Mejoras Algorítmicas y Estructuras de Datos para Búsquedas Altamente Eficientes

Gabriel H. Tolosa[1,2], Esteban Feuerstein[2]

tolosoft@unlu.edu.ar; efeuerst@dc.uba.ar

[1] Departamento de Ciencias Básicas
Universidad Nacional de Luján
Cruce rutas 5 y 7, Luján.

[2] Departamento de Computación, FCEyN
Universidad de Buenos Aires
Pabellón I, Ciudad Universitaria, Buenos Aires.

Resumen

El problema de la búsqueda en Internet presenta desafíos constantes. Los datos son cada vez más ricos y complejos, se utilizan y varían en tiempo real, aportando nuevo valor, pero solamente si están disponibles en tiempo y forma. Los usuarios utilizan cada vez más motores de búsqueda, esperando satisfacer sus necesidades de información, navegación o para hacer transacciones, requiriendo que respondan miles de consultas por segundo.

Para poder manejar eficientemente el tamaño de una colección de documentos recolectados desde la web, los motores de búsqueda utilizan estructuras de datos distribuidas para hacer eficiente la búsqueda y técnicas de *caching* para optimizar los tiempos de respuesta. En este proyecto se propone diseñar y evaluar estructuras de datos avanzadas junto con nuevas técnicas algorítmicas que permitan mejorar la performance en las búsquedas para colecciones de datos de escala web.

Palabras clave: motores de búsqueda, estructuras de datas, técnicas de caching.

Contexto

Esta presentación se encuentra enmarcada en el proyecto de tesis de doctorado del primer autor en el Departamento de Computación de la FCEyN (UBA). El mismo se desarrolla en el grupo de investigación del Dr. Esteban Feuerstein (director) denominado "*Algoritmos y Herramientas Computacionales para Motores de Búsqueda y Publicidad Online en Internet*" en el mencionado Departamento.

Introducción

Internet es la más grande, diversa y compleja plataforma de comunicaciones existente en el mundo[1]. Millones de usuarios acceden diariamente a la red con propósitos variados: comunicarse, informarse, entretenerse y realizar transacciones, entre otras. En particular, la Web se ha convertido en el repositorio de información más grande que existe para la humanidad y sobre el cual se soportan decenas de tipos de servicios distribuidos de naturaleza diversa [Berners-Lee, 2000] [Wu, 2002] [Escudeiro, 2008]. Algunas de las características principales de la web son su heterogeneidad, tamaño y dinamismo [Baeza, 2003], incluyendo una amplia variedad de sitios web como portales empresariales, redes sociales, webmail, servicios multimedia y blogs, entre otros.

Sobre este escenario, encontrar información relevante a una necesidad concreta es un desafío. La "búsqueda" se convirtió en un proceso central. Los datos son cada vez mas ricos y complejos, se utilizan y varían en tiempo real, aportando nuevo valor, pero solamente si están disponibles en tiempo y forma [Hall, 2009]. Para navegar por este cúmulo de datos, los usuarios utilizan cada vez más motores de búsqueda, esperando satisfacer sus necesidades de información, de navegación (conocer la dirección de un sitio) o para hacer una transacción. Los motores de búsqueda como Yahoo! o Google responden millones de consultas (*queries*) por día y deben ser eficientes, para poder devolver las respuestas a los usuarios en unos pocos segundos, y efectivos para que las

1 http://www.internetworldstats.com/

mismas sean relevantes. Para lograr estos objetivos se requiere implementar técnicas sofisticadas que involucran diferentes aspectos del problema como la recolección de páginas (*crawling*), la indexación masiva, la recuperación eficiente y el ranking, principalmente.

Para poder manejar eficientemente el tamaño de una colección de documentos recolectados desde la web, los motores de búsqueda son normalmente implementados en un *cluster* de computadoras conectadas a una red local de alta velocidad, que habilita el procesamiento en paralelo. Esta arquitectura permite la distribución de la carga de trabajo para el procesamiento de tal colección entre muchos nodos y utiliza estructuras de datos distribuidas para hacer eficiente la búsqueda. Existen dos enfoques clásicos para las distribución de la colección entre los nodos: particionado por documentos y por términos [Badue, 2001], como así también esquemas híbridos sobre los que se ha trabajado en los últimos años, denominado índice 2D [Feuerstein, 2009].

La estructura de datos utilizada comúnmente en cada nodo es el índice invertido [Zobel, 2006]. Las consultas de los usuarios son procesadas sobre este índice invertido aplicando un modelo de recuperación de información que permite calcular un *score* para cada documento, lo que determina su relevancia. Adicionalmente, se calculan datos accesorios como un *snippet*[2] para armar la página de resultados que se presenta al usuario.

