Administración y Recuperación de Datos Multimedia Masivos

Luis Britos, Fernando Kasián, Verónica Ludueña, Franco Merenda, Diego Olivares, Marcela Printista, Nora Reyes, Patricia Roggero, Pablo Samat
LIDIC, Dpto. de Informática, Fac. de Cs. Físico Matemáticas y Naturales, Universidad Nacional de San Luis
{*fkasian, vlud, mprinti, nreyes, proggero*}@unsl.edu.ar,
{*ldoorz, merenda.franco83, olivarestello.diego, samatpablo*}@gmail.com
Karina Figueroa
Universidad Michoacana de San Nicolás de Hidalgo, México
karina@fismat.umich.mx
Claudia Deco
Facultad de Ciencias Exactas, Ingeniería y Agrimensura, Universidad Nacional de Rosario
deco@fceia.unr.edu.ar

Resumen

Dado el crecimiento sostenido en los últimos tiempos de la disponibilidad de diferentes tipos de datos multimedia, por el uso masivo de dispositivos que producen datos digitalizados, se ha vuelto necesario contar con grandes repositorios de datos no estructurados provenientes de distintas fuentes, los cuales son difíciles de administrar bajo el modelo relacional de base de datos. Este crecimiento sostenido tanto de la cantidad de datos en formato digital disponible, como en la variedad de los mismos, ha ido acompañado por la rápida evolución de las tecnologías de la información y comunicación. Al considerar tipos de datos tales como texto libre, imágenes, audio, video, secuencias biológicas de ADN o proteínas, entre otros, su administración y las distintas formas de recuperación de datos no se corresponden a las habituales y forzar una modelización de los mismos no adecuada podría restringir de antemano el tipo de operaciones que se puedan realizar sobre ellos.

Así, al considerar la administración y recuperación sobre grandes conjuntos de datos no estructurados, se hace evidente la necesidad de considerar nuevos modelos y herramientas para su administración y su indexación, considerando las operaciones de interés. Además, como el objetivo de cualquier sistema de recuperación de información es obtener información útil para el usuario, mediante consultas a la base de datos, no sólo se deben soportar distintos tipos de consultas sino también resolverlas de manera eficiente usando índices apropiados.

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

1. Contexto

Esta línea de investigación "Recuperación de Datos e Información" se encuentra enmarcada dentro del Proyecto Consolidado 3-03-2018 de la Universidad Nacional de San Luis (UNSL - Res. CS 126/18) y en el Programa de Incentivos (Código 22/F834): "Tecnologías Avanzadas Aplicadas al Procesamiento de Datos Masivos", se desarrolla en el Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC) de la UNSL y su finalización, debido a la pandemia, se postergó a 2023.

El principal objetivo de esta línea es desarrollar herramientas eficientes para administrar bases de datos masivas, que almacenan datos no estructurados. En este sentido, se analizan nuevas técnicas que provean una buena interacción con el usuario y se desarrollan nuevas estructuras de datos capaces de indexar un gran volumen de datos no estructurados, que permitan resolver eficientemente consultas de interés para la recuperación de información sobre estos tipos de datos. Para ello, se considera el diseño y desarrollo de índices para conjuntos de datos no estructurados masivos (datos multimedia, texto, secuencias de ADN, huellas digitales, etc.), que sean eficientes y escalables, para memorias jerárquicas, aplicando técnicas de computación de alto desempeño (HPC). Además, se busca su incorporación en un sistema de administración para estas bases de datos, que apoye la recuperación de información sobre datos multimedia masivos.

2. Introducción y Motivación

Debido al uso masivo de internet, a la gran disponibilidad de dispositivos electrónicos y a la evolución de las tecnologías de información y comunicación, se están generando una cantidad masiva de datos. Además, por provenir de fuentes muy diversas, los tipos de datos generados son también muy variados. En este contexto de problemas de "big data"

(porque aparecen sus dos características importantes, que son: el volumen de datos y variedad de los mismos) se hace imposible restringir las consultas a búsquedas sobre datos estructurados tradicionales, porque obligaría a representar una visión parcial del problema. Por ello, para no perder información relevante, se deben redefinir las técnicas de procesamiento, análisis y obtención de información útil y formular nuevas metodologías.

Un enfoque útil para sistemas de recuperación de información es *la búsqueda basada en contenidos*, donde se usa el dato no estructurado en sí mismo para describir lo que se está buscando. En este escenario son aplicables las búsquedas por similitud, por ser más generales y aplicables. Para lograr resolver dichas búsquedas eficientemente, se deben considerar *métodos de acceso* o *índices métricos* [3].

