Cloud Computing

- PaaS Techniques
  - Programming Model

Agenda

- Overview
  - Hadoop & Google
- PaaS Techniques
  - File System
    - GFS, HDFS
  - Programming Model
    - MapReduce, Pregel
  - Storage System for Structured Data
    - Bigtable, Hbase

How to process large data sets and easily utilize the resources of a large distributed system …

MapReduce

Introduction

Programming Model

Implementation

Refinement

Hadoop MapReduce

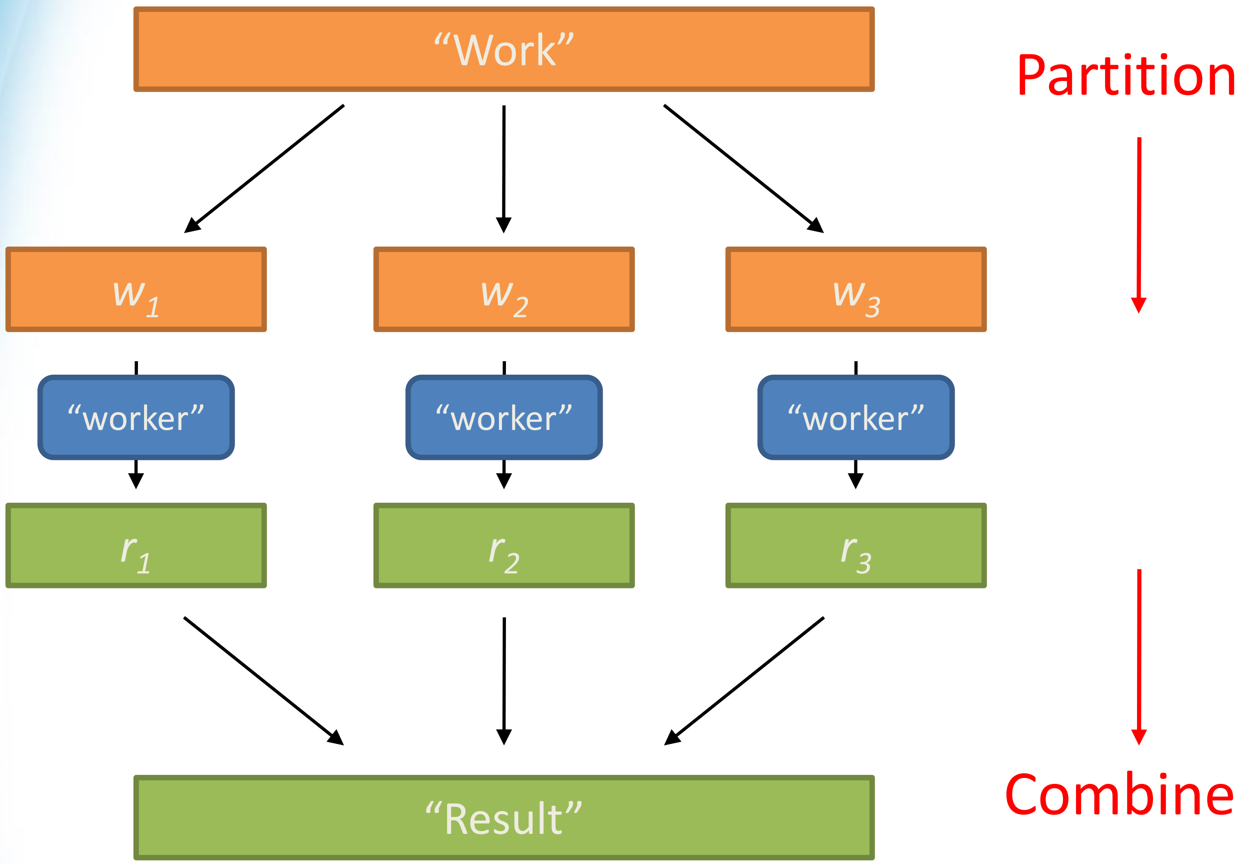
MAPREDUCE

How much data?

- Facebook internally distributes 800PB of "hot data" daily (2022)
- Wayback Machine 70PB (December 2020)
- How about the future…

640K ought to be enough for anybody.