Como esta tarea se realiza en varias máquinas en paralelo se requiere un nivel de coordinación para la distribución de la consulta, la ejecución de la búsqueda y la fusión (*merge*) de los resultados. En toda la tarea existen costos involucrados de determinan el tiempo total de procesamiento de la consulta, tanto en comunicación (a través de la red) como de acceso a discos (para recuperar porciones del índice invertido) y procesamiento (cómputos de scores, sincronización, ranking y fusión de resultados).

2 Un *snippet* es un breve resumen del contenido del documento, usualmente de 2 o 3 líneas, que contiene el contexto donde ocurren los términos de la consulta.

Dado que uno de los desafíos para los motores de búsqueda es realizar el procesamiento de la consulta y responder al usuario en un muy breve período de tiempo es necesario implementar estructuras de datos y algoritmos eficientes que permitan mantener la performance (*throughput*), manejar picos de demanda y poder ser escalables. Para alcanzar estos objetivos se utilizan principalmente técnicas de caching y algoritmos de distribución y reducción del índice invertido.

Las técnicas de caching se utilizan en múltiples problemas de computación, desde el microprocesador [Genua, 2004] y el sistema operativo [Cheriton, 1995] hasta aplicaciones como los sistemas de base de datos [Effelsberg, 1984]. El principio fundamental del caching es almacenar en una memoria de rápido acceso (memoria principal) aquellos ítems que van a volver a aparecer en un futuro cercano. En el ámbito de los motores de búsqueda está demostrado que existe una marcada localidad temporal en la aparición de las consultas [Xie, 2002], es decir, un número significativo de queries van a volver a aparecer (enviados por el mismo u otro usuario) en un futuro próximo [Kumar, 2009]. Esta característica permite que el uso de una memoria caché para los resultados de búsqueda sea útil para reducir la carga de trabajo ya que se puede responder al usuario sin necesidad de procesamiento en los nodos.

Sin embargo, en un motor de búsqueda existen otros procesos cuya eficiencia se mejora mediante el uso de memorias caché. Estos son los casos del acceso a los documentos de la colección, a las listas del posting del índice invertido o el cómputo de intersecciones para consultas conjuntivas. En algunos casos, la combinación de diferentes técnicas de caching, de reducción del índice invertido y algoritmos de distribución eficientes permiten alcanzar una mejor performance que su uso de forma aislada.

Líneas de investigación y desarrollo

Los motores de búsqueda de escala web responden miles de *queries* en pequeñas fracciones de tiempo (típicamente milisegundos) a también miles de usuarios distribuidos por todo

el mundo. De forma normal, están formados por un conjunto de nodos de búsqueda entre los cuales se divide la colección de documentos a indexar. Además, existe un nodo *broker* que se encarga de gestionar las consultas y las respuestas (Figura 1).

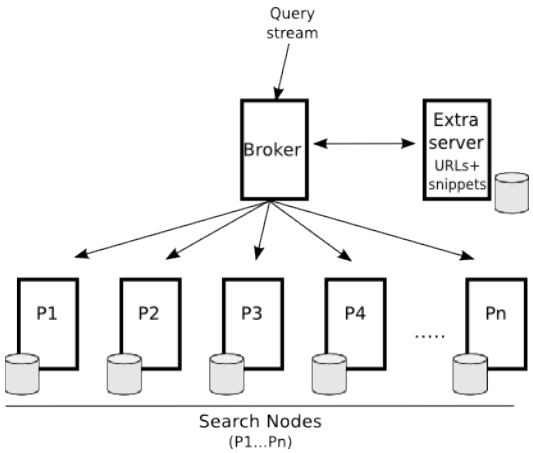

Figura 1 – Arquitectura de un motor de búsqueda web

2.1 Distribución del índice

La estructura de datos comúnmente utilizada para la indexación es el índice invertido. De forma simple, está compuesta por un vocabulario (V) con todos los términos extraídos de los documentos y – por cada uno de éstos – un lista de los documentos donde aparece dicho término junto con información de frecuencia (*posting list*). Para la distribución de la información entre los nodos de búsqueda existen dos enfoques clásicos [Badue, 2001]:

*** Particionado por documentos (índice local)**: El conjunto de documentos (C) es dividido entre los P procesadores del sistema, los cuales almacenan una porción del índice C/P. En esta estrategia todos los nodos participan de la resolución de cada consulta.

*** Particionado por términos (índice global):** Cada nodo mantiene información de las listas de posting completas de solamente un subconjunto de los términos. De la forma más trivial, el vocabulario V es dividido entre los P nodos y a cada uno de éstos se le asignan V/P listas. Para la resolución de la consulta sólo participan aquellos nodos que poseen la información de los términos involucrados.

Recientemente, se han propuesto esquemas híbridos como en índice 2D [Feuerstein, 2009] y el 3D. El primero consiste en organizar el conjunto de P procesadores en un array bidimensional (C columnas x R filas) en el cual se aplica el particionado por documentos en cada columna y el particionado por términos a nivel de filas (Figura 2).

