Parallelizing Apriori Algorithm on GPU

K. Spandana
Assistant Professor
Dept. of CSE
CBIT
Gandipet, Hyderabad.

D. Sirisha
Application Developer
Oracle

S. Shahida
Tech. Associate
Bank of America

ABSTRACT

Parallel computing is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently. Now Graphics Processing Unit (GPU) has taken a major role in high performance computing for generic applications. Compute Unified Device Architecture (CUDA) programming model provides the programmers adequate C-Language like API's to better exploit the power of GPU. Data Mining has significant applications in various domains. Currently, these techniques cannot meet the requirement of applications with large scale databases in terms of computation and speed. Association Rules Mining (ARM) is one of the most widely used techniques in data mining and has tremendous applications. Apriori is the most influential ARM algorithm. It has been included in all the existing commercial and non-commercial data mining. This paper provides a parallel Apriori algorithm on GPU with CUDA and focuses on computation time compared with execution time of serial program in CPU.

General Terms

Data Mining Algorithm parallelization

Keywords

Parallel Computing, GPGPU, GPU, CUDA, Data Mining, Parallel Apriori Algorithm

1. INTRODUCTION

Data mining has become a hot research domain in recent years as it is being used in almost all application. There are many algorithms under this domain. However, these algorithms are potentially unable to handle today's increasing data sets or would take a tremendous amount of time. Therefore, users have to turn to rely on parallel and distributed computing techniques to accelerate the computation.

One of the parallel computing techniques is the General Process Computing on Graphics Processing Unit (GPGPU). This technique uses the power of GPU and CPU to perform computations on applications traditionally handled only by CPU.

A graphics processing unit (GPU) is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display. It allows blocks of data to be processed in parallel. Currently, high level languages have emerged to support easy programming on GPUs. CUDA is a parallel computing platform and programming model invented by NVIDIA. It enables dramatic increases in computing performance by harnessing the power of the graphics processing unit (GPU). Low cost is another factor of GPU. Therefore, recently, there has been a trend to accelerate computational intensive applications on a GPU + CPU heterogeneous system where the GPU acts as the computation accelerator.

Apriori Algorithm is one of the techniques of Data Mining for finding Frequent Itemsets useful in Business Analysis. This is an efficient algorithm for mining Itemsets. But in reality, this algorithm is computationally intensive due to multiple scans of the database in order to find the frequent itemsets. It works efficiently for small databases whereas for large databases, due to its large size, its performance decreases. So to increase its performance and to make it efficient even for the large databases, the algorithm is parallelized on GPU using CUDA. Comparison of the execution time of Parallel-Apriori with an efficient serial Apriori program is used to show the speedup on a real-world data set and even on synthetic data set.

2. LITERATURE REVIEW

2.1 Association Rule Mining

Association Rules:

Association Rules is used to find associations between sets of attributes. Association Rules are interesting association relationship among huge amounts of transactions. An association rule is an expression of the form X => Y, where X and Y are sets of items.

Goal is to find all association rules that satisfy user-specified minimum support and minimum confidence threshold.

Given a set of records each of which contains some number of items from a given collection, Association Rules produce dependency rules which will predict occurrence of an item based on occurrences of other items.

Association rule R: Itemset1 => Itemset2

Itemset1, 2 are disjoint and Itemset2 is non-empty i.e. if transaction includes Itemset1 then it also has Itemset2

Rule Evaluation Metrics

- Support (s): Fraction of transactions that contain both X and Y
- Confidence (c): Measures how often items in Y appear in transactions that contain X

2.2 Apriori Algorithm

Apriori is a classic algorithm for learning association rules. Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation). Other algorithms are designed for finding association rules in data having no transactions (Winepi and Minepi), or having no timestamps (DNA sequencing).

