Order or Shuffle:
Empirically Evaluating Vertex Order Impact on Parallel Graph Computations

George M. Slota\textsuperscript{1} \quad Sivasankaran Rajamanickam\textsuperscript{2} \\
Kamesh Madduri\textsuperscript{3}

\textsuperscript{1}Rensselaer Polytechnic Institute, \textsuperscript{2}Sandia National Labs, \textsuperscript{3}The Pennsylvania State University
slotag@rpi.edu, srajama@sandia.gov, madduri@cse.psu.edu

GABB2017 \quad 29 May 2017
Highlights

- We present an empirical evaluation of the effect of partitioning and ordering on the performance of several prototypical graph computations.
- We give experimental evidence that uncommonly-used quality metrics for partitioning and ordering might better predict performance for certain computations.
- We introduce a BFS-based ordering method that approximates RCM, gives better computational performance, and can be computed twice as fast.
Main Idea

- Generally understood that graph layout (partitioning + ordering) has a performance impact on the distributed graph computations (GABBs) comprising more complex analytics.
- All graph computations/analytics/frameworks require some layout choice.
- Goals of this preliminary study:
  - Examine the general relationship between layout choices and performance impact; especially relative to naïve methods such as randomization.
  - E.g., attempt to correlate partitioning and ordering quality metrics with real performance given some graph computation with known properties.
  - Identify avenues for future investigation.
Graph Layout

- **Graph Layout:** (vertex ordering + partitioning) pair
- We consider specifically 1D partitioning along with per-part vertex reordering

**Impacts and Considerations:**
- Load balance
- Communication reduction
- NUMA and memory access locality
- Overheads for partitioning and ordering methods
Partitioning

(1D) Partitioning: separating a graph into vertex-disjoint sets under certain balance constraints and optimization criteria

- **Balance constraints:** Computational workload per task/part
  - Workload is application specific
  - Might depend on number (or summed weights) of vertices and/or number of edges per task

- **Optimization Criteria:** Communication workload
  - Again, application specific
  - Might depend on total edges cut (edge cut), total summed size of one-hop neighborhoods (communication volume), or maximum communication load of a single task (max per-part cut/volume)
Partitioning

Methods for varying numbers of optimization criteria and balance constraints:

- **No criteria, no constraint**: Random/Hash partitioning
- **No criteria, single constraint**: Block methods
- **Single criteria, single constraint**: KaHIP tools, Scotch, many others
- **Single criteria, multi constraint**: (Par)METIS, PaToH
- **Multi criteria, multi constraint**: (Xtra)PuLP

*Can we empirically show what kinds of applications might benefit from these varying methods?*
Vertex Ordering

Vertex (re)ordering: creating some specific ordering of the numeric vertex identifiers of a graph

- Want to minimize some metric related to the numeric identifiers $f(v)$ of vertices $v \in V(G)$ in a graph
- Minimizing differences between identifiers of vertices in the neighborhood of some vertex might improve spatial locality
- Minimizing overall differences between neighborhoods of neighboring vertices might improve temporal locality
Ordering
Optimization Criteria

**Bandwidth:** Max distance from diagonal to nonzero in any row of the adjacency matrix

\[
\max\{|f(v_i) - f(v_j)| : \forall (v_i, v_j) = e \in E(G)\}
\]

**Co-location ratio:** How often nonzeros appear in consecutive columns of the same row of the adjacency matrix

\[
\frac{1}{|V|} \sum_{v \in V} \frac{1}{|N(v)|} \left\{1 + \sum_{i=2}^{|N(v)|} C(v_i)\right\} : C(v_i) = \begin{cases} 1, & \text{if } f(v_i) - f(v_{i-1}) = 1 \\ 0, & \text{otherwise} \end{cases}
\]

**Logarithmic Gap Arrangement:** Total spread of nonzeros in all rows of the adjacency matrix

\[
\sum_{v \in V} A(v) : A(v) = \sum_{i=2}^{|N(v)|} \log |f(u_i) - f(u_{i-1})|, u_i \in N(v)
\]
Vertex Ordering

Prior Methods:

- **No criteria:** Random methods
- **Community/clustering methods:** Label propagation, Rabbit Order
- **Neighborhood methods:** Gray Order, Shingle Order
- **Traversal-based methods:** BFS, Cuthill-McGee, Reverse Cuthill-McGee (RCM)

Our new method:

- BFS-based approximation of RCM that avoids explicit sorting
- Gives approximately $2 \times$ speedup relative to RCM
- We term this method combined with PuLP-MM as DGL (Distributed graph layout); by itself in the results, we just abbreviate it as BFS
Experimental Approach

