PowerGraph

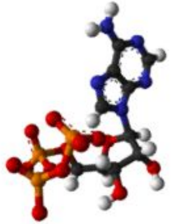
Distributed Graph-Parallel Computation on Natural Graphs

Lingyun (Luke) Li

Large-scale graph-structured computation

- Central to tasks ranging from targeted advertising to natural language processing
- Billions of vertices, edges and extremely rich data

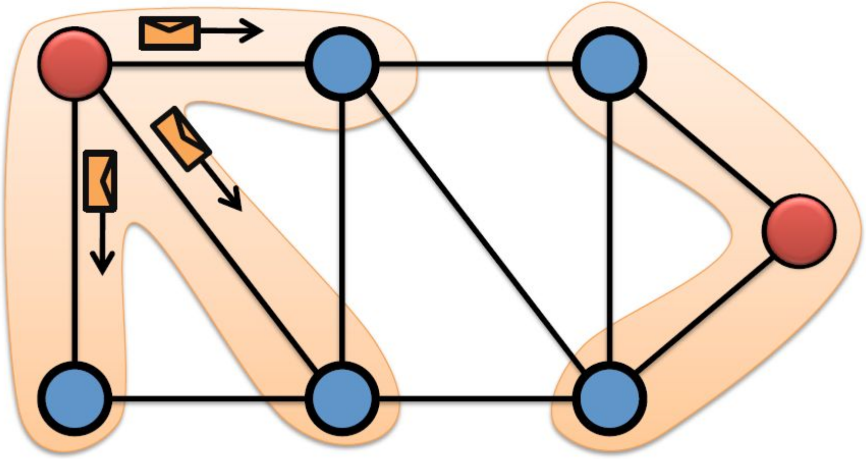
Social Media | **Science** | **Advertising** | **Web**



NETFLIX

Existing Graph-Parallel Abstractions

- A sparse graph G = {V,E}
- A vertex-program Q which is executed in parallel on each vertex v ∈ V
- Q(v) interact with neighboring instances Q(u) where (u, v) ∈ E
- Communication through shared-state in GraphLab, or messages in Pregel

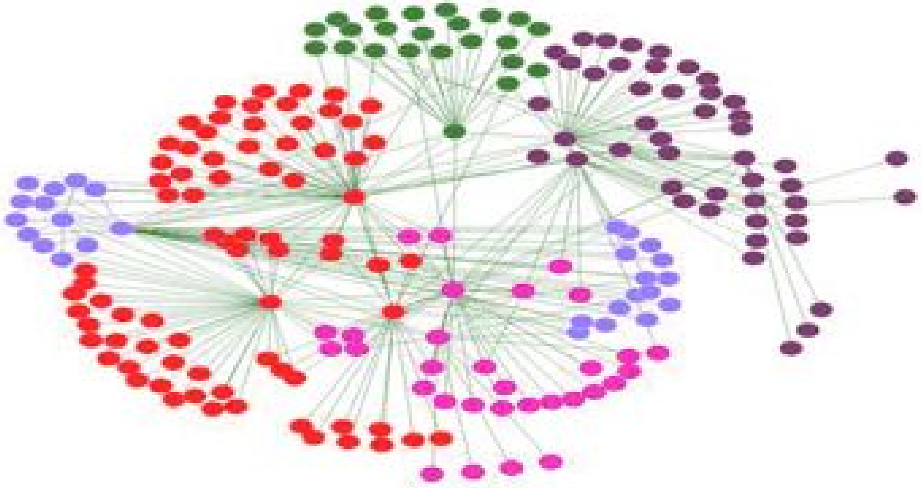


Existing Graph-Parallel Abstractions

- GAS: three conceptual phases of a vertex-program: Gather, Apply, and Scatter
- Constrain the interaction of vertex program to a graph structure to enable the optimization of data-layout and communication
- Rely on each vertex having a small neighborhood to maximize parallelism and effective partitioning to minimize communication

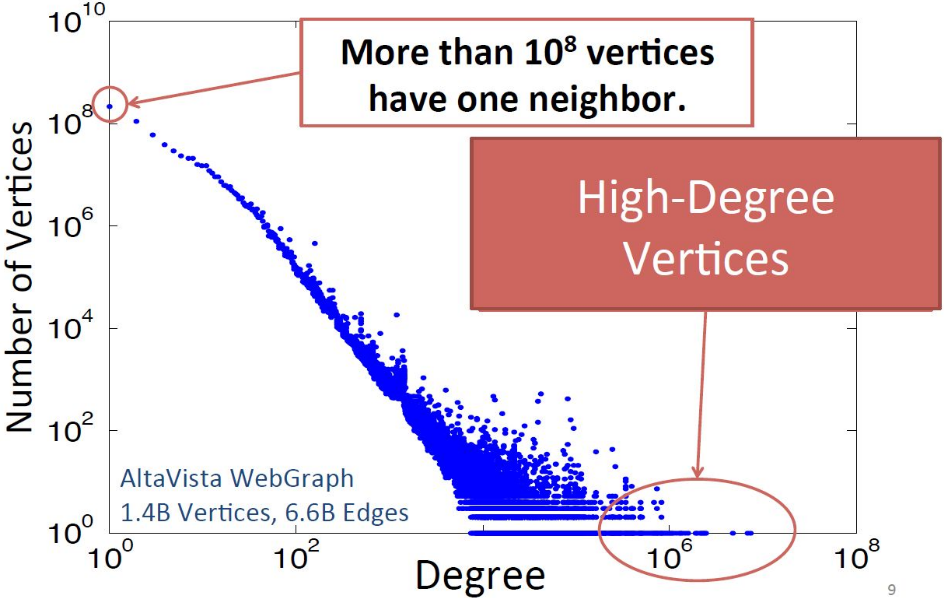
Natural Graphs

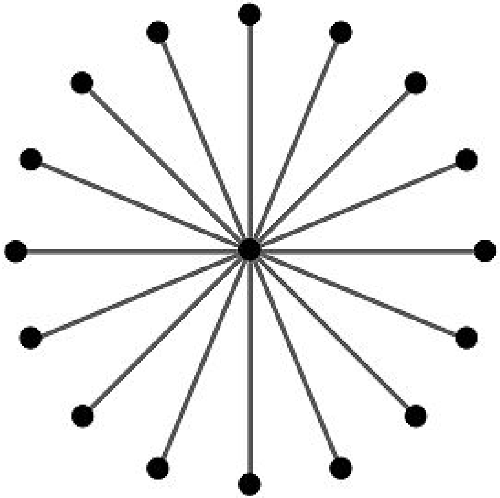
- Commonly found in the real-world
- Highly skewed power-law degree distributions
- Poor performance on existing distributed graph computation systems



