High Performance Computing Paper Review

Hiroki Kanezashi

13M38152

Reviewed Paper 1

"Mizan: A System for Dynamic Load Balancing in Large-scale Graph Processing"

[EuroSys '13 Proceedingsof the 8th ACM European Conference on Computer Systems]

Zuhair Khayyat[1] Karim Awara[1] Amani Alonazi[1]

Hani Jamjoom[2] Dan Williams[2] Panos Kalnis[1]

[1]King Abdullah University of Science and Technology, Saudi Arabia

[2]IBM T. J. Watson Research Center, Yorktown Heights, NY

Reviewed Paper 2

"Breaking the Speed and Scalability Barriers for Graph Exploration on Distributed-memory Machines"

[International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2012]

Fabio Checconi, Fabrizio Petrini[1],

Jeremiah Willcock, Andrew Lumsdaine[2],

Anamitra Roy Choudhury, Yogish Sabharwal[3]

[1]IBM T. J. Watson Research Center, Yorktown Heights, NY 10598

[2]CREST, Indiana University Bloomington, IN 47405

[3]IBM India Research, New Delhi, DL 110070, India

Reviewed Paper 3

“Parallel Breadth-First Search on Distributed Memory Systems”

[SC '11 Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis]

Aydın Buluç and Kamesh Madduri

Computational Research Division Lawrence Berkeley National Laboratory Berkeley, CA

Outline

1. Introduction
2. Dynamic Behavior of Algorithms
3. Mizan
4. Implementation
5. Evaluation
6. Related Work
7. Future Work
8. Conclusion

- My Impressions

1. Introduction

- To make better use of graph data and mining algorithms, many platforms are proposed.
  - Pregel
  - HADI
  - PEGASUS
  - X-RIME
- <u>This paper focused on Pregel</u>.

About Pregel

- Pregel is used for large graph minings recently.
  - Message passing-based
  - Performs better than MapReduce
  - Built on the Bulk Synchronous Parallel (BSP) model
    - Computation is divided into “supersteps”.
    - These supersteps are separated by global barrier.

Load balancing

- In a Pregel system, balanced computation and communication is fundamental.
- Pregel and other implemented platforms have systems to do so.
  - Giraph
  - GoldenOrb
  - Hama
  - Surfer

Recent Approaches for Balancing

- Use hash- / range-based graph partitioning
- Entrust developers to use their own partitioning scheme or pre-partition data
- Provide sophisticated techniques
- Utilize distributed <u>data</u> stores and indexing on vertices and edges
- Perform coarse-grained load balancing

The Efficacy of Recent Methods

- Are these method effective for large graph?
  - They are static approaches.
  - Developers should predict the behavior.
  - Developers should know runtime characteristics.

2. Dynamic Behavior of Algorithms

- There are many factors affect the runtime performance in Pregel.
  - When vertices are active, they compute, send and receive messages.
  - Some messages are sent to another workers (nodes).
- Some factors can be masked by overlapping or running many vertices.

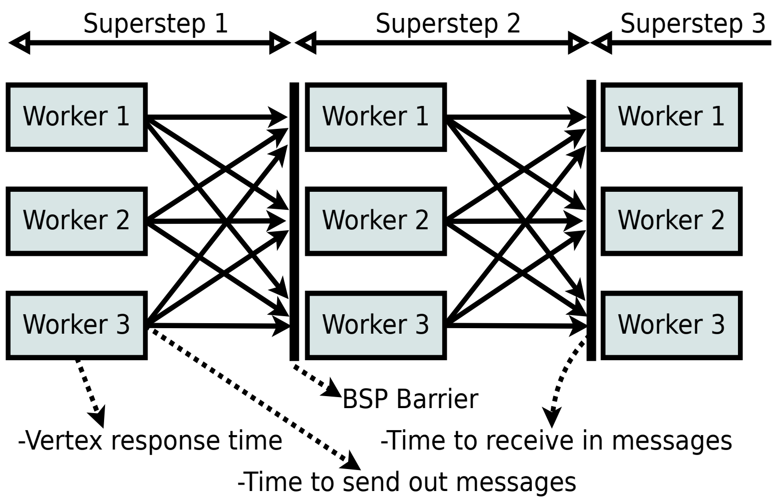


Figure 1. Factors that can affect the runtime in the Pregel framework

All figures and tables are retrived from the reviewed paper.

