Recuperación Eficiente de Datos en Bases de Datos Multimedia

Luis Britos, María E. Di Gennaro, Jacqueline Fernández, Veronica Gil-Costa, Fernando Kasián, Verónica Ludueña, Marcela Printista, Nora Reyes, Patricia Roggero
LIDIC, Dpto. de Informática, Fac. de Cs. Físico Matemáticas y Naturales, Universidad Nacional de San Luis
{*lebritos, mdigena, jmfer, gvcosta, fkasian, vlud, mprinti, nreyes,proggero*}@unsl.edu.ar
Edgar Chávez
Escuela de Cs. Físico–Matemáticas, Universidad Michoacana de San Nicolás de Hidalgo
elchavez@umich.mx
Claudia Deco
Facultad de Ciencias Exactas, Ingeniería y Agrimensura, Universidad Nacional de Rosario
deco@fceia.unr.edu.ar

Resumen

Los sistemas de información necesitan realizar de manera eficiente la búsqueda de los datos a través de estructuras de almacenamiento. Dicha búsqueda consiste en obtener los elementos que cumplen una serie de condiciones. En general, es tan difícil para los usuarios que intentan recuperar información multimedia poder especificar claramente sus intereses, o las condiciones a cumplir, a través de una consulta bien definida, como para los diseñadores del sistema decidir qué características de los objetos multimedia pueden ser relevantes. Dada una consulta, el objetivo de un sistema de recuperación de información es obtener lo que podría ser útil o relevante para el usuario, en general usando un índice especialmente diseñado para responder eficientemente la consulta.

Así, nuestra línea de investigación tiene como principal objetivo desarrollar herramientas eficientes para la recuperación de información multimedia. Se investigan nuevas técnicas que soporten la interacción con el usuario, nuevas estructuras de datos (índices) capaces de manipular eficientemente datos multimedia y que permitan manejar grandes volúmenes de este tipo de datos.

Palabras Claves: *Recuperación de Información, Bases de Datos Multimedia, Indexación, Paralelismo.*

Contexto

Esta línea de investigación se encuentra enmarcada dentro del Proyecto Consolidado 30310 de la Universidad Nacional de San Luis y en el Programa de Incentivos (código 22/F034): "Nuevas Tecnologías para el Tratamiento Integral de Datos Multimedia", dentro de la línea "Recuperación de Datos e Información Multimedia", desarrollada en el ámbito del Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC) de la UNSL.

En este marco se pretende avanzar en la integración de las investigaciones sobre adquisición, preprocesamiento y análisis de datos no estructurados y su aplicación en dominios no convencionales. Como principal aporte de esta propuesta se pretende incorporar información no estructurada en los procesos de toma de decisiones y resolución de problemas que quedan sin considerar en los enfoques clásicos.

Dentro de este contexto nuestra línea se dedica, principalmente, al diseño de índices eficientes que sirvan de apoyo a sistemas de recuperación de información orientados a datos no estructurados, en particular datos multimedia. Se espera así contribuir a estos sistemas obteniendo índices más eficientes para memorias jerárquicas, dinámicos, con I/O eficiente y capaces de manejar grandes volúmenes de datos. Se propone analizar las estructuras de datos existentes, proponer optimizaciones o diseñar nuevas estructuras, para manipular y recuperar algunos de los tipos de datos no estructurados que aparecen en entornos multimedia, considerando el uso de técnicas de computación de alto desempeño en los mismos, con el objetivo de lograr una recuperación eficiente.

1. Introducción y Motivación

Los sistemas de computación hacen uso intensivo de información estructurada, es decir datos elementales o estructuras, generadas con un formato específico. Una característica principal en estos casos, es que la estructura o formato de esta información puede ser fácilmente interpretada y directamente utilizada por un programa de computadora. Pero el hecho de restringirse al uso de este tipo de información conduce, muchas veces, a representar una visión parcial del problema y dejar fuera información

que podría ser importante para la resolución efectiva del mismo. En este contexto gran parte de la información que se requiere para la toma de decisiones y la resolución de problemas de índole general proviene de información no estructurada.

En general, para responder eficientemente consultas para recuperación de información sobre bases de datos multimedia se utilizan diferentes métodos de acceso o índices [9], principalmente por el volumen de datos con el que se trabaja. Algunos poseen características que los hacen indicados para aplicaciones reales: eficientes, dinámicos, escalables y resistentes a la *maldición de la dimensión*.

