Tendencias en Arquitecturas y Algoritmos Paralelos.

Marcelo Naiouf, Franco Chichizola, Laura De Giusti, Diego Montezanti, Enzo Rucci, Emmanuel Frati, Adrián Pousa, Fabiana Leibovich, Diego Encinas, Horacio Villagarcía, Fernando Romero, Erica Montes de Oca, Javier Balladini, Armando De Giusti

Instituto de Investigación en Informática LIDI (III-LIDI) - Facultad de Informática - UNLP
{mnaiouf,francoch,ldgiusti,dmontezanti,erucci,fefrati,apousa,fleibovich,dencinas,hvw,fromero,emontesdeoca, degiusti}@lidi.info.unlp.edu.ar; javier.balladini@gmail.com

Con la colaboración en la dirección de Tesis de Posgrado de la Universidad Autónoma de Barcelona (España) y la Universidad Complutense de Madrid (España)

CONTEXTO

Se presenta una línea de Investigación que es parte de los Proyectos 11/F010 "Arquitecturas multiprocesador distribuidas. Modelos, Software de Base y Aplicaciones" y 11/F011 "Procesamiento paralelo y distribuido. Fundamentos y aplicaciones en Sistemas Inteligentes y Tratamiento de imágenes y video" del III-LIDI acreditados por el Ministerio de Educación.
Asimismo los proyectos "Eficiencia energética en Sistemas Paralelos" y "Algoritmos Paralelos utilizando GPGPUs. Análisis de rendimiento" financiados por la Facultad de Informática de la UNLP.
En el tema hay cooperación con varias Universidades de Argentina y se está trabajando con Universidades de América Latina y Europa en proyectos financiados por CyTED, AECID y la OEI (Organización de Estados Iberoamericanos).
Se participa en iniciativas como el Programa IberoTIC de intercambio de Profesores y Alumnos de Doctorado en el área de Informática.
Por otra parte, se tiene financiamiento de Telefónica de Argentina en Becas de grado y posgrado. Asimismo el III-LIDI forma parte del Sistema Nacional de Cómputo de Alto Desempeño (SNCAD) del MINCYT.

RESUMEN

El eje de esta línea de I/D lo constituye el estudio de tendencias actuales en las áreas de arquitecturas y algoritmos paralelos.

Incluye como temas centrales:

- Arquitecturas híbridas (diferentes combinaciones de multicores y GPUs) y Arquitecturas heterogéneas.
- Lenguajes y Estructuras de Datos para nuevas arquitecturas de cómputo paralelo.
- Desarrollo y evaluación de algoritmos paralelos sobre nuevas arquitecturas y su evaluación de rendimiento.
- Aspectos del consumo energético, en particular en relación con clases de instrucciones y algoritmos paralelos.
- Empleo de contadores de hardware, en particular en toma de decisiones en tiempo de ejecución.

Las temáticas mencionadas se observan como aristas promisorias en el futuro del cómputo paralelo de altas prestaciones.

Palabras clave: Sistemas Paralelos. Multicores y GPUs. Clusters híbridos. Algoritmos y lenguajes en clusters heterogéneos/hìbridos. Perfomance y eficiencia energética. Contadores de hardware.

INTRODUCCION

Una de las áreas de creciente interés lo constituye el cómputo de altas prestaciones, en el cual el rendimiento está relacionado con dos aspectos: por un lado las arquitecturas de soporte y por el otro los algoritmos que hacen uso de las mismas.

A la aparición de arquitecturas con múltiples núcleos (como los procesadores multicore o los many-core como las GPU), se han sumado en el último tiempo combinaciones de estos, dando lugar a plataformas híbridas con diferentes características. [CHA2011][LIN2011].

Lógicamente, esto trae aparejado una revisión de los conceptos del diseño de algoritmos paralelos (incluyendo los lenguajes mismos de programación y el software de base), así como la evaluación de las soluciones que éstos implementan.
Las estrategias de distribución de datos y procesos necesitan ser investigada a fin de optimizar la performance.

Además de las evaluaciones clásicas de rendimiento prestacional como el Speed-up y la Eficiencia, otros aspectos comienzan a ser de interés, tales como el estudio del consumo y la Eficiencia energética de tales sistemas paralelos. [CAS2012].

Una herramienta que ha comenzado a ser utilizada para la monitorización y evaluación de soluciones paralelas en tiempo de ejecución son los contadores de hardware, que (en sus diferentes variantes) permiten detectar fallas de concurrencia o estimar problemas de consumo instantáneo o decidir una migración de datos o procesos. [BOR2005] [SPR2002].

