The Little Engine(s) That Could: Scaling Online Social Networks

Josep M. Pujol, Vijay Erramilli,
Georgos Siganos

Outline

- Introduction
- Social Partitioning and Replication
- SPAR Join Partitioning and Replication Algorithm
- SPAR System Architecture
- Evaluation
- Conclusion

Introduction

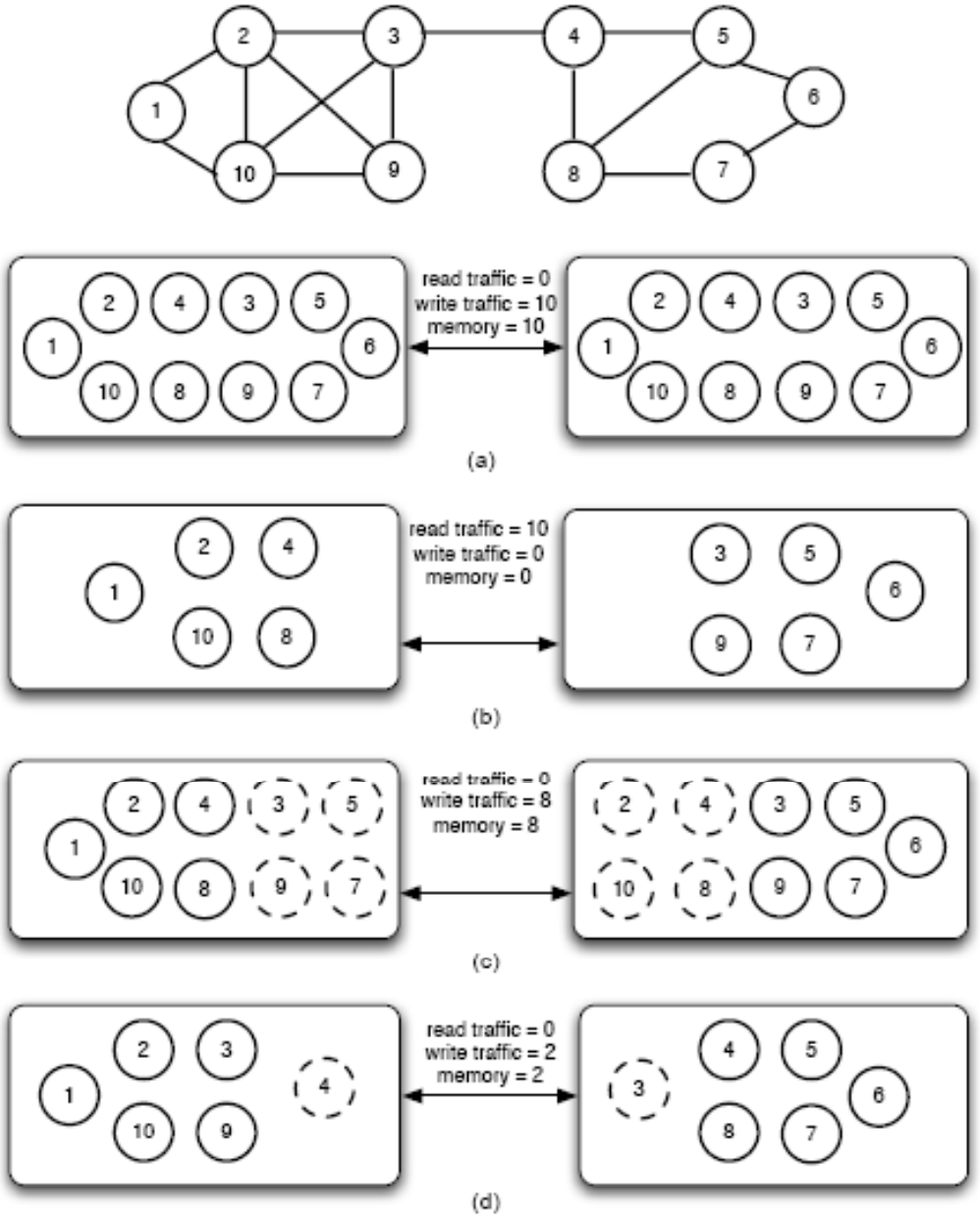
- Scaling real system and traditional solutions
  - Vertical scaling method
    - Upgrade existing hardware
    - Expensive
  - Horizontal scaling method
    - Engage a higher number of cheap commodity servers and partition the load
    - Require that data can be partitioned into disjoint components

Introduction

- OSN scaling
  - Operations are mainly based on the data of a user and his neighbors
  - Data highly interconnected
  - No disjoint partitions
  - High inter-server traffic load or replication load

Social Partitioning And Replication

- Constricts all relevant data for a user on a server
- Enforces local semantics at the data level
- Avoids the problems of querying traffic among multiple servers across a network



(a) Full Replication

(b) Partition using DHT (random partitioning)

(c) Random Partitioning with replication of the neighbors,

(d) SPAR, socially aware partition and replication of the neighbors

SPAR Join Partitioning and Replication Protocols

- SPAR requirements
  - Maintain local semantics
    - for every master replica of a users, either a master or a slave replica of all his direct neighbors is co-located in the same server
  - Balance loads
    - An even distribution of masters among servers
  - Be amenable to machine failures
  - Be stable
  - Minimize replication overhead

