Fundamentos de cómputo paralelo y distribuido de altas prestaciones. Construcción y evaluación de aplicaciones.

Marcelo Naiouf[(1)], Armando De Giusti[(1)(2)], Laura De Giusti[(1)], Franco Chichizola[(1)], Victoria Sanz[(1)(2)], Fabiana Leibovich[(1)], Enzo Rucci[(1)(2)], Silvana Gallo[(1)(2)], Erica Montes de Oca[(1)], Emmanuel Frati[(1)], Mariano Sanchez[(1)], Adriana Gaudiani[(3)]

[1]**Instituto de Investigación en Informática LIDI (III-LIDI)**
Facultad de Informática – Universidad Nacional de La Plata
[2] **CONICET – Consejo Nacional de Investigaciones Científicas y Técnicas**
[3] **Universidad Nacional de General Sarmiento**
{mnaiouf, degiusti, ldgiusti, francoch, vsanz, fleibovich, erucci, sgallo, emontesdeoca, fefrati, msanchez}@lidi.info.unlp.edu.ar, agaudi@ungs.edu.ar

CONTEXTO

Se presenta una línea de Investigación que 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.

Existe cooperación con Universidades de Argentina, Latinoamérica y Europa a través de proyectos acreditados por AECID, CyTeD, y OEI, y becas de Telefónica de Argentina. Asimismo, el Instituto forma parte del Sistema Nacional de Cómputo de Alto Desempeño (SNCAD).

RESUMEN

El eje central de la línea de I/D lo constituye el estudio de temas de procesamiento paralelo y distribuido para cómputo de altas prestaciones, en lo referente a los fundamentos y a las aplicaciones. Incluye problemas de software asociados con el uso de arquitecturas multiprocesador.

Interesa la construcción, evaluación y optimización de soluciones usando algoritmos concurrentes, paralelos y distribuidos sobre diferentes plataformas de software y arquitecturas con múltiples procesadores (multicore, clusters de multicore, GPU, cloud), los lenguajes y paradigmas de programación paralela (puros e híbridos a distintos niveles), los modelos de representación de aplicaciones paralelas, los modelos y paradigmas paralelos, los algoritmos de asignación de procesos a procesadores (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. Las arquitecturas pueden ser homogéneas o heterogéneas.

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, entre otros), con el fin de obtener soluciones de alto rendimiento.

El proyecto coordina con otros dos en curso en el III-LIDI, relacionados con Arquitecturas Distribuidas y Paralelas y Sistemas de Software Distribuido. Existe colaboración, entre otros, con el grupo HPC4EAS (High Performance Computing for Efficient Applications and Simulation) del Departamento de Arquitectura de Computadores y Sistemas Operativos de la Universidad Autónoma de Barcelona en la dirección de tesis de postgrado.

Palabras clave: *Cómputo paralelo y distribuido de altas prestaciones. Algoritmos paralelos y distribuidos. Clusters. Multicore. GPU. Balance de carga. Aplicaciones. Evaluación de performance. Consumo energético.*

1. INTRODUCCION

Debido al interés por el desarrollo de soluciones a problemas con creciente demanda computacional y de almacenamiento, el procesamiento paralelo y distribuido se ha convertido en un área clave dentro de la Ciencia de la Computación, produciendo transformaciones en las líneas de I/D [RAU10][HAG10][PAC11][COO12][KIR12].

Ligado a las mejoras en la evolución de las arquitecturas físicas, el desafío se centra en cómo aprovechar sus prestaciones. En este sentido, interesa realizar I/D en la especificación, transformación, optimización y evaluación de algoritmos, En esta línea de I/D la mayor importancia está en los algoritmos paralelos y en los métodos utilizados para su construcción y análisis [QIU08].

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. También ha surgido la utilización de arquitecturas many-core como las placas gráficas de uso general como máquinas paralelas de memoria compartida, lo que constituye una plataforma con un paradigma de programación propio asociado. Asimismo, los entornos de computación cloud introducen un nuevo foco desde el punto de vista de la computación de altas prestaciones, brindando un soporte "a medida" para la ejecución de aplicaciones sin la necesidad de adquirir hardware.

