Búsquedas por Rango sobre Plataformas GPU en Espacios Métricos

Osiris SOFIA[1], Jacobo SALVADOR[1], Eder DOS SANTOS[1]
Roberto URIBE PAREDES[2]
1: Unidad Académica Rio Gallegos, Universidad Nacional de La Patagonia Austral.
2: Departamento de Ingeniería en Computación, Universidad de Magallanes, Chile.
sistemasuarg@gmail.com

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. Actualmente es un tema de gran interés para la comunidad científica debido a sus múltiples campos de aplicación, como reconocimiento de patrones, recuperación de la información, bases de datos multimedia, entre otros. La búsqueda por similitud o en proximidad se modela matemáticamente a través de un espacio métrico, en el cual un objeto es representado como una caja negra, donde la única información disponible es la función de distancia de este objeto a los otros.

En general, el cálculo de la distancia es costoso, por ello el objetivo es reducir la cantidad de evaluaciones de distancia necesarias para resolver la consulta. Para esto se han desarrollado numerosas estructuras métricas, que realizan un preprocesamiento de los datos a fin de disminuir las evaluaciones de distancia al momento de la búsqueda.

En la actualidad, la necesidad de procesar grandes volúmenes de datos hace poco factible la utilización de una estructura en aplicaciones reales si ésta no considera la utilización de sistemas de altas prestaciones en entornos de procesamiento paralelo. Existen una serie de tecnologías para realizar implementaciones paralelas, siendo una de las más nuevas, las plataformas basadas en GPU / Multi-GPU, que son interesantes debido a la cantidad de procesadores y los bajos costes involucrados.

Palabras clave: *Búsquedas por similitud, Espacios Métricos, Paralelismo, GPU, CUDA, Bases de Datos.*

CONTEXTO

Este trabajo está enmarcado en el proyecto 29/A274-1 de la Unidad Académica Río Gallegos de la Universidad Nacional de la Patagonia Austral.

1. INTRODUCCION

La búsqueda de objetos similares sobre un gran conjunto de datos se ha convertido en una línea de investigación de gran interés. Aplicaciones de estas técnicas pueden ser encontradas en reconocimiento de voz e imagen, en problemas de minería de datos, detección de plagios y muchas otras.

En general, se utilizan diversas estructuras de datos con vistas a mejorar la eficiencia en términos de los cálculos de distancia, comparados con la búsqueda secuencial (algoritmo *fuerza bruta*) dentro la base de datos.

Por otro lado, se busca reducir los costos en términos de tiempo mediante el procesamiento en paralelo de grandes volúmenes de datos [12]. Existe una variada gama de plataformas paralelas, siendo una de las más recientes aquella basada en tarjetas gráficas GPU (*Graphic Processor Units*). En este sentido, estas nuevas tecnologías permiten un alto nivel de paralelismo a un muy bajo coste.

1.1 Búsqueda por Similitud

La similitud se modeliza en muchos casos a través de un espacio métrico, y la búsqueda de objetos más similares a través de una búsqueda por rango o de k vecinos más cercanos. Un espacio métrico es un conjunto X con una función de distancia $d : \mathrm{X}^2 \rightarrow \Re$, tal que $\forall x, y, z \in \mathrm{X}$, se deben cumplir las propiedades de: positividad $(d(x, y) \geq 0 \; and \; d(x, y) = 0 \; ssi \; x = y)$, simetría $(d(x, y) = d(y, x))$ y desigualdad triangular $((d(x, y) + d(y, z)) \geq d(x, z))$.

Sobre un espacio métrico (X,d) y un conjunto de datos finito $Y \subseteq X$, se pueden realizar una serie de consultas. La consulta básica es la *consulta por rango*. Sea una consulta $x \in X$, y un rango $r \in \Re$. La consulta de rango alrededor de x con rango r es el conjunto de puntos $y \in Y$, tal que $d(x,y) \leq r$. Un segundo tipo de consulta, que puede construirse usando la consulta por rango, es los k *vecinos más cercanos*. Sea 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 $|A| = k$ y no existe un objeto $y \in A$ que $d(x,y)$ sea menor a la distancia de algún objeto de A a x.