SPAR Join Partitioning and Replication Protocols

- Problem formulation
  - G=(V,E): social graph
  - N=|V|: total number of users
  - M: the number of available servers
  - p_{ij} : 1 if and only the primary of users i is assigned to partition j; 0 else.
  - r_{ij} :1 if and only if a replica of user I assigned to partition j; 0 else.
  - $\varepsilon_{ii'}$: 1 if user i and i' capture the friendship relationship

SPAR Join Partitioning and Replication Protocols

- Min_ Replica problem

$$\min \sum_i \sum_j r_{ij}$$

$$\text{s.t.} \quad \sum_{\forall j} p_{ij} = 1 \tag{1}$$

$$p_{ij} + \epsilon_{ii'} \leq p_{i'j} + r_{i'j} + 1, \forall i, j, i' \tag{2}$$

$$\sum_i (p_{ij}) = \sum_i (p_{i(j+1)}), 1 \leq j \leq M - 1 \tag{3}$$

$$\sum_j r_{ij} \geq k, \forall i \in V \tag{4}$$

SPAR Join Partitioning and Replication Algorithm

- A greedy optimization solution using local information
- Node addition
  - A new node is assigned to the partition with the fewest number of master replicas
  - K slave replicas are assigned to random partitions
- Node removal
  - Master and slave replications are removed
  - The State of users having an edge with it are updated

SPAR Join Partitioning and Replication Algorithm

- Edge addition between u and v
  - Check whether both masters are already co-located with each other or with a master's slave
  - If no, calculate the number of replicas generated for three possible configurations
    - No movements of masters
    - The master of u goes to the partition containing the master of v
    - The master of v goes to the partition containing the master of u

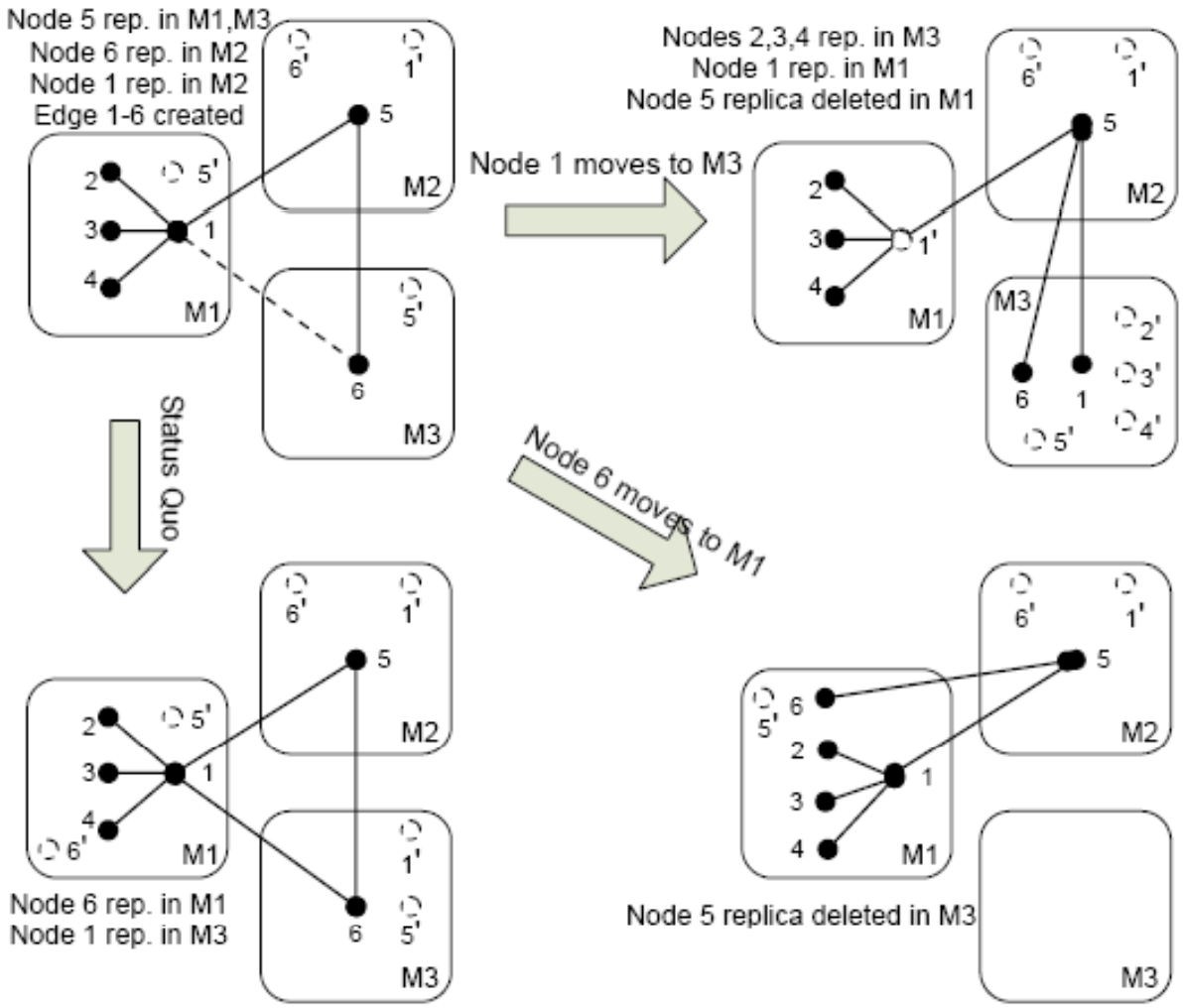
SPAR Join Partitioning and Replication Algorithm

