Desarrollo de Aplicaciones sobre Cluster Multi-core

Marcelo Alaniz, Verónica Gil-Costa, Virginia Mancini, Marcela Printista
Departamento de Computación
Facultad de Ciencias Físico Matemáticas y Naturales
Universidad Nacional de San Luis
Ejército de los Andes 950, 1º piso. (02652-420823)

CONTEXTO

La línea de investigación presentada en este trabajo recurre a dos grandes proyectos de investigación de la Universidad Nacional de San Luis, a) el proyecto de sistemas distribuidos y paralelos sobre arquitecturas multi-core; b) el proyecto de recuperación de la información y estructuras de datos.

En el primer proyecto nos enfocamos en el uso adecuado de tecnologías que permiten el desarrollo de nuevos algoritmos paralelos y distribuidos utilizando el modelo de programación paralela BSP [Val90] y librerías de programación asíncronas como MPI o PVM.

Haciendo uso de las características particulares de las arquitecturas conjuntamente con algoritmos de mapping, es posible delinear pautas que formen parte de una metodología de programación que tenga en cuenta la optimización de algoritmos BSP. Con los algoritmos de mapping se investiga la mejor asignación posible de tareas BSP a los recursos de procesamientos disponibles (nodos, cores o threads físicos) con el objetivo de disminuir la latencia de las comunicaciones, y al mismo tiempo mantener una cota mínima en el coeficiente de eficiencia de la utilización de recursos.

En el segundo proyecto nos enfocamos en el diseño y desarrollo de nuevas estructuras de indexación y algoritmos de búsqueda para datos, y también algoritmos de mapping que puedan ser ejecutados eficientemente sobre arquitecturas paralelas. En particular, los algoritmos de búsqueda e indexación son estudiados sobre colecciones de datos u objetos multimediales (sonidos, imágenes, video, etc.).

RESUMEN

Con la aparición de las CPU multi-cores (o Chip-level-Multi-Processor -CMP-), es importante el desarrollo de las técnicas que exploten las ventajas de las CMP para acelerar las aplicaciones paralelas que poseen una gran demanda de cómputo paralelo. En particular, para aplicaciones que requieren de un gran poder computacional de los recursos disponibles, es esencial poder desarrollar estrategias y algoritmos que aprovechen el uso adecuado del hardware.

En este trabajo se presentan los objetivos y los desafíos de una línea de investigación que abarca los problemas de mapping, uso adecuado de las nuevas arquitecturas de procesadores, y cómo estas nuevas arquitecturas pueden ser utilizadas para mejorar el desarrollo de algoritmos de computación de grafos y cálculo de matrices, utilizando como base formal el modelo de programación paralela BSP; conjuntamente con algoritmos de búsqueda e indexación sobre grandes colecciones de datos como la Web.

Palabras clave: *Sistemas de computación Híbridos. Tecnología Multi-core. Mapping. BSP. Bulk Synchronous Parallel. High Performance Computing. Estrategias de búsqueda. Espacios Métricos.*

1.INTRODUCCION

Los programas secuenciales pueden ser modificados para ser ejecutados en un ambiente distribuirlo o paralelo. Este tipo de modificaciones permite: acelerar la velocidad de ejecución y/o mejorar la cantidad de memoria disponible [Hol07].

Una forma de paralelizar los códigos es por medio del uso de ambientes distribuidos. El paradigma de los sistemas distribuidos permite conectar un gran número de máquinas personales de bajo costo. Además, permite utilizar la memoria perteneciente a diferentes procesadores, logrando de esta manera, eliminar el problema de la limitación de memoria RAM.

Otra alternativa surge en el mercado alrededor del año 2004, donde nuevas tecnologías de computación paralela incorporan en los procesadores chips con más de un core. Estos nuevos procesadores capaces de incrementar el poder de cómputo de las PCs, hace resurgir el paradigma de computación paralela olvidado en la década anterior. En la actualidad existe una gran variedad de procesadores multi-core y algunos de ellos incorporan tecnología Hyper-threading sobre los cores.