1.2 Estructuras Métricas Basadas en Pivotes

Existe una serie de estructuras de datos métricas e índices cuyo objetivo es reducir las evaluaciones

de distancias durante la búsqueda, y con ello disminuir el tiempo de procesamiento. Algunas de las estructuras basan la búsqueda en pivotes y otras en *clustering*.

Los algoritmos basados en *clustering* dividen el espacio en áreas o planos, donde cada área tiene un centro y se almacena alguna información sobre el área que permita descartarla completa solo con comparar la consulta con su centro. Ejemplos de estos son BST, GHT, M-Tree, GNAT, EGNAT y muchos otros [1].

En los métodos basados en pivotes, se selecciona un conjunto de pivotes y se precalculan las distancias entre los pivotes y todos los elementos de la base de datos. Los pivotes sirven para descartar objetos durante la búsqueda utilizando la desigualdad triangular. Algunas estructuras de este tipo son: LAESA [2], FQT y sus variantes [3], Spaghettis y sus variantes [4], FQA [5], SSS-Index [6] y otros [10].

Para la construcción de una estructura basada en pivotes, se selecciona un conjunto de pivotes *p1, ..., pk*, los cuales pueden o no pertenecer a la base de datos a indexar.

Existen estructuras de tipo árbol y de tipo arreglo, estas últimas son más adecuadas para ser implementadas en plataformas basadas en GPU [7].

Una estructura métrica genérica (*Generic Metric Structure*: GMS) puede considerarse como una tabla de distancias entre los pivotes y todos los elementos de la base de datos. Es decir, cada celda almacena la distancia $d(yi, pj)$, siendo *yi* un elemento de la base de datos.

Durante la búsqueda por rango sobre esta estructura dada una consulta *q* y un rango *r*, el algoritmo sería:

1. Para cada consulta *q*, se calcula la distancia entre *q* y todos los pivotes *p1, ... , pk*. Con esto se obtienen *k* intervalos de la forma [*a1, b1*], ..., [*ak, bk*], donde *ai = d(pi, q) − r* y *bi = d(pi, q) + r*.
2. Los objetos, representados en la estructura por sus distancias a los pivotes, son candidatos a la consulta *q* si todas sus distancias están dentro de todos los intervalos.
3. Para cada candidato *y*, se calcula la distancia *d(q, y)*, y si *d(q, y)* <= *r*, entonces el objeto y es solución a la consulta *q*.

La Figura 1 representa una estructura de datos métrica genérica (GMS) construida con 4 pivotes.

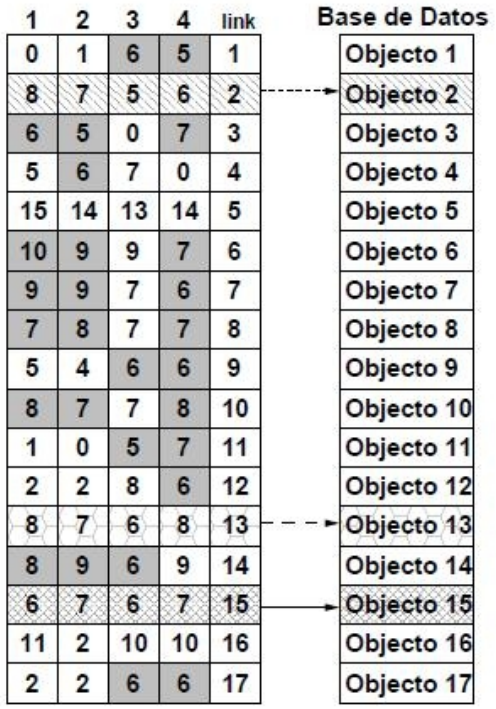

Figura 1. Búsqueda sobre una estructura métrica genérica (GMS) de 4 pivotes. Para cada consulta *q* con distancias a los pivotes *d(q,pi) = {8, 7, 4, 6}* y un rango de búsqueda *r = 2*, define los siguientes intervalos *{(6, 10), (5, 9), (2, 6), (4, 8)}*. Las celdas marcadas en gris oscuro son aquellas que están dentro del intervalo de búsqueda. Las celdas tachadas con líneas son los objetos candidatos (2, 13 y 15), que serán evaluados directamente con la consulta.

