DISTRIBUTED RESEARCH ON EMERGING APPLICATIONS & MACHINES
dream-lab.in | **Indian Institute of Science**, Bangalore

DREAM:Lab

SE252:Lecture 15/16, Mar 3/10
ILO3:Algorithms and Programming Patterns for Cloud Applications *(Giraph)*

Yogesh Simmhan

1. Review of Homework A

Plagiarism Warning

2. Review of Midterm Exam

The 3D's of Big Data

- **Volume**
  - Sheer size of data. Storage, mgmt., bandwidth
- *Velocity*
  - Realtime processing, ephemeral, latency
- Variety
  - Complexity, linked data analysis, compute+I/O
- Not exclusive dimensions, but useful
- Volume is well studied, but Velocity & Variety less so

Hadoop for Graphs

- PageRank is a Graph algorithm

```
Map(nid n, node N)
   p = N.PageRank/|N.AdjacencyList|
   Emit(nid n, N)                    // Pass along graph structure
   forall nodeid m in N.AdjacencyList do
      Emit(nid m, p)                 // Pass PageRank to neighbors
Reduce(nid m, P[])
   M = 0
   forall p in P[] do
      if IsNode(p) then M=p          // Recover graph
      else s = s + p                 // Sum incoming PageRank
   M.PageRank = s
   Emit(nid m, node M)
```

- Stationary algorithm
  - No traversal, constant cost per job
  - Traversals are more tricky

Data-Intensive Text Processing with MapReduce, Jimmy Lin and Chris Dyer, 2010

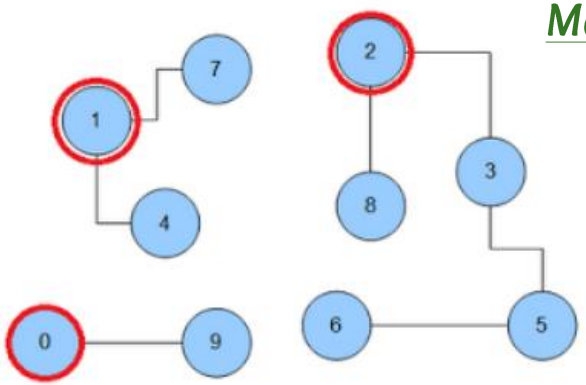
Connected Components in Hadoop

```
vert: vid, min, neighbours[], active
Map(key, vert)
  if(iter==1)                              // init min, active
       vert.min = min(vert.neighbours)
       vert.active = true
  Emit(vert.vid, vert)                     // propagate adj list
  if(vert.active) then                     // send message
       foreach neighbour in vert.neighbours[]
            Emit(neighbour, vert.min)

 Reduce(vid, vert[])
  vert[0].active = false
  if vert[0].min > min(vert[1..n]) then  // messages
       vert[0].min = min(vert[1..n])
       vert[0].active = true
  Emit(vid, vert)                          // save (updated) graph
```

http://codingwiththomas.blogspot.in/2011/04/graph-exploration-with-hadoop-mapreduce.html

Connected Components in Hadoop



Adjacency List

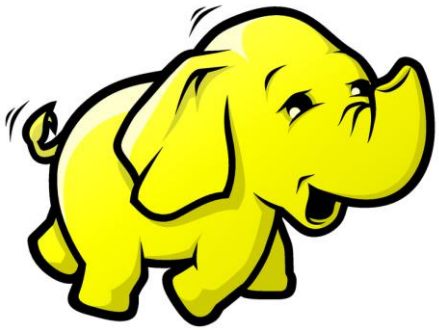
```
0
1   4   7
2   3   8
3   5
4   1
5   6
6
7
8
9   0
```

	Iteration 1		Iteration 2		Iteration 3	
Map	0, 0, *, +	4, 1	0, 0, *, -	5, 2	0, 0, *, -	6, 2
	1, 1, *, +	7, 1	1, 1, *, -	6, 3	1, 1, *, -	
	2, 2, *, +	3, 2	2, 2, *, -		2, 2, *, -	
	3, 3, *, +	8, 2	3, 2, *, +		3, 2, *, -	
	4, 1, *, +	5, 3	4, 1, *, -		4, 1, *, -	
	5, 5, *, +	1, 1	5, 3, *, +		5, 2, *, +	
	6, 6, *, +	6, 5	6, 5, *, +		6, 3, *, +	
	7, 7, *, +	0, 0	7, 1, *, +		7, 1, *, -	
	8, 8, *, +		8, 2, *, +		8, 2, *, -	
	9, 0, *, +		9, 0, *, -		9, 0, *, -	
Reduce	0, 0, *, -		0, 0, *, -		0, 0, *, -	
	1, 1, *, -		1, 1, *, -		1, 1, *, -	
	2, 2, *, -		2, 2, *, -		2, 2, *, -	
	3, 2, *, +		3, 2, *, -		3, 2, *, -	
	4, 1, *, -		4, 1, *, -		4, 1, *, -	
	5, 3, *, +		5, 2, *, +		5, 2, *, -	
	6, 5, *, +		6, 3, *, +		6, 2, *, +	
	7, 1, *, +		7, 1, *, -		7, 1, *, -	
	8, 2, *, +		8, 2, *, -		8, 2, *, -	
	9, 0, *, -		9, 0, *, -		9, 0, *, -	

