Navigating the Maze of Graph Analytics Frameworks using Massive Graph Datasets

Nadathur Satish, Narayanan Sundaram, Mostofa Ali Patwary, Jiwon Seo, Jongsoo Park, M. Amber Hassaan, Shubho Sengupta, Zhaoming Yin, and Pradeep Dubey

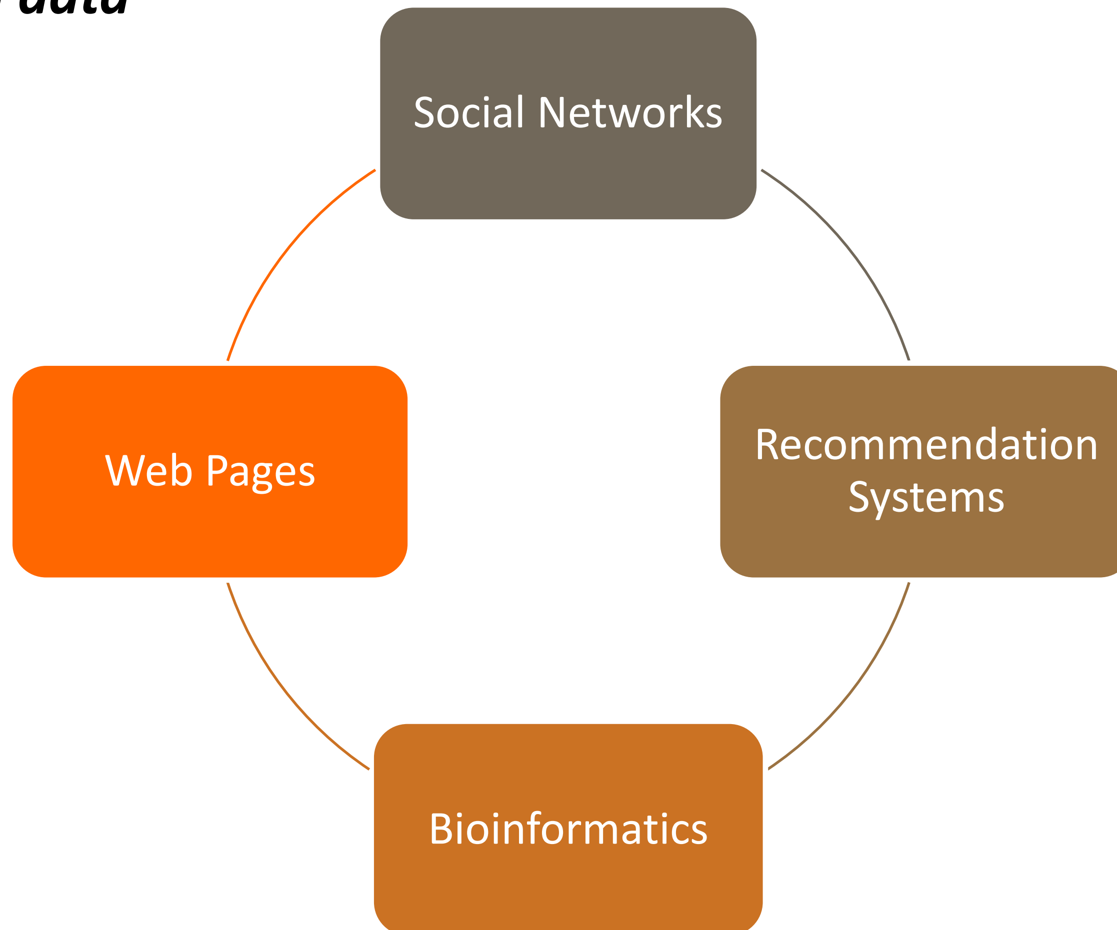
Presented by Guoyao Feng

Agenda

- Introduction
- Graph Algorithms
- Graph Analytics Frameworks
- Experimental Setup
- Experiment Results
- Optimizations and Recommendations
- Conclusion
- Discussion

