CLOUD COMPUTING CONCEPTS

with Indranil Gupta (Indy)

DISTRIBUTED GRAPH PROCESSING

Lecture A

DISTRIBUTED GRAPH PROCESSING

What We'll Cover

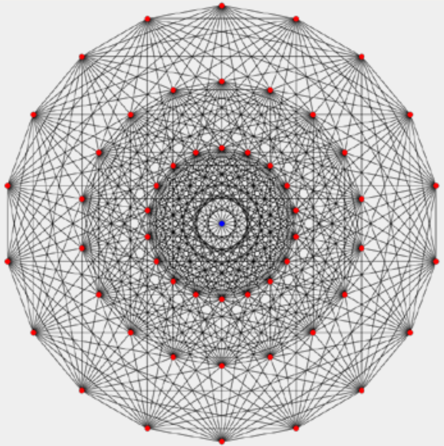
- Distributed Graph Processing
- Google's Pregel system
  - Inspiration for many newer graph processing systems: Piccolo, Giraph, GraphLab, PowerGraph, LFGraph, X-Stream, etc.

What's a Graph?

- A graph is not a plot!
- A graph is a “network”
- A graph has vertices (i.e., nodes)
  - E.g., in the Facebook graph, each user = a vertex (or a node)
- A graph has edges that connect pairs of vertices
  - E.g., in the Facebook graph, a friend relationship = an edge

Lots of Graphs

- Large graphs are all around us
  - Internet Graph: vertices are routers/switches and edges are links
  - World Wide Web: vertices are webpages, and edges are URL links on a webpage pointing to another webpage
    - Called "Directed" graph as edges are uni-directional
  - Social graphs: Facebook, Twitter, LinkedIn
  - Biological graphs: DNA interaction graphs, ecosystem graphs, etc.



Source: Wikimedia Commons

Graph Processing Operations

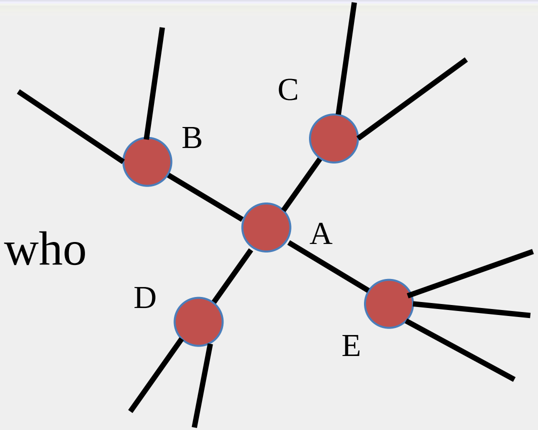
- Need to derive properties from these graphs
- Need to summarize these graphs into statistics
- E.g., find shortest paths between pairs of vertices
  - Internet (for routing)
  - LinkedIn (degrees of separation)
- E.g., do matching
  - Dating graphs in match.com (for better dates)
- And many (many) other examples!

Why Hard?

- Because these graphs are large!
  - Human social network has 100s Millions of vertices and Billions of edges
  - WWW has Millions of vertices and edges
- Hard to store the entire graph on one server and process it
  - Slow on one server (even if beefy!)
- Use distributed cluster/cloud!

Typical Graph Processing Application

- Works in *iterations*
- Each vertex assigned a *value*
- In each iteration, each vertex:
  1. Gathers values from its immediate neighbors (vertices who join it directly with an edge). E.g., @A: B→A, C→A, D→A,…
  2. Does some computation using its own value and its neighbors values.
  3. Updates its new value and sends it out to its neighboring vertices. E.g., A→B, C, D, E
- Graph processing terminates after: i) fixed iterations, or ii) vertices stop changing values



Hadoop/MapReduce to the Rescue?

