IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 4, Ver. II (July – Aug. 2015), PP 57-63
www.iosrjournals.org

Parallelizing Graph Algorithms on GPU for Optimization

Trupti R. Desale
(Computer Department, MCOERC, Nashik, Savitribai Phule Pune University, India)

Abstract : *Many practical applications include image processing, space searching, network analysis, graph partitioning etc. in that large graphs having a millions of vertices are commonly used and to process on that vertices is difficult task. Using high-end computers practical-time implementations are reported but are accessible only to a few. Efficient performance of those applications requires fast implementation of graph processing and hence Graphics Processing Units (GPUs) of today having a high computational power of accelerating capacity are deployed. The NVIDIA GPU can be treated as a SIMD processor array using the CUDA programming model. In this paper Breadth-First Search and All Pair shortest path and traveling salesmen problem graph algorithms are performed on GPU capabilities. The algorithms are introduced to optimize such that they can efficiently adopt GPU. Also an optimization technique that reduce data transfer rate CPU to GPU and reduce access of global memory is designed to reduce latency. Analysis of All pair shortest path algorithm by performing on different memories of GPU which shows that using shared memory can reduce execution time and increase speedup over CPU than global memory and coalescing access of data. TSP algorithm shows that increasing number of blocks and iteration obtained optimized tour length.*

Keyword: *Graphics Processing unit, CUDA, BFS, All Pair Shortest path, TSP, Graph processing, optimization.*

I. Introduction

Graphs data structure are mainly used to store data in many practical applications like image processing, data mining, space searching, network analysis, social networking etc. In this application developers use graph operations as a fundamental tools. Graph operation like breadth first search, single source shortest path, all pair shortest path etc. Improve performance of graph processing for better efficiency of practical application. Graph processing is one of the integrative and important research area.

Graph theory is a branch of mathematics that studies and models pair wise relation between elements of set wise objects. Graph refers to collection of nodes and their relation through edges. Each edge connects pair of nodes. Graphs represent through G=(V,E) where V represents set of nodes and E represent set of edges. BFS problem is, given graph G(V,E) and source S, find the minimum number of edges needed to reach every vertex V in G from source vertex S. All Pair shortest path(APSP) problem is, given weight graph and in that find smallest path between each pair of vertices. Travelling Salesmen Problem (TSP) is, given graph G and source S, find the minimum cost tour length where all vertices are visited at least once.

In past year there are many approaches to accelerate graph algorithm. Fast implementation of sequential graph algorithm[1] exit but algorithm become impractical on very large graph. Then parallel algorithm which achieves practical times on basic graph operation but required high hardware cost. Bader et. al[2] perform graph algorithm by using CRAY supercomputer while this method is fast, but hardware used in them is very expensive.

Now a days the increasing adoption of GPGPU (General-Purpose computation on Graphics Processing Unit) in many application[3]. To accelerate various graph processing application the GPU has been used. While those, GPU-based solution give a significant performance improvement over CPU-based implementation. In the propose system graph algorithm are implement in parallel approach through the use of GPU.

In this paper we proposed the graph algorithms which are Breadth-First Search and All Pair Shortest Path and Travelling Salesmen Problem through use of GPU .To efficiently utilized GPU power, proposed algorithm usages several optimization technique like reducing global memory accessing, and modify algorithm such that it use a maximum computing capacity of GPU and also more use of GPU texture memory for data accessing to threads because it require less latency than other.

The contributation of this work include :

- The implementation of GPU based graph Algorithm which is BFS, APSP and TSP.
- Improving Performance of GPU using optimizing graph algorithm through the use of memory hierarchy of GPU.
- Reducing data transfer rate from CPU to GPU.

The remaining paper is organized as follows. Section II related work of graph algorithm and Section III Introduced NVIDIA GT 720M architecture. In Section IV describe implementation details of proposed system. In section V we discuss our Dataset and performance comparison of serially implemented BFS algorithm and we conclude in section VI.

