Algoritmos para Listas Invertidas Paralelas
Línea de Investigación: Distribución y Paralelismo

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

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

Introducción

Las listas invertidas [1] son estructuras de datos populares frecuentemente utilizadas como índices para bases de datos textuales. Su propósito es acelerar las operaciones de consultas sobre grandes colecciones de texto. Actualmente su aplicación más importante es sobre las búsquedas en la Web. Para estos casos, el servidor debe ser capaz de procesar eficientemente miles de consultas provenientes de los usuarios de Internet, por unidad de tiempo. La demanda creciente de este tipo de servicios ha llevado a considerar la realización paralela de las listas invertidas [9].

Asumiendo una colección de textos compuesta de un gran conjunto de documentos, una lista invertida es básicamente una tabla (el vocabulario) que mantiene todas las palabras relevantes encontradas en el texto y una lista, llamada lista invertida, por cada una de esas palabras que registra todas las ocurrencias de la palabra en el texto (identificador del documento y otra información utilizada para construir las respuestas a las consultas de los usuarios). Un número de estrategias han sido propuestas recientemente. Entre ellas tenemos las denominadas listas invertidas locales y globales.

La construcción paralela de índices globales es como sigue. La colección total de documentos es utilizada para producir un único archivo de indexación invertido que es idéntico al secuencial. Luego los T términos que forman la tabla de términos global (vocabulario), son distribuidos uniformemente sobre los P procesadores junto con sus respectivas listas de identificadores. Por lo tanto, luego de mapear los términos a los distintos procesadores, cada uno contiene aproximadamente T/P términos.
Para realizar el procesamiento paralelo de las consultas provenientes de las máquinas de los usuarios, cada término de la consulta es ruteado a un procesador del servidor a través de la máquina *broker*. Para cada término w perteneciente a la consulta u, se recupera la lista invertida asociada al término en el procesador correspondiente, luego estas listas son enviadas al

procesador *ranker* definido para la consulta u y de esta manera obtener los mejores documentos a ser enviados en forma de resultado al usuario.

La indexación local es un caso muy simple en la generación de índices, debido a que cada procesador del servidor posee el conjunto de documentos a partir del cual va a construir su lista invertida. La construcción de índices locales es como sigue. La base de datos textual sobre la cual se obtienen los términos de importancia, se encuentra distribuida entre los procesadores del servidor. Cada máquina busca en dichos documentos los términos de interés, calculando la cantidad de veces que el término se repite y opcionalmente las posiciones en las que se encuentra, para obtener su ubicación (que es la información adicional necesaria para construir la lista invertida).

Una vez que todos los documentos han sido procesados y todos los términos han sido incorporados a la lista invertida, ésta es ordenada lexicográficamente para facilitar la posterior resolución de consultas. Cuando la construcción de las listas invertidas ha finalizado, cada máquina del servidor posee una de estas listas, que han sido generadas considerando únicamente los documentos locales. Por lo tanto, se puede observar que cada una de las máquinas contendrá los mismos T términos, pero la longitud de la lista de identificadores de documentos es aproximadamente $1/P$ del índice global, donde P es el número de máquinas.
El procesamiento paralelo de una consulta u, consiste en seleccionar una máquina del servidor que recibirá dicha consulta para luego realizar un *broadcast*, y así todos los procesadores obtendrán la misma consulta a resolver, continuando de la misma forma en que se hace utilizando la estrategia de indexación global [3].

Para realizar la construcción de los índices y el procesamiento de las consultas en forma paralela, nos hemos basado en el modelo de computación *Bulk-Synchronous Parallel - BSP*, propuesto en 1990 por Leslie Valiant [13]. La idea fundamental de BSP 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.

La arquitectura del sistema sobre la cual se ejecutaron los algoritmos implementados, consiste de una máquina denominada *broker*, que es la encargada de recibir las consultas provenientes de los usuarios y distribuirlas entre las máquinas que conforman el *servidor BSP*. El servidor es quien debe realizar el procesamiento de consultas, seleccionando mediante una operación de *ranking* sobre la base de datos textual, los mejores documentos a ser enviados como resultados a la máquina *broker*.

Análisis Teórico vs. Análisis Empírico

Para realizar un análisis empírico se han ejecutado distintos casos de pruebas, tanto sobre una base de datos textual uniforme, donde todos los términos de las consultas tienen la misma probabilidad de aparecer, y sobre una base de datos no uniforme, donde en los términos de las consultas aparecen las cuatro letras del idioma español más frecuentes (a, c, m, p). El primer caso de prueba, consiste en resolver consultas que arriban al servidor en forma superpuesta, es decir que antes de terminar de procesar una consulta llega la siguiente. El segundo caso,

resuelve una única consulta formada por un número variable de términos.