Workload imbalance

- It is difficult to achieve a balanced workload for graph structure and algorithms behavior.
- Some nodes may take a long time to compute many nodes, send and receive many messages.
- As this paper introduced, many approaches are used in Pregel systems.

Evaluation of recent methods

- At first, three common approaches are evaluated using these datasets.
  - Hash-based
  - Range-based
  - Minimum-cuts

$G(N,E)$	$\|N\|$	$\|E\|$
kg1	1,048,576	5,360,368
kg4m68m	4,194,304	68,671,566
web-Google	875,713	5,105,039
LiveJournal1	4,847,571	68,993,773
hollywood-2011	2,180,759	228,985,632
arabic-2005	22,744,080	639,999,458

Table 1. Datasets—N, E denote nodes and edges, respectively. Graphs with prefix kg are synthetic.

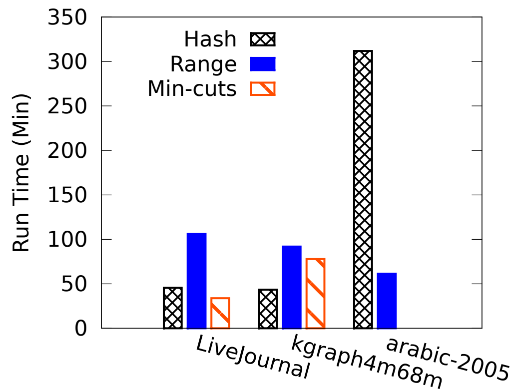


Figure 2. The difference in execution time when processing PageRank on different graphs using hash-based, range-based and min-cuts partitioning. Because of its size, `arabic-2005` cannot be partitioned using min-cuts (ParMETIS) in our local cluster.

Categolize Graph Algorithms

- Not only graph structure, but also graph algorithms can affect the workload balance.

- They can be categolized according to communication characteristics.
  - Stationary
  - Non-stationary

Categolize Graph Algorithms

Stationary

- Distributions of sent messages do not change.
- Example
  - PageRank
  - Diameter estimation
  - Finding weakly connected components

Non-Stationary

- Destinations or sizes of messages can change.
- Example
  - Distributed minimal spanning tree construction (DMST)
  - Graph queries
  - Simulations on social network graphs

Categolize Graph Algorithms

- While running two algorithms in 21 nodes, Non-stationary algorithms (DMST) sent more and more messages.

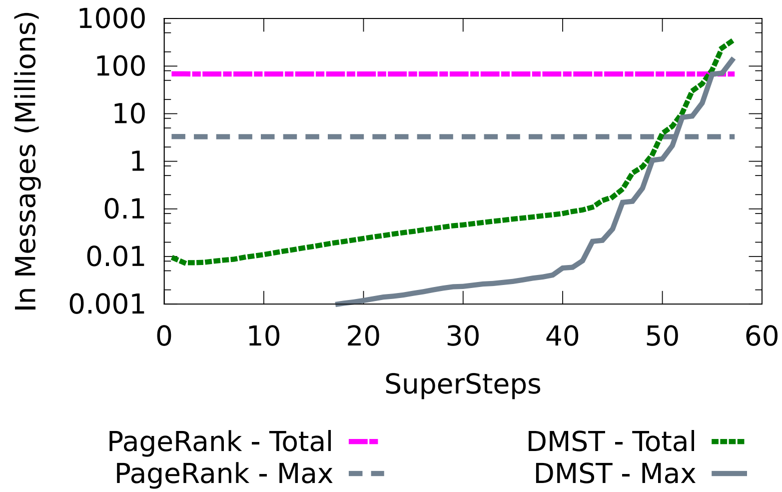


Figure 3. The difference between stationary and non-stationary graph algorithms with respect to the incoming messages. *Total* represents the sum across all workers and *Max* represents the maximum amount (on a single worker) across all workers.

3. Mizan

- Common point with Pregel
  - A BSP-based graph processing system
  - Reads graph and partition before supersteps
- Different point with Pregel
  - Focuses on efficient dynamic load balancing of both computation and communication
  - Moves some vertices across workers (migration)
    - Distributed runtime monitoring
    - Distributed migration planner

Monitoring

- Mizan system monitors three metrics
  - The number of outgoing messages
    - Counts messages to other vertices in remote workers.
    - Local outgoing ones never affect network cost.
  - The number of total incoming messages
    - Counts ones from remote vertices and locally generated.
  - The response time (execution time)
    - Measured for each vertex at each superstep.