- Edge addition
  - No movement
    - A replica is added if it does not already exist in the complementary partition
  - Master u goes to the partition containing master v
    - Create a slave replica of itself in the old partition to service the master of the neighbors left in that partitions
    - Masters of neighbors create a slave replica in new partition
    - Remove slave replica in old partition only to serve the master of node u
  - Master v goes to the partition containing master u
    - Similar with the above

SPAR Join Partitioning and Replication Algorithm

- Edge addition
  - Greedily choose the configuration that yield the smallest aggregate number of replicas
  - Subject to the load balance of the masters: movement happens to a partition:
    - With fewer masters
    - For which the savings in terms of number of replicas is greater than the current ratio of load imbalance between partitions.

SPAR Join Partitioning and Replication



SPAR Join Partitioning and Replication Algorithm

- Edge removal between u and v
  - removes the replica of u in the partition holding the master of node v if no other node requires it

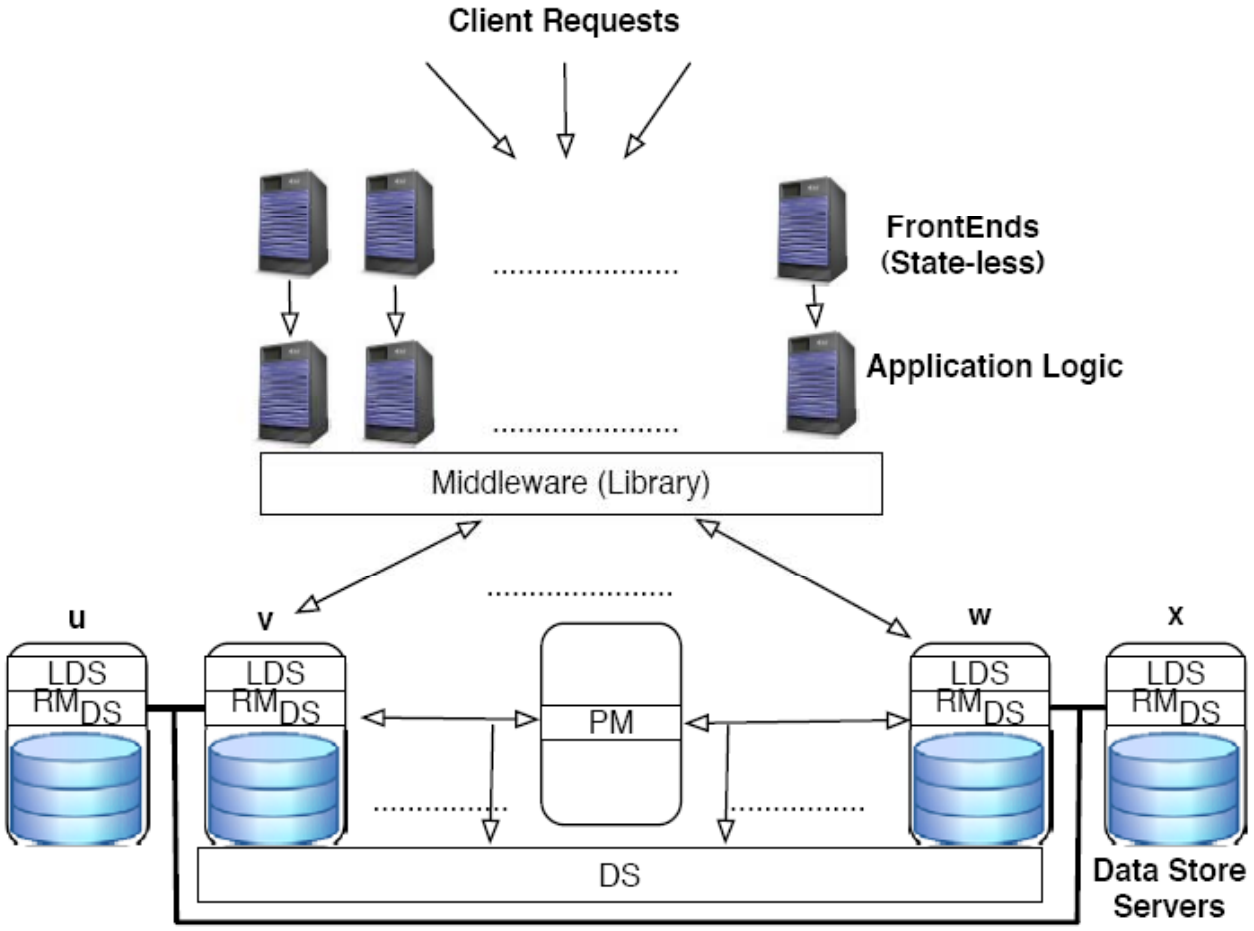
SPAR Join Partitioning and Replication Algorithm

- Server addition
  - Force redistribution of the masters from the other servers to the new one
    - select at least $\frac{N}{M^2+M}$ replicated masters from the M server and move them to the new server
  - Let the re-distribution of the masters be the result of the node and edge arrival processes.

