

COMPUTACIÓN DE ALTO DESEMPEÑO: ALGORITMOS Y EVALUACION DE RENDIMIENTO EN DIFERENTES PLATAFORMAS

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

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

²CONICET – Consejo Nacional de Investigaciones Científicas y Técnicas
³CIC – Comisión de Investigación Científica de la Provincia de Buenos Aires

⁴Universidad Nacional de General Sarmiento
{mnaouf, degiusti, ldgiusti, francoch, vsanz, apousa, erucci, sgallo, emontesdeoca, fefrati,
msanchez, mjbassgall}@lidi.info.unlp.edu.ar, agaudi@ungs.edu.ar

RESUMEN

El eje central de la línea de I/D son los temas de procesamiento paralelo y distribuido para HPC (fundamentos y aplicaciones). Interesa la construcción, evaluación y optimización de soluciones sobre diferentes plataformas de software y arquitecturas con múltiples procesadores (multicore, clusters, cloud, aceleradores y placas de bajo costo), 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 computacional y 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 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; con el Departamento de Arquitectura de Computadores y Automática de la Universidad Complutense de Madrid; y con el grupo Soft Computing and Intelligent Information Systems (SCI2S) de la Universidad de Granada, entre otros.

Palabras clave: Cómputo paralelo y distribuido de altas prestaciones. Algoritmos paralelos y distribuidos. Clusters. Multicore. Aceleradores. Consumo energético. Balance de carga. Aplicaciones. Performance.

CONTEXTO

La línea de I/D que se presenta es parte del Proyecto “Computación de Alto Desempeño: Arquitecturas, Algoritmos, Métricas de rendimiento y Aplicaciones en HPC, Big Data, Robótica, Señales y Tiempo Real.” del III-LIDI acreditado por el Ministerio de Educación, y de

proyectos acreditados y subsidiados por la Facultad de Informática de la UNLP. Además, existe cooperación con Universidades de Argentina, Latinoamérica y Europa a través de proyectos acreditados por AECID, CyTeD, OEI y CIC y becas de Telefónica de Argentina. Asimismo, el III-LIDI forma parte del Sistema Nacional de Cómputo de Alto Desempeño (SNCAD).

1. INTRODUCCIÓN

El área de cómputo de altas prestaciones (HPC, High-Performance Computing) es clave dentro de las Ciencias de la Computación, debido al creciente interés por el desarrollo de soluciones a problemas con alta demanda computacional y de almacenamiento, produciendo transformaciones profundas en las líneas de I/D [1].

El rendimiento en este caso está relacionado con dos aspectos: las arquitecturas de soporte y los algoritmos que hacen uso de las mismas, y el desafío se centra en cómo aprovechar las prestaciones obtenidas a partir de la evolución de las arquitecturas físicas. En esta línea 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 como pueden ser las unidades de procesamiento gráfico (GPU, Graphic Processing Unit) de NVIDIA y AMD, los coprocesadores Xeon Phi de Intel [2] o los aceleradores basados en circuitos integrados reconfigurables (FPGAs, Field Programmable Gate Array) [3]. En la actualidad se comercializan placas de bajo costo como Raspberry PI [4] u Odroid [5] que poseen múltiples núcleos de baja complejidad y en

algunos casos son procesadores multicore asimétricos (AMPs) con el mismo repertorio de instrucciones. Es de interés estudiar como explotar el paralelismo en estos dispositivos para mejorar el rendimiento y/o consumo energético de las aplicaciones [6], así como las características de scheduling en los mismos [7]. Asimismo, los entornos de computación cloud introducen un nuevo foco desde el punto de vista del HPC, brindando un soporte “a medida” sin la necesidad de adquirir el hardware.

La creación de algoritmos paralelos en arquitecturas multiprocesador no es un proceso directo [8]. El costo puede ser alto en términos del esfuerzo de programación y el manejo de la concurrencia adquiere un rol central en el desarrollo. Si bien en las primeras etapas el diseñador de una aplicación paralela 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. 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 [9].

