Parallel Data Mining Techniques on Graphics Processing Unit with CUDA

Liheng Jian Ying Liu(yingliu@gucas.ac.cn) Shenshen Liang
School of Information Science and Engineering,
Graduate University of Chinese Academy of Sciences, Beijing, China

Introduction

Data mining is widely used in various domains and has significant applications. However, current data mining tools cannot meet the requirement of applications with large-scale databases in terms of speed. We propose three techniques to accelerate fundamental kernels in data mining algorithms on CUDA platform, *scalable thread scheduling scheme for irregular pattern*, *parallel distributed top-k scheme*, and *parallel high dimension reduction scheme*. They play a key role in our GUCAS_CU-Miner, including three representative data mining algorithms, *CU-Apriori*, *CU-KNN* and *CU-K-means*.

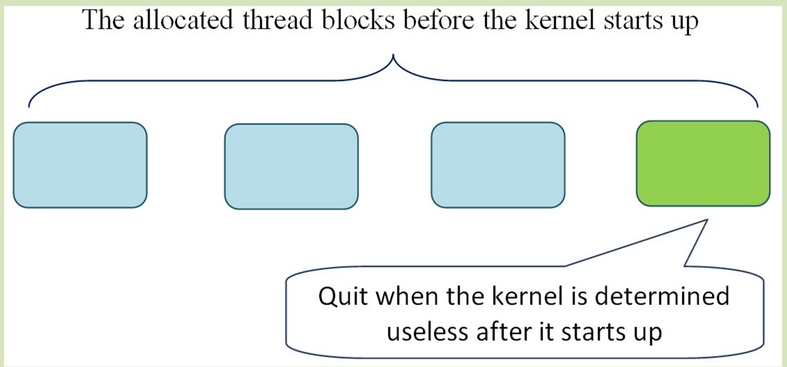
Scalable Threads Scheduling Scheme for Irregular Pattern

Problem: In irregular pattern, the size of a problem changes dynamically during a process, making a challenge for CUDA computing model.

Solution:

1) Estimate the upper bound of threads/thread blocks, then allocate the GPU resources.

2) Let the #threads/#blocks quit immediately if it is determined useless when the processing begins.



Comparing to re-starting a GPU kernel with an updated size, the overall performance can be improved by sacrificing part of the computation resource, since launching or exiting a thread block incurs trivial cost.

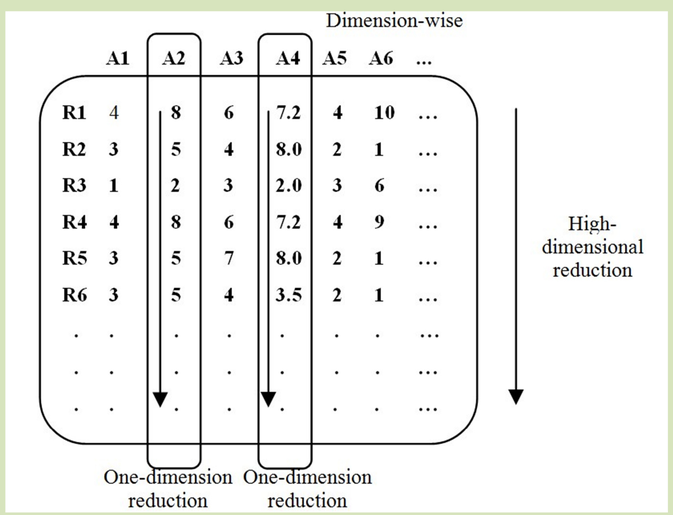
Parallel High Dimension Reduction Scheme

Problem: Data with high dimensionality makes the cost for data manipulation and temporal results storage very high.

Solution:

1) We see the same attribute (dimension) on all data as a vector, and perform reduction on each attribute rather than on each row. Each thread block only takes care of one distinct attribute.

