Motores de Búsqueda Web Paralelas y Multimediales

Gil-Costa V., Printista M. [*]

LIDIC, Dpto. de Informática
UNSL, Argentina
e-mail: {*gvcosta, mprinti*}*@unsl.edu.ar*

Marín M.

Yahoo! Reseach,
Universidad de Santiago, Chile
e-mail: *mmarin@yahoo.com*

Resumen

Las máquinas de búsqueda para la Web son motores que requieren de un gran poder computacional y poseen de grandes bases de datos que deben ser indexadas eficientemente para lograr de esta manera reducir los tiempos de respuestas para las consultas ingresadas. A través de la computación paralela es posible encontrar nuevos algoritmos que permiten reducir los tiempos de respuestas logrando balancear tanto el cómputo realizado en cada procesador como la comunicación requerida.
En principio, nuestra investigación estuvo concentrada en búsquedas sobre base de datos de texto. Este reporte discute y referencia alguna de las principales conclusiones obtenidas en esta dirección.
Actualmente el énfasis está en resolver búsquedas de objetos multimediales sobre un servidor Web. Sobre el estudio y análisis de varias estructuras aptas para búsquedas en este tipo de dominios, en este trabajo se discute la implementación paralela de una de ella, el *Spatial Approximation Tree (SAT)*.
Finalmente, este reporte introduce la perspectiva de nuestros siguientes pasos, los cuales básicamente consisten en realizar nuevas implementaciones paralelas de otras estructuras de indexación de objetos multimediales, ya sea que permitan búsquedas en espacios métricos o búsquedas espacio-temporales.
Toda la investigación desarrollada en esta línea de investigación esta basada en el modelo de computación paralela $Bulk - SynchonousParallel$, BSP, el cual siendo un modelo sincrónico, es competitivo en performance cuando es comparado con librerías de pasaje de mensajes asíncronas.
Keywords: computación paralela, búsqueda en espacios métricos, búsqueda en bases de datos espacio-temporales, búsqueda de texto.

1. Introducción

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 [20], 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 indentificadores 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 [1, 4, 10, 6, 8, 12, 13, 11, 22]. 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

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

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 [12, 21]. 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.

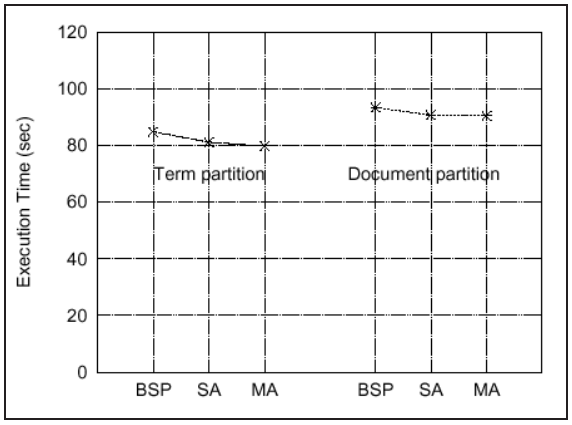

Figura 1: Resultados con 8 procesadores. El eje x indica las realizaciones de los algoritmos en BSP y dos realizaciones en MPI de las listas invertidas. MA=completamente asíncrono con MPI, SA=semi-asíncrono con MPI y BSP=sincrónico con BSP.

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 $threads$ son fuentes potenciales de $overheads$ y pueden producir salidas impredecibles en términos de tiempo de ejecución.

Alternativamente, toda nuestra investigación en el área esta basada en una forma de computación paralela más conservativa pero igualmente efectiva, el Modelo Bulk-Synchonous Parallel, BSP [19, 18]. 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.

BSP es un modelo libre de $deadlocks$ y tiene una manera particular de organizar sincrónicamente el cómputo en superpasos, y la performance obtenida es muy similar a la obtenida con algoritmos completamente asíncronos. La Figura 1 muestra un análisis comparativo de performance entre una implementación realizada bajo esta metodología y dos realizaciones de listas invertidas semi-asíncronas y completamente asíncronas. Para hacer los costos de comunicación similares, utilizamos la librería de comunicación $BSPonMPI$, que es una realización creciente del modelo BSP sobre las primitivas de MPI. Estos resultados muestran que BSP obtiene una performance competitiva.

2. Búsqueda de Texto

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. Esta 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 [16].

