ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

COMPARATIVE STUDY ON SINGLE SOURCE SHORTEST PATH ALGORITHM

Ms. Rasika P. Arbat.
Department of Computer Engineering, SND College of Engineering & Research Centre,
Yeola, Nashik-423401, Maharashtra, India

DOI: 10.5281/zenodo.230948

ABSTRACT

We propose architecture for graph analysis to find out the single source shortest path to all other vertices is a common problem. The solution to this problem is Bellman- Ford's algorithm that solves such a single source shortest path (SSSP) problem and better applies to be parallelized for many core architectures. In this we get, the high degree of parallelism is guaranteed at the cost of low work efficiency which is compared to similar algorithms in literature (e.g. Dijkstra's) involves much more redundant work and a consequent waste of power consumption. This architecture is a parallel implementation of the Bellman-Ford algorithm that explains the architectural characteristics of recent GPU architectures (i.e., NVIDIA Kepler, Maxwell) to improve performance as well as work efficiency. The architecture also presents different optimizations to the implementation, which are oriented both to the algorithm and to the architecture. The experiment will show that the proposed implementation provides an average speedup of 5x higher than the existing most efficient parallel implementations for SSSP, that it works on graphs where those implementations cannot work or are inefficient (e.g., graphs with negative weight edges, sparse graphs), and it reduces the redundant work caused by the parallelization process.

KEYWORDS: Parallel computation, CUDA architecture, GPU computing, shortest path.

INTRODUCTION

Number of problems that arise in real world networks requires the computation of the shortest path and its distances from a source to one or more destination points. Its examples are car navigation systems, traffic simulations, spatial databases, Internet route planners, and web searching. Algorithms to solve the shortest path problem are computationally costly, and in many cases commercial products implement heuristic approaches to generate approximate solutions instead. Although heuristics are usually faster and do not need much amount of data storage or pre computation, they do not guarantee the optimal path. The Single-Source Shortest Path (SSSP) problem is a classical problem of optimization.

Given a graph **G = (V,E)**, a function **w(e) : e ∈ E** that associates a weight to the edges of the graph, and a source node **s**, it consists on computing the shortest paths from s to every node **v ∈ V** . If the weights of the graph range only in positive values **w(e) ≥ 0 : e ∈ E**, we are facing the so called Non negative Single-source Shortest Path (NSSP) problem. The classical algorithm that solves the NSSP problem is Dijkstra's algorithm. Being n = |V | and m = |E|, the complexity time of this algorithm is O (n^2). Dijkstra's algorithm is a greedy algorithm whose efficiency is based in the ordering of previously computed results. This feature makes parallelization a difficult task. However, there are certain situations where parts of this ordering can be permuted without leading to wrong results neither performance losses. Other algorithms for this problem, such as the Bellman-Ford algorithm, are more easily parallelizable. However, with an asymptotical complexity of O(m · n), this algorithm is not as efficient as Dijkstra's algorithm solving this problem.

An emerging way of parallel computation includes the use of hardware accelerators, such as graphic processing units (GPUs). Their powerful capability have triggered their massive use to speed up high level parallel computations.

ISSN: 2277-9655
[Arbat* *et al.*, 6(1): January, 2017] Impact Factor: 4.116
IC™ Value: 3.00 CODEN: IJESS7

RELATED WORK

1. **Parallel computation**
   Parallel computing is a type of computation in which many calculations are carried out simultaneously, or the execution of processes are carried out simultaneously. Large problems can often be divided into smaller ones, which can then be solved at the same time.
2. **Shortest Path**
   Dijkstra's algorithm is also known as single source shortest path algorithm. It calculates the shortest path from the source to each of the unvisited vertices in the graph. Dijkstra's Algorithm is used as a method of increasing node by node to get a shortest path tree which makes the starting point as its root. Whereas the **Bellman-Ford algorithm** uses relaxation to find single source shortest paths on directed graphs. and it is also contain negative edges. The algorithm also detect if there are any negative weight cycles (such that there is no solution). If talking about distances on a map, there is no any negative distance. The basic structure of bellman-ford algorithm is similar to dijkstra algorithm. It relaxes all the edges, and does this |V| - 1 time, where |V| is the number of vertices in the graph. The cost of a path is the sum of edge weights in the path.