2) The sequential addressing reduction in CUDA SDK is chosen for one-dimensional reduction.



This scheme aims at maximizing the thread parallelism, and eliminating the shared memory overflow problem with a number of large data elements.

Parallel Distributed Top-k Scheme

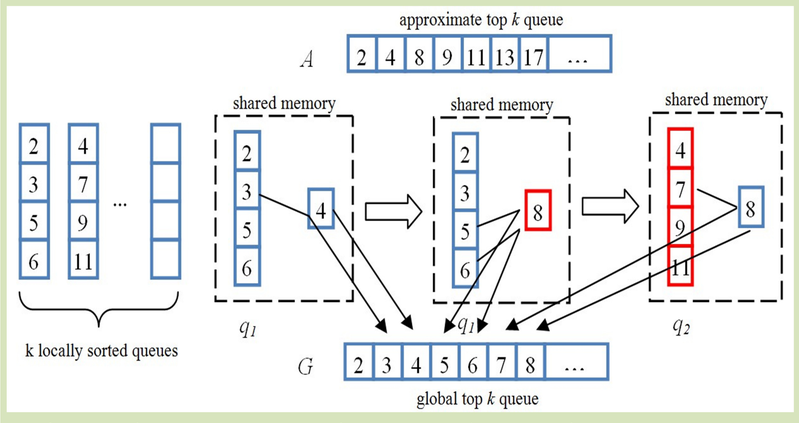
Problem: Top-k problem is to select *k* minimum or maximum elements from a data collection.

Solution:

Step 1: Local sort. Divide the data collection into small data partitions with equal size, then store and sort them in the shared memory concurrently.

Step 2: Approximate top-k queue. Form a data collection using the heads of each sorted queues, then a insertion sort is performed on it to pick out the approximate top-k queue.

Step 3: Global top-k queue. Based on the approximate top-k queue, A, and *k* local sorted queues, whose heads are in A and sorted according to their heads, the exclusive property is applied to produce the global top-*k* queue.



This scheme not only cuts down the comparison between elements, but also maximizes parallelization by classifying data concurrently.

Exclusive property: If element *a* is less than element *b* which belongs to a sorted queue *q*, any element greater than *b* in *q* cannot be less than *a*.

GUCAS_CU-Miner

It is a CUDA-based data mining toolkit, now including three algorithms. The Aforementioned techniques play a key role in enhancing the performance.

CU-Apriori: It is to identify frequently co-occurring itemsets in a database. We implement Candidate Generation kernel as a 2D thread grid using our *scalable threads scheduling scheme for irregular pattern*, so as to get an easy control and higher efficiency. Each thread tests whether two frequent itemsets can be joined.

CU-KNN: It is a method for classifying objects based on their closest reference objects. We implement the Neighbor Selecting kernel using our *parallel distributed top-k scheme*, since the selection of *k* nearest neighbors of a query object is a typical top-*k* problem.

CU-K-Means: It is to group the data into clusters so that objects within a cluster have high similarity while objects in differernt clusters have high dissimilarity. There are two CUDA kernels, reduction of objects' attributes and counts, and detection of centroids movement, which adopt our *parallel high dimension reduction scheme* to maximize the thread parallelization.

Acknowledgements

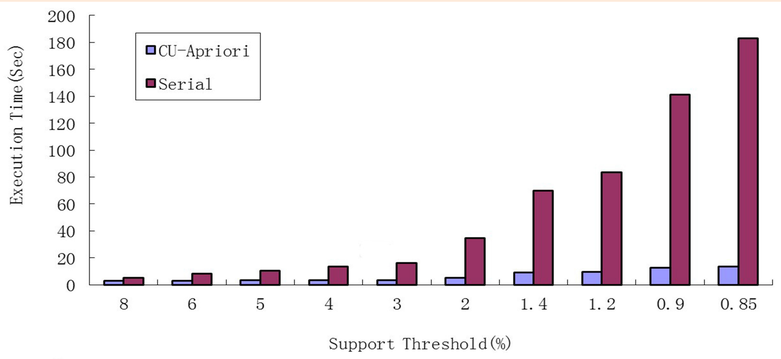
This work has been supported by 2009 NVIDIA's Professor Partnership.

