Estrategias Paralelas para una Máquina de Búsqueda Línea de investigación: Distribución y Paralelismo

V. Gil Costa, M. Printista
Departamento de Informática
Universidad Nacional de San Luis
San Luis, Argentina
{gvcosta,mprinti}@unsl.edu.ar

M. Marín
Departamento de Computación
Universidad de Magallanes
Punta Arenas, Chile
mmarin@ona.fi.umag.cl

Introducción

A través de los años con el avance de la tecnología y el crecimiento de Internet, el hombre ha deseado obtener una mayor capacidad de almacenamiento para sus datos y un mayor poder de procesamiento, que permita mejorar los tiempos de respuestas de sus tareas. Debido a ésto, numerosos estudios se han abocado al desarrollo de nuevos modelos que permitan satisfacer dichas demandas, a través de la computación paralela, que ha demostrado ser un paradigma que permite mejorar los tiempos de ejecución de los algoritmos.

En trabajos anteriores se han analizado y desarrollado algoritmos paralelos, para estructuras de datos que acceden eficientemente a base de datos textuales distribuidas. Estos algoritmos, denominados Listas Invertidas Locales y Listas Invertidas Globales, fueron diseñados e implementados siguiendo una metodología de diseño y de análisis bien estructurada, a través del modelo de computación *BSP* [2, 4]. Actualmente se están estudiando diferentes alternativas, denominadas estrategias de buckets, que permiten reducir los costos asociados a los algoritmos de búsqueda en la Web, y también la relación existente entre la máquina de búsqueda y los módulos correspondientes al indexador y el crawler.

El crawler es el proceso encargado de recuperar información de la Web, y el indexador es quien organiza esta información en el sistema. La eficiencia y escalabilidad de un motor de búsqueda (máquina de búsqueda, indexador y crawler), están relacionadas con el indexador, que mantiene actualizadas los índices utilizados por la máquina de búsqueda. La metodología seleccionada para realizar las búsquedas e indexar los datos, en la máquina de búsqueda, determina las operaciones que deberá desempeñar el indexador al momento de actualizar las listas invertidas.

Las nuevas estrategias de búsqueda que se proponen, permiten tener una mayor cantidad de alternativas al momento de elegir una estrategia de indexación, y están relacionadas con el concepto de *buckets* o bloques, a través de los cuales se pretende reducir los costos de cpu, así como el espacio de almacenamiento requerido en memoria principal, para mantener las estructuras que permiten indexar las consultas a la base de datos textual. Para ello, se trabaja con un servidor formado por P procesadores y al menos una máquina *broker* que actúa como intermediaria entre los procesadores del servidor y los usuarios. La máquina broker es la encargada de recibir las consultas provenientes de los usuarios y con alguna metodología debe derivarlas a las máquinas del servidor. Además, por cada consulta recibida, uno de los P procesadores del servidor cumplirá el papel de máquina *ranker*, la cual es seleccionada por el broker durante la distribución de las consultas. Esta máquina será la encargada de realizar el ranking final de documentos que será entregado como resultado al usuario. La máquina broker,

dependiendo de la estrategia de distribución que se esté utilizando, también deberá seleccionar una máquina *víctima* a través de la cual, las consultas arribarán a servidor.

El diseño del servidor está basado en el modelo de computación *Bulk-Synchronous Parallel - BSP*, propuesto en 1990 por Leslie Valiant [8]. La idea fundamental de este modelo es la división entre computación y comunicación. Se define un "paso" como una operación básica que se realiza sobre los datos locales de un procesador. Todo programa BSP consiste en un conjunto de estos pasos, denominados superpasos. En los cuales primero existe una fase de computación local independiente, le sigue una fase global de comunicaciones y finalmente una sincronización por barrera que permite separar los diferentes superpasos. Cualquier petición de datos remotos se puede realizar durante el superpaso, pero estos datos no se podrán utilizar hasta entrar al siguiente superpaso, después de la sincronización.

Para resolver las consultas provenientes de las máquinas de los usuarios, la máquina de búsqueda indexa cada una de las palabras que integran a las consultas a través de la lista invertida. Esta lista puede ser vista como una tabla de vocabulario, donde cada entrada contiene un término (palabra de importancia en el documento) y una lista asociada de referencias [1]. Cada entrada de la lista contiene el identificador de los documentos que contienen al término, y sus frecuencias.