The purpose of the Apriori Algorithm is to find associations between different sets of data. It is sometimes referred to as

"Market Basket Analysis". Each set of data has a number of items and is called a transaction. The output of Apriori is set of Rules that tell us how often items are contained in sets of data. Apriori uses breadth-first search and a Hash tree structure to count candidate item sets efficiently. It generates candidate item sets of length from item sets of length k-1. Then it prunes the candidates which have an infrequent sub pattern. According to the downward closure lemma, the candidate set contains all frequent k length item sets. After that, it scans the transaction database to determine frequent item sets among the candidates. Apriori, while historically significant, suffers from a number of inefficiencies or trade-offs, which have spawned other algorithms. Candidate generation generates large numbers of subsets (the algorithm attempts to load up the candidate set with as many as possible before each scan). Bottom-up subset exploration (essentially a breadth-first traversal of the subset lattice) finds any Maximal subset S only after all 2|s|-1 of its proper subsets.

The whole point of the algorithm (and data mining, in general) is extract useful information from large amounts of data.

For example, the information that a customer who purchases a keyboard also tends to buy a mouse at the same time is acquired from the association rule below:

Support

The percentage of task-relevant data transactions for which the pattern is true is called support.

$$\text{Support}(\text{KB}\rightarrow\text{Mouse}) = \frac{\text{No. of transactions containing KB \& mouse}}{\text{No. of total transactions}}$$

Confidence

The measure of certainty or trustworthiness associated with each discovered pattern is called confidence.

$$\text{Confidence}(\text{KB}\rightarrow\text{Mouse}) = \frac{\text{No. of transactions containing KB \& mouse}}{\text{No. of total transactions}}$$

The algorithm aims to find the rules which satisfy both a minimum support threshold and a minimum confidence threshold (Strong Rules).

Item: article in the basket.

Item set: a group of items purchased together in a single transaction.

Let I = {i1, i2, . . . ,im} be a set of m distinct attributes, or called items. Each transaction T in the database D has a unique identifier, and contains a set of items, in the form of _TID, i1, i2, . . . ,ip.

An itemset with k items is called a k-itemset. The support of a k-itemset X is the fraction of the transactions in D containing X. X is a frequent itemset if X's support is greater than the user-specified minimum support threshold ε.

The goal of frequent itemset mining is to find the complete set of frequent itemsets in a database. The downward closure property (any subset of a frequent itemset must also be frequent) is successfully introduced to cut down the number of candidates generated in each iteration. It performs a level-wise search. In each iteration (line 2 to 10), firstly, a set of k-candidates is generated by joining two frequent (k − 1)-itemsets if they share a common (k − 2)-prefix. A pruning procedure is invoked to eliminate any candidate which contains an infrequent subset; secondly, the support of every candidate itemset is counted by scanning the database. The loop terminates when no more frequent itemsets are discovered.

Existing Algorithm

Pseudo-code of main function

Input: Database, D, minimum support threshold, min_sup

Output: Frequent item sets, L

```
L1 = find_frequent_1-itemsets (D);
for (k = 2; Lk−1 _= ∅; k++){
        Ck= apriori_gen (Lk−1);
        for each transaction t ∈D{
                Ct = subset (Ck, t);
                for each candidate c ∈Ct
                        c.count++;
        }
        Lk= {c ∈Ck; c.count≥ min_sup};
}
return L = UkLk;
```

Pseudo-code of sub-functions of Apriori

```
Procedure apriori_gen (Lk-1) //candidate generation
for each itemset l1 ∈Lk-1
for each itemset l2 ∈Lk-1
if((l1[1] = l2[1])∧· · ·∧(l1[k −1] < l2[k −1])){
        c= l1 _ _l2;
if has_infrequent_subset(c,Lk−1) then
        delete c;
else add c to Ck;
}
return Ck;
Procedure has_infrequent_subset(c; Lk−1)
for each (k −1)-subset s of c
if s ∈Lk−1 then
return TRUE;
return FALSE;
```