Los chips multi-core agregan nuevos niveles en la jerarquía de caché. Cada core posee una cache denominada L1 capaz de almacenar unos pocos Kb. Un segundo nivel de caché denominada L2 es compartida por todos los cores agrupados en el mismo chip y tiene una capacidad de almacenamiento de unos pocos Mb.

La tendencia actual de la tecnología destaca la aparición de procesadores CMP [Str69, Hamm97, Hamm00]. En realidad, la mayoría de los sistemas que incorporan estos chips descartan la vieja idea de un sistema multiprocesador como varios nodos mono-procesador. Las tendencias de la tecnología indican que el número de cores (núcleos) en un chip seguirá creciendo a medida que lo indiquen los planes de trabajo de los fabricantes más importantes. Hoy en día, AMD trabaja con chips de cuatro núcleos (Quad tecnología nativa) e Intel ha comenzado a incorporar el procesador Intel CoreTM de cuatro y ocho núcleos de procesador en sus sistemas.

Por lo tanto, existen dos claras alternativas para paralelizar aplicaciones secuenciales: (a) utilizar clusters de computadoras, y (b) utilizar sistemas multi-cores. Los sistemas multi-core difieren en varios aspectos respecto de un cluster de computadoras. Un sistema multi-core ofrece a todos los cores un acceso rápido a una única memoria compartida, evitando la transferencia de datos entre las máquinas a través de la red del cluster, pero aún así la cantidad de memoria disponible es limitada. En cambio los clusters de computadoras permiten incrementar el espacio de almacenamiento (principal y secundario) aunque deben pagar el precio de la latencia de la red. Una tercera alternativa (c) consiste en un sistema híbrido que permite combinar las características de ambos sistemas, e incrementar aún más la capacidad y el poder de los sistemas computacionales. Además, permiten combinar diferentes librerías de programación como MPI y OpenMP.

Esta combinación de diferentes jerarquías o niveles de hardware y lenguajes y librerías de programación, permite ejecutar **P** procesos en diferentes nodos o procesadores conectados mediante una red de intercomunicación y dentro de cada proceso **PT** threads (hilos) en paralelo, cada uno en un núcleo diferente utilizando un esquema en el cual todos los threads tienen acceso a la misma memoria principal. El desafío en este tipo de arquitecturas es reducir el tiempo de ejecución de los programas, lo cual para algunas aplicaciones como mapping de aplicaciones o las búsquedas multimediales sobre espacios métricos [Samet05] puede ser aún más difícil de lograr, debido a las competencias por los recursos causando por el acceso concurrente de los threads a los datos compartidos.

1.1 Modelo BSP

Existe una gran cantidad de problemas que se analizan, estudian e implementan utilizando el modelo de programación paralela BSP. Tales como Graph Computing, Matrix Computing, Page Ranking, y otros. En el modelo BSP [Val90] cada tarea realiza procesamiento local para luego realizar comunicaciones colectivas y finalmente sincronizar tareas. Estos 3 pasos se utilizan iterativamente para resolver un problema en particular.

Más formalmente el modelo BSP comprende una computadora abstracta, una clase de algoritmos, y una función de costos.
La computadora BSP, consiste de un conjunto de procesadores cada uno con su memoria privada y un canal de comunicación que permite acceder a un procesador a la memoria de los otros procesadores. Como se podrá suponer el acceso a memoria privada tiene un costo menor al de acceder a la memoria de otro procesador ya que el acceso deberá viajar a través del canal de comunicación.

Los Algoritmos BSP consisten de una secuencia de superpasos. Un **superpaso** contiene un número cómputos ó un número de comunicaciones, seguido por una barrera de sincronización global. En un **superpaso de cómputos**, cada procesador realiza una secuencia de operaciones sobre datos privados

(o locales), típicamente operaciones de punto flotante (flops). En un **superpaso de comunicación** cada procesador envía y recibe un número de mensajes. Al final del superpaso todos los procesadores **sincronizan**. En el caso de un superpaso de computación, la sincronización se realiza chequeando que se hayan terminado todos los cómputos de todos los procesadores. Y en el caso de un superpaso de comunicación se verifica que se hayan enviado todos los mensajes que se deberían haber enviado y si se recibió todos los mensajes que deberían recibirse. Este proceso de superpaso se utiliza interactivamente para el diseño de algoritmos paralelos.