- Multi-stage Hadoop
- Each stage == 1 graph iteration
- Assign vertex ids as keys in the reduce phase

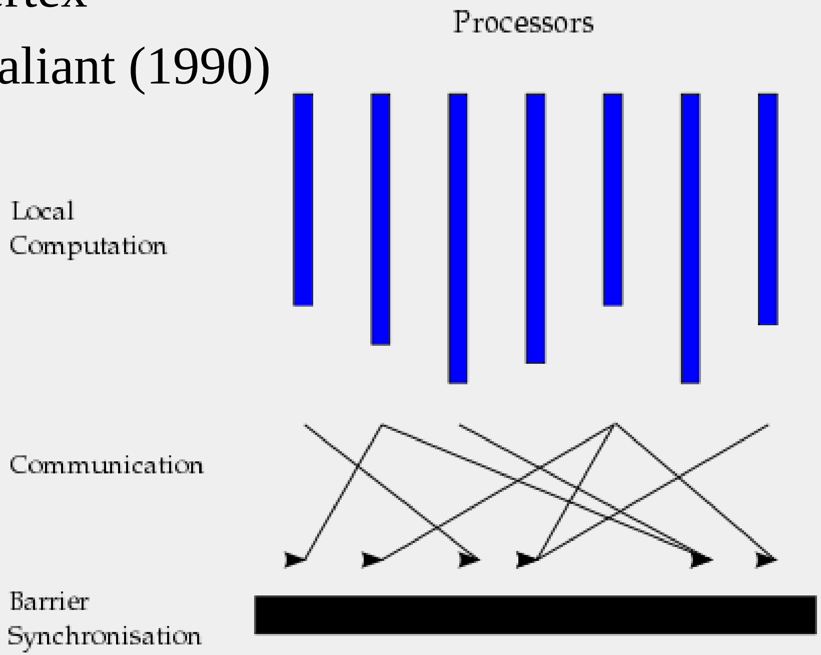
☺ Well-known

☹ At the end of every stage, transfer all vertices over network

- ☹ All vertex values written to HDFS (file system)
- ☹ Very slow!

Bulk Synchronous Parallel Model

- "Think like a vertex"
- Originally by Valiant (1990)



Source: http://en.wikipedia.org/wiki/Bulk_synchronous_parallel

Basic Distributed Graph Processing

- “Think like a vertex”
- Assign each vertex to one server
- Each server thus gets a subset of vertices
- In each iteration, each server performs **Gather-Apply-Scatter** for all its assigned vertices
  - Gather: get all neighboring vertices’ values
  - Apply: compute own new value from own old value and gathered neighbors’ values
  - Scatter: send own new value to neighboring vertices

Assigning Vertices

- How to decide which server a given vertex is assigned to?
- Different options
  - Hash-based: Hash(vertex id) modulo number of servers
    - Remember consistent hashing from P2P systems?!
  - Locality-based: Assign vertices with more neighbors to the same server as its neighbors
    - Reduces server to server communication volume after each iteration

Pregel System By Google

- Pregel uses the master/worker model
  - Master (one server)
    - Maintains list of worker servers
    - Monitors workers; restarts them on failure
    - Provides Web-UI monitoring tool of job progress
  - Worker (rest of the servers)
    - Processes its vertices
    - Communicates with the other workers
- Persistent data is stored as files on a distributed storage system (such as GFS or BigTable)
- Temporary data is stored on local disk

Pregel Execution

1. Many copies of the program begin executing on a cluster
2. The master assigns a partition of input (vertices) to each worker
   - Each worker loads the vertices and marks them as *active*
3. The master instructs each worker to perform a iteration
   - Each worker loops through its active vertices & computes for each vertex
   - Messages can be sent whenever, but need to be delivered before the end of the iteration (i.e., the barrier)
   - When all workers reach iteration barrier, master starts next iteration
4. Computation halts when, in some iteration: no vertices are active and when no messages are in transit
5. Master instructs each worker to save its portion of the graph

Fault-Tolerance in Pregel

- **Checkpointing**
  - Periodically, master instructs the workers to save state of their partitions to persistent storage
    - e.g., Vertex values, edge values, incoming messages
- **Failure detection**
  - Using periodic "ping" messages from master → worker
- **Recovery**
  - The master reassigns graph partitions to the currently available workers
  - The workers all reload their partition state from most recent available checkpoint

How Fast Is It?

- Shortest paths from one vertex to all vertices
  - SSSP: “Single Source Shortest Path”
- On 1 Billion vertex graph (tree)
  - 50 workers: 180 seconds
  - 800 workers: 20 seconds
- 50 B vertices on 800 workers: 700 seconds (~12 minutes)
- Pretty Fast!

Summary

- Lots of (large) graphs around us
- Need to process these
- MapReduce not a good match
- Distributed Graph Processing systems: Pregel by Google
- Many follow-up systems
  - Piccolo, Giraph: Pregel-like
  - GraphLab, PowerGraph, LFGraph, X-Stream: more advanced