Introduction: Background

- Growing interest in creating, storing and ***processing large graph data***



Introduction: Motivation

- Graph algorithm implementation
  - Irregular computation
  - Resource under-utilization
  - Large performance gap: Naive implementation vs. hand-optimized code
- No standard "building block"
  - Sparse matrix, vertex-centric programming, etc.
- Performance varies depending on both frameworks and algorithms
  - A headache to choose frameworks

Create a roadmap to improve graph frameworks' performance
Bridge the performance gap against native code

Graph Algorithms

- **PageRank**
  - Iteratively computes rank (web page popularity) for each vertex (web page) in a directed graph (reference web)

Probability of a random jump

Pagerank of vertex j at iteration t

$$PR^{t+1}(i) = r + (1-r) * \sum_{j|(j,i)\in E} \frac{PR^{t}(j)}{\text{degree}(j)}$$

- **Breadth Frist Search (BFS)**
  - Traverses an undirected, unweighted graph from one vertex and compute the minimal distance
  - In each iteration:

$$Distance(i) = \min_{j|(j,i)\in E} Distance(j) + 1$$

Figures from "Navigating the maze of graph analytics frameworks using massive graph datasets"

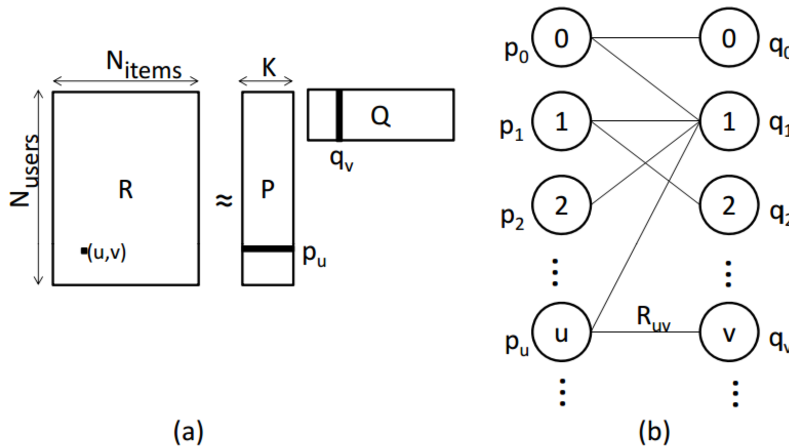
Graph Algorithms

- **Triangle Counting**
  - Each pair of vertices in an edge compare their neighbourhood lists and count the number of shared neighbours

$$N_{triangles} = \sum_{i,j,k,i<j<k} E_{ij} \wedge E_{jk} \wedge E_{ik}$$

Existence of edge between i and k

- **Collaborative Filtering**
  - Estimates the rating of an item by a given user



$$\min_{\mathbf{p},\mathbf{q}} \sum_{(u,v)\in R} (\mathbf{R}_{uv} - \mathbf{p}_u^T \mathbf{q}_v)^2 + \lambda_p ||\mathbf{p}_u||^2 + \lambda_q ||\mathbf{q}_v||^2$$

Figures from "Navigating the maze of graph analytics frameworks using massive graph datasets"

Graph Analytics Frameworks: GraphLab

- Graph algorithms expressed as programs running on a vertex
- Each vertex reads incoming messages, updates states and sends message asynchronously
- **PageRank**

Algorithm 1: Vertex program for one iteration of page rank

begin
  $PR \longleftarrow r$
  for $msg \in$ incoming messages **do**
    $PR \longleftarrow PR + (1 - r) * msg$
  Send $\frac{PR}{degree}$ to all outgoing edges

- **BFS**

Algorithm 2: Vertex program for one iteration of BFS.

begin
  for $msg \in$ incoming messages **do**
    $Distance \longleftarrow \min(Distance, msg + 1)$
  Send $Distance$ to all outgoing edges

