Motores de Búsqueda Web Síncronos/Asíncronos

Gil-Costa V. and Printista M. *

LIDIC, Dpto. de Informática
UNSL
+54 (2652) 424027 - Fax: +54 (2652) 430224
Ejército de los Andes 950
5700 - San Luis, Argentina
e-mail: {*gvcosta, mprinti*}*@unsl.edu.ar*

Resumen

Los paradigmas de programación paralela síncronos y asíncronos son considerados como dos escuelas de modelado diferentes. La mayoría de los investigadores en el área de sistemas distribuidos y paralelismo tienden a creer que los programas asíncronos son más eficientes debido a que no requieren una sincronización periódica global. Sin embargo, existen algunos casos para los cuales esta creencia puede no ser correcta. En este trabajo presentamos una comparación de las dos estrategias de búsqueda paralela más populares sobre texto para motores de búsqueda Web que son implementados para aceptar cadenas de consultas en forma on-line, y diseñados para mejorar el pasaje de mensajes en $masa$ ($bulk$) para la cual los modelos síncronos tienden a tener un mejor rendimiento. Para la evaluación experimental utilizamos bases de datos reales sobre un cluster de computadoras de alta-performance obteniendo resultados que son consistentes a través de los modelos de computación y las máquinas. Nuestros algoritmos de procesamiento de consultas aseguran que ambas estrategias son comparados bajo las mismas condiciones.

Keywords: computación paralela, búsqueda de texto, listas invertidas.

1. Introducción

La mayoría de los investigadores tienden a creer que los métodos de comunicación asíncronos son más eficientes y permiten mejorar el tiempo de respuesta. Pero, ¿qué sucede cuando ésta conjetura no se mantiene para todos los casos de estudio? Este es un debate abierto que ha motivado el desarrollo de varios modelos de programación paralela, donde la mayoría de ellos intentan encontrar un modelo de costo con un adecuado nivel de abstracción [11, 8].

En este trabajo utilizamos dos métodos de pasaje de mensajes para implementar los algoritmos de búsqueda que se aplican sobre estructuras de indexación. El primer método es el esquema asíncrono

*Grupo soportado por la UNSL y ANPCYT (Agencia Nac. para la Prom. de la Ciencia y Tec.)

donde los procesadores trabajan independientemente, envían mensajes y reciben mensajes sin bloquearse. En este caso no se requiere una sincronización por barrera global. La librería seleccionada para implementar los algoritmos asíncronos es la bien conocida PVM [4].

Por otro lado, utilizamos el modelo de computación paralela *Bulk Synchronous Parallel - BSP* para implementar los algoritmos de búsqueda síncronos. En éste modelo, una computadora puede ser vista como una composición de P procesadores con memoria local que se comunican entre sí a través de mensajes. El cómputo se organiza en una secuencia de superpasos. Durante un superpaso, los procesadores pueden realizas cómputo secuencial local sobre los datos locales y/o enviar mensajes a otros procesadores. Los mensajes enviados se encuentran disponibles para su procesamiento al comienzo del siguiente superpaso, y cada superpaso finaliza con una sincronización por barrera [8].

El modelo de programación de BSP es SPMD, el cual se logra mediante copias de programas escritos en C y C++ ejecutados sobre P procesadores, donde la comunicación y sincronización entre las copias de programas se realizan a través de librerías tales como BSPlib [10].

BSP es un modelo una forma de computación paralela más conservativa pero igualmente efectiva. Nuestro estudio comparativo está basado en el modelo bulk-synchronous BSP [11, 8]. Éste es un modelo libre de $deadlocks$ y tiene una manera particular de organizar el cómputo en superpasos, y la performance obtenida es muy similar a la obtenida con algoritmos completamente asíncronos. En este trabajo, utilizamos la librería BSPonMPI que permite ejecutar con primitivas de comunicación MPI programas diseñados mediante el modelo BSP. De esta forma, logramos obtener una comparación justa de los modelos de comunicación.

Con el objetivo de balancear la carga de trabajo y obtener un sistema de comparación justo para ambos paradigmas de programación paralela (sync/async), aplicamos el concepto de round-robin para asignar un $quantum$ de trabajo a cada consulta dentro del sistema. Nos referimos a la estrategia clásica de round-robin para administrar un conjunto de tareas que compiten para obtener tiempo de CPU. Este esquema puede ser visto como una *sincronización en masa* en el sentido de que las tareas pueden realizar un número fijo de operaciones durante su $quantum$. Esta asignación limitada de un $quantum$ a las consultas permite un mejor uso global de los recursos del servidor evitando la monopolización de los mismos; y a su vez mejorar el tiempo de ejecución para las consultas que requieren menos uso de estos recursos.

2. Plataforma Paralela

La plataforma seleccionada para realizar las implementaciones y los experimentos consiste en un cluster de computadoras conectadas mediante una tecnología *fast switching*. Asumimos un servidor que opera sobre un conjunto de P máquinas, cada una con su memoria local. Los requerimientos de los clientes son enviadas a una máquina broker quien a su vez distribuye uniformemente estos requerimientos sobre los P procesadores del servidor. Los requerimientos son consultas que deben ser resueltas utilizando los datos almacenados en los P procesadores. Básicamente, cada procesador debe tratar con dos tipos de consultas, aquellas provenientes de la máquina broker en cuyo caso la consulta comienza a resolverse en el procesador que recibe la consulta; y aquellas consultas asignadas a otros procesadores pero que debieron continuar su búsqueda local en éste procesador.

