Optimización de Algoritmos utilizando Sistemas de Cómputo Híbridos

Marcela Printista, Verónica Gil-Costa, Marcelo Alaniz y Fabricio Bustos
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; 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. Haciendo uso de las características particulares de las arquitecturas es posible delinear pautas que formen parte de una metodología de programación con la finalidad de optimizar los algoritmos implementados.

En el segundo proyecto nos enfocamos en el diseño y desarrollo de nuevas estructuras de indexación y algoritmos de búsqueda, y también algoritmos de maping 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.). Mientras que en los algoritmos de mapping interesa estudiar la mejor asignación de procesos a nodos, de manera de optimizar la comunicación entre los procesos y al mismo tiempo, mantener un buen balance de carga.

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, las 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. Esto es especialmente crítico cuando se consideran sistemas o aplicaciones en las que los requerimientos ingresan en intervalos variables. En este trabajo se propone el desarrollo de técnicas híbridas basadas en el uso de MPI para la comunicación entre procesadores y OpenMP para la comunicación entre cores de un mismo procesador. OpenMP ha sido desarrollado para tomar ventaja de las facilidades multithreading de los nodos CMP.

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 búsqueda e indexación sobre grandes colecciones de datos como la Web.

Palabras clave: *Sistemas de computación Híbridos. Tecnología Multi-core. Mapping. Estrategias de búsqueda.*

1. INTRODUCCION

Modificar un programa secuencial para distribuirlo o para que se ejecute en paralelo tiene dos atractivas razones: acelerar la velocidad de ejecución y/o mejorar la cantidad de memoria disponible [Hol07].

Los primeros esfuerzos de la computación paralela se enfocaron en sistemas de memoria compartida. Grandes mainframes eran el centro de atención de los centros de investigación más importantes. Sin embargo, la época dorada del paradigma de computación paralela con memoria distribuida llega a su fin a finales de la década del ‘80. Estos sistemas estaban limitados por: (a) La falta de abstracción de las librerías de programación existentes, (b) La velocidad de crecimiento y avance de las nuevas tecnologías (chips de procesamiento) y (c) Por no poder administrar grandes volúmenes de datos en forma eficiente. En las mainframes, al utilizar un volumen de dato que superaba la capacidad de

almacenamiento de la memoria RAM, el sistema operativo debía realizar operaciones de swaping, demorando de esta manera, los tiempos de respuesta de los algoritmos ejecutados.

Posteriormente, surge el tiempo de los sistemas distribuidos. Este paradigma 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. Este paradigma surge como la solución a grandes problemas de la computación paralela, incrementando la popularidad y el auge de los sistemas distribuidos y paralelos.

Aproximadamente en el año 2004, surgen en el mercado, nuevas tecnologías de computación paralela que 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 siendo el i7 de Intel el más reciente. Este procesador incorpora tecnología Hyper-threading sobre cuatro 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 Figura 1 muestra la jerarquía de caché de un sistema quad-core.

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, Actualmente existen dos 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 parar 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 aun más la capacidad y el poder de los sistemas computacionales. Esta última alternativa es el eje central de la línea de investigación propuesta en este trabajo. Sin embargo el diseño de aplicaciones para este tipo de sistema trae aparejado una serie de complejidades que se describen en esta sección.

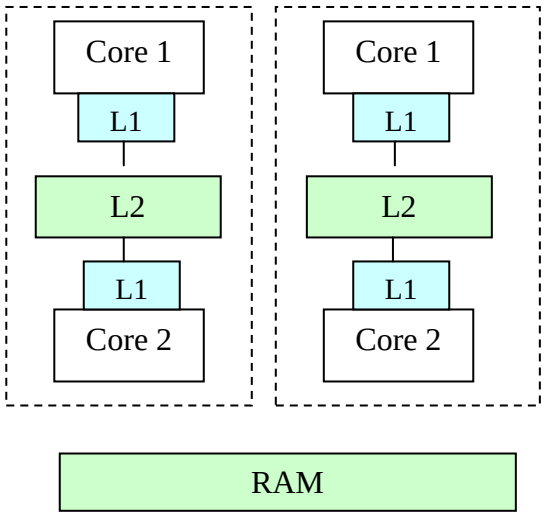

Figura 1: Jerarquía de memorias caché en un sistema quad-core.