1.3 Unidades de Procesamiento Gráfico

Actualmente las unidades de procesamiento gráfico (GPU) disponen de un alto número de *cores*[1] con un alto ancho de banda. Este tipo de dispositivos permite aumentar la capacidad de procesamiento respecto de las CPU [8] y arquitecturas *multi-core* [19]. Una tendencia denominada *Computación de Propósito General sobre GPU* o GPGPU ha orientado la utilización de GPU sobre nuevos tipos de aplicaciones.

En general, las GPU utilizan todos sus recursos en tener la mayor cantidad de elementos de procesamiento y gran ancho de banda con memoria, lo que posibilita una potencia de cálculo superior al de las CPUs. Así, uno de los objetivos de las GPU es aumentar la capacidad de procesamiento explotando el paralelismo a nivel de datos para problemas paralelos o que se puedan paralelizar.

Existen nuevos lenguajes o extensiones para los lenguajes de alto nivel propuestos por algunos fabricantes; un ejemplo es CUDA [11]. El modelo de Programación CUDA de NVIDIA considera a la

[1]En la actualidad, se encuentran disponibles modelos de GPU con más de 2600 *cores*.

GPU como un dispositivo capaz de ejecutar un alto número de hilos en paralelo sobre la GPU (*device*) y su DRAM (*device memory*). CUDA incluye herramientas de desarrollo de software C/C++ y un mecanismo de abstracción que oculta los detalles de hardware al desarrollador.

En CUDA los hilos son organizados en bloques del mismo tamaño y los cálculos son distribuidos en una malla o *grid* de bloques. Estos hilos ejecutan el código de la GPU, denominado *kernel*. Un *kernel* CUDA ejecuta un código secuencial en un gran número de hilos en paralelo.Los hilos en un bloque pueden trabajar juntos eficientemente, intercambiando datos localmente en una memoria compartida y sincronizando a baja latencia en la ejecución mediante barreras de sincronización. Por el contrario, los hilos ubicados en diferentes bloques pueden compartir datos, solamente accediendo a la memoria global de la GPU o *device memory*, que es una memoria de más alta latencia.

1.4 GPU y espacios métricos

Existe una importante cantidad de índices o estructuras métricas para búsquedas sobre espacios métricos. Sin embargo, existe una serie de características poco comunes en estas estructuras que hacen difícil su implementación directa para aplicaciones reales. La primera tiene relación con capacidades dinámicas, la mayoría de estas estructuras deben ser reconstruidas si existen nuevos objetos que indexar o si hubiese que eliminar objetos de la base de datos.

Otra característica, aún menos frecuente, está relacionada con estructuras que permitan la manipulación eficiente de los datos en memoria secundaria, donde deben ser considerados parámetros de costo adicional como son el número de accesos a disco y el tamaño del índice entre otros. En si, dichas estructuras han sido diseñadas como prototipos, por lo que su utilización en aplicaciones reales no ha sido probada, mas aún considerando que en aplicaciones reales los objetos son de gran tamaño y/o la base de datos es considerablemente grande, donde es posible pensar que sólo un bajo porcentaje de datos podría entrar en memoria principal. En general, las estructuras métricas no consideran la jerarquía de memoria, por ello, resulta relevante considerar este aspecto para poder obtener un mayor rendimiento y eficiencia ante la posibilidad de utilizar nuevas tecnologías, como es el caso de las GPU que poseen distintos niveles de memoria dentro de su sistema.

Otra característica es la relación con el procesamiento masivo de datos. Resulta poco factible la utilización de una estructura en aplicaciones reales si ésta no considera su implementación en entornos paralelos. En la paralelización de estructuras métricas deben considerarse, entre otros aspectos, la distribución adecuada de la base de datos sobre el entorno, por ejemplo en un cluster de PCs; la paralelización de los métodos de búsqueda; eficiencia en la comunicación entre los procesadores; etc.

