Elaboración de Estrategias Paralelas para Búsquedas por Similitud en Espacios Métricos

Norma Beatriz Perez, Mario Berón
Facultad de Ciencias Físico Matemáticas y Naturales - Universidad Nacional de San Luis
Ejército de los Andes, 950 CP: D5700HHW - San Luis - Argentina
email: {nbperez, mberon}@unsl.edu.ar

Fernando Magno Quintão Pereira
Departamento de Ciência da Computação - Universidade Federal de Minas Gerais
Av. Antônio Carlos, 6627 - Prédio do ICEx - Pampulha - CEP: 31270-010
Belo Horizonte Minas Gerais - Brasil
email: fernando@dcc.ufmg.br

Resumen

Realizar una búsqueda secuencial de elementos específicos sobre grandes volúmenes de información no es una aproximación apropiada. Esto se debe a que los tiempos de respuesta obtenidos no se adecuan a la exigencia de los usuarios. Una forma de abordar la problemática previamente mencionada consiste en aprovechar los avances de la tecnología, en lo que a software y hardware se refiere, para elaborar e implementar nuevos algoritmos que realicen búsquedas más eficientes, como lo son las búsquedas por similitud paralelas. Llevar a cabo la tarea antes mencionada no es fácil debido a que implica: i) El estudio de diferentes estructuras de datos que puedan ser paralelizables, ii) La selección de un modelo de computación paralela que posea una infraestructura que facilite la tarea del programador y iii) La elaboración de algoritmos que utilicen eficientemente las estructura de datos elegidas y que aprovechen al máximo las capacidades del modelo de computación paralela seleccionado y su infraestructura asociada. En este artículo se describe una línea de investigación que aborda las temáticas previamente mencionadas y cuyo principal objetivo es: La elaboración de algoritmos de búsqueda por similitud paralelos, basados en estructuras de indexación eficientes, cuya eficiencia escala en forma lineal respecto del número de procesadores disponibles en redes de tamaño moderado.

Palabras clave: Watershed, Filter-Stream, Modelos de Computación paralela, D-Index, Búsquedas por Similitud.

1. Contexto

La línea de investigación descrita en este trabajo se encuentra enmarcada en el contexto de dos proyectos de investigación. El primero, denominado *"Ingeniería del Software: Aspectos de Alta Complejidad Sensibilidad en el Ejercicio de la Profesión de Ingeniero de Software"*, se desarrolla en la Universidad Nacional de San Luis. Dicho proyecto, es reconocido por el programa de incentivos y es la continuación de diferentes proyectos de investigación de gran éxito a nivel nacional e internacional. El segundo, llamado *"Proyecto de co-tutela CAFP-BA 004/08"* se lleva a cabo en la Universidad Nacional de San Luis - Argentina y en la Universidade Federal de Minas Gerais, Belo Horizonte - Brasil. Este proyecto fue aprobado por la Secretaría de Políticas Universitarias (SPU) dependiente del Ministerio de Ciencia, Tecnología e Innovación Productiva de la Nación (MinCyT) [1]. Ambos entes soportan económicamente la realización de diferentes misiones de investigación desde Argentina a Brasil y viceversa.

2. Introducción

En los últimos años, la recuperación de la información sobre grandes conjuntos de datos tales como: registros de datos científicos, aplicaciones multimedia, bioinformática, etc., se han vuelto un problema de gran interés. Los tipos de datos que componen estas aplicaciones son, generalmente, complejos por lo que el proceso de manipulación de ellos no es una tarea simple. En muchos de esos casos las búsquedas exactas, que son la manera típica de buscar en bases de datos tradicionales, dejan de ser aplicables. Esto ha motivado el surgimiento de diversas

técnicas y métodos que permiten construir estructuras de indexación. Estas estructuras permiten recuperar la información de manera más eficiente y en menos tiempo. Esté tipo de métodos se ubican en una categoría denominada *búquedas por similitud* [2]. En esta categoría se ubican todos aquellos métodos que consisten en buscar aquellos objetos en un conjunto de datos que sean similares a un objeto de consulta dado.

