LFGRAPH: SIMPLE AND FAST DISTRIBUTED GRAPH ANALYTICS

Presented by :
Chaitanya Datye

Hoque, Imranul, Vmware Inc. and Gupta, Indranil, University of Illinois at Urbana-Champaign – TRIOS '13

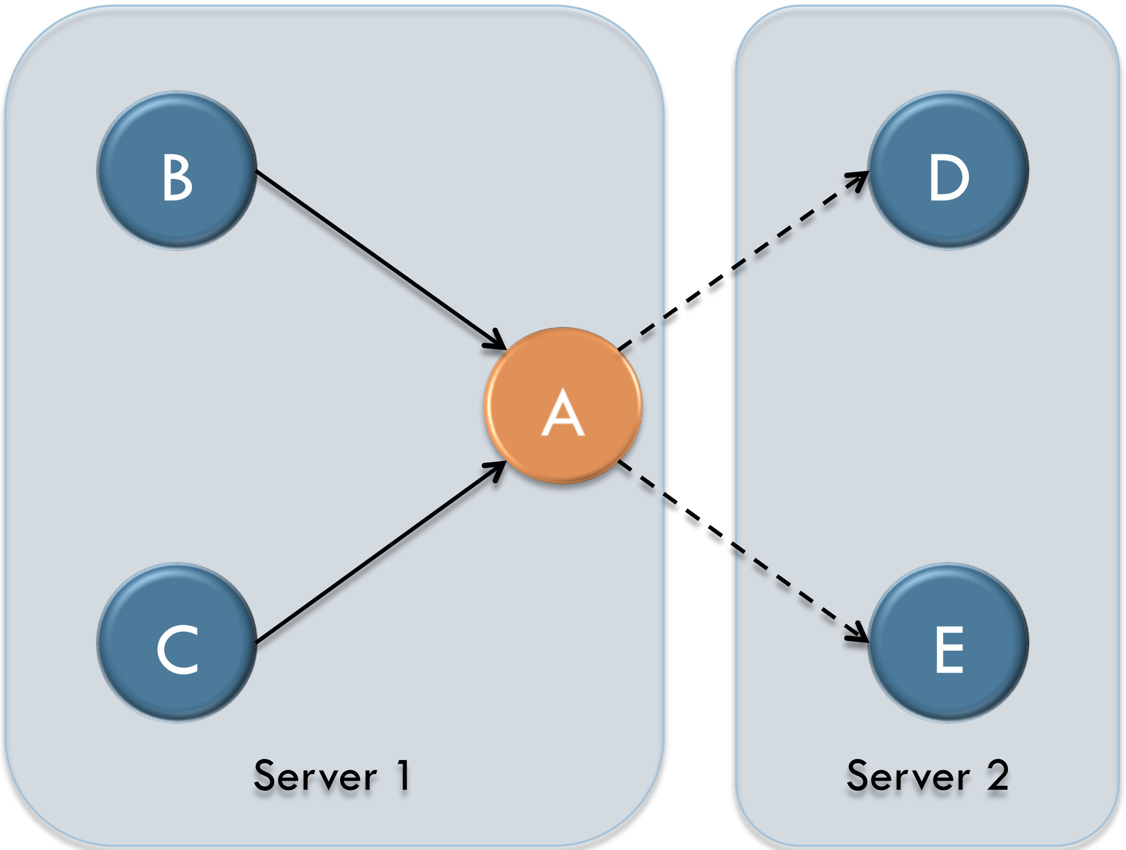
Why Distributed Graph Processing ??

- Graphs are everywhere!! – Social Networks, Finance, Stocks, Transportation Networks, Search engines, etc
- Well, These graphs are HUGE !!! – Millions and billions of vertices and edges

Distributed Graph Analytics Engine – Key Aspects

- Computations – **Low** and **Load balanced**
- Communications – **Low** and **Load balanced**
- **Low** Preprocessing Cost
- **Smaller** Memory Footprint
- System should be **Scalable**

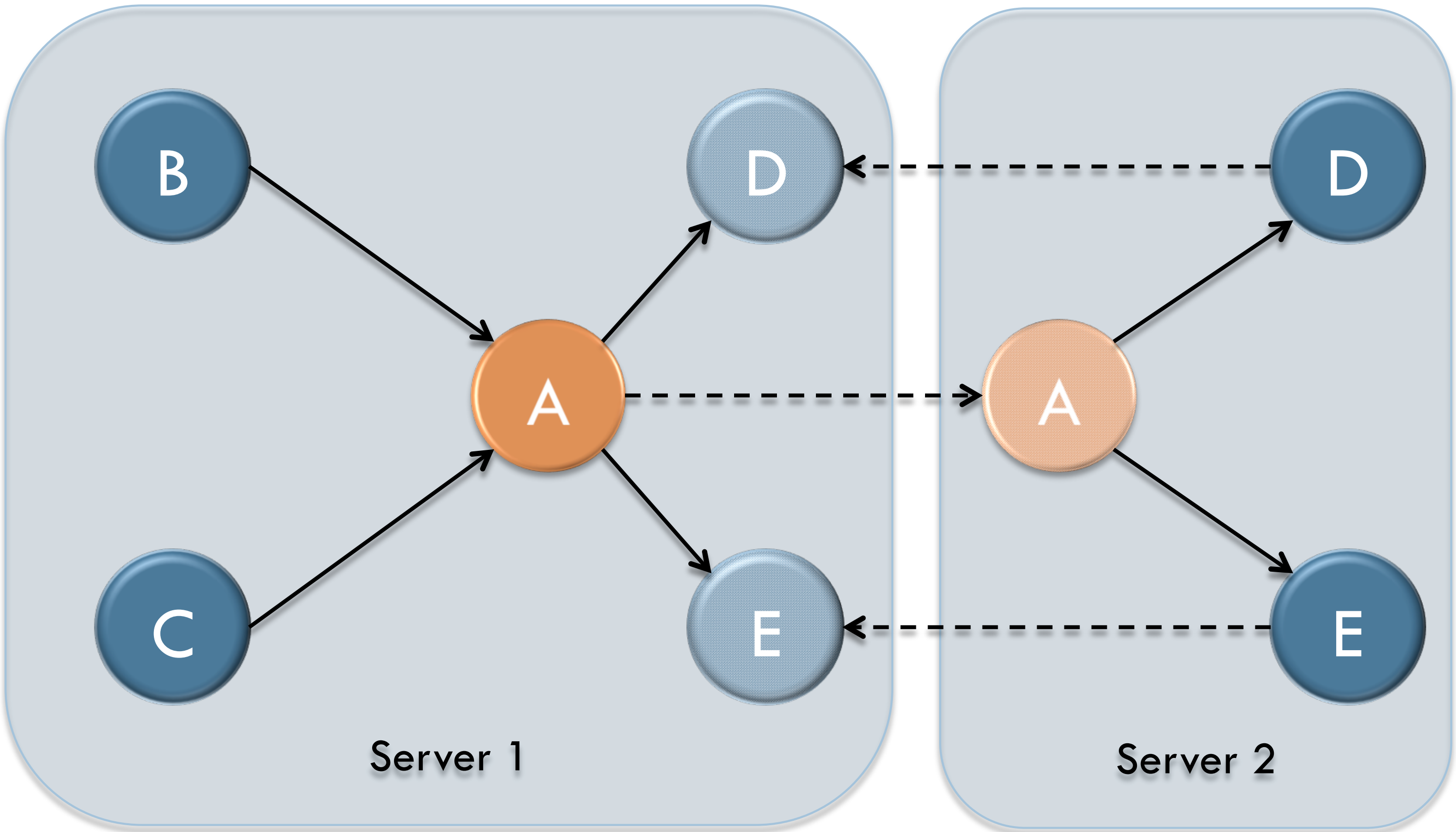
Pregel



Is there any better option?

