Assessing metric structures on GPGPU environments

DOS SANTOS, Eder[1], SOFIA, Albert A. O.[1],URIBE PAREDES, Roberto[2]
[1]Unidad Académica Río Gallegos - Universidad Nacional de la Patagonia Austral
Lisandro La Torre 1070 – 0054-2966-442313 – Río Gallegos – Santa Cruz – Argentina
[2]Departamento de Ingeniería en Computación, Universidad de Magallanes, Chile.
esantos@unpa.edu.ar, sistemasuarg@gmail.com, roberto.uribeparedes@gmail.com

Abstract. Similarity search consists on retrieving objects within a database that are similar or relevant to a particular query. It is a topic of great interest to scientific community because of its many fields of application, such as searching for words and images on the World Wide Web, pattern recognition, detection of plagiarism, multimedia databases, among others. It is modeled through metric spaces, in which objects are represented in a black-box that contains only the distance between objects; calculating the distance function is costly and search systems operate at a high query rate. Metrical structures have been developed to optimize this process; such structures work as indexes and preprocess data to decrease the distance evaluations during the search. Processing large volumes of data makes unfeasible the use of such structures without using parallel processing environments. Technologies based on multi-CPU and GPU architectures are among the most force due to its costs and performance.

Keywords: Similarity search, metric spaces, metric structures parallel processing, GPU.

1 Introduction

The search of similar objects in a large collection of stored objects in a metric database has become a most interesting problem. This kind of search can be found in different applications such as voice and image recognition, data mining, plagiarism and many others.

In general, various data structures are used in order to improve efficiency in terms of distance calculations, compared with the sequential search in the database (known as a brute force algorithm). On the other hand, parallel processing seeks to reduce costs in terms of time of processing large data volumes [1].

There are a number of technologies for parallel processing implementations. Technologies based on multi-CPU architectures, GPU and multi-GPU are among the most current. In this sense, the application of these new technologies on similarity searches in metric spaces [2][3] allow a high level of parallelism at a very low cost.

This paper focuses on similarity search and implementation of metric structures on parallel environments. Section 2 presents the state of the art on issues related to search by similarity metric structures and parallelization technologies. Comparative analysis of experiments are proposed section 3, seeking to identify the behavior of a set of

metric spaces and metric structures on processing platforms based on multicore and GPU. Finally, conclusions and suggestions for future works are in section 4.

1.1. Similarity Search in Metric Spaces

Similarity is modeled in many interesting cases through metric spaces, and the search of similar objects through range search or nearest neighbors. a metric space is a set X and a distance function$d:X2 \rightarrow R$, so that $\forall x, y, z \in X$, fulfills the properties of positiveness($d(x,y)>=0$ *and* $d(x,y)=0$ *iff* $x=y$), simetry$(d(x,y)=d(y,x))$and triangle inequality$(d(x,y)+d(y,z)>=d(x,z))$.

In a given metric space (X, d) and a finite data set $Y \subset X$, a series of queries can be made. The basic query is the **range query**, a query being an object $x \in X$ and a range $r \in R$. The query range around x with range r is the set of objects $y \in Y$ so that $d(x,y)<= r$. A second type of query that can be built using the range query is **k nearest neighbors**, a query being an object $x \in X$ and an object k. The k nearest neighbors to x are a subset A of objects Y, such that if $|A|=k$ and there is no object $y \in A$ such that $d(y,x)$ is lower than the distance of some object of A to x.

The aim of search algorithms is to minimize the number of distance evaluations performed to resolve a query. The methods for searching in metric spaces are mainly based on dividing the space using the distance to one or more selected objects. Not working with the particular characteristics of each application has the advantage of being more general, because the algorithms work with any type of object [4].

1.2. Metric data structures

There are different structures for searching metric spaces, which can use discrete or continuous distance functions. Metric space data structures can be grouped into two classes, clustering-based and pivots-based methods.

The clustering-based structures divide the space into areas, where each area has a so-called center. Some data is stored in each area, which allows easy discarding the whole area by just comparing the query with its center. Algorithms based on clustering are better suited for high-dimensional metric spaces, which is the most difficult problem in practice. Some clustering-based indexes are GNAT [5], M-Tree [6], SAT [7], Slim-Tree [8], EGNAT [9] and others [4].

In the pivots-based methods, a set of pivots are selected and the distances between the pivots and database elements are pre-calculated. When a query is made, the query distance to each pivot is calculated and the triangle inequality is used to discard the candidates. Its objective is to filter objects during a request through the use of a triangular inequality without really measuring the distance between the object under request and the discarded objects. Some pivots-based indexes are: LAESA [10], FQT and its variants [11], Spaghettis and its variants [12][13], FQA [14], SSS-Index [15] and many others [16].