Skewed Power-Law Degree Distribution

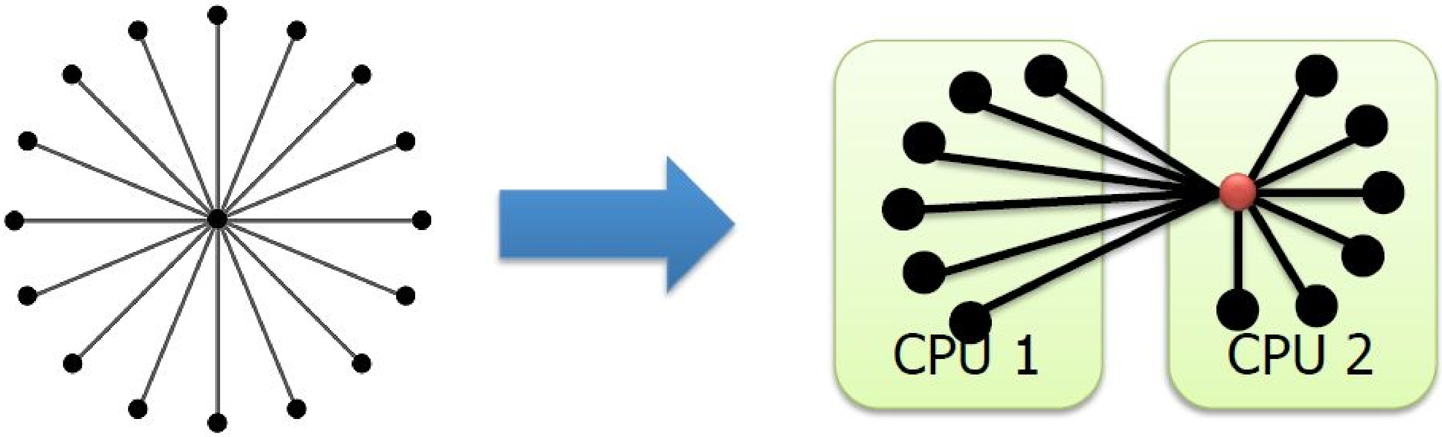
- Most vertices have relatively few neighbors while a few have many neighbors
- Star like graph





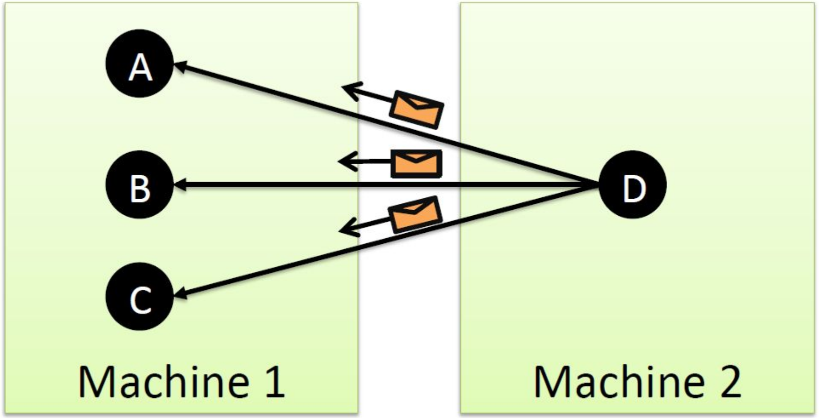
Challenges of Natural Graphs

- Partitioning
  - GraphLab and Pregel depend on graph partitioning to minimize communication and ensure work balance.
  - Performs poorly on power-law graphs



Challenges of Natural Graphs

- Work imbalance
- Communication
  - Communication asymmetry
  - Generate and send many identical messages