Goal	Pregel
Computation	2 passes, Combiners
Communication	$\propto$ #Edge-cuts
Pre-Processing	Cheap(Hash)
Memory	High(store out-edges + buffered messages)
Scalability	Good but needs a min #servers

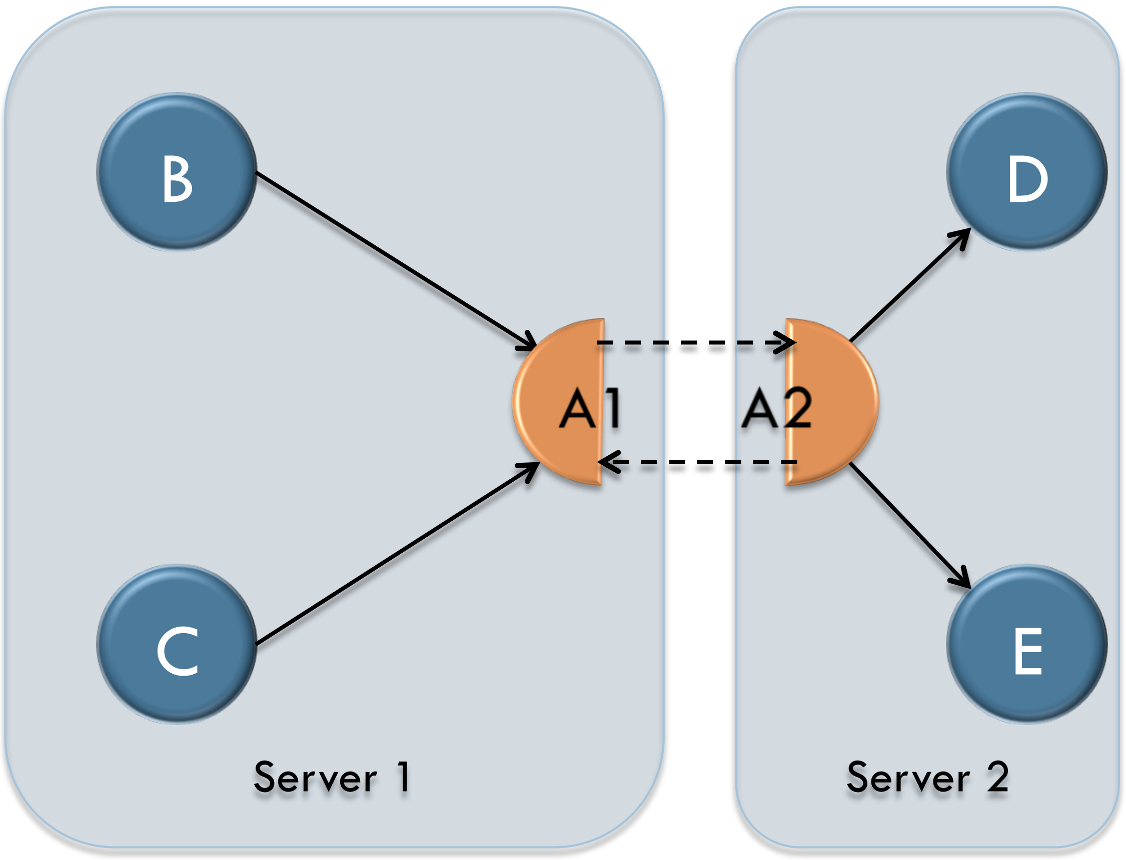
GraphLab



Can we do better ?

Goal	GraphLab
Computation	2 passes
Communication	$\propto$ #vertex ghosts
Pre-Processing	Cheap(Hash)
Memory	High(store in- and out-edges + ghost values)
Scalability	Good but needs a min #servers

PowerGraph



Can we still do better ?

Goal	PowerGraph
Computation	2 passes
Communication	∝ #vertex mirrors
Pre-Processing	Expensive (Intelligent)
Memory	High(store in- and out-edges + mirror values)
Scalability	Good but needs a min #servers

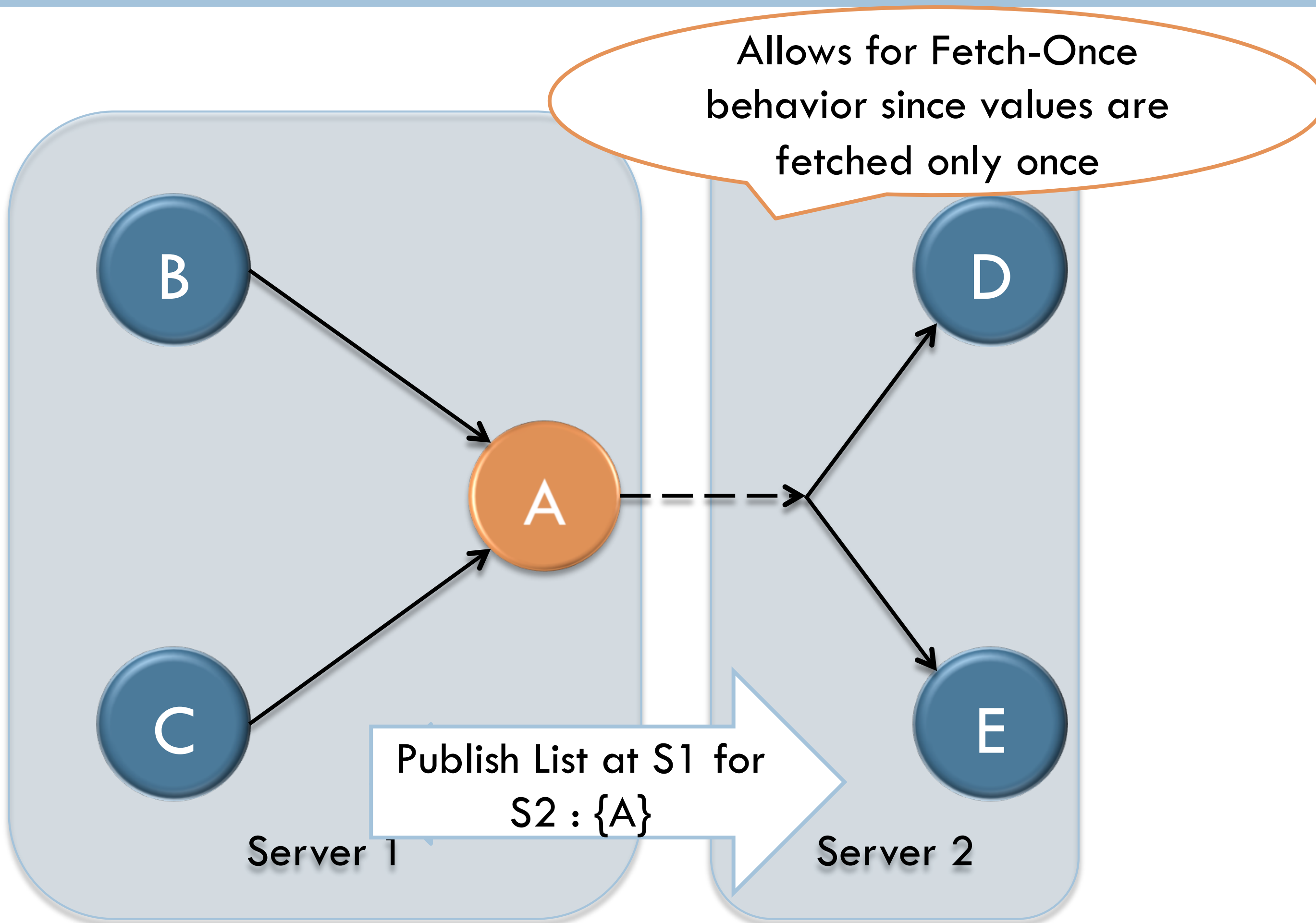
LFGraph – YES, We Can !!!