II. Related Work

2.1 Breadth-First Search Algorithm

The Breadth first search (BFS) has number of applications in different areas. These include image processing, space searching, network analysis, graph partitioning, automatic theorem proving etc. The BFS aims to find out the minimum number of edges required to reach each vertex V in G from source vertex S. The best time complexity reported for sequential algorithm is O(V+E).

A cost effective parallel platform is provided by using graphics hardware to solve many general problems. Many problems are benefited from GPU in speed and parallel processing. Harish and Narayanan proposed accelerated large graph algorithm using CUDA[4]. This method is capable of handling large graphs, for GPU implementation. Here in BFS, one thread is assigned to every vertex. Frontier array, visited array and integer array as cost c which stores the minimum count of edges of every vertex from the source vertex S. each vertex looks at frontier array if true, then update the cost c of its and neighbors. But some cases like scale free graphs BFS works slower because of the large degree at few vertices, loop inside the kernel which causes the more lookups to device memory and slowing down the kernel execution time.

In the BFS implementation used by Vibhav et al. [5] performed on vertex compaction process with the help of prefix sum which assign threads only for active vertices. For removing unnecessary threads vertex compaction process is very useful. At particular time, small number of vertices may be active. They carried out experiments on various types of graphs and compared the results with the best sequential implementation of BFS and experiment shows lower performance on low degree graphs. Lijuan luo[6] proposed effective GPU implementation of Breadth-First Search. In this paper a hierarchical technique to designed efficiently implement a queue structure on the GPU. To reduce synchronization overhead a hierarchical kernel arrangement was used. Their experimental result showed same computational complexity as fastest CPU version and achieved up to 10 times speedup.

Hong, kim implemented a novel wrap-centric [7] programming method that reduces the inefficiency in an intuitive but effective way that exposes the traits of underlying GPU architecture to users. Their experimental result showed significant speedup against pervious studied GPU implementations as well as multithreaded CPUs.

2.2 Parallel Approaches for all pair shortest path

In all pairs shortest path problem (APSP), given an weighted graph G(V, E, W) with positive weights, and that aim is to find out least minimum weighted path from each & every vertex to every other vertex. Floyd-Warshall's, the well known APSP algorithm.

Micikelvicius[8] proposed to solve all pair shortest path using graphics hardware. In that unique distance matrix entry corresponded to each pixel, so to perform Floyd-warshall algorithm used fragment shader. But this algorithm cannot work on large graph. The Harish and Narayanan[3] proposed graph algorithm that is Floyd-warshall's all pair shortest path algorithm requires O(V3) time and O(V2) space. Here used a adjacency matrix for graphs and Floyd warshall algorithm implemented using O(V) threads, each running a loop same size inside it. This approach is slower because of sequential access of entire vertex array by every thread. Other approach is to running single source path to every vertex. This methods require O(V) threads where Floyd warshall's algorithm require O(V2)threads and which creates extra overheads for context switching for threads.Gary J. Katz and Joseph T. Kider proposed all pair shortest path for large graph[9]. Here graph size problem due to memory availability is solved. Their approach handles graph size larger than GPU on board available memory this achieved through breaking the graphs into blocks. Convert into blocks in nontrivial on-chip shared memory cache to increase the performance in efficient manner. The algorithm is implemented by blocked formulation.

The basic idea for implemented algorithm is revise original Floyd warshall algorithm into hierarchically parallel methods which can be distributed across on GPU with multiple processors. Matrix is divided into sub blocks with equal size then processed. This implementation of algorithm provides 60-130x speedup over a standard CPU solution O(V3). 45-100x speedups to blocked CPU implementation that specified by Venkataraman et al. [10] also this methods provides speedup of 5.0-6.5x compared to standard GPU implementation [3]

2.3 Parallel Approach for Traveling Salesmen Problem

The traveling salesmen problem is mostly studied as a combinatorial optimization problem. The TSP problem, given an weighted graph and that aims to find out optimized tour length with less amount of time. TSP problem is solved with factorial algorithm complexity which motivated the research on two typed of algorithm, exact algorithm and heuristics algorithm. Through the use of branch and bound, linear programming algorithm finding optimal solution. These techniques are usually hard to parallelize. TSP heuristics algorithms based on genetic and evolutionary algorithms [15], simulated annealing [16], Tabu search, ant systems and that heuristics solution are approximation algorithm and reach to approximate solution means it is closer to the optimal solution.