Las búsquedas por similitud pueden ser descritas a través de la definición formal de los *Espacios Métricos* [3, 4]. Un Espacio Métrico esta compuesto por un par ordenado formado por un universo de objetos de un conjunto finito y de una función de distancia que satisface las propiedades de: *positividad*, *simetría* y *desigualdad triangular*. En donde, los objetos del universo serán comparados directamente utilizando la función de distancia, que indica el grado de *similitud* entre dos objetos.

Se ha comprobado que la implementación de algoritmos secuenciales de búsqueda por similitud no es apropiada debido a que los mismos no escalan bien para conjuntos de datos grandes como por ejemplo: registros de bases de datos médicas, conjuntos de datos disponibilizados en redes sociales *on-line* y servicios Web. Por esta razón, las investigaciones recientes se han centrado en tres aspectos fundamentales:

1. La búsqueda de estructuras de datos que sean eficientes e inherentemente paralelas.

2. La selección un modelo de computación paralela simple de entender y fácil de utilizar.

3. La elaboración de algoritmos paralelos de búsqueda por similitud eficientes.

Las tres temáticas antes mencionadas son el foco de estudio de la línea de investigación descrita en el presente trabajo.

El artículo está organizado como sigue. En la sección 3 se describe la línea de investigación. La sección 4 presenta los resultados obtenidos a partir del trabajo realizado por el equipo de investigación. Finalmente la sección 5 explica suscintamente los resultados obtenidos y esperados con respecto a la formación de recursos humanos.

3. Línea de investigación y desarrollo

La línea de investigación descripta en este artículo consta de tres temáticas principales las cuales se describen en las siguientes subsecciones.

3.1. Paralelización de Estructuras de Datos para Búsquedas por Similitud

En el estado del arte se pudo observar la existencia de un amplio conjunto de estructuras de datos [5] especializadas que permiten resolver búsquedas por similitud en los espacios métricos. Este conjunto se clasifica en dos grupos basados en:

1. Clustering o Particiones Compactas: consiste en particionar el conjunto de datos en áreas. Cada área tiene un centro, y se intentan descartar áreas completas sólo comparando la consulta con el centro del área. Son ejemplos de este grupo las siguientes estructuras: M-trees (MT) [6], Spatial Approximation Trees (SAT) [3].

2. Pivotes: almacenan las distancias que han sido precalculadas de cada objeto en el conjunto de datos a un conjunto de pivotes. Luego esas distancias se utilizan durante la búsqueda para descartar objetos del conjunto resultado. A este grupo pertenecen estructuras tales como: Approximating Eliminating Search Algorithm (AESA) [7], Fixed Queries trees (FQT) [8].

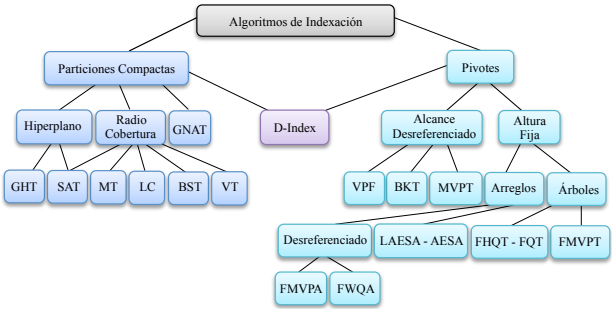

Figura 1: Taxonomía de los Algoritmos.

La Figura 1 ilustra una clasificación de las estructuras de datos para búsquedas por similitud en los espacios métricos existentes en la literatura de acuerdo a sus características generales.

El D-Index secuencial [9] es considerando uno de los métodos más rápidos de acceso disponible para resolver búquedas por similitud [10] debido a que posee las siguientes características:

- Representa uno de los avances más recientes en el escenario de los espacios métricos
- Une la familia de algoritmos basadas en clustering y pivotes [5].
- Se basa en: i) La definición de una función que divide a los objetos, y ii) El establecimiento de una jerarquía de buckets que almacenan estos objetos.

- Resulta adecuado en espacios de alta dimensionalidad, es decir, espacios en donde el problema de búsqueda por similitud es inherentemente difícil [10].

- Las características estructurales lo hacen adecuado para ser paralelizado.