- Cheap Hash based partitioning
- Decoupling Computation and Communication
- Publish – Subscribe Mechanism
- Single – Pass Computations
- No Locking
- In – Neighbor Storage

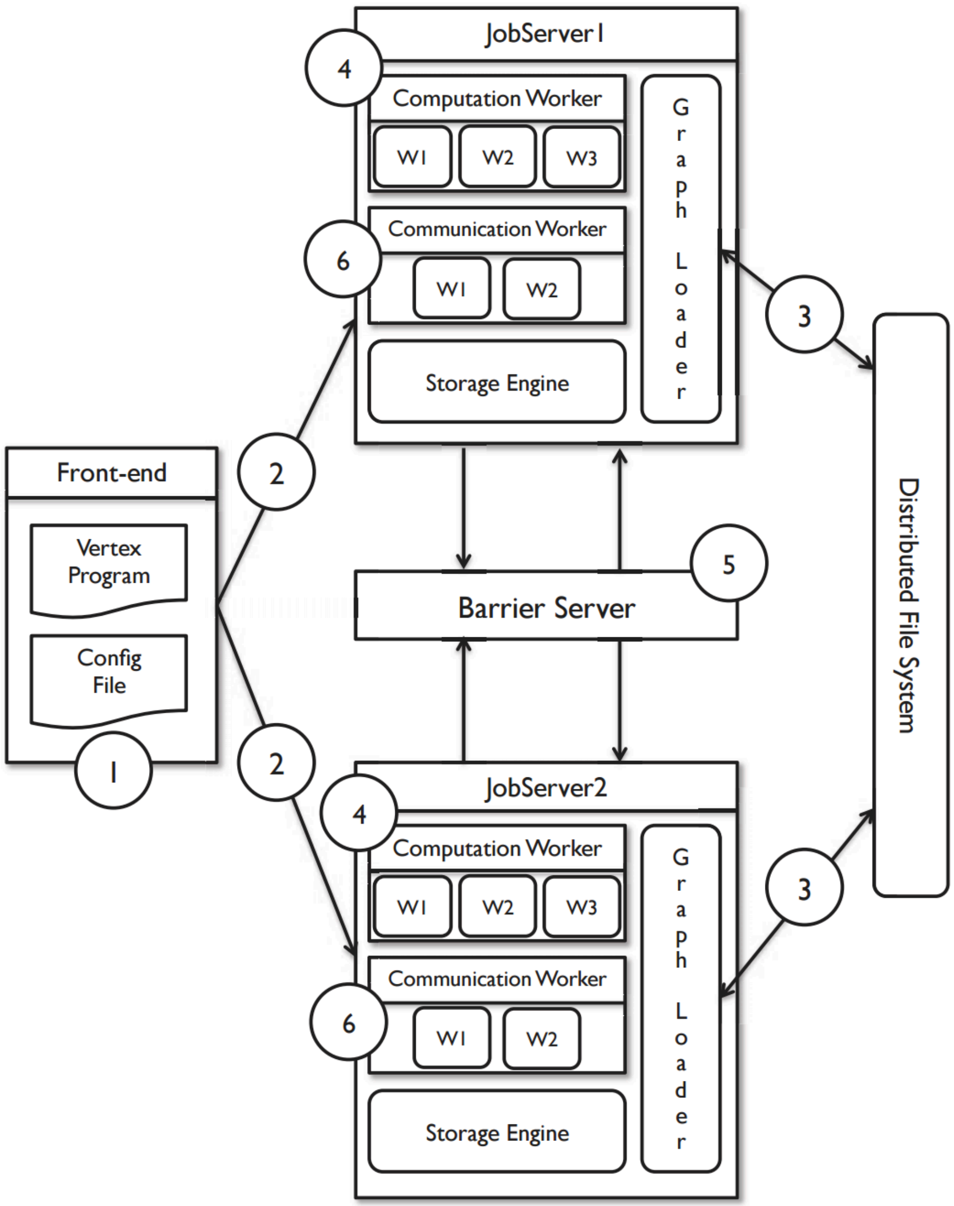
Publish Subscribe Mechanism

- Subscribe Lists
  - Created during preprocessing and are short lived
  - Per remote server
  - List contains vertices to be fetched from that server.
  - Garbage collected after preprocessing iteration
- Publish Lists
  - Created based on the Subscribe lists.
  - Each server maintains a Publish list for each remote server consisting of the vertices it needs to send to that server.

