Implementación de un Digesto Digital Paralelo para Búsquedas por Similitud sobre Documentos*

Roberto Solar-Gallardo[1,2], Roberto Uribe-Paredes[1,2],
Esteban Gesto[3], and Osiris Sofia[3]

[1] Depto. de Ingeniería en Computación,
Universidad de Magallanes, Chile
[2] Grupo de Bases de Datos - Unidad Académica de Río Turbio ,
Universidad Nacional de la Patagonia Austral, Río Turbio, Argentina
[3] Unidad Académica de Río Gallegos,
Universidad Nacional de la Patagonia Austral, Río Turbio, Argentina
e-mails: {rsolar,ruribe}@ona.fi.umag.cl - {egesto,osofia}@unpa.edu.ar

Abstract. *Similarity search* consists in retrieving those objects within a database that are alike or relevant in a given query. This concept has a broad range of applications in diverse areas such as multimedia database, pattern recognition, data mining, information retrieval, etc. The necessity to process large amounts of data to find fast answers to queries makes the structures that support this system parallel.
Besides, the quantity of information generated by public and private institutions and the need to recuperate documents in a much more complex way allows for the union of parallel and similarity search areas to makes a real application.
This article presents an efficient solution with a low cost parallel search engine as an alternative to queries in an *Institutional Digital Digest*, document search by similitude.
keywords: database, data structures, algorithms, metric spaces, similarity search, parallel processing, modelo BSP.

Resumen La *búsqueda por similitud* consiste en recuperar todos aquellos objetos dentro de una base de datos que sean parecidos o relevantes a una determinada consulta. Este concepto tiene una amplia gama de aplicaciones en áreas como bases de datos multimedia, reconocimiento de patrones, minería de datos, recuperación de información, etc. La necesidad de procesar grandes volúmenes de datos y de poner a disposición de los usuarios respuestas rápidas a sus consultas hace que las estructuras que soportan este tipo de búsquedas deban ser paralelizadas.
Por otro lado, la cantidad de información generada por instituciones públicas y privadas y la necesidad de recuperar documentos de formas más complejas, permite la unión de las áreas de paralelismo y búsqueda por similitud en una aplicación real.
El presente artículo presenta un solución eficiente y de bajo costo de un motor de búsqueda paralelo, presentando una alternativa para consultas en un *Digesto Digital Institucional*, la búsqueda de documentos por similitud.
Palabras claves: bases de datos, estructuras de datos, algoritmos, espacios métricos, consultas por similaridad, paralelismo, modelo BSP.

1. Marco Teórico

1.1. Antecedentes

Uno de los problemas de gran interés en ciencias de la computación es el de "búsqueda por similitud", es decir, encontrar los elementos de un conjunto más similares a una

* Parcialmente financiado por Universidad de Magallanes (programa de investigación PR-F1-02IC-08), Chile y Universidad Nacional de la Patagonia Austral (proyectos de investigación 29/C035/1, Unidad Académica de Río Turbio y 29/A216-1, Unidad Académica de Río Gallegos), Argentina.

muestra. Esta búsqueda es necesaria en múltiples aplicaciones, como ser en reconocimiento de voz e imagen, compresión de video, genética, minería de datos, recuperación de información, etc. En casi todas las aplicaciones la evaluación de la similaridad entre dos elementos es cara, por lo que usualmente se trata como medida del costo de la búsqueda la cantidad de similitudes que se evalúan.

Interesa el caso donde la similaridad describe un espacio métrico, es decir, está modelada por una función de distancia que respeta la desigualdad triangular. En este caso, el problema más común y difícil es en aquellos espacios de "alta dimensión" donde el histograma de distancias es concentrado, es decir, todos los objetos están más o menos a la misma distancia unos de otros.

El aumento de tamaño de las bases de datos y la aparición de nuevos tipos de datos sobre los cuales no interesa realizar búsquedas exactas, crean la necesidad de plantear nuevas estructuras para búsqueda por similaridad o búsqueda aproximada. Asimismo, se necesita que dichas estructuras sean dinámicas, es decir, que permitan agregar o eliminar elementos sin necesidad de crearlas nuevamente, así como también que sean óptimas en la administración de memoria secundaria. La necesidad de procesar grandes volúmenes de datos obligan a aumentar la capacidad de procesamiento y con ello la paralelización de los algoritmos y la distribución de las bases de datos.