La creación de algoritmos paralelos en arquitecturas multiprocesador, o la paralelización de un algoritmo secuencial, no es un proceso directo [MCC12]. El costo puede ser alto en términos del esfuerzo de programación

[SHA08], 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. Si bien en las primeras etapas el diseñador puede abstraerse de la máquina sobre la cual ejecutará el algoritmo, para obtener buen rendimiento en general debe tenerse en cuenta la plataforma de destino, dando lugar al concepto de *sistema paralelo* como combinación de hardware y software.

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 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 (2 cores del mismo chip), Inter CMP (2 cores que radican en distintos chips pero en el mismo nodo), e Inter Nodo (2 cores de 2 nodos distintos).

Esto impacta sobre el desarrollo de algoritmos que aprovechen adecuadamente las arquitecturas, y motiva el 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 [MUR11].

Para algunos problemas ha crecido la utilización de arquitecturas many-core como las placas gráficas de uso general (GPGPU, o *general purpose graphic processing unit*) como máquinas paralelas de memoria compartida [PIC11][KIR12]. Esto se debe a la gran cantidad de núcleos de procesamiento disponibles, buena performance y costo accesible. Dentro de los lenguajes asociados pueden mencionarse CUDA y OpenCL [LUE08][NOT09][NVI08].

La combinación de arquitecturas con múltiples núcleos dio lugar a plataformas híbridas con diferentes características [CHA11][LIN11A][LIN11B]. Los desafíos son múltiples, sobre todo en lo referido a las estrategias de distribución de datos y procesos, que necesitan ser investigadas a fin de optimizar la performance. Una tentencia promisoria es la que brinda Intel a partir de los procesadores MIC (Many Integrated Core Architecture), permitiendo utilizar métodos y herramientas estándar de programación con altas prestaciones. [JEF13]

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][OPE13]. El uso de cloud para HPC presenta desafíos atractivos de I/D, brindando un entorno reconfigurable dinámicamente sin la necesidad de adquirir hardware, y es una excelente plataforma para testear escalabilidad de algoritmos. Sin embargo, aún 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.

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.

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.

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 tal que cada una tenga una cantidad de trabajo que demande aproximadamente el mismo tiempo. Esto es más complejo si los procesadores (y comunicaciones) son heterogéneos. Dado que el problema general de mapping es *NP*-completo, pueden usarse enfoques que brindan soluciones subóptimas aceptables [OLI08]. 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 [LIU07][DUM08].

Evaluación de performance. Aplicaciones

Es importante referirse a un algoritmo paralelo mencionando el modelo de computación para el que fue diseñado. Uno de los objetivos en la definición del modelo es la posibilidad de predicción de performance que brinde el mismo, teniendo en cuenta conceptos tales como comunicación, sincronización y arquitectura física. El desarrollo de nuevos modelos requiere caracterizar el contexto de comunicaciones entre los procesadores y la asociación entre los algoritmos, el paradigma de cómputo paralelo elegido y la arquitectura de soporte.

En 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, eficiencia y escalabilidad. Un aspecto de interés que se ha sumado como métrica es el del consumo energético requerido [BAL12]

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.
- Estudio de complejidad de algoritmos paralelos, considerando multicore y heterogeneidad.
- Modelos y paradigmas de computación paralela. Modelo Map-reduce.
- Modelos de representación y predicción de performance de algoritmos paralelos.
- Arquitecturas multicore y many-core. Multithreading en multicore. Arquitecturas tipo MIC.
- Multiprocesadores distribuidos.
- Arquitecturas híbridas (diferentes combinaciones de 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)
- 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.
- Análisis de los problemas de migración y asignación de procesos y datos a procesadores.
- Evaluación de performance prestacional de algoritmos paralelos
- Análisis de consumo y eficiencia energética en particular en relación con clases de instrucciones y algoritmos paralelos.
- Ambientes para la enseñanza de programación concurrente
- Desarrollo de soluciones paralelas a problemas de cómputo intensivo y/o con grandes volúmenes de datos (búsquedas, simulaciones, n-body, aplicaciones científicas, "big data"). sobre diferentes modelos de arquitectura homogéneas y heterogéneas (multicores, clusters, clusters de multicore, GPU y cloud).