Publish Subscribe Mechanism



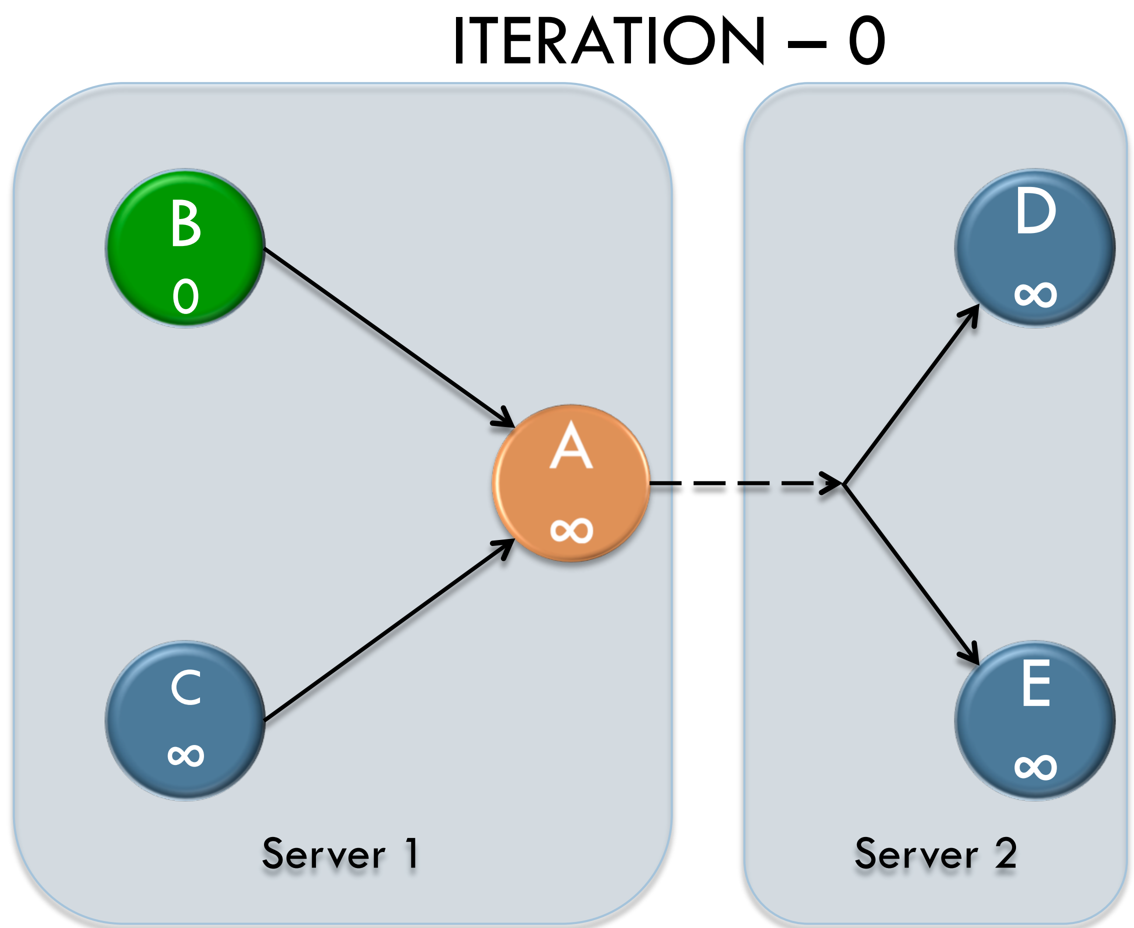
LFGraph System Design



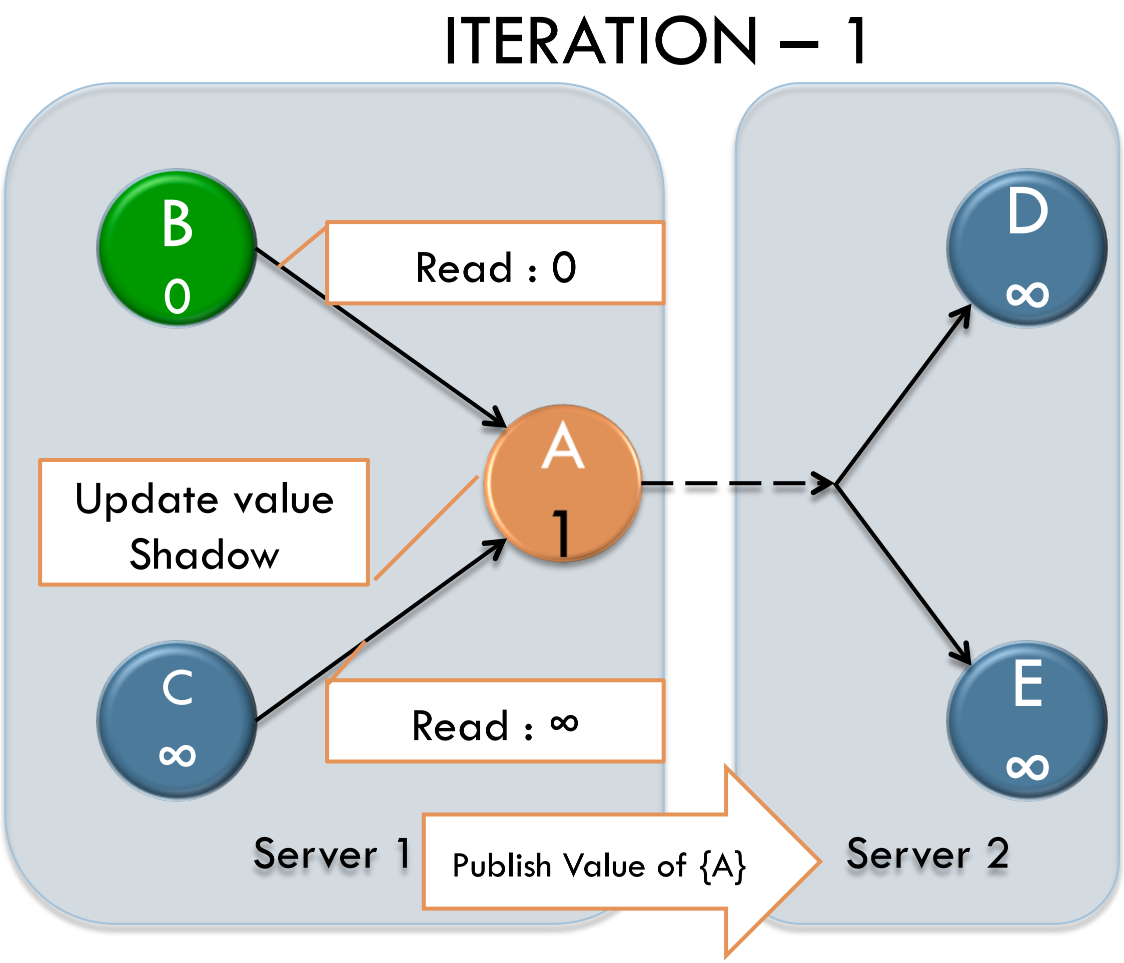
Local and Remote Value Stores

- Local Value Store
  - Real Version (Reads), Shadow Version (Writes)
  - Decoupled reads and writes – No Locking required
  - Shared across the computation workers in a Job Server
  - Flag set whenever shadow value written - used by communication workers to send values
- Remote Value Store
  - Stores values for each in-neighbor of a vertex at a Job Server.
  - Uses a flag – set only if updated value is received – Allows to skip vertices which aren't updated in that iteration.

