SURVEY FOR GPU ACCELERATED DATA MINING

Pravin Gurjar[*]

Prof.Varshapriya JN*

Abstract:

Business is highly depended on information. Information always has time as a prime factor. Information with time makes value. The concept of accelerated data mining provides the dual advantage of retrieving more information in less time. GPU accelerated data mining is basically a parallel data mining which uses the processing power of GPU which is also known as GPU computing. Its highly parallel architecture gives the parallelism in a single system.GPU provides massively multi-threaded SIMD (Single Instruction, Multiple-Data) architecture. These processors are highly optimized for graphics rendering and controlled by CPU. We can implement parallel data mining algorithm for the use of GPU and increase the performance of data mining tool.

Keyword: Parallel Data mining, GPU, CUDA, Open CL,

[*] Computer Technology Department, VJTI, Mumbai

I. Introduction

GPU (Graphics Processing Unit) is a processor optimized for 2D/3D graphics, video, visual computing, and display. It is highly parallel, highly multithreaded multiprocessor optimized for visual computing. It provides real-time visual interaction with computed objects via graphics images, and video. It serves as both a programmable graphics processor and a scalable parallel computing platform. Data mining is the task of finding hidden information from data. Various algorithms have been developed for different data mining tasks. If we talk about serial algorithm many serial algorithms have been developed and their parallel forms have also been developed. In this paper I have studied parallel data mining algorithms for clustering. Modern GPU has highly parallel architecture and great computing power so we can use this parallel architecture and computing power to implement these parallel algorithms this will provide higher computing power in limited cost and space. GPUs are basically massively parallel computers. They work well on problems that can use large scale data decomposition and they offer orders of magnitude speedups on those problems. However, individual processing units in a GPU cannot match a CPU for general purpose performance. They are much simpler and do not have optimizations like long pipelines, out-of-order execution and instruction-level-parallelization.

Name	No of core	Memory	Power (Watt)	Cost (Rs.)
AMD 3.6 GHz AM3+ FX4100	4	DDR3-1866 Bandwidth Up to 21 GB/s	95	6046/-
Intel 3.5 GHz LGA 1155 Core i7 3770K	4	DDR3-1333 DDR3-1600 Bandwidth25.6 GB/s	77	20607/-
AMD 3.3 GHz AM3 FX6100	6	DDR3-1866 Bandwidth 21GB/s	95	7758/-

GeForce GT 610 Synergy Edition 2 GB DDR3 Graphics Card	48	DDR3-1066	300	2782/-
AMD/ATI Radeon HD 6670 1 GB DDR3 Graphics Card	480	GDDR5 Bandwidth 64 GB/s	400	4788/-
AMD/ATI HD 7770 Direct CU 1 GB GDDR5 Graphics Card	640	GDDR5 Bandwidth 72 GB/s	500	10486/-
AMD HD 6470Graphics Card	160	GDDR5		5190/-

Table 2.1 CPU GPU Specification

GPUs also have other drawbacks. Firstly, you users have to have one, which you cannot rely on unless you control the hardware. Also there are overheads in transferring the data from main memory to GPU memory and back. So it depends on our requirements in some cases GPUs are clear winner but in other cases our work cannot be decomposed to make full use of a GPU and the overheads then make CPUs the better choice. If we consider other factor like cost, space and power consumption then it is also good to use GPU. We have given some specification of some CPUs and GPUs in Table2.1 first three are CPUs and after this all are GPUs. We have shown number of cores, memory type, memory bandwidth, power required and price for each CPU and

GPU in table. As shown in table 2.1 a 4 core CPU has price Rs.6076/- and a 48 core GPU has price Rs.2072/- so by the given table we can compare other thing also. In the above table we have given cost benefit of GPU over CPU and if we see theoretically GPUs is specially design for parallel computing so it would be faster as compare to parallel CPU. So we will implement parallel data mining algorithm in OpenCL, so that these algorithm can run on GPU and accelerated the data mining in limited cost and space.