2.3 Graphics Processing Unit (GPU)

The GPU, as a specialized processor, addresses the demands of real-time high-resolution 3D graphics compute-intensive tasks. As of 2012, GPUs have evolved into highly parallel multi-core systems allowing very efficient manipulation of large blocks of data. This design is more effective than general-purpose CPUs for algorithms where processing of large blocks of data is done in parallel, such as,

push-reliable maximum flow algorithm

fast sort algorithms of large lists

two-dimensional fast wavelet transform

molecular dynamics simulations

GPGPU (General Purpose Graphics Processing Unit) supported programming models better exploit the parallel power of the many core GPU architectures. Present Data mining tools are not able to meet the requirement of large-scale databases in terms of speed and scalability. GPU technology is the extensive scalable platform to deploy the data mining algorithms. Due to the inherent parallelism of GPU many core architecture, it has become possible to program GPU processors directly, as massively parallel processors. GPU computation is highly scalable, inexpensive and high performance per dollar. Combining CPU with the GPU massively parallelism helps to scale up the algorithms for knowledge discovery by applying the data mining algorithm on the entire dataset.

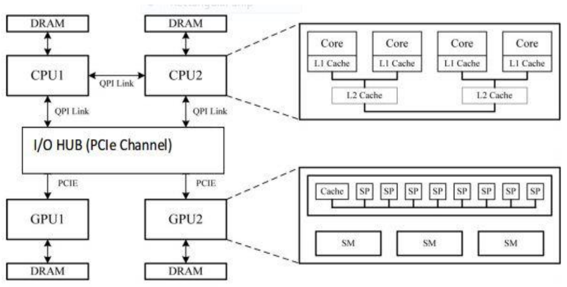


Fig. 1 CPU+GPU Parallel Architecture

Fig. 1 shows the architecture of GPU. The GPU architecture has Streaming Multiprocessors, which has many blocks. Each block has multiple threads. All these threads executes on concurrent parallelism. The GPU implementation for Apriori Algorithm to implement the scalability of the algorithm explores the CUDA memory hierarchy, which helps to exploit the parallelism on large datasets. The data access on device memory, global memory, texture memory, and constant memory varies the performance of the algorithm. Also, CUDA environment gives flexibility to share combined memory, offers very less communication overheads.

The maximum possible threads to be active for parallel execution can be computed by occupancy formula as given below.

$$\text{Occupancy} = \frac{\text{Blocks per SM x Threads per block}}{\text{Maximum threads per SM}}$$

Where,

SM is the acronym used for Streaming Multiprocessors in GPU architecture.

CUDA threads are lightweight and fast switching, 1000s of threads execute simultaneously. All threads execute the same code, each thread has an ID, Threads are grouped into blocks, Blocks are grouped into a grid and a kernel is executed as a grid of blocks of threads.

2.4 GPU Accelerated Computing

GPU accelerated Computing is the use of a graphics processing unit (GPU) together with a CPU to accelerate scientific, analytics, engineering, consumer, and enterprise applications. From the Fig.2, it is clearly understood that GPU-accelerated computing offers unprecedented application performance by offloading compute-intensive portions of the application to the GPU, while the remainder of the code still runs on the CPU. From a user's perspective, applications simply run significantly faster.

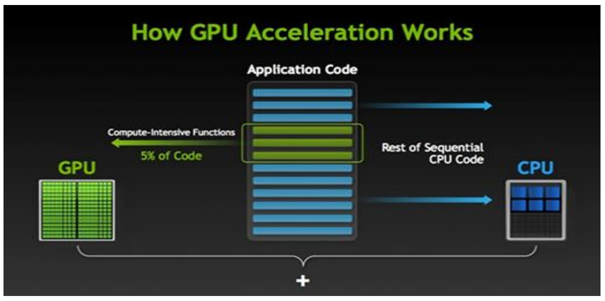

How GPU Acceleration Works
Application Code
Compute-Intensive Functions
5% of Code
GPU
Rest of Sequential CPU Code
CPU
+

Fig. 2 How GPU acceleration works

CUDA

Compute Unified Device Architecture (CUDA) is a parallel computing platform and programming model created by NVIDIA and implemented by the graphics processing units (GPUs) that they produce. CUDA gives developers direct access to the virtual instruction set and memory of the parallel computational elements in CUDA GPUs.