En esta línea de I/D se trabaja sobre estos aspectos que marcan tendencias en el área.

Algunos conceptos

Un procesador multicore integra dos o más núcleos computacionales dentro de un mismo "chip", con el objetivo de incrementar el rendimiento reduciendo el consumo de energía en cada núcleo.

Una GPU (Graphics Processing Unit) es una arquitectura dedicada a procesamiento gráfico, con un gran número de núcleos simples (suele mencionársela como many-core). En la actualidad se utiliza su potencia de cómputo en aplicaciones de propósito general logrando un alto rendimiento y dando lugar al concepto de GPGPU (General-Purpose Computing on Graphics Processing Units). [KIR2010][PIC2011].

Un cluster es un sistema de cómputo de procesamiento paralelo compuesto por un conjunto de máquinas (o nodos) interconectadas por una red, que al cooperar configuran un recurso que se ve como "único e integrado", más allá de la distribución física de sus componentes. En particular, cada nodo puede estar compuesto por diferente hardware y sistema operativo (inclusive puede ser un multiprocesador). [CHA2007][SUR2007]

Multicores

Respecto de las arquitecturas multicore, se puede realizar la siguiente clasificación según las características de sus núcleos:

- Arquitecturas Multicores Homogéneas: Todos sus cores poseen las mismas características.
- Arquitecturas Multicores Heterogéneas: Poseen cores con diferentes características en particular con distinto ISA (instruction set architecture). La desventaja principal es que el código fuente de un programa debe ser compilado para cada tipo de core.
- Arquitecturas Multicores Asimétricas: Todos sus cores usan el mismo ISA (instruction set architecture), de manera que no es necesario compilar el código fuente para cada tipo core, pero los cores difieren en el rendimiento y tienen diferentes características de energía.

Un multicore asimétrico (AMP) típico consiste de algunos cores potentes y complejos, llamados *Fast* o *Big* cores, con características como alta frecuencia de clock, pipeline fuera de orden, predicción de saltos y alto consumo de energía; y un gran número de cores simples, llamados *Slow* o *Small* cores, con características como baja frecuencia de clock, un pipeline simple y bajo consumo de energía. [SAE2010] [SAE2011A]

Los Fast cores son apropiados para aplicaciones secuenciales o partes secuenciales de aplicaciones paralelas de uso intensivo de CPU que realmente requieren de las características de estos cores; por otro lado, los Slow cores son adecuados en aplicaciones paralelas altamente escalables o para aplicaciones que no necesiten de las características de un Fast core como por ejemplo las intensivas en memoria o en entrada-salida.

Tener cores de diferentes tipos permite optimizar el rendimiento, además de realizar especialización distribuyendo las tareas eficientemente y logrando mayor eficiencia en la relación rendimiento/energía. Las aplicaciones paralelas tienden a ser mucho más eficientes si se ejecutan sobre muchos cores simples que sobre pocos cores más potentes y complejos que cubren un área similar y consumen la misma energía; a su vez, una aplicación paralela puede tener fases secuenciales que necesitan de la potencia de los Fast cores, en este caso la utilización de un Fast core permitiría acelerar la fase secuencial. [KOU2010].

La tarea de distribuir eficientemente los hilos de las aplicaciones es parte del planificador de tareas (scheduler) del sistema operativo, los sistemas operativos actuales no poseen la capacidad de hacer planificación de tareas en arquitecturas asimétricas.
Trabajos previos agregaron nuevos planificadores de tareas en el kernel del sistema operativo Solaris, estos planificadores incluyen características que permiten determinar cuándo un hilo se debe ejecutar en un Fast core y cuando en un Slow core.[SAE2009] [SAE2011B] [FED2009A] [FED2009B]

Para determinar si un hilo debe ser migrado de un tipo de core a otro se utiliza información de los procesos dentro del sistema operativo y los contadores hardware que posee cada arquitectura. Actualmente se está trabajando en migrar los planificadores en el sistema operativo Solaris al sistema operativo Linux.

GPU

En 2007 Nvidia lanzó la implementación de CUDA, plataforma hardware y software que extiende al lenguaje C con un reducido conjunto de abstracciones para la programación paralela de propósito general en GPU. La arquitectura de una GPU-Nvidia CUDA está organizada en Streams Multiprocessors (SMs), los cuales tienen un determinado número de Streams Processors (SPs) que comparten la lógica de control y la caché de instrucciones.