Por lo tanto, al considerar grandes cantidades de datos no estructurados y ser necesario responder consultas de recuperación de información, se pueden utilizar estos índices métricos para obtener las respuestas de manera más eficiente. En aplicaciones reales, estos índices en general además de eficientes, deben permitir actualizaciones (altas y bajas de elementos) y ser escalables; es decir, no degradar su desempeño significativamente a medida que crece el volumen de datos a indexar.

Los *espacios métricos*, compuestos por un *universo* $\mathcal{U}$ de objetos y una *función de distancia* $d : \mathcal{U} \times \mathcal{U} \longrightarrow \mathbb{R}^+$, resultan adecuados para modelizar las búsquedas por similitud. En particular, d cumple con las propiedades de una métrica (reflexividad, positividad estricta, simetría y desigualdad triangular). La función de distancia permite medir la disimilitud entre dos objetos. Es habitual considerar dos tipos básicos de búsqueda por similitud: la *búsqueda por rango* y la *búsqueda de los k vecinos más cercanos*. El cálculo de distancia sobre algunos tipos de datos no estructurados puede ser muy costoso. Por lo tanto, un objetivo importante en el diseño de los índices es ahorrar cálculos de distancia. Una de las principales ventajas de este modelo es que, además de brindar un marco formal, es independiente del dominio de la aplicación. Al considerar una base de datos $S \subseteq \mathcal{U}$ tal que $\mid \mathcal{S} \mid \quad n$, cualquier consulta se resuelve de manera trivial calculando n evaluaciones de distancia. Las distancias en general son costosas de calcular (por ej.: comparación de huellas digitales). Por ello, en la mayoría de las aplicaciones sobre datos masivos no basta con resolver el problema de manera trivial. Así, para minimizar los cálculos de distancia, se preprocesa S para construir un índice.

Sin embargo, en muchos casos es probable que la base de datos, el índice, o ambos, no puedan almacenarse en memoria principal y deban alojarse en memoria secundaria. Esto puede incrementar mucho los costos de las búsquedas, dado que las operaciones de E/S en memoria secundaria suelen ser muy costosas. Por lo tanto, para lograr eficiencia, no basta con realizar pocos cálculos de d sino que también se debe minimizar el número de operaciones de E/S. Para ello, hay que considerar el nivel de la jerarquía de memorias sobre la que se trabaja, utilzar técnicas de computación de alto desempeño (HPC) y, en algunos casos, admitir respuestas no exactas.

En esta línea, se busca desarrollar nuevas técnicas y aplicaciones que soporten la interacción con el usuario, diseñar índices que permitan la manipulación eficiente de grandes volúmenes de datos no estructurados y faciliten la realización de diferentes tipos de consultas, para construir herramientas de recuperación de información sobre conjuntos masivos de datos. De esta manera, se espera contribuir al desarrollo de soluciones a problemas de big data para aplicaciones reales.

3. Líneas de Investigación

Dado que en esta línea se pretende contribuir a distintos aspectos de los sistemas de recuperación de información sobre grandes volúmenes de datos no estructurados, se ha considerado proveer de un administrador de sistemas para este tipo de bases de datos y, por otra parte, optimizar índices existentes, diseñar nuevos índices, resolver distintas clases de consultas, y lograr eficiencia y escalabilidad para grandes volúmenes de datos.

DBMS para Bases de Datos Multimedia

A pesar de que las operaciones más comunes sobre bases de datos multimedia son las búsquedas por rango o de k-vecinos más cercanos, la operación de *join* por similitud debería brindarse en un sistema administrador para bases de datos multimedia [10].

Hay distintas variantes del join por similitud entre dos bases de datos A y B, con $A, B \subseteq \mathcal{U}$, dependiendo del criterio de similitud Φ utilizado. El resultado de cualquiera de las variantes de esta operación es un conjunto de pares formados (x, y), $x \in A$ e $y \in B$, tales que se satisface el criterio de similitud Φ entre x e y. Las variantes más conocidas son: el join por rango, el join de k-vecinos más cercanos y el join de k pares de vecinos más cercanos; entre

otras. Formalmente, dadas $A, B \subseteq \mathcal{U}$, el *join por similitud* entre A y B es $(A \underset{\Phi}{\bowtie} B) = \{(x, y)/x \in A \wedge y \in B \wedge \Phi(x, y)\}$. Si $A = B$, la operación se denomina *auto-join*. Las variantes más conocidas son: el join por rango, el join de k-vecinos más cercanos y el join de k pares de vecinos más cercanos; entre otras. Al resolver el join por similitud es posible 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 [9]. Por lo tanto, un DBMS para estas bases de datos debe considerar cómo resolver eficientemente las consultas en función del índice utilizado.

