Comparative Analysis of Floyd Warshall and Dijkstras Algorithm using Opencl

Asad Mohammad
Gyan Ganga College of Technology

Vikram Garg
Gyan Ganga College of Technology

ABSTRACT

Shortest path algorithms finds applications in large real world domains. All pair shortest path (APSP) and single source shortest path (SSSP) both have their special applications domains. All though every SSSP can be applied for all vertices to calculate APSP. But APSP cant. In this paper heterogeneous implementation of Floyd warshalls algorithm and Dijkstra's algorithm is compared on dense graphs have positive edge weights ranging from 1 to 10. It is found that Dijkstra's algorithm is better than Floyd warshall algorithm in sequencial implementation. But as there is less parallelism identified in dijkstra algorithm as compared to parallel to parallel FW gives less execution time as compared to Dijkstra's.

Keywords

Floyd Warshall (FW), Dijkstra algorithm, SSSP, APSP, OpenCL.

1. INTRODUCTION

The shortest path problem refers to the problem of finding the shortest path or minimal cost route from a specific source to a particular destination. Generally, graphs are most widely used for representation of such problems. Shortest Path problem basically deals with graphs and in specific, with problems belonging to weighted directed graph category. It finds applications in large number of domains such as, in network routing protocols, VLSI design, robotics and intelligent transportation systems. A graph is a finite set of vertices and edges represented as, G (V, E). Vertices are connected using edges.

A directed graph is a graph in which each edge could be traversed only in a given direction. To calculate the distance between any two vertices (also called nodes), edges are given some values called weights (or cost). These weights measure the cost or distance between any two nodes via different possible edge sequence. The shortest path analysis is not restricted to the shortest distance between the source and destination, but also refers to other measurement units such as time, cost and the capacity of the link.

There are number of optimization and high performance computing techniques like vectorization, loop unrolling using parallel heterogeneous computing platform. In this paper a heterogeneous computing platform which provided parallelism on multiple devices is explored for implementing shortest path algorithms.

OpenCL is an open source computing platform which performs following steps :

- First it identifies platforms available on the device we have run the implementation.
  - In this paper intel and NVIDIA platform is used.
- Second it identifies devices available on available platforms.
  - In intel platform we have 2 devices available Intel i5 CPU and Intel graphics card.
  - In NVIDIA platform we have NVIDIA Geforce graphics card.
- Then for the devices we want to exploit a context is created. And devices in same context can only be synchronized. So if we want to synchronize 2 devices both need to be in same context.
- Then in fourth step command queues are created for every device independently. As we want to give parallel commands to both the devices. 2 command queues are created.
- Parallel kernels are created using OpenCL for both the algorithms. And each kernel is processed on separate devices in same context.

2. RELATED WORK

In [4] three parallelfriendly and work-efficient methods to solve this Single-Source Shortest Paths (SSSP) problem: *Workfront Sweep*, *Near-Far* and *Bucketing*. All of these methods do much less work than traditional Bellman Ford method. All these techniques for shortest path algorithms are implemented on GPU. In this paper Dijkstra algorithm is compared with BellmanFord algorithm on which above mentioned three techniques are applied. All these algorithms are studied for different data structures and traversal techniques. In [10] Floyd Warshall and Dijkstra algorithm are compared by applying divide and conquer on FW to make use of multi GPU cluster using OpenCL.

In this paper Floyd Warshall algorithms is compared with Dijkstra algorithm for dense graphs and Dijkstra algorithm is also compared with FW with its APSP implementation.

3. FLOYD WARSHALL ALGORITHM

APSP is a fundamental problem in graph theory. Floyd-Warshall (FW) is a well known algorithm for its solution. FW sequential implementation uses three nested loops.

Consider a weighted graph G (V, E) stored using adjacency matrix representation by a weight matrix W of order N*N where N is number of vertices in G.where, $w_{ij} \epsilon$ W for all (i,j) ϵ E.

This matrix W contains zero for diagonal elements as both corresponds to same vetex. And infinity for the vertices which are not connected directly and weight is there for vertices which are connected directly or edges available in graph.

Floyd algorithm :

```
For(int k=0;k<N;k++)
  For (int i=0;i<N;i++)
    For(int j=0;j<N;j++)
      $d_{ij}^{(k)} \leftarrow \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
```

Identified parallelism :
As it is clear from the above algorithm that value of kth iteration depends on k-1 so this loop contains dependency so it cannot be removed to perform parallelism. But rest 2 loops can be called in parallel for N2 threads using OpenCL.

Please use a 9-point Times Roman font, or other Roman font with serifs, as close as possible in appearance to Times Roman in which these guidelines have been set. The goal is to have a 9-point text, as you see here. Please use sans-serif or non-proportional fonts only for special purposes, such as distinguishing source code text. If Times Roman is not available, try the font named Computer Modern Roman. On a Macintosh, use the font named Times. Right margins should be justified, not ragged.

4. DIJKSTRA ALGORITHM

Dijkstra algorithm ia a single source shortest path algorithm and it can only be applied to connected graphs having positive edge weights.

The algorithm consists of following steps :

- Distance to source vertex is set to zero.
- Set all other distances to infinity.
- S is set of visited vertices which is empty initially.
- Q is the queue which initially contains all the vertices.
- Then until Q is empty an element is selected from Q with minimum distance.
- And then this u is added to visited vertex list.
- If new shortest path found is shortest among all it is set as new shortest path till this step.

```
Dist[S] $\leftarrow$ 0
For all v $\epsilon$ V – {S}
    Do dist[v] $\leftarrow \infty$
S $\leftarrow \varphi$
Q $\leftarrow$V
While Q $\neq \varphi$
Do u $\leftarrow$min(Q,dist)
  S $\leftarrow$S U {u}
  For all v $\epsilon$ neighbours[u]
    Do if dist[v] > dist[u] + W(u,v)
       Then d[v] $\leftarrow$d[u] + W(u,v)


Return dist
```

Identified parallelism :
For all the vertices in Q we can execute below steps in parallel and find the vertex using write-write consistency which holds minimum distance value. So that the distance returned will be the least of all the vertices processed in parallel.