- La performance depende de varios parámetros, y la configuración óptima de estos sigue siendo un problema abierto [11].

Por todas las características mencionadas previamente, el D-Index ha sido seleccionado, en primera instancia, como estructura soporte para los diferentes tipos de algoritmos que han sido propuestos por los integrantes del grupo de investigación.

3.2. Modelos de Computación Paralela

No sólo se requiere seleccionar la estructura de datos apropiada para la implementación de búsquedas por similitud sino que también es necesario determinar el modelo y la infraestructura de computación paralela adecuada a ser utilizada como soporte en la implementación de los algoritmos antes mencionados. Existe un gran número de modelos [12, 13] e infraestructuras [14, 15]. Estos han excedido la cantidad de arquitecturas diferentes y por lo general, la mayoría de ellos, son inadecuados o ineficientes cuando se desea paralelizar un índice.

Generalmente, los programadores al paralelizar sus índices deben tener conocimientos detallados de las componentes del modelo a utilizar. Esta particularidad es imprescindible para evitar, en los algoritmos paralelos, incrementar considerablemente el número de líneas de código y su complejidad. Por lo tanto, el uso y diseño de modelos e infraestrurcturas no es una tarea fácil e involucra un proceso altamente creativo.

Luego del estudios de varios modelos de computación paralela y sus infraestructuras asociadas se pudo observar que Watershed [16], una infraestructura asociada al modelo *Filter-Stream* [17], ofrece un marco de trabajo simple y unificado que posee las siguientes características: i) Permite el diseño y programación de aplicaciones *on-line* y *off-line* distribuidas, y ii) Posee un mayor poder cómputo al posibilitar resolver consultas en tiempos aceptables.

Una aplicación en Watershed está compuesta por una cadena de elementos (filtros) de procesamiento. Los filtros se comunican por medio de flujos continuos de datos. Cada filtro representa una etapa del procesamiento realizado por la aplicación. Un flujo de datos está compuesto por resultados intermediarios de alguna etapa de procesamiento que son las entradas para etapas posteriores. La Figura 2 ilustra el procesamiento que realiza Watershed a través de filtros que intervienen en la aplicación.

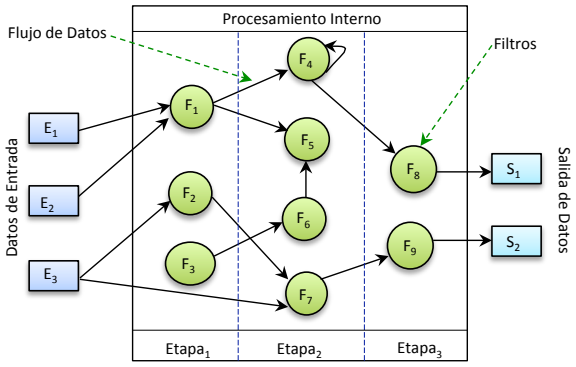

Figura 2: Vista del Programador de la Infraestructura Watershed para una Aplicación Arbitraria.

El interés de trabajar con Watershed como soporte para la implementación de los algoritmos paralelos se basa en las siguientes características:

- Permite que las aplicaciones puedan ejecutarse en modo flujo (donde no existe el concepto de finalización) o modo por lotes (donde los filtros finalizan tan pronto como sea procesado su último mensaje).

- Provee un mecanismo de adición y remoción de filtros en tiempo de ejecución.

- Permite la realización de alteraciones en la topología de las aplicaciones en tiempo de ejecución.

- Permite la reutilización de filtros en ejecución.

Como es de suponer, las características mencionadas en los ítems precedentes guiaron al grupo de investigación a seleccionar al modelo *Filter-Stream* y a la infrastructura Watershed como soporte para la implementación de los algoritmos propuestos.

