International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 02 Issue: 03 | June-2015 www.irjet.net p-ISSN: 2395-0072

Augmenting Speed of SQL Database Operations Using NVIDIA GPU

Sandip M. Walunj[1], Rajendra A. Patta[2], Anuraj R. Kurup[3], Hrishikesh S. Bajad[4]

[1,2,3,4] *Department of Computer Engineering, Sandip Institute of Technology & Research Centre, Maharashtra, India*

Abstract- ***In the early stages GPU were used to develop graphic processing algorithms, since the development of new parallel computing frameworks and the recent development in the GPU architectures non-graphic algorithms can also be implemented. Various advancements in the parallel databases have proved to be more efficient over primitive CPU based databases. This system is developed using CUDA framework provided by NVIDIA which is more approachable and appreciated by programmers. The main focus of the automated system is to optimize SQL SELECT queries by offloading it to the GPU. Algorithms were implemented on different data sets with the assistance of a programmable GPU. The results of the implemented system implies that using GPU as a co-processor, a considerable amount of improvement in the execution of database operations were achieved compared to the traditional CPU based database systems. The results obtained by the system show speedup of about 20x. These results show that GPU can be exploited to achieve high level of parallelism and throughput in the execution of database operations.***

Key Words: *CUDA, Databases, GPU, Parallelism, Query processing, SQL.*

1. INTRODUCTION

Originally GPUs were developed purely for the function of graphic acceleration. Due to the recent developments in the architecture of GPUs vast numbers of additional computational tasks are processed with the help of GPUs. While CPUs were built to process 4 or maybe 8 threads at a time, GPUs were developed with the perspective of simultaneously managing thousands of threads efficiently and provide with throughputs in the range of 100 GBps [2]. Thus more number of programmers are using this computation power to accelerate different applications. These accelerations are observed considering the fact that the data has to be transferred to the GPU from the memory and back for processing.

The task of parallelizing the database operations was achieved using the CUDA framework developed by NVIDIA which supports the stream programming paradigm. The framework provided in CUDA provides the functionality of executing a single kernel function multiple number of times simultaneously by converting each call into a thread and process on a block of data. Stream programming is achieved by organizing large number of processor cores of the GPU into streaming multiprocessor groups. The limitations occurring in these kind of processing is the limited memory size and data transfer between host and GPU which can be overcome by the future upcoming architectures in the market. A wide number of applications can fit into the domain of this research. Also, SQL is a widely popular database query processing language is used which is suitable for database processing. Complex relational operations on data set like join and aggregation can easily be implemented using SQL. Little or no change has to be done in the source code by the programmer to implement it thus making the task of parallelising the query execution very simple.

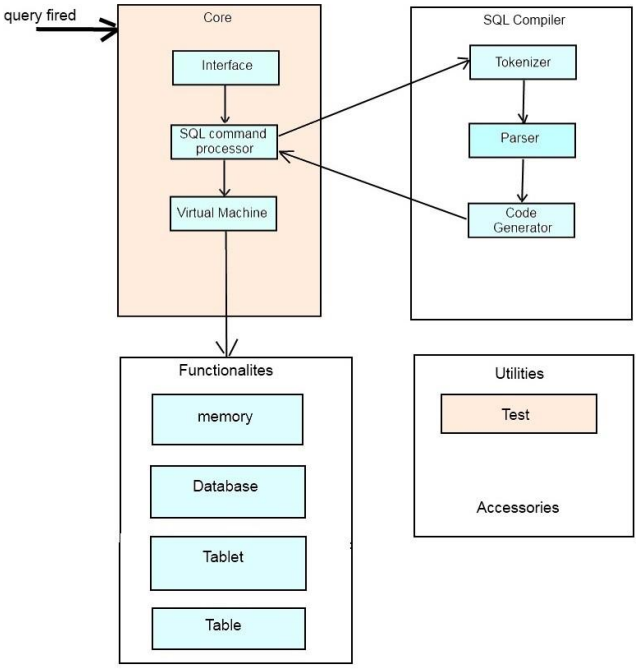
Through the implementation of the suggested system it is demonstrated that database operations on huge data sets can be accelerated by the using the GPU as a co-processor. The comparison between the results obtained from the multi-core CPU based database implementations and the highly optimized GPU based database system implemented in this research. The results are based on basic SELECT queries which are compiled into an opcode language which can be executed either on a CPU or a GPU depending on the requirement. The system not only simplifies the GPU data processing but also the results obtained by the system suggests that GPU based systems significantly outperforms the conventional ordered CPU execution.

2. METHODOLOGY

The system includes a query parser which accepts SQL queries in the form of input and the output will be in the form of intermediate code or opcode. The virtual machine will execute the opcode, process the records and generate a result set. The Tablet structure will divide the datasets for compatibility with the GPU memory. This will help us to run queries on datasets larger than the GPU memory the main objective of our system is to compare execution on CPU and GPU. This can be done by building an API for executing certain SQL queries on GPU.