Muchos problemas algorítmicos se vieron impactados por los multicore y clusters de multicore. A partir de incorporar varios chips multicore dentro de un nodo y conectar múltiples nodos vía red, se puede crear una arquitectura NUMA, de modo que los cores en un chip compartan memoria principal, y puedan acceder remotamente a la memoria dedicada de otro chip, aunque ese acceso sea más costoso, surgiendo así varios niveles de comunicación. Esto impacta sobre el desarrollo de algoritmos que aprovechen adecuadamente las arquitecturas, y motiva el estudio de performance en sistemas híbridos. Además, es necesario estudiar la utilización de lenguajes y bibliotecas ya que aún no se cuenta con un standard, aunque puede mencionarse el uso de los tradicionales MPI, OpenMP y Pthreads [10][11] o los más recientemente explorados UPC, Chapel y Titanium del modelo PGAS [12].

La combinación de arquitecturas con múltiples núcleos con aceleradores dio lugar a plataformas híbridas con diferentes características. Más allá del acelerador utilizado, la programación de estas plataformas representa un desafío. Para lograr aplicaciones de alto rendimiento, los programadores enfrentan dificultades como: estudiar características específicas de cada arquitectura y aplicar técnicas de programación y optimización particulares de cada una, lograr un balance de carga adecuado entre los dispositivos de procesamiento y afrontar la ausencia de estándares para este tipo de sistemas.

Por otra parte, los avances en las tecnologías de virtualización han llevado a que Cloud Computing sea una alternativa a los tradicionales sistemas de cluster [13]. El uso de cloud para HPC presenta desafíos atractivos, brindando un entorno reconfigurable dinámicamente sin

la necesidad de adquirir hardware, y es una excelente plataforma para testear escalabilidad de algoritmos aunque queda mucho por hacer en cuanto al diseño, lenguajes y programación

Métricas de evaluación del rendimiento y balance de carga

La diversidad de opciones vuelve complejo el análisis de performance de los Sistemas Paralelos, ya que los ejes sobre los cuales pueden compararse dos sistemas son varios. Existe un gran número de métricas para evaluar el rendimiento, siendo las tradicionales: tiempo de ejecución, speedup, eficiencia. Por su parte, la *escalabilidad* permite capturar características de un algoritmo paralelo y la arquitectura en que se lo implementa. Posibilita testear la performance de un programa sobre pocos procesadores y predecirla en un número mayor, así como caracterizar la cantidad de paralelismo inherente en un algoritmo.

Un aspecto de interés que se ha sumado como métrica, a partir de las plataformas con gran cantidad de procesadores, es el del *consumo* y la *eficiencia energética* [14]. Muchos esfuerzos están orientados a tratar el consumo como eje de I/D, como métrica de evaluación, y también a la necesidad de metodologías para medirlo.

El objetivo principal del cómputo paralelo es reducir el tiempo de ejecución haciendo uso eficiente de los recursos. El *balance de carga* es un aspecto central y consiste en, dado un conjunto de tareas que comprenden un algoritmo y un conjunto de procesadores, encontrar el mapeo (asignación) de tareas a procesadores tal que cada una tenga una cantidad de trabajo que demande aproximadamente el mismo tiempo, y esto es más complejo si hay heterogeneidad. Dado que el problema general de mapping es *NP-completo*, pueden usarse enfoques que dan soluciones subóptimas aceptables. Las técnicas de planificación a nivel micro (dentro de cada procesador) y macro (en un cluster) deben ser capaces de obtener buen balance de carga. Existen técnicas estáticas y dinámicas cuyo uso depende del conocimiento que se tenga sobre las tareas de la aplicación.

2. LÍNEAS DE INVESTIGACIÓN, DESARROLLO E INNOVACIÓN

- Investigar en temas de cómputo paralelo y distribuido de alto desempeño, en lo referido a los fundamentos y a la construcción y evaluación de las aplicaciones. Esto incluye los problemas de software asociados con el uso de arquitecturas multiprocesador:
 - Lenguajes, modelos y paradigmas de programación paralela (puros e híbridos a distintos niveles).
 - Asignación de procesos a procesadores optimizando el balance de la carga de procesamiento.