En este trabajo se han implementado y comparado distintas estrategias que permiten realizar el procesamiento de consultas. Entre ellas están las mencionadas anteriormente: partición de documentos y partición de términos, a las que denominaremos con las letras D y T respectivamente. En el caso de T, la máquina ranker es seleccionada teniendo en cuenta la cantidad de trabajo

planificada hasta el momento. Por lo tanto quisimos comprobar qué sucedía si la máquina ranker era seleccionada aleatoreamente (TR).

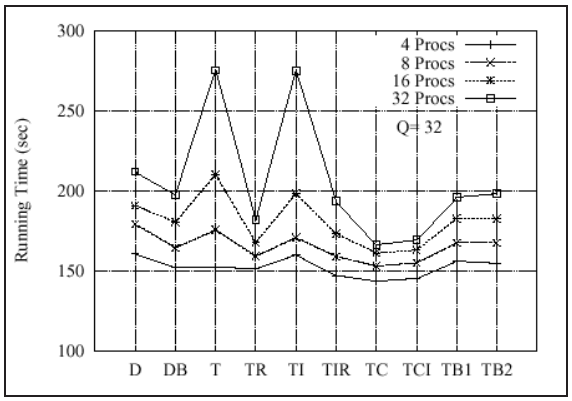

Figura 2: Procesamiento de 10,000 consultas insertando 256 nuevas consultas en cada superpaso.

Para el caso de la intersección, es decir consultas de tipo "and" determinamos la máquina en la que se almacena cada término y luego para cada log de consultas contamos la frecuencia en que los pares de términos (t_i, t_j) aparecen (TRI). A patir de alli, se arma una lista ordenada por frecuencia, luego se eliminan los pares del comienzo de la lista y se ubican en un procesador siguiendo la siguiente regla. Si ningún término ha sido asignado a un procesador, entonces colóquelos en el procesador con menos cantidad de términos. Si uno de los términos ha sido asignado, entonces su compañero va al mismo procesador.

También diseñamos algoritmos de indexación que trabajan con buckets (TB). Es decir que las listas son divididas en bloques para luego distribuirlas uniformemente entre los procesadores. Para la estrategia de particionado de documento, también usamos una estrategia similar (DB), donde en este caso el número de los buckets es igual al número de procesadores en el servidor. El objetivo principal de los buckets, es poder balancear tanto el cómputo realizado en cada procesador, como también la comunicación requerida para resolver una consulta, lo cual permite sacar una gran ventaja al utilizar el modelo BSP porque las comunicaciones se realizan en masa ($bulk$).

Por último presentamos TC y TCI que son versiones de TR y TRI respectivamente en los cuales los rankers mantienen en sus memorias cache los trozos de listas de los términos que aparecen en las consultas. En la Figura 2 se muestran los resultados obtenidos para lote de 10,000 consultas con las diferentes estrategias. Esta figura muestra los resultados obtenidos para un índice invertido que cabe completamente en memoria RAM. El estudio analítico de todas las estrategias desarrolladas en esta dirección pueden encontrarse en [9].

3. Búsqueda en Espacios Métricos

Con el crecimiento de información no textual en la Web, cada día es más importante almacenar, indexar y buscar imágenes, sonidos, audio y colecciones de video. Existe una gran variedad de estudios realizados sobre estructuras de datos multimediales, algunos de ellos son BK-Tree [3], GNAT [5], MTree [2], etc. Estas estructuras son utilizadas para realizar búsquedas por similitud en espacios métricos. Un espacio métrico está formado por una colección de U objetos y una función de distancia d definida entre ellos, la cual satisface la diferencia triangular. El objetivo es, dado un conjunto de objetos y una consulta recuperar aquellos objetos que se encuentran suficientemente cerca de la consulta.

Una estructura propuesta recientemente para este tipo de problemas es el *Spatial Approximation Tree (SAT)* que permite realizar búsquedas eficientes en espacios de alta dimensionalidad [14, 15]. El trabajo [17] examina el desempeño de SAT y muestra un análisis comparativo de performance de esta estructura con otras del mismo estilo.

En trabajos previos hemos paralelizado exitosamente esta estructura, logrando reducir significantemente el número de distancias calculadas en cada búsqueda, debido a que ésta es la medida de performance que se desea optimizar [7]. Hemos propuesto básicamente tres estrategias de paralelización. La estrategia local en la que los datos se distribuyen en el servidor y luego cada máquina construye su estructura localmente. La estrategia multiplexado, donde se construye una estructura única y luego los nodos son distribuidos entre los procesadores en forma multiplexada. Otra estrategia denominada $LOAD$ donde los nodos de las estructura de indexación son distribuidos teniendo en cuenta la cantidad de nodos que cada procesador posee, permitiendo balancear la carga. Finalmente, hemos optimizado los algoritmos de búsqueda colocando un límite V

al número de cómputo realizado en cada superpaso. Este límite es autoadaptativo, ya que cada cierta cantidad de superpasos, se recolecta información que permite evaluar la carga de trabajo de cada ç procesador y luego se actualiza el límite de cómputo V.