Un enfoque prometedor para sistemas de recuperación usando búsqueda por similitud es una búsqueda basada en contenidos, la cual usa el dato multimedia mismo. Para calcular la similitud entre dos objetos multimedia, se debe definir una función de distancia. Dicha función mide la similitud, o más bien la disimilitud, entre dos objetos. En muchos casos para modelar la similitud de objetos multimedia se transforman los objetos en puntos de un espacio vectorial, el cual es un tipo particular de espacio métrico. Cada objeto es representado por un vector de características o descriptor, generalmente de alta dimensionalidad. Sobre espacios vectoriales existen numerosas funciones de distancia. El tipo de aplicación, las características a explotar o la dimensionalidad son aspectos a considerar para definir la mejor función de distancia a utilizar.

El concepto de búsqueda por similitud se puede definir a partir del de espacios métricos, que da un marco formal independiente del dominio de aplicación. Un espacio métrico está compuesto por un *universo* $\mathcal{U}$ de objetos y una función de distancia $d : \mathcal{U} \times \mathcal{U} \longrightarrow R^+$, que satisface las propiedades que la hacen una métrica. Las consultas por similitud, sobre una *base de datos* $\mathcal{S} \subseteq \mathcal{U}$, son usualmente de dos tipos: *Búsqueda por rango* y *Búsqueda de los* k *vecinos más cercanos*.

En el caso de los espacios métricos, la función de similitud (distancia) mide el mínimo esfuerzo (costo) necesario para transformar un objeto en otro. Dependiendo de los tipos de datos multimedia reales la función de similitud puede ser muy compleja. En particular, para poder ahorrar cálculos de distancia es importante que la distancia satisfaga la propiedad triangular. Si la base de datos $\mathcal{S}$ posee n objetos, las consultas pueden ser respondidas llevando a cabo n evaluaciones de distancia. Sin embargo, en la mayoría de las aplicaciones las distancias son costosas de computar (por ej.: comparación de huellas digitales), por lo que la búsqueda secuencial no sirve para problemas de tamaño medio o grande, que son los tamaños más habituales de las bases de datos multimedia. Así debemos preprocesar la base de datos, construyendo un índice, para responder a las consultas con la menor cantidad de cálculos de distancia. Además, es probable que la base de datos, el índice o ambos no puedan almacenarse en memoria principal con lo cual se debe considerar minimizar también el número de operaciones de E/S, tener siempre presente la jerarquía de memorias y tratar de lograr mayor eficiencia a través de técnicas paralelas.

En suma, esta propuesta se enfoca en mejorar las herramientas de recuperación desarrollando nuevas técnicas que soporten la interacción con el usuario, diseñando estructuras de datos (índices), capaces de manipular eficientemente grandes volúmenes de datos multimedia y facilitando la realización de operaciones, de modo de acercarse al nivel de desarrollo que poseen las bases de datos tradicionales.

2. Líneas de Investigación

Se pretende investigar sobre distintos aspectos de los sistemas de recuperación de información multimedia: diseñar nuevos índices, definir representaciones que reflejen características de interés de los objetos y manejar distintas operaciones sobre estos tipos de bases de datos, considerando trabajar eficientemente sobre grandes volúmenes de datos.

Diseño de Índices

Un catálogo importante de índices para espacios métricos aparece en [9]. La mayoría usan la desigualdad triangular para evitar el análisis secuencial de la base de datos. La distancia entre la consulta q y los objetos de la base de datos puede ser estimada calculando de antemano algunas distancias a objetos distinguidos llamados *pivotes* y sin calcular las distancias reales desde q a los objetos de la base de datos durante una búsqueda. Otra técnica común es indexar a través de una partición del espacio en regiones denominadas *particiones compactas*.

Existen dos posibles situaciones por el tipo de base de datos con la que se va a trabajar, que determinan una característica importante que debe tener el índice que la manipulará: los objetos de la base de datos se conocen de antemano y por lo tanto el índice se creará de una sola vez y se realizarán consultas sobre él (índices estáticos); o no se conocen los objetos de antemano y por lo tanto el índice se debe ir

creando a medida que arriban los elementos y preferentemente de manera incremental (índices dinámicos). Las estructuras estáticas se benefician desde el conocimiento de la base de datos seleccionando los mejores puntos de referencia para una estructura de datos determinada, lo cual no es posible en las estructuras de datos dinámicas donde tanto los objetos como las consultas arriban al azar.

Cuando se trabaja sobre grandes volúmenes de datos una manera de lograr eficiencia en las operaciones sobre los índices es aplicando técnicas de computación de alto desempeño y en otros casos mediante la adaptación o diseño de las estructuras para que sean concientes de la jerarquía de memorias, minimizando no sólo la cantidad de cálculos de la función de distancia, sino también el número de operaciones de E/S. Además, el estudio detallado de qué hace que un índice sea intrínsecamente más eficiente ayuda a mejorar los costos de las operaciones.

