A Fast CAST-based Clustering Algorithm for Very Large Database

Kawuu W. Lin
Department of Computer Science and Information Engineering
National Kaohsiung University of Applied Sciences
Kaohsiung, Taiwan
linwc@kuas.edu.tw

Chun-Hung Lin
Department of Computer Science and Information Engineering
National Kaohsiung University of Applied Sciences
Kaohsiung, Taiwan
tnssh931111@gmail.com

***Abstract*—**The advances in nanometer technology and integrated circuit technology enable the graphics card to attach individual memory and one or more processing units, named GPU, in which most of the graphing instructions can be processed parallelly. Obviously, the computation resource can be used to improve the execution efficiency of not only graphing applications but other time consuming applications like data mining. CAST (Clustering Affinity Search Technique) is a famous clustering algorithm, which is widely used in clustering the biological data. In this paper, we will propose two algorithms, namely Calculation-On-Demand CAST, abbreviated as COD-CAST and Calculation-On-Demand CAST with GPU, abbreviated as COD-CAST-GPU, respectively. The first proposed COD-CAST algorithm is a refined CAST algorithm that can process large amount of objects more efficiently in terms of execution time. The proposed COD-CAST-GPU algorithm can utilize the GPU and the individual memory of graphics card to accelerate the COD-CAST. The experimental results show that our proposed algorithms deliver excellent performance in terms of execution time and required memory.

***Keywords*—Data Mining, Clustering, CAST, GPU**

I. INTRODUCTION

The high definition (HD) and three dimensions (3D) technologies can bring the users the smooth graphing, which is important to many interesting applications like 3D game, high resolution displaying and so forth. Part of the recent progress in improving the graphing efficiency was achieved by the development of graphic processing unit (GPU) technology. In the past, all of the graphing related instructions are processed in CPU, and the memory is shared from the system. The displaying action involves the CPU calculation and the data transmission from the memory to displaying memory. Therefore, the central processing unit (CPU) and memory unit are the most important units that affect the performance of graphing. The advances in nanometer technology and integrated circuit technology enable the graphics card to attach individual memory and one or more processing units [19], named GPU, in which most of the graphing instructions can be processed parallelly. In this architectural design, the graphing efficiency can be greatly improved. The popular graphics card nowadays has one or more GPUs, hundreds of cores and thousands MB of memory. Obviously, the computation resource can be used to improve the execution efficiency of not only graphing applications but other time consuming applications like data mining.

Data mining consists of four main topics, association rules mining [1][12], sequential patterns mining, classification and clustering. The goal of data mining is to discover the hidden useful information from large databases. Clustering objects is an important problem and the cluster information is useful in many fields. CAST (Clustering Affinity Search Technique) [5] is a famous clustering algorithm, which is widely used in clustering the biological data. A complete clustering process when using CAST to cluster data requires a lot of float point computing. For example, CAST needs to pre-calculate a similarity matrix for store object similarity. Suppose there are n objects, to derive the matrix the complexity is $O(n^2)$. Unlike CPU, GPU is not a general purpose processing unit, and it is designed to process complex float point computing. In fact, the size of biological data is always huge. As the size of database increases, the computation time and the required memory increase severely. The traditional clustering methods suffer from the data size. The difficulty of clustering large amount of objects launched the research of designing new algorithms that can utilize the available resources to solve the problem. In this paper, we will propose two algorithms, namely Calculation-On-Demand CAST, abbreviated as COD-CAST and Calculation-On-Demand CAST with GPU, abbreviated as COD-CAST-GPU, respectively. The first proposed COD-CAST algorithm is a refined CAST algorithm that can process large amount of objects more efficiently in terms of execution time. The proposed COD-CAST-GPU algorithm can utilize the GPU and the individual memory of graphics card to accelerate the COD-CAST.

In the following sections, we briefly review related work in Section 2. In Section 3, the proposed algorithms, COD-CAST and COD-CAST-GPU, are presented. The empirical evaluation

for performance study is made in Section 4. The conclusions are given in Section 5.