http://codingwiththomas.blogspot.in/2011/04/graph-exploration-with-hadoop-mapreduce.html

Hadoop for Graphs

- Traversals are more tricky
  - BFS one level at a time
  - Need to perform several MR jobs, Overheads
  - Read/Write graphs to HDFS multiple times
  - Shuffle for each step



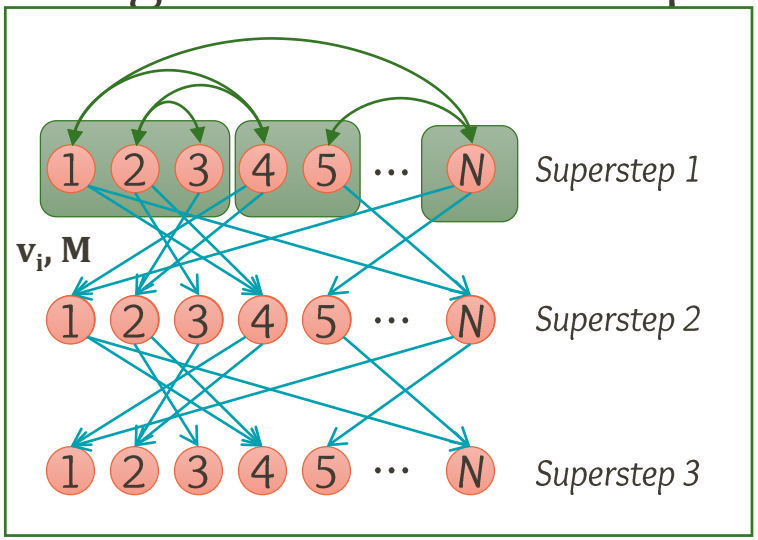
Google Pregel

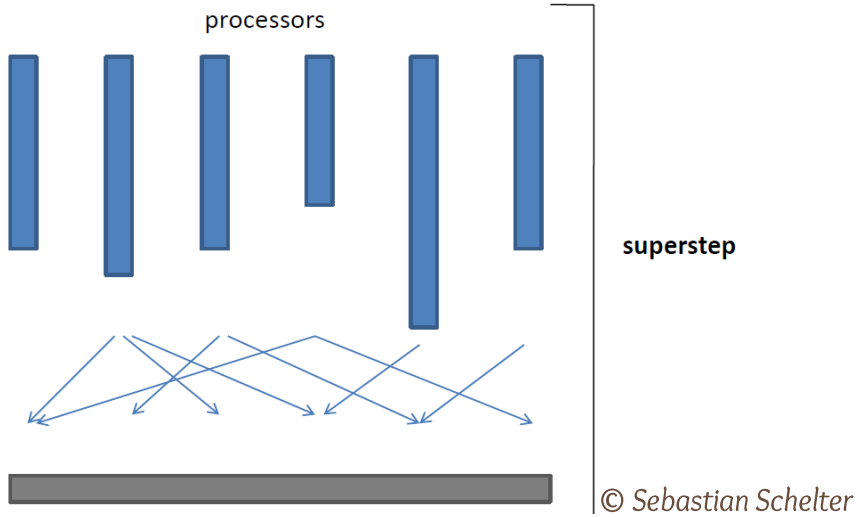
- Shift from tuple centric (MR) to vertex centric
  - Logic written from perspective of a vertex
- Message passing between vertices
  - Avoid graph transmission
- Implicit iterative execution
  - Bulk Synchronous Parallel (BSP)

Pregel: A System for Large-Scale Graph Processing, Grzegorz Malewicz, et al., 2010 ACM SIGMOD

