PowerGraph

Distributed Graph-Parallel Computation on Natural Graphs

by Gonzalez, Joseph E., et al. at Carnegie Mellon

What is PowerGraph?

- A graph-parallel system that is a distributed version of GraphLab

- Defines program in terms of gather, apply, sum and scatter operations.

- Attempts to handle natural graph problems more efficiently than predecessors (Pregel)

A PowerGraph program

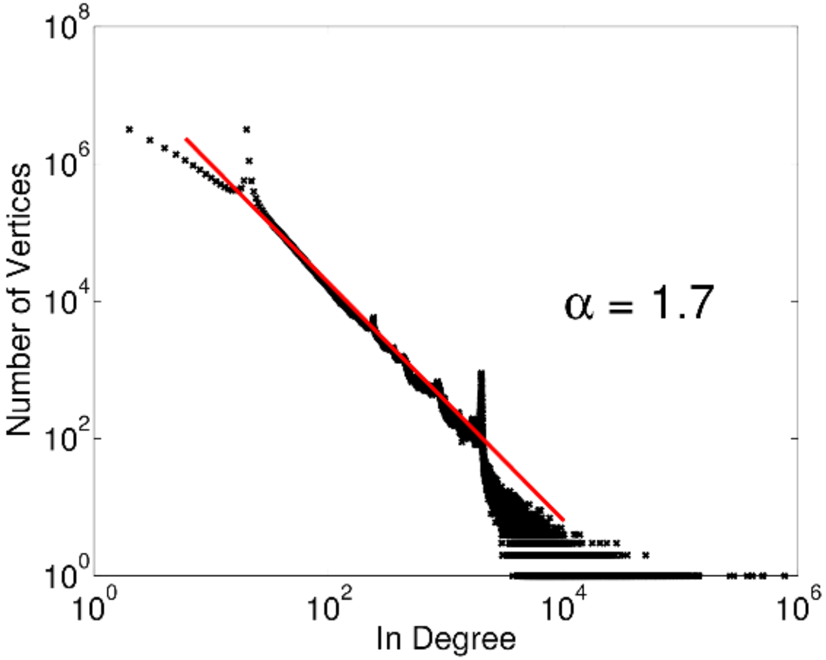
PageRank

```
// gather_nbrs: IN_NBRS
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
```

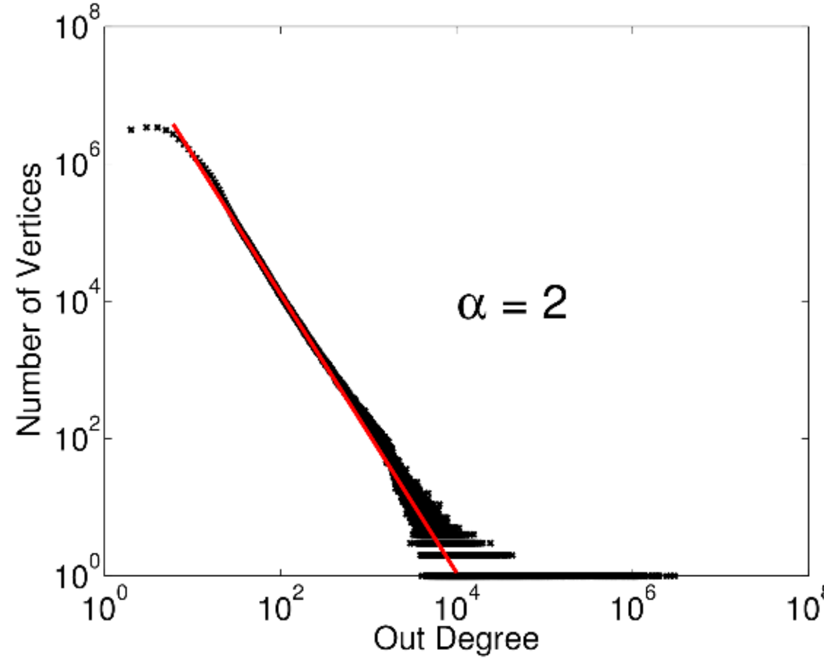
Why do we care about natural graphs?

Why do we care about natural graphs?

- They are natural - we want to work with real world phenomenons
- They often have skewed power-law distributions
- Probability of degree d, $P(d) = d^{-\alpha}$



(a) Twitter In-Degree



(b) Twitter Out-Degree

Challenges of Natural Graphs

- Work Balance
- Partitioning
- Communication
- Storage

How is efficiency obtained with PowerGraph?