PostgreSQL es un DBMS que permite realizar consultas por similitud sobre algunos atributos. Por ejemplo, si se han creado índices *KNN-GiST* sobre texto, se podrían resolver búsquedas de k-vecinos más cercanos sobre ellos. Sin embargo, estos índices brindan plantillas sólo para crear *árboles balanceados* tales como: *B-tree* o *R-tree*. Pero, se ha mostrado que el "balance" no siempre es bueno para los índices sobre espacios métricos [1]. Por otro lado, *PostgreSQL* no dispone de este tipo de consultas sobre todo tipo de datos métricos. Por lo tanto, es importante incorporar en un DBMS la capacidad de manejar distintos tipos de datos no estructurados y todas las operaciones de interés sobre ellos.

En las respuestas a consultas de join por similitud pueden existir pares muy similares entre sí. Por lo tanto, para acelerar los tiempos de respuesta, disminuyendo los tamaños de los conjuntos obtenidos, se pueden aplicar técnicas de diversificación de las respuestas [11]. En este caso, se obtiene un conjunto más pequeño de pares, con respuestas útiles y diversificadas, y en menor tiempo. Otra manera de acelerar el tiempo de respuesta es brindando una respuesta aproximada a la consulta. Por ello, se ha propuesto un algoritmo simple y eficiente que permite responder consultas de join por similitud de k vecinos más cercanos aproximados dentro del mismo conjunto (auto-join), con una razonable precisión en la respuesta [4]. Por lo tanto, se espera incorporar éste y otros de los desarrollos de esta línea a *PostgreSQL* para obtener un DBMS más aplicable en sistemas de información sobre datos multimedia masivos.

Métodos de Acceso Métricos

Como se ha mencionado, los métodos de acceso métricos (MAMs) permiten responder a las búsquedas sobre conjuntos de datos no estructurados de manera más eficiente que si se examina toda la base de datos [3]. Los MAMs aprovechan las distancias almacenadas en el índice y que la función de distancia satisface la desigualdad triangular para ahorrar cálculos de distancia y tiempo. En general, los MAMs mantienen algunas distancias precalculadas entre los elementos de la base de datos y objetos particulares o distinguidos. Así, durante las búsquedas se puede estimar la distancia entre cualquier objeto de consulta q y los elementos de la base de datos utilizando la desigualdad triangular, sin calcular d realmente. Existen dos enfoques principales para los MAMs, dependiendo si esos objetos distinguidos son *pivotes* o *centros*. Si los elementos distinguidos son pivotes, se almacenan las distancias de todos los objetos de la base de datos a ellos. Por el contrario, si son centros se particiona el espacio en zonas denominadas *particiones compactas*, por cercanía a los centros. Estas particiones almacenan un radio de cobertura que determina la zona de cada centro.

Cuando se diseñan MAMs, se deben considerar varios aspectos: dinamismo (que soporten altas y bajas), en qué nivel de la jerarquía de memorias deben almacenarse, si es posible aplicar técnicas de HPC para mejorar los tiempos de respuesta, si son de respuesta exacta o de respuesta aproximada y la dimensionalidad del espacio métrico considerado.

Dinamismo: Si la base de datos se conoce de antemano, se construye el índice (*estático*) y luego se realizan las consultas. Por el contrario, la única manera de construir el índice (*dinámico*) es a medida que se incorporan los elementos; es decir, de manera incremental, se considera que las búsquedas pueden realizarse en cualquier momento. Los índices estáticos pueden seleccionar los mejores objetos distinguidos para el índice, los dinámicos no.

Jerarquía de Memorias: Otro aspecto importante para buscar una solución es saber si se puede trabajar en memoria principal o, por el contrario, si por ser conjuntos de datos masivos se deberá trabajar en otros niveles de la jerarquía de memorias. En caso que el índice deba alojarse en memoria secundaria, se deben minimizar la cantidad de cálculos de distancia y también el número de operaciones de E/S.

Computación de Alto Desempeño: En algunos casos, si no se logra la eficiencia deseada mediante la optimización del índice en sí mismo, se pueden aplicar técnicas de computación de alto desempeño (HPC) para acelerar el tiempo de respuesta.

