iBFS: Concurrent Breadth-First Search on GPUs

Hang Liu  H. Howie Huang  Yang Hu

The George Washington University
Graphs are Everywhere …
Many Algorithms Need Concurrent Traversals

- Single-Source Shortest Path
- Multi-Source Shortest Path
- All-Pairs Shortest Path
- Reachability Index Construction
- Centrality
Concurrent Traversals

Graph

BFS-1, Root Vertex 0

BFS-2, Root Vertex 3

BFS-3, Root Vertex 6

BFS-4, Root Vertex 8
Opportunities

Graph

BFS-1, Root Vertex 0

BFS-2, Root Vertex 3

Shared Frontiers

BFS-3, Root Vertex 6

BFS-4, Root Vertex 8
Frontier Sharing is Common

Average 39%
iBFS Techniques

- Joint Traversal
- *Bitwise Optimization*
- GroupBy
Joint Frontier Queue and Joint Status Array

Frontier Queue

Status Array

Number, e.g., 1 or 2 - Depth
F - Frontier
U - Unvisited

Joint Frontier Queue (JFQ)

Joint Status Array (JSA)

Vertex ID

Frontier Queue

Status Array

Vertex 0

Vertex 8
Benefits of Joint Traversal

- Eliminate redundant frontiers
- Load neighbors once
- Coalesce memory access to status array
Improving Frontier Sharing

❖ The performance of iBFS, to some extent, is determined by how many frontiers are shared at each level during the joint traversal of each group.

❖ For any BFS group, say group A, its sharing ratio is equal to the expected value of Speedup_A (Lemma 1).
It is observed that a higher sharing ratio of the initial levels can lead to a higher expected sharing ratio in later levels (Theorem 1 and Lemma 2)
It is observed that a higher sharing ratio of the initial levels can lead to a higher expected sharing ratio in later levels (Theorem 1 and Lemma 2).
GroupBy

It is observed that a higher sharing ratio of the initial levels can lead to a higher expected sharing ratio in later levels (Theorem 1 and Lemma 2)
GroupBy

It is observed that a higher sharing ratio of the initial levels can lead to a higher expected sharing ratio in later levels (Theorem 1 and Lemma 2)
It is observed that a higher sharing ratio of the initial levels can lead to a higher expected sharing ratio in later levels (Theorem 1 and Lemma 2)
GroupBy

It is observed that a higher sharing ratio of the initial levels can lead to a higher expected sharing ratio in later levels (Theorem 1 and Lemma 2)

How to Select Group A?

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Group A</th>
<th>Group B</th>
<th>Random group</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>16</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>54</td>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>72</td>
<td>54</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>90</td>
<td>72</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>90</td>
<td>90</td>
<td>0</td>
</tr>
</tbody>
</table>
Outdegree-Based GroupBy Rules

- Rule 1: The outdegrees of two source vertices are less than $p$.
  - Ensure small outdegrees of the source vertices

- Rule 2: Two source vertices connect to at least one common vertex whose outdegree is greater than $q$.
  - Ensure that other non-shared neighbors will not amortize the sharing ratio
Evaluation

- Implement in 4,000 lines of C++ and CUDA code
  - Also implement a CPU version
- Extend our SC’15 Enterprise work (GPU-based single source BFS)
- Run on NVIDIA K40 and K20 GPUs
- Scale to over 100 nodes on TACC Stampede
- Report TEPS (traversed edges per second)
Speedup Insights

Joint traversal improves performance by **40%**, Bitwise optimization additional **11x**, and GroupBy additional **2x**.
Scalability

Workload imbalance is the only issue for scalability. No communications during computation.
Conclusion

iBFS achieves 57,000 billion TEPS on 112 GPUs
❖ Joint Traversal
❖ Bitwise Optimization
❖ GroupBy Strategy

iBFS vs. MS-BFS [VLDB ’15] (State-of-the-art)
❖ MS-BFS extends single threaded BFS
❖ MS-BFS bitwise operation cannot support early termination
❖ MS-BFS does not have GroupBy
Acknowledgement
Thank You

{asherliu, howie, huyang}@gwu.edu

Please visit us at Poster #28
in Grand Ballroom A at 3:30PM today!