Heuristic algorithm generates an initial solution randomly and then attempt to improve it by using heuristic techniques until get locally optimal solution. O'Neil et. al [17] describe solution to the traveling salesmen problem by evaluating parallel implementation of iterative hill climbing with random restart for getting high quality solution.

Another approach to parallel GPU optimization is given by Luong et al. [18] here, random solution is taken and building a neighborhood by pairwise exchange operations and evaluation of candidate solution and put result into fitness structure and copy that on CPU and repeat the steps select optimized solution. Here, in this approach data transfer rate CPU to GPU is high hence in the proposed algorithm.

III. Details of Proposed System

The focus of the proposed work is to show the advantages by performing graph algorithm on GPU for optimization. Graph algorithms that is graph operations performed as fundamental tools in many application hence to increase speed of application require to reduce execution time of graph algorithm. Here BFS, APSP and TSP algorithm performed.

3.1 Problem Definition

Given a graph G as input to BFS, APSP and TSP graph algorithm. BFS for visited node in minimum time. APSP algorithm performed on GPU memories that is global memory, shared memory and coalescing access of data and analysis of result. Traveling salesman Problem performed on GPU by reducing global memory access and performed operation on GPU itself hence, to reduce execution time and transfer rate and find minimum tour length.

3.2 Proposed System Architecture

In proposed system NVIDIA GT 720M GPU is used to perform a parallel graph algorithm. In that graph data are stored in main memory that converts into an adjacency list. Fig.1 shows a system architecture diagram of proposed system. In that CUDA processing flow are as follows 1st host allocate device memory for graph data. The processing data transfer from main memory to GPU memory. In the 2nd stage CPU instruct the GPU for processing, in that CUDA Kernel function defined for BFS kernel function that execute on each vertex and in single source shortest path algorithm subgroup the vertex that work in parallel approach. In the 3rd stage thread execution manager of GPU executes kernel function in all cores of GPU in parallel. At the last collect result from all threads and transfer to CPU.

Each algorithm explained in detail below:

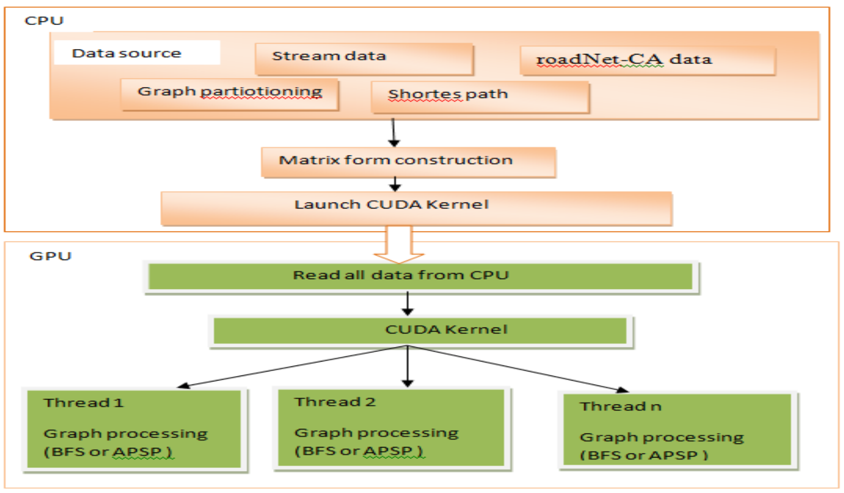

Fig.1. Block Diagram of Proposed System

3.2.1 Parallel Approach for BFS Algorithm

Parallel BFS uses layer synchronization to parallelize breadth-first search. S be a source vertex then define layer or search minimum distance to V that is meet in minimum number of edges. Parallel BFS algorithm work as follows:

```
Input: G (V, E), vertex array V, Frontier array F, Visited array    X, edge array E, cost array C
Output: Search vertex
 Begin:
    •  Initialize F and X as false and C to ∞
    •  F[S] ← true  and C[s] ← 0
    •  While frontier array not empty do
o   For each vertex V in parallel do
                            Calculate connected number of edges
                            Assign thread to each edge.
                            Each threads visit vertices
                            If check whether already not visited
                                  C[Vi] ← C[Vi-1]+1
                                  And store in visited vertex X
                            Else
                                  Terminate thread
o   End for
    •  Gather result from all threads
    •  Display result.
```

3.2.2 Parallel Approach for All Pair Shortest Path Algorithm

In the proposed algorithm find all pair shortest path, here convert graph into adjency matrix and then perform algorithm by using different memories of GPU like global memory, texture memory, shared memory and analysis result.

Parallel All pair shortest path algorithm work as follows:

Input: Graph G(V,E)

Output: all pair shortest path in adjency matrix

Begin:

- Convert graph data into adjency matrix
- Allocate memory for problem inputs and solution in GPU
- Copy problem input to GPU memory
- Parallel do find the distance from node to all other distance if that distance is less than previous then update distance between pair of vertices.
- Copy that resulted matrix to the CPU
- Display result.

3.2.3 Parallel Approach for Traveling Salesmen Problem

In the proposed system finding minimum tour length with less amount of time where all vertices visited atleast once. In the proposed algorithm traditional local search metaheuristic method is used to select tour. In the TSP algorithm, thread allocation is based on thread control function because if threads are not used then data transfer rate and allocation of thread is waste hence only require numbers of threads to be created. Parallel TSP work as follows:

Input: graph G(V,E)

Output: optimized tour length

Begin:

```
•  Choose an initial solution
•  Evaluate solution and LSM initialization
•  Allocate memory on GPU for problem input, solution, fitnesses structure and additional structure
•  Copy problem inputs, initial solution and additional structure on GPU memory.
•  For each neighbor in parallel do
        Evaluation of neighbor candidate solution and insert the resulting fitness into fitness structure.
•  Solution selection strategy is applied to the fitness structure and new candidate solution has been selected.
•  Finding neighbor of new candidate solution and repeat step 6 and 7upto stopping criterion satisfied.
•  Copy chosen solution on CPU and display minimum tour length.
```

IV. Experimental Setup and Results

4.1 Experimental Setup

The experimental setup to perform graph algorithms on GPU for optimization, Here for Breadth First search algorithm are performed on varied number of nodes graph such that 6, 4096 and 65536 nodes of graph. To performing All Pair Shortest Path 4, 10 and 50 numbers of nodes of graphs are used. And to performing Traveling salesman problem TSPLIB dataset is used, there instances like pcb442, rat783 etc. Here I3 CPU is used and nVIDIA gpu is used with CUDA 5.5 and linux

Harish's work provides an open source implementation of BFS using CUDA and we use it as basic implementation. The underlying difference between basic BFS implementation and optimized BFS is that optimized applies L threads to vertex if that vertex has L edges, while in basic BFS threads applied to vertex only. As result, table 1 shows that optimized algorithm require less execution time than basic BFS implementation.

Table 1: Comparison of basic BFS and optimized BFS implementation

Dataset Name	Number of nodes	Serial Algorithm	Parallel Algorithm
TSP	6	2sec	24us
coPapersCiteseer (Cite)	4069	4sec	191us
Huge	65536	11sec	1611us

All Pair Shortest Path algorithm performed on 10 and 50 number of nodes of graph on CPU and GPU and also performed APSP algorithm on GPU by using global memory, shared memory and coalescing access of data. Comparing result by using required execution time to performed each method. Table 2, result or comparison shows that APSP required less execution time when using shared memory of GPU to stored and access problem inputs from it.