Índices Estáticos

En este caso, al conocer de antemano los elementos a indexar, es posible elegir con más información cómo hacerlo de manera tal que las búsquedas sean eficientes. Sin embargo, hay ejemplos como el del *Árbol de Aproximación Espacial*, SAT, que por ser una estructura estática debería ser más eficiente que la versión dinámica, el DSAT [13], y no lo es. En estos casos ha sido posible investigar alguno de los motivos por los que la versión dinámica, usando menos información, proporciona búsquedas mas rápidas. En nuestras investigaciones hemos detectado que una condición clave para mejorar la performance de SAT es modificar la estrategia de selección de vecinos, es por ello que se ha estado trabajando en diferentes heurísticas, como la de utilizar un orden de inserción arbitrario de los vecinos o hasta elegirlos de manera totalmente contraria a lo que la versión original lo hacía y se están consiguiendo en este caso resultados muy interesantes [6]. El análisis detallado del comportamiento de esta estructura nos ha llevado a demostrar que el SAT se beneficia significativamente al mejorar la distribución de los vecinos de un nodo y por lo tanto los hiperplanos y radios de cobertura resultantes, tan importantes para podar nodos o subárboles durante las búsquedas, son mejores. El DiSAT, ha demostrado no sólo superar a la versión original de la estructura, sino también a los mejores representantes del estado del arte, como es la *Lista de Clusters* [8], y además ser escalable y ser resistente a la maldición de la dimensión [7].

Índices Dinámicos

Aquí el interés está en mejorar el desempeño de índices dinámicos jerárquicos (árboles), que es el caso de algunos de los índices para espacios métricos. Estos índices dinámicos, en general, se construyen incrementalmente vía inserciones. De tal manera, la raíz del árbol es el primer objeto que llega, y esto se repite recursivamente en cada nivel del árbol.

Como ya mencionamos, el DiSAT es un índice para espacios métricos muy eficiente, considerando el número de cálculos de distancias realizados tanto en construcción como en búsquedas. La desventaja del DiSAT es que no admite inserciones ni eliminaciones y no es posible construirlo incrementalmente. Sin embargo, una opción para transformarlo en una estructura dinámica es mediante la aplicación de la técnica de Bentley y Saxe [2] que permite lograr dinamismo a partir de una estructura estática cuando la búsqueda sobre ella cumple con ser un problema que se puede descomponer en partes independientes (descomponible). Por lo tanto, se está desarrollando una versión dinámica del DiSAT, que además pueda resolver eficientemente las búsquedas de los k-vecinos más cercanos de cualquier objeto, búsqueda que no cumple con ser un problema descomponible.

También en esta línea se ha propuesto una técnica donde el “buffering” logra un buen compromiso entre una estructura estática, construída con toda la información necesaria y una dinámica con conocimiento local de los datos. Entonces, en lugar de elegir al primer elemento como la raíz, se demora la selección hasta que hayan arribado suficientes elementos para estar en condiciones de realizar dicha selección, y de esta manera se toma una decisión en base a más información. Dado que las consultas arriban a un ritmo desconocido, para mantener el dinamismo es necesario contar con un índice que responda a las consultas con mejor desempeño que la técnica de fuerza bruta. Esta técnica provee un marco adecuado para diseñar estructuras de datos dinámicas estables. Por lo tanto, un “buffer” en todos los niveles de una estructura jerárquica podría ser útil en estrategias de ruteo para guiar las búsquedas, lo cual resulta un área promisoria de investigación [10].

En muchos casos los volúmenes de información con los que se debe trabajar (millones de imágenes en la Web), hacen necesario que los índices sean almacenados en memoria secundaria. En este caso, para hacerlos eficientes, no sólo se debe considerar que durante las búsquedas se realice el menor número de cálculos de distancia sino también, dado el costo

de las operaciones sobre disco, se efectúe la menor cantidad posible de operaciones de E/S. Por ello, en esta línea nos hemos dedicado a diseñar índices especialmente adaptados para trabajar en memoria secundaria, logrando un buen desempeño de los mismos, principalmente en las búsquedas.

Hemos diseñado e implementado las siguientes estructuras *DSACL*-tree* y el *DSACL+-tree* [4], las cuáles son optimizaciones para memoria secundaria de la estructura propuesta en [1] y demostraron ser competitivas frente a otras de las estructuras conocidas tales como el *M-tree* y *DSA*-tree* y *DSA+-tree* [13]. Además, existen nuevas propuestas en evaluación que prometen ser aún más adecuadas para memoria secundaria. Por otro lado, nos proponemos optimizarlas todavía más gracias a la aplicación de técnicas de computación de alto desempeño, aplicando y comparando distintas estrategias de paralelización con el fin de determinar la más adecuada.