- We consider several different distributed graph computations
- Run computations under several partitioning+ordering combinations
- Look at total execution times and end-to-end times (including partitioning and ordering)
- Also look at computation and communication times separately to see specific impacts of partitioning and ordering
Experimental Setup

- **Test Systems:**
  - *Compton* at Sandia National Labs and *Blue Waters* at NCSA - 2x Sandy Bridge CPUs with 64 GB memory

- **Layout Methods:**
  - Partitioners: METIS, METIS-M, PuLP-MM, PuLP-M, Random (w/random order)
  - Orderings: RCM, DGL, Random (w/PuLP-MM partition)

- **Graph Computations:**
  - PageRank, Subgraph Isomorphism, BFS, SSSP, SPARQL queries

- **Test Instances:**
  - Social Networks: LiveJournal, Orkut, Twitter
  - RDF Graphs: BSBM, LUBM, DBpedia
Partitioner Comparison

- Geometric mean over all graphs for 16 and 64 parts
- Max vertex imbalance, max edge imbalance
- Values given for edge cut ($EC$) and max per-part cut ($EC_{max}$) are mean improvements relative to the random method

<table>
<thead>
<tr>
<th>Partitioner</th>
<th>$V_{max}$</th>
<th>$E_{max}$</th>
<th>$EC$ (imp)</th>
<th>$EC_{max}$ (imp)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>avg/min/max</td>
<td>avg/min/max</td>
</tr>
<tr>
<td>Random</td>
<td>1.15</td>
<td>1.7</td>
<td>1/1/1</td>
<td>1/1/1</td>
</tr>
<tr>
<td>METIS</td>
<td>1.1</td>
<td>3.88</td>
<td>7.71/1.5/107</td>
<td>2.39/0.25/63</td>
</tr>
<tr>
<td>METIS-M</td>
<td>1.1</td>
<td>1.5</td>
<td>4.4/1.02/41</td>
<td>2.16/0.77/22</td>
</tr>
<tr>
<td>PuLP-M</td>
<td>1.1</td>
<td>1.5</td>
<td>5.5/1.17/64</td>
<td>2.1/0.54/23</td>
</tr>
<tr>
<td>PuLP-MM</td>
<td>1.1</td>
<td>1.5</td>
<td>5/1.19/63</td>
<td>3.18/2.54/204</td>
</tr>
</tbody>
</table>
Ordering Comparison

- Geometric mean ordering performance over all graphs and all five partitioning methods
- Performance measured by co-location ratio and logarithmic gap sum ratio (normalized version of gap sum metric)

<table>
<thead>
<tr>
<th></th>
<th>Co-loc. Ratio</th>
<th>Gap Sum Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(Higher better)</td>
<td>(Lower better)</td>
</tr>
<tr>
<td></td>
<td>BFS</td>
<td>RCM</td>
</tr>
<tr>
<td>16 parts</td>
<td>0.298</td>
<td>0.112</td>
</tr>
<tr>
<td>64 parts</td>
<td>0.274</td>
<td>0.119</td>
</tr>
</tbody>
</table>
PageRank

- Power method for fixed iterations (20)
- Parallelization: Bulk synchronous OpenMP + MPI
- Communication: Moderate, $\sim O(m)$ per iteration
- Computation: Moderate, $\sim O(m)$ per iteration
PageRank: Results

16 tasks

- Speedups vs. random methods for partitioning (top) and ordering (bottom)
- Overall mean speedup is best for PuLP-MM at $4.5 \times$ vs. $3.4-3.6 \times$ for other methods
- Mean speedup for BFS is $2.0 \times$ vs. $1.9 \times$ for RCM
PageRank: Execution Timelines
16 tasks running on WebBase

Clockwise from top left: Random, METIS, PuLP-MM, METIS-M
Subgraph Counting

- Alon et al. color-coding algorithm [Alon et al., 1995]
- Parallelization: Bulk synchronous OpenMP + MPI
- Communication: High, $\sim O(m2^k)$
- Computation: High, $\sim O(m2^k)$
Subgraph Counting: Results

16 tasks

- Overall mean speedup is best for P\textsuperscript{u}LP-M at $3.1 \times$ vs. $\sim 3.0 \times$ for other methods
- Mean speedup for BFS is $1.06 \times$ vs. $1.05 \times$ for RCM
- High memory storage/access per vertex minimizes cache improvements for ordering

![Graph showing speedup comparisons for different tasks and methods]
Subgraph Counting: Execution Timelines
16 tasks running on LiveJournal

Clockwise from top left: Random, METIS, PULP-MM, METIS-M
Subgraph Counting: End-to-end times
16 tasks running on LiveJournal

- End-to-end times for subgraph counting - partitioning + ordering, total computation, and total communication