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 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.
- Estudiar y desarrollar modelos de representación de aplicaciones paralelas y distribuidas y los algoritmos de mapeo (estático y dinámico).
- Desarrollar algoritmos paralelos sobre GPU. Para problemas regulares y con alta demanda de cómputo, comparar los resultados con otras plataformas.
- Estudiar el impacto producido por los modelos de programación, lenguajes y algoritmos sobre el consumo y la eficiencia energética.
- Estudiar los modelos de predicción y evaluación de performance con diferentes paradigmas de interacción entre procesos, en esquemas multicore, cluster de multicore, GPU y cloud. Proponer las adecuaciones necesarias.
- Investigar la paralelización en plataformas que combinan multicore y GPU, o que disponen de más de una GPU. 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).
- Determinar la mejor configuración de soporte multiprocesador (puras e híbridas) para diferentes tipos de problemas resueltos con aplicaciones paralelas para HPC.
- Emplear experimentalmente contadores de hardware orientados a la detección de fallas de concurrencia y evaluación del rendimiento.
- 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 utilizaron y analizaron diferentes arquitecturas homogéneas o heterogéneas,

incluyendo multicores, cluster de multicores con 128 núcleos y GPU.

- 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 los aplicaciones estudiadas y algoritmos implementados, se trabajó fundamentalmente con los siguientes problemas:

➢ **Best-first search paralelo sobre multicore y cluster de multicore:** El algoritmo de búsqueda A* (variante de Best-First Search) es utilizado como base para resolver problemas combinatorios y de planificación, donde se requiere encontrar una secuencia de acciones que minimicen una función objetivo para transformar una configuración inicial (problema a resolver) en una configuración final (solución). El alto requerimiento de memoria y cómputo causados por el crecimiento exponencial o factorial del grafo generado dinámicamente hacen imprescindible su paralelización, que permite beneficiarse de: (a) la gran cantidad de RAM y potencia de cómputo que provee un *cluster*, (b) la potencia de cómputo que proveen los procesadores *multicore*, (c) ambas características en caso de *cluster de muticore.* Tomando como caso de estudio el problema del N-Puzzle, se implementó el A* secuencial para resolverlo, y dos versiones del paralelo Hash Distributed A* (HDA*[KIS12] [BUR10]) que realiza el balance de carga de los nodos generados del grafo mediante una función de Hash: (1) una versión utiliza MPI, haciendo posible su ejecución tanto sobre arquitecturas con memoria distribuida y memoria compartida, y (2) otra versión utilizando Pthreads, para su ejecución sobre un multiprocesador con memoria compartida, que elimina ciertas ineficiencias respecto a (1) cuando la ejecución se realiza sobre memoria compartida, como son las replicaciones de datos entre procesos de estructuras comunes utilizadas y la serialización de datos para la comunicación de nodos entre procesos. Interesa realizar un análisis comparativo del rendimiento alcanzado por HDA*-MPI y HDA*-Pthreads sobre un multicore y el desarrollo de una versión de HDA* híbrida (Pthreads + MPI) para obtener mayor eficiencia al utilizar un cluster de multicore con respecto a la versión HDA*-MPI.