Valiant's BSP Model

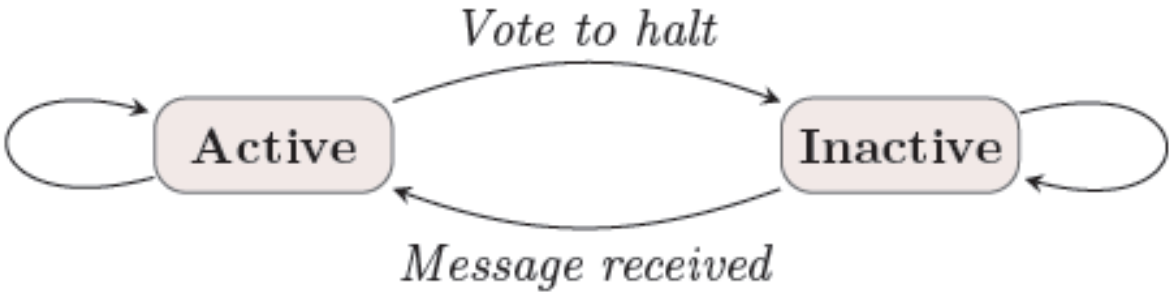
- Series of synchronized supersteps
- Message passing between independent tasks on distributed hosts
  - Messages delivered at superstep boundaries





- Extended to vertices (tasks) across partitions (hosts)
- Message passing along edges between vertices

Vertex Centric Logic



- In superstep 1, every vertex is *active*
- A vertex deactivates itself by voting to halt.
- Vertex can be reactivated by receiving a message
- Algorithm terminates when every vertex *votes to halt*

Pregel: A System for Large-Scale Graph Processing, Grzegorz Malewicz, et al., 2010 ACM SIGMOD

Vertex Centric Logic

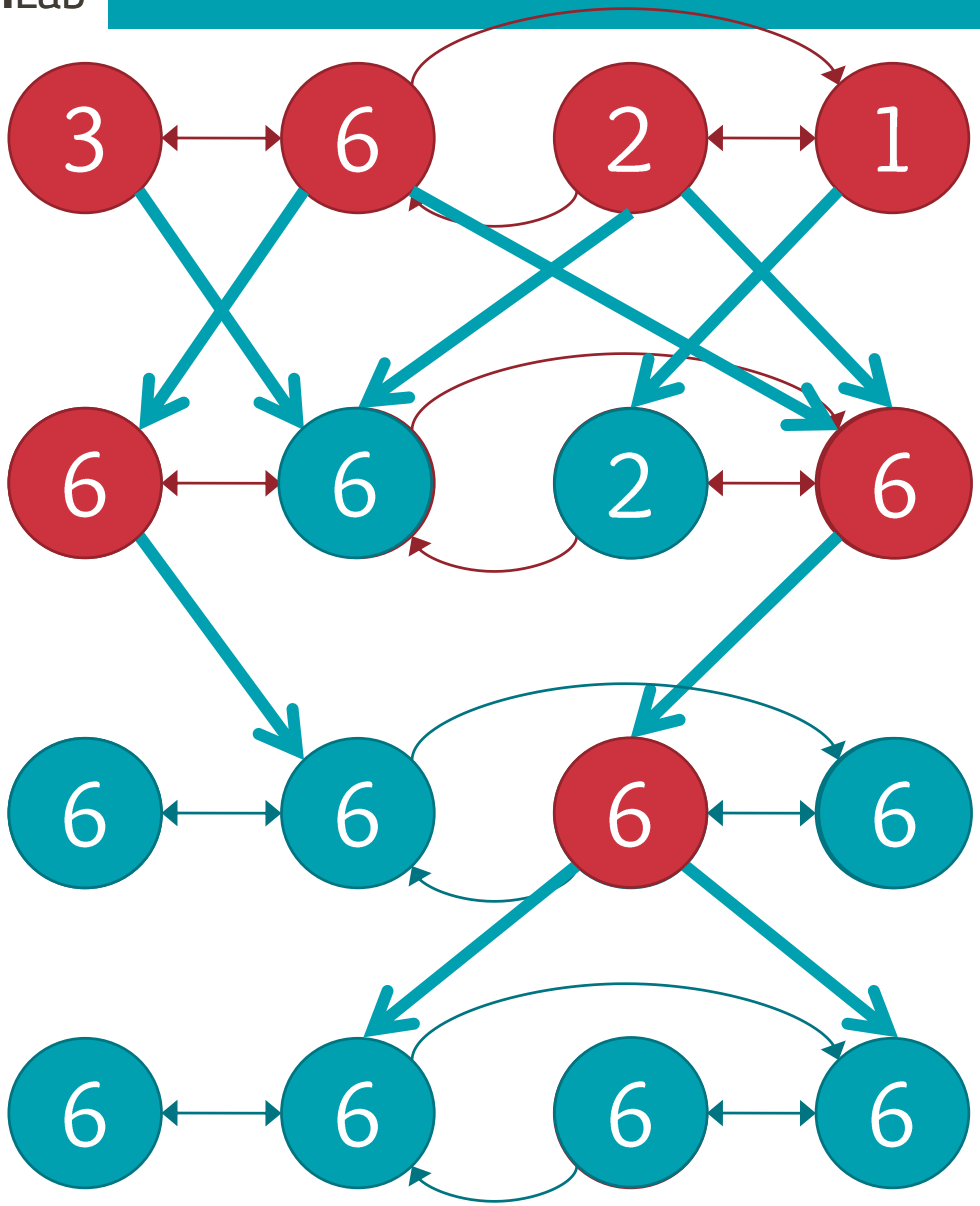
compute(Message msgs[])