II. Related Work

In this section, we briefly review the most related work including clustering algorithms, CAST algorithm, general purpose computing on GPU (GPGPU), and data mining on GPGPU.

A. Clustering algorithms

Clustering analysis is an approach to put objects in clusters, in which the objects in the same cluster are similar and the objects falling in different clusters are dissimilar. The main types of clustering algorithms are listed with its famous algorithms:

1) Partitioning-based: K-Means [17], K-Medoids[15], PAM [15], CLARA [15], CLARANS [18], CAST [5], etc.

2) Hierarchical-based: HAC [22], BIRCH [24][25], CURE [10], ROCK [11], CHAMELEON [14], etc.

3) Density-based: CAST [5], DBSCAN [7], OPTICS[4], CLIQUE [2], WaveCluster [21], etc.

4) Grid-based: STING [23], CLIQUE [2], WaveCluster [21], etc.

5) Model-based: SOM [16], COBWEB [8], CLASSIT [9], AutoClass [6], etc.

B. CAST(Clustering Affinity Search Technique)

CAST is the algorithm on which we focus to refine and extend. The input of CAST consists of 1) a similarity matrix to store the distances of all of the objects, and 2) an affinity threshold. The algorithm works as follows. It first initializes a set *C* for the clusters and a set *U* containing the unclustered objects. For each cluster in *C*, we calculate the affinity of each object. For each object in *U*, we calculate the similarity between it and a targeted object, and the similarity values are summed as the affinity of the targeted object. If the affinity is greater than or equal to the affinity threshold, this object should be added to the current cluster and be marked as clustered in *U*. In the same time, the affinity of each object in the cluster should be updated by adding the similarity between it and the newly added object. If the affinity is less than the affinity threshold, the object with lowest affinity will be removed from this cluster and this object is marked as unclustered in *U*. The affinity of each object in the cluster should be updated by subtracting the similarity between it and the removed object. The add and remove actions are then repeated until there is no change. The clustering for the current cluster also terminates. The algorithm starts another clustering for finding new cluster by repeating the steps, and terminates when all of the objects of *U* are marked as clustered.

C. General-Purpose Computing on GPU (GPGPU)

The rapid progress in GPU upgrades the computing power of personal computer. Several studies have started to explore the topic of combining GPU and CPU to fasten time consuming problems. Some well-known platforms were also developed, such as Compute Unified Device Architecture (CUDA) [19] of NVIDIA, FireStream [3] of AMD, Open Computing Language (OpenCL) [20]. The CUDA and FireStream are designed specific to their own graphics cards, and OpenCL is an open platform for supported graphics cards.

D. Data Mining on GPGPU

In [26], the authors proposed a Apriori-based method named FPM-GPU to discover the frequent patterns by using GPGPU. FPM-GPU uses a new data structure in order to reduce the data size and then transmit the data to the memory of graphics card for further mining by GPU. In [13], the authors proposed a clustering method named GPU-based K-Means, which is a K-Means-based method. Two approaches were proposed. The first approach focuses on the centroid. It calculates the distance between the centroid and each object and then returns the result to CPU for clustering. The second approach assigns each core objects and then calculates the distance among the objects. Afterwards, the result is returned to CPU for clustering.

III. Proposed Method

In this section, we will introduce the proposed algorithms, COD-CAST and COD-CAST-GPU. CAST algorithm takes two inputs, 1) a similarity matrix that stores the similarity between objects and 2) an affinity threshold, and outputs the clusters. Deriving the similarity matrix is a time consuming task. The previous studies however consider the deriving as a preprocessing task and do not pay attention on this part. The matrix deriving in fact is the performance bottleneck especially when the number of objects is large. The proposed COD-CAST and COD-CAST-GPU are able to cluster large dataset efficiently in terms of execution time and required memory.

A. Calculation-On-Demand CAST (COD-CAST)

The COD-CAST algorithm is as shown in **Figure 1**. Note that the input to this algorithm is the n objects while not the n by n similarity matrix. When we use CAST to cluster a very large dataset, the memory usually cannot afford to load the entire matrix. Although some operation systems or programming languages can simulate the hard-disk space to memory to load the entire matrix, the speed will be very slow.

