PROBLEM
High-performance large-scale graph analytics
PROBLEM
High-performance large-scale graph analytics on GPUs
## Why Using GPUs for Graph Analytics?

<table>
<thead>
<tr>
<th></th>
<th>Memory Bandwidth</th>
<th>Memory Size</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU (NVIDIA K40c)</td>
<td>288GB/s</td>
<td>12GB</td>
<td>PCI-e 3.0*16 bus (16G/s bi-direction bandwidth)</td>
</tr>
<tr>
<td>CPU (Intel Xeon E5)</td>
<td>88.6GB/s</td>
<td>Up to TB</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Why Using GPUs for Graph Analytics?

- Graphs are ubiquitous
- Data size is becoming very large
- Graph analytics systems demand more performance
Large-scale Graph Analytics Is Difficult

- Irregularity of data access and control flow limits performance and scalability
- GPU programming is complex
Related Work

- Single-node CPU-based systems
- Distributed CPU-based systems
- Specialized GPU algorithms
- GPU-based systems
IDEA: Performance AND expressiveness

- Performance: Integrating high performance GPU computing primitives and optimizations into the core.

- Expressiveness: A data-centric abstraction designed specifically for the GPU
IDEA: Bulk-Synchronous Programming and Data-Centric

- Graph algorithms as iterative convergent processes
  - Operations run a series of steps
  - Large amount of parallelism within each step

- Manipulate frontiers
  - Generating/reorganizing frontier in parallel
  - Computing on frontier in parallel
Expressiveness: Gunrock's Data-Centric Abstraction
Gunrock's Key Abstraction Is **FRONTIER**

Most graph algorithms have two major operations:

- **Traverse**: moving in the graph and generating new frontier
- **Compute**: doing computation on frontier
Gunrock's Traversal Step

- **Advance**: visiting the neighbors of the current frontier
- **Filter**: choosing a subset of the current frontier

![Graph Diagram]

1. 2
2. 3
3. 4
4. 5
5. 6

Edges: e1, e2, e3, e4, e5, e6, e7, e8, e9
Gunrock's Traversal Step

- **Advance**: visiting the neighbors of the current frontier
- **Filter**: choosing a subset of the current frontier
Gunrock's Traversal Step

- Advance: visiting the neighbors of the current frontier
- Filter: choosing a subset of the current frontier
Gunrock's Traversal Step

- Advance: visiting the neighbors of the current frontier
- Filter: choosing a subset of the current frontier
Gunrock's Traversal Step

- **Advance**: visiting the neighbors of the current frontier
- **Filter**: choosing a subset of the current frontier
Gunrock's Compute Step

Functors that apply to \{edges, vertices\}

- "cond" functor: returns a boolean value
- "apply" functor: performs a computation
Graph Primitives in Gunrock (In only Three Files)

- **Problem**: Initialize the graph data and frontier
- **Enactor**: GPU kernel entry function which defines a series of operations on frontier
- **Functor**: User-specified per-node/per-edge computation on frontier
Performance: Generalized Optimization Strategies
Graph Data Representation: CSR
Workload Mapping and Load-balancing

- Naive method: Let one thread handle the neighbor list of one vertex
- Problem: Highly uneven distribution of node degrees in scale-free graphs

Need load balancing strategy!
Workload Mapping and Load-balancing

- Tradeoff between extra processing and load balancing
- A worthwhile extra effort: 2x–20x speedup over non-load balancing library (Medusa v1)
Data-Centric Abstraction Enables Optimizations

Idempotent operations (frontier reorganization)
Data-Centric Abstraction Enables Optimizations

Pull vs. push operations (frontier generation)

Input frontier: $v_1, v_2, v_3$

Output frontier: $v_4, v_5, v_6, v_7$

Explored edges (gray ones are failures):
- From $v_1$: $v_4, v_5$
- From $v_2$: $v_3$
- From $v_3$: $v_5$

Explored edges (blue ones are valid ones):
- From $v_4$: $v_5$
- From $v_5$: $v_6$
- From $v_7$: $v_6$

Final output frontier:
- $v_4, v_5, v_6, v_7$
Data-Centric Abstraction Enables Optimizations

Priority Queue (frontier reorganization)
Results, Conclusion, and Future Work
Performance Against Other Graph Processing Systems

![Graph Performance Chart]
Scalability on Multi-GPUs

Strong Scaling on RMAT_n22_48
Scalability on Multi-GPUs

PageRank has better strong scaling due to stable cross-GPU communication overhead.
Scalability on Multi-GPUs

Weak Scaling on RMAT_48 with 180M edges on one GPU
Scalability on Multi-GPUs

SSSP and CC have worse weak scaling, need more investigation.

Weak Scaling on RMAT_48 with 180M edges on one GPU
Table 2: Comparison with previous work on GPU BFS. Merrill et al.’s results on 3-year-old hardware are particularly impressive, though we note their implementation, as is Fu et al.’s, was customized only to BFS. Medusa (Zhong et al.), like Gunrock, is a programmable framework.

<table>
<thead>
<tr>
<th>ref.</th>
<th>ref. hardware</th>
<th>ref. performance</th>
<th>our hardware</th>
<th>our performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>rmat_n20_128</td>
<td>Merrill et al. [23]</td>
<td>4x Tesla C2050</td>
<td>8.3 GTEPS</td>
<td>4x Tesla K40</td>
</tr>
<tr>
<td>rmat_n20_16</td>
<td>Zhong et al. [32]</td>
<td>4x Tesla C2050</td>
<td>15.4 ms</td>
<td>4x Tesla K40</td>
</tr>
<tr>
<td>peak performance</td>
<td>Fu et al. [10]</td>
<td>16x Tesla K20 (cluster)</td>
<td>15 GTEPS</td>
<td>6x Tesla K40</td>
</tr>
<tr>
<td>peak performance</td>
<td>Fu et al. [10]</td>
<td>64x Tesla K20 (cluster)</td>
<td>29.1 GTEPS</td>
<td>6x Tesla K40</td>
</tr>
</tbody>
</table>
Expressiveness and Usability

Currently have over 10 graph primitives

- Traversal-based, Node-ranking, Global (connected component, MST)
- LOC under 300 for each primitive

Working on more graph primitives

- Graph coloring, Maximal Independent Set
- Community Detection
- Subgraph Matching
Future Works

- Dynamic graphs
- Global, neighborhood, and sampling operations
- Kernel fusion
- Scalability
  - GPUDirect for multi-GPU one-node
  - NVLink for multi-GPU multi-node
Conclusions

High-level Abstraction is essential for GPUs to make an impact in graph analytics.

Gunrock's data-centric, frontier-focused abstraction has good balance between expressiveness and performance.
Conclusions

High-level Abstraction is essential for GPUs to make an impact in graph analytics.

Gunrock's data-centric, frontier-focused abstraction has good balance between expressiveness and performance.

GPUs+Datacenter = Future of Large-Scale Graph Analytics