Processing Flow in CUDA

From the Fig.3, the processing flow in CUDA is:

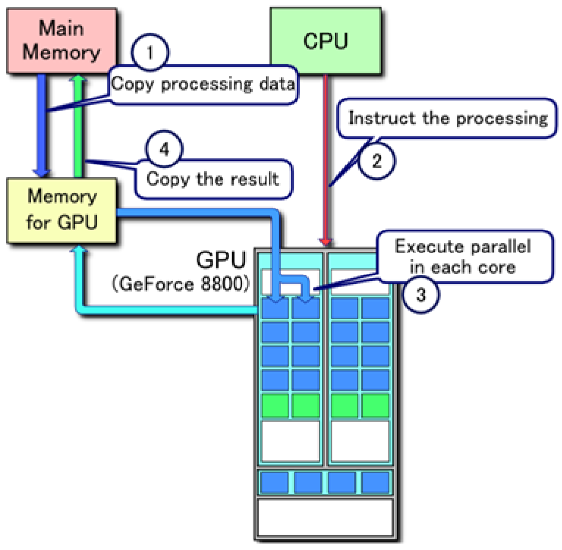


Fig. 3 Processing Flow on CUDA

CUDA Programming Model

At the software level, the CUDA model is a collection of threads running in parallel. The unit of work issued by the host computer to the GPU is called a kernel. CUDA program is running in a thread-parallel fashion. Computation is organized as a grid of thread blocks which consists of a set of threads as shown in Fig. 4. At instruction level, 32 consecutive threads in a thread block make up of a minimum unit of execution, which is called a thread warp. Each SM executes one or more thread blocks concurrently. A block is a batch of SIMD-parallel threads that runs on the same SM at a given moment. For a given thread, its index determines the portion of data to be processed.

Fig. 4 shows the CUDA programming model which has threads divided in blocks. Threads in a single block communicate through the shared memory. CUDA consists of a set of C language extensions and a runtime library that provides APIs to control the GPU. Thus, CUDA programming model allows the programmers to better exploit the parallel power of the GPU for general-purpose computing.

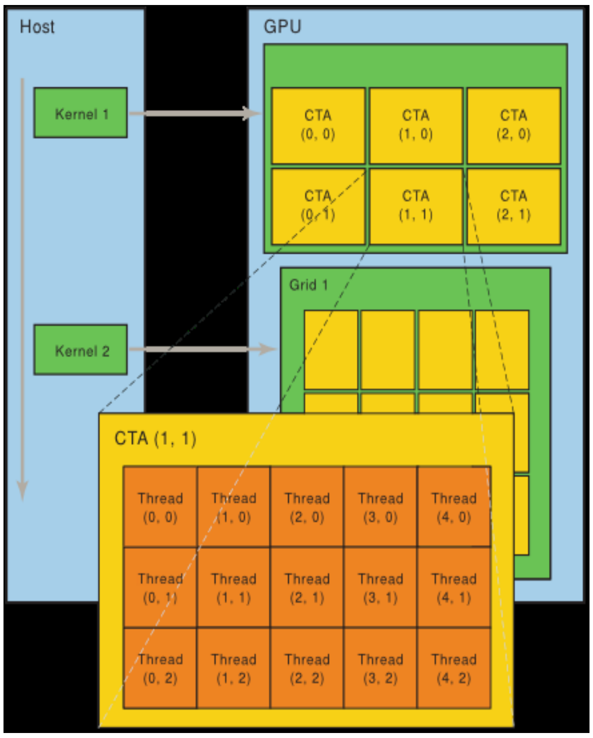


Fig. 4 CUDA Programming Model

Programming in CUDA

With the CUDA architecture and tools, developers are achieving dramatic speedups in fields such as medical imaging and natural resource exploration, and cryptography. One of the major benefits of CUDA as compared to other GPU programming systems is its use of a C dialect, such that the original function for the

CPU can often be transformed into a CUDA kernel with only slight modifications. CUDA provides to developers C libraries that expose all device functionalities needed to integrate CUDA into C program.