SPAR Join Partitioning and Replication Algorithm

- Server removal
  - Re-allocate the master nodes to the remaining servers
  - Highly connected nodes, with potentially many replicas to be moved, get to first choose the server they go to, based on the ratio of neighbors that already exist on that server.
  - Follows simple water-filling strategy

SPAR System Architecture



DS: Directory Service

LDS: Local Directory Service

PM: Partition Manager

RM: Replication Manager

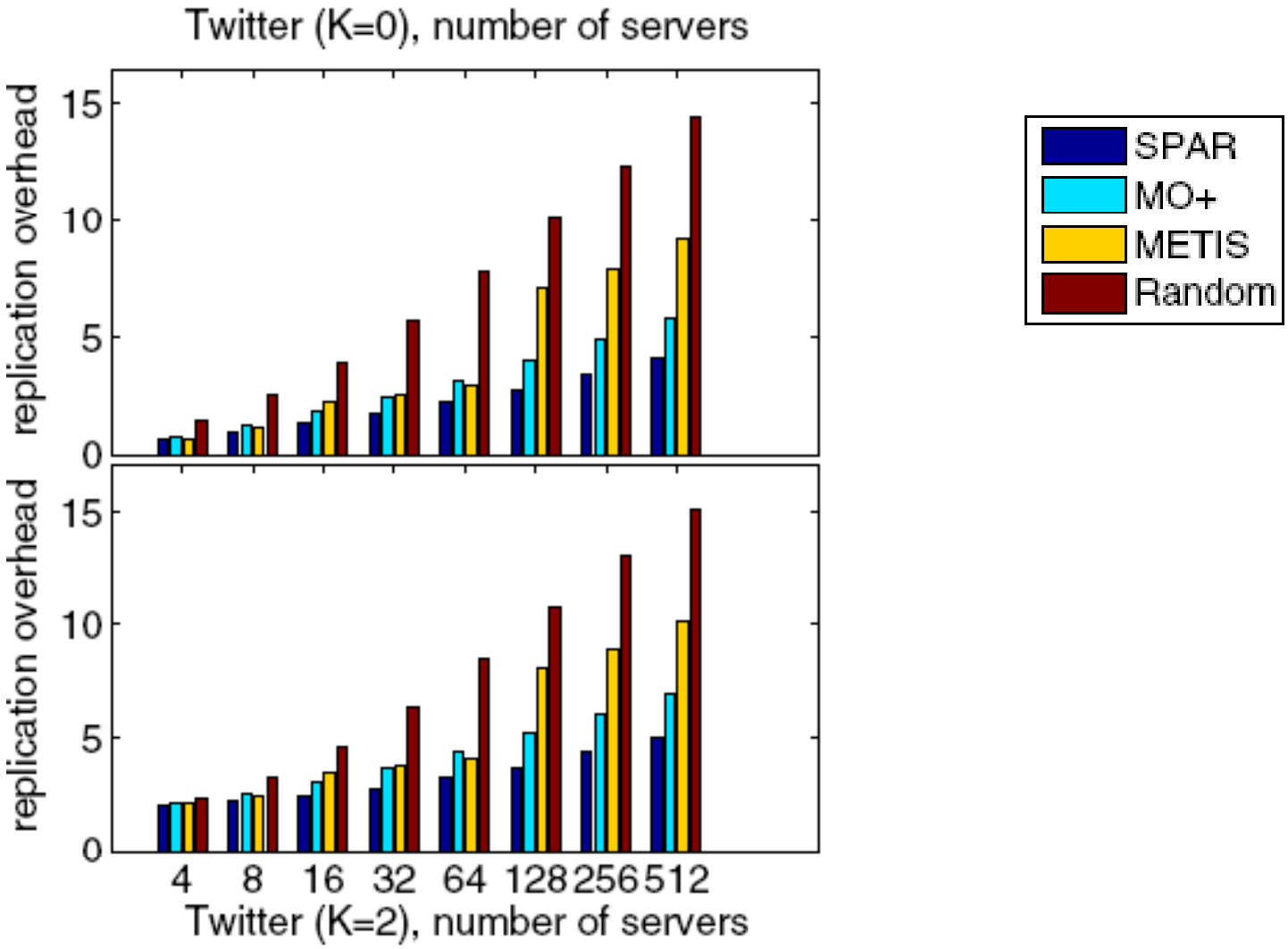
SPAR System Architecture

- System operations:
  - Data distribution
  - Data partitioning
  - Data movements
  - Data replication and consistency
  - Adding and removing servers
  - Handling failures