Las estrategias de buckets dividen las listas asociadas de cada término en bloques de tamaño K, y los distribuyen entre los procesadores del servidor para reducir los tiempos de procesamiento, y disminuir la cantidad de datos que deben ser recuperados desde la memoria secundaria para realizar el procesamiento de consultas. Se han propuesto dos estrategias: Buckets distribuidos en diferentes Procesadores y Buckets distribuidos en diferentes superpasos, de acuerdo al modelo BSP. Para la primera estrategia, es posible utilizar cuatro tipos de distribuciones: Uniforme-Secuencial, Uniforme-Circular, utilizando una función de Hash y una distribución Random. Para realizar la construcción de la lista invertida se considera toda la colección de documentos de la base de datos textual, y las listas asociadas a cada término son divididas en buckets de tamaño K. Las listas asociadas de cada término, consisten de pares $< d, f_{d,t} >$, donde d es el identificador del documento, y $f_{d,t}$ es la frecuencia del término t en el documento d.

En la distribución uniforme secuencial, cada procesador recibe aproximadamente la misma cantidad de pares por cada término (si N es el número de pares para un término, luego $K = N/P$), de forma tal que el $bucket_i$ del término t es enviado al $procesador_i$. Una desventaja de esta distribución, es el desbalance de carga existente durante el procesamiento de las consultas, debido a que los procesadores que poseen un valor de identificador lógico bajo, tendrán buckets con frecuencias más altas. Por otro lado, la distribución circular agrupa los pares en buckets de tamaño K, pero a diferencia de la distribución anterior, los buckets son distribuidos entre los procesadores en forma circular. Es decir que los buckets del $término_1$ son distribuidos siguiendo la secuencia $P_0, P_1, P_2,..,P_{P-1}$, luego los buckets del $término_2$ son distribuidos a $P_1, P_2,..,P_{P-1}, P_0$, y así sucesivamente, permitiendo obtener un mejor balance de carga durante el procesamiento de las consultas.

El procesamiento de consultas para estas dos estrategias consiste de tres superpasos. En el primero, la máquina víctima (previamente seleccionada por el broker), recibe las consultas y realiza un *broadcast* de las mismas. Notar que este superpaso requiere una gran cantidad de comunicación, pero el cómputo realizado es mínimo. En el segundo superpaso, todas las máquinas del servidor reciben las consultas y realizan un preranking para obtener un resultado parcial con los mejores documentos encontrados en cada máquina. Luego, envían este resultado parcial a la máquina ranker. Finalmente, en el último superpaso, la máquina ranker efectúa el ranking final con los resultados parciales recibidos, y le envía los mejores documentos seleccionados al broker.

En las últimas dos distribuciones, hash y random, se trabaja con buckets de tamaño variable en un rango de $2, .., N - 1$. Para distribuir los buckets, en la tercera distribución presentada,

se utiliza una función de hash que considera el término (debido a que algunos términos tienen mayor probabilidad de aparecer que otros), el número de identificador del bucket (para que no todos vayan al mismo procesador), y el número de procesadores que se está utilizando. El tamaño del K determinará la granularidad de las operaciones realizadas en cada superpaso, y dependiendo de esta granularidad (fina o gruesa) es la cantidad de cómputo que deberá realizarse en cada procesador, afectando directamente el costo de las comunicaciones efectuadas.

La diferencia que existe entre la distribución de hash y la random, es que la segunda utiliza una estructura adicional para saber qué procesadores poseen buckets para un determinado término. Por lo tanto el broker deberá utilizar esta estructura al momento de seleccionar la máquina víctima que recibirá la consulta. La finalidad de utilizar una distribución aleatoria, es para poder medir que tan buena es la función de hash seleccionada. Estas dos distribuciones reducen la probabilidad de que un mismo procesador reciba más de un bucket perteneciente al mismo término, cuando se trabaja con buckets suficientemente grandes, permitiendo de esta manera obtener un mejor balance de carga durante el procesamiento de las consultas.

Durante el procesamiento de las consultas, los procesadores reciben las consultas desde el broker (primer superpaso), recuperan las listas asociadas para cada término de la consulta para seleccionar los mejores documentos, y envían un resultado parcial a la máquina ranker. En el segundo superpaso, el ranker recibe los resultados parciales y realiza el ranking final para seleccionar los mejores documentos y le envía el resultado a la máquina broker.

Finalmente, la estrategia que distribuye los buckets a través de los diferentes superpasos definidos por el modelo de computación BSP, consiste en dividir las listas asociadas de cada término en buckets de tamaño K, manteniendo sólo uno de ellos en memoria, el que posee las frecuencias más altas, pero a diferencia de la estrategia anterior cada procesador del servidor tiene los términos de la tabla de vocabulario que le fueron asignados con la lista asociada completa. Es decir que para efectuar la construcción de la lista invertida distribuida, los términos son repartidos entre los procesadores del servidor junto con sus listas asociadas completas.