The programmer, in order to write a CUDA program, normally begins from a sequential version and proceeds trough the following steps

Identify a kernel, and package it as a separate function.

Specify the grid of GPU threads that executes it, and partition the kernel computation among these threads, by using blockIdx and threadIdx inside the kernel function.

Manage data transfer between the host memory and the GPU memories(global, constant and texture), before and after the kernel invocation. This includes redirecting variable accesses in the kernel to the corresponding copies allocated in the GPU memories.

Perform memory optimizations in the kernel, such as utilizing the shared memory and coalescing accesses to global memory.

Perform other optimizations in the kernel in order to achieve an optimal balance between single-thread performance and the level of parallelism.

3. PARALLEL APRIORI ALGORITHM

3.1 Data representation for Parallel Apriori Algorithm

Apriori algorithm assumes that candidate sets to be in memory, which can be expensive when the candidate set is very large. Concerning speedup, memory optimized usage and sensitivity of parameters, the data structure to accommodate the complete transactional data, and search efficiency, the researchers has suggested Trie, Hash Tree. However tree structure is not convenient as compared to the transactional data; as transactional dataset is always the best candidate to parallelize the transactions.

Horizontal Representation

The most straight forward way to store transactions is to store a list of items that comprise each transaction, shown in Fig. 5. This is called the horizontal representation.

Tid↓	Items ->			
1	A	D	E	
2	A	B		
3	C	D	E	
4	A	B	D	E

Fig. 5 Horizontal Representation

Vertical Representation

Stores the list of the transactions ids corresponding to items, shown in Fig. 6. The vertical representation has been referred by variants of Apriori algorithms. Experimental results show that the vertical representations usually can speed up the algorithm by one order of magnitude on most of the test dataset. This approach is referred to as a "Tidset".

Items ->	A	B	C	D	E
	1	2	3	1	1
	2	4		3	3
	4			4	4

Fig. 6 Vertical Representation

BitMap Representation

The transaction list can also be represented as a bit corresponds to the transactions and item id, as shown in Fig. 7, which we can refer as a "bitset". When the candidates are represented as bitsets, it takes comparatively less memory space than horizontal and vertical representations, as it takes only Byte per item for representation. This could be best effective if the matrix is dense.

↓Tid \ Items →	A	B	C	D	E
1	1	0	0	1	1
2	1	1	0	0	0
3	0	0	1	1	1
4	1	1	0	1	1

Fig. 7 BitMap Representation

The bitset representation is more suitable for designing a parallel set join operation, which is better suited for GPU.

3.2 Proposed Algorithm

```
Input: Transaction Database, D; Min Support- Smin;
Output: Frequent Item sets, FI
Apriori (D,Smin){
        While (!EOF ) do{
  DatasetBuff = ReadFile(row);
  Row++;
  FindMaxColmn();
        }
        Malloc (Matrix_Size)
        ConvertToBitMatrix();
        // Allocate memory space in GPU;
        Transform Smin and bitMatrix to GPU memory
}
GPU_Kernel {
ReadItemSet Cm, Cn
SupportCount = Sum(Cm &&Cn)
If SupportCount < Smin  prune (Cm,Cn)
else
FI = AddCandidate()
}Return FI.
```

4. RESULTS

4.1 Graph Display

Based on the transaction count and item count in the input files given by the user, the execution times of CPU, GPU and CPU+GPU are compared and shown in the form of graphs. The graph is drawn with size of the transaction database on X-axis and execution time (in milliseconds) on Y-axis. Fig. 8 shows the graphs output in the form of bar graph and line graph. Detailed description of graph is given from Fig. 8(a). These graphs are drawn based on transaction databases of sizes 100, 200, 400, 800 and 1000.