II. GPU Architecture

GPU (Graphics Processing Unit) is an integral part of machine. Previously design as co-processor of CPU which is specially aimed for graphics processing but now day's GPUs are highly advance because of faster and higher definition graphics requirement. Semiconductor capability, driven by advances in fabrication technology, increases at the same rate for both platforms. The disparity can be attributed to fundamental architectural differences: CPUs are optimized for high performance on sequential code, with many transistors dedicated to extracting instruction-level parallelism with techniques such as branch prediction and out-of order execution. On the other hand, the highly data-parallel nature of graphics computations enables GPUs to use additional transistors more directly for computation, achieving higher arithmetic intensity with the same transistor count [7]. Graphics processing required fast computation, higher memory bandwidth and better memory transfer speed. These features of GPU make it useful as hardware accelerator in general purpose computing that why it is also called GPGPU (General purpose GPU). We are describing the architecture of an Nvidia GPU GeForce 8800 as shown in figure 2.1 it has

- 16 Multiprocessors Blocks
- Each MP Block Has:
    - 8 Streaming floating point processors @1.5Ghz
    - 16K Shared Memory
    - 64K Constant Cache
    - 8K Texture Cache
- 1.5 GB Shared RAM with 86Gb/s bandwidth
- 500 Gflop on one chip (single precision)

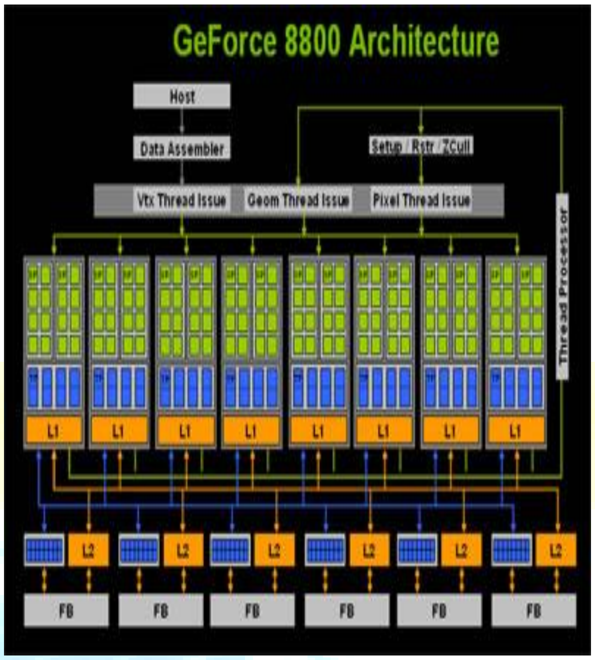

Figure 2.1 GeForce 8800 Architecture

Figure 2.2 Geforce 8800 GPU

Figure 2.2 shows architecture of a single multiprocessor block each multiprocessor block has its own texture address texture filtering unit and separate caches.

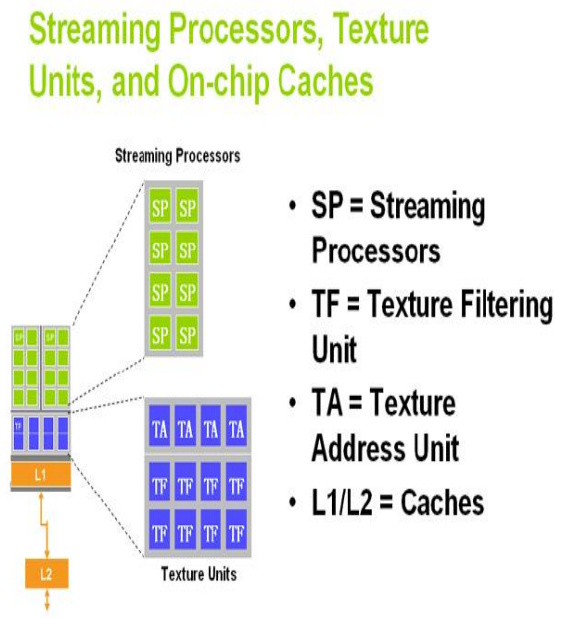
Streaming Processors, Texture Units, and On-chip Caches
Streaming Processors
SP
TA
TF
L1
L2
Texture Units
• SP = Streaming Processors
• TF = Texture Filtering Unit
• TA = Texture Address Unit
• L1/L2 = Caches

Figure 2.3: Stream Processor

GPU GeForce 8800 has 16 such multi processing unit. In this each multi processor block has its own cache which is limited in their context during GPU computing.

III. GPU Computing

GPU computing is nothing but the uses of graphics-processing unit (GPU) to accelerate compute-intensive-tasks. GPU's rapid evolution from a configurable graphics processor to a programmable parallel processor, the ubiquitous GPU in every PC, laptop, desktop, and workstation is a many-core multithreaded multiprocessor that excels at both graphics and computing applications. Today's GPUs use hundreds of parallel processor cores executing tens of thousands of parallel threads to rapidly solve large problems having substantial inherent parallelism which is proving very useful in today's high performance computing.GPU computing has become an important part of high performance computing. The programmable units of the GPU follow a single program multiple-data (SPMD) programming model. For efficiency, the GPU processes many elements (vertices or fragments) in parallel using the same program. Each element is independent from the other elements, and in the base programming model, elements cannot communicate with each other [8].CUDA and OpenCL are two different interfaces for programming GPUs.