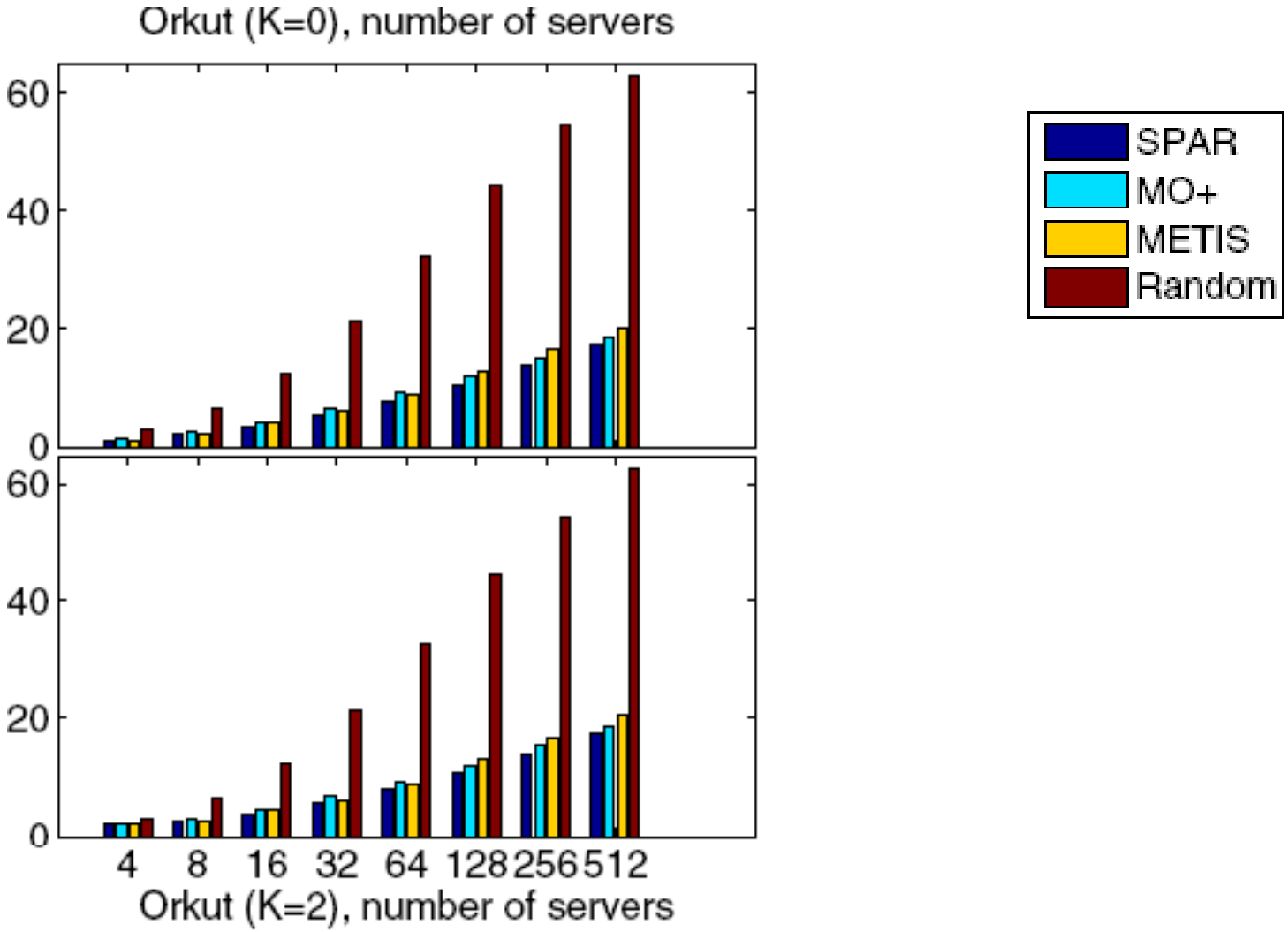
Evaluation

- Performance of SPAR
  - Datasets:
    - Twitter: 2, 408, 534 nodes and 48, 776, 888 edges
    - Facebook: a public dataset of the New Orleans Facebook network, containing 60,290 nodes and 1,545,686 edges
    - Orkut: 3, 072, 441 nodes and 223, 534, 301 edges

