Recuperación y Procesamiento en Grandes Volúmenes de Datos

Luis Britos, María E. Di Gennaro, Jacqueline Fernández, Veronica Gil-Costa, Fernando Kasián, Jair Lobos, Verónica Ludueña, Marcela Printista, Nora Reyes, Patricia Roggero, Guillermo Trabes
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
jairlobos@gmail.com, guillermotrabes@hotmail.com
Edgar Chávez
Centro de Investigación Científica y de Educación Superior de Ensenada, México
elchavez@cicese.mx
Claudia Deco
Facultad de Ciencias Exactas, Ingeniería y Agrimensura, Universidad Nacional de Rosario
deco@fceia.unr.edu.ar

Resumen

Los sistemas de información actuales necesitan realizar búsquedas eficientes sobre diferentes tipos de datos, tales como texto libre, audio, video, secuencias de ADN, etc.. 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 una estructura de almacenamiento especialmente diseñada para responderla de manera eficiente. En algunos casos es necesario además considerar que estos datos no estructurados provienen de flujos continuos y no son conocidos de antemano.

Así, nuestra línea de investigación tiene como principal objetivo desarrollar herramientas eficientes para sistemas de información sobre bases de datos masivas, conteniendo datos multimedia. Con este fin, 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 bases de datos masivas de este tipo de datos y se desarrollan nuevas aplicaciones. que soporten la recolección y el procesamiento de grandes volúmenes de flujo continuo de datos no estructurados.

Palabras Claves: *recuperación de información, computación de alto desempeño, grandes bases de datos.*

1. Contexto

Esta línea de investigación se encuentra enmarcada dentro del Proyecto Consolidado 3-30114 de la Universidad Nacional de San Luis (UNSL) y en el Programa de Incentivos (código 22/F434): "Tecnologías Avanzadas Aplicadas al Procesamiento de Datos Masivos", dentro de la línea "Recuperación de Datos e Información", desarrollada en el Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC) de la UNSL.

En este contexto, se pretende aportar a la incorporación de información no estructurada en los procesos de toma de decisiones y resolución de problemas, no considerados en los enfoques clásicos. Por lo tanto, nuestra línea se dedica, principalmente, al diseño de índices eficientes que sirvan de apoyo a sistemas de recuperación de información orientados a conjuntos masivos de datos multimedia. Se espera así contribuir a estos sistemas obteniendo índices más eficientes para memorias jerárquicas, dinámicos, con E/S eficiente, escalables (capaces de manejar grandes volúmenes de datos), considerando técnicas de computación de alto desempeño (HPC) y cuyos datos puedan provenir de un flujo contínuo de datos.

2. Introducción y Motivación

Con el uso masivo de internet, estamos en presencia de un fenómeno donde la rápida aceleración tanto del crecimiento del volumen de datos capturados y almacenados, como la creciente variación en los tipos de datos requeridos, hace que las técnicas tradicionales para el procesamiento, análisis y obtención de información útil deban ser redefinidas para formular nuevas metodologías de abordaje.

Los sistemas de computación tradicionales 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 direc-

tamente 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 [6], principalmente por el gran 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.

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*. 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 desigualdad 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). En conjuntos masivos de datos la búsqueda secuencial es impráctica y los repositorios de datos multimedia o los flujos de datos multimedia continuos son grandes volúmenes de datos. Así, en caso de disponer de los datos, debemos preprocesar la base de datos para construir un índice, para responder a las consultas con la menor cantidad de cálculos de distancia. Además, en estos casos 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.

Entonces, esta propuesta está enfocada en mejorar herramientas de recuperación, desarrollando nuevas técnicas y aplicaciones que soporten la interacción con el usuario, diseñando estructuras de datos (índices), capaces de manipular eficientemente grandes volúmenes de datos y facilitando la realización de diferentes operaciones y considerando que en algunos casos los datos son provistos mediante un flujo continuo, de modo de acercarse al nivel de desarrollo que se posee en bases de datos tradicionales.

3. 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.