- Métricas de evaluación de complejidad y rendimiento: speedup, eficiencia, escalabilidad, consumo energético, costo de programación.
- Construir, evaluar y optimizar soluciones utilizando algoritmos concurrentes, paralelos y distribuidos sobre diferentes plataformas de software y arquitecturas con múltiples procesadores:
 - Arquitecturas de trabajo homogéneas, heterogéneas e híbridas: multicores, clusters, GPU, Xeon Phi, FPGA, placas de bajo costo y cloud.
 - Se propone aplicar los conceptos en problemas numéricos y no numéricos de cómputo intensivo y/o sobre grandes volúmenes de datos (aplicaciones científicas, búsquedas, simulaciones, imágenes, realidad virtual y aumentada, bioinformática, big data, n-body).
- Analizar y desarrollar ambientes para la enseñanza de programación concurrente y paralela.
 - Caracterizar diferentes modelos de arquitecturas paralelas.
 - Representar distintos modelos de comunicación/sincronización.
 - Definir métricas de evaluación de rendimiento y eficiencia energética.

3. RESULTADOS OBTENIDOS/ESPERADOS

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

En este marco, pueden mencionarse los siguientes resultados:

- Para la experimentación se han utilizado y analizado diferentes arquitecturas homogéneas o heterogéneas,

incluyendo multicores, cluster de multicores (con 128 núcleos), GPU y cluster de GPU, Xeon Phi y FPGA.

- Se experimentó la paralelización en arquitecturas híbridas, con el objetivo de estudiar el impacto del mapeo de datos y procesos, así como de los lenguajes y librerías.
- Respecto de las aplicaciones y temas estudiados, se trabajó fundamentalmente con los siguientes problemas:

➤ **Aceleración de aplicaciones con cómputo colaborativo CPU-GPU.** Las computadoras comerciales actuales incluyen decenas de cores y al menos una GPU. El uso de ambas unidades de procesamiento de forma colaborativa puede mejorar significativamente el rendimiento de una aplicación. Sin embargo, esto supone un desafío para los programadores ya que dichas unidades difieren en arquitectura, modelo de programación y rendimiento. En [15] se propuso un modelo híbrido para estructurar código a ser ejecutado sobre un sistema heterogéneo con múltiples cores y 1 GPU (utilizando todos los recursos disponibles). Utilizando este modelo se desarrolló un algoritmo paralelo de pattern matching para sistemas heterogéneos CPU-GPU. Los resultados revelaron que este algoritmo supera en rendimiento a trabajos previos, desarrollados para sistemas multicore y GPUs, para datos de tamaño considerable.

➤ **Alineamiento de secuencias biológicas.** Esta operación consiste en comparar dos o más secuencias biológicas, como pueden ser las de ADN o las de proteínas, y resulta fundamental en investigaciones de la bioinformática y la biología molecular. El algoritmo de Smith-Waterman es considerado el método de alineamiento más preciso. Desafortunadamente, este algoritmo resulta costoso debido a su complejidad computacional cuadrática mientras que la situación se agrava aún más a causa del crecimiento exponencial de datos biológicos en los últimos años [16]. El reciente surgimiento de aceleradores en HPC (GPU, Xeon Phi, FPGA, entre otros) da la oportunidad de acelerar los alineamientos sobre hardware comúnmente disponible a un costo accesible. En primer lugar, se exploró el empleo del modelo de programación OpenCL sobre FPGAs para acelerar el alineamiento de secuencias largas de ADN [17]. Luego, se desarrolló una herramienta capaz de procesar alineamientos de secuencias de ADN sin restricciones de tamaño y se analizó su rendimiento comparándolo con el de otras implementaciones basadas en multicores, Xeon Phi y GPUs [18]. Por otra parte, la salida al mercado de las nuevas generaciones de procesadores Xeon (Skylake) y Xeon Phi (Knights Landing) brindó la oportunidad de evaluar su uso para búsquedas de similitud en bases de datos de proteínas [19][20]. A futuro, interesa explorar el uso de próximas generaciones de procesadores y aceleradores para esta aplicación.

➤ **Cálculo de los caminos mínimos.** es uno de los problemas básicos y de mayor antigüedad de la teoría de

grafos teniendo aplicación en el dominio de las comunicaciones, del ruteo de tráfico, de la bioinformática, entre otros. El algoritmo de Floyd-Warshall (FW) permite computar la distancia mínima entre todos los pares de un grafo. Además de poseer una alta demanda de ancho de banda, FW resulta costoso computacionalmente al ser $O(n^3)$. Empezando por una implementación secuencial de base, se estudió y cuantificó cómo diferentes optimizaciones a nivel de datos, hilos y compilador permiten mejorar su rendimiento sobre los nuevos procesadores Xeon Phi Knights Landing [21]. Como trabajo futuro, interesa realizar un estudio comparativo con otros aceleradores (como GPUs), no sólo desde el punto de vista del rendimiento sino también de la eficiencia energética.