Índices para Sistemas Paralelos

El desarrollo de nuevas tecnologías permite combinar las facilidades provistas por los sistemas de memoria compartida y los sistemas de memoria distribuida (cluster de nodos o procesadores). Es decir, permiten acelerar las operaciones de cómputo al contar con una mayor cantidad de cores (núcleos) y aumentar la capacidad de almacenamiento. Esta combinación de diferentes jerarquías o niveles de hardware, permite ejecutar P procesos en diferentes nodos conectados mediante una red de intercomunicación y dentro de cada proceso ejecutar PT threads (hilos) en paralelo, cada uno en un "core" diferente utilizando un esquema en el cual todos los threads tienen acceso a la misma memoria principal. El desafío en este tipo de arquitecturas es reducir el tiempo de ejecución de los programas, lo cual para algunas aplicaciones como las búsquedas de texto y búsqueda multimedia sobre espacios métricos puede ser aún más difícil de lograr, debido a las competencias por los recursos causadas por el acceso concurrente de los threads a los datos compartidos.

Para afrontar este desafío exitosamente, es necesario estudiar y diseñar diferentes técnicas de particionado de índices sobre el sistema de memoria distribuido y diseñar estrategias de asignación de trabajo a los diferentes threads del sistema de memoria compartida. Estas técnicas deben explotar las características inherentes de los índices para maximizar el cómputo paralelo en forma balanceada y reducir los costos de comunicación y sincronización.

En particular, el trabajo presentado en [12] evalúa diferentes métricas como la eficiencia, *throughput* y *speedup* de diferentes técnicas de particionado sobre un índice métrico denominado *Sparse Spatial Selection* (SSS) [3]. Por otro lado, en el trabajo presentado en [14] se utilizan herramientas de evaluación de performance para evaluar la eficiencia en el uso de jerarquías de memorias caché de un sistema multicore sobre el índice de Arreglos de Sufijos.

Consultas sobre Bases de Datos Multimedia

Aunque las operaciones más comunes sobre bases de datos multimedia son las búsquedas por rango o de k-vecinos más cercanos, existen otras operaciones de interés como las distintas variantes del *join* por similitud. Para estas operaciones se consideran dos bases de datos A y B, ambas subconjuntos del mismo universo del espacio métrico $\mathcal{U}$. El resultado de cualquier operación de join por similitud entre A y B obtiene el conjunto de pares formados por un objeto de A y otro de B, tales que entre ellos se satisface el predicado de similitud considerado. Las variantes más conocidas del join por similitud son: el join por rango, el join de k-vecinos más cercanos y el join de vecino más cercano; entre otras.

Formalmente, dadas $A, B \subseteq \mathcal{U}$, se define el *join por similitud* entre A y B ($A \underset{\Phi}{\bowtie} B$) como el conjunto de todos los pares (x, y), donde $x \in A$ e $y \in B$; es decir, $(x, y) \in A \times B$, tal que entre x e y se satisface el criterio de similitud considerado Φ. De acuerdo al criterio de similitud el join puede llamarse: Join por rango o Join de k-vecinos más cercanos.

Existen dos situaciones distintas sobre las que se puede trabajar, para resolver el join por similitud: que ambas bases de datos se encuentren indexadas, por separado; o que ambas bases de datos se indexen conjuntamente, con un índice diseñado para el join. Como calcular cualquiera de las variantes del join por similitud de manera exacta es muy costoso [15], vale la pena analizar posibilidades de obtener más rápidamente una respuesta aproximada al join, logrando una respuesta rápida y de buena calidad.

PostgreSQL es el primer sistema de base de datos que permite realizar consultas por similitud sobre algunos atributos, particularmente indexación para búsquedas de k-vecinos más cercanos (KNN-GiST indexes). Estos índices pueden ser usados sobre texto, comparación de ubicación geoespacial, etc.. Sin embargo, los índices K-NN GiST proveen plantillas para índices con estructura de árbol balanceado (B-tree, R-tree), aunque el balance no siempre es bueno para los índices que se utilizan en búsquedas por si-

militud [5]. Además, no se dispone de este tipo de consultas para todo tipo de datos métricos. Así, es importante proveer un DBMS para bases de datos métricas que maneje todos los posibles datos métricos y las operaciones de interés sobre ellos [11].