El procesamiento de las consultas siguiendo el modelo BSP, consiste de tres pasos iterativos. (1) Primero los procesadores del servidor, reciben las consultas, las procesan recuperando las listas asociadas y seleccionando los mejores documentos encontrados, para luego enviarle al ranker el resultado parcial. (2) Luego el ranker recibe estos resultados parciales y verifica si algún procesador posee un bucket en memoria secundaria, con frecuencias más altas que las recibidas. Si esto no sucede, el ranker realiza el ranking final de los documentos y envía el resultado al broker. En el otro caso, si se necesitan más buckets porque algún procesador posee un bucket en memoria secundaria con mayores frecuencias que las recibidas, el ranker envía un mensaje con los términos correspondientes al procesador que posee buckets con estas frecuencias más altas. (3) Este superpaso es ejecutado sólo si en el superpaso anterior se requirieron más buckets. Aquí, los procesadores reciben los mensajes con los términos que requieren el próximo bucket, lo recuperan y se lo envían nuevamente al ranker. Luego se regresa al segundo paso.

Una optimización que se ha aplicado a estas estrategias de búsqueda es la utilización de filtros [6] para descartar, al momento de procesar una consulta, los documentos de menor importancia y ganar así una reducción en el tiempo de cpu requerido. Aunque esta técnica de filtrado no beneficia a todas las distribuciones presentadas para la estrategia de buckets distribuidos en diferentes procesadores, debido a que la distribución uniforme secuencial se verá perjudicada. Esto es porque los procesadores con identificadores lógicos bajos poseen documentos con frecuencias más altas que los procesadores con identificadores lógicos altos, dejando a éstos últimos ociosos al momento de procesar una consulta, provocando un mayor desbalance de carga.

Conclusiones

Para los algoritmos presentados, utilizados por una maquina de búsqueda web y basados en el modelo de computación paralelo BSP, se ha podido observar que las estrategias que utilizan buckets tienen un mejor comportamiento que las estrategias anteriormente estudiadas (la indexación local y global), debido a que éstas tienen como objetivo dividir las listas invertidas en buckets de tamaño K, y distribuirlos entre los procesadores del servidor, para poder reducir por un lado el espacio de almacenamiento requerido en memoria principal por las listas invertidas en cada procesador, y por otro lado, el tiempo de procesamiento requerido (tiempo de cpu) para realizar la operación de ranking.

El motivo por el cual se ha trabajado con las listas invertidas como estructuras de datos a través de la cual se realiza la indexación, es porque éstas permiten obtener un procesamiento eficiente de consultas a bases de datos textuales.

Trabajo Futuro

Como trabajo futuro se pretende profundizar el análisis de resultados obtenidos por las estrategias presentadas utilizando distintas base de datos reales como la Web Chilena, Fibrosis (http://www.dcc.ufmg.br/irbook/cfc.html) y Maggazine (ftp://ftp.cs.cornell.edu/pub/smart/time/), para comparar exhaustivamente el desempeño de cada una ellas y compararlas con las estrategias ya existentes (listas invertidas locales y globales).

Además se estudiarán otro tipo de estructuras de datos como las estructuras de árboles (SAT,dSAT,etc.), que también permiten realizar búsquedas de textos y otra información multimedial como sonidos, videos, etc.

Referencias

[1] R. Baeza and B. Ribeiro. "Modern Information Retrieval". Addison-Wesley. 1999.

[2] G. V. Gil Costa, "Procesamiento Paralelo de Queries sobre Base de Datos Textuales". Universidad Nacional de San Luis. 2003

[3] Veronica Gil Costa, A. Marcela Printista. "Estrategia de Buckets para Listas Invertidas Paralelas". XII Jornadas Chilenas de computación. Arica, Chile. 8-12 de noviembre del 2004.

[4] M. Marin, C. Bonacic y S. Casas. "Analysis of two indexing structures for text databases", Actas del VIII Congreso Argentino de Ciencias de la Computación (CACIC2002). Buenos Aires, Argentina, Octubre 15 - 19, 2002.

[5] R. Miller. "A library for Bulk Syncrhonous Parallel programming". Proceedings of the BCS Parallel Processing Specialist Group workshop on General Purpose Parallel Computing, Pp 100-108. December, 1993.

[6] M. Persin, J. Zobel, R. Sacks-Davis. "Filteres Document Retriaval with Frequency-Stores Indexes". Journal of the American Society for Information Science, 1996.

[7] C. Santos Badue, R. Baeza-Yates, B. Ribeiro-Neto, and N. Ziviani. "Concurrent query processing using distributed inverted files". In the 8th. International Symposium on String Processing and Information Retrieval, pages 10-20, 2001.

[8] L.G. Valiant. "A Bridging Model for Parallel Computation". Communications of the ACM, Vol. 33, Pp 103-111, 1990.