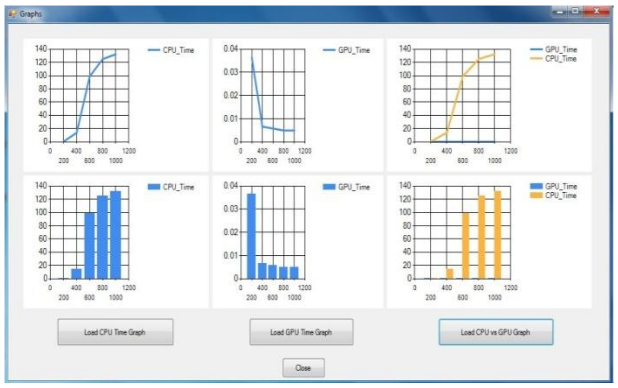
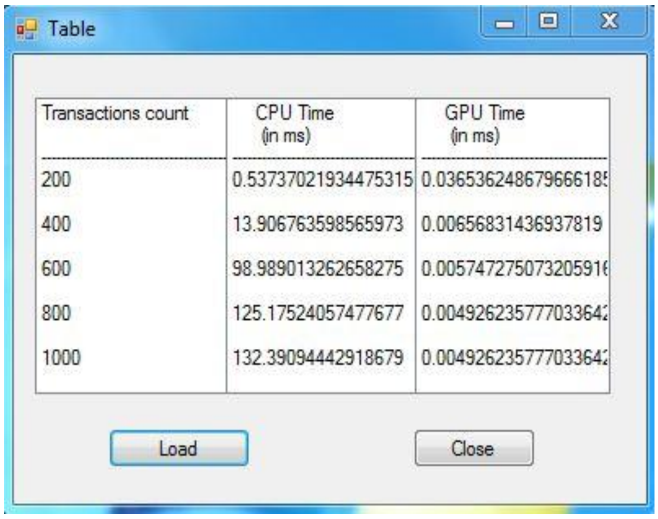


Fig. 8 Graph Display

Fig. 9 shows the execution times of CPU andCPU+GPU when the algorithm is executed with 100,200, 400, 800, 1000 transactions respectively.



Table

Transactions count	CPU Time (in ms)	GPU Time (in ms)
200	0.53737021934475315	0.03653624867966618
400	13.906763598565973	0.00656831436937819
600	98.989013262658275	0.00574727507320591
800	125.17524057477677	0.00492623577703364
1000	132.39094442918679	0.00492623577703364

Fig. 9 Execution Time in ms

From the above output screenshots, it is clear that parallel apriori algorithm which runs on GPU is much efficient than that of serial apriori algorithm which runs only on CPU.

5. CONCLUSIONS AND FUTURE ENHANCEMENTS

This paper focuses on parallelization of apriori algorithm on the GPU with CUDA architecture. To exploit the new parallel platform for data mining, we proposed optimized CUDA-based parallel technique i.e. parallel apriori algorithm. This algorithm shows a speedup over existing serial apriori algorithm. As the size of dataset increases, speedup also increases. GPU with CUDA parallel computing architecture will provide compelling benefits for data mining applications. In addition, its superior floating-point computation capability and low cost will definitely appeal to medium-sized business and individuals. Applications that used to rely on a cluster or a supercomputer to process will be solved on a desktop.

The future work includes parallelizing other FIM Algorithms(Frequent Itemset Mining) such as FPGrowth algorithm and Eclat algorithm on GPU.