➤ **Simulación de enfermedades infecciosas transmitida por vectores utilizando modelos basados en agentes.** Se comenzó la adaptación de un modelo [22] de simulación sobre las enfermedades infecciosas dengue, zika y chikungunya, que permita abordar conocimientos que ayuden en la toma de decisiones y acciones de prevención. Dada la potencia de cómputo necesaria para su simulación se propone el uso de las plataformas cluster de multicore y GPU como infraestructura para disponer de simulaciones escalables y tiempo de simulación aceptables. Posteriormente, se llevará a cabo la validación del modelo desarrollado. Se propone realizar un estudio y análisis de performance en las arquitecturas propuestas. Además, se evaluar el consumo energético para determinar las relaciones entre escalabilidad, consumo y eficiencia.

➤ **Problemas de tipo N-body.** Se utilizaron las plataformas de memoria compartida GPU y cluster de multicore para la resolución de problemas con alta demanda computacional del tipo N-body. Se emplearon diferentes modelos de comunicación: memoria compartida (Pthreads en CPU y CUDA en GPU), pasaje de mensajes (MPI) y soluciones híbridas (MPI-Pthreads). Se han mostrado los beneficios del uso de la GPU en problemas con características similares al caso planteado. El trabajo experimental ha dado como resultado una buena aceleración obtenida utilizando cluster de GPU. Además, se observó claramente que el uso del cluster de GPU logró una aceleración proporcional al speedup conseguido con el cluster de CPU pero con tiempos de ejecución significativamente menores [23]. También se han desarrollado diferentes alternativas de distribución de trabajado usando un Cluster de GPUs heterogéneas [24]. Además del uso de un Cluster de GPU se está utilizando MultiGPU, pero haciendo énfasis en el consumo energético. Se desarrolló un modelo de estimación de tiempo junto con un modelo de estimación de energía para el uso de una GPU, un Cluster de GPU y una MultiGPU. El trabajo experimental realizado se centró tanto en la performance como en el consumo energético de las

distintas arquitecturas GPU empleadas, obteniéndose de este modo resultados que permitieron determinar las ventajas de una arquitectura sobre otra [25].

➤ **Problemas de optimización de simulación de sistemas dinámicos complejos mediante heurísticas.** La búsqueda de un conjunto de parámetros de entrada que optimicen el funcionamiento de un simulador de un sistema físico es un proceso de alto costo computacional que puede considerarse intratable y requiere de heurísticas que permitan disminuir el tiempo de ejecución. En los nuevos avances realizados en esta línea se aprovechó la continuidad en los valores de los parámetros físicos distribuidos sobre el dominio del sistema, de manera de realizar búsquedas de parámetros ajustados sobre espacios de búsquedas de tamaño mucho más reducidos que los utilizados en una primera etapa del trabajo. Esta metodología mucho más eficiente se puede extender a otros simuladores de fenómenos físicos y en particular con simuladores de inundaciones de ríos con las que se llevaron adelante las experiencias. Las experiencias se pueden correr de manera colaborativa beneficiándose ampliamente del uso de plataformas de clusters de procesadores [26] [27].

➤ **Ambientes para la enseñanza de concurrencia.** Se desarrolló el entorno CMRE 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 [28]. 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 [29]. Se ha ampliado para incorporar conceptos básicos de computación en la nube (Cloud Computing) [29].

➤ **HPC y Big Data.** Actualmente los escenarios Big Data son cada vez más comunes debido a la gran cantidad de datos que se genera constantemente. Para poder procesar dicha cantidad de datos y, en tiempos razonables de ejecución, es necesario utilizar cómputos de altas prestaciones. Además, actualmente existen diferentes herramientas pensadas para trabajar con este tipo de problemas. Una de ellas, y de las más ampliamente utilizadas, es conocida como Apache Spark. Este framework está basado en el paradigma MapReduce y, entre sus más importantes características se encuentran los mecanismos eficientes de tolerancia a fallos y el uso intensivo de memoria RAM que emplea. En esta línea de trabajo, se están adaptando técnicas

tradicionales aplicadas en pequeños datasets [30] como también desarrollando nuevas técnicas para el preprocesamiento de datos para problemas de clasificación no balanceada en Big Data, utilizando el framework Apache Spark.