Cada procesador procesa una secuencia de iteraciones formadas por tres fases principales: recibir mensajes (consultas), procesar mensajes y enviar mensajes. Dentro de la fase los procesadores pueden realizar tres operaciones dependiendo del tipo de mensaje recibido: a) broadcast del mensaje, b) buscar documentos relevantes para la consulta y c) ranking de los documentos recuperados.

La implementación de los algoritmos se ha llevado a cabo a través de la programación orientada a objetos y utilizamos programas $esqueletos$ para hacer los programas portables. Durante el envío de

mensajes a través de la red se evita el envío de paquetes pequeños, y se favorece el envío en masa copiando pequeños paquetes en una único mensaje por procesador.

3. Motores de Búsqueda Web para Texto

No hay duda de que la Web es un enorme desafío con el que se debe tratar hoy en día. Varios estudios han estimado el tamaño de la Web [12], y mientras la diferencia reportada por éstos es mínima, la mayoría está de acuerdo en que existe más de un billón de páginas disponibles. Los buscadores Web son aquellas máquinas que nos facilitan la búsqueda de información en este inmenso espacio. Actualmente estas máquinas sólo permiten realizar búsquedas de texto, y para ello utilizan los índices invertidos o listas invertidas como estructuras de indexación. Las listas invertidas son estructuras de datos de indexación que permiten realizar búsquedas rápidas sobre grandes colecciones de texto, y consisten de una tabla de vocabulario que posee todos los términos o palabras relevantes encontradas en la colección de documentos y una lista asociada por cada término. La lista asociada consiste de pares de identificadores de documentos y la frecuencia con la que aparece el término en el documento.

Varias publicaciones han presentado experimentos y propuestas para el procesamiento paralelo eficiente de consultas sobre las listas invertidas que están distribuidas en P procesadores [2, 3, 1, 5, 6, 9, 13, 14]. Es evidente que la eficiencia sobre un cluster de computadoras sólo se logra usando estrategias que permiten reducir la cantidad de comunicación entre los procesadores, y mantener un balance razonable sobre la cantidad de cómputo y comunicación realizada por cada procesador para resolver la búsqueda de consultas.

Existen dos estrategias de distribución de listas invertidas sobre un conjunto de procesadores predominantes: (**a**) la partición de documentos en la cual los documentos son uniformemente distribuidos sobre los procesadores y la lista invertida se construye en cada procesadores usando el respectivo subconjunto de documentos, y (**b**) la partición de términos donde se construye un único índice invertido secuencial y luego se distribuye cada término con su lista invertida sobre los procesadores. Además de estas dos estrategias predominantes existen algunas estrategias híbridas que intentan mejorar el balance de carga [5, 13]. La forma en que las listas invertidas son particionadas entre los procesadores determina la manera en que se realiza el procesamiento paralelo de las consultas.

La mayoría de las implementaciones de listas invertidas presentadas hasta el momento, están basadas en la programación paralela con pasaje de mensajes en las que se pueden ver combinaciones de $multithreaded$ y sistemas de solapamiento de cómputo/comunicación. Utilizando estas formas desordenadas de computación paralela es bastante riesgoso hacer afirmaciones razonables sobre la performance de los algoritmos. El problema con estas aproximaciones es que las ejecuciones son muy dependientes del estado particular de la máquina y sus fluctuaciones. Por otro lado, el uso de artefactos como los $threads$ son fuentes potenciales de $overheads$ y pueden producir salidas impredecibles en términos de tiempo de ejecución. La principal ventaja de BSP es que tiene un modelo de costo que permite evaluar los costos de cómputo y comunicación de los algoritmos paralelos.

El procesamiento paralelo de consultas está compuesto básicamente en una fase en la que es necesario obtener las listas invertidas de cada término de la consulta y realizar un ranking de documentos para producir los resultados. Las consultas llegan al servidor paralelo desde una máquina recepcionista llamada $broker$. Luego esta máquina broker envía las consultas a una máquina del servidor que es seleccionada en forma circular. Ésta máquina también será la encargada de realizar posteriormente la operación de ranking.

Cada consulta es procesada en dos etapas: la primera consiste en buscar las listas de tamaño K para cada término de la consulta y enviarlo al ranker. En la segunda, el ranker realiza el ranking de

documentos y si es necesario pide otras listas de tamaño K. A este esquema lo denominamos ranking iterativo. Para realizar el ranking de documentos utilizamos el modelo vectorial con una técnica de filtro propuesta en [7].

