Comparative Analysis of Floyd Warshall and Dijkstra’s 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 sequential 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, \( w_{ij} \in W \) for all \((i,j) \in E\). This matrix W contains zero for diagonal elements as both corresponds to same vertex. 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++)

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.

4. DIJKSTRA ALGORITHM
Dijkstra algorithm is 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 \in V - \{S\}
Do dist[v] \leftarrow \infty
S \leftarrow v
Q \leftarrow \{v\}
While Q \neq \emptyset
Do u \leftarrow \min(Q, \text{dist})
S \leftarrow S \cup \{u\}
For all v \in \text{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.

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

Figure 1: graph showing comparison 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^2 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.

<table>
<thead>
<tr>
<th>VERTICES</th>
<th>PARALLEL FW</th>
<th>PARALLEL DIJKSTRA (SSSP)</th>
<th>PARALLEL DIJKSTRA (APSP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>128</td>
<td>15</td>
<td>7</td>
<td>18</td>
</tr>
<tr>
<td>256</td>
<td>21</td>
<td>12</td>
<td>24</td>
</tr>
<tr>
<td>512</td>
<td>89</td>
<td>112</td>
<td>171</td>
</tr>
<tr>
<td>1024</td>
<td>904</td>
<td>1153</td>
<td>1763</td>
</tr>
<tr>
<td>2048</td>
<td>5232</td>
<td>5987</td>
<td>7289</td>
</tr>
<tr>
<td>4096</td>
<td>37127</td>
<td>27645</td>
<td>36915</td>
</tr>
</tbody>
</table>
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