Challenges of Natural Graphs

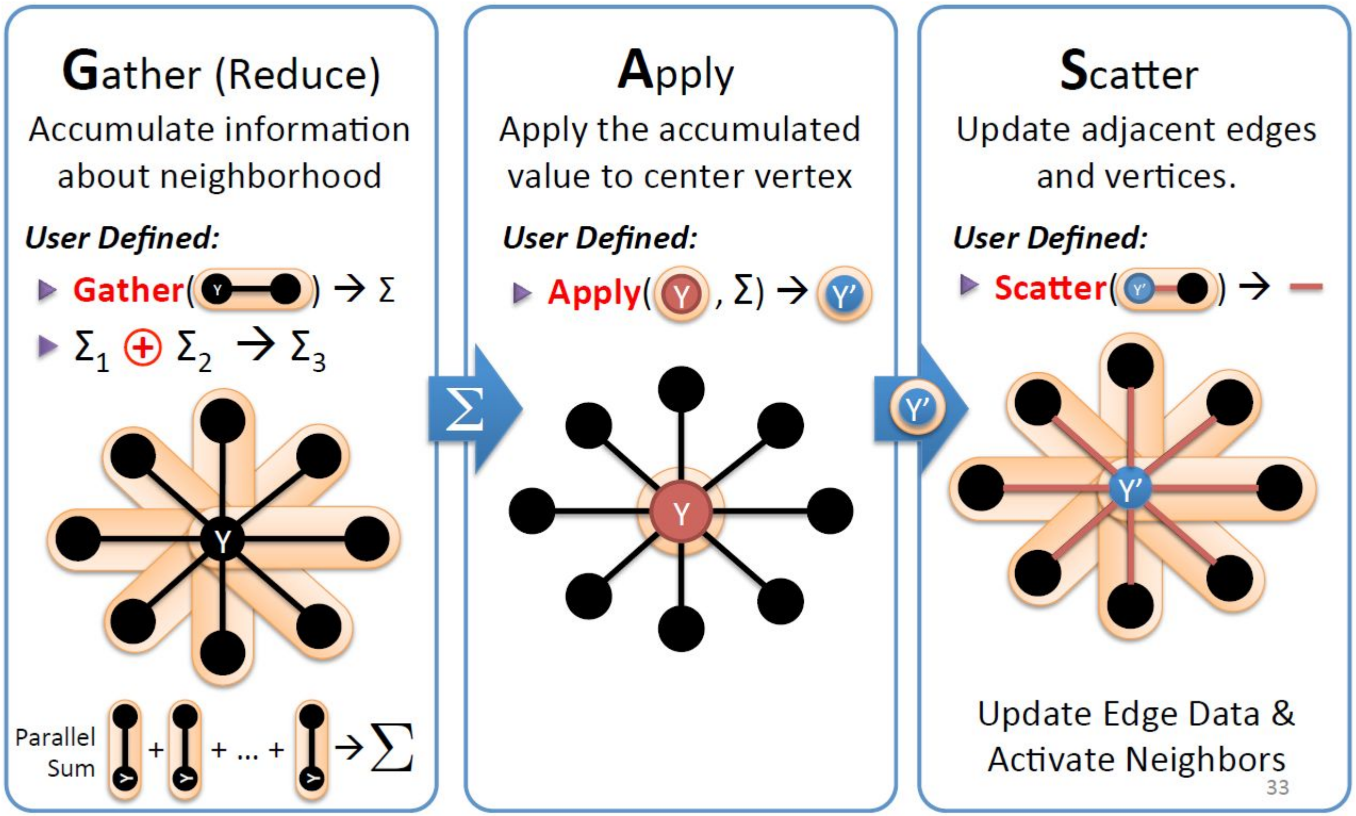
- Storage
  - Locally store the adjacency information for each vertex
  - Storage linear in degree of vertex

- Computation
  - No parallelism within individual vertex-programs
  - limiting scalability on high-degree vertices

PowerGraph

- GAS decomposition to distribute a single vertex-program over multiple machines
- Vertex partitioning: effectively distribute large power--law graphs
- Eliminates the degree dependence of the vertex-program

