MST USING PRIM’S ALGORITHM IN CUDA

Presented by: Suhas Reddy
Outline

• CPU vs GPU Evolution
• Problem Statement
• Applications
• Sequential Implementation
• Parallel Implementation
• Results
CPU vs GPU Evolution

Theoretical peak (GFLOPs)

Release date


NVIDIA GPU SP
NVIDIA GPU DP
Intel SP
Intel DP

GeForce TX TITAN
GeForce GTX 680
GeForce GTX 580
GeForce GTX 480
GeForce GTX 280
GeForce 8800 GTX
GeForce 7800 GTX
GeForce 6800 Ultra
GeForce FX 5800
Willamette
Prescott
Woodcrest
Harpertown
Bloomfield
Sandy Bridge
Ivy Bridge
Haswell
K20X
GPGPU Architecture Implementation
Problem Statement

- Prim’s algorithm is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph.
- Developed in 1930 by Czech mathematician Vojtech Jarnik and later rediscovered and republished by computer scientists Robert C. Prim in 1957 and E.W Dijkstra in 1959.
Problem Statement

**Minimum Spanning Tree** in an undirected connected weighted graph is a spanning tree of minimum weight. Example:

Weighted Graph

Spanning Tree 1, $w=74$

Spanning Tree 2, (Minimum Spanning Tree) $w=71$

Spanning Tree 3, $w=72$
Applications

• Routing algorithms
• Clustering
• Travelling Salesman Problem
• Finding airline routes
Prim’s Algorithm

**MST-Prim**(\(G, w\))

1. \(s \leftarrow \text{arbitrary vertex in } G\)
2. \(S \leftarrow \emptyset, d(s) \leftarrow 0\) and \(d[v] \leftarrow \infty\) for every \(v \in V \setminus \{s\}\)
3. while \(S \neq V\) do
4. \(u \leftarrow \text{vertex in } V \setminus S\) with the minimum \(d[u]\)
5. \(S \leftarrow S \cup \{u\}\)
6. for each \(v \in V \setminus S\) such that \((u, v) \in E\) do
   - if \(w(u, v) < d[v]\) then
     - \(d[v] \leftarrow w(u, v)\)
     - \(\pi[v] \leftarrow u\)
7. return \(\{(u, \pi[u]) | u \in V \setminus \{s\}\}\)
Prim’s Algorithm
Prim’s Implementation in CUDA Approach

- One straight forward approach is to introduce CUDA directly to the above algorithm.
- We can bring parallelism here exploring the unvisited vertices in parallel leveraging the multi-core processors.
- Parallelly explore all the unvisited edges of the vertices already visited and add the unexplored vertex of the corresponding min edge.
## Results

<table>
<thead>
<tr>
<th>Graph Size (1000 Nodes)</th>
<th>CPU Runtime (s)</th>
<th>GPU Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.45</td>
<td>0.15</td>
</tr>
<tr>
<td>4</td>
<td>8.55</td>
<td>0.35</td>
</tr>
<tr>
<td>8</td>
<td>21.14</td>
<td>0.77</td>
</tr>
<tr>
<td>16</td>
<td>98.01</td>
<td>1.88</td>
</tr>
<tr>
<td>20</td>
<td>133.02</td>
<td>3.07</td>
</tr>
<tr>
<td>32</td>
<td>155</td>
<td>4.04</td>
</tr>
<tr>
<td>64</td>
<td>192.88</td>
<td>5.41</td>
</tr>
</tbody>
</table>
GPU Runtime
Runtime Plot Comparison CPU vs GPU
References

• https://developer.nvidia.com/cuda-toolkit
• Prim's Algorithm
• Evolution_GPU_vs_CPU
THANK YOU