3.1. Desarrollo de Aplicaciones "Stream Processing"

"Stream processing" corresponde a un paradigma de computación distribuida que soporta la recolección y el proceso de grandes volúmenes de flujo continuo de datos heterogéneos, lo que permite generar análisis de estos datos en tiempo real para determinar decisiones. Este paradigma tiene una diferencia respecto al análisis tradicional, el cual radica básicamente en como la información es leída. En una arquitectura convencional se considera que la información se encuentra en un repositorio. En una arquitectura de "stream processing", por el contrario, la información está dividida en segmentos y los segmentos se procesan antes de que otro llegue. Ejemplos donde se puede aplicar este paradigma es la Web 2.0, sensores de redes, análisis de mercado, transacciones comerciales, etc.. Dentro del mercado existen herramientas disponibles para realizar stream processing sobre "clusters" de computadoras como lo son: *SPC (IBM)* [1], *S4 (Yahoo!)* [13][1], *Storm (Twit-*

[1] S4 Distributed stream computing platform disponible desde `http://incubator.apache.org/s4/`, 2014.

ter) [2], *Kinesis (Amazon Web Services)* [3], entre otros.

Dentro del proyecto de investigación descripto en este trabajo, se considera la plataforma de stream processing *S4*. Dentro de *S4* la unidad básica de procesamiento son los *elementos de procesamiento* (*PEs*) y los mensajes se intercambian entre ellos. Se ha desarrollado un conjunto de aplicaciones "benchmarks" para determinar las operaciones más relevantes y costosas de la plataforma. Por medio de estas *aplicaciones benchmarks* [11] las operaciones más importantes están relacionadas al proceso y control de "threads". Otro aspecto importante, corresponde al estudio y mejoras del diseño de sistemas stream processing utilizando el concepto de *elasticidad*. Para este caso se considera un modelo que tenga en cuenta lo siguiente: **(a)** *Un módulo que analiza la cola para los elementos de procesamiento*; **(b)** *Un módulo de balance de carga que mueve los elementos de procesamiento que considera operadores en uso y no uso* y **(c)** *Un módulo de elasticidad que asigna/desasigna elementos de procesamientos para adaptar el procesamiento paralelo a las condiciones del tráfico de datos.*

Como parte del proyecto de investigación se tiene por objetivo **(a)** Evaluar estrategias para mejorar el proceso y comportamiento de plataformas de stream processing y **(b)** Diseñar e implementar un simulador para la plataforma que permita probar y mejorar algoritmos para plataformas de stream processing.

Diseño de Índices

Existe un gran número de índices para espacios métricos [6]. En su gran mayoría usan la desigualdad triangular para evitar el análisis secuencial de la base de datos. Mediante la desigualdad triangular se puede estimar la distancia entre la consulta q y los objetos de la base de datos, calculando de antemano algunas distancias a objetos distinguidos, y así evitar calcular las distancias reales desde q a algunos objetos durante una búsqueda. Las distintas técnicas difieren en si esos objetos distinguidos son *pivotes* o *centros*. Si son pivotes se almacenan las distancias de todos los objetos de la base de datos a ellos. Si son centros se particiona el espacio en regiones denominadas *particiones compactas*, por cercanía a los centros y se almacena un radio de cobertura para determinar la zona de cada centro.

[2] [Storm Storm [Online], disponible desde `https://github.com/nathanmarz/storm/wiki`

[3] [Kinesis disponible desde `aws.amazon.com/kinesis/`

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 conjuntos masivos 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. Otra manera de acelerar la respuesta a una consulta es admitir una respuesta aproximada a la consulta; es decir bajar la calidad o exactitud de la respuesta para conseguirla en menor tiempo.

En [8] se define un índice para búsquedas aproximadas que combina un algoritmo basado en *Permutaciones* con una *Lista de Clusters*. Dicho índice permite ahorrar espacio al calcular la distancia entre permutaciones y además se puede evitar el recorrido secuencial del índice, en el algoritmo basado en permutaciones, mediante la definición de un parámetro. Sin embargo, este índice está definido en memoria principal y por lo tanto se pretende obtener una implementación que permita trabajar con conjuntos de datos masivos en memoria secundaria.

Además nos interesa mejorar el desempeño de índices dinámicos jerárquicos (árboles) 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.

El *árbol de aproximación espacial distante DiSAT* [7] es un índice para espacios métricos muy eficiente

en cuanto al 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 [3] 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* para resolver eficientemente el problema, no "descomponible", de la búsqueda de k-vecinos más cercanos.