A Generic Metric Structure (GMS) is an array-type structure based on pivots, which are obtained randomly. It contains the set of distances between the pivots and

the database objects. Each cell stores the distance d(yi, pj) being a database object yi. The algorithm for range search on this structure, given a query q and a range r is:

Figure 1 represents a GMS built with 4 pivots. For each query q being the distances to pivots d(q,pi) = {8, 7, 4, 6} and the search range r = 2, intervals {(6, 10), (5, 9), (2, 6), (4, 8)} are defined. Cells marked in dark gray are within the search range. The strikethrough cells are the candidate objects (2, 13 and 15), which will be evaluated directly with the query.

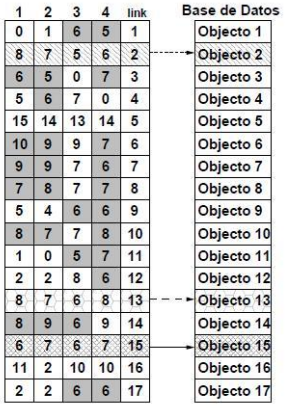

Figure 1. Search over a generic metric structure.

1.3. Parallel Processing Platforms

Currently, there is a wide variety of parallel platforms on which metric structures can be implemented. In this context, many studies have initially focused on distributed memory platforms using high level libraries such as MPI [MPI94] or PVM [PVM94] and shared memory using OpenMP [OMP07] [GKKG03] directives. In this sense, [GBMB10] proposes a strategy for organizing query processing on metric spaces in multi-core nodes. Finally, recent work has dealt newer shared memory technologies, including GPU-based platforms [16] [17] [18].

OpenMP is an application programming interface (API) for shared memory multiprocessing programming on multiple platforms. It is a specification of a set of compiler directives, library routines and environment variables that can be used for high-level parallelism in programs written in C, C ++ and Fortran, based on the fork-join execution model. It is available in many architectures, including Unix and Microsoft Windows platforms, and distributes processing to the microprocessor cores.

On the other hand, graphical processing units (GPU) have a high number of cores with high bandwidth. These devices can increase processing capacity with respect to CPU [19]. A trend called General Purpose Computing on GPU or GPGPU has guided the use of GPU on new types of applications. In this sense, the manufacturers of GPU devices have proposed new languages or extensions for high level languages. An example is CUDA, which is a software platform that allows use GPU devices for general purpose applications, with the ability to handle a large number of threads.

Finally, an application built with a hybrid parallel programming model can run on a cluster of computers using OpenMP and other technologies such as GPU or MPI, or through OpenMP extensions for distributed memory systems.

1.4. Parallelism on Metric Spaces

Metric structures have some unusual features that make it difficult to its direct implementation in real applications. The first is related to dynamic capabilities: most of these structures must be rebuilt if there are new items to index or if it is needed to remove objects from the database. Another feature, even rarer, is related with structures that enable an efficient data handling in secondary memory, which should be considered additional cost parameters such as the number of disk accesses and the index size, among others. Finally, metric structures generally do not consider the memory hierarchy; therefore, it is important to consider this aspect in order to achieve greater performance and efficiency with the ability to use technologies such as GPU, which have different memory levels in its system.

It is impractical to use metric structures in real applications if it does not consider its implementation in parallel environments. To parallelize metric structures it must be considered, among other criteria: the proper distribution of the database on the environment, eg. in a cluster of PCs; the parallelization of search methods; the efficiency in communication between processors; etc.

According to available data and previous work, it has been raised to implement different solutions on disparate spaces, such as color histograms, databases of words, coordinate vectors, Gauss vectors, NASA images and other data, in order to identify the optimal distribution of data in such spaces, both the database and the structures, under GPU and / or Multi-GPU environments, on memory managed by the CPUs and GPU dedicated memories. Finally, it aims is to assess the scalability of implementations according to the database size, and consequently the storage and processing of the Databases, and running queries on secondary memory. The following shows case studies, results and conclusions.