4. FORMACIÓN DE RECURSOS HUMANOS

Dentro de la temática de la línea de I/D en el año 2018 se concluyó 1 Trabajo Final de Especialización y 1 Tesina de Grado de Licenciatura. Se encuentran en curso en el marco del proyecto 3 tesis doctorales, 2 de maestría, 3 trabajos de Especialización y 3 Tesinas de grado.

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

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

5. BIBLIOGRAFÍA

[1]. Giles MB, Reguly I. "Trends in high-performance computing for engineering calculations". *Phil.Trans.R.Soc.A* 372: 20130319. 2014. <http://dx.doi.org/10.1098/rsta.2013.0319>

[2]. Jeffers, James; Reinders, James. "Intel Xeon Phi Coprocessor High Performance Programming". Morgan Kaufmann. 2013.

[3]. Sean Settle. "High-performance Dynamic Programming on FPGAs with OpenCL". *IEEE High Performance Extreme Computing Conf (HPEC '13)*. 2013. <https://doi.org/10.1016/j.physa.2017.04.159>.

[4]. Raspberry PI. <https://www.raspberrypi.org/> Accedido 21 de Marzo de 2016.

[5]. Odroid <http://www.hardkernel.com> Accedido 21 de Marzo de 2016.

[6]. Annamalai A., Rodrigues R., Koren I., Kundu S., "Dynamic Thread Scheduling in Asymmetric Multicores to Maximize Performance-per-Watt," 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, pp. 964-971, 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, 2012.

[7]. Juan Carlos Saez, Adrian Pousa, Daniel Chaver, Fernando Castro, Manuel Prieto Matias. "ACFS: A Completely Fair Scheduler for Asymmetric Single-ISA Multicore Systems". *ACM SAC 2015 (The 30TH ACM/SIGAPP Symposium on applied computing)*. 2015.

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

[9]. De Giusti L, Naiouf M., Chichizola F., Luque E., De Giusti A. "Dynamic Scheduling in Heterogeneous Multiprocessor Architectures. Efficiency Analysis". *Computer Science and Technology Series – XV Argentine Congress of Computer Science Selected*

Papers. La Plata (Buenos Aires): Editorial de la Universidad de La Plata (edulp). 2010. p85 - 95. isbn 978-950-34-0684-7.

[10]. Chapman, B., Jost, G. & Van der Pas. "Using OpenMP – Portable Shared Memory Parallel Programming". (2008). UK: MIT Press.

[11]. Hager, G. & Wellein, G. "Introduction to HPC for Scientists and Engineers". (2011) EEUU: CRC Press.

[12]. De Wael, M; Marr, S; De Fraine, B; Van Cutsem, T; De Meuter, W. "Partitioned Global Address Space Languages". *ACM Computing Surveys (CSUR)* 47 (4), 2015.

[13]. Amazon Elastic Compute Cloud (Amazon EC2). <http://aws.amazon.com/es/ec2/>. Febrero 2013.

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

[15]. V. Sanz, A. Pousa, M. Naiouf, and A. De Giusti, "Accelerating Pattern Matching with CPU-GPU Collaborative Computing", *Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2018)*, ISBN: 978-3-030-05051-1, págs. 310-322, doi. https://doi.org/10.1007/978-3-030-05051-1_22, 2018.

[16]. 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). ISBN: 978-3-319-41278-8 (print) 978-3-319-41279-5 (online), Springer, págs. 197-223, 2016.

[17]. E. Rucci, C. Garcia, G. Botella, A. De Giusti, M. Naiouf, and M. Prieto-Matias. "Accelerating Smith-Waterman Alignment of Long DNA Sequences with OpenCL on FPGA". En: *Bioinformatics and Biomedical Engineering*. IWBBIO 2017. *Lecture Notes in Computer Science*, vol 10209., ISBN: 978-3-319-56154-7, Springer, Cham, págs. 500-511, 2017.

[18]. E. Rucci, C. Garcia, G. Botella, A. De Giusti, M. Naiouf, and M. Prieto-Matias. "SWIFOLD:Smith-Waterman Implementation on FPGA with OpenCL for Long DNA Sequences". *BMC Systems Biology*. ISSN: 1752-0509. In press. 2018.