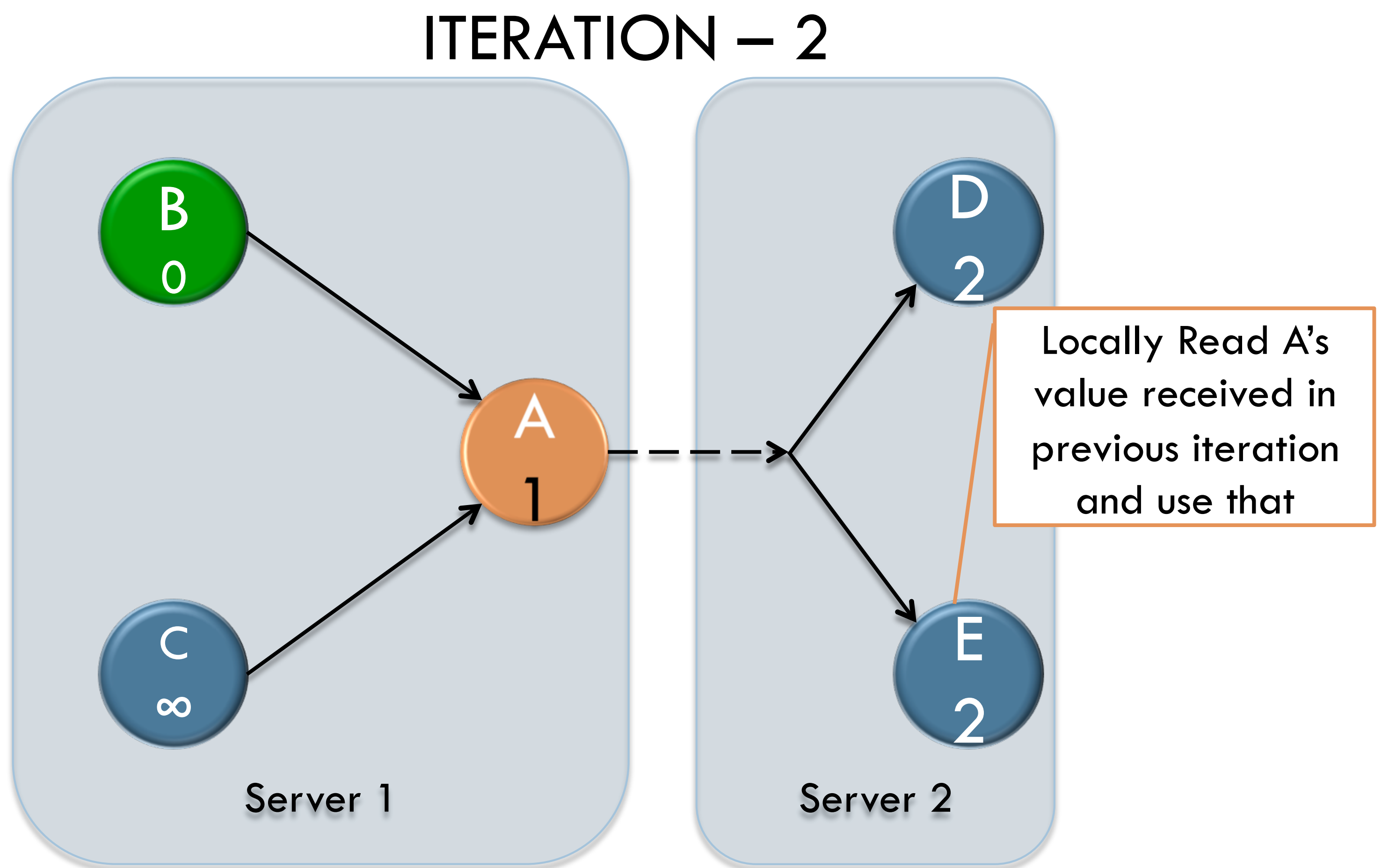
Example : SSSP using LFGraph



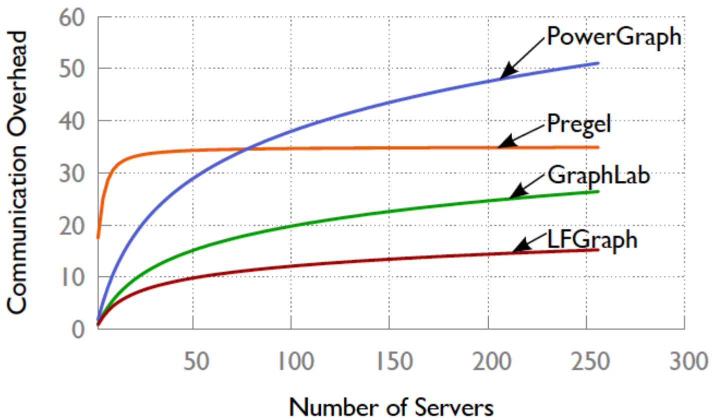
Example : SSSP using LFGraph



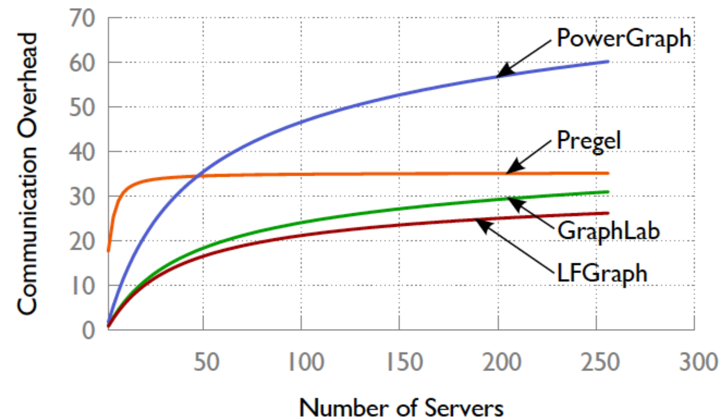
Example : SSSP using LFGraph



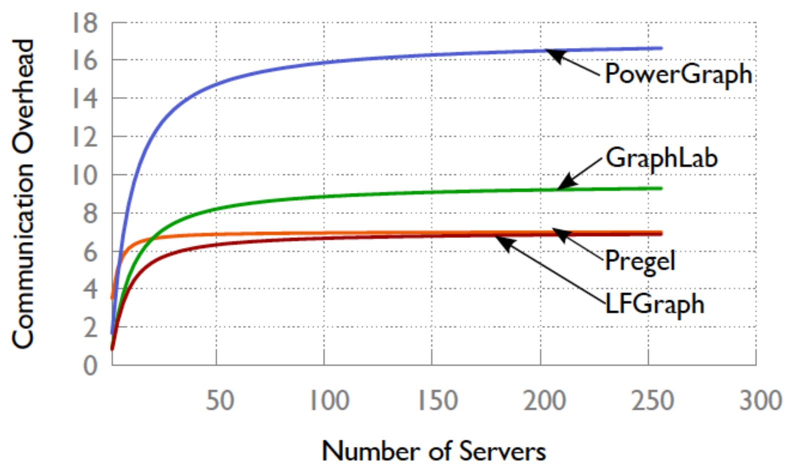
Communication Overhead analysis



(a) Twitter

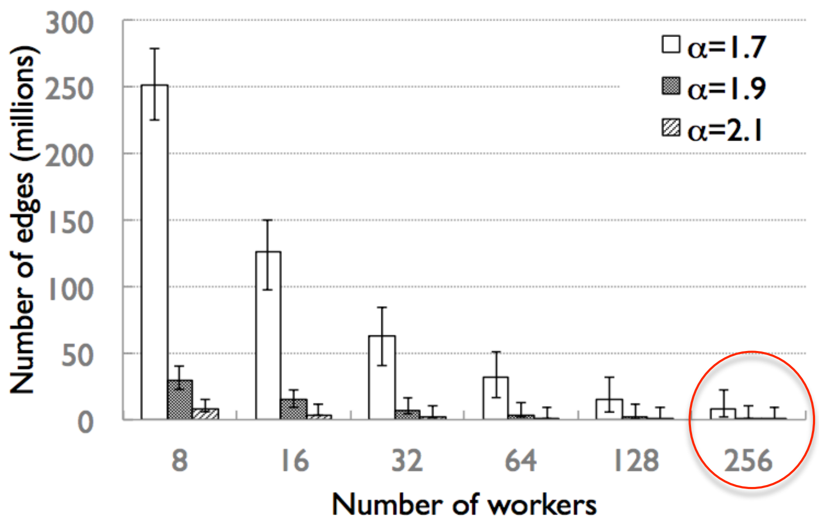


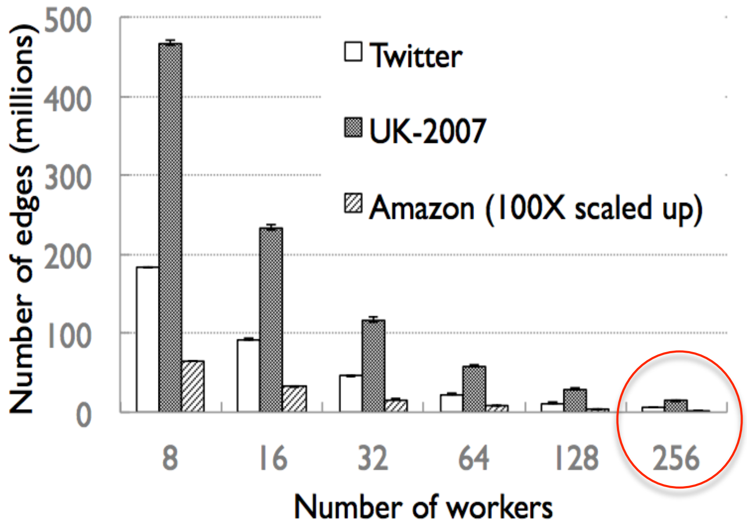
(b) UK-2007 Web Graph

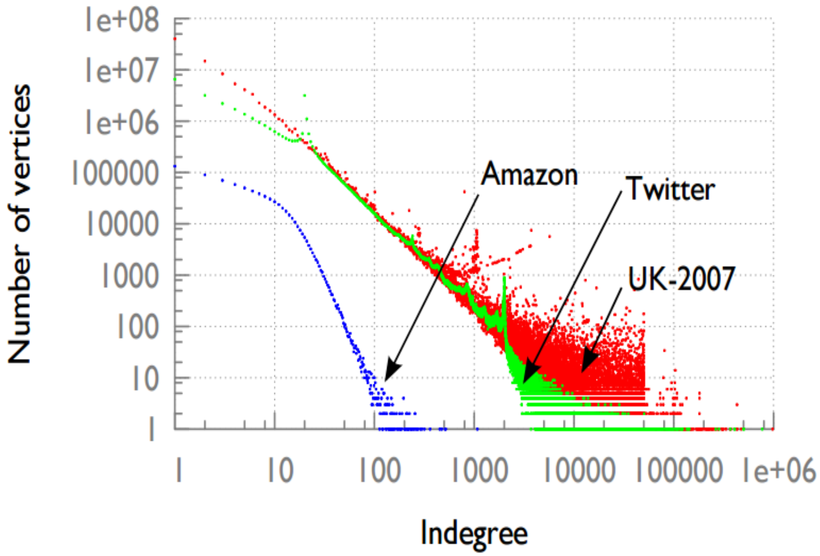


(c) Amazon Recommendation Graph