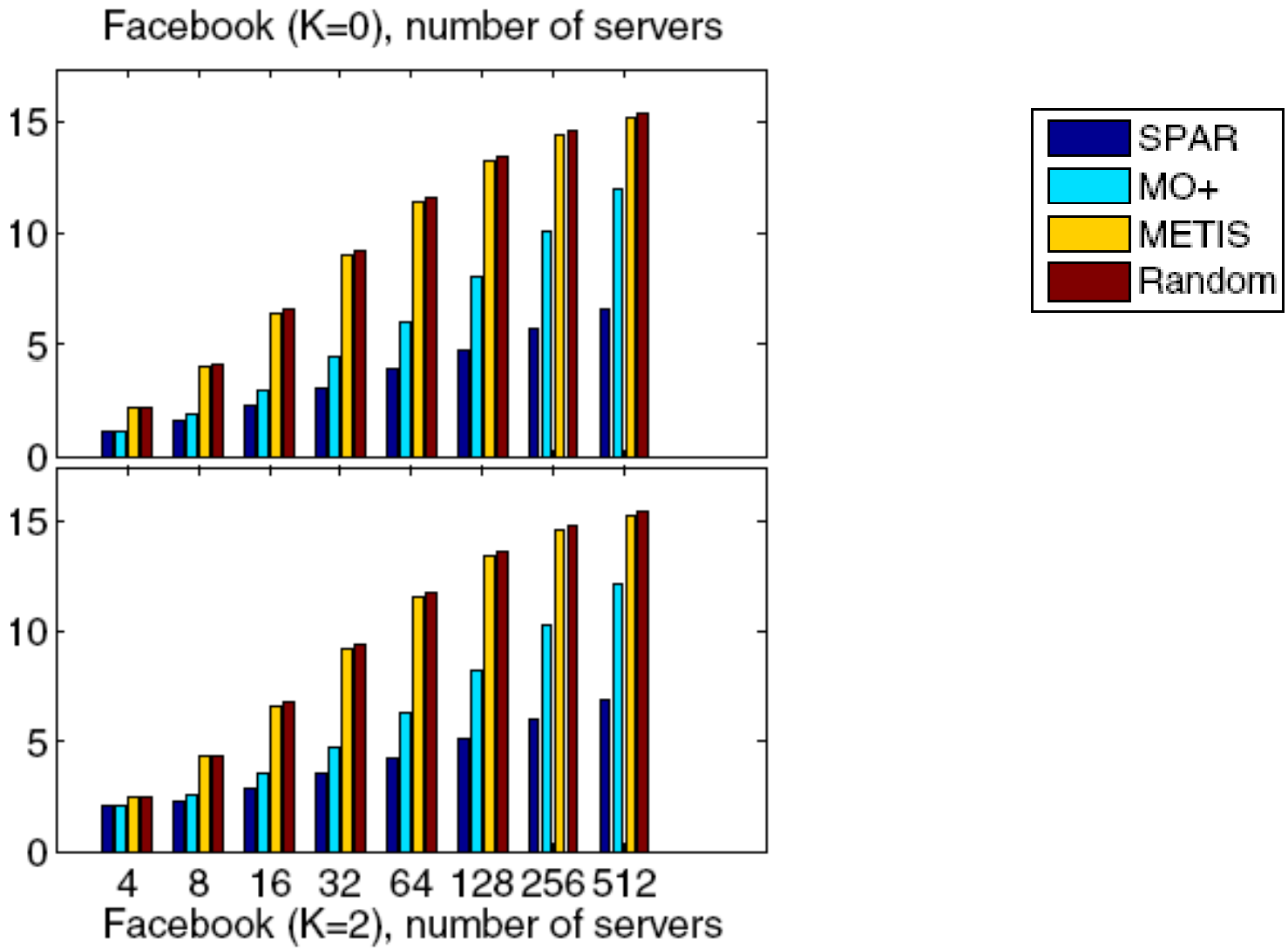
Evaluation


Twitter (K=0), number of servers
replication overhead
15
10
5
0
replication overhead
15
10
5
0
4
8
16
32
64
128
256
512
Twitter (K=2), number of servers
SPAR
MO+
METIS
Random

Evaluation


Orkut (K=0), number of servers
60
40
20
0
60
40
20
0
4
8
16
32
64
128
256
512
Orkut (K=2), number of servers
SPAR
MO+
METIS
Random

Evaluation


Facebook (K=0), number of servers
15
10
5
0
15
10
5
0
4
8
16
32
64
128
256
512
Facebook (K=2), number of servers
SPAR
MO+
METIS
Random