La organización de los threads puede verse por nivel o jerárquicamente de la siguiente manera: grilla o grid (formado por bloques de threads), bloque (conformado por threads) y threads. En un bloque, los hilos están organizados en warps (en general de 32 hilos). Todos los hilos de un warp son planificados juntos durante la ejecución.

El sistema de memoria de la GPU está compuesto por distintos tipos de memorias que se diferencian tanto en la forma de acceso de los hilos, las limitaciones en las operaciones permitidas y su ubicación dentro del dispositivo. Está compuesto por: Memoria Global, Memoria de Textura, Memoria Constante, Memoria Shared o compartida, Memoria Local y Registro.

CUDA permite definir una función denominada kernel que es ejecutada n veces, siendo n la cantidad de threads de la aplicación. Dicha función solo corre en la GPU o Device, y puede invocar código secuencial para ser ejecutado en la CPU. El kernel debe ser configurado con el número de hilos por bloque y por grid. Como el kernel se ejecuta en el device, la memoria debe ser alocada antes de que las función sea invocada y los datos que requiera utilizar copiados desde la memoria del host a la memoria del device. Una vez ejecutada la función kernel, los datos de la memoria del device deben ser copiados a la memoria del host. [NVI2008]

Actualmente, ha comenzado a utilizarse OpenCL como lenguaje para las GPU, ya que el mismo provee una abstracción respecto de la plataforma hardware utilizada. Sin embargo, si el objetivo final es obtener alta performance, su utilización debe ser estudiada cuidadosamente. [NOT2009]

Multicores y GPU

Estas dos plataformas brindan un vasto conjunto de posibilidades de investigación en arquitecturas híbridas, a partir de diferentes combinaciones a saber:

- cluster de máquinas unicore cada una con placa GPU, lo que permite combinar herramientas de programación paralela como MPI-CUDA
- máquinas multicore con más de una GPU, que combinan herramientas de programación paralela como OpenMP – CUDA o Pthread – CUDA.
- cluster de máquinas multicore cada una con plata de GPU, lo que permite combinar OpenMP-MPI-CUDA o Pthread-MPI-CUDA

Los desafíos que se plantean son múltiples, sobre todo en lo referido a distribución de datos y procesos en tales arquitecturas híbridas a fin de optimizar el rendimiento de las soluciones.

El tema del consumo

Un aspecto de interés creciente en la informática actual, principalmente a partir de las plataformas con gran cantidad de procesadores, es el del consumo energético que los mismos producen.

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.

Entre los puntos de interés pueden mencionarse:

- ✓ Caracterización energética de las instrucciones, tanto sobre procesadores multicores como GPUs.
- ✓ Caracterización de algoritmos complejos y de sistemas paralelos, desde el punto de vista energético (potencia máxima y consumo total).
- ✓ Diseño de microbenchmarks utilizables desde el punto de vista energético, para estudiar patrones en algoritmos de HPC.
- ✓ Modelos de predicción de perfomance energética.
- ✓ Análisis de esquemas de distribución de procesos entre procesadores, con ajuste dinámico de la frecuencia de clock en función del consumo.

Contadores de hardware

Todos los procesadores actuales poseen un conjunto de registros especiales denominados contadores de hardware [SPR2002]. Estos registros se pueden programar para contar el número de veces que ocurre un evento dentro del procesador durante la ejecución de una aplicación. Tales eventos pueden proveer información sobre diferentes aspectos de la ejecución de un programa (por ejemplo el número de instrucciones ejecutadas, la cantidad de fallos cache en L1, cuántas operaciones en punto flotante se ejecutaron, etc). Los procesadores actuales poseen una gran cantidad de eventos (más de 300) y la capacidad de usar hasta 11 registros simultáneamente [INT2012]. El acceso a estos recursos de monitorización se puede llevar a cabo usando diferentes herramientas en función del nivel de abstracción deseado: por ejemplo, linux provee una aplicación de usuario llamada perf [TUT2013] que puede ser utilizada para monitorizar eventos generados durante la ejecución de un programa sin necesidad de recompilar el código; PAPI [GAR2000] es una API que permite al programador obtener información sobre segmentos específicos de sus programas.