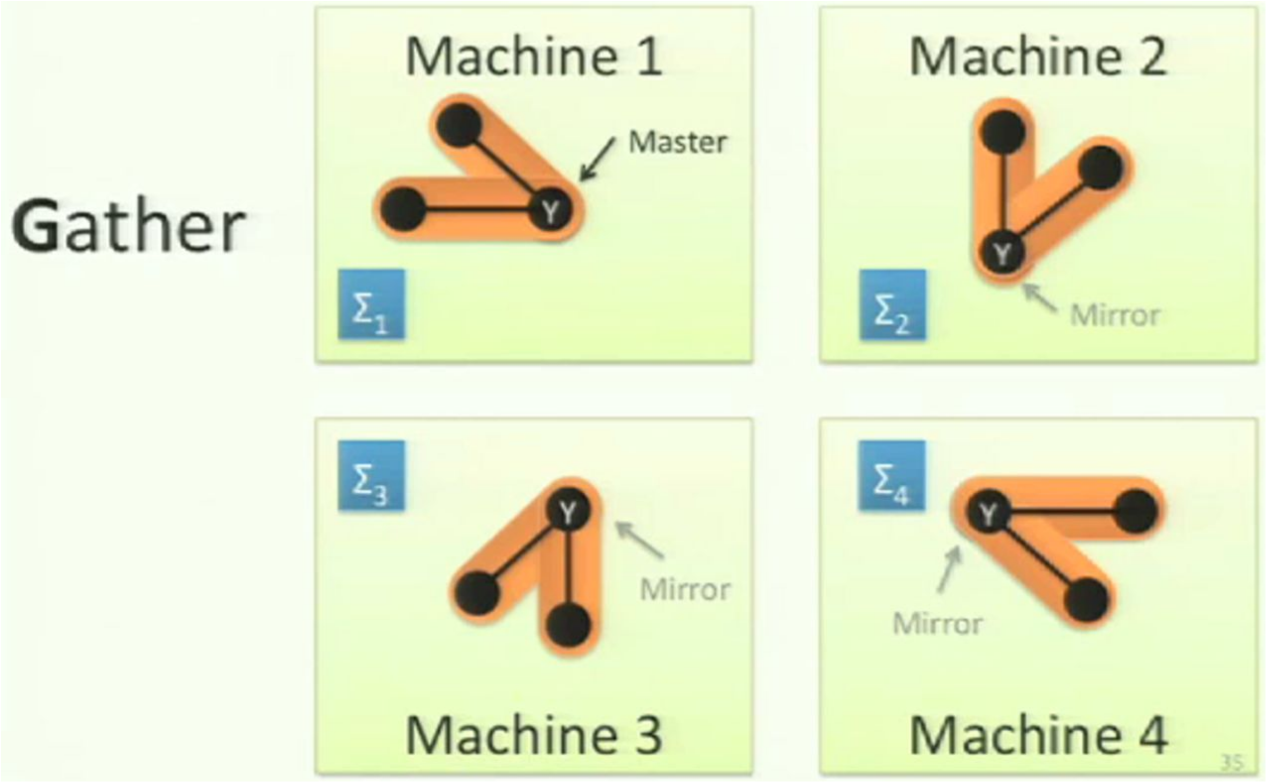
GAS Decomposition



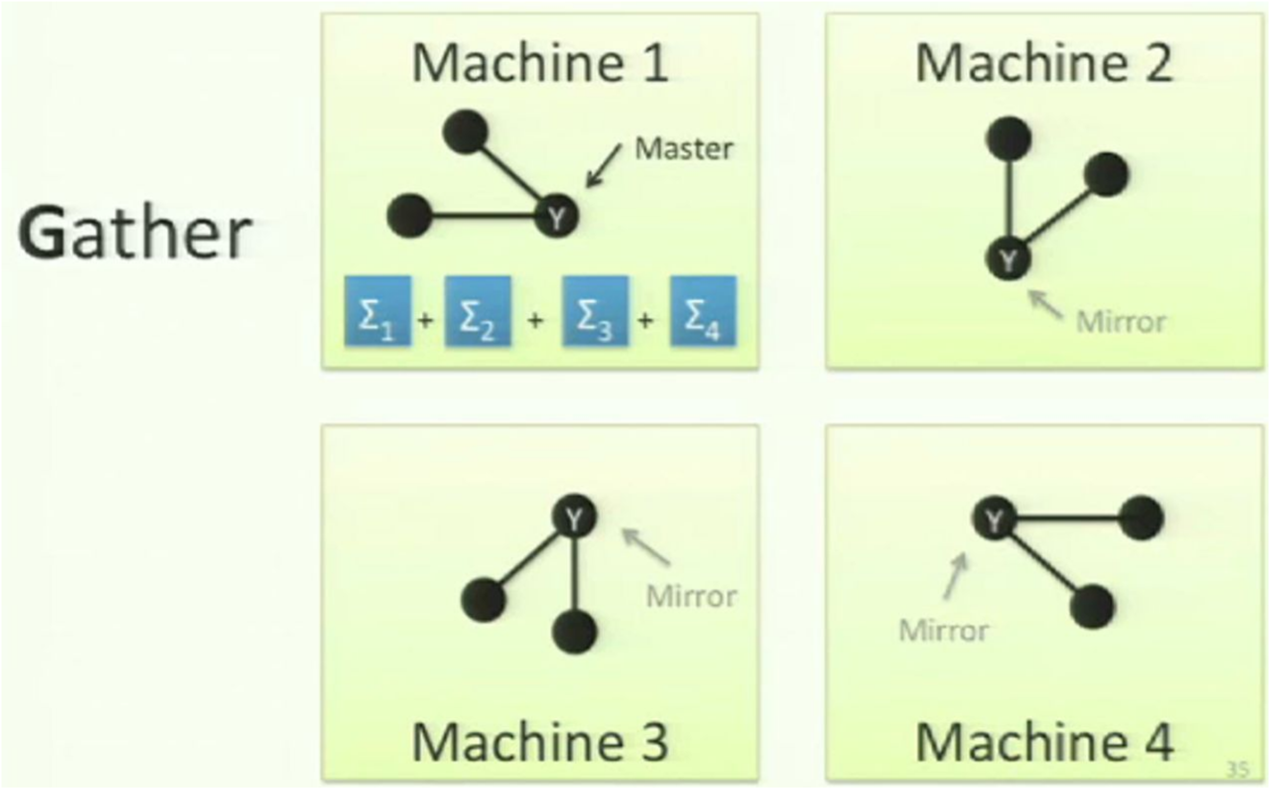
PageRank

```
gather(D_u, D_(u,v), D_v):
   return D_v.rank / #outNbrs(v)
sum(a, b): return a + b
apply(D_u, acc):
   rnew = 0.15 + 0.85 * acc
   D_u.delta = (rnew - D_u.rank)/
               #outNbrs(u)
   D_u.rank = rnew
// scatter_nbrs: OUT_NBRS
scatter(D_u,D_(u,v),D_v):
   if(|D_u.delta|>ε) Activate(v)
   return delta
```

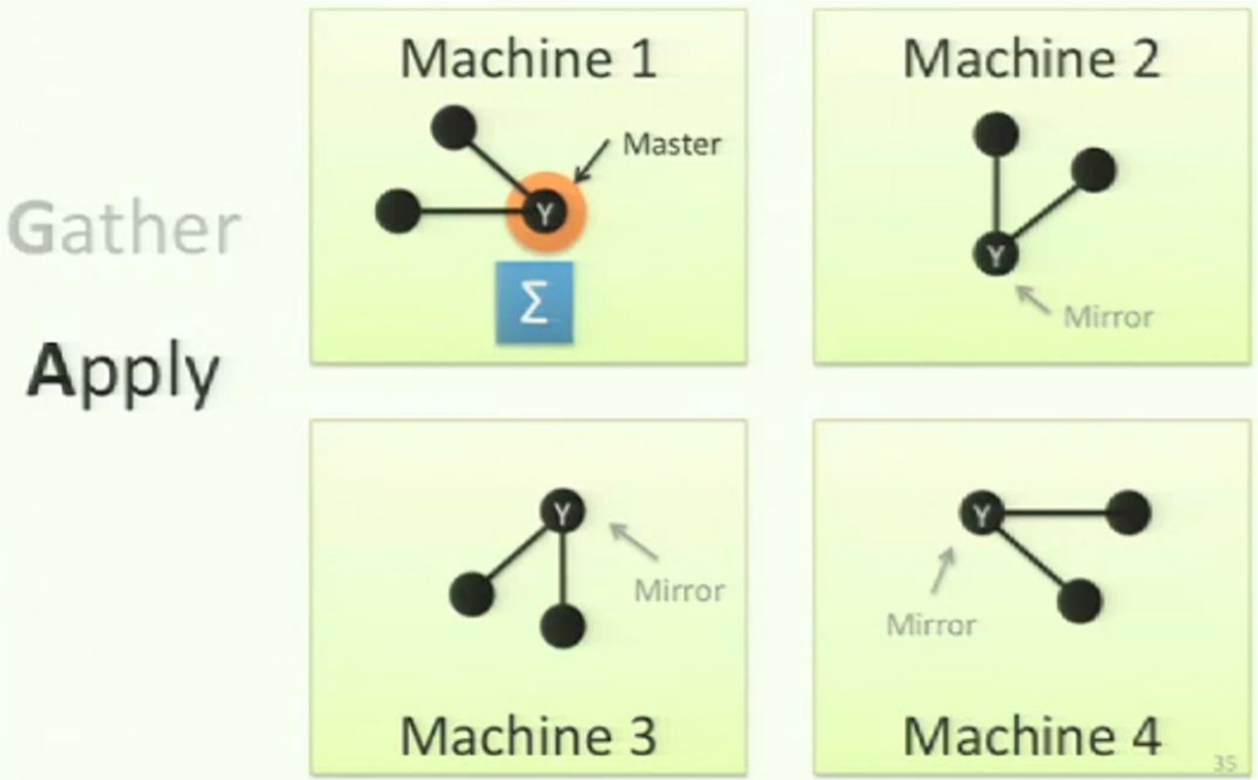
Distribution of A PowerGraph Vertex-Program