En este contexto, la información generada en documentos de instituciones públicas y privadas presenta un problema de interés en términos de búsqueda de documentos. Muchas veces no se requiere una búsqueda tradicional, si no más específica en función de obtener documentos similares a uno dado como consulta.

1.2. Búsqueda por Similitud

La similaridad se modeliza en muchos casos interesantes a través de un espacio métrico, y la búsqueda de objetos más similares a través de una búsqueda por rango o de vecinos más cercanos.

Definición 1 (*Espacios Métricos*): Un espacio métrico es un conjunto X con una función de distancia $d : X^2 \rightarrow R$, tal que $\forall x, y, z \in X$,

1. $d(x, y) \geq 0$ *and* $d(x, y) = 0$ *ssi* $x = y$. *(positividad)*
2. $d(x, y) = d(y, x)$. *(Simetría)*
3. $d(x, y) + d(y, z) \geq (d(x, z)$. *(Desigualdad Triangular)*

Definición 2 (*Consulta por Rango*): Sea un espacio métrico *(X,d)*, un conjunto de datos finito $Y \subseteq X$, una consulta $x \in X$, y un rango $r \in R$. La consulta de rango alrededor de x con rango r es el conjunto de puntos $y \in Y$, tal que $d(x, y) \leq r$.

Definición 3 (*Los k Vecinos más Cercanos*): Sea un espacio métrico *(X,d)*, un conjunto de datos finito $Y \subseteq X$, una consulta $x \in X$ *y un entero* k. Los k vecinos más cercanos a x son un subconjunto A de objetos de Y, donde la $|A| = k$ y no existe un objeto $y \in A$ tal que *d(y,x)* sea menor a la distancia de algún objeto de A a x.

El objetivo de los algoritmos de búsqueda es minimizar la cantidad de evaluaciones de distancia realizadas para resolver la consulta. Los métodos para buscar en espacios métricos se basan principalmente en dividir el espacio empleando la distancia a uno o más objetos seleccionados. El no trabajar con las características particulares de cada aplicación

tiene la ventaja de ser más general, pues los algoritmos funcionan con cualquier tipo de objeto [8].

Existen distintas estructuras para buscar en espacios métricos, las cuales pueden ocupar funciones discretas o continuas de distancia. Algunos son BKTree [6], MetricTree [15], GNAT [4], VpTree [19], FQTree [1], MTree [9], SAT [11], Slim-Tree [14], EGNAT [17].

Algunas de las estructuras anteriores basan la búsqueda en pivotes y otras en clustering. En el primer caso se seleccionan pivotes del conjunto de datos y se precalculan las distancias entre los elementos y los pivotes. Cuando se realiza una consulta, se calcula la distancia de la consulta a los pivotes y se usa la desigualdad triangular para descartar candidatos.

Los algoritmos basados en clustering dividen el espacio en áreas, donde cada área tiene un *centro*. Se almacena alguna información sobre el área que permita descartar toda el área mediante sólo comparar la consulta con su centro. Los algoritmos de clustering son los mejores para espacios de alta dimensión, que es el problema más difícil en la práctica.

Existen dos criterios para delimitar las áreas en las estructuras basadas en clustering, *hiperplanos* y *radio cobertor* (*covering radius*). El primero divide el espacio en particiones de *Voronoi* y determina el hiperplano al cual pertenece la consulta según a qué centro corresponde. El criterio de radio cobertor divide el espacio en esferas que pueden intersectarse y una consulta puede pertenecer a más de una esfera.

Definición 4 (*Diagrama de Voronoi*): Considérese un conjunto de puntos $\{c_1, c_2, \ldots, c_n\}$(centros). Se define el diagrama de Voronoi como la subdivisión del plano en n áreas, una por cada c_i, tal que $q \in$ al área c_i sí y sólo sí la distancia euclidiana $d(q, c_i) < d(q, c_j)$ para cada c_j, con $j \neq i$.

En el presente trabajo se mostrará los resultados iniciales obtenidos con dos estruturas, el *GNAT* [4] y *EGNAT* [17]. El *EGNAT* está basado en el *GNAT* que es una generalización del *Generalized Hyperplane Tree* (*GHT*) [15]. Ambas estructuras son basadas en clustering y usan diagramas de Voronoi para dividir el espacio. Para la búsqueda usan el criterio de Hiperplano, aunque igualmente usan radio cobertor.