Evaluation

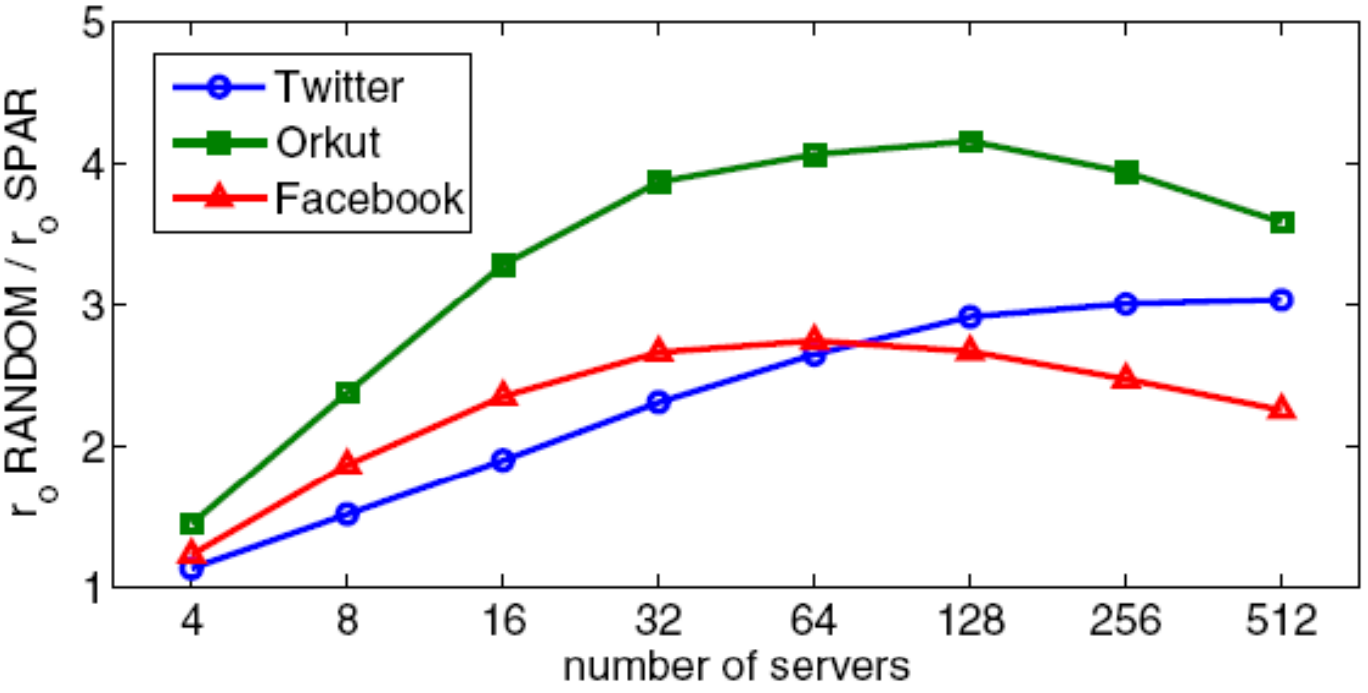


Figure 5: SPAR versus Random for $K = 2$

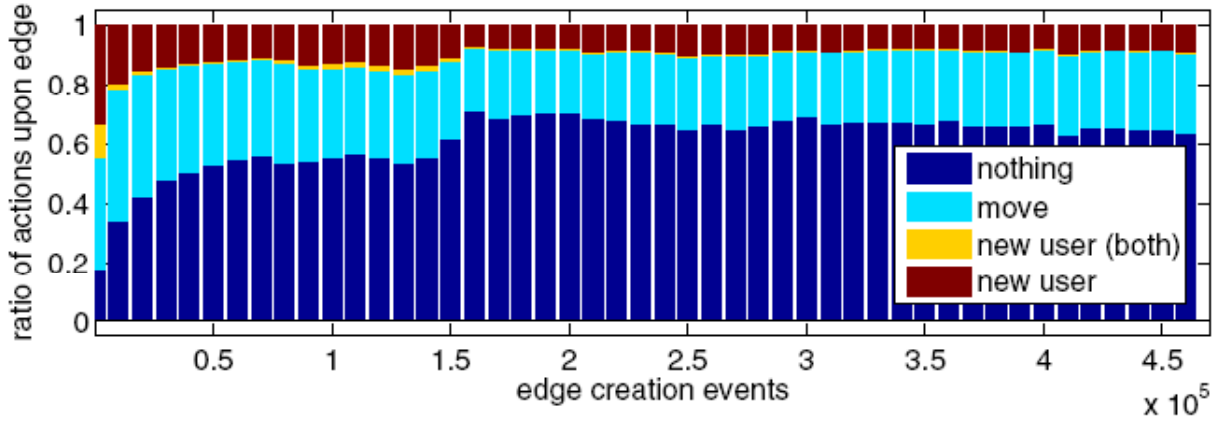


Figure 6: Ratio of actions performed by SPAR as edge creation events occur for Facebook

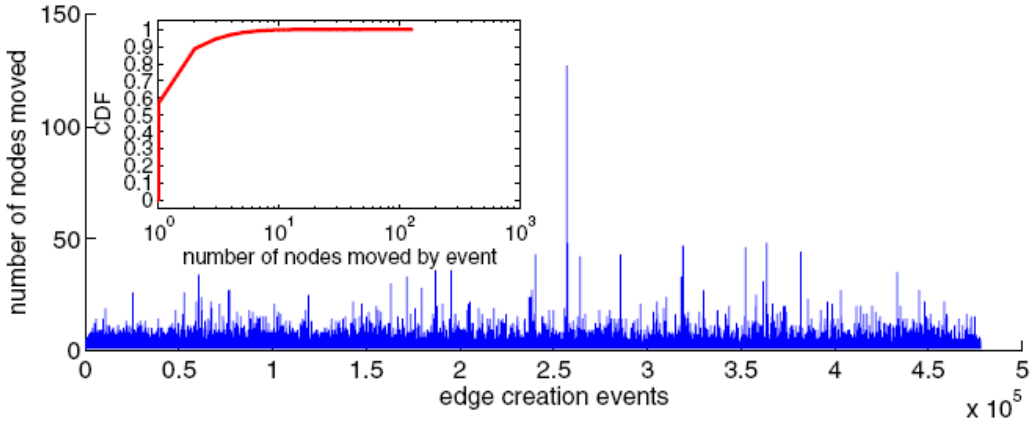


Figure 7: Number of movements per edge creation event for Facebook

Evaluation

- Evaluation on two data stores:
  - Testbed: 16 low-end commodity server
- Comparison between SPAR and random replication on Cassandra