2. Analysis, Discussion and Results

Parallelizing metric structures on GPU and multicore clusters are little explored research areas. In this context, two research groups are developing works around metric structures on GPU-based platforms. The first group of the Complutense University of Madrid has focused his studies over two metrics tructures, List of Clusters and SSS-Index, and has submitted various proposals for kNN and queries by range**¡Error! No se encuentra el origen de la referencia.¡Error! No se encuentra el origen de la referencia.**. In the second group, of the University of Castilla La Mancha, Professor Roberto Uribe-Paredes is involved. The lines of research addressed by this group are aimed at developing and strengthening the generic structure presented above on a hybrid environment [16] [17] [20]. Finally, metric structures on a GPU are used in [2], and their results are compared with sequential versions; a metric structure is used and the results are compared with sequential and

multicore-CPU versions, showing a noticeable improvement when using this new platform.

Based on the experiences of the research group members, it has proposed to confirm or refute the convenience of using generic metric structures based on pivots in GPU environments. To do this, initially it is proposed:

- On one hand, it aims to evaluate the search performance on GMS structures compared to brute force search.
- Furthermore, the GPU performance is compared with other processing options. It seeks to determine spaces and dimensions in which is really convenient to use metric structures on GPU-based platforms.

The analysis has carried out an exhaustive set of laboratory tests on a wide range of parallel environments and metric spaces.

2.1. Study Cases

Different representative metric spaces have been selected in the experiments conducted by the research group. In order to classify those spaces, the distance function for similarity search is used. Thus, it is possible to identify two groups of metric spaces.

The first set of areas corresponds to different language dictionaries. The distance function used is the edit distance, which corresponds to the minimum number of insertions, deletions or substitutions needed for a word equal to another. Similarity ranges applied in the search vary from 1 to 4.

The second group of metric spaces contains databases vectors of different sizes, which represent different objects; they are predominantly colored histograms of different images (photos of faces, satellite images, diagrams), wherein such histograms are represented as vectors. For this second group, the Euclidean distance has been chosen as the distance, and ranges recovered 0.01%, 0.1% and 1% of similarity from the dataset used.

To perform the experiments, both databases are divided into two random sets; The first one contains 90% of the database objects, and it is used to build the metric structures; the remaining 10% is used as a set of query objects.

As a platform for experimental evaluation, the mentioned metric spaces have been used under multicore and GPU environments using the GMS structure based on different numbers of pivots and using also the brute force search algorithm. For each routine, the application runs 4 times and an average is obtained. The experiments generated for each metric space consist of a Cartesian product of the variables described below:

- Index / Structures / Algorithm: Brute Force and pivots-based GMS.
- Number of Pivots (for GMS): 1, 2, 4, 8, 16, 32, 64, 128, 256, 512 y 1024.
- Processors: 1 CPU (secuential), 4 CPU (multi-core), 8 CPU (multi-core with hyper-threading) and GPU (using GPU global and shared memory).

To quantify the experiments, a summary of the variables considered and experiments realized is shown in Table 3.

Database Types	Spaces	Parallelism Methods	Algorithms	Search Ranges	Repetitions	Total
Gauss Diagrams	5	5	12	3	4	3600
Other Vectors	3	5	12	3	4	2160
Dictionaries	10	5	12	4	4	9600
TOTAL						15360

Table 1. Summary of experiments realized.

Finally, the hardware used corresponds to an Intel® Core™ i7-2600 CPU @ 3.40GHz 4-core and Hyper-Threading support, 12GB of main memory and two EVGA Nvidia cards DDR5 384 CUDA cores and 1 GB of global memory each one. The codification of the metric structures and search algorithms has been performed using the C language (gcc 4.3.4), executed on Ubuntu Linux 12.04 LTS (Precise Pangolin); CUDA SDK v3.2 has been adopted for parallelizing GPU applications, and OpenMP library has been used for multi-CPU parallelization.

2.2. Results

Next, results of experimental evaluation are detailed. It should be noted that the experiments were performed so that it is possible make the following comparisons:

1. Efficiency of different processing options: experiments with codes designed for different processing options:
    a. Sequential: using 1 core in its maximum capacity.
    b. Multi-core: using a 4 cores processor at its maximum capacity with OpenMP.
    c. Hyper-Threading(TM): using a 4 cores processor with Hyper-Threading at its maximum capacity with OpenMP.
    d. GPU using its global memory.
    e. GPU using its shared memory.
2. GMS vs. Brute Force algorithm: first. Experiments using the brute force algorithm and GMS with different numbers of pivots (1, 2, 4, 8, 16, 32, 64, 128, 256, 512and1024) where performed in order to identify the most efficient number of pivots in each experiment.

Tests were performed over the different databases specified in the previous paragraph. Here is a selection of relevant results that guide the analysis.