1.3. Modelo de computación paralela BSP

El modelo BSP de computación paralela fue propuesto en 1990 con el objetivo de permitir que el desarrollo de software sea portable y tenga desempeño eficiente y escalable [18,12]. BSP propone alcanzar este objetivo mediante la estructuración de la computación en una secuencia de pasos llamados *supersteps* y el empleo de técnicas aleatorias para el ruteo de mensajes entre procesadores. El computador paralelo, independiente de su arquitectura, es visto como un conjunto de pares procesadores-memoria, los cuales son conectados mediante una red de comunicación cuya topología es transparente al programador. Los supersteps son delimitados mediante la sincronización de procesadores. Los procesadores proceden al siguiente superstep una vez que todos ellos han alcanzado el final del superstep, los cuales son agrupados en bloques para optimizar la eficiencia de la comunicación. Durante un superstep, los procesadores trabajan asincrónicamente con datos almacenados en sus memorias locales. Cualquier mensaje enviado por un procesador está disponible para procesamiento en el procesador destino sólo al comienzo

del siguiente superstep. Dada la estructura particular del modelo de computación, el costo de los programas BSP puede ser obtenido utilizando técnicas similares a las empleadas en el análisis de algoritmos secuenciales. En BSP, el costo de cada superstep esta dado por la suma del costo en computación (el máximo entre los procesadores), el costo de sincronización entre procesadores, y el costo de comunicación entre procesadores (el máximo enviado/recibido entre procesadores).

BSPonMPI es una biblioteca de software independiente de la plataforma de hardware para desarrollar programas paralelos, realizada muy recientemente. Implementa el estándar BSPlib (con una pequeña excepción) y corre sobre todas las máquinas que ejecutan MPI. Esta última característica es el rasgo principal de esta biblioteca y de esta manera se distingue de otras bibliotecas como Oxford BSP Toolset y la BSP PUB. MPI (Message Passing Interface o Interface de paso de mensajes) debería hacer más fácil la escritura de un programa paralelo. Sin embargo en la práctica todavía es muy complicado, porque esta API se compone de cientos de funciones. También es necesario cuidar en la programación que no se produzcan errores típicos de los entornos paralelos, como los deadlocks y los comportamientos no determinísticos. Por otro lado BSP consiste de sólo 20 primitivas, que proporcionan la misma funcionalidad y velocidad. BSPlib, como se conoce a esta API, permite escribir programas paralelos según el paradigma BSP [3]. Este paradigma permite desarrollar un algoritmo paralelo de una manera estructurada, generando código legible y eficiente. BSPlib ya es puesto en práctica para varios superordenadores y clusters, pero como es menos popular que MPI, no es puesto en práctica para todas las plataformas de hardware. Actualmente hay dos implementaciones principales de BSPlib: Oxford BSP Toolset y PUB. Ambos son implementaciones para plataformas de hardware específicas (Cray T3E o el Origen SGI, etc), y poseen una versión independiente de la plataforma sobre MPI. Sin embargo la arquitectura de su biblioteca de software es optimizada para el empleo de características de hardware específicas, ya que no fue el objetivo primario el desarrollo sobre MPI. Evaluaciones preliminares han demostrado que la implementación de BSPonMPI es mucho más eficiente que las implementaciones existentes [13].

2. *Digesto Digital Institucional*

El presente trabajo presenta un prototipo para un buscador por similitud sobre un espacio de documentos. La aplicación, en etapa de prueba, se ha denominado *Digesto Digital Institucional*. Originalmente el término *Digesto* se aplicó a la codificación del Derecho Romano, pero actualmente y por extensión se conoce como digesto a la compilación ordenada de toda norma jurídica. El Digesto Institucional permite acceder a todo lo actuado, sancionado y legislado en el tiempo, por una Institución dada. Constituye el cuerpo de leyes o reglamentaciones por el cual se rige la actuación y las decisiones de una administración, compendiando además, todo lo resuelto o actuado en función y con atención a ese conjunto de reglamentaciones básicas.

Se optó por implementar el Digesto Digital con la información de una de las Universidades a la que pertenecen los autores, ya que, inicialmente, el volumen de datos involucrados resultaba atractivo para realizar las pruebas de laboratorio de este trabajo. Se ha estimado, a futuro, un volumen de datos de al menos 500.000 páginas de documentos correspondientes a los órganos de gobierno de la Universidad de los últimos 10 año.