En aplicaciones sobre conjuntos de datos masivos 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 [2] y demostraron ser competitivas frente a otras de las estructuras conocidas tales como el *M-tree* y *DSA*-tree* y *DSA+-tree* [12]. Además, existen nuevas propuestas en evaluación (una nueva versión del *DSATCL-tree* y un *DiSAT* dinámico) que prometen ser aún más adecuadas para memoria secundaria. Por otro lado, mediante la aplicación de técnicas de computación de alto desempeño es posible lograr mejor desempeño. Por lo tanto, se buscará aplicar y comparar distintas estrategias de paralelización con el fin de determinar la más adecuada.

Consultas sobre Bases de Datos Multimedia

Las operaciones tradicionales sobre bases de datos multimedia son las búsquedas por rango o de k-vecinos más cercanos. Sin embargo, 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 Φ 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 situaciones distintas posibles al resolver el join por similitud: que ambas, una o ninguna de la bases de datos posean un índice; 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 [14], 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 indexa 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 similitud [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 [9].

4. Resultados

Se ha comprobado experimentalmente que el *DSAT* es mejor que su versión estática [12], por dejar como "vecinos" a objetos alejados, lo que permite avanzar en la exploración espacial a "pasos más grandes", obteniendo así el *DiSAT* [7]. Se obtuvo un nuevo método de eliminación para el *DSAT* [10].

Se implementaron dos versiones: *DSACL*-tree* y *DSACL+-tree*, que trabajan con grandes volúmenes de datos, 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.

Como se ha mencionado previamente, se ha desarrollado un conjunto de aplicaciones de comparación para determinar las operaciones más relevantes y costosas de la plataforma S4, que resultaron ser las relacionadas al proceso y control de "threads" [11].

5. 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: una sobre "Planeación de Capacidad en Centros de Datos para Sistemas Escalables para la Web" (con beca de posgrado de CONICET).

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 y la otra sólo aplicando optimizaciones que consideran la jerarquía de memorias, y una sobre un DBMS para bases de datos métricas.

Trabajo Final de Licenciatura en Cs. de la Computación: uno sobre "Aplicación de Multi-BSP para la estimación de costo de algoritmos sobre plataformas multi-core".

Referencias

[1] L. Amini, H. Andrade, R. Bhagwan, F. Eskesen, R. King, P. Selo, Y. Park, and C. Venkatramani. Spc: A distributed, scalable platform for data mining. In *Proc. of the 4th Int. Workshop on Data Mining Standards, Services and Platforms*, pages 27–37, NY, USA, 2006. ACM.

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

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

[4] L. Britos, A. M. Printista, and N. Reyes. Dynamic spatial approximation trees with clusters for secondary memory. In G. Simari, P. Pesado, and J. Paganini, editors, *XVI CACIC Selected Papers*, Computer Science & Technology Series, pages 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*, page 26, 2008.

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

[7] E. Chávez, V. Ludueña, N. Reyes, and P. Roggero. Faster proximity searching with the distal SAT. In A. Traina, C. Traina Jr., and R. Cordeiro, editors, *SISAP*, vol. 8821 of *LNCS*, pages 58–69. Springer, 2014.

[8] K. Figueroa and R. Paredes. List of clustered permutations for proximity searching. In N. Brisaboa, O. Pedreira, and P. Zezula, editors, *SISAP*, vol. 8199 of *LNCS*, pages 50–58. Springer, 2013.

[9] F. Kasián and N. Reyes. Búsquedas por similitud en PostgreSQL. In *Actas del XVIII Congreso Argentino de Ciencias de la Computación (CACiC)*, pages 1098–1107, 2012.

[10] F. Kasián, V. Ludueña, N. Reyes, and P. Roggero. An efficient alternative for deletions in dynamic spatial approximation trees. *Journal of Computer Science & Technology*, 14(1):39 – 45, April 2014.

[11] J. Lobos, V. Gil-Costa, and M. Marin. Benchmark applications for stream processing profiling. In *IV Workshop de Sistemas Distribuidos y Paralelismo (WSDP 2014), JCC*, 2014.

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

[13] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari. S4: Distributed stream computing platform. In *Proceedings of the 2010 IEEE International Conference on Data Mining Workshops*, ICDMW '10, pages 170–177, Washington, DC, USA, 2010. IEEE Computer Society.

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