Figures from "Navigating the maze of graph analytics frameworks using massive graph datasets"

Graph Analytics Frameworks: CombBLAS

- Provides linear algebra primitives for graph analytics
- Operates on sparse matrix and vectors
- Edge-based partitioning (2-D partitioning)
- **PageRank**

Page rank values at iteration t+1

Adjacency matrix

$$\mathbf{p}_{t+1} = r\mathbf{1} + (1-r)\mathbf{A}^T\tilde{\mathbf{p}}_t$$

- **BFS**

vector of starting vertices

$$\mathbf{v} = \mathbf{A}^T\mathbf{s} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 1 \end{pmatrix}$$

Next vectices to explore

Figures from "Navigating the maze of graph analytics frameworks using massive graph datasets"

Graph Analytics Frameworks: SociaLite

- Declarative language running recursive queries
- Horizontally partitioned for parallelism
- **PageRank**

Page rank of node n at iteration t+1

$$\textsc{Rank}[n](t+1, \$\textsc{Sum}(v)) \text{ :- } v = r$$

$$\text{:- } \textsc{InEdge}[n](s), \textsc{Rank}[s](t, v_0), \textsc{OutDeg}[s](d), v = \frac{(1-r)v_0}{d}$$

- **Triangle Counting**

$$\textsc{Triangle}(0, \$\textsc{Inc}(1)) : -\textsc{Edge}(x, y), \textsc{Edge}(y, z), \textsc{Edge}(x, z)$$

Figures from "Navigating the maze of graph analytics frameworks using massive graph datasets"

Graph Analytics Frameworks: Giraph

- Bulk synchronous graph processing system on Hadoop
- Vertex partitioning (1-D partitioning)
- **Collaborative Filtering**
  - Gradient Descent
  - In one iteration, every vertex
    1. Aggregates information from neighbours
    2. Sends updated vector to neighbours

$$\mathbf{p}_u^* = \mathbf{p}_u + \gamma_t \sum_{v|(u,v)\in E} [\mathbf{R}_{uv}\mathbf{q}_v - (\mathbf{p}_u^T\mathbf{q}_v)\mathbf{q}_v - \lambda_p\mathbf{p}_u]$$

$$\mathbf{q}_v^* = \mathbf{q}_v + \gamma_t \sum_{u|(u,v)\in E} [\mathbf{R}_{uv}\mathbf{p}_u - (\mathbf{p}_u^T\mathbf{q}_v)\mathbf{p}_u - \lambda_q\mathbf{q}_v]$$

Figures from "Navigating the maze of graph analytics frameworks using massive graph datasets"

Graph Analytics Frameworks: Galois

- Framework designed for irregular computation
- Work-item based parallelization
- Automatous scheduling and scalable data structures
- Runs on a single node
- **Triangle Counting**

Algorithm 4: Galois program for Triangle counting.

begin
  Graph G
  numTriangles = 0
  foreach *(Node n: G) in parallel* **do**
    S_1 = { m in G.neighbors(n) | m > n }
    for *(m in S1)* **do**
      S_2 = { p in G.neighbors(m) | p > m }
      numTriangles $\longleftarrow$ numTriangles + $|S_1 \cap S_2|$

Figures from "Navigating the maze of graph analytics frameworks using massive graph datasets"

Experimental Setup

Dataset	# Vertices	# Edges
Facebook [1] Wikipedia [2] LiveJournal [2] Netflix [3]	2,937,612 3,566,908 4,847,571 480,189 users 17,770 movies	41,919,708 84,751,827 85,702,475 99,072,112 ratings
Twitter [4] Yahoo Music [5]	61,578,415 1,000,990 users 624,961 items	1,468,365,182 252,800,275 ratings
Synthetic Graph500	536,870,912	8,589,926,431
Synthetic Collaborative Filtering	63,367,472 users 1,342,176 items	16,742,847,256 ratings