2.2.1. Sequential Processing vs. Parallel Processing

This first block analysis includes the various processing options (sequential, multicore, GPU using its global memory, GPU using its shared memory) on the Brute

Force algorithm; This allows us to visualize the performance level of processing the entire database without using indexes or metric structures. A metric space of vectors (histograms with absolute size 112) and words in Spanish dictionary were used as case studies, which where adopted initially to show in a general way the performance of applications over the different distance functions described herein selection.

In order to quantify the results, processing times (in seconds) are used as a measurement unit. Then, the ratio is calculated between two results A and B, giving rise to the latency speed-up. In this first analysis, we can observe a significant difference between sequential processing and parallel processing. Upcoming it shows the processing times achieved and the corresponding speedup values.

Option	Time (s.)	SpeedUp			
		GPU	GPU SM	MC	SEC
GPU	647,30	--	*0,09*	*0,86*	5,25
GPU SM	56,48	11,46	----	9,83	60,20
MC	555,09	1,17	*0,10*	--	6,13
SEC	3400,15	*0,19*	*0,02*	*0,16*	--

Table 2. Speedup values on a search of rank 1 in a metric space of words.

The higher difference is observed with the use of shared memory (SM GPU) compared to sequential processing, which results in a speedup of 60.2, 56.2, 54.21 and 52.89 for ranks 1, 2, 3 and 4 respectively.It should be noted that those results are not enough to corroborate the efficiency of GPU processing, so more experiments were performed over different metric spaces. The results obtained in experiments on a vectors metric space are shown below.

Option	Time	SpeedUp (Range 0.01)			
		GPU	GPUSM	MC	SEC
GPU	144,16	--	*0,97*	*0,99*	3,76
GPU SM	139,84	1,03	--	1,02	3,88
MC	143,22	1,01	*0,98*	--	3,79
SEC	542,26	*0,27*	*0,26*	*0,26*	--
Option	**Time**	**SpeedUp (Range 0.1)**			
		GPU	GPUSM	MC	SEC
GPU	144,17	--	*0,97*	*0,99*	3,74
GPU SM	139,77	1,3		1,02	3,88
MC	143,16	1,01	*0,98*	--	3,76
SEC	538,92	*0,27*	*0,26*	*0,27*	--
Option	**Time**	**SpeedUp (Range 1)**			
		GPU	GPUSM	MC	SEC
GPU	144,26	--	*0,97*	*0,99*	3,72
GPU SM	139,82	1,03	--	1,03	3,83
MC	143,39	1,01	*0,98*	--	3,74
SEC	535,99	*0,27*	*0,26*	*0,27*	--

Table 3. Speedup obtained in the search for a metric space of vectors.

As shown, the speedup obtained in shared memory processing compared to sequential processing were 3.88, 3.86 and 3.83 for the search ranges 0.01%, 0.1% and 1%, respectively. In this sense, this second experiment shows more equity in processing times compared to the metric space previously used in regard to the performance of the various options for parallelism. In the former case, it is possible to

see speedup values near 11.4 (GPU shared memory vs. GPU global memory) and 9.8 (GPU shared memory vs. multicore with Hyper-Threading). In the last example, the speedup values obtained were respectively of 1.03 / 1.02, 1.03 / 1.02 and 1.03 / 1.03.

2.2.2. Metric structures vs. Brute force under GPU

The above analysis shows that parallel processing using the shared memory of the GPU has a substantially superior yield to the other techniques used under Brute Force algorithm. However, the analysis with metric structures must be made considering the costs of processing and storage of such structures in the hierarchy of memory devices, so it is necessary to make different assessments as to the size of the database metric and respective data structures, and their behavior in different amounts as to pivot and thus to structures of different sizes.

To assess the performance of metric structures under different techniques of parallelism, some experiments have been performed using structures with different numbers of pivots (1, 2, 4, 8, 16, 32, 64, 128, 256, 512 and 1024 pivots) on metric spaces with different properties. The results of experiments show that the search times on the GMS have been predominantly better than brute force processing different metric spaces using the shared memory of the GPU.

2.2.3. Impact of search range and the number of pivots on the performance of Metric Structures

The experiments reveal a variation in processing times according to the search range applied in routines that use the GMS structures, contrary to the brute force algorithm (in which processing times are uniforms for different search ranges). In this sense, it has also been possible to confirm that the processing time on metric structures tends to increase as the search range increases, as it tends to make more distance evaluations extent that the search range is increased.

In addition, experiments were performed using the GPU's shared memory on metric structures with 1, 2, 4, 8, 16, 32, 64, 128, 256, 512 and 1024 pivots, in order to visualize the behavior of the structures with different numbers of pivots. Two trends were identified: on the one hand, the decrease in performance as the search range is increased in structures with high amounts of pivots. Furthermore, the performance of structures with few pivots is better as that the search range increases.