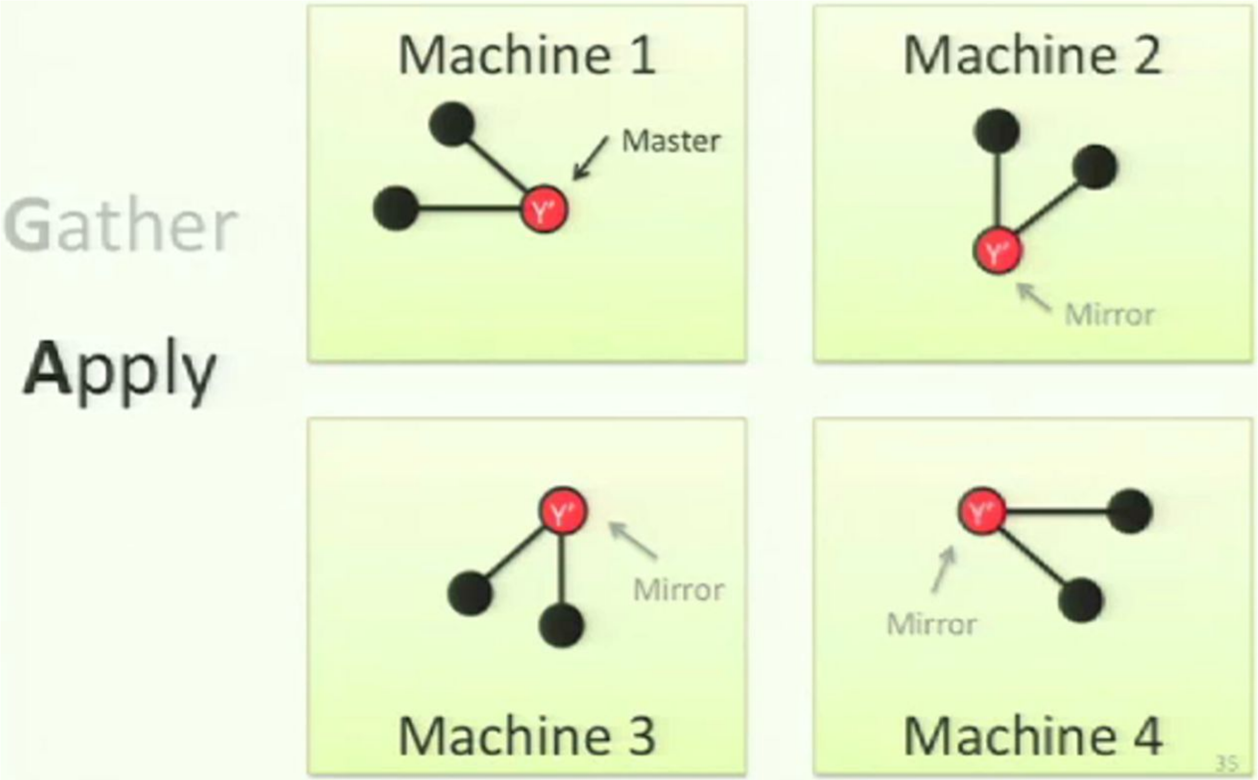
Distribution of A PowerGraph Vertex-Program



