Memory-Optimized Distributed Graph Processing through Novel Compression Techniques

Katia Papakonstantinopoulou

Joint work with Panagiotis Liakos and Alex Delis

University of Athens

Athens Colloquium in Algorithms and Complexity
UoA, August 26th, 2016
Motivation

graph data whose size continuously grows
⇒
distributed graph processing systems (e.g., Pregel & Apache Giraph)

scale of real-world graphs hardens graph processing even in distributed environments
⇒
we need efficient distributed representations of such graphs

We address this problem by exploiting empirically-observed properties demonstrated by behavior graphs
Most of these approaches adopt the “think like a vertex” programming paradigm introduced with Pregel [5] (⇒ intuitive parallelizable algorithms).

A graph partitioned on a vertex basis in a distributed environment:

- The proposed frameworks [1, 6, 4, 7] fail to handle the huge scale of real-world graphs, as a result of ineffective memory usage [2].
- The partitioning hardens the task of compression.
The distributed graph-processing systems face significant memory-related issues. Some memory optimization approaches are:

Apache Giraph [1] (graph processing system that follows Pregel) with contributions by Facebook
- focused entirely on a more careful implementation for the representation of the out-edges of a vertex, without exploiting the redundancy in real-world graphs

Gbase[3] (a number of alternative compression techniques to reduce storage and hence network traffic and query execution time)
- does not follow the vertex-centric model
- requires decompression
We follow the Pregel paradigm and partition the graph vertices among the nodes of a distributed computing environment.

In this context, we present a number of novel techniques that:

1. offer space efficient-representations of the out-edges of vertices,
2. allow fast mining (in-situ) of the graph elements without the need of decompression,
3. enable the execution of graph algorithms in memory-constrained settings, and
4. ease the task of memory management, thus allowing faster execution.

Our work lies in the intersection of distributed graph processing systems and compressed graph representations.
In non-distributed settings, we can exploit the fact that vertices tend to exhibit similarities (copy property).

In order to achieve memory optimization, we need representations that allow mining of the graph’s elements without decompression.
Giraph’s adjacency-list representations: *ByteArrayEdges*

- The bytes required per out-neighbor are determined by the data type used for its id and weight; for integer numbers $4+4=8$ bytes are required.
Consider the following sequence of neighbors to be represented: (2, 9, 10, 11, 12, 14, 17, 18, 20, 127).

In the context of graph compression, Elias’ $\gamma$ coding is preferred for the representation of rather small values of $x$, whereas $\zeta$ coding is more proper for potentially large values.
Consider again the sequence of neighbors: \((2, 9, 10, 11, 12, 14, 17, 18, 20, 127)\).
Again using the sequence: \((2, 9, 10, 11, 12, 14, 17, 18, 20, 127)\).
How much more **space-efficient** is each of our three compressed out-edge representations compared to ByteArrayEdges?

Are our techniques competitive **speed-wise** when memory is not a concern?

How much more **efficient** are our compressed representations when the available memory is **constrained**?

Can we **execute algorithms** for large graphs in settings where it was **not possible** before?
Memory requirements of Giraph’s ByteArrayEdges and our three representations for the small and large-scale graphs of our dataset:

<table>
<thead>
<tr>
<th>graph</th>
<th>ByteArrayEdges</th>
<th>BVEdges</th>
<th>IntervalResidualEdges</th>
<th>IndexedBitArrayEdges</th>
</tr>
</thead>
<tbody>
<tr>
<td>uk-2007-05@100000</td>
<td>22.61MB</td>
<td>6.41MB</td>
<td>7.92MB</td>
<td>8.91MB</td>
</tr>
<tr>
<td>uk-2007-05@1000000</td>
<td>279.16MB</td>
<td>67.36MB</td>
<td>82.7MB</td>
<td>97.79MB</td>
</tr>
<tr>
<td>indochina-2004</td>
<td>1,511.67MB</td>
<td>442.34MB</td>
<td>646.03MB</td>
<td>554.23MB</td>
</tr>
<tr>
<td>hollywood-2011</td>
<td>1,381.91MB</td>
<td>287.53MB</td>
<td>613.52MB</td>
<td>676.88MB</td>
</tr>
<tr>
<td>uk-2002</td>
<td>2,733.6MB</td>
<td>1,092.82MB</td>
<td>1,224.07MB</td>
<td>1,255.67MB</td>
</tr>
<tr>
<td>arabic-2005</td>
<td>4,820.09MB</td>
<td>1,428.97MB</td>
<td>1,674.75MB</td>
<td>1,849.83MB</td>
</tr>
<tr>
<td>uk-2005</td>
<td>7,401.88MB</td>
<td>2,383.54MB</td>
<td>2,728.74MB</td>
<td>2,928.81MB</td>
</tr>
<tr>
<td>sk-2005</td>
<td>14,829.64MB</td>
<td>4,889.85MB</td>
<td>5,657.79MB</td>
<td>6,354.17MB</td>
</tr>
</tbody>
</table>
Execution time of PageRank algorithm for graph \textit{indochina-2004} using a setup of 2, 4, and 8 workers:
Execution time for each *superstep* of PageRank algorithm for graph *uk-2005* using 5 workers:

- **ByteArrayEdges** performance fluctuates due to extensive garbage collection.
Execution time of PageRank algorithm for graph *uk-2005*:

- **ByteArrayEdges**
- **BVEdges**
- **IntervalResidualEdges**
- **IndexedBitArrayEdges**

- IntervalResidualEdges and IndexedBitArrayEdges outperform ByteArrayEdges.
Our experimental results indicate significant improvements on space-efficiency for all proposed techniques.

- We reduced memory requirements up to 5 times in comparison with currently applied techniques.

In settings where earlier approaches were able to execute graph algorithms, we achieve significant performance improvements.

- We reduced execution time up to 31% due to memory optimization.

These findings establish our structures as the preferable option for web graphs, or any other type of behavior graphs.
Future Directions

- Design representation methods that favor mutations of the graph.
- Examine the compression of edge weights.
http://giraph.apache.org/.

An Experimental Comparison of Pregel-like Graph Processing Systems.  

GBASE: an efficient analysis platform for large graphs.  

Distributed GraphLab: A Framework for Machine Learning in the Cloud.  

Pregel: A System for Large-Scale Graph Processing.  
In ACM SIGMOD, 2010.

GPS: a graph processing system.  
In SSDBM, 2013.

[7] Da Yan, James Cheng, Yi Lu, and Wilfred Ng.  
Effective Techniques for Message Reduction and Load Balancing in Distributed Graph Computation.  
In WWW, 2015.
Thank you for your attention!

for further details visit:
http://hive.di.uoa.gr/network-analysis/

or email me at: katia@di.uoa.gr