Monitoring

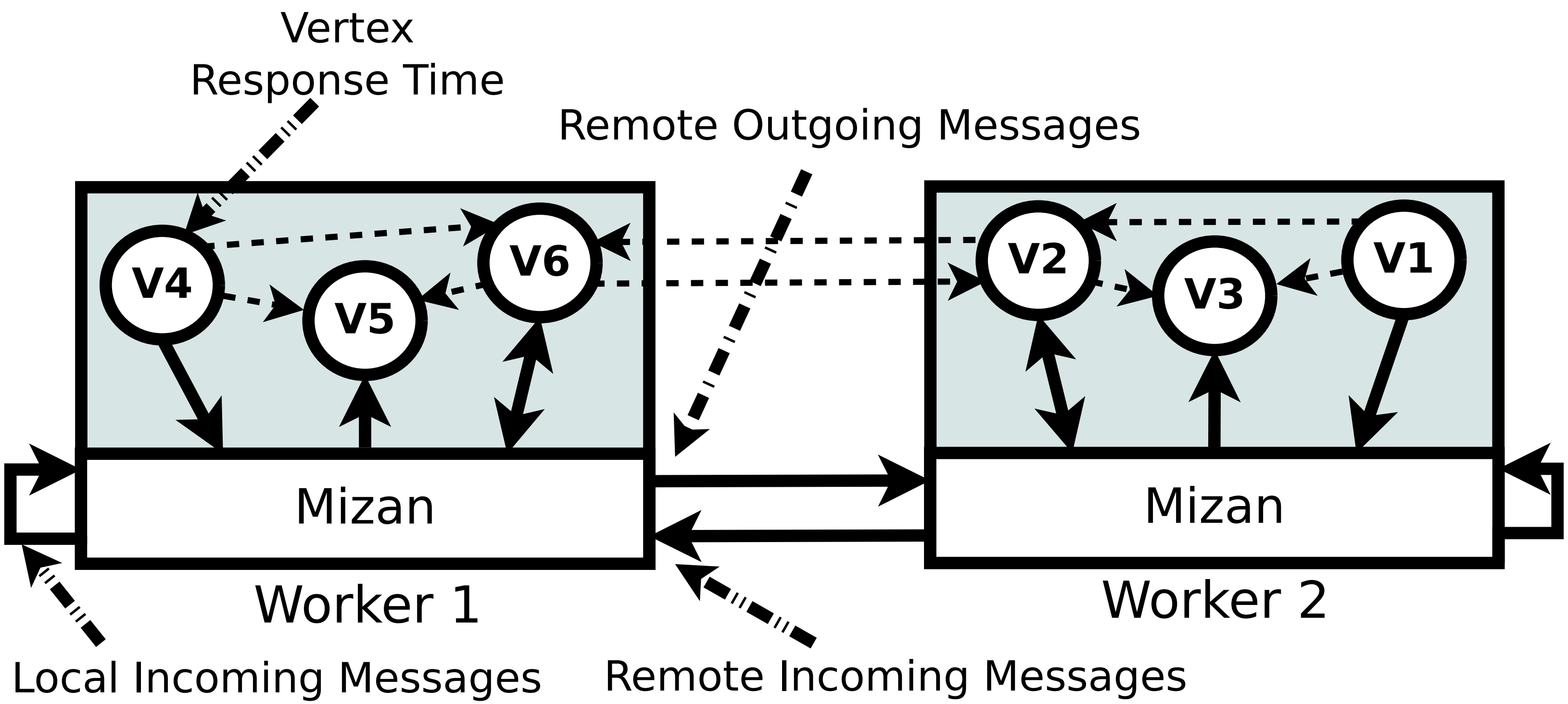


Figure 4. The statistics monitored by Mizan

Migration Planning

1. Identify the source of imbalance.
2. Select the migration objective.
3. Pair over-utilized and under-utilized workers.
4. Select vertices to migrate.
5. Migrate vertices.