Experiment Results: Native Code

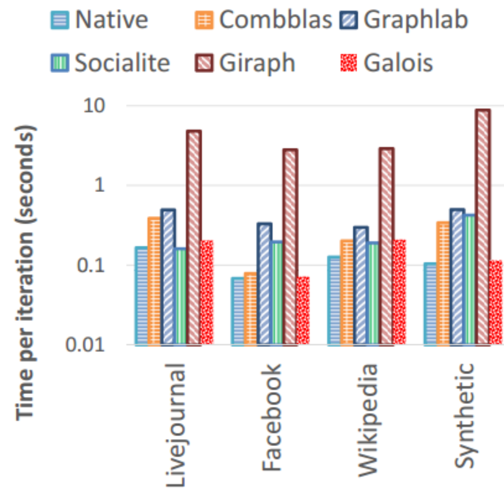
- Native hand-optimized implementation efficiency

Algorithm	Single Node		4 Nodes	
	H/W limitation	Efficiency	H/W limitation	Efficiency
PageRank	Memory BW	78 GBps (92%)	Network BW	2.3 GBps (42%)
BFS	Memory BW	64 GBps (74%)	Memory BW	54 GBps (63%)
Coll. Filtering	Memory BW	47 GBps (54%)	Memory BW	35 GBps (41%)
Triangle Count.	Memory BW	45 GBps (52%)	Network BW	2.2 GBps (40%)

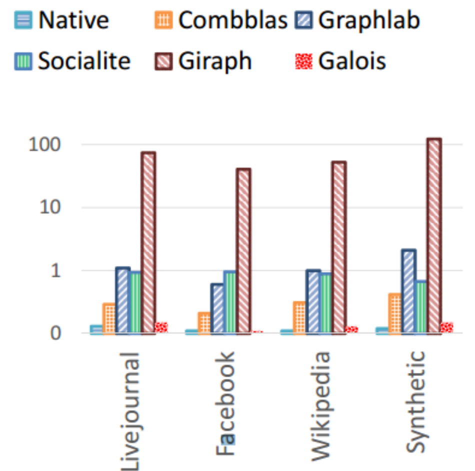
Figures from "Navigating the maze of graph analytics frameworks using massive graph datasets"

Experiment Results: Single Node

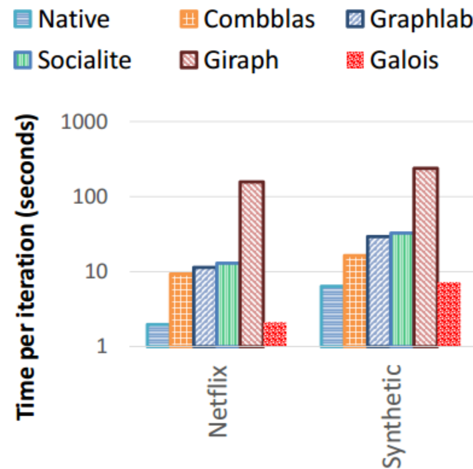
- Performance on a single node with real world and synthetic graphs