- Edge-based distribution of work
- Delta caching
- Asynchronous relaxations
- Greedy vertex cutting / allocation

How is efficiency obtained with PowerGraph?

- **Edge-based distribution of work**

- Delta caching

- **Asynchronous relaxations**

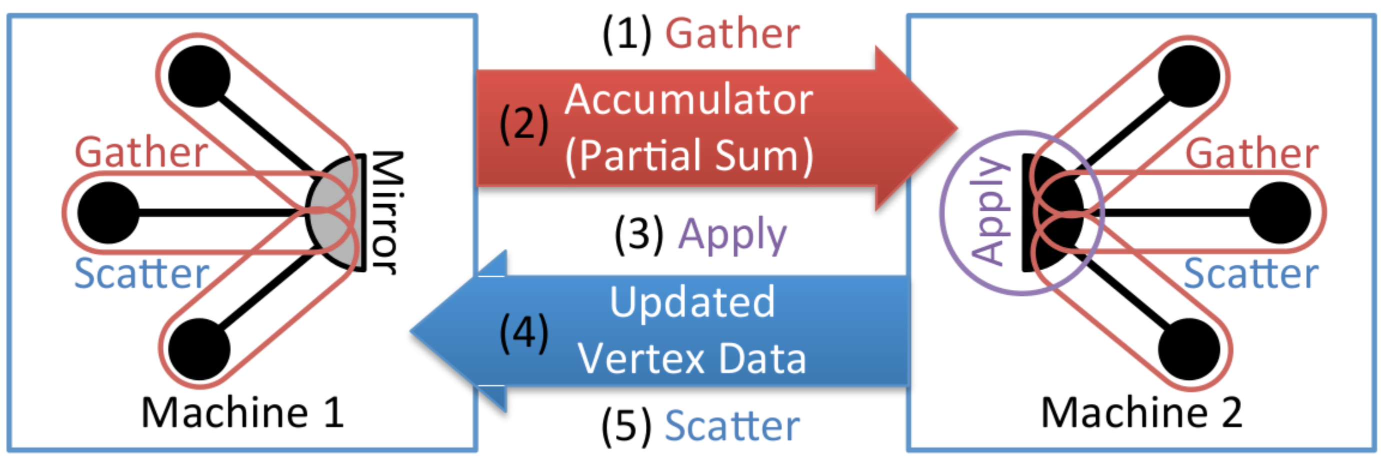
- **Greedy vertex cutting / allocation**

What happens when we can’t fit all edges of a vertex on one machine?

What happens when we can't fit all edges of a vertex on one machine?

Answer: Vertex Mirroring!

- Data mirrored for locality to all nodes
- Apply function only performed on the master nodes



Placement of edges

Let A(v) be the set of machines containing the adjacent edges of vertex v.

For each edge (u,v):