A. CUDA

In November 2006, NVIDIA introduced CUDA™, a general purpose parallel computing architecture – with a new parallel programming model and instruction set architecture – that leverages the parallel compute engine in NVIDIA GPUs to solve many complex computational problems in a more efficient way than on a CPU. CUDA comes with a software environment that allows developers to use C as a high-level programming language. CUDA (Compute Unified Device Architecture) is a proprietary API and set of language extensions that works only on NVIDIA's GPUs. The CUDA parallel programming model is designed for transparently scales application's parallelism to leverage the increasing number of processor cores on many core GPU.CUDA program can execute on any number of processor cores as illustrated by Figure 1-4, and only the runtime system needs to know the physical processor count.

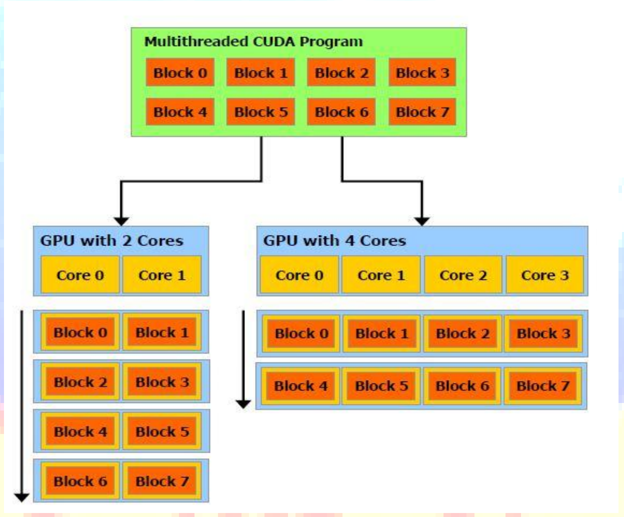

Figure 3.1 partitioning of threads

Figure 3.1 shows a multithreaded program is partitioned into blocks of threads that execute independently from each other, so that a GPU with more cores will automatically execute the program in less time than a GPU with fewer cores. CUDA C extends C by allowing the programmer to define C functions, called kernels, that, when called, are executed N times in parallel by N different CUDA threads, as opposed to only once like regular C functions. The following sample code adds two vectors A and B of size N and stores the result into vector C.

```
// Kernel definition
__global__ void VecAdd(float* A, float* B, float* C)
{
```

```
inti = threadIdx.x;
C[i] = A[i] + B[i];
}
int main()
{
...
// Kernel invocation with N threads
VecAdd<<<1, N>>>(A, B, C);
}
```

B. OpenCL

OpenCL (Open Computing Language)by the Khronos Group, is an open royalty-free standard for general purpose parallel programming across CPUs, GPUs and other processors, giving software developers portable and efficient access to the power of these heterogeneous processing platforms. OpenCL consists of an API for coordinating parallel computation across heterogeneous processors and a cross-platform programming language with a well specified computation environment. The OpenCL standard

- Supports both data- and task-based parallel programming models
- Utilizes a subset of ISO C99 with extensions for parallelism
- Defines consistent numerical requirements based on IEEE 754
- Defines a configuration profile for handheld and embedded devices
- Efficiently interoperates with OpenGL, OpenGL ES and other graphics APIs

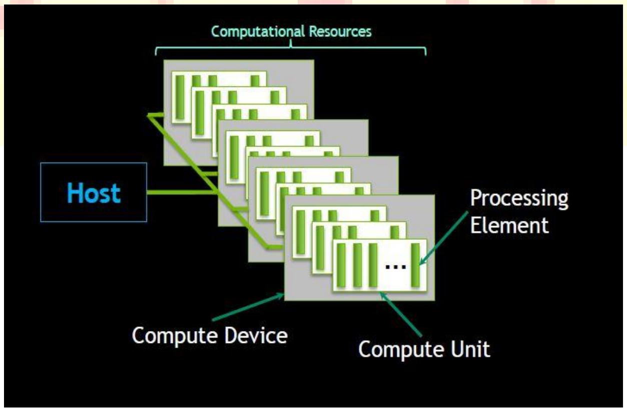

Figure 3.2 OpenCL platform model

Figure 3.2 shows the platform model of OpenCL here CPU works as a host and other processing device (GPU) work as device. Here each compute device has many compute units which will work in their context and share data. Host will direct device to execute instruction so we can say that under the guidance of CPU, GPU will work.