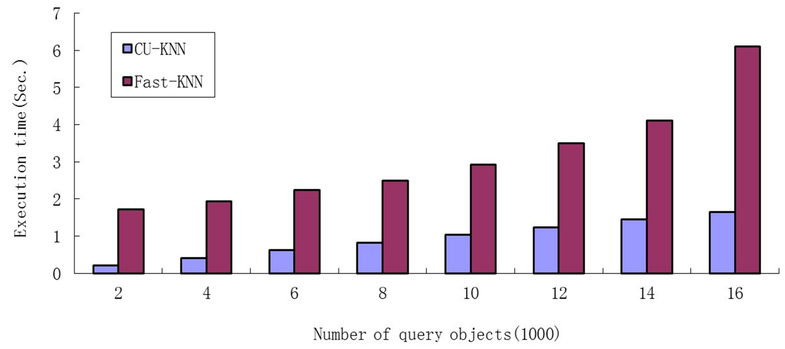
Experimental Results

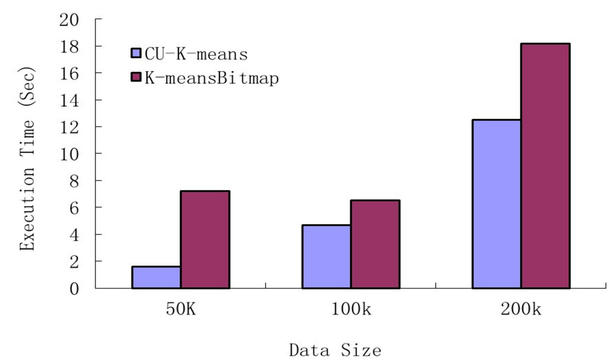
Hardware: HP xw8600 workstation, *Core-quad* 2.66 GHz Intel Xeon CPU, 4 GB host memory, Tesla C1060 card (240 1.30 GHz SPs, 4 GB device memory)

Software: Red Hat Enterprise Linux WS 4.7, CUDA 2.1, gcc 3.4.6

The serial Apriori in NU-MineBench [1], Fast-KNN [2], and K-meansBitmap in GPUMiner [3] are used for comparison with our corresponding algorithms, 13.5-times, 8.31-times, and 5-times speedup are observed as shown in the following figures, respectively . See [4] for more detail.







Conclusion

Our experiments demonstrate our proposed techniques work efficiently, and our toolkit also indicates that GPU + CUDA parallel architecture is feasible and promising for data mining applications.

Novelty: Our work is the first to propose a parallel distributed top-*k* scheme on CUDA, which can be treated as an independent algorithm.

References

[1] Liu Y, Pisharath J, Liao WK, Memik G, et al. (2004) Performance evaluation and characterization of scalable data mining algorithms. 16th IASTED International Conference on Parallel and Distributed Computing and Systems, pp. 620-625, MIT Cambridge, Massachusetts, USA.

[2] Garcia V, Debreuve E, Barlaud M. (2008) Fast k nearest neighbor search using GPU. IEEE Conference on Computer Vision and Patter Recognition Workshops, Vol. 1-3, pp.1107-1112.

[3] Fang WB, Lau KK, Lu M, et al. (2008) Parallel data mining on graphics processors. Technical Report HKUST-CS08-07, http://code.google.com/p/gpuminer/.

[4] Jian LH, Wang C, Liu Y, Liang SS, et al. (2011) Parallel data mining techniques on Graphics Processing Unit with Compute Unified Device Architecture (CUDA). Journal of Supercomputing, DOI: 10.1007/s11227-011-0672-7.