Divide and Conquer



Pregel

A system for large-scale graph processing

Introduction

- The Internet made the Web graph a poplar object of analysis and research.
- In Google, MapReduce is used for 80% of all the data processing needs.
- The other 20% is handled by a lesser known infrastructure called Pregel which is optimized to mine relationships from graphs.

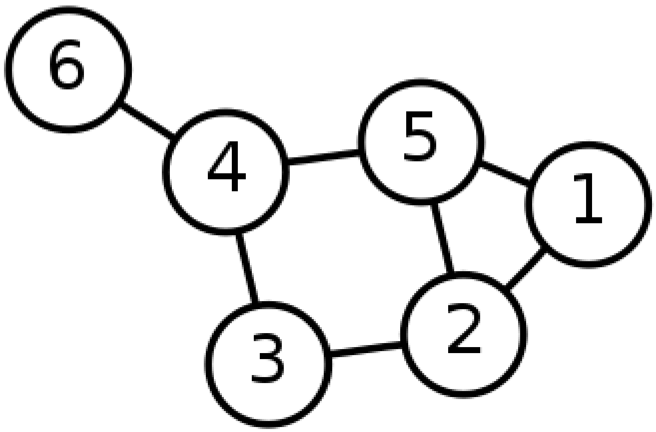
Introduction(cont.)

- *Graph is a collection of vertices or nodes and a collection of edges that connect pair of nodes.*

  - wikipedia

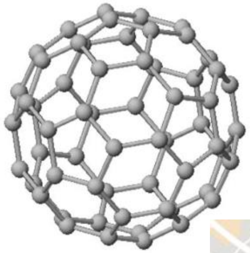
- *A graph is a collection of points and lines connecting some (possibly empty) subset of them.*

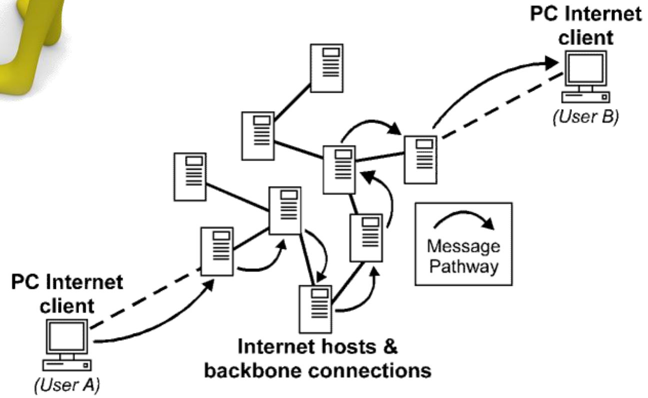
  - mathworld

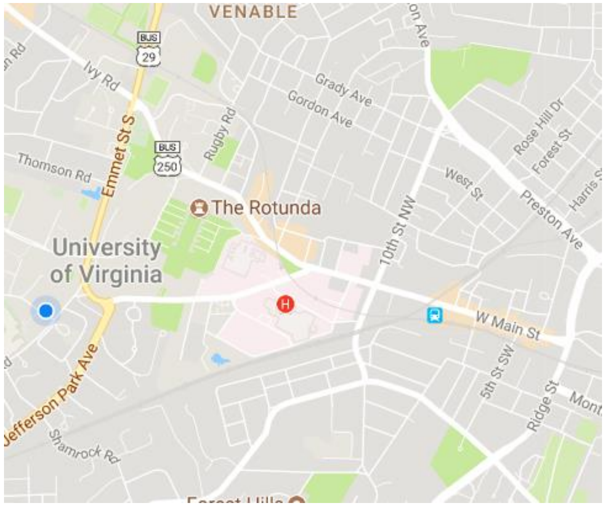


Introduction(cont.)

- Graph does not just mean the image, most of the time in Internet, graph means the relations between nodes.







