Cómputo paralelo y distribuido para HPC. Fundamentos, construcción y evaluación de aplicaciones.

Marcelo Naiouf[(1)], Armando De Giusti[(1)(2)], Laura De Giusti[(1)], Franco Chichizola[(1)], Victoria Sanz[(1)(2)], Enzo Rucci[(1)(2)], Fabiana Leibovich[(1)], Silvana Gallo[(1)(2)], Erica Montes de Oca[(1)], Emmanuel Frati[(1)], Mariano Sánchez[(1)], María José Basgall[(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, erucci, fleibovich, sgallo, emontesdeoca, fefrati, msanchez, mjbasgall}@lidi.info.unlp.edu.ar, agaudi@ungs.edu.ar

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, hay 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 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 multicore, GPU, cloud), 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.

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, etc), con el fin de obtener soluciones de alto rendimiento.

En la dirección de tesis de postgrado existe colaboración, entre otros, 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 la Universidad Complutense de Madrid.

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

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][PAC11][COO12][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 utilización de arquitecturas *many-core* (como las placas gráficas de uso general) como máquinas paralelas de memoria compartida, constituyendo una plataforma 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 [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, 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 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, 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 [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]. Por otra parte, ha comenzado a surgir en el área el uso de FPGA (*Field Programmable Gate Array*), debido al potencial incremento de productividad y mejora de rendimiento energético, aunque con alto costo de programación; en este sentido, son valoradas las herramientas de alto nivel que reduzcan dicho costo [SET13].

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. Una tendencia promisoria es la que brinda Intel a partir de los procesadores MIC (Many Integrated Core Architecture), permitiendo utilizar métodos y herramientas estándar de HPC [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, 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.

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 eficientes es un tema de interés.

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 [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 [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. El desarrollo de nuevos modelos requiere caracterizar el contexto de comunicaciones entre los procesadores y la asociación entre los algoritmos, el paradigma elegido y la arquitectura. 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
- Desarrollo de soluciones paralelas empleando FPGAs. Análisis y evaluación de costo de programación, rendimiento y consumo energético de sistemas con FPGAs.
- 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, GPU y cluster de 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 del árbol de búsqueda 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 algoritmo A* secuencial para resolverlo, y dos versiones propias optimizadas del algoritmo paralelo Hash Distributed A* (HDA*[KIS12] [BUR10]) que realiza el balance de carga de los nodos generados del grafo mediante una función hash: (1) una versión utiliza MPI, haciendo posible su ejecución tanto sobre arquitecturas con memoria distribuida y memoria compartida, y (2) otra versión utiliza 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 [SAN14a] [SAN14b]. Se analizó el rendimiento de ambas versiones cuando corren sobre una máquina multicore, al aumentar la cantidad de cores y la carga de trabajo, comprobando su buena escalabilidad sobre dicha arquitectura. También se comprobó que HDA* Pthreads obtiene mejor rendimiento y reduce el consumo de memoria respecto a HDA* MPI cuando corren sobre una máquina multicore, estos resultados indican que será beneficioso aplicar programación híbrida para este algoritmo de búsqueda cuando la arquitectura subyacente será un cluster de multicore y en consecuencia se sentaron las bases para el algoritmo HDA* híbrido. Asimismo se comprobó la buena escalabilidad de HDA* MPI sobre un cluster de multicore convencional. Se destaca que las optimizaciones añadidas en las versiones desarrolladas mejoraron el rendimiento respecto a los algoritmos originales. Como líneas de trabajo futuro se plantea implementar el algoritmo HDA* híbrido y analizar su rendimiento sobre un cluster de multicore.

- Aplicaciones con paralelismo de datos. Se continuó con la línea de investigación que se estaba llevando adelante, centrándose en el aprovechamiento de la jerarquía de memoria, particularmente en el uso de la Caché L1 en cluster de multicore, analizando escalabilidad tanto al incrementar el tamaño del problema como la cantidad de núcleos involucrados en la experimentación y luego de implementar y analizar diferentes soluciones híbridas (pasaje de mensajes y memoria compartida utilizando MPI + OpenMP) [LEI13], se continuó con la aplicación de los conocimientos en el análisis de la paralelización de aplicaciones de simulación atmosféricas. Las mismas 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.

- **Alineamiento de secuencias**: Esta operación resulta esencial es muchas áreas como la biología, la química y la medicina forense. El algoritmo Smith-Waterman (SW) es un método popular capaz de calcular el puntaje de alineamiento óptimo en tiempo O(n2). La reciente aparición de arquitecturas many-core, como las placas GPUs y los coprocesadores Intel MIC, da la oportunidad de acelerar las búsquedas biológicas sobre hardware comúnmente disponible a un costo accesible. Por ese motivo, se desarrollaron diferentes soluciones paralelas para una arquitectura heterogénea basada en procesadores Intel Xeon con una placa Intel Xeon Phi [RUC14]. Se analizaron sus rendimientos y los costos de programación asociados. Interesa extender el análisis considerando otros factores, como por ejemplo, el consumo energético, y experimentar sobre FPGA Además, evaluar otros tipos de plataformas heterogéneas, como las que incluyen GPUs.

