability analysis of Hash Distributed A* on commodity cluster: results on the 15-puzzle problem.," in PDPTA'16 - The 22nd International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, 2016, pp. 221-227.
[SET13] High-performance Dynamic Programming on FPGAs with OpenCL. Sean Settle. 2013 IEEE High Performance Extreme Computing Conf (HPEC '13), 2013.