3.3. Algoritmos de Búsqueda por Similitud Paralelos

Los algoritmos desarrollados por el equipo de investigación usan el Modelo de Computación Paralela *Filter-Stream*, la infrasetructura Watershed y están basado en la estrucutra de datos D-Index. Dichos algoritmos constan de dos partes claramente definidas. Cada una de ellas se describe a continuación:

Construcción del Índice: esta parte se relaciona con la construcción de la estructura de datos que almacena los índices para facilitar la búsqueda, en este caso dicha estructura es el D-Index. Esta tarea puede ser llevada a cabo usando tres aproximaciones, a saber:

- **Algoritmo Naive_build**: El conjunto de datos se replica en cada uno de los procesadores que forman parte del cluster y que intervienen en la búsqueda. Sobre cada procesador se construye un D-Index, obteniendo como resultado una replica del índice de todo el conjunto de datos en cada procesador.
- **Algoritmo Local_build**: El conjunto de datos es dividido de manera uniforme en la cantidad de procesadores que forman parte del cluster y que intervienen en la búsqueda. Sobre cada procesador se construye un D-Index utilizando la partición del conjunto de datos correspondiente. Es relevante destacar que cada procesador tiene asociado una partición diferente del conjunto de datos.
- **Algoritmo Global_build**: Se construye un único D-Index con el conjunto de datos completo. Luego el D-Index es dividido por niveles, donde cada nivel es asignado a un procesador diferente. Se emplea la política *round-robin* para la asignación de los niveles.

Búsqueda: esta parte se centra en la estrategia de búsqueda propiamente dicha. La misma puede ser llevada a cabo como sigue:

- **Algoritmo Naive_search**: Las consultas son enviadas a los procesadores de manera alternada, utilizando una política tipo *round-robin*. De esta manera las consultas se reparten de manera equitativa entre los procesadores. Cada procesador realiza el proceso de búsqueda sobre su D-index local de la manera usual.
- **Algoritmo Local_search**: Las consultas son enviadas a todos los procesadores, utilizando una política tipo *broadcast*. De esta manera la misma consulta es realizada simultáneamente en una partición diferente del conjunto de datos.
- **Algoritmo Global_search**: Las consultas son enviadas a todo los procesadores, utilizando una política tipo *broadcast*. De esta manera la misma consulta es enviada a cada procesador. Cada procesador realiza el proceso de búsqueda sobre el nivel del D-index asociado al procesador.

En los casos de los algoritmos de construcción del D-Index descritos, la infraestructura Watershed realiza este proceso a través de un filtro. Donde: i) El dato de entrada del filtro es un conjunto de datos arbitrario, ii) El Procesamiento interno realiza la construcción del D-Index de acuerdo al enfoque seleccionado y iii) Los datos de salida del filtro constan de un mensaje enviado al filtro receptor para indicar que puede continuar con la siguiente etapa de procesamiento de la aplicación.

En los casos de los algoritmos de búsqueda del D-Index descritos, la infraestructura Watershed realiza este proceso a través de un filtro. Donde: i) Los datos de entrada del filtro constan de: a) El D-Index construido de acuerdo a uno de los enfoques descritos anteriormente y b) Un conjunto de datos de consultas, ii) El procesamiento interno del filtro realiza la búsqueda en el D-Index de acuerdo al enfoque seleccionado y iii) Los datos de salida del filtro, es decir los resultados de la búsqueda, son enviados al filtro receptor. Dicho filtro se encarga de mostrar los resultados de acuerdo algún criterio establecido. Finalmente un mensaje es enviado para indicar que puede continuar con la siguiente etapa de procesamiento de la aplicación en caso de existir.

4. Resultados

En este proyecto, se ha logrado: (a) Analizar diferentes estructuras utilizadas para las búsquedas por similitud; (b) Seleccionar al D-Index como estructura de datos óptima para la realización de búsquedas por similitud; (c) Estudiar diferentes Modelos de Computación Paralela y sus infraestructuras asociadas; (d) Explorar el modelo Filter-Stream y la infraestructura Watershed; (e) Implementar algoritmos paralelos basados en D-Index sobre Watershed; (d) Implementar los mismos algoritmos paralelos bajo otro modelo de programación paralela para su posterior análisis y comparación con la versión implementada en Watershed; (d) Realizar experimentos utilizando dos conjuntos de datos para determinar su desempeño.