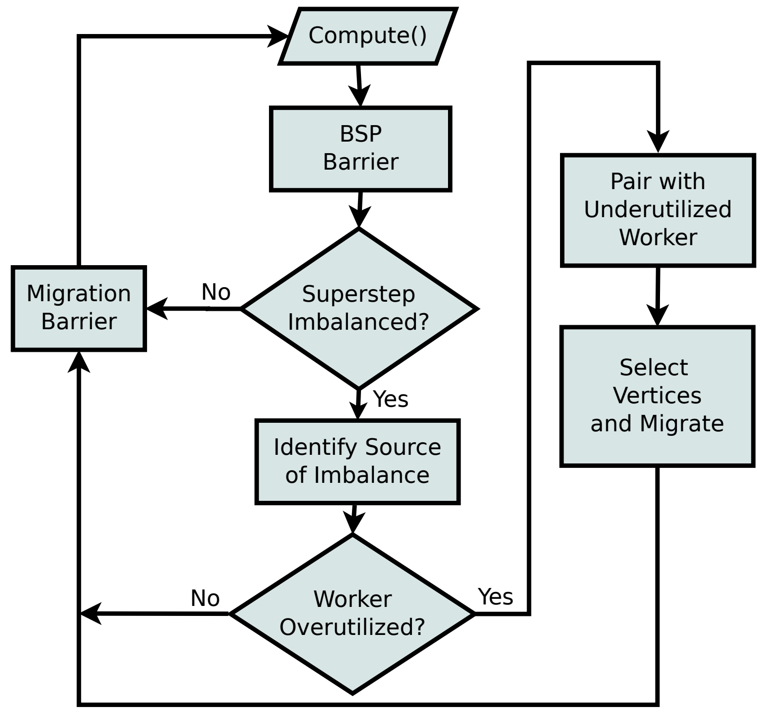


Figure 6. Summary of Mizan's Migration Planner

Detecting Imbalance

1. Check outlier workers comparing the summary statistics of all ones.
2. Select the objective what it will optimize with calculating correlation of the metrics.
   - To balance outgoing messages
   - To balance incoming messages
   - To balance computation time (default)

Selecting Vertices

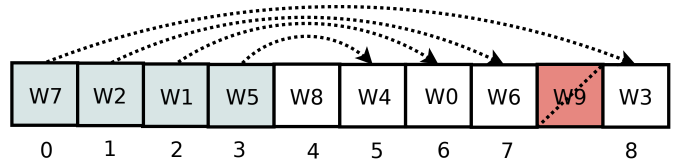


Figure 7. Matching senders (workers with vertices to migrate) with receivers using their summary statistics

3. Pair workers by metrics.
   - If there are *n* workers, top *n* and *n-i* workers should be pair.
   - Workers without enough memory are unavailable.
4. Select vertices to move in order to minimize the difference of sum of workloads to be migrated and sum of those outliers.

Migrating Vertices

5. To migrate vertices, each worker will do that in the migration barrier:
   - Sending worker
     - Sends encoded stream with vertex ID, state, edge information and received messages.
     - After the stream is sent, deletes the vertices.
   - Receiving worker
     - Receives vertices and messages.
     - Prepares to run them in the next superstep.

4. Implementation

- In Mizan, each worker has 4 modules
  - BSP Processor
    - Implemented Pregel APIs
  - Storage Manager
    - Maintains the graph data always correct
  - Communicator
    - Uses MPI to communicate with other workers
  - Migration Planner
    - Operate across barriers

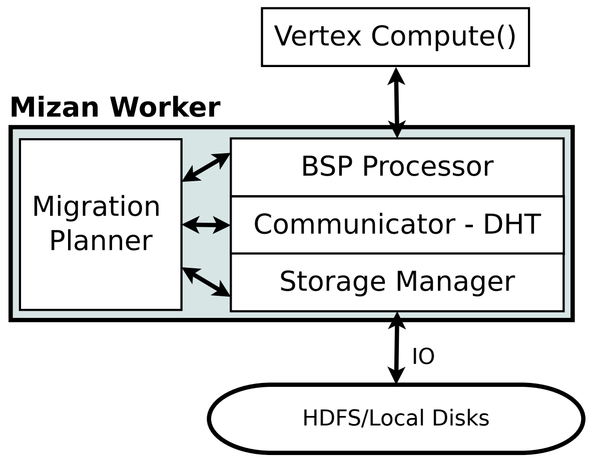


Figure 8. Architecture of Mizan

Vertex Ownership

- Mizan will not maintain centralized vertex management especially with huge one.