Table 2: Comparison of APSP execution time CPU vs GPU vs GPU-Coalescing vs GPU-Shared memory

	Execution time in millisecond(ms)									
No. of Nodes	**10**					**50**				
No of Blocks	**8**	**16**	**32**	**64**	**128**	**8**	**16**	**32**	**64**	**128**
CPU	0.175	0.171	0.168	0.170	0.237	13.749	13.373	13.436	13.542	13.752
GPU	0.070	0.069	0.070	0.043	0.044	9.244	9.165	9.098	9.053	9.001
GPU-Coalescing	0.076	0.060	0.074	0.048	0.047	8.301	8.038	7.986	7.96	7.77
GPU-Shared memory	0.058	0.059	0.057	0.029	0.031	7.410	6.915	6.676	6.526	5.008

Test the performance of Traveling Salesman Problem on GT 820M GPU. Datasets of varying number of cities ranging from 127 to 24978 are tested with varying number of blocks. Increase the number of blocks means iteration of finding neighbor solution is increase hence, probability will increase to get optimized tour length. Table 3 shows that, for higher number of blocks requires more execution time but it gives better optimized tour length. Proposed algorithm addresses the effectiveness of the solution.

Table 3: TSP parallel implementation on Bier127 data on different number o blocks

Dataset	No of Cities	No of Blocks	Time	Cost of Tour
Bier127	127	100	0.0139 s	119892
		10000	1.0561 s	119109
		20000	1.957 s	118607

Table 4: TSP parallel implementation on CH130

Dataset	No of Cities	No of Blocks	Time	Cost of Tour
CH130	130	100	0.0146 s	6321
		200	0.0290 s	6263
		400	0.0551 s	6258
		500	0.0600 s	6258
		1000	1.0689 s	6159
		2000	2.1223 s	6157

Table 5: TSP parallel implementation on PR1002

Dataset	No of Cities	No of Blocks	Time	Cost of Tour
PR1002	1002	**100**	1.5762 s	278381
		200	3.2850 s	278621
		400	4.8314 s	276084
		500	29.1702 s	274778
		1000	144.6441 s	274025

Table 6: TSP parallel implementation on datasets

Dataset	No of Cities	No of Blocks	Time	Cost of Tour
Usa13509	13509	2	2310.1165 s	22090071
		50	9321.2440 s	21999132
D18512		2	5770.9347 s	719576
Sw24978		16	15071.5507 s	949792

Here the comparison of Rocki's TSP implementation and our optimized TSP implementation. Table 7, shows that the our optimized TSP algorithm gives the optimized tour length than previous work for ch130 data.

Table 7: TSP tour length comparison

Dataset	No. Of Cities	Optimized Tour Length	Time	Optimized Tour Length, Rocki K.et.al	Time Required to Reach First Local Minimum
ch130	130	6321	0.0146 s	7041	0.001183 s
fnl4461	4461	203096	77.0407 s	194746	0.3271 s
usa13509	13509	22090071	2310.1165 s	20984503	6.5251 s
d18512	18512	719576	5770.9347 s	675638	14.975 s
sw24978	24978	949792	15071.5507 s	908598	37.284 s

V. Conclusion

This system provides a parallel approach for graph algorithms that are Breadth First Search, All Pair Shortest Path and Traveling Salesman Problem. In this system, optimized BFS algorithms achieve parallelism through edges wise and achieve better execution time. Here visit node with less cost. The APSP algorithm perform by using different memories of GPU and experimental result shows that for shared memory required less execution time hence can optimized APSP algorithm using shared memory. In the TSP algorithm use a shared memory and thread control function to optimized algorithm. Experimental result of TSP shows that algorithm gives a optimized tour length. Result also shows that increasing block size means iterations it gives optimized tour length. This TSP algorithm works on effectiveness of solution.

