Diseño de Algoritmos para Plataformas Paralelas

Marcelo Alaniz, Fabricio Bustos, Verónica Gil-Costa, Virginia Mancini, Cesar Ochoa, 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 y de la estimación de costos que surgen del desarrollo de nuevos algoritmos paralelos y distribuidos. El diseño de estos algoritmos generalmente se realiza utilizando el modelo de programación paralela BSP [Val90], Multi-BSP[Val11] y librerías de programación asíncronas como MPI o PVM para comunicación inter-nodo y openMP o pthreads para comunicación intra-nodo.

Haciendo uso de las características particulares de las arquitecturas, es posible delinear pautas que formen parte de una metodología de programación que tenga en cuenta: la optimización de algoritmos BSP y su modelo de costo para una aproximación adecuada del tiempo de ejecución total del algoritmo en arquitecturas específicas. El desarrollo de estos temas de investigación también se enmarca dentro del proyecto 13STIC-06 – SEHLOC financiado por STIC-AMSUD.

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

RESUMEN

Con el creciente avance de la tecnología y la capacidad de computo que proveen las CPU multi-cores, es importante el diseño y desarrollo de las técnicas que exploten las ventajas de estos procesadores multi-cores 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 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. Multi-core. BSP. High Performance Computing. Estrategias de búsqueda. Espacios Métricos.*

1. INTRODUCCION

Estudios realizados recientemente indican que el incremento promedio en performance de procesadores ha excedido el incremento en la velocidad de reloj [Hennesy07, Hennesy08], lo que muestra que el incremento en el número de transistores en un chip da lugar a mejoras en la arquitectura de un procesador que reducen el tiempo promedio de ejecución de una instrucción. El número de transistores dentro de un chip puede ser usado como un estimador de su complejidad y performance. La ley de

Moore es una observación empírica que establece que el número de transistores de un típico chip se duplica cada 18-24 meses [Rauber10]. Esta observación fue realizada en 1965 por Gordon Moore y se ha mantenido vigente por más de 40 años.

Varias técnicas de diseño de microprocesadores han sido utilizadas para explotar el paralelismo interno de un único procesador. Entre las técnicas más relevantes se pueden mencionar paralelismo a nivel de bit, técnicas de pipelining y multiplicidad de unidades funcionales, entre otras [Culler99]. Estas tres técnicas asumen un único flujo de control secuencial el cual es provisto por el compilador y el cual determina el orden de ejecución en el caso de existir dependencias entre instrucciones. Para un programador, este tipo de técnicas internas tiene la ventaja de permitir ejecución paralela de instrucciones por medio de un lenguaje de programación secuencial. Sin embargo, el grado de paralelismo obtenido por pipelining y múltiples unidades funcionales está limitado.

Un enfoque alternativo para aprovechar el número creciente de transistores en un chip, es ubicar múltiples cores sobre un único chip de procesador [Hager11]. Este enfoque se ha estado utilizando en los procesadores de escritorio desde 2005 y son denominados procesadores multi-core. Estos nuevos procesadores capaces de incrementar el poder de cómputo, hacen resurgir el paradigma de computación paralela olvidado en la década anterior.

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. Un ejemplo es la máquina de Intel Sandy Bridge-E que tiene hasta 8 cores en un único chip, 32 Kb en la cache L1, 256 Kb en la cache L2 (esta cache no es compartida por los cores, sino que es privada a cada uno), y hasta 20Mb de cache L3 que es compartida por todos los cores de un mismo chip.

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. Los procesadores actuales como el IntelXeon Quad-core o el Intel Core i7 proveen cuatro CPUs físicas y ocho lógicas que pueden ser administradas por el programador mediante librerías como pthreads o OpenMP que posee un alto nivel de abstracción. Existen varias librerías de programación para sistemas multi-core como TBB [Reinders07] que utiliza ciclos para describir el paralelismo de datos y otros como IBM X10[Charles05] y Fortess [Allen07] que se focaliza en el paralelismo de datos pero también provee paralelismo de tareas.

Un sistema multi-core ofrece a todos los cores un acceso rápido a una única memoria compartida, pero la cantidad de memoria disponible es limitada.

El desafío en este tipo de arquitecturas es reducir el tiempo de ejecución de los programas. En particular, se propone estudiar, implementar y comparar la eficiencia y performance de algoritmos de búsqueda de texto (usando índices como el Arreglo de Sufijos[Manber93]) y datos multimediales utilizando diferentes índices métricos, y optimizar estos algoritmos al ser ejecutados en clusters de procesadores que soportan multi-core. Además, caracterizar las distintas arquitecturas de hardware multi-core utilizando benchmark desarrollados para tal propósito, y herramientas como hwlock y contadores de hardware (librería PAPI).

1.1 Diseño de Algoritmos Paralelos basados en el modelo BSP y Multi-BSP

El modelo Bulk-Synchronous Parallel (BSP)[Val90] propone que los algoritmos paralelos sean diseñados y evaluados no sólo por el balance clásico entre el tiempo y el número de procesadores, sino también por la comunicación y la sincronización. Este modelo establece un puente entre los algoritmos paralelos y las arquitecturas de hardware. La sencillez de la programación, la portabilidad y el modelo de costo asociado han estimulado el desarrollo de un gran número de algoritmos BSP paralelos.