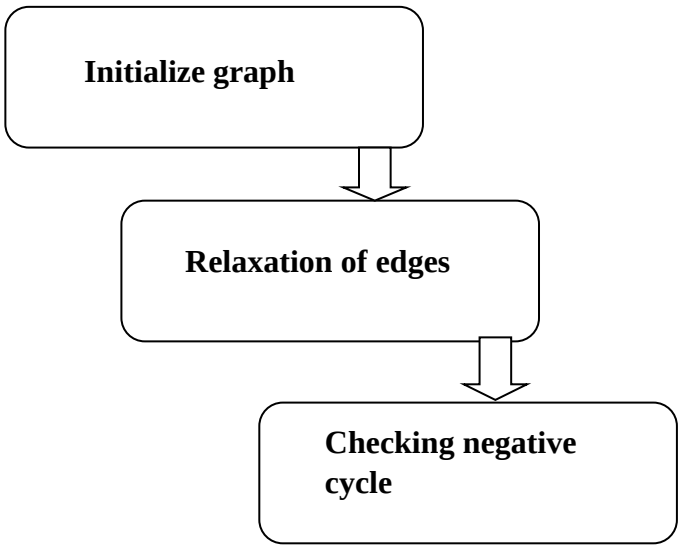

Fig. 1Process of Bellman-Ford Algorithm

3. CUDA Architecture

CUDA is the hardware and software architecture that enables NVIDIA GPUs to execute programs written with C, C++, Fortran, Open CL, Direct Compute, and other languages[1].

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

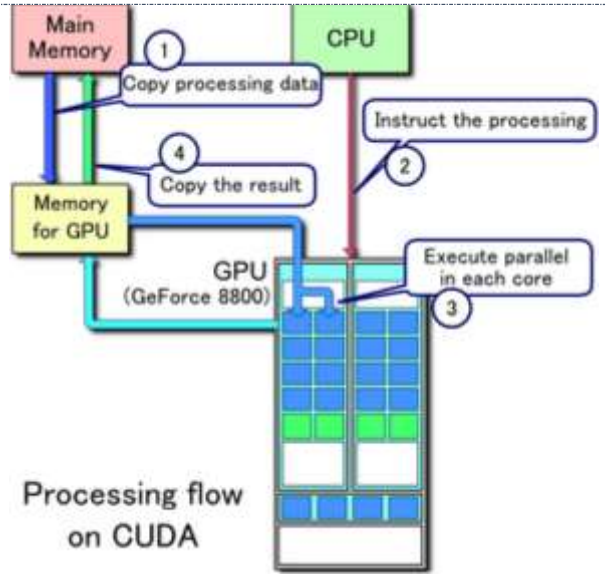

Fig. 2 CUDA Memory Architecture[15]

From fig. 2 CPU architecture must minimize latency within each thread and GPU architecture hides latency with computation from other threads.

4. Processing Flow

1. Copy input data from CPU memory to GPU memory
2. Load GPU program and execute, caching data on chip for performance
3. Copy results from GPU memory to CPU memory[1].

Serial code executes in a Host (CPU) thread and sparallel code executes in many Device (GPU) threads across multiple processing elements.

Advantages Of CUDA

CUDA has several advantages over traditional general-purpose computation on GPUs (GPGPU) using graphics APIs:

1. Scattered reads – code can read from arbitrary addresses in memory
2. Unified virtual memory (CUDA 4.0 and above)
3. Unified memory (CUDA 6.0 and above)
4. Shared memory – CUDA exposes a fast shared memory region that can be shared among threads. This can be used as a user-managed cache, enabling higher bandwidth than is possible using texture lookups.
5. Faster downloads and read backs to and from the GPU.
6. Full support for integer and bitwise operations, including integer texture lookups.

