V-Combiner: Speeding-up Iterative Graph Processing on a Shared-memory Platform with Vertex Merging

Azin Heidarshenas†, Serif Yesil†, Dimitrios Skarlatos†, Sasa Misailovic†, Adam Morrison*, Josep Torrellas†

University of Illinois Urbana-Champaign†
Tel-Aviv University*

International Conference on Supercomputing (ICS), June 2020
Iterative graph processing

50-200 Iterations

Update all vertices in parallel

parallel for v in vertices
for u in v.neighbors
...
// update v

Converged?

yes

Finish

Computational complexity $\propto \#\text{Iterations}$

Page Rank
Community Detection
HITS
Belief Propagation
Graph processing can be approximate

**Example:** CEO of Company X wants to invest *only* on the most influential customers in their network

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Page Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0510103</td>
</tr>
<tr>
<td>3</td>
<td>0.0255164</td>
</tr>
<tr>
<td>4</td>
<td>7.3626e-05</td>
</tr>
<tr>
<td>2</td>
<td>5.16674e-05</td>
</tr>
</tbody>
</table>

Computing Page Ranks of Vertices 2 and 4 is useless.
Pruning graphs can be effective

Removing useless computation ↔ Removing certain vertices / edges (pruning)

Pre-processing

Build Graph

Graph Algorithm

Time

Compute

Original graph

Pre-processing

Prune

Build Graph

Graph Algorithm

end-to-end savings

Approximate graph
Overview of Sparsification and K-core

Sparsification
Prunes only edges, probabilistically from dense regions

K-core
Prunes vertices (along with their edges), until the remaining vertices have a degree of at least K

Limitations of Sparsification and K-core

Desirable speedup > 2x

Accuracy is the ratio of vertices found in the top ranking.

At the highest accuracy (~80%), Sparsification achieves 1.6x for Page Rank.

Accuracy is the ratio of vertices with correct communities.

High speedup is achieved only at low Accuracy (<60%) for Community Detection.
Addressing the Limitations

Sparsification\(^1\)  
Prunes *only edges*, probabilistically from dense regions

K-core\(^2\)  
Prunes *vertices* (along with their edges), until the remaining vertices have a degree of at least \(K\)

V-Combiner  
Prunes *and merges* certain vertices into hubs (*in the direction of information flow*), so that hubs stay connected to the rest of the graph

---

Overview of V-Combiner

More merging vs. pre-processing time vs. performance savings
## Different Vertex Merging Scenarios

<table>
<thead>
<tr>
<th>Example App.</th>
<th>Edges</th>
<th>Information flow</th>
<th>Information flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Rank, Comm. Detection</td>
<td>Directed</td>
<td>One-way</td>
<td>Merge in-neighbors</td>
</tr>
<tr>
<td>HITS</td>
<td>Directed</td>
<td>Two-way</td>
<td>Merge in-neighbors Merge out-neighbors</td>
</tr>
<tr>
<td>Belief Propagation</td>
<td>Undirected</td>
<td>Two-way</td>
<td>Merge all neighbors</td>
</tr>
</tbody>
</table>
Classification of Vertices in V-Combiner

**Supernode**: Large in-degree (but not too large)
- Large in-degree for supernode → More mergings per supernode

**Subnode**: Small in- and out-degree, at least one supernode in its out-neighborhood
- Small in- and out-degree for subnode → Less distortion after pruning

**Regular**: Neither a supernode nor a subnode
for e in edges
  //MERGE
  if e.dst is a subnode and e.src is NOT a subnode then
    // Increment in-degree of the supernode by one

  //PRUNE
  if e.src is a subnode and e.dst is NOT a subnode then
    // Decrement in-degree of the e.dst by one

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Old in-degree</th>
<th>New in-degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

One increment and one decrement cancel out.
No subnodes in the approximate graph
Recover using the in-neighbors’ values and the graph algorithm operator
• More efficient using Delta graph
• As if an extra iteration of the algorithm is run, but only for the subnodes

For Page Rank: $Pr[2] = 0.85 \frac{Pr[1]}{2} + 0.15$
Evaluation Setup

End-to-end speedup measured.

44 Intel Xeon cores, no hyper-threading and DVFS

4 graph applications:
• Page Rank (PR)
• Community Detection (CD)
• Hyperlink-Induced Topic Search (HITS)
• Belief Propagation (BP)

5 graph inputs
• Friendster social network (FS)
• Twitter social network (TW)
• Page-Level Domain graph (PLD)
• Arabic domain network (AR)
• Dbpedia network (DB)
Accuracy Metrics

**Top-K Accuracy:**
The ratio of vertices in the top ranking of the exact result that are also in the top ranking of the approximate result

- Page Rank
- HITS
- Belief Propagation

**Classification Accuracy:**
The ratio of vertices that have been correctly assigned to their communities

- Community Detection

Accuracy threshold of 90%.
End-to-End Performance

Build

Algorithm
End-to-End Performance: V-Combiner

1.25 end-to-end speedup at mean accuracy of 91.8%
End-to-End Performance: Sparsification

Sparsification fails to meet accuracy threshold in 1 benchmark
End-to-End Performance: K-core

K-core fails to meet accuracy threshold in 4 benchmarks
More in the Paper

- Details of other scenarios of the merging
- Choosing the merging parameters
- Algorithm performance and accuracy analysis
- Analysis of connectivity
- Analysis of the average length of the paths
- Analysis of pruning/merging parameters
- ...
Take-away

• Iterative graph processing is computationally expensive and can be approximate.

• V-Combiner is a pruning + merging + recovery technique

• It has the following advantages over the state-of-the-art pruning techniques:
  – Preserving average length of the paths
  – Maintaining connectivity
  – Improving load balancing
  – Modest pre-processing overhead
V-Combiner: Speeding-up Iterative Graph Processing on a Shared-memory Platform with Vertex Merging

Azin Heidarshenas†, Serif Yesil†, Dimitrios Skarlatos†, Sasa Misailovic†, Adam Morrison*, Josep Torrellas†

University of Illinois Urbana-Champaign†
Tel-Aviv University*