Solving Path Problems on Multiple GPUs

Terence L. Henriod*, David J. Feil-Seifer*, Frederick C. Harris, Jr.*, Lee A. Barford*
*University of Nevada, Reno

***Abstract*—We explore improving the performance of the computation of shortest paths on Graphics Processing Units (GPUs). Shortest path computation can be cast into a class of algorithms that use a block-recursive elimination strategy (along with transitive closure and LU decomposition without pivoting). Computations of this sort have been improved upon in both CPU implementations, and on the GPU. At least one GPU implmentation (by Buluc et al.) has been shown to utilize the computational resources of the GPU very effectively. Our implementation improves on this implementation by utilizing shared memory between threads and by adjusting the amount of work performed per thread with each kernel launch. Further, we discuss possible methods and pitfalls for porting the algorithm to a multiple GPU implementation for further parallelism and presumably greater speedup.**

I. Introduction

The All-Pairs Shortest Paths (APSP) problem is one of the most important, and most studied, algorithmic graph problems [1]. The problem is formulated as follows: given a directed graph $G(V, E)$, where V is the set of vertices belonging to the graph and E is the set of edges connecting those vertices, determine the optimal cost c of each path from each vertex v_i to every other vertex v_j in the graph along the edges in E. In the case of the unweighted graph, the solution is simply the number of edges that need to be traversed to determine transitive closure (reachability) between each pair of vertices. The more interesting case is when the edges are weighted, and more developed methods must be used.

Shortest paths computations have a wide variety of applications, ranging from the obvious optimal path planning in physical transportation, to network and circuit design, to speech processing and other probabilistic applications, amond many others [2]. It is also known that shortest paths computations can be useful in finding important features in graphs, such as ideal dispersion centers for communication efforts.

The shortest paths problem is one with a history of "multiple invention," meaning that once the problem of finding the shortest paths in a graph became the problem of focus for some reasearchers, many quickly found similar, if not identical solutions, almost concurrently and independently. The history of the problem is steeped in the use of matrix methods, reaching back to around 1946 when matrices were used to represent networks and researchers sought to find transitive closure for networks, that is, determining whether one point in a graph is reachable from another [3].

II. Previous Work by Others

III. Sequential Computation of APSP

When solving APSP sequentially (as in not in parallel), there are two main algorithms of interest: the Floyd-Warshall algorithm and Johnson's algorithm (perhaps more widely recognized as applying Dijkstra's algorithm to every vertex in the graph.) The Flod-Warshall algorithm is thought to be better suited for dense graphs[4], while using Dijkstra's method is thought to be better for applications with sparse graphs [5].

IV. Parallel Approaches to APSP

To our knowledge, all successful approaches to parallel compuation of APSP make use of a block-recursive Gaussian elimination strategy. This is of course, ideal for parallel computation, as being able to distribute sub-blocks of a matrix to each processor is much more feasible than methods that might require some sort of shared memory amoung all processors.

There are many researchers who have done work to improve the computation of APSP in parallel recently. Unfortunately, (to our knowledge) much of this work is algebraic in nature and few practical implementations are presented.

Tiskin was able to demonstrate an efficient *bulk-synchronous parallel (BSP) computation* algorithm for APSP, however they gave no implementation. The algorithm was demonstrated algebraically to reduce the number of golbal sychronizations required from at most $O(p^{\frac{2}{3}})$ global synchronizations, where p is the number of processors or parallel units used for the computation. The use of Djikstra's method was shown to have a higher communication cost than the Floyd-Warshall algorithm, but to have a lower synchroniztion cost [2]. This is advantageous, as synchroniztion if often the main detriment to parallelized algorithms, both in terms of implementation complexity and time cost for performance. Unfortunately, Tiskin could not guarantee practical significance of their findings because (1) no real implementation was produced, and (2) there is "significant potential overhead of dealing with path matrices, instead of ordinary numerical matrices (as e.g. in the Floyd-Warshall algorithm)" [2].

The only recent implementation of parallel APSP (to our konwledge) is that of Buluc, Gilbert, and Budak. They present a parallel APSP implementation for the GPU that showed large improvements over previous GPU implementations. Their strategy used a Gaussian Elimination, block recursive strategy that was tailored to the massively parallel nature of the GPU.

V. Results

A. Sequential Results

B. Parallel Results

C. Comparisons

1) Parallel vs. Sequential:

2) New Parallel vs. Old Parallel:

VI. Conclusion

VII. Introduction

This demo file is intended to serve as a "starter file" for IEEE conference papers produced under LaTeX using IEEEtran.cls version 1.8a and later. I wish you the best of success[4]. Other stuff.

mds
September 17, 2014

A. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

References

[1] Y. Peres, D. Sotnikov, B. Sudakov, and U. Zwick, "All-pairs shortest paths in o(n2) time with high probability," *J. ACM*, vol. 60, no. 4, pp. 26:1–26:25, Sep. 2013. [Online]. Available: http://doi.acm.org/10.1145/2505988

[2]

[3]

[4] A. Buluç, J. R. Gilbert, and C. Budak, "Solving path problems on the gpu," *Parallel Comput.*, vol. 36, no. 5-6, pp. 241–253, Jun. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.parco.2009.12.002

[5] S. Pettie, "A new approach to all-pairs shortest paths on real-weighted graphs," *Theoretical Computer Science*, vol. 312, no. 1, pp. 47 – 74, 2004, automata, Languages and Programming. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S030439750300402X

[6] NVIDIA Corporation. (2015) NVIDIA CUDA. [Online]. Available: http://www.nvidia.com/object/cuda_home_new.html

[7] ——. (2015) NVIDIA CUDA Programming Guide. [Online]. Available: http://docs.nvidia.com/cuda/cuda-c-programming-guide/#axzz3ZrApAMsb