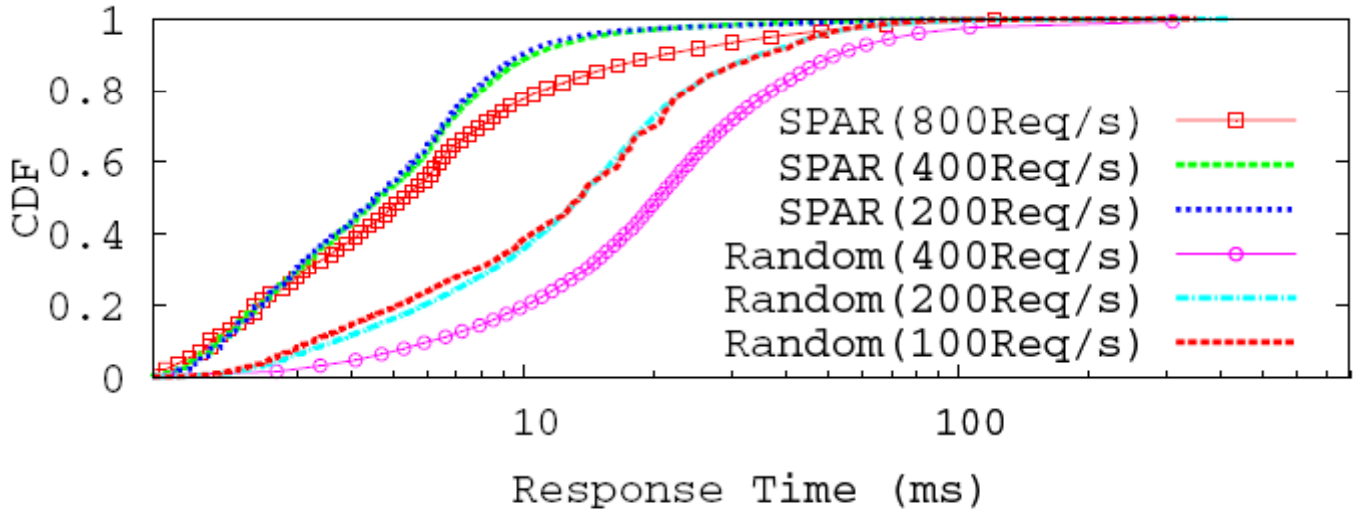


Figure 10: Response time of SPAR on top of Cassandra.

Evaluation

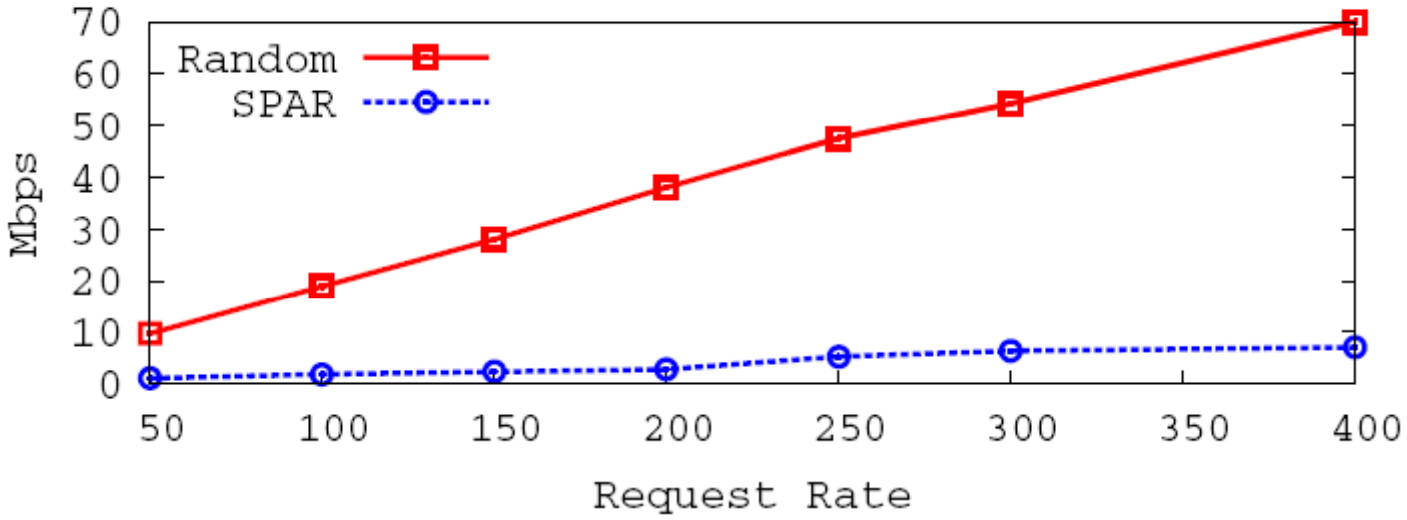


Figure 11: Network activity in SPAR on top of Cassandra.

Evaluation

- Comparison between SPAR and full replication on MySQL
  - Full replication: 95th percentile of the response time: 113ms for 16req/s, 151ms for 160 req/s, 245ms for 320 req/s
  - SPAR: 99th percentile of the response time: 150ms/ for 1500 req/s

Conclusion

- An efficient greedy optimization algorithm based on local information
- A large scale evaluation