Se llevaron a cabo experimentos utilizando las estrategias de indexación global y local, sobre un cluster de 12 SMP duales, conectadas mediante una red *Fast Ethernet*, analizando si existe una coherencia entre el análisis teórico y los resultados reales obtenidos. Para ello se ha modelado analíticamente ambas estrategias de indexación para todos los casos de prueba chequeados por el análisis empírico [3]. Esto tiene una gran importancia, debido a que permite comprobar si el modelo *BSP* puede realmente aproximar teóricamente a priori los costos que tendrán los algoritmos que se desean implementar, y de esta forma el diseñador de los algoritmos podrá decidir si realmente conviene o no construirlos.

Influencia de la Cantidad de Procesadores y el Número de Consultas

Para la misma arquitectura de sistema y plataforma de ejecución, se ha analizado la existencia de una relación entre el número de procesadores y el número de consultas a procesar, es decir, si se produce una saturación del sistema al incrementar la cantidad de consultas que arriban al servidor con un número fijo de procesadores, y de ser así hasta donde conviene hacer crecer al servidor.

Conclusiones

Para los algoritmos de resolución de consultas presentados en este trabajo, utilizando tanto la estrategia de indexación local como la global, y basados en el modelo *BSP*, se ha podido observar que en sistemas con listas invertidas pequeñas, se visualiza una modesta mejora de performance de indexamiento global sobre la local. Pero para sistemas con listas invertidas de gran tamaño, que es donde se justifica el uso del paralelismo, el costo de ambas estrategias tiende a ser similar.
Las experiencias realizadas hasta el momento exhiben, en general y para la mayoría de los casos de prueba llevados a cabo, un alto grado de coincidencia entre los resultados teóricos y los experimentales, lo cual indica una predicción confiable del comportamiento de los algoritmos que implementan las estrategias de listas invertidas locales y listas invertidas globales.
Con respecto al análisis de la insidencia del número de procesadores y el número de consultas sobre la performance, se puede observar que no sólo es necesario incrementar la cantidad de procesadores que conforman el servidor para poder mantener un *speedup* alto en los distintos casos de prueba que se llevaron a cabo, sino que también es necesario contar con una red adecuada que permita el traslado de los paquetes de información en forma rápida y eficiente.

Referencias

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

[2] Gonzalez, J.A., Leon, C., Piccoli, M.F., Printista, M, Roda, J.L., Rodriguez, C., Sande, F. The Collective Computing Model. Journal of Computer Science and Technology, Special Issue: Concurrent, Parallel and Distributed Processing, PGNAUT, ISTEC. Argentina. Octubre 2000. http://journal.info.unlp.edu.ar/ journal/HOME.HTML

[3] G. V. Gil Costa, Procesamiento Paralelo de Queries sobre Base de Datos TExtuales. çUniversidad Nacional de San Luis. 2003

[4] Gonzalez, J.A., Leon, C., Piccoli, M.F.,Printista, M, Roda, J.L., Rodriguez, C., Sande, F. Collective Computing. V Congreso Argentino de Ciencias de la Computación en la Universidad del Centro. Tandil, Argentina. Octubre 1999.

[5] M. Goudreau and K. Lang and S. Rao and T. Tsantilas. The Green BSP Library. Universyty of Central Florida, 1995.

[6] M. Goudreau and J. Hill and K. Lang and B. Mc Coll and S. Rao. A Proposal for the BSP Worldwide Standar Library. http://www.bsp-worldwide.org/standar/stand2.html. 1996.

[7] M. Goudreau and K. Lang and S. Rao and T. Suel and T. Tsantilas. Towards Efficiency and Portability: Programming with the BSP mode. SPAA'96: 8th Annual ACM SPAA, Pp 1-12, 1996.

[8] C. Leopold. Parallel and Distributed Computing. John Wiley and Sons. 2001.

[9] M. Marín. Parallel text query processing using Composite Inverted Lists. In Second Intenational Conference on Hybrid Intelligent Systems (Web Computing Session). IO Press, Feb. 2003.

[10] 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.

[11] D.B. Skillcorn and J. Hill and W.F. McColl. Questions and Answers about BSP. Oxford University Computing Laboratory. PRG-TR-15-96. 1996.

[12] Snir, M., Otto, S., Huss-Lederman, S., Walker, D., Dongarra, J. MPI: The Complete Reference. Cambridge, MA: MIT Press, 1996.

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