- **Simulación distribuida de modelos orientados al individuo.** Estos modelos representan la interacción entre individuos de un sistema, y en consecuencia brindan el comportamiento de la población como resultado de dichas interacciones 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. Sin embargo, esto implica grandes necesidades de cómputo y comunicación para lograr resultados cercanos a la realidad simulada. La aparición de arquitecturas distribuidas y procesadores con varios núcleos ha permitido el desarrollo de modelos más complejos y a gran escala utilizando técnicas de simulación distribuida. Lograr mejoras en la eficiencia de dichas aplicaciones y hacer un buen aprovechamiento de las diferentes arquitecturas existentes es un desafío que continúa vigente. Anteriormente, se aplicó 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 sobre la arquitectura disponible, de modo tal que la comunicación entre los mismos se vea beneficiada. Actualmente, se encuentran en fase de prueba las modificaciones necesarias para lograr la reducción del overhead del subsistema de comunicaciones del simulador. Por otro lado, se pretende iniciar el análisis del desempeño del simulador en cuanto al balance de cómputo, ya que al trabajar con grandes cantidades de individuos que se desplazan pseudo aleatoriamente, y debido a la cantidad de cómputo asociada a la selección de vecinos para realizar el movimiento, se producen desbalances en cuanto a la cantidad de trabajo de cada uno de los diferentes procesos lógicos. Para realizar esta tarea, será necesario explorar diferentes técnicas de balance de cómputo y desarrollar las mejoras pertinentes.

- **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]. Se modificó la solución del problema para ejecutarla sobre un cluster de GPU, utilizando la combinación de MPI-CUDA. El trabajo experimental ha dado como resultado una buena aceleración obtenida utilizando cluster de GPU. Además, se comparó la aceleración de la aplicación ejecutada en un cluster de CPU y en el cluster de GPU; se observó claramente que el uso de este último logró una aceleración similar al uso del cluster de CPU pero con tiempos de ejecución significativamente menores [MON14]. Actualmente, se trabaja en la medición de consumo energético sobre cluster de GPU.

- **Problemas de simulación relacionados con fenómenos naturales (inundaciones)**. 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 [GAU14a] [GAU14b]. 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.

- ***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. 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].

- **Aplicaciones en Big Data .** Actualmente, la cantidad de datos que se crea cada dos días se estima que es equivalente a lo generado desde el principio de los tiempos hasta 2003. Los datos producidos son del orden superior a los petabytes (10^{15} bytes), y crece en torno al 40\% cada año. Muchas de las técnicas para el tratamiento sobre "Grandes Datos" pertenecen a la minería de datos, y en este sentido es de interés estudiar la paralelización de las mismas a fin de obtener resultados en tiempos razonables. Se trata de una línea incipiente en el grupo.

4. FORMACION DE RECURSOS HUMANOS

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

[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

[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

[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.

[GAU14a] A. Gaudiani, E. Luque, P. García, M. Re, M. Naiouf, A. De Giusti. "Computational method for prediction enhancement of a river flood simulation". EnviroInfo 2014 – ICT for Energy Efficiency - 28th Int'l Conf. on Informatics for Environmental Protection. Jorge Marx Gómez, Michael Sonnenschein, Ute Vogel, Andreas Winter, Barbara Rapp, Nils Giesen (Eds.) Oldenburg, Germany September 10-12, 2014. Pp 325-332.

[GAU14b]. A. Gaudiani, E. Luque, P. García, M. Re, M. Naiouf, A. De Giusti. "Computing, a powerful tool for improving the parameters simulation quality in flood prediction". Procedia Computer Science (Elsevier). Vol 29, pp 299-309. June 2014. ISSN: 1877-0509.

[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

[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) – Workshop de Procesamiento Distribuido y Paralelo. Octubre 2014. Pp 1220-1227. ISBN: 978-987-3806-05-6

[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.

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

[RUC14] "Smith-Waterman Algorithm on Heterogeneous Computing: A Case of Study". Enzo Rucci, Armando De Giusti, Marcelo Naiouf, Carlos García Sanchez, Guillermo Botella Juan, Manuel Prieto-Matías. Proceedings of 2014 IEEE International Conference on Cluster Computing (CLUSTER). 22 al 26 de Septiembre de 2014. Madrid, España. ISBN: 978-1-4799-5547-3. Págs. 323-330.

[SAN14a] On the Optimization of HDA* for Multicore Machines. Performance Analysis. Victoria Sanz, Armando De Giusti, Marcelo Naiouf. PDPTA'14. The 2014 International Conference on Parallel and Distributed Processing Techniques and Applications. Las Vegas, USA. Julio 2014. ISBN 1-60132-282-8

[SAN14b] Ajuste de rendimiento del algoritmo HDA* para máquinas multicore. Victoria Sanz, Armando De Giusti, Marcelo Naiouf. CACIC 2014. XX Congreso Argentino de Ciencias de la Computación. San Justo, Bs. As., Argentina. Octubre 2014. ISBN: 978-987-3806-05-6

[SET13] High-performance Dynamic Programming on FPGAs with OpenCL. Sean Settle. 2013 IEEE High Performance Extreme Computing Conf (HPEC '13), 2013.

[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.

[TEL08] Teller J., Ozguner F., Ewing R., Scheduling Task Graphs on Heterogeneous Multiprocessors with Reconfigurable Hardware, Proc. 37th ICPP'08 IEEE CS, Sept. 2008.