2.2.4. Parallelism on GMS Structures

The last analysis proposed in this paper aims to evaluate the most efficient method of parallelism on generic metric structures in different metric spaces, in order to verify the convenience of using GPU metric structures. To do this, experiments on different metric spaces under various forms of parallel processing described in this paper (OpenMP, GPU GPU global memory and shared memory) and different amounts of pivots were made.

The results revealed two trends in the behavior of structures: first, search times with the generic metric structure increases as long as the search ranges increases under all parallel environments and any number of pivots. On the other hand, can be observed that the performance of shared memory processing with GPU is convenient as the search range increases (consequently the number of assessments carried away).

3. Conclusions and Future Work

The need to process large volumes of data requires to increase processing capacity and to reduce search times. In this context, it is relevant to study parallelization algorithms and the distribution of the databases. On the other hand, the increased size of the databases and the emergence of new types of data on which there's no interest in performing searches with exact matches create the need to establish new structures for similarity search.

This paper has presented the main features of the so-called metric spaces and the generic metric structure used to perform similarity searches in such spaces. It has deepened the discussion on those experiments based on the use of generic structures based on pivots, compared with linear or sequential processing known as brute force. It also has shows results of experiments using different strategies for parallel processing, analyzing the search performance in environments with GPU (CUDA) and multi-core (OpenMP).

As future work, it is suggested to discuss the scalability of the use of those platforms, as currently proposed experiments on individual computers. Hybrid aims to reach solutions to the distribution of metric spaces and their structures, so that it is possible to make applications on data volumes in production scale. Finally, another line of great interest to be addressed is related to the energy efficiency of various devices and technologies for parallel processing of large volumes of data with regard to the similarity search of metric spaces.

References

1. GRAMA, Ananth and KARYPIS, George and KUMAR, Vipin and GUPTA, Anshul. Introduction to Parallel Computing (2nd Edition). Addison Wesley. 2003.
2. 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.
3. E. ARIAS, D. CAZORLA, J. L. SÁNCHEZ, R. URIBE-PAREDES. "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.
4. Edgar CHÁVEZ, Gonzalo NAVARRO, Ricardo BAEZA-YATES, and José Luis MARROQUÍN. Searching in metric spaces. ACM Computing Surveys, 33(3):273--321, 2001.

5. Sergei Brin. Near neighbor search in large metric spaces. In the 21st VLDB Conference, pages 574—584. Morgan Kaufmann Publishers, 1995.
6. Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-Tree : An efficient access method for similarity search in metric spaces. In the 23st International Conference on VLDB, pages 426--435, 1997.
7. Gonzalo Navarro. Searching in metric spaces by spatial approximation. The Very Large Databases Journal (VLDBJ), 11(1):28--46, 2002.
8. 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.
9. Roberto Uribe-Paredes and Gonzalo Navarro. Egnat: A fully dynamic metric access method for secondary memory. In Proc. 2nd International Workshop on Similarity Search and Applications (SISAP), pages 57--64, Prague, Czech Republic, August 2009. IEEE CS Press.
10. María Luisa MICÓ, José ONCINA, and Enrique VIDAL. A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements. Pattern Recognition Letters, 15(1):9--17, January 1994.
11. R. BAEZA-YATES, W. CUNTO, U. MANBER, and S. Wu. “Proximity matching using fixed queries trees”. In 5th Combinatorial Pattern Matching (CPM’94), 1994, LNCS 807, pp. 198–212.
12. Edgar CHÁVEZ, José L. MARROQUÍN, and Ricardo 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.
13. S. Nene and S. Nayar. A simple algorithm for nearest neighbor search in high dimensions. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(9):989--1003, 1997.
14. 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.
15. 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.
16. 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.
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. Vincent Garcia, Eric Debreuve, and Michel Barlaud. Fast k nearest neighbor search using GPU. Computer Vision and Pattern Recognition Workshop, 0:1--6, 2008.
19. Wu-Feng and Dinesh Manocha. High-performance computing using accelerators. Parallel Computing, 33:645--647, 2007.
20. O. SOFIA, J. SALVADOR, E. DOS SANTOS, R. URIBE PAREDES. Búsquedas por Rango sobre Plataformas GPU en Espacios Métricos. pp 658 - 662. XV Workshop de Investigadores en Ciencias de la Computación - WICC 2013. Proceedings published in CD format by Universidad Autónoma de Entre Ríos. RedUNCI. ISBN13: 978-987-28179-6-1.