Junto con la estructura métrica que soportará toda la información, adicionalmente se almacenaran los documentos originales en formato texto en una base de datos distribuida y en formato de archivos PDF almacenados en el servidor web (para poder obtener un copia con autenticación por parte de la Universidad).

2.1. Modelo Vectorial para Documentos

En recuperación de información [2] se define un *documento* como una *unidad de recuperación*, la cual puede ser un párrafo, una sección, un capítulo, una página web, un artículo o un libro completo. Los modelos clásicos en recuperación de la información consideran que cada documento está descrito por un conjunto representativo de palabras claves llamadas *términos*, que son palabras cuya semántica ayuda a definir los temas principales del documento.

Uno de estos modelos, el *modelo vectorial*, considera un documento como un vector *t-dimensional*, donde t representa el número total de términos de la colección. Cada coordenada i del vector está asociada a un término del documento, cuyo valor corresponde a un "peso" positivo w_{ij} si es que dicho término pertenece al documento j o 0 en caso contrario. Si $\mathbb{D}$ es el conjunto de documentos y d_j es el *j-ésimo* documento perteneciente a $\mathbb{D}$, entonces $d_j = (w_{1j}, w_{2j}, ..., w_{ij})$.

En el modelo vectorial se calcula el *grado de similitud* entre un documento d y una consulta q, la cual puede ser vista como un conjunto de términos o como un documento completo, como el grado de similitud entre vectores $\overrightarrow{d_j}$ y $\overrightarrow{q}$. Esta correlación puede ser cuantificada, por ejemplo, como el coseno del ángulo formado entre ambos vectores:

$$sim(d_j, q) = \frac{\overrightarrow{d_j} \cdot \overrightarrow{q}}{\left|\overrightarrow{d_j}\right| \times |\overrightarrow{q}|} = \frac{\sum_{i=1}^{t} w_{ij} \times w_{iq}}{\sqrt{\sum_{i=1}^{t} w_{ij}^2 \times \sum_{i=1}^{t} w_{iq}^2}}$$

donde w_{iq} es el peso del *i-ésimo* término en la consulta q.

Los pesos de los términos pueden calcularse en varias formas. Una de las más importantes es utilizando *esquemas tf-idf*, en donde los pesos están dados por:

$$w_{ij} = f_{i,j} \times \log\left(\frac{N}{n_i}\right)$$

donde N es el número total de documentos, n_i es el número de documentos en donde el *i-ésimo* término aparece, y $f_{i,j}$ es la *frecuencia normalizada* del *i-ésimo* término, dada por:

$$f_{i,j} = \frac{freq_{i,j}}{\max\limits_{l=1..t}(freq_{l,j})}$$

donde $freq_{i,j}$ es la frecuencia del *i-ésimo* término en el documento d_j, y $\max\limits_{l=1..t}(freq_{l,j})$ es el máximo valor de la frecuencia sobre todos los términos contenidos en d_j.

Si se considera que los documentos son puntos en un espacio métrico, el problema de búsqueda de documentos similares a una consulta dada se reduce a una búsqueda por similaridad en el espacio métrico. Dado que $sim(d_j, q)$ es una medida de similaridad y no de distancia, se utiliza el ángulo formado entre los vectores $\overrightarrow{d_j}$ y $\overrightarrow{q}$, $d(d_j, q) = \arccos(sim(d_j, q))$, como función de distancia, la cual se denomina *distancia coseno* [2]. Así definido, se puede decir que el par $(\mathbb{D}, d)$ es un espacio métrico.

2.2. Preprocesamiento de Documentos

Durante la etapa de preprocesamiento de documentos, una vez realizados los procesos de eliminación de palabras vacías y *stemming*, se obtuvieron vectores representativos de dimensión 9,341, la dimensión original era sobre 18.000.

Existen complicaciones propias del manejo de documentos reales que entorpecieron el procesamiento de los documentos y disminuyeron el rendimiento de las estructuras. Entre estos, el uso de nombres propios, las faltas ortográficas, la referenciación de documentos previos y el contexto de los documentos. En los dos primeros casos, se provocaba un aumento de dimensión, ya que por cada palabra nueva la dimensión crecía en uno. Respecto del contexto, se puede indicar que los nombres propios de ciudades se repetían continuamente, por ejemplo, en resoluciones propias de una unidad académica, esto implicaba que dicha palabra fuera similar a una palabra vacía.