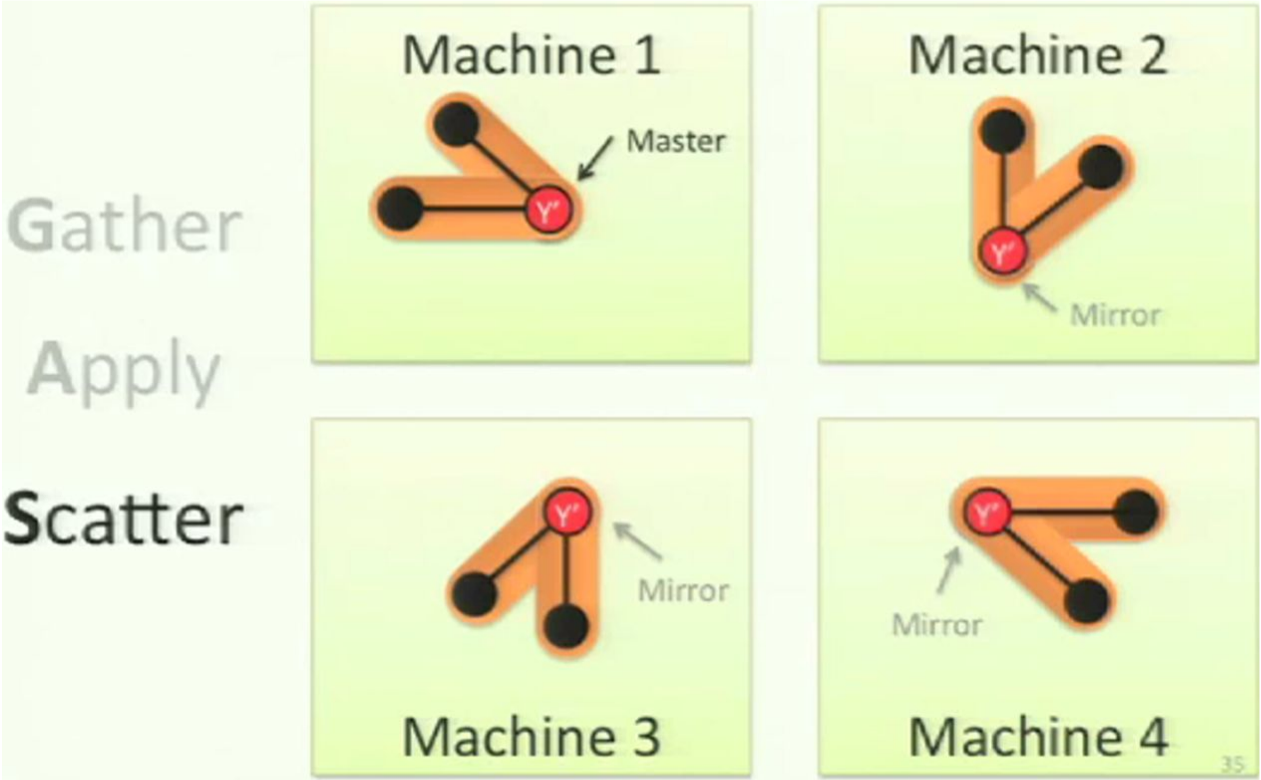
Distribution of A PowerGraph Vertex-Program



Distribution of A PowerGraph Vertex-Program



Distribution of A PowerGraph Vertex-Program



Balanced Vertex-Cut

- Evenly assign edges to machines
- Store edge only once
- Edge data do not need to be communicated

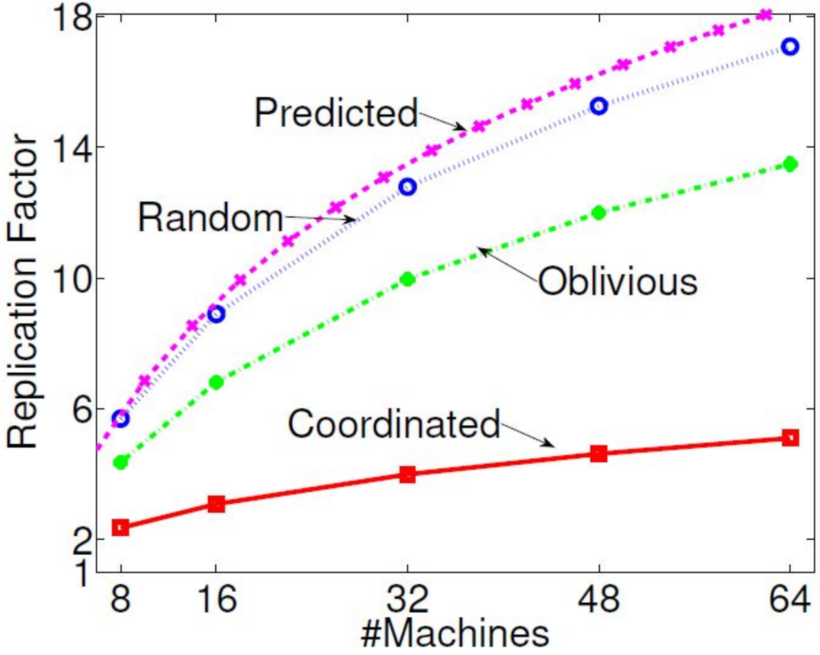
- Allow vertices to span multiple machines
- Changes to a vertex must be copied to all the machines it spans
- **Storage and network overhead depend on the number of machines spanned by each vertex.**

- Theorem: For any edge-cut, we can establish a vertex cut that requires strictly less communication and storage

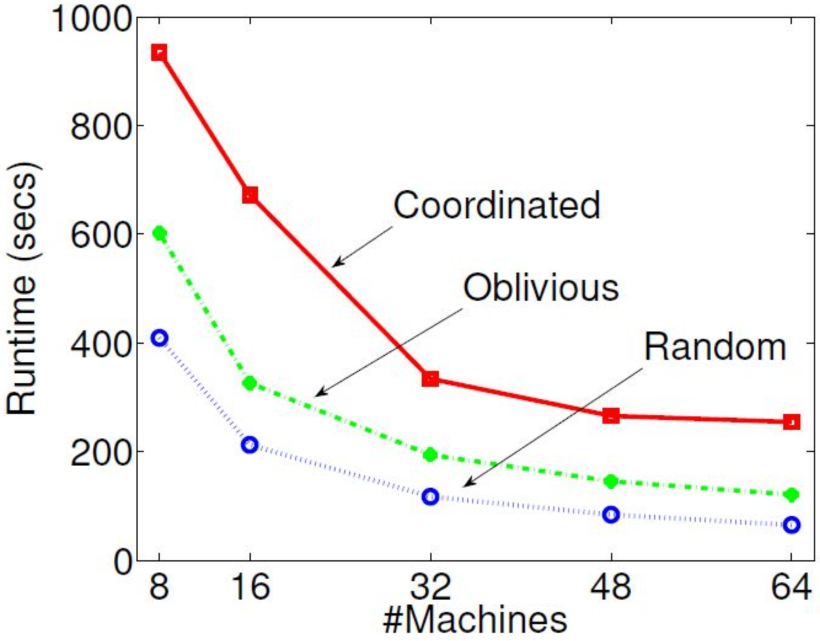
Balanced Vertex-Cut

- Random
- Greedy
  - Assign edge (u, v) to the machine that already contains vertex u or v
  - Assign to least loaded machine if there are multiple choices to ensure work balance
  - Two implementations:
    - Coordinated:
      - Coordination between machines
      - higher quality cuts
      - Slower
    - Oblivious
      - No coordination
      - Low quality cuts
      - Faster