Pero, con el paso del tiempo, uno de los supuestos básicos de BSP en relación a la visión de un hardware paralelo estaba cambiando. Hoy en día, la atención se centra en la utilización óptima de las arquitecturas complejas que no sólo son altamente escalables, pero también jerárquicas y no homogéneas.

El problema surge cuando se intenta adaptar estos algoritmos a la variedad de arquitecturas existentes. Y se debe principalmente a que BSP no tiene en cuenta el impacto de esta nueva complejidad. En estos casos, el desarrollador debe tener en cuenta afinidades de hardware cuando se trata de explotar el rendimiento del hardware real. Por ejemplo, dos tareas que cooperan estrechamente probablemente deberían ser colocadas sobre cores que comparten una memoria caché.

Software como Hardware Locality (hwloc, desarrollado por el grupo de investigación Inria Runtime) permite abstraer información de topología y exponerlo de una manera portable. Aunque esta información está disponible para la capa superior tal como MPI o implementaciones OpenMP, sólo se utiliza para ubicar las tareas sobre los cores en forma segura y basado en políticas proporcionadas por el usuario.

Dentro de este proyecto, se propone colaborar en el desarrollo de sistemas de tiempo de ejecución que combinan características de aplicación con información de topología. Un objetivo es ofrecer automáticamente sugerencias de programación que tratan de respetar las afinidades de hardware y software. Además se desea analizar la convergencia de los resultados obtenidos de los algoritmos con la recientemente propuesta multi-BSP[Val11], modelo que considera niveles anidados de cálculos que corresponden a las capas naturales de arquitecturas de hardware hoy en día.

1.2 Algoritmos de Búsqueda de texto y sobre Espacios Métricos

La línea de investigación presentada en este trabajo, propone el diseño y desarrollo de algoritmos de búsquedas textuales sobre arquitecturas paralelas, utilizando el índice denominado Arreglo de Sufijos. Un primer avance ha sido publicado en [Ochoa13].

Los arreglos de sufijos (también conocido como PAT array [Manber93]) es un arreglo ordenado de todos los sufijos de una palabra. Si representamos cada sufijo de una palabra w con el número de la posición en la palabra w en la que empieza el sufijo, entonces podemos ver al arreglo de sufijos como una permutación de estos números de 1 a n. Finalmente, la permutación particular que el arreglo de sufijos representa es la única permutación que ordena lexicográficamente los sufijos. En otras palabras, es un arreglo de enteros o punteros, y por lo tanto sólo está la información del orden lexicográfico de los sufijos, pero no hay información acerca del texto. Por lo que además del mencionado arreglo, es necesario contar con el texto.

Por otro lado, también se propone el diseño de algoritmos de búsqueda de datos multimediales. Los espacios métricos [Samet05] han demostrado ser útiles y prácticos para representar este tipo de datos multimediales. Además, permiten realizar búsquedas por similitud en grandes colecciones de objetos de datos. El objetivo en este tipo de búsquedas es recuperar los objetos más similares a una consulta dada. El grado de similitud entre dos objetos es determinado por una función que depende de la aplicación utilizada y se la denomina función de distancia. Generalmente, esta función es muy costosa de calcular y por este motivo se utilizan distancias pre-calculadas para indexar la base de datos a fin de reducir el número promedio de llamadas a esta función durante el proceso de búsqueda

Estas aplicaciones complejas requieren soluciones que demandan mayor poder computacional. Para resolver este tipo de problemas, es posible modificar los programas secuenciales para que sean ejecutados en un ambiente distribuido o paralelo. Este tipo de modificaciones permite acelerar la velocidad de ejecución y/o mejorar la cantidad de memoria disponible.

El procesamiento de consultas sobre ambientes distribuidos fue abordado por primera vez en [Papa01]. En este trabajo, los autores presentan resultados analíticos y experimentales que muestran que es posible obtener un sistema escalable sobre este dominio de aplicaciones. El trabajo presentado en [Papa01] es extendido en [Gil09] en el contexto de un índice basado en clustering para colecciones de espacios métricos. Aquí, se estudian varias alternativas para distribuir el índice entre los procesadores, concluyendo que el índice basado en particionado global obtiene una mejor performance que el índice basado en particionado local. El particionado global se refiere a construir un único índice considerando la colección de objetos completa. Luego el índice se distribuye uniformemente entre los procesadores. Sin embargo, una

desventaja del particionado global es su potencial a procesar consultas en forma desbalanceada. Esta situación de desbalance surge cuando varias consultas tienden a seleccionar el mismo conjunto de procesadores. Usualmente, las consultas de los usuarios tienden a ser sesgadas a segmentos particulares de la colección de objetos y este sesgo suele cambiar en forma dinámica y de forma impredecible.