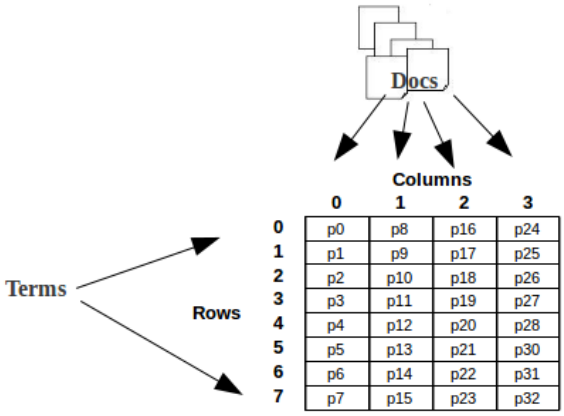

Figura 2 – Índice 2D

Los resultados de esta estrategia muestran que se pueden obtener mejoras si se selecciona adecuadamente el número de filas y columnas del array. Esto se debe a que existe un *trade-off* entre los costos de comunicación y procesamiento incurridos.

En el caso del Indice 3D (Figura 3), se agrega una dimensión (D) de procesadores que trabajan como réplicas. Si bien aún se está estudiando su comportamiento, los resultados preliminares sugieren que existe un comportamiento similar al esquema bidimensional, incluyendo los tres parámetros.

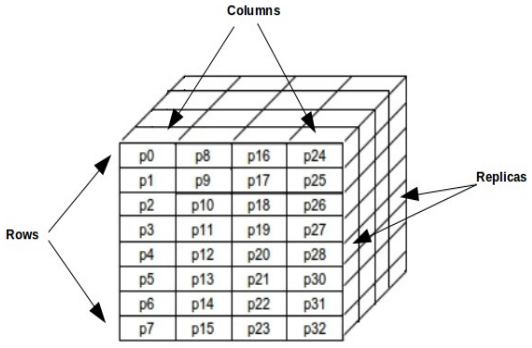

Figura 3 – Índice 3D

2.2 Técnicas de *Caching* en Motores de Búsqueda

Las técnicas de caching se basan en la idea fundamental de almacenar en una memoria de rápido acceso los ítems (objetos) que van a volver a aparecer en un futuro cercano, de

manera de poder obtenerlos desde ésta sin incurrir en costos (procesamiento, acceso a disco, entre otros). Estas técnicas se aplican en múltiples dominios tales como bases de datos, microprocesador, proxies [Cao, 1997] y sistemas operativos. La idea es explotar la localidad temporal que existe entre pedidos sucesivos de un mismo elemento tratando de mantener en memoria aquellos ítems con mayor chance de que ocurran nuevamente.

En motores de búsqueda, habitualmente se implementan cachés para resultados de búsqueda [Markatos, 2001] [Fagni, 2006], [Ozcan, 2008] listas de posting [Saraiva, 2001] [Zhang, 2008], intersecciones [Long, 2005] y documentos [Strohman, 2007]. Si bien se han propuesto diversos enfoques para cada caso, ninguno está completamente resuelto y existen oportunidades de optimización [Marín, 2010].

Por otro lado, la optimización de las cachés de listas de posting e intersecciones se encuentran relacionadas con el esquema utilizado en la distribución de los documentos. Si bien se han realizado aportes sobre los esquemas clásicos (Índice local y global), no existe suficiente investigación en las propuestas híbridas como el índice 2D y 3D.

Resultados y Objetivos

El objetivo de este trabajo es analizar los problemas mencionados relacionados con técnicas de optimización para aplicaciones de búsqueda y proponer mejoras arquitecturales que permitan mejorar la eficiencia de un sistema. Se propone profundizar sobre el estado del arte y definir, analizar y evaluar nuevos enfoques. Además, se analizará bajo qué situaciones las mejoras propuestas pueden ser más eficientes y se definirán arquitecturas que implementen las mejoras para poder evaluar su desempeño. En particular se estudiarán algunas de las siguientes líneas principales:

a) Estructuras de datos distribuidas, en especial aquellas propuestas recientemente a los efectos de evaluar posibles mejoras como las anteriormente descriptas.

b) Técnicas de *caching* de resultados, enfocando el problema no solamente en las políticas de reemplazo, sino también en políticas de admisión, tema que no ha tenido suficiente desarrollo aún. Aquí se propone un enfoque mediante el uso de técnicas de *Web Mining* para establecer y aprovechar propiedades de las consultas que ayuden a definir su “chance“ de mantenerse en memoria caché.

c) Técnicas de *caching* de intersecciones, en particular para consultas conjuntivas el caching de intersecciones presenta aún oportunidades de optimización. Se propone un enfoque de c*aching* basado en múltiples combinaciones de términos para intersecciones, considerando distintos fragmentos de información simples que se pueden almacenar en caché y su relación con consultas más complejas.