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 sencillo 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 y semi-asíncronas.

También hemos comenzado a estudiar y analizar posibles formas paralelas de indexación sobre espacios métricos. Esto permite realizar búsquedas multimediales en motores de búsqueda Web que hasta el momento no han sido presentadas en otros trabajos. Para ello se debe estudiar, analizar e implementar una gran variedad de estructuras de indexación que sean dinámicas, ya que es deseable incorporar nueva información en estos índices en forma periódica.

Nuestro siguiente paso consiste en realizar nuevas implementaciones paralelas y eficientes de otras estructuras de indexación de objetos multimediales, para luego poder analizar exhaustivamente su performance sobre un cluster de computadoras.

También es interesante agregar a esta investigación, la búsqueda espacio-temporales de objetos, que es un tema reciente pero de mucha utilidad para el control de tránsito vehicular, telefonía celular, etc. En este ámbito es posible realizar consultas históricas donde se recupera información de los objetos que se mueven (aparecen, desaparecen, cambian) generalmente en tiempo discreto. Este tipo de consultas se las conoce como $ventanas$ en el tiempo. Las consultas $timestamp$ recuperan los objetos que se encuentran en una ventana en un tiempo específico y las consultas de intervalo incluyen a su vez varios $timestamps$.

Referencias

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

[2] Sergei Brin. Near neighbor search in large metric spaces. In *The 21st VLDB Conference*, 1995.

[3] W. Burkhard and R. Keller. Some approaches to bestmatch file searching. In *Communication of ACM*, 1973.

[4] F. Cacheda, V. Plachouras, and I Ounis. Performance analysis of distributed architectures to index one terabyte of text. In *In S. McDonald and J. Tait, editors, Proc. ECIR European Conf. on IR Research*, pages 395–408, Sunderland, UK, April 2004.

[5] P. Ciaccia, M. Patella, and P. Zezula. An efficient access method for similarity search in metric spaces. In *The 23st International Conference on VLDB*, 1997.

[6] G.V. Gil Costa, M. Printista, and M. Marin. Improving web searches with distributed buckets structures. In *4th Latin American Web Congress*, pages 119–126, Puebla, Mexico, Oct. 2006. (IEEE-CS).

[7] V. Gil Costa, N. Reyes, A.M. Printista, and M. Marín. Multimedia web Searches using Static SAT. In *Congreso Argentino en Ciencias de la Computación (CACIC 2006) (1448-1459)*, Octubre 2006.

[8] V.G. Costa, M. Printista, and M. Marin. A parallel search engine with bsp. In *Third Latin American Web Congress (LaWeb 2005)*, pages 259–268, Buenos Aires, Argentina, Oct 2005. (IEEE-CS).

[9] G.V. Gil-Costa. *Estrategias de Buckets Paralelas para Máquinas de Búsqueda en la Web*. Tesis de Maestría, UNSL, Argentina, octubre, 2006.

[10] B. S. Jeong and E. Omiecinski. Inverted file partitioning schemes in multiple disk systems. *IEEE Transactions on Parallel and Distributed Systems*, 16(2):142–153, 1995.

[11] A. Moffat and J. Zobel. What does it mean to measure performance? *Proc. 5th Int. Conf. on Web Informations Systems,LNCS 3306,Springer*, pages 1–12, Brisbane, Australia,2004.

[12] 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.

[13] W. Moffat, J. Webber, Zobel, and R. Baeza-Yates. A pipelined architecture for distributed text query evaluation. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 348–355, September 2005.

[14] G.Ñavarro. Searching in metric spaces by spatial approximation. In *The Very Large Databases Journal (VLDBJ), (11(1):2846)*, 2002.

[15] G.Ñavarro and N. Reyes. Fully dynamic spatial approximation trees. In *In Proceedings of the 9th International Symposium on String Processing and Information Retrieval (SPIRE 2002),*, 2002.

[16] 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.

[17] S.Berchtold, C. Bohm, and D. Kein. Searching in highdimensional spaces: Index structures for improving the performance of multimedia databases. In *ACM Computing Surveys, (33(3):322373)*, 2001.

[18] 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.

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

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

[21] 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.

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