Model

Implement

Communication

MODEL

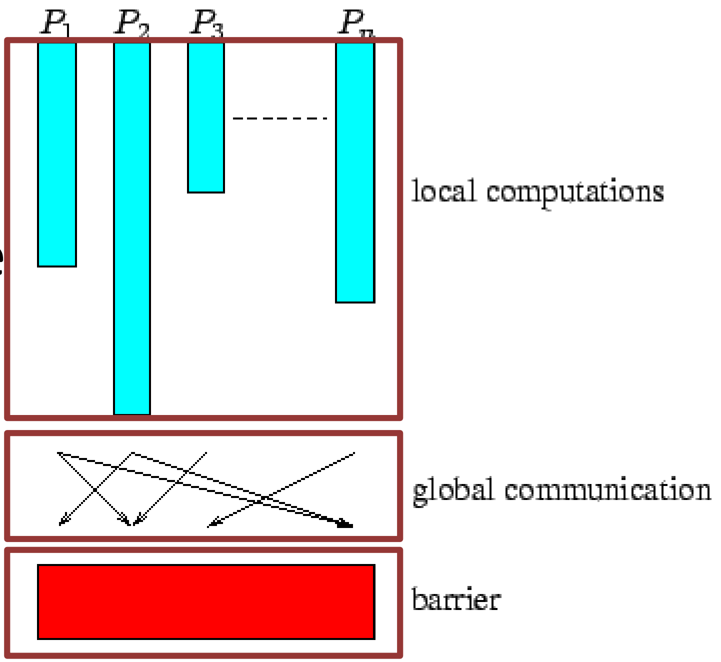
Model

- The high-level organization of Pregel programs is inspired by *Valiant’s Bulk Synchronous Parallel* (BSP) model.
- The synchronicity of this model makes it easier to reason about program semantics when implementing algorithms.
- Pregel programs are inherently free of deadlocks and data races common in asynchronous systems.

BSP Model

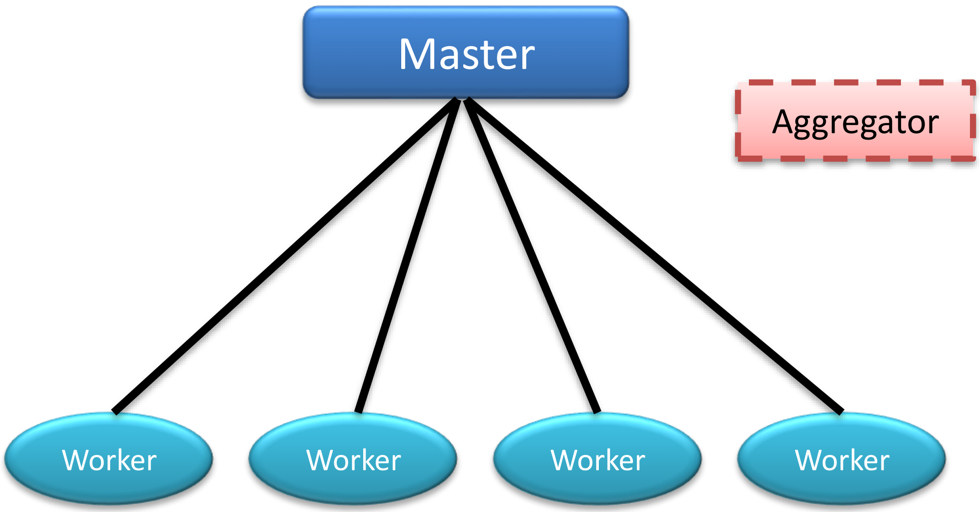
- A BSP computation proceeds in a series of global supersteps.
  1. Local computation
  2. Global communication
  3. Barrier synchronization