- A distributed hash table (DHT) is used to implement a distributed lookup service.
  - It stores key (ID) and value (physical location) sets.
  - A "home" worker maintains current location of assigned vertices.

Distributed Hash Table Updates

- Vertex whose home worker is 3 will migrate from Worker 1 to 2.
  1. The vertex migrates.
  2. Destination worker inform migration to home worker.
  3. The home worker updates DHTs.

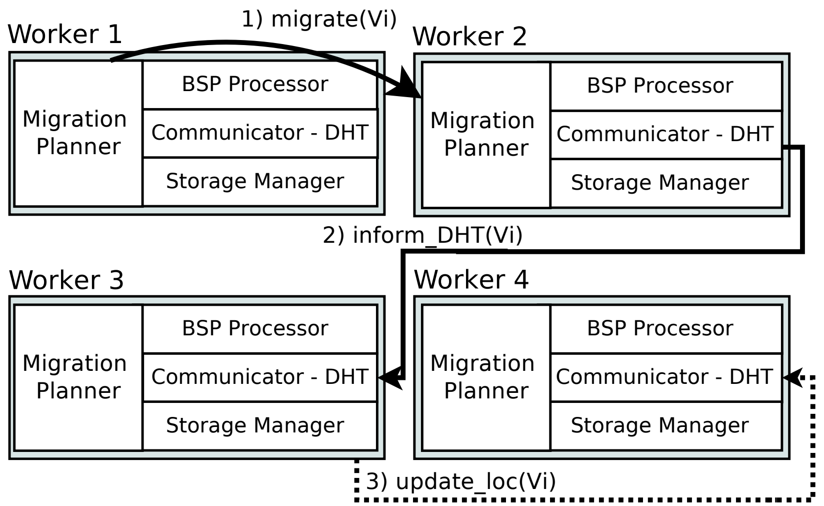


Figure 9. Migrating vertex v_i from Worker 1 to Worker 2, while updating its DHT home worker (Worker 3)

Migrating Large Vertices

- If a vertex has many messages, it costs very much when it migrates.
- Mizan uses a delayed migration process.
  - It takes two supersteps.
  - Only moves the vertex's information and the ownership, not large message information.

Delayed Migration

1. An ownership of the migrated vertex is moved to *Worker_new*.
   – Messages will be sent to *Worker_new*.
2. *Worker_old* sends the edge information.
3. The vertex is fully migrated.

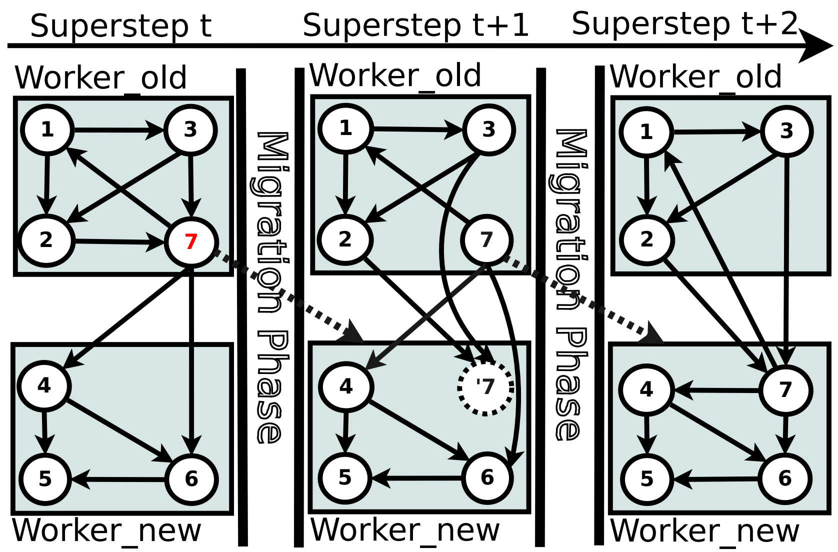


Figure 10. Delayed vertex migration

5. Evaluation

- Mizan was implemented using C++ and MPI.
  - Compared against Giraph (Java based Plegel clone)
- Computation nodes
  - Local clusters with 21 machines, mix of i5 and i7, 16GB RAM
  - IBM Blue Gene/P, 1024 PowerPC-450 CPUs with 4 cores, 4GB RAM