[19]. E. Rucci, C. Garcia, G. Botella, A. De Giusti, M. Naiouf, and M. Prieto-Matias. "First Experiences Accelerating Smith-Waterman on Intel's Knights Landing Processor". *Algorithms and Architectures for Parallel Processing*. ICA3pp 2017. *Lecture Notes in Computer Science*, vol 10393, ISBN: 978-3-319-65482-9, Springer International Publishing, págs. 569-579, 2017.

[20]. Enzo Rucci, Carlos García, Guillermo Botella, Armando De Giusti, Marcelo Naiouf and Manuel Prieto-Matías. "SWIMM 2.0: enhanced Smith-Waterman on Intel's Multicore and Manycore architectures based on AVX-512 vector extensions". *International Journal of*

Parallel Programming, Springer US, 2018. DOI: <https://doi.org/10.1007/s10766-018-0585-7>

[21]. E. Rucci, A. De Giusti, and M. Naiouf, "Blocked All-Pairs Shortest Paths Algorithm on Intel Xeon Phi KNL Processor: A Case Study". En: Computer Science – CACIC 2017. Communications in Computer and Information Science, vol 790, ISBN: 978-3-319-75213-6 978-3-319-75214-3, Springer, Cham, págs. 47-57, 2018.

[22]. Borges F., Gutierrez-Milla A., Suppi R., Luque E., de Brito Arduino M., "An agent-based model for assessment of aedes Aegypti pupal productivity". Proceedings of the 2015 Winter Simulation Conference. ISBN: 978-1-4673-9743-8/15. IEEE. (2015)

[23]. E. Montes de Oca, L. De Giusti, F. Chichizola, A. De Giusti, M. Naiouf. "Utilización de Cluster de GPU en HPC. Un caso de estudio". Proceedings del XX Congreso Argentino de Ciencias de la Computación (CACIC 2014), pp. 1220-1227, 2014.

[24]. E. Montes de Oca, L. C. De Giusti, F. Chichizola, A. E. De Giusti, M. Naiouf. "Análisis de uso de un algoritmo de balanceo de carga estático en un Cluster Multi-GPU Heterogéneo". Proceedings del XXII Congreso Argentino de Ciencias de la Computación (CACIC 2016), pp. 159-168, 2016.

[25]. Montes de Oca E. S. "Análisis de consumo energético en Cluster de GPU y MultiGPU en un problema de alta demanda computacional". Trabajo Final Carrera de Especialista en Cómputo de Altas Prestaciones y Tecnología GRID. Facultad de Informática, Universidad Nacional de La Plata. (2018).

[26]. Trigila M., Gaudiani A., Luque E. (2018) "Agile Tuning Method in Successive Steps for a River Flow Simulator". In: Shi Y. et al. (eds) Computational Science – ICCS 2018. ICCS 2018. Lecture Notes in Computer Science, vol 10862. Springer, Cham

[27]. Trigila, Mariano, Gaudiani, Adriana, Luque, Emilio. "Adjustment of a simulator of a complex dynamic system with emphasis on the reduction of computational resources". Actas del Workshop. p. 142-147. 2018.

[28]. J. Castro, L. D. Giusti, G. Gorga, M. Sánchez, and M. Naiouf,. "ECMRE: Extended Concurrent Multi Robot Environment". Computer Science – CACIC 2017. Communications in Computer and Information Science, vol 790, ISBN: 978-3-319-75213-6 978-3-319-75214-3, Springer, Cham, págs. 285-294. 2018.

[29]. L. C. De Giusti, F. Chichizola, S. Rodriguez Eguren, M. Sanchez, J. M. Paniego, A. E. De Giusti. "Introduciendo conceptos de Cloud Computing utilizando el entorno CMRE". Proceedings del XXII Congreso Argentino de Ciencias de la Computación (CACIC 2016) – Workshop de Innovación en Educación en Informática. Octubre 2016. Pp 1357-1365.

[30]. Basgall, M., Hasperué, W., Naiouf, M., Fernández, A., & Herrera, F. (2018). "SMOTE-BD: An Exact and Scalable Oversampling Method for Imbalanced Classification in Big Data". Journal of Computer Science and Technology, 18(03), e23. <https://doi.org/10.24215/16666038.18.e23>.