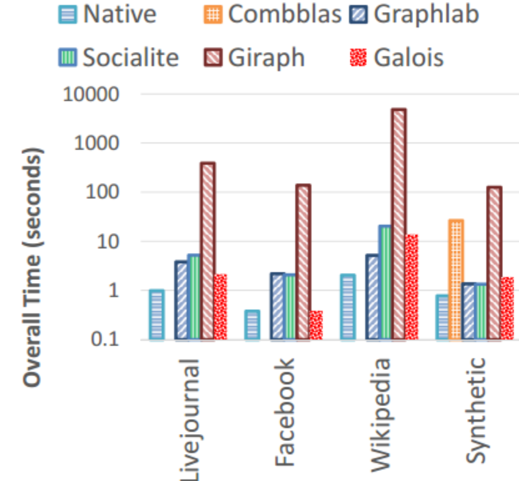
(a) PageRank



(b) Breadth-First Search



(c) Collaborative Filtering

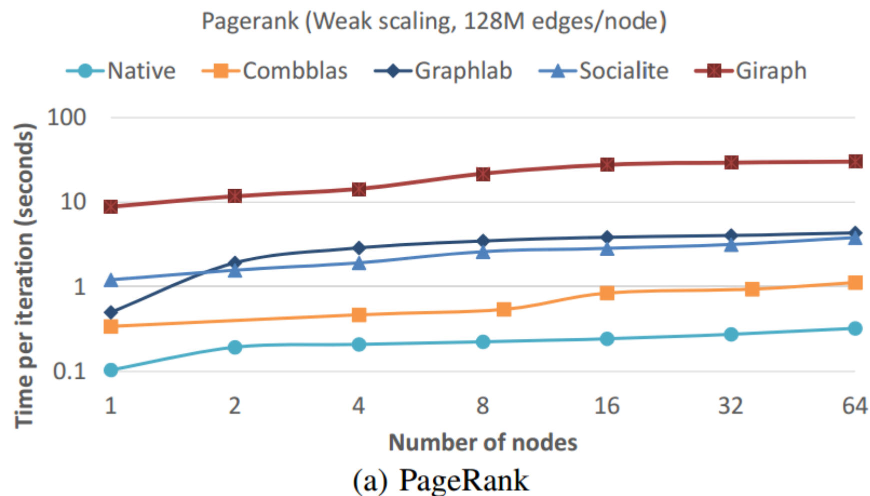


(d) Triangle counting

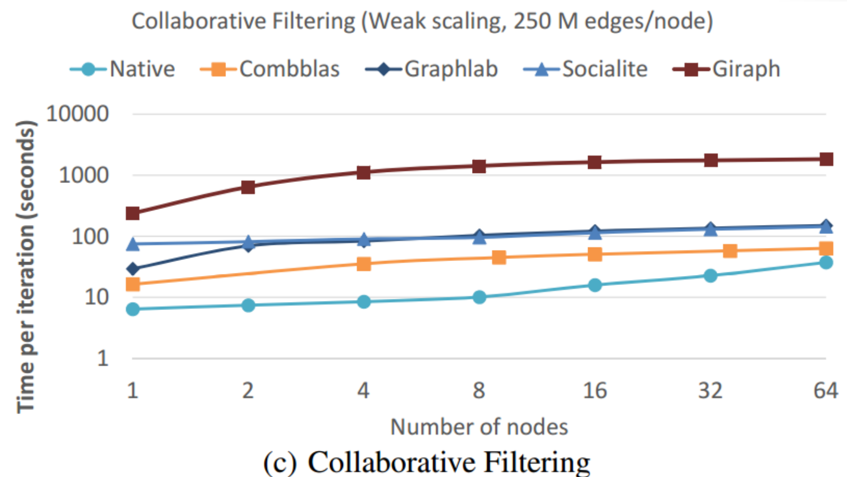
Figures from "Navigating the maze of graph analytics frameworks using massive graph datasets"