$G(N, E)$	$\|N\|$	$\|E\|$
kg1	1,048,576	5,360,368
kg4m68m	4,194,304	68,671,566
web-Google	875,713	5,105,039
LiveJournal1	4,847,571	68,993,773
hollywood-2011	2,180,759	228,985,632
arabic-2005	22,744,080	639,999,458

Table 1. Datasets—N, E denote nodes and edges, respectively. Graphs with prefix kg are synthetic.

Synthetic datasets are generated by Kronecker generator.

Giraph vs. Mizan

- First, Static Mizan was compared to Giraph.
  - In Static Mizan, dynamic migrations never occur and graph pre-partitioning is used.
- In Figure 11 and 12, Static Mizan is faster than Giraph.

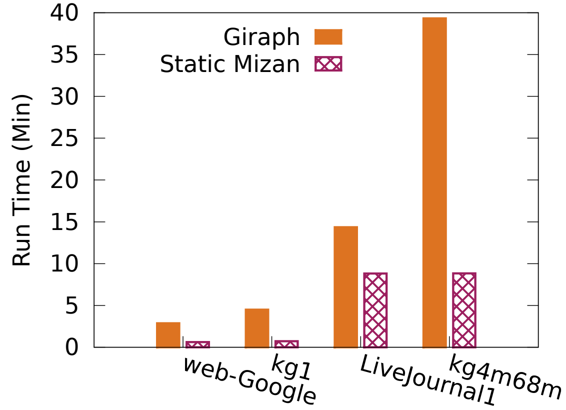


Figure 11. Comparing Static Mizan vs. Giraph using PageRank on social network and random graphs

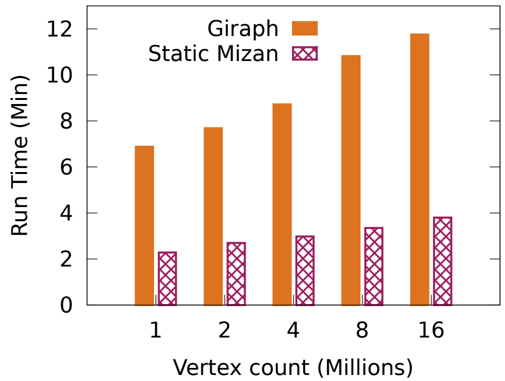


Figure 12. Comparing Mizan vs. Giraph using PageRank on regular random graphs, the graphs are uniformly distributed with each has around 17M edge

Dynamic Vertex Migration

- Hash-based and METIS partitioning make little differences.
- However, effectiveness of dynamic migration is showed in the range-based partitioning.