Exactitud de la Respuesta: Otra manera de acelerar la respuesta a una consulta por similitud es admitir una respuesta aproximada, permitiendo que la misma no sea exacta, pero si muy rápida.

Por el volumen de los datos con los que se trabaja (por ejemplo, millones de imágenes en la Web) se debe considerar que posiblemente los índices deban almacenarse en otros niveles de la jerarquía de memoria. En general, se considera directamente su almacenamiento en memoria secundaria y sólo disponer de memoria principal para mantener información muy útil que permita determinar que partes del índice deben recuperarse desde el disco. Por ello, para lograr eficiencia, el índice debe minimizar tanto el número de cálculos de distancia como de operaciones sobre el disco (E/S) en las búsquedas.

Así, una parte de esta línea se dedica a diseñar MAMs, preferentemente dinámicos y adaptados para memoria secundaria, cuyo desempeño en las búsquedas sea adecuado para las aplicaciones de interés. Algunos de ellos se basan en la *Lista de Clusters* (*LC*) [3, 2] son totalmente dinámicos, es decir, admiten inserciones y eliminaciones de objetos y están especialmente diseñados para trabajar sobre grandes volúmenes de datos [8]. Uno de ellos, el *Conjunto Dinámico de Clusters* (*DSC*), que tiene una buena ocupación de página y operaciones eficientes tanto en cálculos de distancia como en operaciones de E/S, logra un buen desempeño en espacios de alta dimensión. El *DSC* mantiene los clusters en memoria secundaria y organiza los centros de clusters en un *DSAT* en memoria principal, para lograr búsquedas con menos cálculos de distancia y accesos páginas/clusters. La información en el *DSAT* permite mejorar los costos en cálculos de distancia, y mantener bajos los costos de acceso a disco.

Sin embargo, cada inserción en un *DSC* debe localizar el clúster donde se insertará el nuevo elemento. Luego, se realiza una lectura del clúster desde el disco y se verifica si el nuevo objeto podría ser un mejor centro para el clúster, haciendo si es posible el clúster más compacto. Finalmente, se graba nuevamente el clúster en el disco, actualizando además el *DSAT* de centros si el nuevo objeto se transformó en el nuevo centro. Aunque esta operación, además de los cálculos de distancia necesarios, realiza dos operaciones de E/S en el disco (podrían ser 3 si el clúster estaba ya lleno), es posible considerar bajar esta cantidad de operaciones de E/S. Para ello, se está analizando una variante para el *DSC* en la cual, en lugar de insertar los elementos en el índice a medida que van llegando, se demora la incorporación de cada nuevo objeto a un clúster, hasta tener varios elementos y poder determinar un mejor agrupamiento de los mismos, que mejore los costos de búsqueda, al lograr clusters aún más compactos y que aseguren una total ocupación de la página del disco, achicando el tamaño del archivo y reduciendo los tiempos de acceso. Esta opción, además podría reducir significativamente el costo de construcción, por amortizar el costo de la escritura de *un* clúster en disco, luego de varias inserciones.

Así, se ha diseñado el *Buffered On Line Dynamic Set of Clusters* (BOLDSC), una variante del *DSC*, que agrega un espacio en memoria para mantener los objetos que son insertados (bolsa), además de un índice auxiliar (de pivotes) sobre estos elementos. Como pivotes se eligen elementos de la bolsa, y cuando ésta se llena, se selecciona el pivote que necesita el menor radio de cobertura para encerrar la cantidad de elementos que caben en un clúster y todos éstos se sacan de la bolsa y se graban en un clúster completo. La grabación de clusters llenos mejora la cantidad de E/S, pero puede provocar que objetos cercanos, que se insertan en diferentes momentos, queden en clusters diferentes, ocasionando degradación en las búsquedas. Por ello, se está estudiando una variante de la BOLDSC que graba clusters con una cantidad de objetos que está en función de un factor de carga ρ (casi llenos). Como los clusters no estarán llenos, ellos tendrán un espacio libre en cada clúster. Esto se considera al insertar un nuevo objeto. Primero se verifica si hay clusters que lo pueden incorporar y de ellos se elige al más cercano a él. Se verifica también si el nuevo elemento sería un mejor centro para el clúster y si es así, se actualiza el *DSAT* de centros. Si no es posible incorporarlo a un clúster (todos llenos), se lo incorpora a la bolsa y se actualiza el índice de pivotes. Así, los clústers se adaptan y se mejoran los costos en las búsquedas.