Desde el punto de vista de las GPU, esencialmente se han abordado soluciones para las consultas kNN sin utilizar estructuras de datos, es decir, se utilizan para paralelizar búsquedas exhaustivas (fuerza bruta) [13, 14, 15]. Todos estos trabajos abordan el problema desde el supuesto de que los elementos de la base de datos poseen una gran dimensión (con una gran dimensión intrínseca), lo cual implica poder descartar muy pocos objetos usando desigualdad triangular, y en donde la indización es ineficiente frente a la búsqueda secuencial.

La paralelización de estructuras métricas sobre GPU y en clusters de multicores son áreas de investigación poco exploradas. En este contexto, se conocen dos grupos de investigación que están desarrollando trabajados en torno a estructuras métricas sobre plataformas basadas en GPU. El primero de estos grupos, de la Universidad Complutense de Madrid, ha enfocado sus estudios a dos estructura métricas, la *Lista de Clusters* y *SSS-Index*, y ha presentado diversos trabajos en propuestas para kNN y consultas por rango [15, 16]. En el segundo grupo, de la Universidad de Castilla La Mancha, participa el profesor Roberto Uribe-Paredes, las líneas de investigación abordadas por este grupo están orientadas a desarrollar y potenciar, sobre un ambiente híbrido, la estructura genérica presentada en la sección 1.2 [7, 10, 17, 18].

2. LINEAS DE INVESTIGACION y DESARROLLO

De acuerdo a los antecedentes disponibles y los trabajos previos realizados, se han planteado las siguientes líneas de investigación:

- Implementación de distintas soluciones sobre espacios disímiles, tales como histogramas de colores, bases de datos de palabras, vectores de coordenadas, vectores de gauss, imágenes NASA y otros.
- Forma óptima de distribución de los datos, tanto de la base de datos como de las consultas, en ambientes de GPU y/o Multi-GPU, en las memorias administradas por las CPUs y las memorias dedicadas de las GPU.
- Escalabilidad de las implementaciones de acuerdo al tamaño de la Base de Datos, y consecuentemente el almacenamiento y procesamiento de BD y consultas en memoria secundaria.

3. RESULTADOS OBTENIDOS / ESPERADOS

Basados en las experiencias de los integrantes del grupo de investigación, se realizarán experimentos para determinar espacios y dimensiones donde es realmente conveniente utilizar estructuras métricas sobre plataformas basadas en GPU. Se espera confirmar o refutar la hipótesis de la conveniencia de utilización de fuerza bruta o a lo sumo estructuras genéricas basadas en pivotes en ambientes GPU, extendiendo las pruebas de laboratorio a un amplio conjunto de ambientes y espacios.

Como evaluación experimental, actualmente se está trabajando con espacios métricos en un entorno *multicore* y un entorno GPU utilizando la estructura GMS basada en pivotes.

Para los casos de estudio se han considerado dos conjuntos de datos: el primero es un diccionario con palabras en español con 86.061 elementos. La función de distancia utilizada es la *distancia de edición*, definida como el mínimo número de inserciones, eliminaciones o sustituciones de caracteres necesarios para que una palabra sea convertida en otra. Para este espacio, los rangos de búsqueda son discretos y con rangos entre r = 1 y r = 4. El segundo espacio corresponde a un conjunto de 112.682 histogramas de colores (vectores de dimensión 112) de una base de datos de imágenes. Para este espacio se seleccionó la distancia euclidiana. Los radios utilizados son aquellos que permiten recuperar el 0.01%, 0.1% y 1% de la base de datos. Para ambos espacios se construye la estructura con el 90% de los datos, dejando el restante 10% para consultas.

Se han elegido estas condiciones para la realización de las pruebas ya que son las típicamente usadas para este tipo de experimentos. Para cada rutina, se ejecuta la aplicación 4 veces y se obtiene un promedio.

La plataforma utilizada corresponde a un Intel® Core™ i7-2600 CPU @3.40GHz de 4 núcleos y soporte a Hyper-Threading, 08GB de memoria principal y dos tarjetas Nvidia EVGA DDR5 de 384 cores CUDA y 1 GB de memoria global cada, usando CUDA SDK v3.2 [14]. La codificación se ha realizado utilizando el lenguaje C (gcc 4.3.4) y librerías OpenMP. La Figura 2 nos permite tener una visión inicial respecto de la utilización de una plataforma multicore y una plataforma GPU en comparación con su procesamiento secuencial, sobre el espacio de histogramas mencionado anteriormente.

