ADMINISTRIVIA

- Midterm grades (end of) this week
- Course Projects sign up for meetings
- Google Cloud credits
GRAPH DATA

Datasets

Application
GRAPH ANALYTICS

Perform computations on graph-structured data

Examples

  PageRank
  Shortest path
  Connected components
  ...

Message combiner(Message m1, Message m2):
   return Message(m1.value() + m2.value());

void PregelPageRank(Message msg):
   float total = msg.value();
   vertex.val = 0.15 + 0.85*total;
   foreach(nbr in out_neighbors):
      SendMsg(nbr, vertex.val/num_out_nbrs);
NATURAL GRAPHS

(a) Twitter In-Degree

$\alpha = 1.7$
POWERGRAPH

Programming Model:
Gather-Apply-Scatter

Better Graph Partitioning
with vertex cuts

Distributed execution (Sync, Async)
GATHER-APPLY-SCATTER

**Gather:** Accumulate info from nbrs

```
// gather_nbrs: IN_NBRS
gather(Du, D(u,v), Dv):
    return Dv.rank / #outNbrs(v)
```

**Apply:** Accumulated value to vertex

```
sum(a, b): return a+b
```

```
apply(Du, acc):
    rnew = 0.15 + 0.85 * acc
    Du.delta = (rnew - Du.rank)/
               #outNbrs(u)
    Du.rank = rnew
```

**Scatter:** Update adjacent edges, vertices

```
// scatter_nbrs: OUT_NBRS
scatter(Du,D(u,v),Dv):
    if(|Du.delta|> ε) Activate(v)
    return delta
```
EXECUTION MODEL, CACHING

Active Queue

Delta caching
  Cache accumulator value for vertex
  Optionally scatter returns a delta
  Accumulate deltas
SYNC VS ASYNC

Sync Execution
  Gather for all active vertices,
  followed by Apply, Scatter
  Barrier after each minor-step

Async Execution
  Execute active vertices,
  as cores become available
  No Barriers! Optionally serializable
DISTRIBUTED EXECUTION

Symmetric system, no coordinator

Load graph into each machine

Communicate across machines to spread updates, read state
GRAPH PARTITIONING

(a) Edge-Cut

(b) Vertex-Cut
RANDOM, GREEDY OBLIVIOUS

Three distributed approaches:
Random Placement

Coordinated Greedy Placement

Oblivious Greedy Placement
OTHER FEATURES

Async Serializable engine
  Preventing adjacent vertex from running simultaneously
  Acquire locks for all adjacent vertices

Fault Tolerance
  Checkpoint at the end of super-step for sync
  For Async?
DISCUSSION

https://forms.gle/t2Tj4sEFDNZ8aDBo7
Consider the PageRank implementation in Spark vs synchronous PageRank in PowerGraph. What are some reasons why PowerGraph might be faster?
(a) Twitter PageRank Runtime

(b) Twitter PageRank Comms

(c) Twitter PageRank Delta Cache
What could be one shortcoming of PowerGraph compared to prior systems like MapReduce or Spark?
Next steps

Next class: GraphX
Sign up for project check-ins!