La Figura 1 muestra los tiempos de ejecución obtenidos con las dos estrategias de indexación de texto para máquinas de búsqueda Web más populares. A la derecha, se pueden observar los valores obtenidos con un modelo de computación paralela síncrona (BSP) y a la izquierda se muestran los tiempo de ejecución obtenidos por un modelo asíncrono (PVM). Es ésta figura se analiza el comportamiento de ambas estrategias de indexación D y T con diferentes cargas de trabajo en cada iteración o superpaso. A medida que se aumenta el tamaño del lote de consultas insertado por superaso (eje X), el modelo síncrono tiende a mejorar su rendimiento. Caso contrario a lo que sucede con el modelo asíncrono.

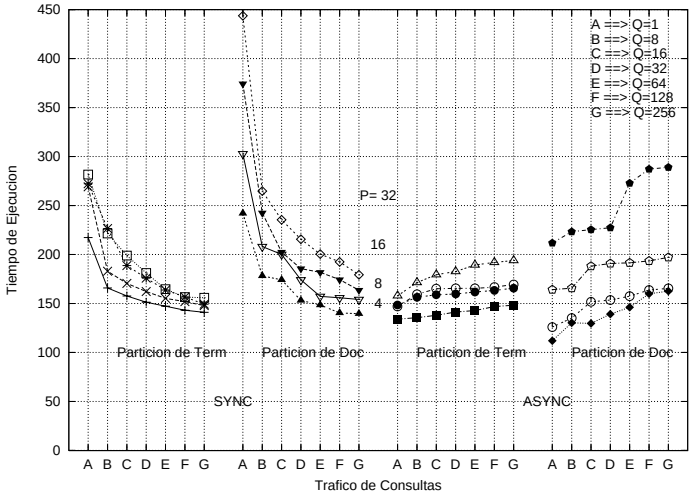

Figura 1: Sistema síncrono vs. asíncrono para una máquina de búsqueda Web utilizando listas invertidas como estructuras de indexación.

4. Conclusiones y Trabajo Futuro

Hasta el momento hemos logrado estudiar en profundidad las estrategias de indexación existentes en el contexto de búsquedas en la Web. Hemos implementado las estrategias existentes y hemos propuesto nuevas estrategias que permiten balancear no sólo la carga de trabajo que tiene cada procesador en el servidor, sino también la comunicación realizada entre estos durante la resolución de consultas.

El trabajo ha sido realizado utilizando el modelo de computación paralela BSP, que es un modelo sincrónico y censillo de utilizar. La implementación de los algoritmos se lleva a cabo utilizando la librería BSPlib y BSPonMPI que permite ejecutar códigos escritos siguiendo el modelo BSP bajo la plataforma mpi. Hemos podido comprobar que las aplicaciones en BSP son competitivas con otras implementaciones asíncronas.

Nuestro siguiente paso consiste en verificar los resultados obtenidos hasta el momento sobre estructuras no convencionales como las estructuras de indexación utilizadas en espacios métricos para la búsqueda de objetos multimediales.

Referencias

[1] C. Badue, R. Baeza-Yates, B. Ribeiro, and N. Ziviani. Distributed query processing using partitioned inverted files. *Eighth Symposium on String Processing and Information Retrieval (SPIRE'01)*, pages 10–20, Nov. 2001.

[2] R. Baeza and B. Ribeiro. *Modern Information Retrieval*. Addison-Wesley., 1999].

[3] A. Barroso, J. Dean, and U. H olzle. Web search for a planet: The google cluster architecture. *IEEE Micro*, 23(2):22–28, 2002.

[4] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Mancheck, and V. Sunderam. *PVM: Parallel Virtual Machine - A Users Guide and Tutorial for Network Parallel Computing*, 1994. MIT Press.

[5] Alistair Moffat, William Webber, and Justin Zobel. Load balancing for term-distributed parallel retrieval. *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 348–355, 2006.

[6] S. Orlando, R. Perego, and F. Silvestri. Design of a parallel and distributed web search engine. *In Proc. 2001 Parallel Computing Conf.*, pages 197–204, 2001.

[7] M. Persin, J. Zobel, and R. Sacks-Davis. Filtered document retrieval with frequency-sorted indexes. *Journal of the American Society for Information Science*, 47(10):749–764, 1996.

[8] D.B. Skillicorn, J.M.D. Hill, and W.F. McColl. Questions and answers about BSP. Technical Report PRG-TR-15-96, 1996. Also in *Journal of Scientific Programming*, V.6 N.3, 1997.

[9] A. Tomasic and H. Garcia-Molina. Performance of inverted indices in shared-nothing distributed text document information retrieval systems. *Second International Conference on Parallel and Distributed Information Systems*, pages 8–17, 1993.

[10] URL. BSP and Worldwide Standard, http://www.bsp-worldwide.org/.

[11] L.G. Valiant. A bridging model for parallel computation. *Comm. ACM*, 33:103–111, Aug. 1990.

[12] I. H. Witten, A. Moffat, and T.C. Bell. *Managing gigabytes: Compressing and indexing documents and images*. 2nd ed. San Francisco, Morgan Kaufmann, 1999.

[13] Wensi Xi, Ohm Sornil, Ming Luo, and Edward A. Fox. Hybrid partition inverted files: Experimental validation. In *ECDL '02: Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries*, pages 422–431, London, UK 2002.

[14] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Computing Surveys*, 38(2), 2006.