Interesa plantear la utilización de contadores de hardware en dos líneas:

- Optimización de programas. Los procesadores actuales en lugar de escalar su performance a través del incremento de su frecuencia, lo hacen duplicando recursos. Problemas como el exceso de consumo y calor debido a elevadas frecuencias de cómputo, la latencia del sistema de memoria y la ineficiencia del modelo de Von Neumann en procesadores muy rápidos que impulsaron a la industria en esta dirección siguen igual de vigentes. La dependencia que tiene un programa paralelo sobre su arquitectura para ser altamente eficiente requiere comprender los motivos que penalizan su desempeño. Los contadores hardware permiten acceder a información precisa sobre aspectos específicos de la ejecución de los programas, ayudando al programador en la tarea de encontrar esos motivos y comparar con datos concretos los beneficios de los cambios que realiza [TIN2013].

- Sintonización dinámica de aplicaciones. La información que proporciona la PMU (Unidad de Monitorización de Performance) de los procesadores puede ser utilizada para modificar el comportamiento o tomar decisiones en tiempo de ejecución. Esto permite construir algoritmos dinámicos de gran precisión, que se ajustan a los eventos que ocurren en el hardware, como por ejemplo algoritmos de planificación o balance de carga. En esta línea se está desarrollando una tesis de doctorado sobre optimización de herramientas de detección de errores de concurrencia [FRA2011] [FRA2012A] [FRA2012B], centrada en activar/desactivar herramientas de monitorización en función de los eventos que genera el protocolo de coherencia cache de los procesadores actuales.

LINEAS DE INVESTIGACION y DESARROLLO

- Arquitecturas híbridas (diferentes combinaciones de multicores y GPUs). Diseño de algoritmos paralelos sobre las mismas.
- Arquitecturas heterogéneas. Optimización de algoritmos en función de la potencia de cómputo.
- Lenguajes y Estructuras de Datos para nuevas arquitecturas de cómputo paralelo.
- Consumo energético, en particular en relación con clases de instrucciones y algoritmos paralelos. Modelos y predicción de perfomance energética en sistemas paralelos.
- Contadores de hardware. Aplicaciones en la optimización de aplicaciones paralelas.

Investigación experimental

- Desarrollo y evaluación de algoritmos paralelos sobre nuevas arquitecturas y análisis de rendimiento y consumo.
- Utilización combinada de cluster de multicores y cluster de GPUs.
- Empleo experimental de contadores de hardware, orientados a la detección de fallas de concurrencia y a la toma de decisiones sobre la frecuencia de clock de los procesadores en función del consumo.

FORMACION DE RECURSOS HUMANOS

Dentro de la temática de la línea de I/D se concluyeron 2 trabajos de Especialización y se encuentran en curso 6 tesis de Doctorado en Ciencias Informáticas y 3 tesinas de grado de Licenciatura.
Además, se participa en el dictado de las carreras de Doctorado en Ciencias Informáticas, y Magíster y Especialización en Cómputo de Altas Prestaciones de la Facultad de Informática de la UNLP (todas acreditadas A por la Coneau), por lo que potencialmente pueden generarse nuevas Tesis de Doctorado y Maestría y Trabajos Finales de Especialización.

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

BIBLIOGRAFIA

[BOR2005]S. Y. Borkar, P. Dubey, K. C. Kahn, D. J. Kuck, H. Mulder, S. S. Pawlowski, y J. R. Rattner, «Platform 2015: Intel® Processor and Platform Evolution for the Next Decade», Intel Corporation, White Paper, 2005.

[CAS2012] Casanova B., Balladini J., De Giusti A., Suppi R., Rexachs D., Luque E.. "Mejora de la eficiencia energética en sistemas de computación de altas prestaciones". XII Workshop de Procesamiento Distribuido y Paralelo. CACIC 2012. ISBN: 978987-1648-34-4. Pág. 377-386. Bahía Blanca, Buenos Aires, Argentina, Octubre 2012.

[CHA2007] Chapman B., The Multicore Programming Challenge, Advanced Parallel Processing Technologies; 7th International Symposium, (7th APPT'07), Lecture Notes in Computer Science (LNCS), Vol. 4847, p. 3, Springer-Verlag (New York), November 2007.

[CHA2011] 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.

[FED2009A] Alexandra Fedorova, Juan Carlos Saez, Daniel Shelepov and Manuel Prieto. Maximizing Power Efficiency with Asymmetric Multicore Systems. Communications of the ACM, Vol. 52 (12), pp 48-57. December 2009.

[FED2009B] Alexandra Fedorova, Juan Carlos Saez, Daniel Shelepov and Manuel Prieto. Maximizing Power Efficiency with Asymmetric Multicore Systems. ACM Queue, Vol. 7(10), pp. 30-45. November 2009.