![Bar charts showing end-to-end times for subgraph counting across different partitioners and datasets.](image)
Breadth-first Search

- Optimized “top-down” implementation
- Parallelization: Bulk synchronous MPI
- Communication: Low, $\sim O(m)$ total
- Computation: Low, $\sim O(m)$ total

Single Source Shortest Paths

- Optimized $\Delta$-stepping algorithm [Meyer and Sanders, 2003]
- Parallelization: Bulk synchronous MPI
- Communication: Low, $\sim O(m)$ total
- Computation: Low, $\sim O(m)$ total
BFS/SSSP: Results
64 tasks

- Show communication speedups for BFS (top) and computation speedups for SSSP (bottom)
- METIS gives best overall performance for BFS and SSSP; communication cost correlates highly with edge cut
- BFS: is $1.30 \times$ with BFS ordering vs. $1.26 \times$ for RCM
- SSSP: $1.08 \times$ with BFS ordering vs. $1.04 \times$ for RCM
SPARQL Query Processing

- Parallel implementation of RDF-3X store [Neumann and Weikum, 2010]
- Parallelization: MPI with $n$-hop replication
- Communication: Very Low, $\sim O(k)$
- Computation: High, $\sim$ polynomial


RDF special terms

RDFS special terms

ex:dog1 \(\text{rdf:}\text{type}\) ex:animal

ex:cat1 \(\text{rdf:}\text{type}\) ex:cat

ex:cat2 \(\text{zoo:}\text{host}\) ex:zoo1

ex:cat \(\text{rdfs:}\text{subClassOf}\) ex:animal

ex:cat \(\text{rdfs:}\text{range}\) ex:zoo1

2-hop replication ratios for each partitioning and sum query time (selection from Berlin SPARQL Benchmark) over all 3 graphs for each ordering methods

Performance impacts less obvious than with other analytics. Some dependence on 2-hop ratio as expected but paradoxical performance relationship with ordering (likely due to pre-processing methods)

<table>
<thead>
<tr>
<th>Partitioning</th>
<th>2-hop Rep. Ratio</th>
<th>Sum Query Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BSBM</td>
<td>LUBM</td>
</tr>
<tr>
<td>Random</td>
<td>7.256</td>
<td>10.58</td>
</tr>
<tr>
<td>METIS</td>
<td>5.566</td>
<td>9.714</td>
</tr>
<tr>
<td>METIS-M</td>
<td>5.577</td>
<td>9.146</td>
</tr>
<tr>
<td>PuLP-M</td>
<td>5.308</td>
<td><strong>8.944</strong></td>
</tr>
<tr>
<td>PuLP-MM</td>
<td><strong>5.112</strong></td>
<td>9.227</td>
</tr>
</tbody>
</table>
Experimental Limitations and Future Work

**Graph Instances:**
- Moderate problem sizes (only up to 2 B edges); was limited by subgraph counting and METIS memory requirements
- Run future experiments focusing on larger instances and techniques only applicable to such instances

**Analytic and layout:**
- Higher quality partitioners (for certain objectives) available (e.g. KaHIP); other ordering approaches (e.g. Rabbit order) available
- Analytics in experiments used only 1D partitioning; explore 2D and hybrid partitioning in conjunction with these partitioning and ordering methods

**Parallelization:**
- Only 16 and 64 tasks and bulk synchronous; limited by large number of experiments being run (Analytic × Partitioner × Ordering for each task count)
- Run to larger parallelization along with larger problem sizes; investigate methods or modifications applicable to asynchronous execution

**Overall Goal:** more explicitly quantify performance impact for partition and order quality given the memory access and communication patterns of some graph-based computation
Acknowledgments

- **Sandia and FASTMATH**
  - This research is supported by NSF grants CCF-1439057 and the DOE Office of Science through the FASTMath SciDAC Institute. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energys National Nuclear Security Administration under contract DE-AC04-94AL85000.

- **Blue Waters Fellowship**
  - This research is part of the Blue Waters sustained petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070, ACI-1238993, and ACI-1444747) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana Champaign and its National Center for Supercomputing Applications.

- **Kamesh Madduri’s CAREER Award**
  - This research was also supported by NSF grant ACI-1253881.
Conclusions

- Use of the secondary objective (max per-part cut) had a large impact on communication-heavy synchronous graph computations.

- Our BFS-based ordering demonstrated better average performance relative to RCM in all experiments; demonstrates that gap-based metrics might be a better way to quantify order quality for irregular graphs.

- Overall, performance impact scales in proportion with improvements in partitioning and ordering quality.

- The optimal choice of graph layout is highly dependent on the characteristics of the graph computation being processed.

Thanks! slotag@rpi.edu, www.gmslota.com