➢ ***Diagonalización de matrices por el método de Jacobi sobre arquitecturas multicore***. El método de Jacobi para diagonalizar matrices simétricas tiene aplicaciones en áreas como biometría, visión artificial, procesamiento digital de señales, entre otros. El incremento en el volumen de datos de entrada provoca un aumento significativo en el tiempo de cómputo. La combinación de librerías de álgebra lineal optimizadas para la arquitectura subyacente, junto con la potencia que brinda un multicore y herramientas de programación paralela para dicha arquitectura permiten reducir el tiempo de ejecución. Se abordó el análisis del problema, y se estudiaron distintas implementaciones del algoritmo secuencial y optimizaciones posibles, la adaptación para usar una implementación de la API BLAS (Basic Linear Algebra Subprograms) optimizada para la arquitectura, y se implementó un algoritmo paralelo utilizando OpenMP. Se realizó un estudio de los tiempos de ejecución de las soluciones secuenciales, observando un mejor rendimiento para los algoritmos que hacen uso de librerías optimizadas para álgebra lineal (ATLAS). Se analizó el rendimiento (speedup, eficiencia) obtenido por el algoritmo paralelo propuesto sobre un multicore, a medida que se incrementa el volumen de datos de entrada (tamaño de la matriz) y al aumentar la cantidad de threads /cores, demostrando la escalabilidad del sistema paralelo propuesto [SAN12]. Como líneas de trabajo futuro se plantea la migración del algoritmo paralelo para ejecutar sobre GPU, el estudio de la escalabilidad sobre dicha arquitectura, y el análisis del consumo energético de los algoritmos paralelos propuestos.

➢ **Aplicaciones con paralelismo de datos**. Se continuó investigando y experimentando en la paralelización de aplicaciones en cluster de multicores, tomando como caso de estudio el problema de la multiplicación de matrices y centrándose en el aprovechamiento de la jerarquía de memoria, particularmente en el uso de la Caché L1. Se estudió la mejora introducida por las soluciones analizando escalabilidad en dos sentidos, al incrementar tanto el tamaño del problema como la cantidad de núcleos en la experimentación. Se implementaron diferentes soluciones híbridas (pasaje de mensajes y memoria compartida utilizando MPI + OpenMP), y se realizó la comparación de performance alcanzable por algoritmos que aprovechan eficientemente el uso de la cache L1 frente a otros que no lo hacen. Para este último análisis se compararon una solución que divide la matriz resultante en bloques para ser procesados al mismo tiempo que una segunda solución en la que las matrices de entrada son divididas en filas de bloques. Al aumentar el tamaño del problema, la primera solución alcanza mejor performance dado que aprovecha la localidad espacial y temporal de la cache L1 más eficientemente que la segunda y esto fue verificado mediante el uso de una herramienta de profiling (perf), en la cual se midieron la tasa de fallos de Caché L1 de ambas soluciones. [LEI13].

➢ ***Análisis de Secuencias de ADN:*** el alineamiento de secuencias tiene múltiples aplicaciones dentro de la bioinformática, tanto para la búsqueda de patrones entre secuencias de aminoácidos y nucleótidos como para la búsqueda de relaciones filogenéticas entre organismos. Entre los métodos existentes, el algoritmo de Smith-Waterman (SW) suele ser el más utilizado. Debido a su alta complejidad computacional, diversas heurísticas fueron desarrolladas para reducir el tiempo de ejecución pero a expensas de disminuir la precisión de los resultados. Con la reciente aparición de tecnologías de aceleradoras y de arquitecturas many-core, como las placas GPUs y los coprocesadores Intel MIC, existe la oportunidad de acelerar el algoritmo SW sobre hardware comúnmente disponible a un costo accesible.

- ***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 conveniente su paralelización [STU88]. Se desarrolló un algoritmo híbido MPI-OpenMP y se lo probó sobre una arquitectura de cluster de multicores. Se estudió el desempeño del algoritmo para diferente número de procesadores y de cargas de trabajo (teniendo cuenta métricas de rendimiento como speedup y eficiencia) además de realizar un análisis de escalabilidad [RUC13].