Therefore, we calculate the similarity when it is necessary but not calculate the entire matrix in advance. The proposed Update_Affinity function is as shown in **Figure 2**, in which we use the Euclidean distance as an example.

B. Calculation-On-Demand CAST with GPU (COD-CAST-GPU)

To accelerate the CAST with GPU, we parallel the computation of Update_Affinity function as shown in **Figure 3**.

Input : the data with n nodes N, and an affinity threshold t
Output : the collection of closed clusters C
Method :

Step 1 : Initialization_U
$C \leftarrow \emptyset$; $U \leftarrow \{1,2,\dots,n\}$;
Step 2 : **While($U \neq \emptyset$) do**
Step 3 : Initialization_C_{open}
$C_{open} \leftarrow \emptyset$; a(·) ← 0; change = false;
Step 4 : ADD :
While($\max\{a(u)|u \in U\} \geq t \times |C_{open}|$) **do**
$C_{open} \leftarrow C_{open} \cup \{u\}$; U ← U \ $\{u\}$
For all node x ∈ U ∪ C_{open}
$a(x) \leftarrow Update\_affinity(u)$;
//Update all node affinity with node u
change = true;
End While(step 4)
Step 5 : REMOVE:
While($\min\{a(u)|u \in C_{open}\} < t \times |C_{open}|$) **do**
$C_{open} \leftarrow C_{open} \setminus \{u\}$; U ← U ∪ $\{u\}$
For all node x ∈ U ∪ C_{open}
$a(x) \leftarrow Update\_affinity(u)$;
//Update all node affinity with node u
change = true;
End While(step 5)
Step 6 : If(change)
change = false; Repeat **Step 4** and **Step 5**;
else
$C \leftarrow C \cup \{C_{open}\}$; Back to **Step 2**;
Step 7 : **End While(step 2)**

Figure 1. The proposed COD-CAST algorithm.

Input : the data with n nodes N, and the variational node u
Output : Affinity array after update a
Method :

Step 1 : $d^{max} = d(max(N), min(N))$;
Step 2 : For all nodes $X_i \in N, 0 < i \leq n$
$d(i,u) = \sqrt{\sum_{k=1}^{m}(x_i^k - x_u^k)^2}$; ……(1)
$S(i,u) = \frac{d^{max} - d(i,u)}{d^{max}}$; ……(2)
Step 3 : If call from ADD step
$a(i) += S(i,u)$;
else call from REMOVE step
$a(i) -= S(i,u)$;
Step 4 : End For

Figure 2. The Update_Affinity function.

Input : the data with n nodes N, and the variational node u
Output : Affinity array after update a
Method :

Step 1 : $d^{max} = d\big(max(N), min(N)\big);$
Step 2 : $N_{GPU} = malloc\ GPU\ memory\big(sizeof(N)\big);$
Step 3 : $a_{GPU} = malloc\ GPU\ memory\big(sizeof(a)\big);$
Step 4 : $MemcpyHostToDevice(from\ N\ to\ N_{GPU});$
Step 5 : $MemcpyHostToDevice(from\ a\ to\ a_{GPU});$
Step 6 : $Affinity\_Update\_on\_GPU(N_{GPU}, a_{GPU}, d^{max}, u)$ (GPU start)
Step 7 : For all nodes $X_i \in N_{GPU}, 0 < i \leq n$

$$d(i,u) = \sqrt{\sum_{k=1}^{m}\left(x_i^k - x_u^k\right)^2} \quad ; \qquad \text{.....(1)}$$

$$S(i,u) = \frac{d^{max} - d(i,u)}{d^{max}}; \qquad \text{.....(2)}$$

Step 8 : If call from ADD step
$a_{GPU}(i) \mathrel{+}= S(i,u);$
else call from REMOVE step
$a_{GPU}(i) \mathrel{-}= S(i,u);$
Step 9 : $End_{Affinity\_Update\_GPU}$ (GPU end)
Step 10 : $MemcpyDeviceToHost(from\ a_{GPU}\ to\ a\ );$
Step 11 : $free\ N_{GPU}\ and\ a_{GPU};$