Balanced Vertex-Cut

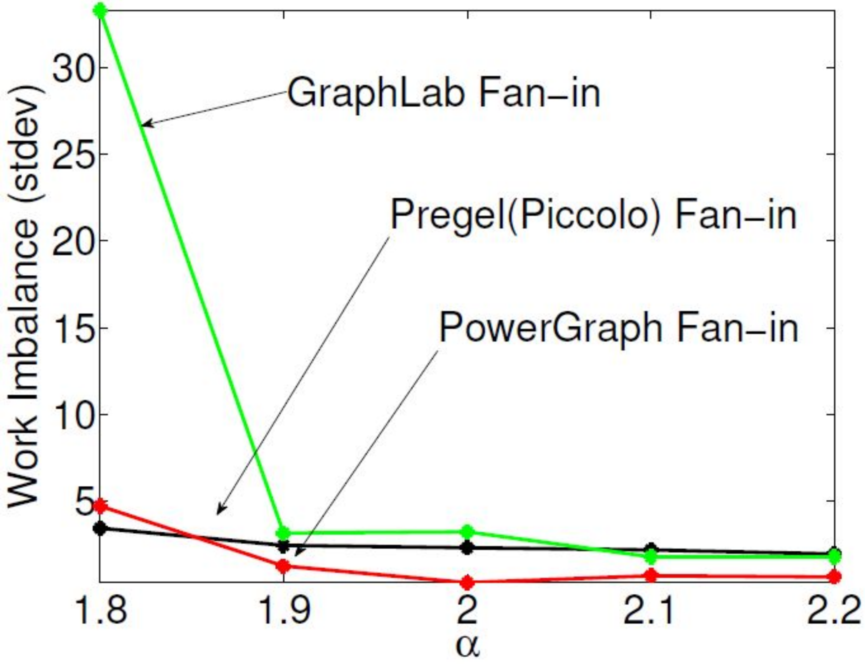


(a) Replication Factor (Twitter)

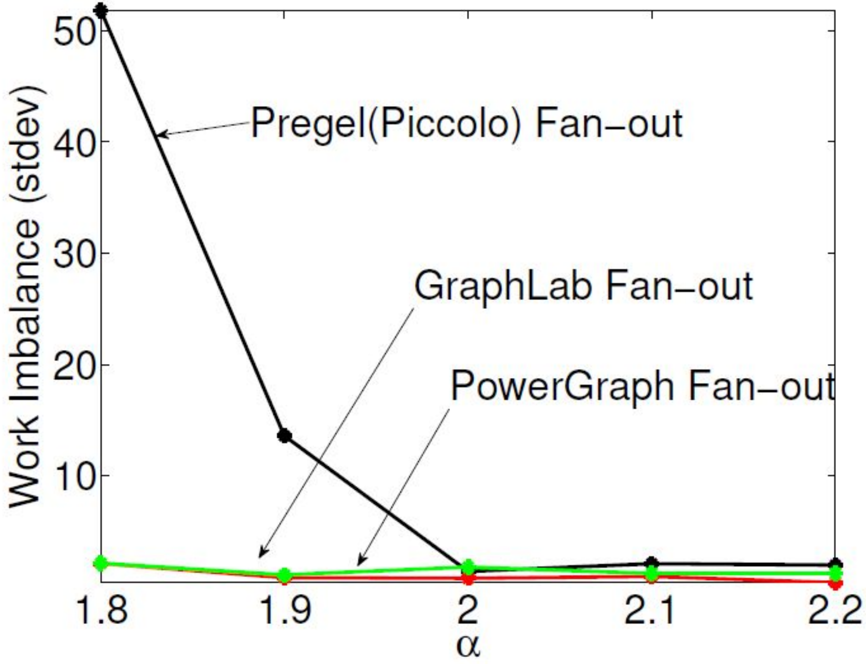


(b) Ingress time (Twitter)