Computation Balance analysis – Real World vs Ideal Power Law graphs



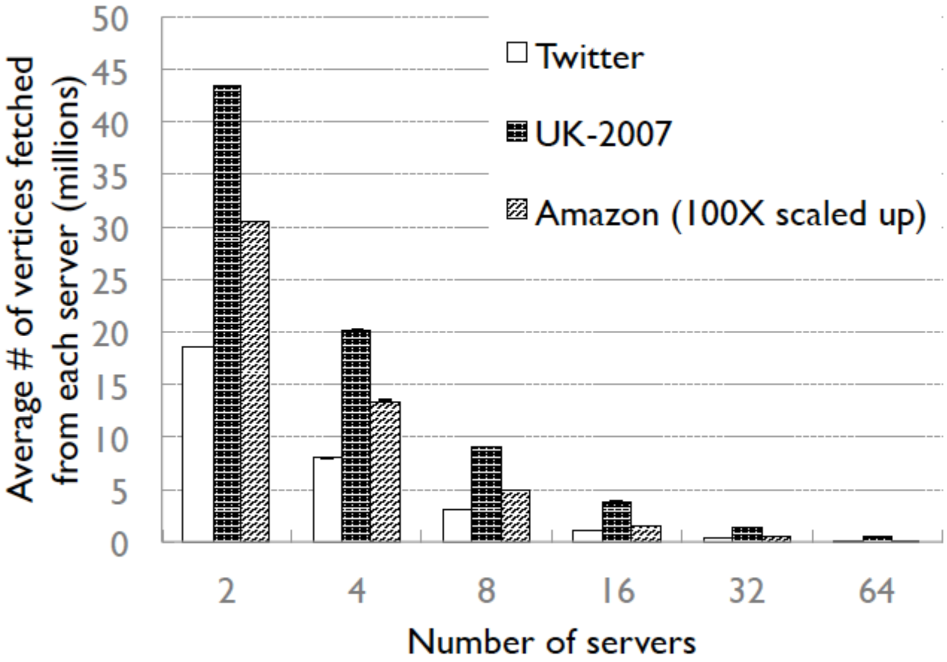




- Cheap partitioning strategy suffices for real world graphs

Communication Balance analysis

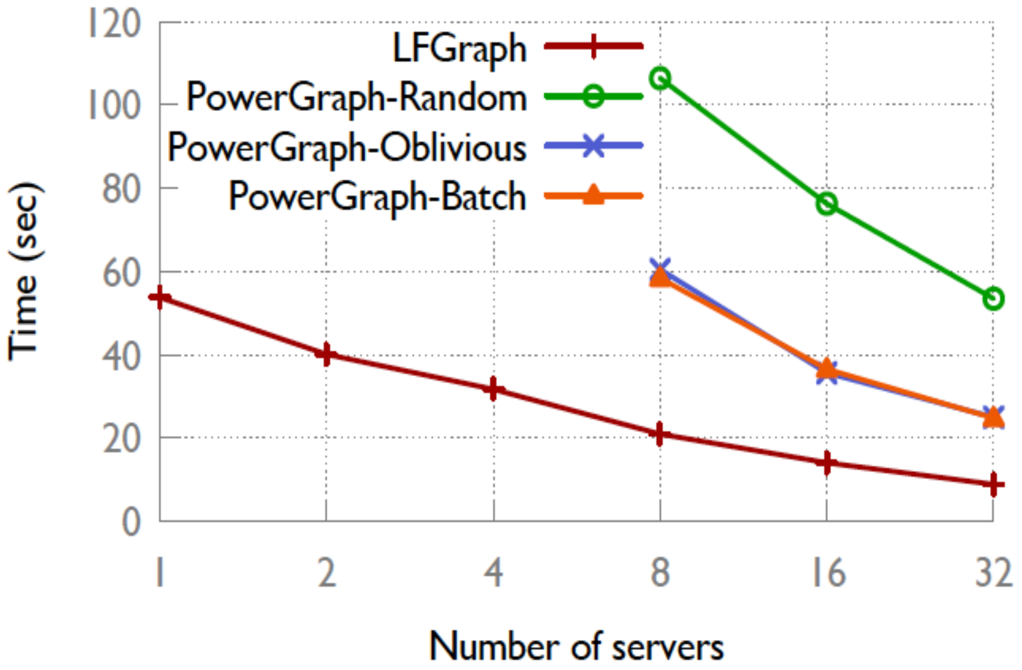
- Communication imbalance → more processing time
- If data sent by server S1 is more than that of S2, overall transfer time increases
- LFGraph balances communication load very well since error bars are small