Figure 3. Update_Affinity_with_GPU (u) Algorithm

IV. Experimental Results

The experiments were conducted on a HP xw6600 workstation with one Quad Core Intel Xeon 2.0 GHz CPU and 4GB main memory. The graphics card is NVIDIA Quadro FX570. The GPU clock rate is 920 MHz, with 16 cores, 2 multi processors, 256 MB memory. The parameter setting for the clustering algorithms and GPU are listed in **Table 1**. In the following section, we select the original CAST for performance study.

Study of varying the number of objects for CAST, COD-CAST and COD-CAST-GPU

In **Figure 4**, we observe the effects of varying the number of objects. We found that as the increase in the number of objects, COD-CAST and COD-CAST-GPU have better performance than CAST in terms of execution time. When the number of objects is 128,000, COD-CAST-GPU requires only 7.7% execution time of CAST.

Table 1. Parameter settings.

CAST Algorithm		
Parameter	Default Value	Description
n		the number of data node
m	2	the number of attribute per node
t	0.8	the affinity threshold
d^{max}	$10\sqrt{2}$ $\div 14.14$ ...	the max of distance between nodes(if $m = 2$)
CAST with GPU Algorithm		
Parameter	Default Value	Description
THREADS_PER_BLOCK	512	the number of threads per block
BLOCK_PER_GRID	2	the number of blocks per grid

V. Conclusions

In this paper, we have proposed two algorithms, namely COD-CAST and COD-CAST-GPU, respectively. The first proposed COD-CAST algorithm is a refined CAST algorithm that can process large amount of objects more efficiently in terms of

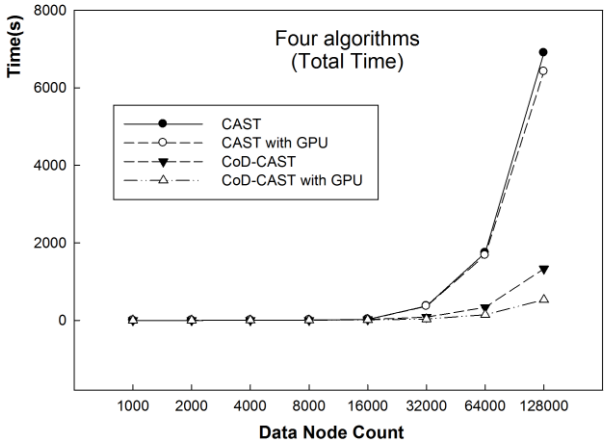

Figure 4. The effects of varying the numbers

execution time. The proposed COD-CAST-GPU algorithm can utilize the GPU and the individual memory of graphics card to accelerate the COD-CAST. The experimental results also show that our proposed method delivers excellent performance in terms of execution time.

Acknowledgement: This research was supported by the National Science Council in Taiwan through Grant NSC 99-2622-E-151-005-CC3 and NSC 99-2221-E-151-046.

REFERENCES

[1] R. Agrawal, R. Srikant. (1994) "Fast Algorithms for Mining Association Rules", Proc. 20th Int. Conf. Very Large Data Bases(VLDB), Santiago, Chile.

[2] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. (1998) "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications", (SIGMOD '98) Proceedings of the 1998 ACM SIGMOD international conference on Management of data, Seattle, Washington, June, Volume 27, Issue 2.

[3] AMD Official Website , FireStream GPU Compute Accelerators , (2011) , http://www.amd.com/us/products/workstation/firestream/Pages/firestream.aspx.

[4] M. Ankerst, M. M. Breunig, H. P. Kriegel, and J. Sander. (1999) "OPTICS: ordering points to identify the clustering structure", Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, Philadephia, Pennsylvania, USA, pages 49-60, June.