- ***Simulación distribuida de modelos orientados al individuo.*** La simulación distribuida de altas prestaciones de Modelos Orientados al Individuo es de gran interés en el ámbito científico ya que permite analizar y extraer conclusiones acerca de un sistema modelado a través de la simulación de los individuos. Pero, en consecuencia, implica grandes necesidades de cómputo y comunicación para lograr resultados cercanos a la realidad simulada. Lograr mejoras en la eficiencia de dichas aplicaciones y hacer un buen aprovechamiento de las diferentes arquitecturas existentes es un desafío que continúa vigente. El trabajo desarrollado es una evolución de los presentados con anterioridad, y se ha realizado la aplicación de la herramienta *hwloc* (Hardware Locality) [GAL13] sobre el simulador distribuido de un modelo *fishschools*, con el fin de mejorar la distribución de los procesos lógicos de la simulación de modo tal que beneficie la comunicación entre los mismos. Actualmente, se encuentra en desarrollo la monitorización y análisis del subsistema de comunicaciones del simulador distribuido para realizar una propuesta que permita reducir el overhead de las mismas y plantear soluciones que obtengan una mejor eficiencia de simulación.

- ***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 (Pthread en CPU y CUDA en GPU), pasaje de mensajes (MPI) y soluciones híbridas (MPI-Pthread). Además, se analizó el problema desde el punto de vista energético, introduciendo los fundamentos de consumo energético. Se han mostrado los beneficios del uso de la GPU en problemas con características similares al caso planteado. Los tiempos de ejecución obtenidos son considerablemente inferiores comparados con las soluciones implementadas en CPU. Los experimentos realizados desde el punto de vista del consumo energético favorecen el uso de la GPU en tales problemas [KIR12] [MON13]. Actualmente se trabaja en migrar la solución a cluster de GPU.

- ***Ambientes para la enseñanza de concurrencia.*** Se desarrolló el entorno R-INFO para la enseñanza de programación concurrente 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. Actualmente se trabaja en la integración con robots físicos del tipo Lego Mindstorm 3.0 [DEG13].

4. FORMACION DE RECURSOS HUMANOS

Dentro de la temática de la línea de I/D se concluyeron 5 Trabajos Finales de Especialización y 1 Tesina de Grado de Licenciatura. Se encuentran en curso en el marco del proyecto 9 tesis doctorales, 2 de maestría, 2 trabajos de Especialización y 3 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.

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

5. BIBLIOGRAFIA

[BAL12] Balladini J., Rucci E., De Giusti A., Naiouf M., Suppi R., Rexachs D., Luque E. "Power Characterisation of Shared-Memory HPC Systems". XII Workshop de Procesamiento Distribuido y Paralelo. CACIC 2012. ISBN: 978987-1648-34-4. Pág. 316-326. Bahía Blanca, Buenos Aires, Argentina, Octubre 2012.

[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.

[CHA07] Chapman B., The Multicore Programming Challenge, Advanced Parallel Processing Technologies; LNCS, Vol. 4847, p3, Springer, November 2007.

[CHA11] Chao-Tung Yang, Chih-Lin Huang, Cheng-Fang Lin, "Hybrid CUDA, OpenMP, and MPI parallel programming on multicore GPU Clusters", Computer Physics Communications 182 (2011) 266–269, Elsevier.

[COO12] Cook, Shane. "CUDA Programming: A Developer's Guide to Parallel Computing with GPUs (Applications of GPU Computing Series)", Morgan Kaufmann, ISBN-13: 978-0124159334, 2012

[DEG13] De Giusti L., Leibovich F., Sanchez M., Chichizola F., Naiouf M., De Giusti A. "Desafíos y herramientas para la enseñanza temprana de concurrencia y paralelismo", Proceedings Congreso Argentino de Ciencias de la Computación 2013. Mar del Plata, Argentina.

[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.

[GAL13] Gallo S., Borges F., Suppi R., Luque E., De Giusti L., Naiouf M.. "Mejoras en la eficiencia mediante *Hardware Locality* en la simulación distribuida de modelos orientados al individuo". CACIC2013. ISBN: 978-987-23963-1-2. Pág. 244-253. Octubre 2013.