Las mejoras que se obtienen al utilizar sistemas multi-cores se reflejan a través del paralelismo desarrollado para las aplicaciones. Si se ejecuta un programa secuencial sobre un sistema multi-core no se obtiene ninguna ganancia. Por lo tanto, la ventaja de estos sistemas consiste en dividir eficientemente la ejecución de una aplicación, para que cada parte sea ejecutada por un thread independiente. La cantidad de threads disponibles en estos

sistemas también es un tema importante a analizar, debido a que la creación y administración de threads requiere del uso de los recursos tales como la memoria; además los threads deben ser cuidadosamente planificados e incorporados en el snack de ejecución.

1.1 Recuperación de la Información

A pesar de que la nueva tecnología multi-core incrementa el poder de procesamiento de los computadores personales, no puede resolver el problema de la administración de grandes volúmenes de datos. 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. En la actualidad existen sistemas experimentales de búsqueda de imágenes y otros sistemas de mayor escala como Fickr (www.flkr.com).

Por lo tanto, con la aparición de los sistemas multi-core y el crecimiento desmedido de la información en sistemas que se ejecutan sobre la Web surge la necesidad de combinar sistemas de memoria compartida que permitan acelerar el procesamiento de datos y sistemas distribuidos que permitan manejar y administrar volúmenes grandes de datos. En este contexto, se enmarca los trabajos de investigación de recuperación de la información sobre sistemas híbridos de cómputo paralelo.

Todavía no hay estudios con esta tecnología que evalúen los paradigmas de programación paralela que trabajan en el contexto de las aplicaciones que resuelven requerimientos de los usuarios en un corto plazo y que requieren de un gran poder computacional, y si son las más apropiadas.

1.2 Algoritmos de Mapping

Los algoritmos de mapping [Guil09, Song09] son ampliamente conocidos y estudiados en el contexto de sistemas distribuidos. Existen problemas como la propagación de fuego, propagación de plagas, etc. que se analizan y estudian mediante el uso de autómatas celulares. En este caso el espacio de datos de entrada se divide en subgrillas, cada una de las cuales será responsabilidad de distintas tareas y puede ser procesada por un procesador diferente. En una aplicación paralela como la utilizada en este trabajo con patrones de comunicación estables, la comunicación entre tareas no cambia a lo largo de su ejecución. Cada tarea ejecuta sobre un procesador y los intercambios de datos se producen en la vecindad de los procesadores. Por lo tanto 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 jerarquía del cluster multi-core, optimicen la carga de comunicación.

Como se mencionó anteriormente, estos algoritmos de mapping ha sido estudiados sobre sistemas distribuidos [Giuse], y existen algunos trabajos preliminares que involucran sistemas multi-core [Guil09]. 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.

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 programación paralela utilizando el lenguaje OpenMP sobre arquitecturas multi-cores. Como se mencionó anteriormente, es de suma importancia reducir las regiones críticas o zonas de memoria compartidas para evitar los cuellos de botellas. También es necesario estudiar las ventajas y desventajas provistas por los sistemas de clusters de computadora que se comunican mediante librerías paralelas como MPI (asíncrona) y/o BSP (síncrona).

Para optimizar la eficiencia de loa algoritmos paralelos, es necesario determinar las características particulares de las aplicaciones. Además, es necesario utilizar estructuras de datos que sean capaces de manejar eficientemente grandes cantidades de datos (miles de Terabytes).

3. RESULTADOS OBTENIDOS ESPERADOS

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.
- Ejecución y análisis de resultados del código híbrido.
- Diseño e implementación de un framework que permita generar código paralelo híbrido (memoria compartida y sistema distribuido) automáticamente a partir de un código secuencial.

Los resultados esperados son:

- Extender el sistema híbrido utilizado para realizar búsquedas multimediales, de forma tal que se pueda generalizar mediante un modelo formalmente.
- Aplicar algoritmos de mapping estáticos sobre el framework construido.
- A través del conocimiento y experiencia adquirida en un tipo de cluster multicore (quad-core) se espera generalizar el mapping propuesto, y extrapolar la investigación a todo tipo de cluster, como por ejemplo, los que contienen 8 cores por nodo, donde la jerarquía de accesos es aún mayor.
- Implementar metodologías que permitieran hacer una abstracción genérica de las topologías actuales y futuras de los clusters. Con algunas diferencias, recientes publicaciones en el tema coinciden en proponer APIs o un sistema de tiempo de corrida (runtime system) para obtener la información de la topología en una manera portable. Esta información podrá ser utilizada convenientemente por el módulo de mapping para que explote al máximo el hardware subyacente, gracias al conocimiento detallado de sus características y para que el módulo pueda mantener un adecuado balance de carga de los procesadores asignados a la ejecución de AC.

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 un alumno de grado próximo a finalizar su tesis.

5. BIBLIOGRAFIA

[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.

[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.

[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.