5. RESULTS AND COMPARATIVE ANALYSIS

In this paper sequential implementations are developed in c and parallel implementations are done using OpenCL whose host is written in c language. All the implementations are analyzed and executed using integrated development environment visual studio. All the implementations are executed on intel i5 cpu @ 2.20 GHz and GPU of NVIDIA Geforce 820M. Execution time is measured in milliseconds. All the implementations are executed for different dense graphs with different vertices with edge weights ranging from 1 to 10.

VERTICES	SERIAL FW	SERIAL DIJKSTRA (SSSP)	SERIAL DIJKSTRA (APSP)
64	32	9	21
128	227	72	178
256	961	276	746
512	8401	2214	11002
1024	14551	6920	16326
2048	71324	11287	69337
4096	378991	65879	400221

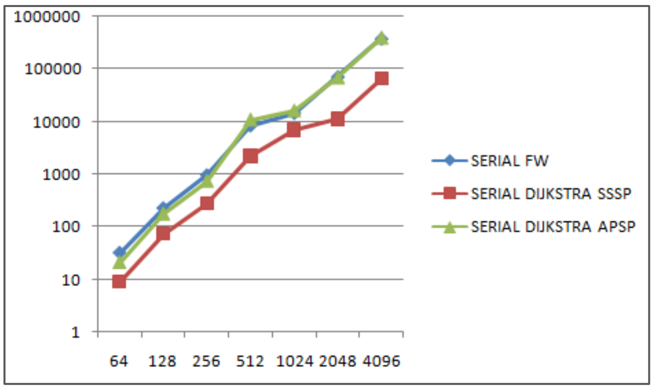

Figure 1: graph showing comparision of execution time of serial FW with serial Dijkstra for SSSP and APSP.

It is clear from the above graph serial dijkstra takes less time as it is order of n*n and serial FW takes comparable time as Dijkstra when implemented for APSP. So parallel implementations are expected to take comparable time for Dijkstra APSP and FW. Which are illustrated below.

VERTICES	PARALLEL FW	PARALLEL DIJKSTRA (SSSP)	PARALLEL DIJKSTRA (APSP)
64	1	1	3
128	15	7	18
256	21	12	24
512	89	112	171
1024	904	1153	1763
2048	5232	5987	7289
4096	37127	27645	36915

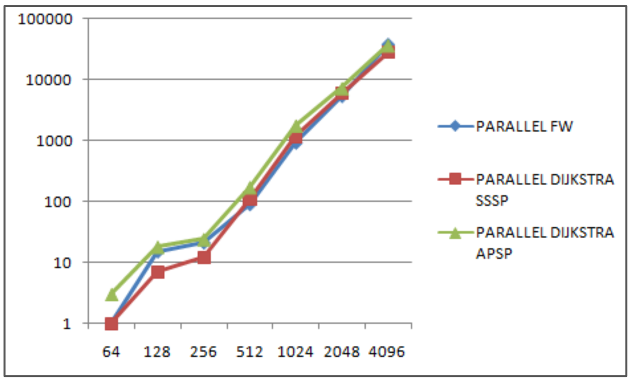

Figure 2: graph showing comparision of execution time of parallel FW with parallel Dijkstra for SSSP and APSP.

6. CONCLUSION

In this paper parallel Floyd Warshall and parallel Dijkstra for SSSP and APSP are compared with each other and sequential implementation of the same. It is found that parallel Dijkstra and parallel FW are comparable with each other. In future optimization techniques like vectorization, loop unrolling can also be applied to these implementations.

7. REFERENCES

[1] R. Bellman. On a routing problem. Quarterly of Applied Mathematics 16:87-90,1958.

[2] Yefim Dinitz , Rotem Itzhak , *Hybrid Bellman-Ford-Dijkstra Algorithm.*

[3] Aydın Buluc , John R. Gilbert and Ceren Budak , "*Solving Path Problems on the GPU" , Journal Parallel Computing Volume 36 Issue 5-6, June,2010 Pages 241-253.*

[4] Andrew Davidson , Sean Baxter, Michael Garland , John D. Owens , "*Work-Efficient Parallel GPU Methods for Single-Source Shortest Path " in International Parallel and Distributed Processing Symposium, 2014*

[5] Owens J.D., Davis, Houston, M., Luebke, D., Green, S., "*GPU Computing*", in: Proceedings of the IEEE, Volume: 96 , Issue: 5 , 2008.

[6] A. Munshi, B. R. Gaster, T.G. Mattson, J. Fung, D. Ginsburg, "*OpenCL Programming Guide*", Addison-Wesley pub., 2011.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, Second Edition. The MIT Press, Sep. 2001.

[8] Kumar, S.; Misra, A.; Tomar, R.S. ,"A modified parallel approach to Single Source Shortest Path Problem for massively dense graphs using CUDA" in *Computer and Communication Technology (ICCCT), 2011 2nd International Conference on* , vol., no., pp.635,639, 15-17 Sept. 2011.

[9] Atul Khanna, John Zinky , "The Revised ARPANET Routing Metric", in 1969 ACM.

[10] Hristo Djidjev and Sunil Thulasidasan,Guillaume Chapuis, Rumen Andonov, and Dominique Lavenier "Efficient Multi-GPU Computation of All-Pairs Shortest Paths" in 2014 IEEE 28th International Parallel & Distributed Processing Symposium.