The system model developed in this paper consists of a Query Parser which accepts SQL queries as input and provides program in discrete steps called opcodes as output. The opcodes are queries broken down into steps consisting of four arguments with the first three

being integer values and the fourth being any type. It has a Virtual Machine to execute the opcode program, to process data records and gives a result set as output.



The Virtual Machine is written on GPU as well as CPU. The work of managing groups of tablets in tables is done by Table Functionality and the code used to manipulate so as to add and read data to the tablet data structure is done by Tablet Functionality. The Tablet structure divides the datasets for compatibility with the GPU memory. This helps in execution of queries on datasets larger than the GPU memory. It consists of a module of Memory Functionality which manages the tablets in memory space which is pre-allocated. Database functionality is used to assure non-volatile data storage by writing metadata into a database file. It also does the work of paging tablets in between disk and memory. The speed comparison between the GPU and CPU can be done to give result analysis.

3. LITERATURE SURVEY

On the basis of previous work in the foundational fields to study what research has been done in the field of general GPU Databases applications is important. The general purpose GPU Databases have been studied and developed in two different classifications. Few of the study have been partial implementations whereas a few were development to existing traditional database systems which explored a particular aspect of the database system. One another field in which study has been done mainly is implementation of database systems which are done on the GPU itself. Some of the additions have been done with the help of PostgreSQL database system [3]. The functions of GPU are written as procedures which are called by PostgreSQL. A particular procedure is called to manipulate an image from a variety of images which are retrieved from the database. It is done by GPU-powered procedure which is stored in the GPU and the results are provided to the users.

One of the areas where research has been done is relational joins [4]. This showed the potential of the GPU in high speed execution of joins of multiple tables. In this research, database primitives for scatter, gather, split, scan and sort operations which were used for developing join algorithms as constructs which showed that using the GPU to perform operations such as join was feasible.

Chang demonstrated that by using column-major storage for the database tables greatly improved the overall performance of data transfer from and to the GPU [5]. In the study existing functions of CUDA and its libraries along with newly added data structures as well as parameters to the existing vanilla SQLite implementation to replace serial functionality which existed with the help of GPU powered parallel functionality. The process helped in acceleration of SELECT queries on data of integer value in a table as well as sorting and grouping of data by using this functionality. One of the most important aspects of the research was the discovery that for small number of records GPU implementation was slower as compared to CPU implementation.

In 2004, Govindaraju along with few others were the first few who investigated GPU as coprocessor for query processing in a database [6]. They were able to accelerated database operations like selections and aggregations. The efficiency of GPUs to perform general purpose work using CUDA or other libraries and frameworks provided new approaches for development. He and others investigated algorithms for join processing in databases with the help of GPUs [7], as well as for few other relational operators [8]. They also investigated optimization of transactions by using GPUs in the prototype GPUTx. They grouped multiple transactions together and executed all of it concurrently on the GPU. The observation was an improved throughput of 4 to10 times compared to a CPU-based processing [9].

4. IMPLEMENTATION

Our model of execution involves the transfer of dataset which is residing on CPU to GPU. Here each row of dataset is assigned to a thread for parallel execution. The identity of each row is maintained by thread indexing. it includes partition of rows to available threads and process them on SELECT WHERE condition. The result set is generated by synchronizing the threads and sending it back to the CPU. Our approach focuses on implementing the dataset by using indexes.

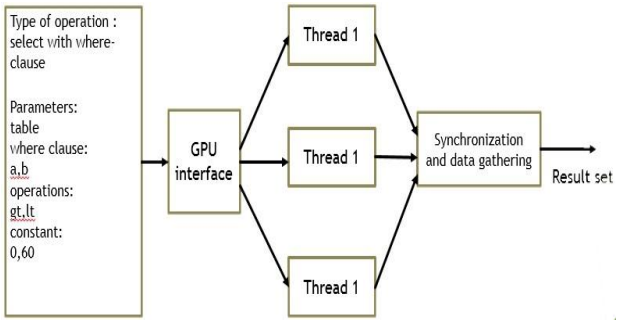


Fig 2. GPU Processing

Results Analysis

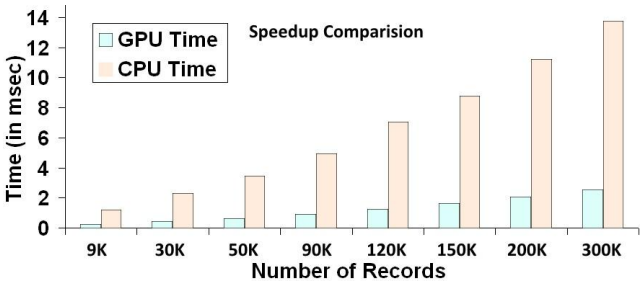


Fig 3. Speedup comparison- CPU Vs GPU

Speedup of about 20x over CPU query execution was observed overall as seen in *fig 3*. The speedups are calculated considering the data transfer time between CPU and GPU along with the computation time. It can be observed that with the increase in size of data set, the GPU performance shows considerable growth.