La función de costos BSP nos permite evaluar la complejidad de los algoritmos desarrollados con este modelo, por lo tanto como podemos suponer BSP constituye una base robusta para el diseño, análisis e implementación de algoritmos paralelos.

1.2 Algoritmos de Mapping

Los algoritmos de mapping [Guil09, Song09] son ampliamente conocidos y estudiados en el contexto de sistemas distribuidos. Es así, que al conocer y estimar el comportamiento de las comunicaciones de un Algoritmo BSP particular, podemos rediseñar nuestra solución utilizando Multi-BSP. **Multi-BSP** es una versión del modelo BSP que tiene en cuenta distintos niveles de comunicación y recursos de procesamiento con distintas velocidades. Concretamente, es una redefinición de BSP para clusters Multi-core. Con Task-Mapping podemos establecer que tareas trabajaran en qué Nivel de MultiBSP, ubicando tareas con más comunicaciones entre sí en procesadores cuyos canales de comunicación sean más rápidos. Por lo tanto, esta reubicación de tareas culminará con una reducción de latencia en las sincronizaciones de los superpasos.

En resumen, lo que se desea es encontrar funciones de mapping que, en base al conocimiento que se tiene de la estructura del problema y de la arquitectura del cluster multi-core, disminuyan la latencia de sincronización en las tareas BSP.

Como se mencionó anteriormente, varios algoritmos de mapping ha sido estudiados sobre sistemas distribuidos [Giuse], y existen algunos trabajos preliminares que involucran sistemas multi-core [Guil09], y trabajos sobre MultiBSP [Val90]. Sin embargo, esta es un área que no ha sido completamente analizada y aún quedan diferentes técnicas que se pueden aplicar como clustering, machine learning, etc.

1.3 Recuperación de la Información

En la actualizad existen aplicaciones, como los motores de búsqueda Web como Google o Yahoo!, que deben ser capaces de indexar cientos de Pbytes y al mismo tiempo deben ser capaces de procesar millones de consultas por segundo [MM07a, MM07b, MM08a, MM09]. Otro tipo de aplicación que está ganando mayor popularidad en los últimos años, se relaciona con la búsqueda de objetos multimediales sobre espacios métricos. En la actualidad existen sistemas experimentales de búsqueda de imágenes y otros sistemas de mayor escala como Flickr (www.flikr.com).

Algunos trabajos presentados recientemente [Gil12,Gil11,Marin10] permiten planificar y optimizar el uso de índices métricos sobre sistemas distribuidos. También existen algunos métodos para planificar las tareas sobre un conjunto de núcleos. En [Hoffmann04] se muestra cómo reducir el *overhead* de un conjunto de tareas. En [Agrawal07] se presenta un *scheduler* (planificador) que utiliza información en tiempo de ejecución para determinar el número de núcleos que se debe asignar a un proceso. En [Hendler02] se presenta un algoritmo para *robar* tareas de otros núcleos de procesamiento. Finalmente en [Gil10] se presentan algunos algoritmos que permiten planificar la asignación de las tareas en un sistema multi-core. Recientemente, el trabajo presentado en [Bustos11] y [Bustos11b] realiza un estudio y análisis de paralelización de un índice para espacios métricos sobre una arquitectura híbrida. En [Bustos11] el índice es paralelizado utilizando un enfoque de particionado local en donde cada procesador recibe un sub-conjunto de objetos de la colección y construye su índice utilizando los datos locales.