Lo anterior podría subsanarse utilizando texto semiestructurado y compartir las ventajas de la búsqueda por similitud y tradicional. Esto queda propuesto para una etapa futura de este trabajo.

Igualmente se puede mencionar, que la dimensión intrinseca de este espacio es elevada, del orden de 158, por lo que la búsqueda resulta dificultosa. Además, la cantidad de palabras de los primeros documentos de prueba eran del orden de 500 (sin palabras vacías y aplicando stemming). Esto dificultaba aún más la búsqueda, principalmente debido a que el vector representativo contenia muchas posiciones en 0, lo que hace que muchos documentos fueran relativamente similares o en términos de distancia, que muchos estuvieran relativamente a distancias similares.

Los inconvenientes anteriores, provocan que este espacio sea un medio dificil para realizar búsquedas, lo que contráriamente proporciona, una excelente oportunidad para experimentar sobre él.

3. Estructuras Métricas

Las estructuras métricas utilizadas en el presente trabajo corresponden a *GNAT* [4] y *EGNAT* [17]. Las estructuras son basadas en clustering y son del tipo árbol. La elección de estas estructuras es debido a su buen desempeño en espacios métricos de alta dimensión. Dichas estructuras son parecidas en la forma y presentan características similares en sus procesos de búsqueda.

Otras características que determinaron el uso de estas estructuras, es que se tienen versiones estables de las mismas, y que fueron implementadas para memoria secundaria, siendo el egnat diseñada especificamente para esto, además de dinámica. Esto permite que los experimentos puedan tener una plataforma que soporte una aplicación real.

3.1. *GNAT*

Durante la construcción, el *gnat* (*Geometric Near-neighbor Access Tree*) selecciona k puntos clave para particionar el espacio, $\{c_1, c_2, \ldots, c_k\}$ denominados centros. Cada punto restante es asignado al centro más cercano, definiéndose así el subárbol de influencia. Cada subárbol es particionado recursivamente. Durante este proceso se van construyendo tablas de rangos entre cada par de centros (c_i, c_j), almacenando las distancias mínima y máxima entre c_i y el conjunto de objetos asociados ac_j, denominado D_{c_j}.

En la figura 1 se muestra un ejemplo de construcción del primer nivel de un gnat con k=*4*, también se muestra la tabla de rangos que debe ser almacenada para cada centro c_i. En este ejemplo se insertaron los puntos en orden al valor numérico que tienen.

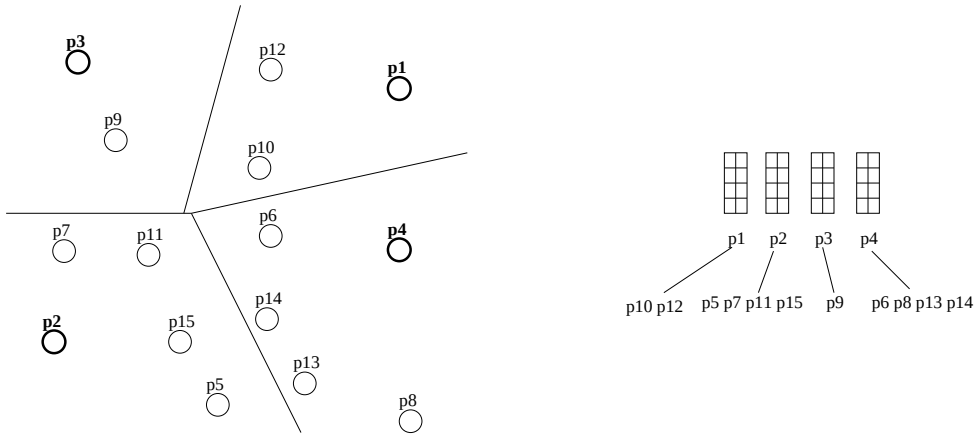


Figura 1. Partición del espacio, representación de subplanos.

Durante la búsqueda se descartan subplanos completos comparando la consulta q contra alguno de los centros, por ejemplo c_0 y si $range(c_0, q) \cap range(c_0, D_{c_x}) = \emptyset$, entonces se puede descartar completamente el subárbol asociado a x (conjunto D_{c_x}).