Thus when the dataset size is relatively small CPU based processing is advised over GPU whereas in case of working with massive datasets GPU based processing would be a finer solution.

Table 1. Performance Analysis

Queries attribute	CPU Time (sec)	GPU Time (sec)	Transfer Time (sec)	Speedup w/o Transfer	Speedup with Transfer
Int	2.384	0.056	0.025	48.11	18.89
Float	3.527	0.059	0.021	62.16	22.68
Char	1.056	0.029	0.036	40.02	16.19

The performance analysis as seen in *Table 1* shows various speedups based on the query attributes of different data types. The table provides us with the statistical data of processing time on a CPU as well as on a GPU. The data transfer time is calculated to measure the speedups with and without the transfer time. It is evident from the table above that transfer between the GPU and CPU is a huge bottleneck in terms of speedups. In spite of the latency due to transfer from GPU to CPU and vice versa, the speedups are quite impressive.

5. CONCLUSIONS

The findings of this paper suggests that, by offloading the database queries to the GPU, it is possible to achieve effective acceleration of database operations. The results obtained by the GPU based optimization showed a progressive speedup of about 20x on an average as compared to the CPU based processing. It can also be concluded that when a dataset size is comparatively small, CPU based processing should be used whereas in case of massive datasets GPU based processing provides great speedup with optimized result. Even though only a SELECT SQL queries are considered in the paper, it provides bases to show that other queries can also be implemented to achieve similar speedups. SQL is a simpler and more widely used system which can be used to gain access to GPU. The features of each query along with the data type being queried as well as the result set size were important aspects in the comparison of CPU and GPU execution factor. Based on the results of our system, it can stated that the data transfer between GPU and CPU memory is not much of a significant barrier in data processing with GPU in comparison to the results obtained still it has to reduce to an extent to provide better results than results already obtained. The results of the system are intended to become better with upcoming series of NVIDIA GPUs. Future research could improve the latency occurred due to transfers from host to GPU memory. Another prospective is that the future of upcoming graphic processing hardware is able to access the host memory more conveniently which would enhance the current results. Although a lot of future study is required, it can be stated that with the help of GPU hardware SQL query processing can be accelerated significantly.

REFERENCES

[1] Rajendra A. Patta, Anuraj R. Kurup, Sandip Walunj, "Enhancing Speed of SQL Database Operations using GPU," in IEEExplore International Conference on Pervasive Computing, Pune, India 2015, DOI 10.1109/PERVASIVE.2015.7087144.

[2] Peter Bakkum, Srimat Chakradhar,"Efficient data management for GPU databases" NEC labs, Princeton, USA, Mar. 2010

[3] Tim Child, Parallel Image Search Using PostgreSQL and PgOpenCL Presentation, PGCon 2011: The PostgreSQL Conference, Ottawa, 2011.

[4]Bingsheng He, Ke Yang, Rui Fang, Mian Lu, Naga K, Govindaraju, Qiong Luo, and Pedro V, Sander, "Relational Joins on Graphics Processors," 2008 ACM SIGMOD International Conference on Management of Data Vancouver, 2007, pp. 511-524

[5] R. Kai Sheu, Y. Chang, J. Hsu and S. Yuan, "Scaling Database Performance on GPUs," in Information Systems Frontiers vol. 14, 2012, pp. 909-924.

[6] N. K. Govindaraju, Lloyd B., Wang W., Lin M., and Manocha D, "Fast computation of database operations using graphics processors," in ACM SIG GRAPH 2005 Courses, NY, 2005, pp. 206.

[7] He B. et al., "Relational query co- processing on graphics processors," ACM Trans. Database System., vol. 34, no. 4, 2009, pp. 139.

[8] B. He and J. X. Yu. "High-Throughput Transaction Executions on Graphics Processors," PVLDB, vol.4, no. 5, 2011, pp. 314-325.

[9] B. He and J. X. Yu. High-Throughput Transaction Executions on Graphics Processors. PVLDB, 4(5):314{325, 2011.

[10] P. Trancoso, D. Othonos, and A. Artemiou, "Data parallel acceleration of decision support queries using cell/be and GPUs," in Proc. of 6th ACM Conf. on Computing frontiers, ACM, 2009, pp. 117–126.

[11] A. di Blas and T. Kaldeway, "Data monster Why graphics processors will transform database processing," IEEE Spectrum, Sept. 2009.

BIOGRAPHIES

Prof. Sandip M. Walunj,
Assistant Professor,
Department Computer Engineering,
Sandip Foundation's
Sandip Institute of Technology & Research Center, Nashik

Rajendra A. Patta
Department Computer Engineering,
Sandip Foundation's
Sandip Institute of Technology & Research Center, Nashik

Anuraj R. Kurup
Department Computer Engineering,
Sandip Foundation's
Sandip Institute of Technology & Research Center, Nashik

Mr. Hrishikesh S. Bajad,
Government College of Engineering,
Amravati, Maharashtra
email: hrishikeshsbajad@gmail.com