PageRank runtime ignoring partition time

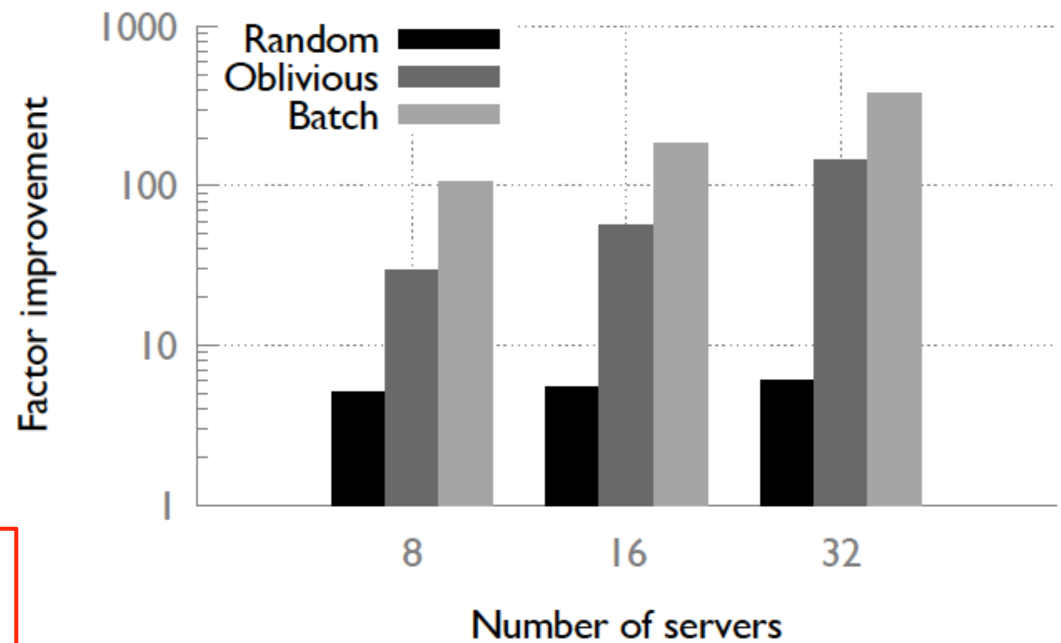
- PowerGraph couldn't load graph at small cluster sizes

- LFGraph wins over the best PowerGraph version by a factor of 2x



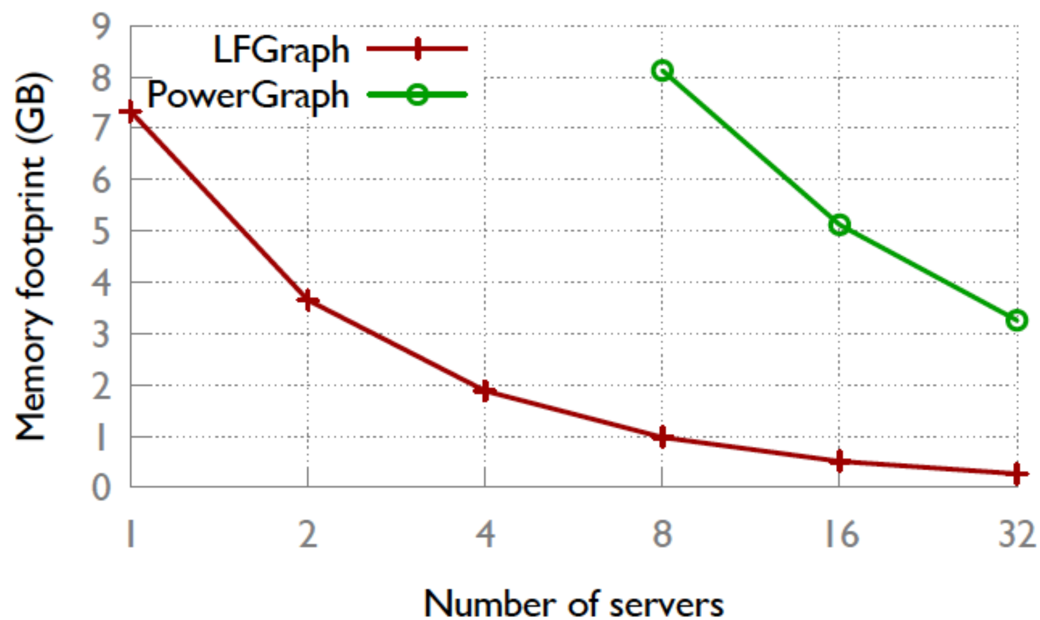
PageRank runtime including partition time

- Improvement is most over the intelligent partitioning schemes of PowerGraph
- 8 servers – 4x to 100x improvement, 32 servers – 5x to 380x improvement
- Intelligent partitioning strategies have little effect