El trabajo presentado en [Marin10] y extendido en [Gil12] propone una solución al problema de desbalance del particionado global para un índice basado en clustering. Sin embargo, el método propuesto requiere el pre-procesamiento con un costo de O(n2), del índice secuencial de tamaño n, para asignar las diferentes partes del índice. Adicionalmente, esta solución falla cuando las consultas cambian su foco a lo largo del tiempo.

El trabajo presentado en [Bustos11, Bustos11b] presenta la implementación de un índice métrico en un ambiente híbrido de computación paralela que combina el uso de memoria compartida con pasaje de mensajes. Se muestra la construcción del índice métrico utilizando un enfoque de particionado local de datos. Luego se explica el proceso de búsqueda en una arquitectura híbrida, especificando las funciones que desempeñan los distintos nodos y threads durante el procesamiento de consultas.

Posteriormente, el trabajo [Mancini12] extiende el sistema hibrido propuesto en [Bustos11] para analizar diferentes técnicas de particionado global.

2. LINEAS DE INVESTIGACION y DESARROLLO

La línea de investigación descripta en las secciones anteriores 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 y sincronizaciones en algoritmos BSP y Multi-BSP. Además para optimizar la eficiencia sobre las cotas establecidas y aumentar el speed-up de los algoritmos paralelos, es necesario determinar las características particulares de las aplicaciones.

Por otro lado es necesario estudiar las características inherentes en los índices tanto textuales como 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 Paralelos modelados con BSP y Multi-BSP

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:

- Diseño e implementación de un benchmark que permita medir el costo de sincronización en sistemas multi-core.
- Implementar metodologías que permitieran hacer una abstracción genérica de las topologías actuales y futuras de los clusters.

Algoritmos de búsqueda

Los resultados obtenidos hasta el momento son:

- Diseño e implementación algoritmos de búsqueda sobre arquitecturas hibridas.
- Diseño e implementación de un índice métrico basado en un particionado de datos local y global.
- Diseño e implementación de un algoritmo de búsqueda textual para sistemas multi-core.

Los resultados esperados son:

- Extender los trabajos presentados en [Bustos11] y [Mancini12] para estudiar algoritmos eficientes para la actualización dinámica del índice.
- Estudio de técnicas eficientes para optimizar el uso de los diferentes niveles de memoria caché.
- Realizar pre-procesamiento de consultas para la asignación eficiente de las mismas a los threads.

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; un alumno de maestría y dos alumnos de grado. Uno de los cuales está próximo 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

[Allen07] E. Allen and D. Chase and J. Hallett and V. Luchangco and W. Maessen and S. Ryu and S. Tobin-Hochstadt, S. The Fortress Language Specification, version 1.0beta. 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.

[Charles05] P. Charles and C. Grothoff and V.A. Saraswat and C. Donawa and A. Kielstra and K. Ebcioglu and von Praun and V. Sarkar. X10: an objectoriented approach to non-uniform cluster computing. In Internacional Conference on Object Oriented Programming, Systems, Languages and Applications. Pg. 519-538. 2005.

[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.

[Gil09] V. Gil-Costa, M. Marin, and N. Reyes. Parallel query processing on distributed clustering indexes. Journal of Discrete Algorithms (Elsevier), 7:3–17, 2009.

[Hager11] G. Hager and G. Wellein. Introduction to High Performance Computing for Scientists and Engineers. CRC Press, New York, 2011.

[Hennesy07] J. L. Hennesy and D. A. Patterson. Computer Architecture - A Quantitative Approach. Morgan Kaufmann, 4th edition, 2007

[Hennesy08]J. L. Hennesy and D. A. Patterson. Computer Organization & Design - The Hardware/Software Interface. Morgan Kaufmann, 4th edition, 2008.

[Manber93] Manber, U. and Myers, E. W. 1993. Suffix Arrays: A new method for on-line string searches. SIAM J. Comput. 22, 5, 935–948.

[Mancini12] “Data Partitioning Evaluation for Multimedia Systems in Hybrid Environments”. Virginia Mancini, Fabricio H. Bustos, Graciela Veronica Gil-Costa, Marcela Printista. 1st International Workshop on Soft Computing Techniques in Cluster and Grid Computing Systems SCCG 2012. November 12-14, 2012, University of Victoria, Victoria, Canada

[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.

[Ochoa13] “Suffix Array Performance Analysis for Multi-core Platforms”. Cesar Ochoa, Veronica Gil Costa, Marcela Printista. 4th International SuperComputing Conference in Mexico (ISUM).5-8 Marzo 2013

[Papoa01] N. Papadopoulos and Y. Manolopoulos. Distributed processing of similarity queries. Distrib. Parallel Databases, 9(1):67–92, 2001.

[Rauber10] Thomas Rauber and Gudula Rünger. Parallel Programming For Multicore and Cluster Systems. Springer, 2010.

[Reinders 07] J. Reinders. Intel Threading Building Blocks: Outfitting C++ for Multicore Processor Parallelism. OReilly (2007)

[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.

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

[Val11] L. G. Valiant. A bridging model for multi-core computing. J. Comput. Syst. Sci. 77(1): 154-166, 2011.