Las búsquedas de consultas sobre espacios métricos, permiten realizar operaciones sobre bases de datos no estructuradas. Este tipo de búsqueda se conoce con el nombre de búsqueda aproximada o búsqueda por similitud, y surge en áreas tales como reconocimiento de voz, reconocimiento

de imágenes, compresión de video, minería de datos (data mining), recuperación de información, detección de copias, bases de datos médicas, etc. La necesidad de una respuesta rápida y adecuada, y un eficiente uso de memoria, hace necesaria la existencia de estructuras de datos especializadas que incluyan estos aspectos.

En este proyecto de investigación se propone extender los trabajos presentados en [Bustos11] y [Bustos11b] para optimizar el proceso de búsqueda por medio de técnicas de particionado de datos, técnicas de optimización de caché y actualizaciones dinámicas del índice.

2. LINEAS DE INVESTIGACION y DESARROLLO

La línea de investigación descripta en la sección anterior involucra una serie de desarrollos individuales que en su conjunto logran obtener el objetivo planteado. Para ello es necesario estudiar formas eficientes de monitorizar y/o estimar las comunicaciones en algoritmos BSP, para definir el mapping mas adecuado sobre el modelo Multi-BSP.

Además para optimizar la eficiencia sobre las cotas establecidas y aumentar el speed-up de los algoritmos BSP, es necesario determinar las características particulares de las aplicaciones. Es por esta razón que en un primer paso nos enfocaremos a problemas puntuales, a saber: Graph Computing y Matrix Computing. Todo esto teniendo en cuenta que es necesario utilizar estructuras de datos que sean capaces de manejar eficientemente grandes cantidades de datos (miles de Terabytes).

Por otro lado es necesario estudiar las características inherentes en los índices métricos para optimizar los procesos de búsqueda y obtener el mejor rendimiento posible aplicando diferentes técnicas de particionado y actualización.

3. RESULTADOS OBTENIDOS ESPERADOS

Algoritmos de Mapping

Los resultados obtenidos hasta el momento son:

- Descubrimiento de la Arquitectura de Clusters en tiempo de ejecución.
- Affinity de tareas combinado con descubrimiento de Arquitectura.
- Estudio de librerías existentes que implementan las primitivas del modelo de programación BSP.

Los resultados esperados son:

- Extender una librería BSP existente para soportar primitivas de Multi-BSP, combinado con descubrimiento de arquitectura y affinity de tareas.
- Extender al modelo Multi-BSP con Task-mapping. De manera tal de formalizar el presente trabajo como una extensión del Modelo Multi-BSP.
- Implementar metodologías que permitieran hacer una abstracción genérica de las topologías actuales y futuras de los clusters.

Algoritmos de búsqueda sobre espacios métricos

Los resultados obtenidos hasta el momento son:

- Diseño e implementación de códigos que combinan librerías de computación paralelas openMP (para memoria compartida) y MPI (para sistemas distribuidos), que permiten realizar búsqueda de consultas multimediales sobre índices que almacenan objetos del mismo tipo.
- Diseño e implementación de un índice métrico basado en un particionado de datos local.

Los resultados esperados son:

- Estudio, diseño e implementación de un índice métrico basado en particionado de datos globales.
- Estudio de algoritmo eficientes para la actualización dinámica del índice.
- Estudio de técnicas eficientes para optimizar el uso de los diferentes niveles de memoria caché.

4. FORMACION DE RECURSOS HUMANOS

Actualmente, se cuenta con dos doctores en ciencias de la computación realizando la investigación teórica y dirección de los algoritmos propuestos. También se cuenta con un alumno de doctorado que se encuentra

iniciando su carrera doctoral; y una alumna de grado próxima a finalizar su tesis.

Mediante este trabajo de investigación se podrán formar profesionales en el área de sistemas distribuidos y paralelismo que puedan modelar, diseñar e implementar algoritmos eficientes que se ejecuten en sistemas de clusters multi-core.

5. BIBLIOGRAFIA

[Agrawal07] K. Agrawal and Y. He and E. Leiserson. Adaptive work stealing with parallelism feedback. In Principles and Practice of Paralle Computing, pages 112-120. 2007.

[Bustos11] F. Bustos. Sistemas de búsquedas Multimediales en Ambientes Paralelos Híbridos. Tesis de Lic. en Cs. de la Computación. UNSL, Argentina 2011.