3.2. *Evolutionary GNAT* (*EGNAT*)

El *gnat evolutivo* es una versión dinámica y optimizada para memoria secundaria del gnat. Permite realizar eliminaciones en línea, disminuye el tamaño de la estructura en disco y reduce los accesos a disco, entre otros.

El egnat es básicamente un árbol donde cada nodo nace como un bucket, sin información alguna respecto de los datos. Al llenar una página de disco, el nodo bucket evoluciona a un nodo gnat, con todas las propiedades de éste, y del cual cuelgan hijos del tipo bucket. Adicionalmente, los objetos ubicados en los bucket guardan la distancia al padre o centro del cluster. El proceso se repite recursivamente sobre los hijos bucket.

Durante la búsqueda se utilizan la propiedades del gnat y también la información guardada para los objetos en los bucket, distancia al padre, una propiedad de las estructuras basadas en pivotes.

4. Paralelización de Estructuras Métricas

El motor de búsqueda se implementó sobre un cluster de PCs, donde se realizaron pruebas utilizando 2, 4 y 8 máquinas en el cluster para tener una visión de la disminución de los costos de procesamiento. Cada máquina del Cluster está compuesta por un procesador Intel Core 2 Duo de 2,2 GHz, memoria RAM de 1Gbytes, discos duros de 7200rpm SATA II.

El contexto común de los experimentos es que existe una máquina *broker* o maestro, ésta se encarga, inicialmente, en el proceso de construcción de distribuir los datos y posteriormente, durante la búsqueda de replicar en cada procesador las consultas y recolectar los resultados.

Se considera que el sistema trabaja bajo régimen estacionario, por lo que la fase de construcción no ha sido presenta en este trabajo.

4.1. Estrategia de Índice Local

Esta estrategia consiste en que cada máquina tiene un árbol independiente y distinto a los demás y cada uno hace el proceso de búsqueda solamente en su estructura local.

Para esto el broker distribuye los datos en cada procesador. Posteriormente, cada procesador construye localmente su estructura datos.

Durante los procesos de búsqueda, el broker hace difusión de las consultas, tomando cada una y enviandola a cada procesador. Lo presentado en este artículo corresponde a un superstep por consulta, por lo que despues de procesada una consulta, los resultados son recolectador por el broker.

Implementaciones distintas al de índice local han tenido comportamientos menos eficientes, éste es el caso de estructuras con índices globales donde se distribuyen los subárboles en el cluster e índice globales que realizan multiplexión de los nodos, es decir, para un árbol se distribuyen sus nodos dentro del cluster. Esto ocurre principalmente debido a que con BSP se paga un costo de comunicación durante la sincronización de procesos, por lo que para índices locales, la sincronización es mucho menor [16].

4.2. Resultados Preliminares

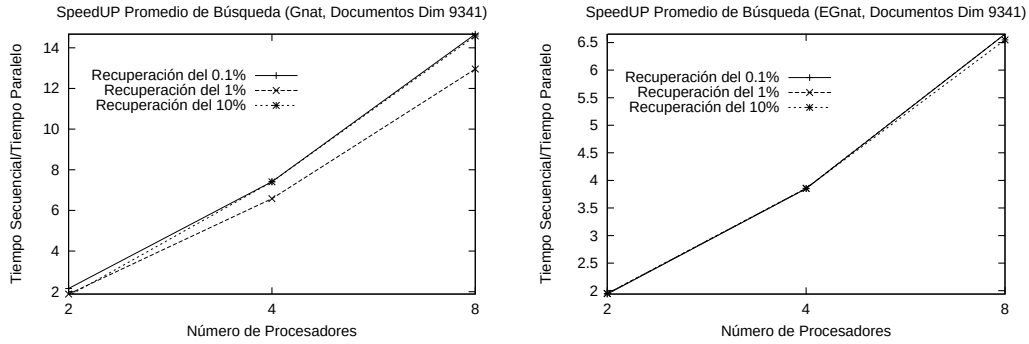
Los experimentos se realizaron sobre un espacio métrico de documentos, cuya cantidad de objetos fue cercana a los 10,000. Cada documento fue preprocesado para obtener un vector representativo de 9,341 componentes (un punto en un espacio de 9,341 dimensiones). El procesamiento de los documentos fue hecha según lo explicado en la sección 2.2.