3. Resultados

Se ha comprobado experimentalmente que las estrategias de "buffering" mejoran el desempeño en un índice dinámico [10]. Se seleccionó el Árbol de Aproximación Espacial Dinámico (DSA-tree) [13] y se obtuvo una mejora sistemática en los costos de las consultas. En particular, se verificó que el DSAT es mejor que su versión estática [13], por dejar como "vecinos" a objetos alejados, permitiendo así avanzar en la exploración espacial a "pasos más grandes", obteniendo así el DiSAT [7].

Se implementaron dos versiones: DSACL*-tree y DSACL+-tree, que trabajan con grandes volúmenes de datos, por haber sido diseñadas para memoria secundaria y que mostraron ser competitivas contra otras estructuras diseñadas para tal fin [4]. Se espera lograr una implementación paralela eficiente de estos índices. Se ha logrado mayor eficiencia en otros índices [12, 14] gracias al uso de técnicas paralelas.

4. Formación de Recursos

Para contribuir al desarrollo de sistemas de recuperación de información multimedia, se están capacitando los siguientes investigadores:

Tesis de Doctorado en Cs. de la Computación: un integrante se encuentra definiendo su plan de doctorado sobre diseño y optimización de índices, para aplicaciones de minería de datos multimedia.

Tesis de Maestría en Cs. de la Computación: una sobre índices dinámicos para búsqueda eficiente en memoria principal, dos sobre índices dinámicos eficientes para datos masivos: una que aplica técnicas de computación de alto desempeño (con beca de posgrado-UNSL) y la otra sólo aplicando optimizaciones considerando la jerarquía de memorias, y una sobre un DBMS para bases de datos métricas.

Referencias

[1] M. Barroso, N. Reyes, and R. Paredes. Enlarging nodes to improve dynamic spatial approximation trees. In *Proc. of the 3rd SISAP*, pages 41–48. ACM Press, 2010.

[2] J. Bentley and J. Saxe. Decomposable searching problems i: Static-to-dynamic transformation. *J. Algorithms*, 1(4):301–358, 1980.

[3] N. R. Brisaboa, A. Fariña, O. Pedreira, and N. Reyes. Similarity search using sparse pivots for efficient multimedia information retrieval. In *Proc. of the ISM*, 881–888. IEEE, 2006.

[4] L. Britos, A. M. Printista, and N. Reyes. Dynamic spatial approximation trees with clusters for secondary memory. *XVI CACIC Selected Papers*, 205–215. Editorial UNLP, 2011.

[5] E. Chávez, V. Ludueña, and N. Reyes. Revisiting the VP-forest: Unbalance to improve the performance. In *Proc. de las JCC08*, 26, 2008.

[6] E. Chávez, V. Ludueña, N. Reyes, and P. Roggero. Reaching near neighbors with far and random proxies. In *CCE, 8th Int. Conf. on*, 1–8, oct. 2011.

[7] E. Chávez, V. Ludueña, N. Reyes, and P. Roggero. Faster proximity searching with the distal sat. Manuscrito IEEE TPAMI-2012-12-0969. 2012.

[8] E. Chávez and G. Navarro. A compact space decomposition for effective metric indexing. *Pattern Recognition Letters*, 26(9):1363–1376, 2005.

[9] E. Chávez, G. Navarro, R. Baeza-Yates, and J. Marroquín. Searching in metric spaces. *ACM*, 33(3):273–321, sep 2001.

[10] E. Chávez, N. Reyes, and P. Roggero. Delayed insertion strategies in dynamic metric indexes. In *Procs. of the SCCC*, 34–42. IEEE Computer Society, 2009.

[11] F. Kasián and N. Reyes. Búsquedas por similitud en PostgreSQL. In *XVIII CACIC*, pages 1098–1107, 2012.

[12] V. Mancini, F. Bustos, V. Gil-Costa, and A.M. Printista. Data partitioning evaluation for multimedia systems in hybrid environments. In *P2P, Parallel, Grid, Cloud and Internet Computing, 7th Int. Conf. on*, 321–326, 2012.

[13] G. Navarro and N. Reyes. Dynamic spatial approximation trees. *J. of Exp. Algorithmics*, 12:1–68, 2008.

[14] C. Ochoa, V. Gil-Costa, and M. Printista. Suffix array performance analysis for multi-core platforms. In *4th Int. SuperComputing Conf. in Mexico*, 2013. Por aparecer.

[15] R. Paredes and N. Reyes. Solving similarity joins and range queries in metric spaces with the list of twin clusters. *JDA*, 7:18–35, 2009.