[FRA2011] F. E. Frati, K. Olcoz Herrero, L. P. Moreno, D. M. Montezanti, M. Naiouf, y A. De Giusti, «Optimización de herramientas de monitoreo de errores de concurrencia a través de contadores de hardware», in Proceedings del XVII Congreso Argentino de Ciencia de la Computación, La Plata, 2011, vol. XVII, p. 10.

[FRA2012A] F. E. Frati, K. Olcoz Herrero, L. Piñuel Moreno, M. R. Naiouf, y A. De Giusti, «Unserializable Interleaving Detection using Hardware Counters», in *Proceedings of the IASTED International Conference Parallel and Distributed Computing and Systems*, Las Vegas, USA, 2012, pp. 230-236.

[FRA2012B] F. E. Frati, K. Olcoz Herrero, L. Piñuel Moreno, M. Naiouf, y A. E. De Giusti, «Detección de interleavings no serializables usando contadores de hardware», presented at the XVIII Congreso Argentino de Ciencias de la Computación, 2012.

[GAR2000] B. D. Garner, S. Browne, J. Dongarra, N. Garner, G. Ho, y P. Mucci, «A Portable Programming Interface for Performance Evaluation on Modern Processors», The International Journal of High Performance Computing Applications, vol. 14, pp. 189–204, 2000.

[INT2012] Intel® 64 and IA-32 Architectures Software Developer's Manual, Intel Corporation, Manual 253669-043US, may 2012. Recuperado a partir de http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html

[KIR2010] Kirk D., Hwu W. "Programming Massively Parallel Processors: A Hands-on Approach (Applications of GPU Computing Series). Morgan-Kaufmann. 2010.

[KOU2010] David Koufaty, Dheeraj Reddy, and Scott Hahn. Bias Scheduling in Heterogeneous Multi-core Architectures. In Proc. of Eurosys '10, 2010.

[LIN2011] 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.

[NOT2009] Nottingham A. y Irwin B. "Gpu packet classification using opencl: a consideration of viable classification methods". In Research Conf. of the South African Inst. of Comp. Sc. and Inf. Technologists. ACM, 2009.

[NVI2008] NVIDIA. "Nvidia cuda compute unified device architecture, programming guide version 2.0". In NVIDIA. 2008.

[PIC2011] M. F. Piccoli, "Computación de Alto Desempeño utilizando GPU". XV Escuela Internacional de Informática. Editorial Edulp, 2011.

[SAE2009] Juan Carlos Saez. Planificación de procesos en sistemas Multicore asimétricos. http://eprints.ucm.es/12668/1/T32909.pdf

[SAE2010] Juan Carlos Saez, Manuel Prieto, Alexandra Fedorova and Sergey Blagodurov. A Comprehensive Scheduler for Asymmetric Multicore Systems. In Proc. Of 5th ACM European Conference on Computer Systems (Eurosys '10), pp. 139-152. Paris, France. April 2010.

[SAE2011A] Juan Carlos Sáez (Universidad Complutense de Madrid), Manuel Prieto (Universidad Complutense de Madrid), Adrian Pousa (Universidad Nacional de La Plata), Alexandra Fedorova (School of Computing Science – Simon Fraser University - Canadá). Explotación de Técnicas de Especialización de Cores para Planificación Eficiente en Procesadores Multicore Asimétricos. XXII Jornadas de paralelismo. Universidad de La Laguna, Tenerife, España.

[SAE2011B] Juan Carlos Saez, Daniel Shelepov, Alexandra Fedorova and Manuel Prieto. Leveraging Workload Diversity Through OS Scheduling to Maximize Performance on Single-ISA Heterogeneous Multicore Systems", In Journal of Parallel and Distributed Computing (JPDC), Vol. 71, pp. 114-131. January 2011.

[SPR2002] B. Sprunt, «The basics of performance-monitoring hardware», IEEE Micro, vol. 22, n.o 4, pp. 64- 71, ago. 2002.

[SUR2007] Suresh Siddha, Venkatesh Pallipadi, Asit Mallick. "Process Scheduling Challenges in the Era of Multicore Processors"Intel Technology Journal, Vol. 11, Issue 04, November 2007.

[TIN2013] F. G. Tinetti, S. M. Martin, F. E. Frati, y M. Méndez, «Optimization and Parallelization experiences using hardware performance counters», in Proceedings of the 4th International Supercomputing Conference in México, Manzanillo, Colima, México, 2013, p. 5.

[TUT2013] Tutorial - Perf Wiki. (s. f.). Recuperado 6 de marzo de 2013, a partir de https://perf.wiki.kernel.org/index.php/Tutorial.