[Bustos11b] F. Bustos, M. Alaniz, V. Gil-Costa and M. Printista. Hybrid Architecture for Metric Space Searches. CACIC 2011, La Plata, Argentina. Pp. 312-323.

[Gil12] V. Gil-Costa, and M. Marin, "Load Balancing Query Processing in Metric-Space Similarity Search", In 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012), May 13-16, 2012, Canada.

[Gil11] V. Gil-Costa and M. Marin, "Approximate Distributed Metric-Space Search", ACM Workshop on Large-Scale and Distributed Information Retrieval (LSDS-IR 2011), Glasgow UK, Oct. 2011.

[Gil10] V. Gil-Costa, R. Barrientos, M. Marin and C. Bonacic, "Scheduling Metric-Space Queries Processing on Multi-Core Processors", , In 18th Euromicro International Conference on Parallel, Distributed and Network-Based Computing (Euro-PDP 2010), Pisa, Italy, Feb. 2010

[Giuse] Spingola G., Zito G., D'Ambrosio D., Rongo R., Spataro W. Dynamical Load Balancing in Cellular Automata Models. International Conference of the Italian Association for Artificial Intelligence" Reggio Emila, 2009.

[Guil09] Guillaume Mercier and Jerome Clet-Ortega. Towards an Efficient Process Placement Policy for MPI Applications in Multicore Environments. in the 16th European PVM/MPI pages 104-155, 2009.

[Ham00] L. Hammond, B. A. Hubbert, M. Siu, M. K. Prabhu, M. Chen, and K. Olukotun. The Stanford Hydra CMP. IEEE Micro, 20(2), 2000.

[Hamm07] L. Hammond, B. Nayfeh, and K. Olukotun. A single-chip multiprocessor. IEEE Computer, 30(9), 1997.

[Hendler02] D. Hendler and N. Shavit. Non-blocking steal-half work queues. In PODC, pages 280-289. 2002.

[Hoffmann04] R. Hoffmann and M. Korch and T. Rauber. Performance Evaluation of Task Pools Based on Hardware Synchronization. In Supercomputing Conference. 2004.

[Hol07] The Design of a Multi-Core Extension of the SPIN Model Checker. Gerard J. Holzmann and Dragan Bošnački. PDMC 2007, J. Electronic Notes in Theoretical Computer Science, pp. 33-46, 2007.

[MM07a] M. Marin and V. Gil-Costa. High-performance distributed inverted files. 16th ACM Conference on Information and Knowledge Management (CIKM 2007), 2007. pp. 935-938,ACM.

[MM09] M. Marin, F. Ferrarotti, M. Mendoza, C. Gomez and V. Gil-Costa. Location Cache for Web Queries. CIKM 2009, pp. 1995-1998.

[MM08a] Mauricio Marín, Carlos Gomez-Pantoja, Senen Gonzalez, Veronica Gil-Costa: Scheduling Intersection Queries in Term Partitioned Inverted Files. Euro-Par 2008: 434-443.

[MM07b] Mauricio Marín, Carolina Bonacic, Veronica Gil-Costa, Carlos Gomez: A Search Engine Accepting On-Line Updates. Euro-Par 2007: 348-357.

[Marin10] M. Marin, F. Ferrarotti and V. Gil-Costa, "Distributing a Metric-Space Search Index onto Processors", In 39th International Conference on Parallel Processing (ICPP 2010), San Diego, California, Sept. 13-16, 2010.

[Samet05] H. Samet. “Foundations of Multidimensional and Metric Data Structures” (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 2005.

[Song09] Static and Dynamic Scheduling for Effective Use of Multicore Systems. PhD Thesis. The University of Tennessee, Knoxville, 2009

[Str69] V. Strassen. Gaussian elimination is not optimal. Numerische Mathematik, 13(4), 1969. Springer.

[Val90] L. Valiant. A bridging model for parallel computation. Communication of the ACM, Vol 33. pp 103-111. 1990.