[5] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. (1999) "Clustering Gene Expression Patterns", Journal of Computational Biology, Volume 6, Numbers 3/4, October 1999, pp. 281-297. Doi : 10.1089/106652799318274.

[6] P. Cheeseman and J. Stutz. (1996) "Bayesian classification (AutoClass): Theory and results", Advances in Knowledge Discovery and Data Mining, pages 153-180, ISBN: 0-262-56097-6.

[7] Martin Ester, Hans-Peter Kriegel, Jorg Sander and Xiaowei Xu. (1996) "A density-based algorithm for discovering clusters in large spatial databases with noise", Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, pages 226-231, Portland, Orgon.

[8] Doug Fisher. (1987) "Improving Inference through Conceptual Clustering", (AAAI'87) Proceedings of the sixth National conference on Artificial intelligence - Volume 2, pages 461-465.

[9] John H. Gennari, Pat Langley, and Doug Fisher. (1989) "Models of incremental concept formation", Volume 40, Issues 1-3, Pages 11-61, September.

[10] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. (1998) "CURE: An efficient clustering algorithm for large databases", (SIGMOD '98) Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, pages 73-84, New York.

[11] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. (1999) "ROCK: a robust clustering algorithm for categorical attributes", Information Systems, Volume 25, Issue 5, July, Pages 345-366.

[12] Jiawei Han, Jian Pei, and Yiwen Yin. (2000) "Mining Frequent Patterns without Candidate Generation", Proc. the 2000 ACM SIGMOD international conference on Management of data.

[13] Bai Hong-tao, He Li-li, Ouyang Dan-tong, Li Zhan-shan, and Li He.(2009) "K-Means on Commodity GPUs with CUDA", 2009 WRI World Congress on Computer Science and Information Engineering, Los Angeles, California USA.

[14] G. Karypis, E. H. Han, and V. Kumar. (1999) "CHAMELEON: A hierarchical clustering algorithm using dynamic modeling", Department of Computer Science, University of Minnesota, Minneapolis, Volume: 32, Issue:8, pages 68-75.

[15] L. Kaufman and P. J. Rousseeuw. (1990) "Finding groups in data: an Introduction to cluster analysis", John Wiley & Sons.

[16] Teuvo Kohonen. (1990) "The self-organizing map", Proceedings of the IEEE, Vol. 78, No 9, pages 1464-1480, September.

[17] J. B. McQueen. (1967) "Some Methods of Classification and Analysis of Multivariate Observations." Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, pages 281-297.

[18] Raymond T. Ng and Jiawei Han. (1994) "Efficient and effective clustering methods for spatial data mining", Proceedings of the 20th VLDB Conference, pages 144-155, Santiago, Chile.

[19] NVIDIA Official Website , CUDA Zone , (2011) , http://www.nvidia.com.tw/object/cuda_home_new_tw.html.

[20] OpenCL Official Website, (2011), http://www.khronos.org/opencl/.

[21] Gholamhosein Sheikholeslami, Surojit Chatterjee, and Aidong Zhang. (1998) "WaveCluster : A multi-resolution clustering approach for very large spatial databases", (VLDB '98) Proceedings of the 24rd International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

[22] E. M. Voorhees. (1986) "Implementing agglomerative hierarchical clustering algorithms for use in document retrieval", Information Processing & Management, 22:465-476.

[23] Wei Wang, Jiong Yang, and Richard Muntz. (1997) "STING: a statistical information grid approach to spatial data mining", Proc. 23rd Int. Conf. on Very Large Data Bases (VLDB), 186-195.

[24] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. (1996) "BIRCH: An Efficient Data Clustering Method for Very Large Databases", Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, pages 103-114, Montreal, Canada.

[25] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. (1997) "BIRCH: A new data clustering algorithm and its applications", Data Mining and Knowledge Discovery, pages 141-182.

[26] . Jiayi Zhou, Kun-Ming Yu, and Bin-Chang Wu. (2010) "Parallel frequent patterns mining algorithm on GPU", Systems Man and Cybernetics SMC 2010 IEEE International Conference on (2010), Pages 435-440.