1. If $A(u) \cap A(v) \neq \varnothing$, assign edge to a machine in the intersection.

2. If $A(u) \cap A(v) = \varnothing$ and $A(u) \neq \varnothing$ or $A(v) \neq \varnothing$:

   Assign edge to the machine of the vertex with the most unassigned edges

3. If only one of the two vertices has been assigned, assign the edge to a machine from the assigned vertex.

4. If neither vertex has been assigned, then assign the edge to the least loaded machine.

Placement and fault tolerance

Placement is done either w.r.t local or global state

- Tradeoff between load-time and algorithm run-time

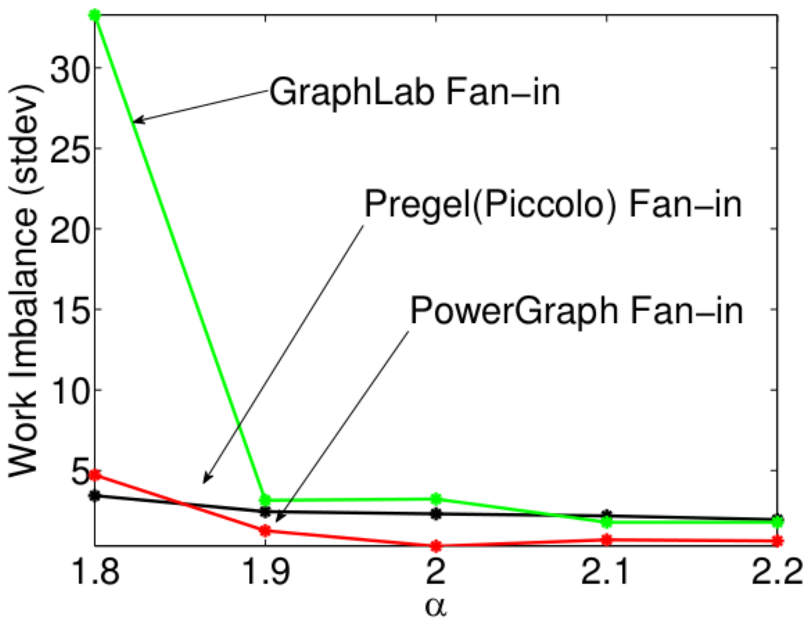
Fault tolerance

- Snapshots are made after each “super-step” i.e. one gather-sum-apply-scatter step

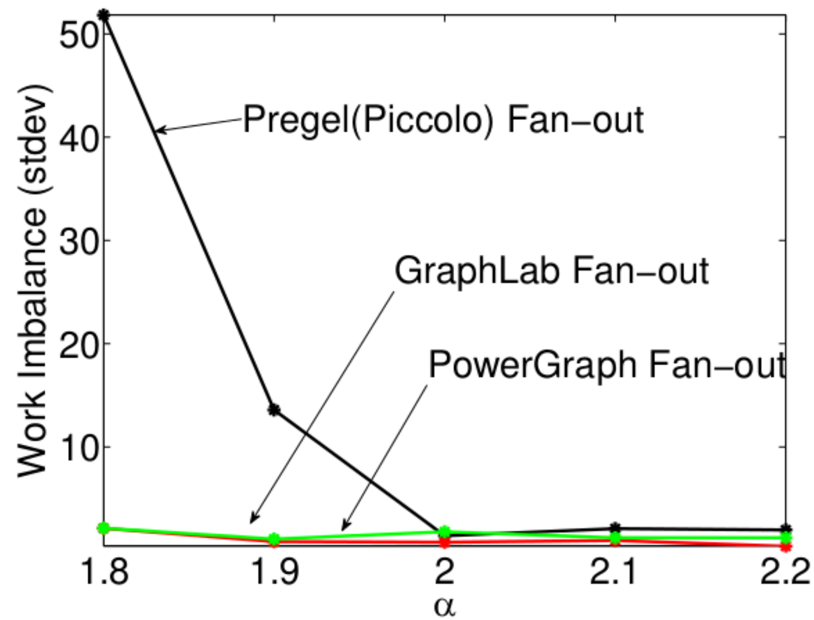
Asynchronicity

- Allows for quicker execution as lock-step barriers are relaxed
- Satisfies sequential consistency and grants exclusive access to arguments
- Attempts to be fair to high degree vertices
- Allows for more rapid convergence for some algorithms