Los posibles dominios de aplicación de todas estas propuestas incluyen:

*** Motores de búsqueda de propósito general para la web**: es el escenario más directo para el cual se desarrollan algunas de estas técnicas.

*** Buscadores verticales**: Estas aplicaciones son la especialización de un motor de búsqueda de propósito general en un área o dominio específico.

*** Redes Sociales**: Su crecimiento (en usuarios e información) propone nuevos desafíos sobre las aplicaciones de búsqueda. Las relaciones establecidas entre los actores de la red se encuentran enriquecidas con metadatos que permiten obtener mejoras en los procesos de búsqueda.

*** Búsquedas móviles**: Los dispositivos móviles actuales permiten el manejo de múltiples canales de información en diversos formatos de archivos. Brindan acceso a grandes repositorios pero por sus limitados recursos requieren que las aplicaciones sean más eficientes.

Referencias

[Badue, 2001] C. Badue, R. Baeza-yates, B. Ribeiro-Neto, N. Ziviani. Distributed query

processing using partitioned inverted files. SPIRE Proc. of the 9th String Processing and Information Retrieval Symposium. 2001.

[Baeza, 2003] R. Baeza-Yates. Information Retrieval in the Web: beyond current search engines. International Journal on Approximated Reasoning 34 (2-3), 2003.

[Berners-Lee, 2000] T. Berners-Lee, M. Fischetti, M.L. Dertouzos. Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web. HarperCollins Pub. 2000.

[Cao, 1997] P. Cao, S. Irani. Cost-Aware WWW Proxy Caching Algorithms. Proceedings of the 1997 Usenix Symposium on Internet Technology and Systems. 1997.

[Cheriton, 1995] D. Cheriton, K. Duda. A caching model of operating system kernel functionality. Newsletter ACM SIGOPS, Operating Systems Review, Vol. 29, Nro.1, 1995.

[Escudeiro, 2008] N. F. Escudeiro, A. M Jorge. Satisfying Information Needs on the Web: a Survey of Web Information Retrieval. Polytechnical Studies Review, Vol 6, No. 2008.

[Fagni, 2006] T. Fagni, R. Perego, F. Silvestri, S. Orlando. Boosting the performance of web search engines: Caching and prefetching query results by exploiting historical usage data. ACM Transactions on Information Systems, 2006.

[Feuerstein, 2009] E. Feuerstein, M. Marín, M. Mizrahi, V. Gil Costa y R. A. Baeza-Yates. Two-dimensional distributed inverted files. In SPIRE 2009, LNCS 5721.

[Genua. 2004] P. Genua. A Cache Primer. Freescale Seminconductor. AN2663 . Rev. 1, 2004.

[Hall, 2009] W. Hall, D. De Roure, N. Shadbolt. The evolution of the Web and implications for eResearch. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, Vol. 367, No. 1890, 2009.

[Kumar, 2009] R. Kumar, A. Tomkins. A Characterization of Online Search Behavior. IEEE Data Engineering B, Vol.32, No 2, 2009

[Long, 2005] X. Long, T. Suel. Three-level caching for efficient query processing in large web search engines. In Proc. of the 14th International Conf. on World Wide Web, 2005.

[Marín, 2010] M. Marin, V. Gil-Costa, and C. Gomez-Pantoja. New caching techniques for web search engines. ACM HPDC, 2010.

[Markatos, 2001] E. P. Markatos. On caching search engine query results. Computing Communications, 24(2), 137–143. 2001.

[Ozcan, 2008] R. Ozcan, I. Altingovde, Ö. Ulusoy. Static query result caching revisited. In Proceeding of the 17th international conference on world wide web (pp. 1169–1170). 2008,

[Saraiva, 2001] P. C. Saraiva, E. S. de Moura, N. Ziviani, W. Meira, R. Fonseca, B. Ribeiro-Neto. Rank-preserving two-level caching for scalable search engines. In Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval (pp. 51–58). 2001.

[Strohman, 2007] T. Strohman, W. B. Croft. Efficient document retrieval in main memory. In Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2007.

[Wu, 2002] W. Hu. World Wide Web Search Technologies, Architectural Issues of Web-Enables Electronic Business, Idea Group Publishing . 2002.

[Xie, 2002] Y. Xie, D. O'Hallaron. Locality in Search Engine Queries and Its Implications for Caching. In IEEE Infocom 2002.

[Zhang, 2008] J. Zhang, X. Long, T. Suel. Performance of compressed inverted list caching in search engines. In Proceedings of the 17th international world wide web conference. 2008.

[Zobel, 2006] J. Zobel, A. Moffat. Inverted files for text search engines. ACM Computing Surveys, 38(2):Article 6, 2006.