Abstraction Comparison: Work Imbalance

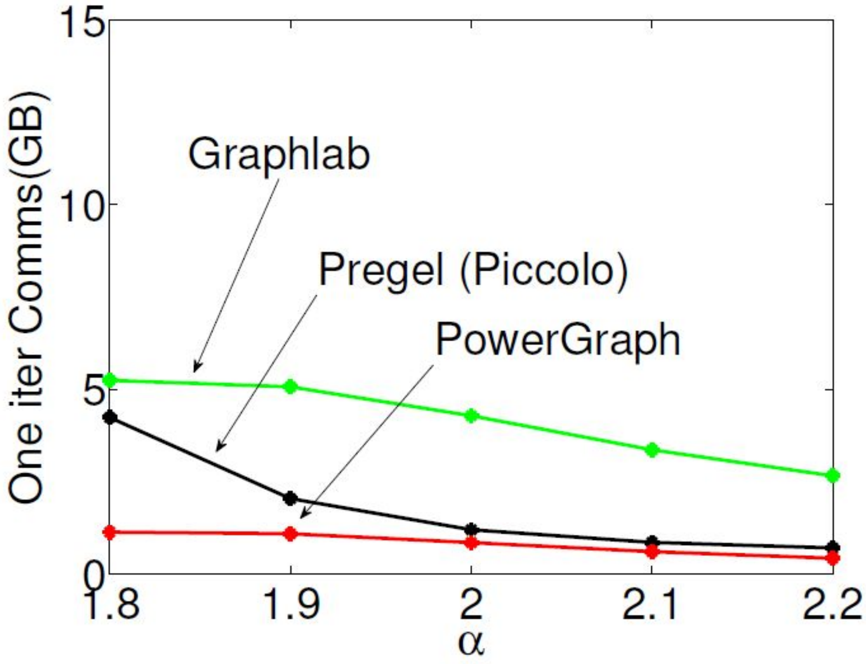


(a) Power-law Fan-In Balance

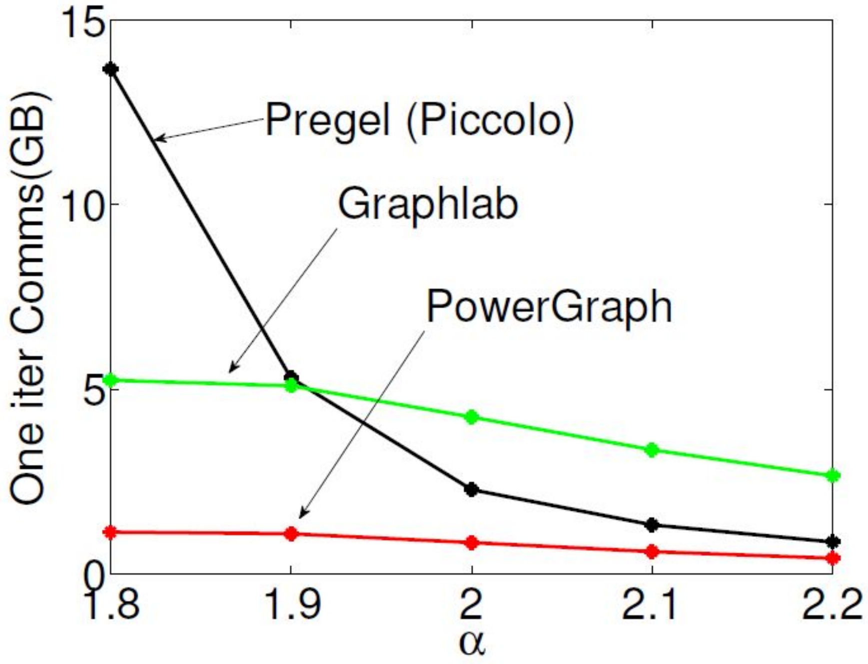


(b) Power-law Fan-Out Balance

Abstraction Comparison: Communication Volume

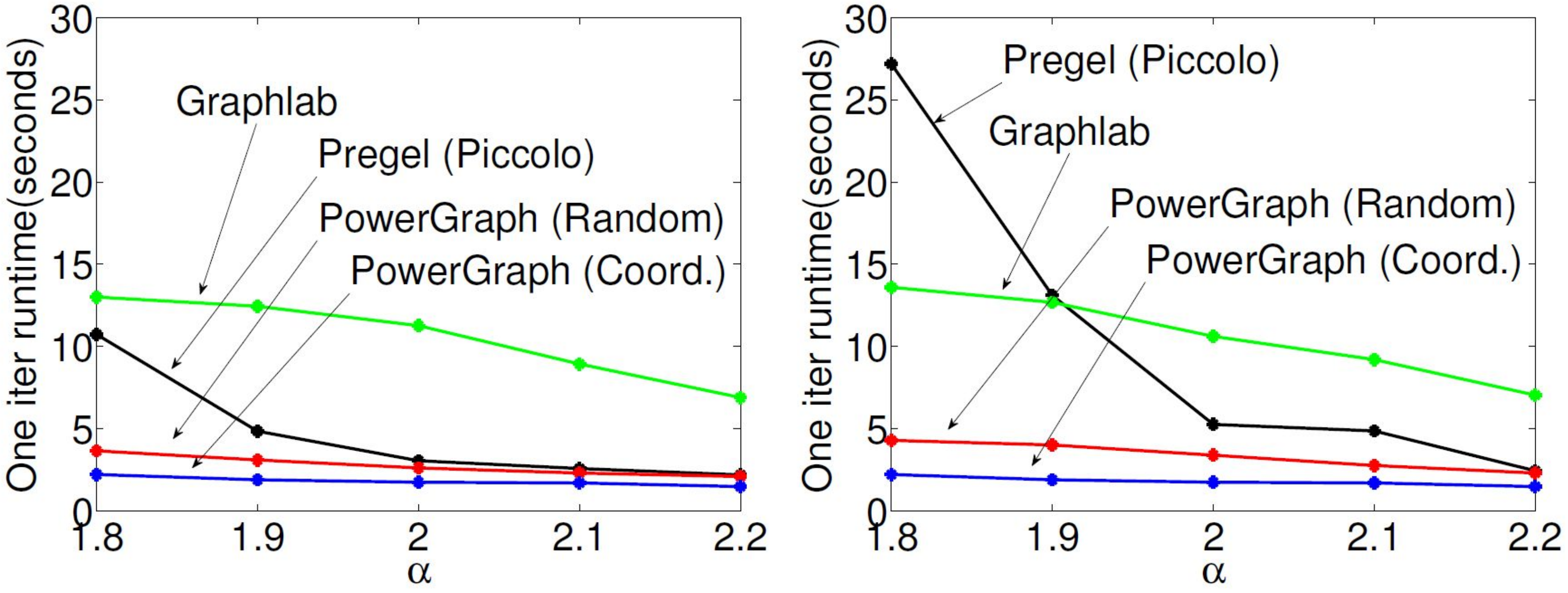


(c) Power-law Fan-In Comm.



(d) Power-law Fan-Out Comm.

Abstraction Comparison: Runtime



(a) Power-law Fan-In Runtime

(b) Power-law Fan-Out Runtime

Summary

- Problem: Computation on large-scale Nature Graphs is challenging
  - High-degree vertices
  - Low quality edge-cut partition
- Solution: PowerGraph
  - GAS Decomposition: distribute vertex programs
  - Balanced Vertex-Cut: partition natural graphs
  - Outperforms existing Graph-Parallel systems

Other Contributions

- A delta caching procedure which allows computation state to be dynamically maintained
- A theoretical characterization of network and storage
- A high-performance open-source implementation of the PowerGraph abstraction
- A comprehensive evaluation of three implementations of PowerGraph on a large EC2 deployment using real-world MLDM applications

Thank you!