Para la construcción de la estrutura, se utiliza un 90 % de la base de datos, el restante 10 % es utilizado como consulta.

Los resultados obtenidos en esta implementación paralela son evaluados en términos de *speed-up*. *Speed-up* se define como la proporción de tiempo que toma solucionar un problema sobre un procesador y el tiempo requerido para solucionar el mismo problema sobre un computador paralelo con p procesadores.

Los gráficos de la figura 2 muestran el speed-up para las estructuras gnat e egnat. Los rangos de búsqueda que aparecen en ambos gráficos corresponden a los rangos que, dada un consulta, permiten recuperar el 0.1 %, 1 % y el 10 % de la base de datos. Los

rangos fueron calculados experimentalmente y su objetivo es poder realizar análisis del comportamiento de las estructuras.



(a) gnat (b) egnat

Figura 2. Gráficos informativos para la búsqueda sobre el espacio de documentos.

Llama la atención que, si bien era de esperar, hubo una considerable disminución de los costos en tiempo. Para el gráfico 2(a), el desempeño en términos de speed-up es mucho mejor que para la estructura egnat. El comportamiento de esta curva es superlineal, es decir, para 4 y 8 procesadores, la estructura se comporta mucho mejor que 4 y 8 veces respectivamente, lo que no ocurre en el gráfico del egnat (2(b)).

Si se observa la figura 3, ésta muestra la cantidad de evaluaciones de distancia para cada estructura para el proceso de recuperar un 10 % de la base de datos. En este gráfico se muestra que ambas estructuras tienen un comportamiento muy parecido, lo que igualmente sucede al recuperar el 0.1 % y 1 %. Lo anterior indica que si bien en términos de evaluaciones de distancia son similares, en términos de tiempo no. El comportamiento de la estructura gnat mejora en función del aumento de procesadores, mucho más que el egnat. Sin embargo, esto se debe principalmente que si bien la primera estructura fue implementada en una versión para memoria secundaria, esta no ha sido diseñada ni optimizada para tal efecto. Al haber menos procesadores, requiere de mayor cantidad de accesos a disco, lo cual no hace en forma eficiente, por lo que su desempeño aumenta considerablemente al disponer de mayor cantidad de RAM.

5. Conclusiones

El trabajo presentado en el presente artículo, corresponde a la fase inicial de un proyecto de implementación de un *Digesto Digital Institucional* para búsqueda de documentos por Similitud. La información presentada demuestra que el proyecto es viable de realizar y que los resultados son prometedores.

Se considera que el aporte más relevante del presente artículo es presentar un trabajo de investigación orientado al desarrollo de una aplicación real, que reune dos áreas de investigación. También se considera un aporte, brindar la posibilidad de aumentar la complejidad de las búsquedas en documentos de texto y entregar versiones paralelas de dos estruturas métricas.

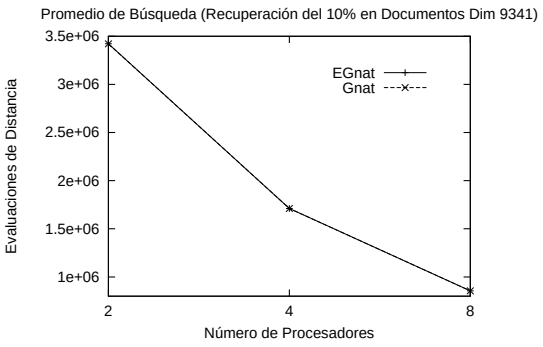


Figura 3. Cálculos de evaluaciones de distancia para las estructuras *gnat* y *egnat* para el espacio de documentos reales..

La versión original del Digesto implementado por los autores [10], básicamente almacenaba el texto de los documentos en tablas almacenadas en un servidor de bases de datos y distribuidas en un Cluster de PCs. El proceso de búsqueda consistía en buscar palabras utilizando un lenguaje de consulta. Con la nueva propuesta, se agrega la posibilidad de realizar búsquedas por similaridad o aproximada de documentos. Lo anterior conlleva a proveer de una herramienta util para los procesos de recuperación de información y finalmente a la toma de decisiones en la institución.