Durante este proyecto, se espera: (a) Continuar explorando exhaustivamente la infraestructura Waterhsed; (b) Obtener una mejora sustancial en el desempeño de los algoritmos basados en los enfoques paralelos propuestos; (c) Realizar estudios comparativos con otros modelos de computación

paralela existentes en la literatura; (d) Realizar experimentos que permitan explorar el amplio rango de aplicación de los algoritmos propuestos.

5. Formación de Recursos Humanos

Los trabajos elaborados en la presente línea de investigación forman parte del desarrollo de tesis para optar a los grados de Magister en Ingeniería del Software (UNSL) y Mestre em Ciência da Computação (UFMG). Es importante mencionar que tanto el equipo de la Argentina como el de Brasil se encuentran dedicados a la captura de alumnos de grado y posgrado para la realización de estudios de investigación relacionados con las temáticas presentadas en este trabajo. Dichos estudios pretenden fortalecer la relación entre UNSL y la UFMG.

Referencias

[1] "Ministerio de ciencia, tecnología e innovación productiva," 2007. [Online]. Available: http://www.mincyt.gov.ar/

[2] P. Zezula, G. Amato, V. Dohnal, and M. Batko, *Similarity Search: The Metric Space Approach*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[3] G. Navarro, "Searching in metric spaces by spatial approximation," in *SPIRE*. IEEE, 1999, pp. 141–148.

[4] E. Chávez, G. Navarro, R. A. Baeza-Yates, and J. L. Marroquín, "Searching in metric spaces," *ACM Comput. Surveys.*, vol. 33, no. 3, pp. 273–321, 2001.

[5] E. Chávez, J. L. Marroquín, and G. Navarro, "Fixed queries array: A fast and economical data structure for proximity searching," *Multimedia Tools Appl.*, vol. 14, no. 2, pp. 113–135, Jun. 2001.

[6] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," in *VLDB*. ACM, 1997, pp. 426–435.

[7] E. V. Ruiz, "An algorithm for finding nearest neighbours in (approximately) constant average time," *Pattern Rec.*, vol. 4, pp. 145–157, 1986.

[8] R. Baeza-Yates, W. Cunto, and S. Wu, "Proximity matching using fixed-queries trees." In Proc. 5th Combinatorial Pattern Matching (CPM'94), LNCS 807, 1994, pp. 198–212.

[9] V. Dohnal, C. Gennaro, P. Savino, and P. Zezula, "D-Index: Distance searching index for metric data sets," *Multimedia Tools and Applications*, vol. 21, pp. 9–33, 2003.

[10] A. J. Müller-Molina and T. Shinohara, "On the configuration of the similarity search data structure d-index for high dimensional objects," in *Proceedings of the 2010 international conference on Computational Science and Its Applications - Volume Part III*, ser. ICCSA'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 443–457.

[11] V. Dohnal, C. Gennaro, P. Savino, and P. Zezula, "D-index: Distance searching index for metric data sets," *Multimedia Tools Appl.*, vol. 21, no. 1, pp. 9–33, 2003.

[12] L. G. Valiant, "A bridging model for parallel computation," *Commun. ACM*, vol. 33, no. 8, pp. 103–111, 1990.

[13] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[14] M. Beynon, C. Chang, U. Catalyurek, T. Kurc, A. Sussman, H. Andrade, R. Ferreira, and J. Saltz, "Processing large-scale multi-dimensional data in parallel and distributed environments," *Parallel Comput*, vol. 28, no. 5, pp. 827–859, May 2002.

[15] D. Fireman, G. Teodoro, A. Cardoso, and R. Ferreira, "A reconfigurable run-time system for filter-stream applications," in *Proceedings of the 2008 20th International Symposium on Computer Architecture and High Performance Computing*, ser. SBAC-PAD '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 149–156.

[16] T. L. Alves de Souza Ramos, R. S. Oliveira, A. P. de Carvalho, R. A. C. Ferreira, and W. Meira Jr., "Watershed: A high performance distributed stream processing system," in *Proceedings of the 2011 23rd International Symposium on Computer Architecture and High Performance Computing*, ser. SBAC-PAD '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 191–198.

[17] D. A. Adams, "A computation model with data flow sequencing," Ph.D. dissertation, Stanford University, 1969.