GPU Computing

GPU architectures are increasingly important in the multi-core era due to their high number of parallel processors. Performance optimization for multicore processors has been a challenge for programmers. Furthermore, optimizing for power consumption is even more difficult. Unfortunately, as a result of the high number of processors, the power consumption of many-core processors such as GPUs has increased significantly

CONCLUSION

In this paper we propose to develop a fast, memory efficient parallel Single source shortest path algorithm using NVIDIA CUDA which takes graph as input and launches n no of parallel threads for SSSP algorithm, where n is the total edges in graph. To increase the performance of applications like data mining, image processing, network analysis, vehicle routing, computer wiring etc. which uses graph processing as fundamental part of processing also reduce latency for improving performance.

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

ACKNOWLEDGEMENT

The authors are grateful to Jianlong Zhong[18] and Bingsheng He for Medusa: Simplified Graph Processing on GPUs in IEEE TRANSACTIONS on Parallel and Distributed Systems, VOL. 25, NO. 6, JUNE 2014. The authors would like to thank the anonymous reviewers for their valuable comments, and Pawan Harish for providing the source code for CUDA-based BFS and shortest paths. This work is partly supported by a MoE AcRF Tier 2 Grant (MOE2012-T2-2-067) and an NVIDIA Academic Partnership Award.

REFERENCES

1. P. Harish and P. J. Narayanan, "Accelerating large graph algorithms on the GPU using cuda", in Proc.14th international conference on HiPC'07, Berlin, Heidelberg: Springer Verlag ,pp.197-208, 2007.
2. David A. Bader and Kamesh Madduri, "Designing Multi threaded Algorithms for Breadth First Search and connectivity on the Cray MTA2",ICPP,pp. 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,pp. 21-51, 2005.
4. V. Vineet and P. Narayanan," CUDA cuts- Fast graph cuts on the GPU ",in Computer Vision and Pattern Recognition, IEEE Computer Society Conference ,pp.1-8 2008.
5. L. Luo, M. Wong and W. M. Hwu,"An Effective GPU Implementation of Breadth-First Search",in Proc. DAC,pp. 52-55, 2010.
6. J. Soman, K. Kishore, and P J Narayanan, "A Fast GPU Algorithm for Graph Connectivity", IEEE Transaction on parallel and distributed system,Vol.2,June 2012.
7. G.J. Katz and J.T. Kider," All-Pairs Shortest-Paths for Large Graphs on the GPU", in Proc. Graph Hardware,pp.47-55,2008.
8. D. Merrill, M. Garland and A. Grimshaw, "Scalable GPU graph traversal", in Proc. of the 17th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming, PPoPP '12,pp. 117-128, 2012.
9. Merrill, D. et al. 2011,High Performance and Scalable GPU Graph Traversal, Technical Report CS2011-05,Department of Computer Science, University of Virginia.
10. Bader, D.A. et al, On the Architectural Requirements for Efficient Execution of Graph Algorithms, International Conference on Parallel Processing, ICPPâA˘ Z05,Oslo,Norway: pp. 547-556,2005
11. E. Dekel, D. Nassimi, and S. Sahni, "Parallel matrix and graph algorithms", SIAM Journal on computing, vol. 10,No. 4, pp. 657-675, 1999.
12. NVIDIA CUDA official site for developer website https://developer.nvidia.com/cuda-zone.
13. GT graph: A suite of synthetic random graph generators: https://sdm.lbl.gov/kamesh/ software /GTgraph/Accessed: 2011,07-11.
14. Cormen, Leiserson, Rivest, And Stein, "Introduction to Algorithms", Book 2012.
15. NVIDIA, Cuda: Compute unified device architecture programming guide. Technical report, NVIDIA, 2014.
16. J. Zhong and B. He,Parallel graph processing on graphics processors made easy, Proc. VLDB Endow.vol. 6, pp. 1270-1273, Aug. 2013.
17. C. P. Mogal and C. R. Barde, "Review Paper on Optimized and accelerated Parallel Graph Algorithm on GPGPU ",IJCSMC, Vol. 4,Issue.12,pg.103 âA ¸S 106, December 2015.
18. J. Zhong and B. He, Kernelet: High-throughput GPU kernel executions with dynamic slicing and scheduling, IEEE Transactions on Parallel and Distributed Systems, vol. 99, no. PrePrints, p. 1, 2013.