Experiment Results: Multiple Nodes

- Performance on multiple nodes using large synthetic graphs



(a) PageRank

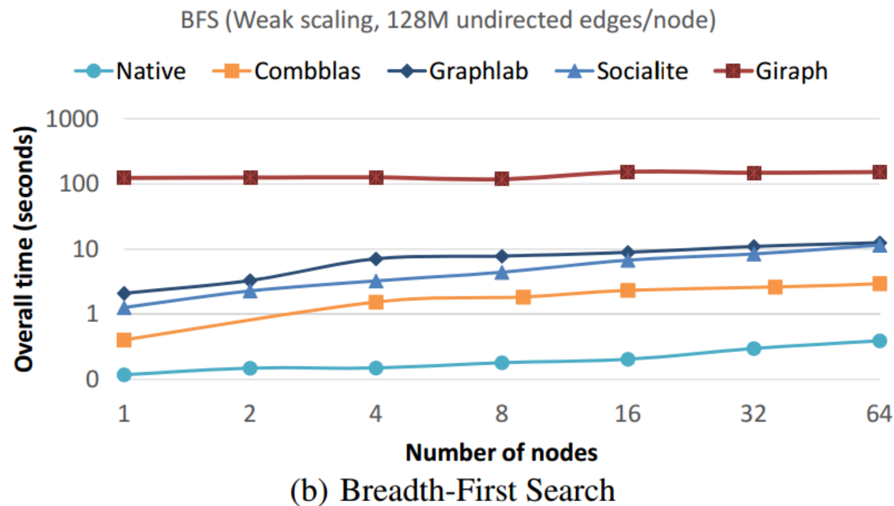


(c) Collaborative Filtering

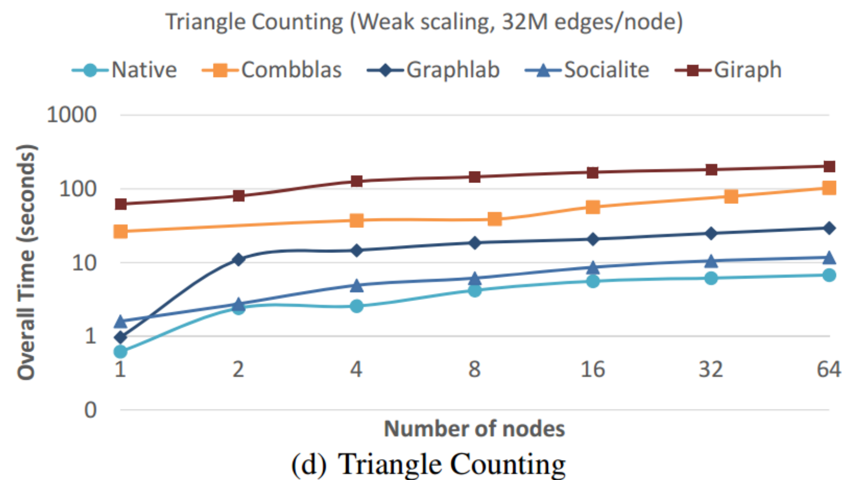
Figures from "Navigating the maze of graph analytics frameworks using massive graph datasets"

Experiment Results: Multiple Nodes

- Performance on multiple nodes using large synthetic graphs



(b) Breadth-First Search



(d) Triangle Counting

Figures from "Navigating the maze of graph analytics frameworks using massive graph datasets"

Experiment Results: Summary

- Slowdown factors of framework performance against native code on a *single node*

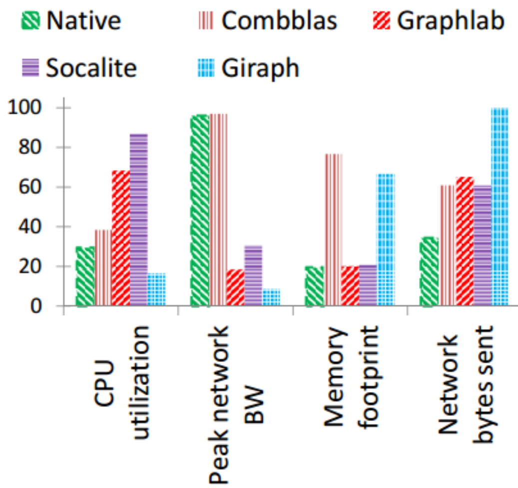
Algorithm	CombBLAS	GraphLab	SociaLite	Giraph	Galois
PageRank	1.9	3.6	2.0	39.0	1.2
BFS	2.5	9.3	7.3	567.8	1.1
Coll. Filtering	3.5	5.1	5.8	54.4	1.1
Triangle Count.	33.9	3.2	4.7	484.3	2.5