Results - Work Imbalance

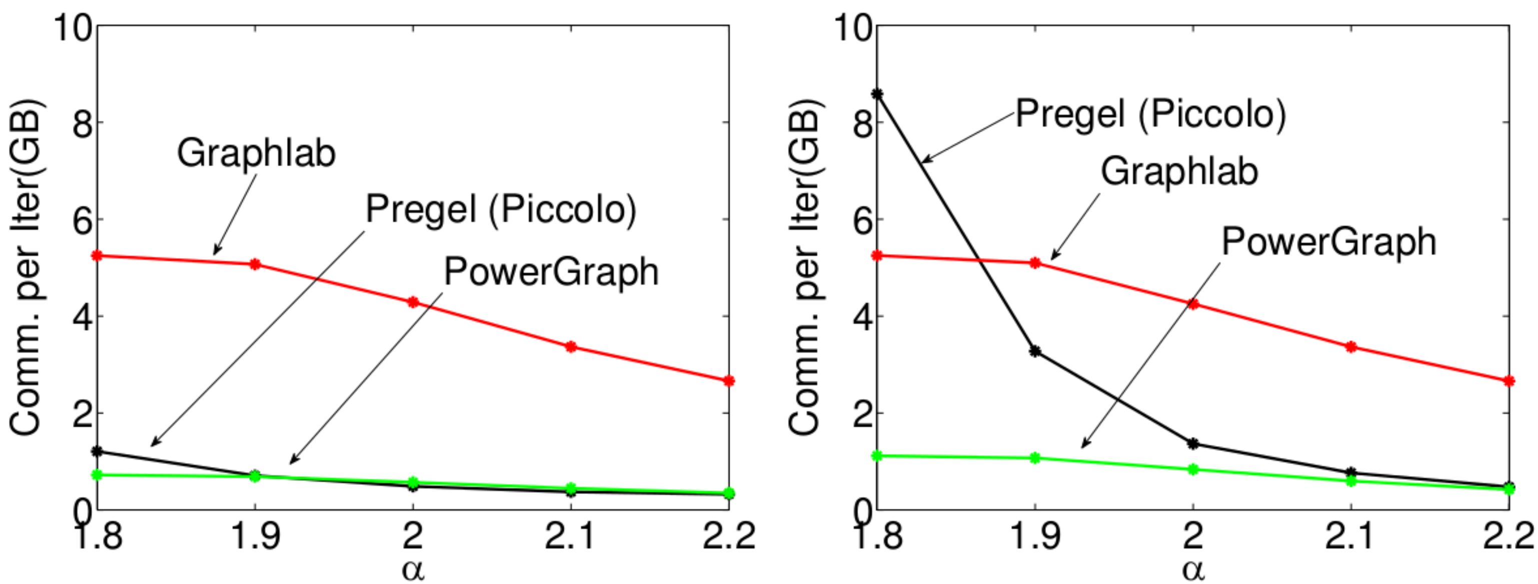


(a) Power-law Fan-In Balance



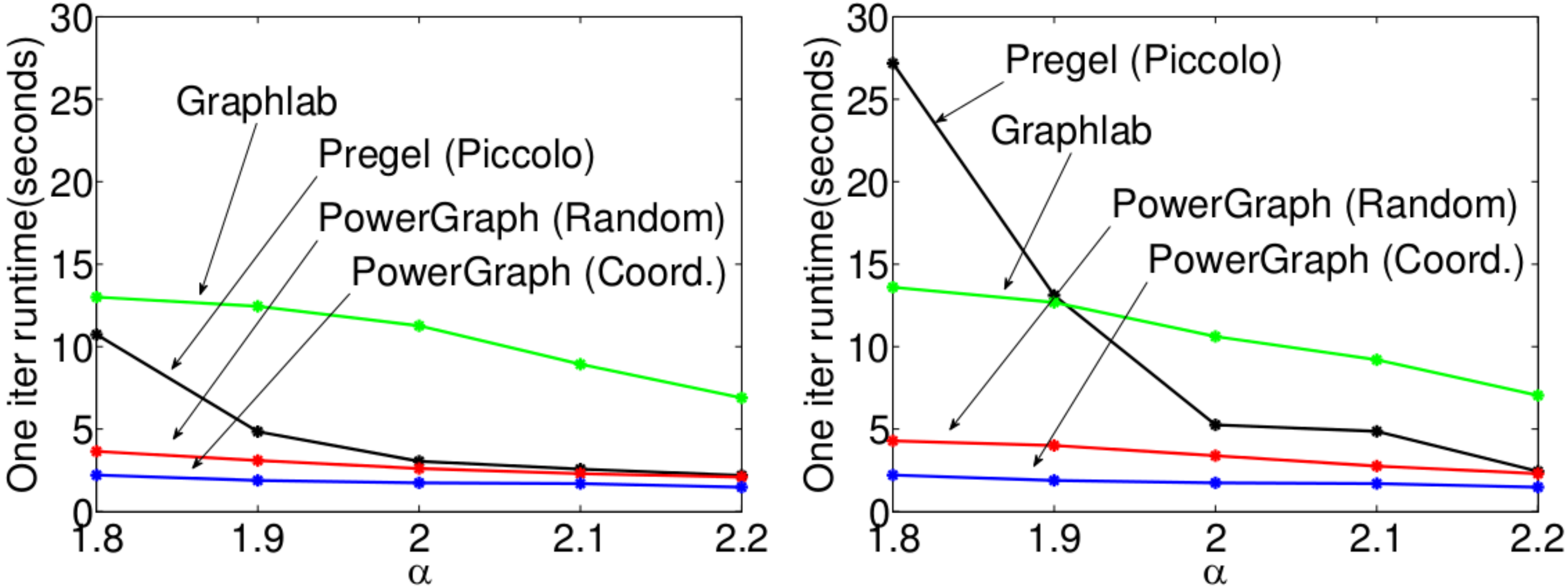
(b) Power-law Fan-Out Balance

Results - Communication



(c) Power-law Fan-In Comm. (d) Power-law Fan-Out Comm.

Results - Runtime



(a) Power-law Fan-In Runtime   (b) Power-law Fan-Out Runtime

Criticism

- Much focus on performance but unfair comparisons for Pregel
- No graphs displaying performance comparisons between synchronous and asynchronous runtimes

Questions?