For the future work the system can also provide a better effectiveness solution of TSP with less execution time.

Acknowledgements

The author wish to thank Matoshri college of Engineering and Research Centre Nasik, HOD of computer department, guide and parents for supporting and motivating for this work because without their blessing this was not possible.

References

[1]. Jun-Dong Cho, Salil Raje, and Majid Sarrafzadeh, “Fast approximation algorithms on maxcut, k-coloring, and k-color ordering for vlsi applications,” IEEE Transactions on Computers, 47(11):1253–1266, 1998.

[2]. David A. Bader and Kamesh Madduri, “Designing multithreaded algorithms for breadth-first search and st-connectivity on MTA-2,” In ICPP, pages 523–530, 2006.

[3]. J.D. Owens, D. Luebke, N.K. Govindaraju, M. Harris, J. Kruger, “A survey of general-purpose computation on graphics hardware, ” in Proc. Eurographics, State Art Rep., 2005, pp. 21-51.

[4]. P. Harish and P.J. Narayanan, “Accelerating Large Graph Algorithms on the GPU Using CUDA,” in Proc. HiPC, 2007, pp. 197-208.

[5]. Vibhav Vineet and P. J. Narayanan,“Large graph algorithms for massively multithreaded architecture” in Proc. HiPC, 2009.

[6]. L. Luo, M. Wong, and W.-M. Hwu, An Effective GPU Implementation of Breadth-First Search, in Proc. DAC, 2010, pp. 52-55.

[7]. S. Hong, S.K. Kim, T. Oguntebi, and K. Olukotun, “Accelerating CUDA Graph Algorithms at Maximum Warp,” in Proc. PPoPP, 2011, pp. 267-276

[8]. MICIKEVICIUS P.: General parallel computation on commodity graphics hardware: Case study with the al lpairs shortest paths problem. In Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA ’04, June 21-24,2004, Las Vegas, Nevada, USA, Volume 3 (2004), CSREA Press, pp. 1359–1365.

[9]. G.J. Katz and J.T. Kider Jr., All-Pairs Shortest-Paths for Large Graphs on the GPU, in Proc. Graph. Hardware, 2008, pp. 47-55.

[10]. VENKATARAMAN G., SAHNI S., MUKHOPADHYAYA S.: A blocked all-pairs shortest-paths algorithm. J. Exp. Algorithmics 8 (2003), 2.2.

[11]. Jianlong Zhong and Bingsheng He,"Medusa: Simplified Graph Processing on GPUs".IEEE Transaction on parallel and distributed system, Vol. 25, NO. 6, JUNE 2014

[12]. CUDA Zone. Official webpage of the nvidia cuda api. Website, http://www.nvidia.com/object/cuda home.html.

[13]. N Wilt. The CUDA Handbook: A Comprehensive Guide to GPU Programming. Addison-Wesley Professional, 2013.

[14]. 10th DIMACS Implementation Challenge, Available: http://www.cc.gatech.edu/dimacs10/index.shtml

[15]. Tsai, H.; Yang, J. Kao, C. Solving traveling salesman problems by combining global and local search mechanisms, Proceedings of the 2002 Congress on Evolutionary Computation (CEC’02), 2002.

[16]. Pepper J.; Golden, B. Wasil, E. Solving the travelling salesman problem with annealing-based heuristics: a computational study. IEEE Transactions on Man and Cybernetics Systems, Part A, Vol. 32, No.1, pp. 72-77, 2002.

[17]. M. A. O’Neil, D. Tamir, and M. Burtscher.: A Parallel GPU Version of the Traveling Salesman Problem. 2011 International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 348-353. July 2011.

[18]. Luong T.V., Melab N., Talbi E.-G.; GPU Computing for Parallel Local Search Metaheuristic Algorithms. IEEE Transactions on Computers, Vol. 62, No. 1, Jan. 2013.