- Slowdown factors of framework performance against native code on *multiple nodes*

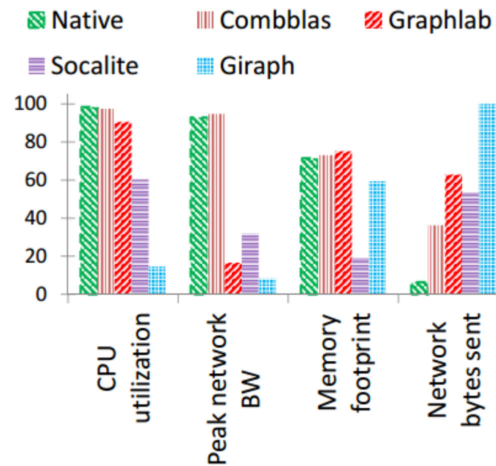
Algorithm	CombBLAS	GraphLab	SociaLite	Giraph
PageRank	2.5	12.1	7.9	74.4
BFS	7.1	29.5	18.9	494.3
Coll. Filtering	3.5	7.1	7.0	87.9
Triangle Count.	13.1	3.6	1.5	54.4

Figures from "Navigating the maze of graph analytics frameworks using massive graph datasets"

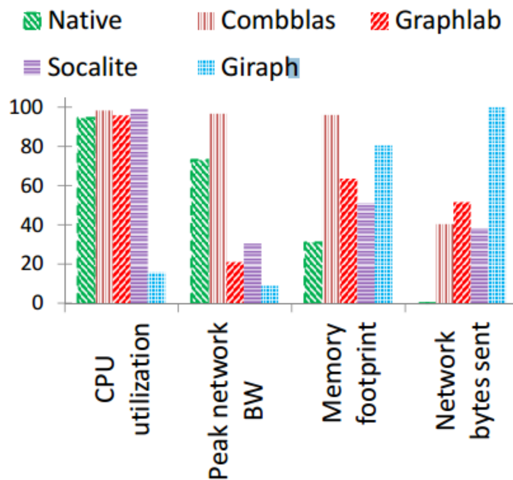
Experiment Results: Framework Analysis



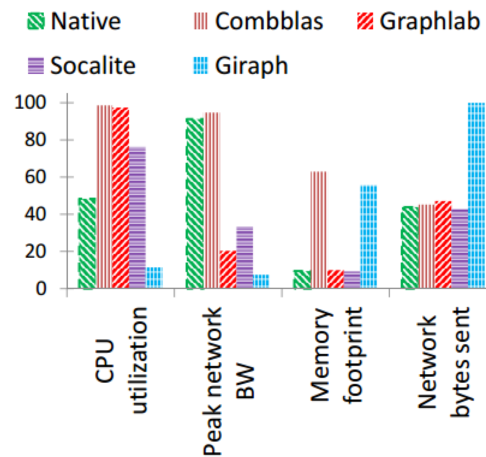
(a) PageRank



(b) Breadth-First Search



(c) Collaborative Filtering

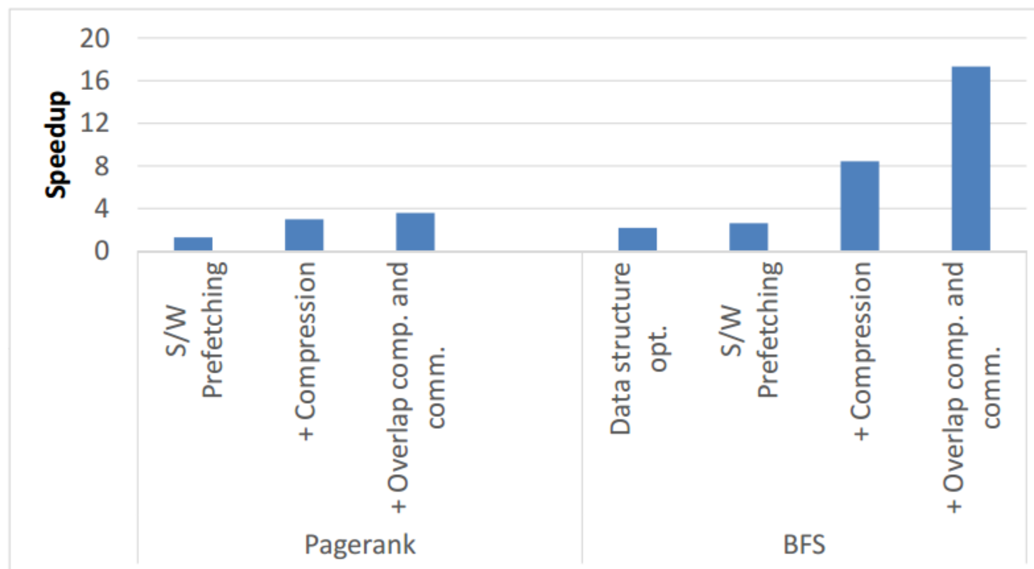


(d) Triangle Counting

Figures from "Navigating the maze of graph analytics frameworks using massive graph datasets"

Optimizations

- Key optimizations in native implementation
  - Data structures
  - Data compression
  - Overlap of Computation and Communication
  - Message passing mechanisms
  - Partitioning schemes



Figures from "Navigating the maze of graph analytics frameworks using massive graph datasets"

Recommendations

- GraphLab
  - Mainly limited by network bandwidth ⇒ MPI
  - Data compression, prefetching, computation and communication overlap
- CombBLAS
  - Use bit-vector for compression in BFS
  - Techniques for inter-operation optimization
- Galois
  - Implemented most optimizations
- Giraph
  - Boost network bandwidth
  - Data compression
  - Reduce memory buffer size for higher memory efficiency
- SociaLite
  - Most algorithms limited by network bandwidth
  - Data compression

Conclusion

- Compares graph frameworks in terms of programming model and implementation of multiple algorithms
- Exposes performance gap (2-30X) between graph frameworks and hand-optimized native code