El *DSC* basa su buen desempeño en la calidad de la partición generada por los centros de sus clusters. Por lo tanto, otro aspecto a considerar es mejorar la poda en estos agrupamientos. Cada clúster en *DSC* está determinado por su centro y su radio de clúster. Sin embargo, puede haber "huecos" dentro de ellos. Por esta razón, es posible seleccionar un conjunto de "pivotes" globales que permitan caracterizar sobre cada uno de los clusters las zonas dentro de las cuales existan efectivamente elementos [7]. De esta manera, se puede mantener para cada clúster información sobre la mínima y la máxima distancia

a la que cada pivote encuentra elementos de dicho clúster. Esta información se utiliza para identificar los "huecos" del clúster para que luego, durante las consultas, se pueda descartar un clúster si la bola de la consulta no posee intersección efectiva con la zona del clúster que realmente contiene elementos de interés. En las búsquedas se recupera desde memoria secundaria cada clúster que intersecta la bola de consulta con centro q y radio r. Al identificar las zonas del clúster que no contienen efectivamente elementos, si la bola de consulta intersecta al clúster en su zona "hueca" se puede descartar el clúster y ahorrar cálculos de distancia y lecturas a disco.

Otra manera de acelerar las búsquedas es paralelizar el índice usando técnicas de HPC. Así, se está trabajando en el diseño e implementación de una versión paralela del *Conjunto Dinámico de Clusters* (*DSC*) [8]; esperando, no sólo bajar la cantidad de cálculos de distancia y operaciones de E/S necesarias para responder a una consulta, sino aumentar la cantidad de consultas simultáneas a resolver, aprovechando al máximo las operaciones de E/S que se realicen durante las mismas. Para ello, se aplicarán y compararán distintas versiones paralelas del *DSC*.

4. Resultados

Recientemente, además de una estrategia para obtener buenas listas de candidatos para las búsquedas por similitud usando permutaciones en un índice invertido [5], se ha publicado una manera de acortar dichas listas de candidatos para acelerar las búsquedas [6]. Por otro lado, se está evaluando experimentalmente la versión paralela del índice *DSC*, diseñada para memoria secundaria, que admite inserciones/eliminaciones de objetos y permitirá responder eficientemente lotes de consultas. Además, se encuentran en etapa de evaluación la *BOLDSC* y en etapa de implementación las otras mejoras al *DSC*. Se continúa trabajando en la extensión de *PostgreSQL*, probando el prototipo de DBMS para consultas por similitud sobre distintos tipos de datos.

5. Formación de Recursos

Se están realizando las tesis de Maestría: (1) - "Estructuras Eficientes sobre Datos Masivos para Búsquedas en Espacios (UNSL) , (2) - "Sistema Administrador para Bases de Datos Métricas" (UNSL), (3) - "Índices Métricos – Optimización del DSC usando Cortes de Regiones" (UNSJ) y (4) - "Optimización del BOLDLC por la Mejora de la Densidad y Solapamiento de los Clusters"(UNSJ).

Referencias

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

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

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

[4] S. Ferrada, B. Bustos, and N. Reyes. An efficient algorithm for approximated self-similarity joins in metric spaces. *Information Systems*, 91:101510, 2020.

[5] K. Figueroa, N. Reyes, and A. Camarena-Ibarrola. Candidate list obtained from metric inverted index for similarity searching. In *Advances in Computational Intelligence*, 29–38, 2020. Springer International Publishing.

[6] K. Figueroa, A. Camarena-Ibarrola, and N. Reyes. Shortening the candidate list for similarity searching using inverted index. In *Pattern Recognition*, 89–97, 2021. Springer International Publishing.

[7] J. Lokoč, J. Moško, P. Čech, and T. Skopal. On indexing metric spaces using cut-regions. *Information Systems*, 43:1–19, 2014.

[8] G. Navarro and N. Reyes. New dynamic metric indices for secondary memory. *Information Systems*, 59:48 – 78, 2016.

[9] 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.

[10] C. Rong, C. Lin, Y. N. Silva, J. Wang, W. Lu, and X. Du. Fast and scalable distributed set similarity joins for big data analytics. In *IEEE 33rd International Conference on Data Engineering (ICDE)*, 1059–1070, 2017.

[11] L. Santos, L. Carvalho, W. Oliveira, A. Traina, and C. Jr. Traina. Diversity in similarity joins. In *Similarity Search and Applications*, vol. 9371, *LNCS*, 42–53. Springer International Publishing, 2015.