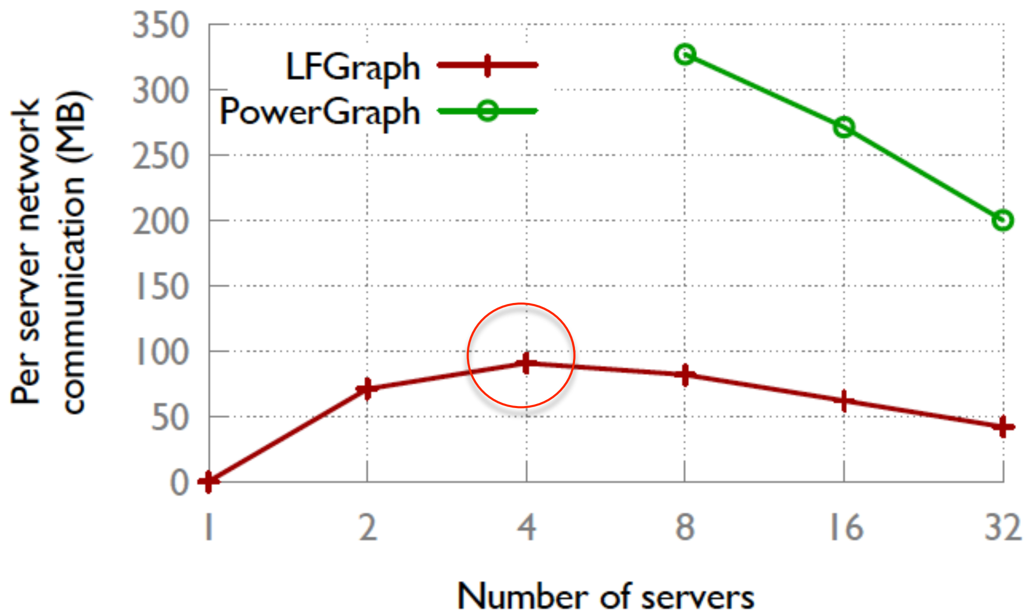
Memory Footprint – LFGraph vs PowerGraph

- LFGraph stores only in-links and publish lists unlike PowerGraph.
- Memory footprint is 8x to 12x lesser than PowerGraph



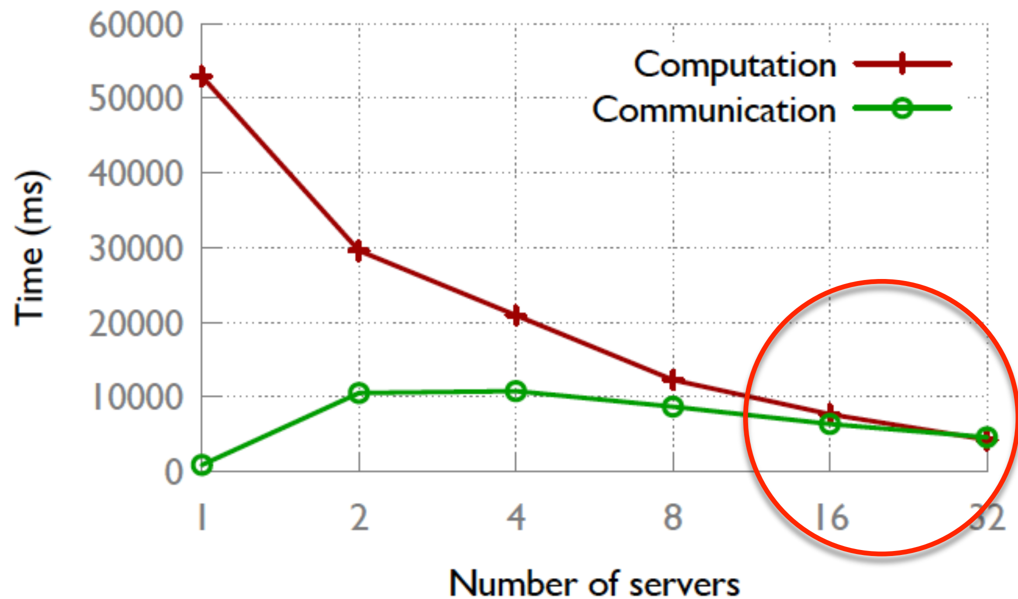
Network Communication – LFGraph vs PowerGraph

- There is first a quick rise in the total communication overhead
- But, as the total communication overhead plateaus out, the cluster size increase takes over dropping the per server overhead
- LFGraph transfers about 4x less data per server than PowerGraph



Computation vs Communication

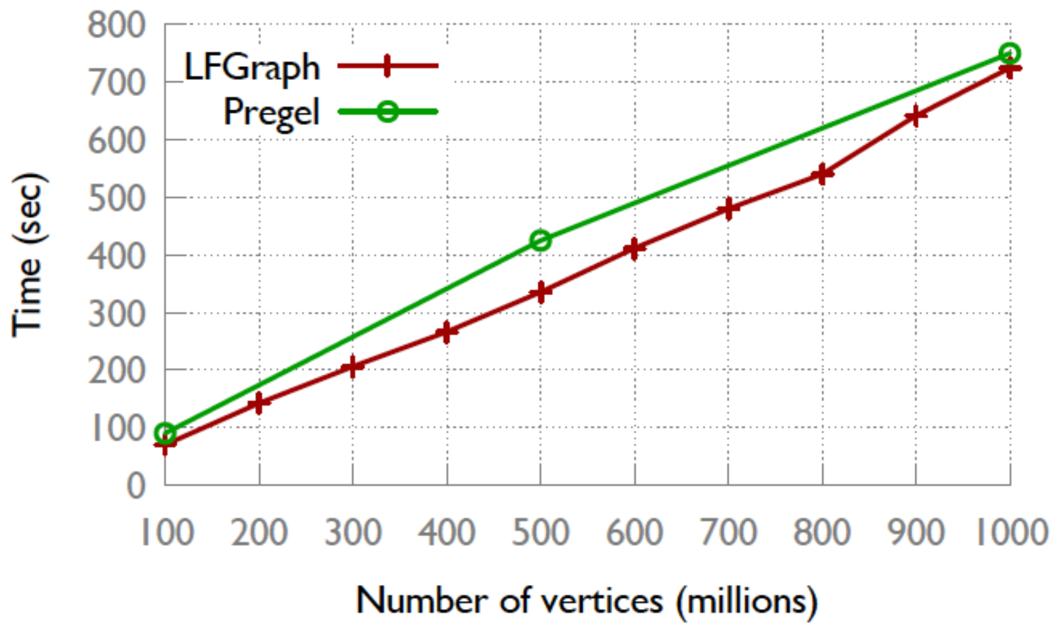
- Computation time decreases with increasing number of servers
- Communication time curve mirrors the per-server network overhead
- Compute dominates communicate in small clusters
- After 16 servers, LFGraph achieves a balance



Scaling to Larger Graphs

- Pregel – 300 servers, 800 workers
- LFGraph – 12 servers, 96 workers
- Runs SSSP benchmark

- Uses 10x less compute power still gives better performance. LFGraph scales well



Pros

- Low computation and communication overheads ☺
- Low memory footprint ☺
- Highly Scalable ☺
- Computations and Communications are balanced ☺
- Cheap partitioning strategy suffices ☺

Cons/Comments/Discussion

- In case of failures, LFGraph restarts computation. More efficient mechanisms for fault tolerance?
- Barrier Server – SPOF!!
- LFGraph requires that sufficient memory is available in the cluster to store the graph and the associated values. What if graph size is large? Or such a cluster is unavailable?
- No techniques to give out partial results in case of LFGraph. Every computation runs to completion. What if there is a deadline?

Questions ?