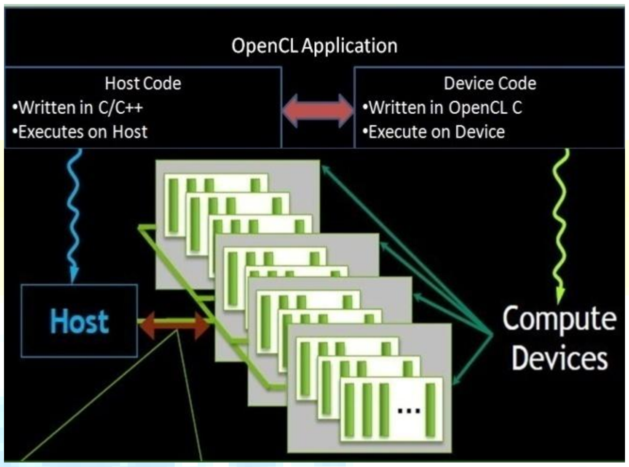

Figure 3.3 OpenCL Application Anatomy

Figure 3.3 is an OpenCL application anatomy which has two parts one is Host code which is serial and execute on CPU and other is Device code which is parallel and execute on GPU cores.

CUDA terminology	OpenCL terminology
GPU	Device
Global Memory	Global Memory
Constant Memory	Constant Memory
Shared Memory	Local Memory
Local Memory	Private Memory
Thread	Work-item
Thread-block	Work-group

Table 3.1 A COMPARISON OF GENERAL TERMS

IV. GPU Based Data Mining Algorithms

K-means is one of the most well-known and commonly used clustering algorithms. It takes an input parameter k, and partitions a set of n objects into k clusters according to a similarity measure. Each object has multiple attributes, or features. The mean values, or centroids , are a summary measure of the similarity of data objects within the same cluster. In parallel k-means, data parallelism is used to divide the workload evenly among all processes. Data objects are statically partitioned into blocks of equal sizes, one for each process. Since the main computation is to compute and compare the distances between each object and the cluster centroids, each process can compute on its own partition of data objects independently if the k cluster centroids are maintained on all processes. The algorithm is summarized in the following steps.

(1) Partition the data objects evenly among all processes;

(2) Select k objects as the initial cluster centroids;

(3) Each process assigns each object in its local partition to the nearest cluster, computes the SSE for all local objects, and sums up local objects belonging to each cluster;

(4) All processes exchange and sum up the local SSE's to get the global SSE for all objects and compute the new cluster centroids;

(5) Repeat (3) - (5) until no change in the global SSE.

V. Conclusion

In this paper we have studied how GPU make revolutionary change in the area of High performance computing and some data mining algorithm. Parallelizations of data mining algorithms on CPU already have been done. But CPU is more costly and rapidly use of GPU in general purpose computing motivates us to do parallelization on GPU so it would be more time and cost effective. Data mining is the prime factor of today's business perspective. In today's age of fastest growing technology time is precious tool. If we combine these two things in single one, we can make a better future for the trading business.

VI. References

1. D. Foti, D. Lipari, C. Pizzuti And D. Talia **“Scalable Parallel Clustering For Data Mining Onmulticomputers**”
2. Jianwei Li,Ying Liu, Wei-Keng Liao,Alok Choudhary “**Parallel Data Mining Algorithms For Association Rules And Clustering**”
3. Wenbin Fang, Ka Keung Lau, Mian Lu, Xiangye Xiao, Chi Kit Lam, Philip Yang Yang, Bingsheng He, Qiong Luo, Pedro V. Sander, and Ke Yang “**Parallel Data Mining on Graphics Processors**”
4. Wenjing Ma, Gagan Agrawa “**A Translation System for Enabling Data Mining Applications on GPUs**”
5. Shengnan Cong, Jiawei Han, David Padua “**Parallel Mining Of Closed Sequential Patterns**”
6. Xiaohong Qiu1, Geoffrey Fox, George Chrysanthakopoulos, Henrik Nielsen “**Parallel Data Mining on Multicore Clusters”**
7. John D. Owens, David Luebke, Naga Govindaraju **“A Survey of General-Purpose Computation on Graphics Hardware”**
8. Mohammed J. Zaki, **“Parallel and Distributed Association Mining a Survey”**
9. **http://www.nvidia.com**
10. **http://www.amd.com**