MapGraph

A High Level API for Fast Development of High Performance Graphic Analytics on GPUs

http://mapgraph.io

Zhisong Fu, Michael Personick and Bryan Thompson
SYSTAP, LLC
Outline

• Motivations
• MapGraph overview
• Results
• Summary
GPUs – A Game Changer for Graph Analytics?

• Graphs are everywhere in data, also getting bigger and bigger

• GPUs may be the technology that finally delivers real-time analytics on large graphs
  • 10x flops over CPU
  • 10x memory bandwidth

• This is a hard problem
  • Irregular memory access
  • Load imbalance

• Significant speed up over CPU on BFS [Merrill2013]
  • Over 10x speedup over CPU
## Low-level VS. High-level

<table>
<thead>
<tr>
<th>Low-level approach</th>
<th>High-level approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>• BFS: [Merrill2013]</td>
<td>• GraphLab [Low2012]</td>
</tr>
<tr>
<td>• PageRank: [Duong2012]</td>
<td>• Medusa [Zhong2013]</td>
</tr>
<tr>
<td>• SSSP: [Davidson2014]</td>
<td>• Totem [Gharaiheh2013]</td>
</tr>
</tbody>
</table>

- **Pros:** High performance
- **Cons:** Difficulty to develop
  Reinvent the wheels

- **Pros:** High programmability
- **Cons:** Low Performance
MapGraph

• High-level graph processing framework
  • High programmability: only C++ sequential
    GPU architecture
    Optimization techniques
    CUDA, OpenCL
  • High performance
    Comparable to low-level approach
GAS Abstraction

Gather
GAS Abstraction

Gather

Apply
GAS Abstraction

Gather

Scatter = Expand + Contract

Apply
GAS Abstraction

Gather

Scatter = Expand + Contract

Apply

Frontier size > 0
MapGraph Runtime Pipeline

1. GATHER
2. APPLY
3. Frontier size > T
   - No: Scatter dynamic: expand+contract
   - Yes: Frontier size > 0
      - Yes: Scatter two-phase: expand+contract
## Experiment Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#vertices</th>
<th>#edges</th>
<th>Max Degree</th>
<th>MTEPS (BFS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Webbase</td>
<td>1,000,005</td>
<td>3,105,536</td>
<td>23</td>
<td>514</td>
</tr>
<tr>
<td>Delaunay</td>
<td>2,097,152</td>
<td>6,291,408</td>
<td>4,700</td>
<td>154</td>
</tr>
<tr>
<td>Bitcoin</td>
<td>6,297,539</td>
<td>28,143,065</td>
<td>4,075,472</td>
<td>75</td>
</tr>
<tr>
<td>Wiki</td>
<td>3,566,907</td>
<td>45,030,389</td>
<td>7,061</td>
<td>821</td>
</tr>
<tr>
<td>Kron</td>
<td>1,048,576</td>
<td>89,239,674</td>
<td>131,505</td>
<td>1,871</td>
</tr>
</tbody>
</table>

![Bar chart showing MTEPS values for different datasets](chart.png)
Results: Compare to Other GPU implementations

MapGraph Speedups vs Other GPU Implementations

- Webbase
- Delaunay
- Bitcoin
- Wiki
- Kron

Speedup

- Medusa
- B40c
BFS Results: Compare to GraphLab

MapGraph Speedup vs GraphLab (BFS)

- Webbase
- Delaunay
- Bitcoin
- Wiki
- Kron

GL-2
GL-4
GL-8
GL-12
MPG
PageRank Results: Compare to GraphLab

MapGraph Speedup vs GraphLab (PR)

- Webbase
- Delaunay
- Bitcoin
- Wiki
- Kron

- GL-2
- GL-4
- GL-8
- GL-12
- MPG
MapGraph API

Gather
- gatherOverEdges
- gather_edge
- gather_sum
- gather_vertex

Expand
- expandOverEdges
- expand_vertex
- expand_edge

Contract
- contract

Apply
- apply

Scatter = Expand + Contract
Example: PageRank Implementation

• Gather, Apply, Scatter phases

<table>
<thead>
<tr>
<th>User Data</th>
<th>VertexType</th>
<th>float* d_ranks; int* d_num_out_edge;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gather</td>
<td>gatherOverEdges</td>
<td>return GATHER_IN_EDGES;</td>
</tr>
<tr>
<td></td>
<td>gather_edge</td>
<td>float nb_rank = d_dist[neighbor_id]; new_rank = nb_rank / d_num_out_edge[neighbor_id]; gather_sum return left + right;</td>
</tr>
<tr>
<td>Apply</td>
<td>apply</td>
<td>float old_value = d_ranks[vertex_id]; float new_value = 0.15f + (1.0f - 0.15f) * gathervalue; changed = fabs(old_value - new_value) &gt;= 0.01f; d_dist[vertex_id] = new_value;</td>
</tr>
<tr>
<td>Expand</td>
<td>expandOverEdges</td>
<td>return EXPAND_OUT_EDGES;</td>
</tr>
<tr>
<td></td>
<td>expand_vertex</td>
<td>return changed;</td>
</tr>
<tr>
<td></td>
<td>expand_edge</td>
<td>frontier = neighbor_id;</td>
</tr>
</tbody>
</table>
Future Work

- GPU cluster: 2D partitioning (aka vertex cuts)
  - In collaboration with SCI Institute of the University of Utah
  - Compute grid defined over virtual nodes.
  - Patches assigned to virtual nodes based on source and target identifier of the edge.

- Topology, message and data compression
Summary

• MapGraph: high-level graph processing framework
  • http://mapgraph.io

• High programmability:
  • GAS abstraction
  • Simple and flexible API

• High performance:
  • Hybrid scheduling strategy
  • Structure Of Arrays
Acknowledgement

• This work was (partially) funded by the DARPA XDATA program under AFRL Contract #FA8750-13-C-0002.

• This work is also supported by the DARPA under Contract No. D14PC00029.

• Many thanks to Dr. Christopher White for the support.