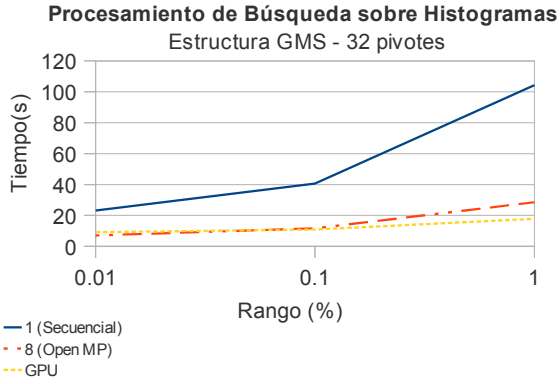

Figura 2. Tiempos de ejecución absolutos para las versiones secuencial (1 *core*), *multi-core* de 8 núcleos (4 núcleos con *Hyper-Threading*) y para la versión GPU. Histogramas de colores de 112.682 vectores.

Como era de esperar, puede apreciarse en la Figura 2 la enorme diferencia al utilizar una plataforma *multicore* o *manycore* sobre una secuencial, observándose una disminución considerable del tiempo de procesamiento. Asimismo, se puede observar un rendimiento levemente mejor de la plataforma GPU a medida en que se aumenta el radio de búsqueda. Similar comportamiento se ha observado en los *tests* realizados sobre la base de datos de palabras. Ello en virtud de que la cantidad de evaluaciones sobre objetos candidatos aumenta notoriamente al aumentar el rango o radio de búsqueda. La cantidad de hilos disponibles para resolver las consultas compensa el tiempo aplicado en la alocación de los recursos en la memoria de la GPU.

Por otra parte, debido al avance tecnológico y la reducción de los costos, se prentede integrar en un solo equipo múltiples CPU y múltiples GPU [9, 16, 17], de manera que resulte posible extender el paralelismo a todos estos ambientes en forma simultánea. Ello crea la posibilidad de evaluar el rendimiento de las distintas estructuras para ambientes de múltiples CPU / GPU. Si bien se han realizado pruebas de laboratorio falta aún definir la mejor estrategia de distribución de datos tanto para el procesamiento como en el almacenamiento en memoria volátil de la CPU / GPU.

Si bien hasta el momento las pruebas de laboratorio se han realizado con bases de datos de importante tamaño, queda aún por validar las conclusiones obtenidas en cuanto a la escalabilidad de las mismas. Las pruebas de escalabilidad llevarán necesariamente a investigar sobre la distribución de los datos, consultas y estructuras métricas entre la memoria principal y la secundaria, debido a que en algún momento no será posible almacenar la base de datos completa en memoria volátil.

4. FORMACION DE RECURSOS HUMANOS

El investigador Roberto Uribe-Paredes, realizó su tesis de Magíster en el área de espacios métricos. Actualmente desarrolla su tesis de doctorado en la misma área desarrollando estructuras sobre entornos basados en GPU, en la Universidad de Castilla La Mancha, España, donde espera rendir su examen el año corriente. Este investigador ha apoyado al grupo en la fase inicial realizando cursos a distancia sobre espacios métricos y sobre tecnologías basadas en GPU. Además, a través de este investigador se están desarrollando lazos de trabajo con los grupos de investigación de la Universidad de Magallanes, Chile y de Castilla La Mancha, España.

Un investigador (Dos Santos) se encuentra cursando la "Especialización en Management Tecnológico" de la Universidad Nacional de la Patagonia Austral, y actualmente desarrolla su tesis titulada "Proyecto de Educación Tecnológica en el Programa de Educación a Distancia de la UNPA".

Un investigador (Salvador) ha obtenido recientemente su título de doctorado en Ingeniería, mención procesamiento de señales e imágenes en la Universidad tecnológica Nacional, regional Buenos Aires, con la tesis titulada "Estudio del comportamiento de la capa de ozono y la radiación UV en la Patagonia Austral y su proyección a la comunidad".