- Analyzes CPU usage, memory footprint, and network traffic to explain performance gap
- Shows performance gains of optimization techniques in native code and recommendations for graph frameworks

*"our goal is **not** to come up with a new graph processing benchmark or propose a new graph framework, but to analyze existing approaches better to **find out where they fall short**"*

Discussion

- The optimization techniques are know when the native code is implemented. Why not apply them directly to the frameworks if possible?
- The paper analyze framework in terms of CPU usage, memory footprint and network traffic. How can we reason about the performance difference based on the programming models?
  - For example, vertex programming vs. parallel graph library
- What are the pros and cons of …
  - Using only one graph framework
  - Selecting the framework to use based on the algorithm
  - Simply developing the native implementations

References

1. C. Wilson, B. Boe, A. Sala, K. P. N. Puttaswamy, and B. Y. Zhao. User interactions in social networks and their implications. In EuroSys, pages 205–218, 2009.
2. T. Davis. The University of Florida Sparse Matrix Collection. http://www.cise.ufl.edu/research/sparse/matrices.
3. J. Bennett and S. Lanning. The Netflix Prize. In KDD Cup and Workshop at ACM SIGKDD, 2007.
4. H. Kwak, C. Lee, H. Park, and S. B. Moon. What is twitter, a social network or a news media? In WWW, pages 591–600, 2010.
5. Yahoo! - Movie, Music, and Images Ratings Data Sets. http://webscope.sandbox.yahoo.com/catalog.php?datatype=r.
6. R. C. Murphy, K. B. Wheeler, B. W. Barrett, and J. A. Ang. Introducing the graph 500. Cray User's Group (CUG), 2010.