[HAG10] Hager, Georg. Introduction to High Performance Computing for Scientists and Engineers (Chapman & Hall/CRC Computational Science), CRC Press, ISBN-13: 978-14398119241 edition, 2010.

[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.

[KIS12] Kishimoto A., Fukunaga A., Botea A. "Evaluation of a Simple, Scalable, Parallel Best-FirstSearch Strategy", Arxivpreprint: arXiv 1201.3204, 2012.

[Lei13] Leibovich F., De Giusti L., Naiouf M., Chichizola F., Tinetti F. G., De Giusti A. "Análisis del impacto de la jerarquía de memoria en clusters de multicores utilizando contadores de hardware". Congreso Español de Informática, Jornadas Sarteco 2013. ISBN:978-84-695-8330-2. Págs: 306-311. 2013.

[LIN11A] Lingyuan Wang, Miaoqing Huang, Vikram K. Narayana, Tarek El-Ghazawi, "Scaling Scientific Applications on Clusters of Hybrid"Multicore/GPU Nodes". CF '11 Proceedings of the 8th ACM International Conference on Computing Frontiers. USA 2011.

[LIN11B] Lingyuan Wang, Miaoqing Huang, and Tarek El-Ghazawi. "Towards efficient GPU sharing on multicore processors". In Proceedings of the second international workshop on Performance modeling, benchmarking and simulation of high performance computing systems (PMBS '11). ACM, New York, NY, USA, 23-24.

[LUE08] Luebke D. "Cuda: Scalable parallel programming for high-performance scientific computing". 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro, 2008.

[MCC12] McCool, Michael. "Structured Parallel Programming: Patterns for Efficient Computation", Morgan Kaufmann, 2012

[MON13] Montes de Oca E., De Giusti L., Rodriguez I., De Giusti A., Naiouf M. Análisis de la escalabilidad y el consumo energético en soluciones paralelas sobre cluster de multicores y GPU para un problema con alta demanda computacional. CACIC2013. ISBN: 978-987-23963-1-2. Pág. 154-163. 2013.

[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.

[NOT09] Nottingham A. y Irwin B. "GPU packet classification using opencl: a consideration of viable classification methods". Research Conf. of the South African Inst. of Comp. Sc. and Inf. Technologists. ACM, 2009.

[NVI08] NVIDIA. "Nvidia CUDA compute unified device architecture, programming guide v.2.0". 2008.

[OLI08] Olivier S., Prins S., Scalable Dynamic Load Balancing Using UPC. Proc. 37th ICPP'08. CD-ROM, IEEE CS, September 2008.

[OPE13B] OpenStack Cloud Software: Open source software for building private and public clouds. http://www.openstack.org. Febrero 2013.

[PAC11] Pacheco, Peter. "An Introduction to Parallel Programming". Morgan Kaufmann, ISBN-13: 978-0123742605, 2011.

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

[PIC11] Piccoli M.F., "Computación de Alto Desempeño utilizando GPU". XV Escuela Internacional de Informática. Editorial Edulp, 2011.

[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 2008.

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

[RUC13] Rucci E., Chichizola F., Naiouf M., De Giusti A. "A Hybrid Parallel Neighbor-Joining Algorithm for Phylogenetic Tree Reconstruction on a Multicore Cluster". To appear in Journal Parallel & Cloud Computing. American V-King Scientific Publishing, LTD, Nueva York (EEUU). ISSN print: 2304-9464. ISSN online: 2304-9456.

[SAN12] Sanz V., De Giusti A., Naiouf M. Performance Analysis of a Matrix Diagonalization Algorithm with Jacobi Method on a Multicore Architecture. Procs PDPTA 2012 Las Vegas, USA. ISBN: 1-60132-227-5. Vol I. pp. 883-889

[SID07] Siddh S., Pallipadi V., Mallick A.. "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. 37th ICPP'08 IEEE CS, Sept. 2008.