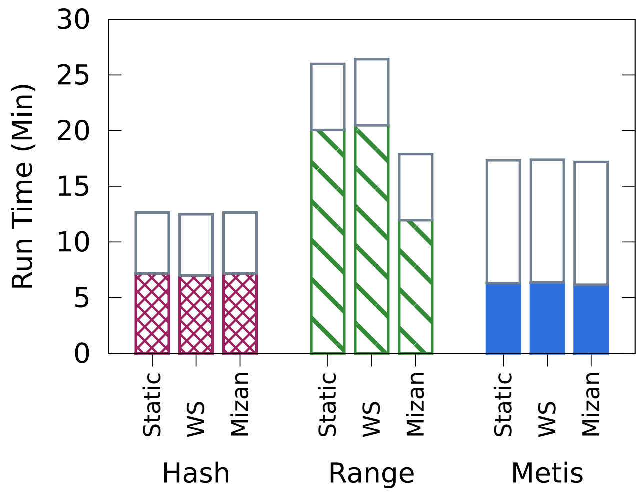
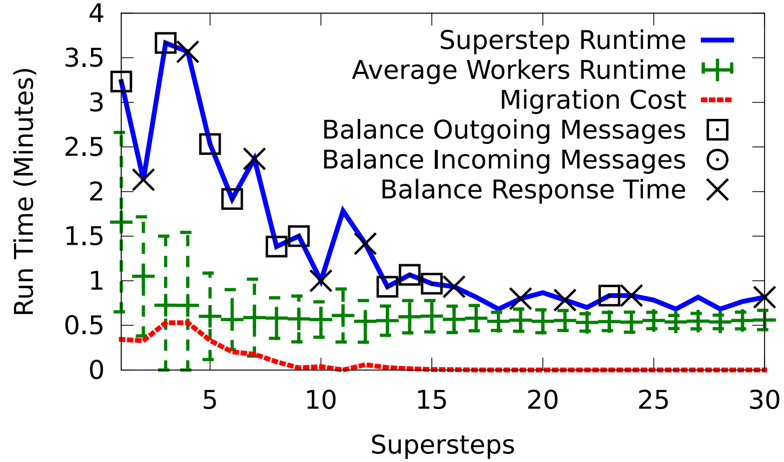
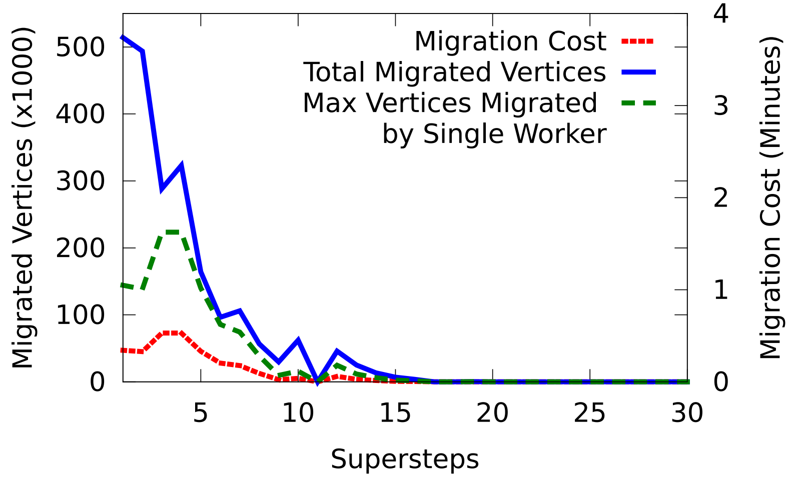


Figure 13. Comparing Static Mizan and Work Stealing (Pregel clone) vs. Mizan using PageRank on a social graph (`LiveJournal1`). The shaded part of each column represents the algorithm runtime while unshaded parts represents the initial partitioning cost of the input graph.

Dynamic Vertex Migration

- In PageRank algorithm, the dynamic migration is correlated with runtime reduction.
  - It would take more supersteps when workload is balanced in other algorithms.





Dynamic Vertex Migration

- In the both of algorithms, Mizan resulted in about 200% speed up.