5. AGRADECIMIENTOS

El presente trabajo fue parcialmente financiado por la Universidad Nacional de la Patagonia Austral, Santa Cruz, Argentina.

6. BIBLIOGRAFIA

[1] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José L. Marroquín. "Searching in metric spaces". In ACM Computing Surveys, 2001, pp. 33(3):273–321.

[2] L. Micó, J. Oncina, and E. Vidal. "A new version of the nearest-neighbor approximating and eliminating search (aesa) with linear preprocessing-time and memory requirements". Pattern Recognition Letters, vol. 15, pp. 9–17, 1994.

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

[4] E. Chávez, 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). IEEE CS Press, 1999, pp. 38–46.

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

[6] Oscar Pedreira and Nieves R. Brisaboa. "Spatial selection of sparse pivots for similarity search in metric spaces". In 33rd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2007), Harrachov, Czech Republic, 2007, vol. 4362 of LNCS, pp. 434–445, Springer.

[7] R. Uribe-Paredes, P. Valero-Lara, E. Arias, J.L. Sanchez, and D. Cazorla. "Similarity search implementations for multi-core and many-core processors". In International Conference on High Performance Computing and Simulation (HPCS), 2011, pp. 656–663.

[8] Wu-Feng and Dinesh Manocha, "High-performance computing using accelerators," Parallel Computing, vol. 33, pp. 645–647, 2007.

[9] Roberto Uribe-Paredes, Enrique Arias, Diego Cazorla, José Luis Sánchez. "Una estructura Métrica Genérica para Búsquedas por Rango sobre una Plataforma Multi-GPU". XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD2012). Sept. 2012, Almería, España.

[10] Roberto Uribe-Paredes, Diego Cazorla, José L. Sánchez, and Enrique Arias. "A comparative study of different metric structures: Thinking on gpu implementations". In *International Conference of Computational Statistics and Data Engineering (ICCSDE'12)*, London, England, July 2012.

[11] NVIDIA CUDA C Programming Guide, Version 4.0, NVIDIA, 2011, http://developer.nvidia.com/object/gpucomputing.html

[12] Ananth Grama, George Karypis, Vipin Kumar, and Anshul Gupta. Introduction to Parallel Computing (2nd Edition). Addison Wesley, 2 edition, 2003.

[13] Quansheng Kuang and Lei Zhao. "A practical GPU based kNN algorithm". International Symposium on Computer Science and Computational Technology (ISCSCT), pp. 151–155, 2009.

[14] Vincent Garcia, Eric Debreuve, and Michel Barlaud. "Fast k nearest neighbor search using GPU". Computer Vision and Pattern Recognition Workshop, vol. 0, pp. 1–6, 2008.

[15] R.J. Barrientos, J.I. Gómez, C. Tenllado, M. Prieto, and M. Marin. "kNN query processing in metric spaces using gpus". in 17th International European Conference on Parallel and Distributed Computing (Euro-Par 2011), Bordeaux, France, 2011, vol. 6852 of LNCS, pp. 380–392, Springer.

[16] R.J. Barrientos, J.I. Gómez, C. Tenllado, M. Prieto, and M. Marin. "Range query processing in a multi-GPU enviroment". In 10th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA 2012), Madrid, Spain, July 2012.

[17] Roberto Uribe-Paredes, Enrique Arias, José L. Sánchez, and Diego Cazorla. "Improving the Performance for the Range Search on Metric Spaces using a Multi-GPU Platform". To appear: 23rd International Conference on Database and Expert Systems Applications (DEXA 2012). Vienna, Austria, Sept. 2012.

[18] Roberto Uribe-Paredes, Diego Cazorla, Enrique Arias and José L. Sánchez. "Acelerando la Búsqueda por Rango con un Sistema Híbrido de Memoria Compartida". To appear: Actas XXIII Jornadas de Paralelismo (JP2012). Sept. 2012, El España.

[19] Roberto Uribe-Paredes, Pedro Valero-Lara, Enrique Arias, José L. Sánchez and Diego Cazorla. "Similarity search implementations for multi-core and many-core processors ". In: 2011 International Conference on High Performance Computing and Simulation (HPCS), pp. 656–663 (July 2011). Istanbul, Turkey.