6. REFERENCES

[1] Parallel Optimized Algorithm for Apriori Association Rule Mining on Graphics Processing Unit with Compute Unified Device Architecture (CUDA) - By Abhaya Kumar Sahoo , Amardeep Das , Mayank Tiwary - Published in International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE) Volume 3, Issue 10, pp 121-1219, October 2013.

[2] Scalable Frequent Itemset Mining using Heterogeneous Computing: ParApriori Algorithm - By B. B. Meshram and V. B. Nikam - Published in International Journal of Distributed and Parallel Systems (IJDPS) Volume 5, Issue No.5, pp 13-26, September 2014.

[3] Parallel and distributed association mining: a survey. - By Zaki MJ - Published in IEEE Concurrency Volume 7, Issue 4, pp 14-25, October 1999.

[4] Parallel Data Mining on Graphics Processors - By Wenbin Fang, Ka Keung Lau, Mian Lu, Xiangye Xiao, Chi Kit Lam, Philip YangYang,Bingsheng He,Qiong Luo, Pedro V. Sander, and Ke Yang.

[5] GPApriori: GPU-Accelerated Frequent Itemset Mining - By Fan Zhang ,Yan Zhang,Jason Bakos - Published in IEEE International Conference on Cluster Computing,2011.

[6] High Speed Association Rule Mining using Apriori Based Algorithm for GPU - By D.William Albert,Dr.K.Fayaz and D.Veerabhadra Babu - Published in International Journal of Multidisciplinary and Current Research.

[7] Frequent Itemset Mining on Graphics Processors – By Wenbin Fang, Mian Lu, Xiao, Bingsheng He, Qiong Luo.

[8] Data mining: concepts and techniques, 2nd Edition. Morgan Kaufmann, SanMateo - By Han J, Kamber M (2005).

[9] Parallel Computing with CUDA - By Mark Harris (NVIDIA Developer Technology).

[10] Parallel Computing with CUDA - By Mark Harris (NVIDIA Developer Technology).

[11] Professional CUDA C Programming - By John Cheng,Max Grossman,Ty McKercher.

[12] CUDA by Example : An Introduction to General-Purpose GPU Programming - By Jason Sanders,Edward Kandrot.

[13] Parallel Programming with CUDA - By Ian Buck.

[14] Heterogeneous Parallel Programming (MOOC) - By University of Illinois https://www.coursera.org/course/hetero

[15] .Net Framework Programming

[16] Wikipedia, "Association Rule Learning", http://en.wikipedia.org/wiki/Association_rule_learning

[17] Wikipedia, "Apriori Algorithm", http://en.wikipedia.org/wiki/Apriori_algorithm

[18] Wikipedia, "Graphics Processing Unit", http://en.wikipedia.org/wiki/Graphics_processing_unit

[19] Wikipedia, "Parallel Computing", http://en.wikipedia.org/wiki/Parallel_computing

[20] Nvidia, "GEForce 210 Specifications", http://www.geforce.com/hardware/desktop-gpus/ geforce-210/specifications

[21] NVIDIA Developer Zone

[22] http://www.nvidia.com/object/what-is-gpu-computing.html

[23] CUDA Zone

[24] https://developer.nvidia.com/cuda-zone

[25] NVIDIA Parallel For All Blog

[26] http://devblogs.nvidia.com/parallelforall/

7. AUTHOR PROFILE

D. Sirisha working as an Application Developer in Oracle. Completed B.E from CBIT, Hyderabad in Computer Science Engineering. Area of interest is Data Mining and Web Logic.

S. Shahida working as a Technical Associate in Bank of America. Completed B.E from CBIT, Hyderabad in Computer Science and Engineering. Area of interest is Data Mining.

K. Spandana working as an Asst.Prof in CSE Department CBIT.Completed B.tech from VREC , Nizamabad ,after that qualified in GATE with good rank and completed M.Tech from GVPCOE, Vishakapatnam. Attended many seminars and workshops. Area of interest is Distributed Systems.