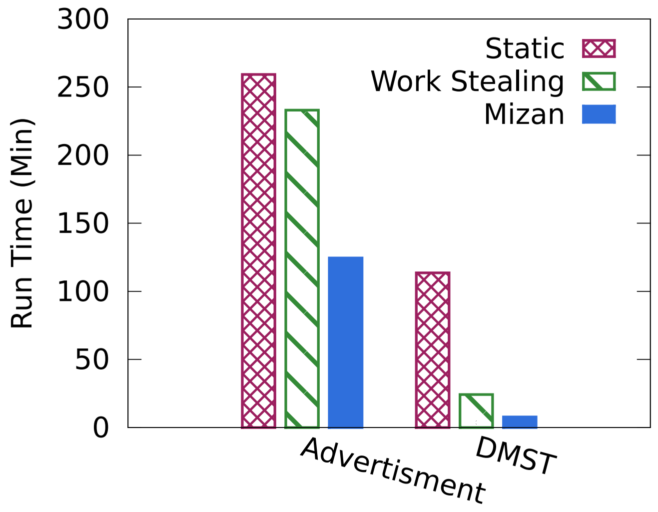
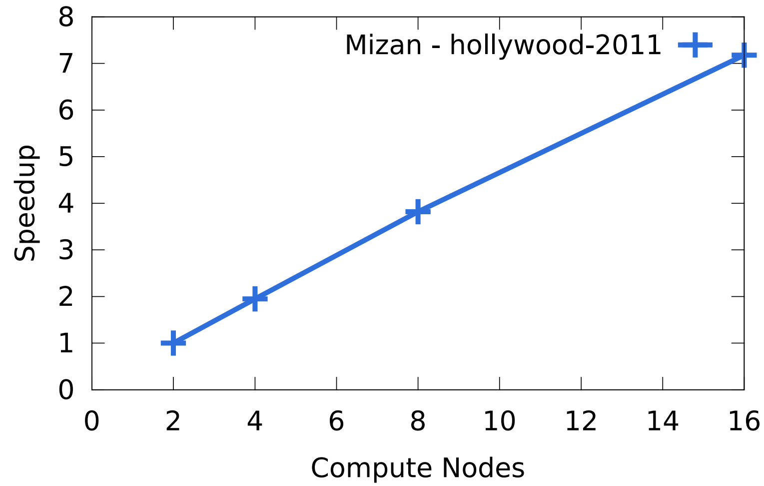
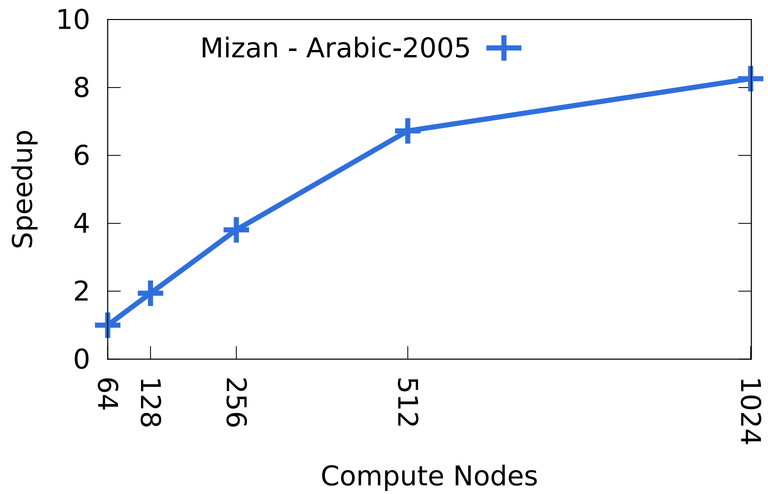


Figure 17. Comparing Work stealing (Pregel clone) vs. Mizan using DMST and Propagation Simulation on a metis partitioned social graph (`LiveJournal1`)

Scalability of Mizan

- While using smaller data sets, it achieved the scalability.
- However, in 1024 nodes with larger graph, the scale became flatten.
  - The reason may be that computing time cannot hide too much communication time.





6. Related Work

- Pregel and its Clones
- Power-law Optimized Graph Processing Systems
- Shared Memory Graph Processing Systems
- Specialized Graph Systems

7. Future Work

- In order to reduce migration costs, the frequency of them should be reduced.
  - Vertex replication proposed by PowerGraph may be useful.
- In the evaluations, graph was partitioned only by single application or algorithm.
  - If experiments using multiple algorithms on the same graph are conducted, better result will gain.

8. Conclusion

- A Pregel system called *Mizan* was presented.
  - Identifies the cause of workload imbalance.
  - Conducts fine-grained vertex migration.
- Performance evaluation showed it had most efficiency and robustness.
  - It also showed the linear scalability to hundreds nodes.

Contributions

- Analyzed some graph algorithm characteristics that can contribute to imbalanced computation of a Pregel system.
- Proposed a dynamic migration model based on runtime monitoring of vertices.
- Implemented Mizan in C++ and MPI as an optimized Pregel system.
- Deployed Mizan and showed linear scalability.

My Impression 1

- Even Mizan assumes many nodes computation, data sets might not large enough.
  - The number of nodes is 23 million at most.
  - I wanted to see the result of evaluations using billion-scale graph.
  - I think that scalability would be less and less with using larger network.
    - Migration cost would be more visible.

My Impression 2

- It has substantial analyzing and evaluations on graph algorithms.
  - Thought of categories "(Non-)stationary algorithms" seems to be useful.
  - I found that many graph partitioning algorithms for preprocessing are worth trying.

My Impression 3

- The series of algorithms and implements of Mizan will serve as a reference of my research.
  - I am researching an efficient traffic simulations using million-scale road network.
  - The main problem in these simulations is also load unbalancing caused by vehicles.
    - We cannot predict the number of vehicles (messages).
  - The idea of “delayed migration” can be implemented to my simulations.

Thank you for listening