View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Servicio de Difusión de la Creación Intelectual

Fundamentos de cómputo paralelo y distribuido para HPC. Construcción y evaluación de aplicaciones.

Marcelo Naiouf[(1)], Armando De Giusti[(1)(2)], Laura De Giusti[(1)], Franco Chichizola[(1)], Victoria Sanz[(1)(2)], 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)], Carolina Actis[(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, apousa, erucci, sgallo, emontesdeoca,fefrati, msanchez, mjbasgall. cactis}@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, 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).

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][PAC11][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 [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 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 [PIC11][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][XIL15].

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][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.

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.
- Modelos de representación y predicción de performance de algoritmos paralelos.
- Arquitecturas multicore y many-core. Arquitecturas FPGA.
- 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 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.
- Análisis de los problemas de migración y asignación de procesos y datos a procesadores.
- 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

Resultados y Objetivos

- 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.
- Investigar la paralelización en plataformas que combinan multicore y aceleradores, o que disponen de más de un acelerador. 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).
- 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, 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 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 en una configuración final. 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 de un *cluster,* (b) la potencia de cómputo de 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, ejecutando tanto sobre arquitecturas con memoria distribuida y memoria compartida, y (2) otra versión con Pthreads, que ejecuta sobre un multiprocesador con memoria compartida, que elimina ciertas ineficiencias respecto a (1). 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 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. Se comprobó la buena escalabilidad de HDA* MPI sobre un cluster de multicore convencional. 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 [SAN15].

- **Criptografía de grandes volúmenes de datos.** Hoy en día, el volumen de datos que se trasmiten en las redes se ha incrementado considerablemente, y en ocasiones suelen ser información sensible, por lo tanto es importante codificarlos para enviarlos por una red pública como lo es InterNet de manera segura. El encriptado y desencriptado de datos requiere un tiempo de cómputo adicional, que dependiendo de su tamaño puede ser 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 del cómputo de encriptar y desencriptar información logrando un excelente rendimiento [POU15].

- **Búsqueda de similitud en bases de datos biológicas.** El algoritmo de Smith-Waterman es el método más preciso para esta clase de búsquedas, la cual resulta ser una tarea cotidiana y recurrente en las investigaciones de la bioinformática y la biología molecular. Desafortunadamente el algoritmo Smith-Waterman resulta costoso debido a su complejidad computacional cuadrática y la situación se agrava 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 las búsquedas biológicas sobre hardware comúnmente disponible a un costo accesible. Por ese motivo, se desarrollaron diferentes soluciones paralelas para arquitecturas heterogéneas basadas en coprocesadores Intel Xeon Phi. 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 CPU y GPU [RUC15a]. Por otra parte, se desarrolló un primer estudio de viabilidad del empleo del modelo de programación OpenCL sobre FPGAs para acelerar este problema [RUC15b]. A futuro, interesa profundizar el estudio de soluciones basadas en FPGAs y comparar su rendimiento, costo de programación y eficiencia energética con el de las empleadas anteriormente.

- **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). 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 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)**. 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 [DEG15], 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 (1015 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. En particular, se está trabajando sobre grandes volúmenes de datos en flujo continuo provenientes de redes de microblogging, particularmente Twitter, estudiando la paralelización de las mismas bajo el paradigma MapReduce [BAS15]. Se trata de una línea incipiente en el grupo.

Formación de Recursos Humanos

Dentro de la temática de la línea de I/D se concluyeron 2 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.

Referencias

[BAL12] 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. Editores: Armando De Giusti, Guillermo Simari, Patricia Pesado. ISBN 978-987-1985-20-3. Págs. 53-65. Editorial de la Universidad de La Plata (edulp). La Plata (Argentina). 2013

[BAS15] Basgall M. J., Aquino G., Lanzarini L., Naiouf M. "Un Enfoque Dinámico para la Detección de Relaciones entre Tópicos en Textos provenientes de Redes Sociales". III Jornadas de Cloud Computing y Big Data (JCC&BD). Facultad de Informática. UNLP. 2015.

[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

[DEG15] De Giusti L., Leibovich F., Chichizola F., Naiouf

M., De Giusti A. “Incorporando conceptos en la enseñanza de Concurrencia y Paralelismo utilizando el entorno CMRE”. Procs XXI Congreso Argentino de Ciencias de la Computación – Workshop de Innovación en Educación. Octubre 2015. Pp 1212-1221. ISBN: 978-987-3724-37-4

[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.

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

[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.

[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.

[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”. XXI Congreso Argentino de Ciencias de la Computación (Junín, 2015). ISBN 978-987-3806-05-6.

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

[RUC15a] Enzo Rucci, Carlos García Sanchez, Guillermo Botella Juan, Armando De Giusti, Marcelo Naiouf, Manuel Prieto-Matías. “An Energy-Aware Performance Analysis of SWIMM: Smith-Waterman Implementation on Intel’s Multicore and Manycore”. Concurrency and Computation: Practice and Experience. Vol. 27 Nro. 18. Págs- 5517-5537. ISSN Online: 1532-0626, 1532-0634 (Online). DOI: 10.1002/cpe.3598.

[RUC15b] Enzo Rucci, Armando De Giusti, Marcelo Naiouf, Carlos García Sanchez, Guillermo Botella Juan, Manuel Prieto-Matías. “Smith-Waterman Protein Search with OpenCL on FPGA”. 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. Págs. 208-213. DOI: 10.1109/Trustcom.2015.634.

[RUC16a] Enzo Rucci, Armando De Giusti, Marcelo Naiouf, Carlos García Sanchez, Guillermo Botella Juan, Manuel Prieto-Matías. “State-of-the-art in Smith-Waterman Protein Database Search”. Big Data Analytics in Genomics. Ka-Chun Wong (Editor). Springer (New York), 2016. En prensa.

[RUC16b] Rucci, Enzo. “Evaluación de rendimiento y eficiencia energética en sistemas heterogéneos para bioinformática”. Tesis de Doctorado en Ciencias Informáticas (Facultad de Informática – UNLP). 2016.

[SAN15] Sanz Victoria. “Análisis de rendimiento y optimización de algoritmos paralelos Best-First Search sobre multicore y cluster de multicore”. Tesis de Doctorado de Ciencias Informáticas (Facultad de Informática – UNLP). 2015. http://hdl.handle.net/10915/44478.

[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.

[XIL15] Xilinx Inc. SDAccel Development Environment. [Online]. Disponible en http://www.xilinx.com/products/design-tools/software-zone/sdaccel.html