El trabajo que se está realizando en la actualidad, es la utilización de otras estructuras que permitan aumentar el rendimiento de las búsquedas. Los primeros experimentos nos han indicados que algunas estructuras definitivamente no responden adecuadamente a este tipo de espacio, es el caso de una version del *SSSTree* [5] y otra del *Spaghettis* [7].

Entre los trabajos futuros que realizar, se encuentran, buscar otras funciones de distancia mas adecuada para este tipo de vectores, realizar ajustes sobre la función de distancia, evaluar la calidad de las búsquedas y resultados obtenidos, experimentar con otras alternativas de paralelización de estruturas métricas, entre otras.

Referencias

1. R. Baeza-Yates, W. Cunto, U. Manber, and S. Wu. Proximity matching using fixedqueries trees. In *5th Combinatorial Pattern Matching (CPM'94)*, LNCS 807, pages 198–212, 1994.
2. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley, 1999.
3. Rob H. Bisseling. *Parallel Scientific Computation: A Structured Approach using BSP and MPI.* Oxford University Press, 2004.
4. Sergei Brin. Near neighbor search in large metric spaces. In *the 21st VLDB Conference*, pages 574–584. Morgan Kaufmann Publishers, 1995.
5. Nieves R. Brisaboa, Oscar Pedreira, Diego Seco, Roberto Solar, and Roberto Uribe. Clustering-based similarity search in metric spaces with sparse spatial centers. In *SOFSEM 2008: 34rd Conference on Current Trends in Theory and Practice of Computer Science*, volume 4910 of *Lecture Notes in Computer Science*, pages 186–197, Novy Smokovec, High Tatras, Slovakia, January, 19-25 2008. Springer.
6. W. Burkhard and R. Keller. Some approaches to best-match file searching. *Communication of ACM*, 16(4):230–236, 1973.
7. E. Chavéz, J. Marroquín, and R. Baeza-Yates. Spaghettis: An array based algorithm for similarity queries in metric spaces. In *6th International Symposium on String Processing and Information Retrieval (SPIRE'99)*, pages 38–46. IEEE CS Press, 1999.
8. Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José L. Marroquín. Searching in metric spaces. In *ACM Computing Surveys*, pages 33(3):273–321, September 2001.
9. P. Ciaccia, M. Patella, and P. Zezula. M-tree : An efficient access method for similarity search in metric spaces. In *the 23st International Conference on VLDB*, pages 426–435, 1997.

10. Esteban Gesto, Daniel Laguia, Natalia Trejo, Osiris Sofia, and Jose Canumán. Implementación de un motor de búsquedas paralelo con bsp. In *32a Conferencia Latinoamericana de informática*, Santiago de Chile - Chile, Agosto 2006. CLEI 2006.
11. Gonzalo Navarro. Searching in metric spaces by spatial approximation. *The Very Large Databases Journal (VLDBJ)*, 11(1):28–46, 2002.
12. D.B. Skillicorn, J.M.D. Hill, and W.F. McColl. Questions and answers about BSP. Technical Report PRG-TR-15-96, Computing Laboratory, Oxford University, 1996. Also in *Journal of Scientific Programming*, V.6 N.3, 1997.
13. Wijnand Suijlen. *Implementing BSPonMPI 0.1.* Lessons Learned & Results, 2006.
14. Caetano Traina, Agma Traina, Bernhard Seeger, and Christos Faloutsos. Slim-trees: High performance metric trees minimizing overlap between nodes. In *VII International Conference on Extending Database Technology*, pages 51–61, 2000.
15. J. Uhlmann. Satisfying general proximity/similarity queries with metric trees. In *Information Processing Letters*, pages 40:175–179, 1991.
16. Roberto Uribe and Ricardo Barrientos. Estrategias de paralelización el egnat. In *XXXII Conferencia Latinoamericana de Estudios en Informática (CLEI2006)*, Santiago, Chile, 2006.
17. Roberto Uribe-Paredes. Manipulación de estructuras métricas en memoria secundaria. Master's thesis, Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile, Santiago, Chile, Abril 2005.
18. L.G. Valiant. A bridging model for parallel computation. *Comm. ACM*, 33:103–111, Aug. 1990.
19. P. Yianilos. Data structures and algoritms for nearest neighbor search in general metric spaces. In *4th ACM-SIAM Symposium on Discrete Algorithms (SODA'93)*, pages 311–321, 1993.