- getVertexId
- get/setVertexValue
- getOutEdges
- getSuperstep
- voteToHalt
- isHalted
- sendMsg(vid, msg)
- sentMsgToAllEdges(msg)

Max Vertex using Vertex Centric Logic

Algorithm 1 Max Vertex Value using Vertex Centric Model

```
1:  procedure COMPUTE(Vertex myVertex, Iterator⟨Message⟩ M)
2:      hasChanged = (superstep == 1) ? true : false
3:      while M.hasNext do          ▶ Update to max message value
4:          Message m ← M.next
5:          if m.value > myVertex.value then
6:              myVertex.value ← m.value
7:              hasChanged = true
8:      if hasChanged then          ▶ Send message to neighbors
9:          SENDTOALLNEIGHBORS(myVertex.value)
10:     else
11:         VOTETOHALT( )
```



Blue Arrows are messages.

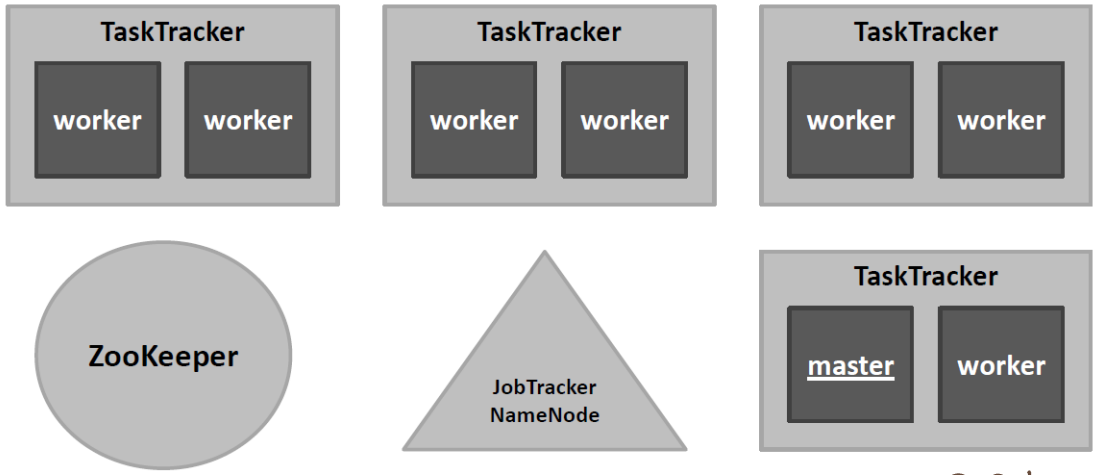
Blue vertices have voted to halt.

Max Vertex using Vertex Centric Logic

Pregel: A System for Large-Scale Graph Processing, Grzegorz Malewicz, et al., 2010 ACM SIGMOD

Apache Giraph – *Pregel using Hadoop*

- Hadoop Map-only Application
- **ZooKeeper:** responsible for **computation state**
  - Partition/worker mapping, global #superstep
- **Master:** responsible for **coordination**
  - Assigns partitions to workers, synchronization
- **Worker:** responsible for **vertices**
  - Invokes active vertices compute() function, sends, receives and assigns messages



Upcoming Deadlines

- Research Paper mid-term draft submission is due on **Tue 10 Mar.**
  - **Submission is required**, even if ungraded!
- Mid-term project report due on **Thu 12 Mar.**
- Demo on Fri 13 Mar
  - Upload code to GitHub
  - Signup for presentation slots from 3-6PM (25mins each; all welcome to attend)
  - *Yogesh to send account info & doodle slots*