1. Run algorithm on each machine
2. Communicate with each other
3. Wait

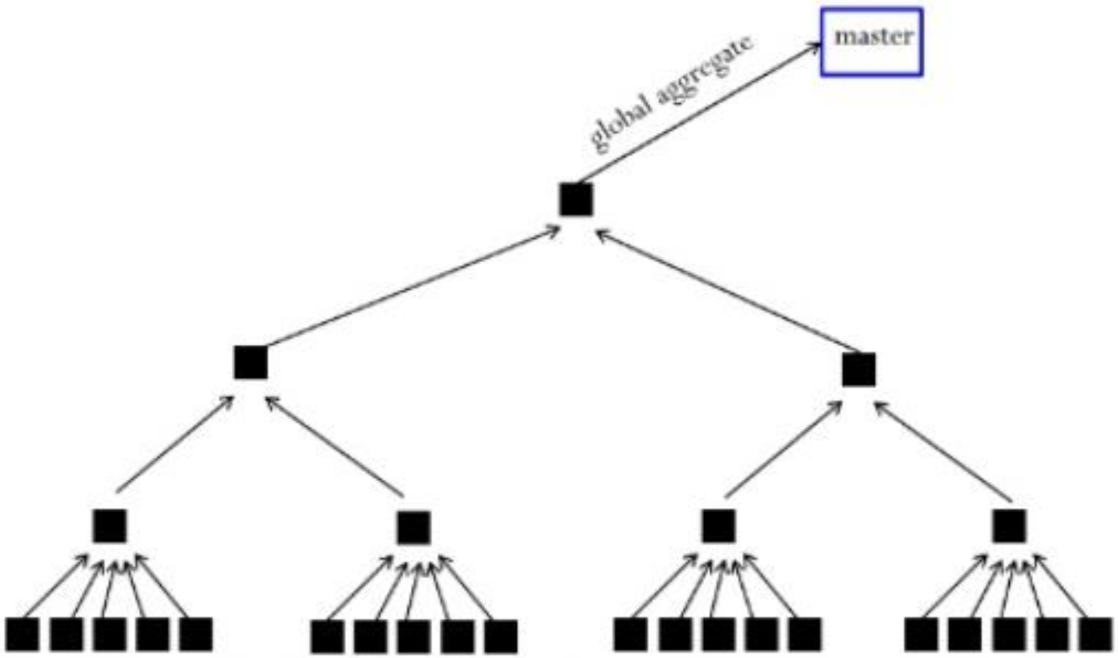


Pregel Model

- The Pregel library divides a graph into partitions, each consisting of a set of vertices and all of those vertices' outgoing edges.
- There are three components in Pregel
  - Master
  - Worker
  - Aggregator



Reduction (Aggregator)



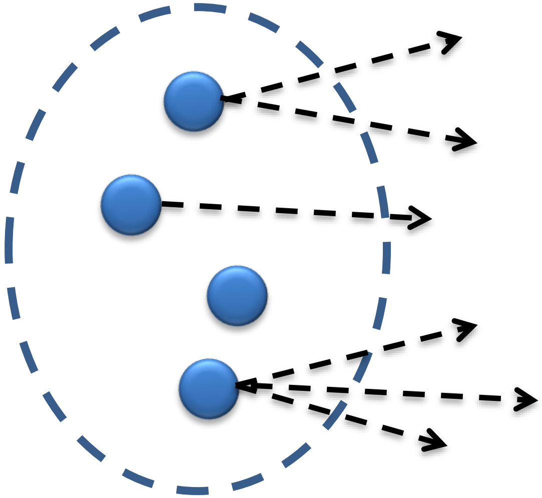
Source: https://wiki.engr.illinois.edu/download/attachments/188588798/pregel.pdf?version=1

Pregel Model

- Master
  - Assign jobs to workers.
  - Receive result from workers.
- Worker
  - Execute jobs from master.
  - Deliver result to master.
- Aggregator
  - A global container that can receive messages from workers.
  - Automatic computation on all the messages according to the user-defined function.

Partition

- In Pregel model, each graph is a directed graph, in which each vertex has a unique id and each edge has a value.
- Graph can be divided into partitions
    - A set of vertices
    - All of these vertices' outgoing edges



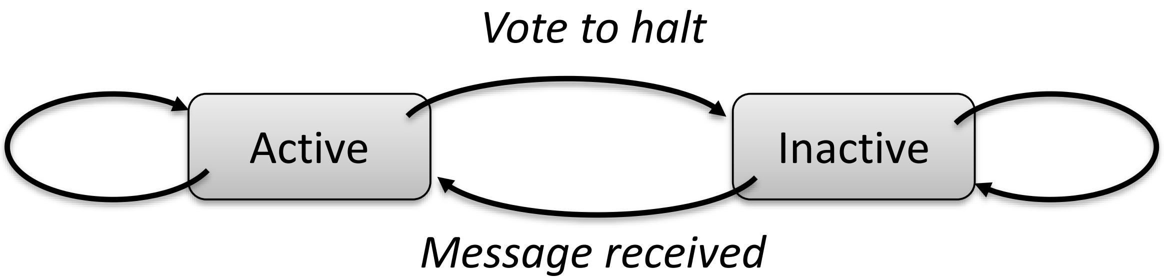
Partition

Partition(cont.)

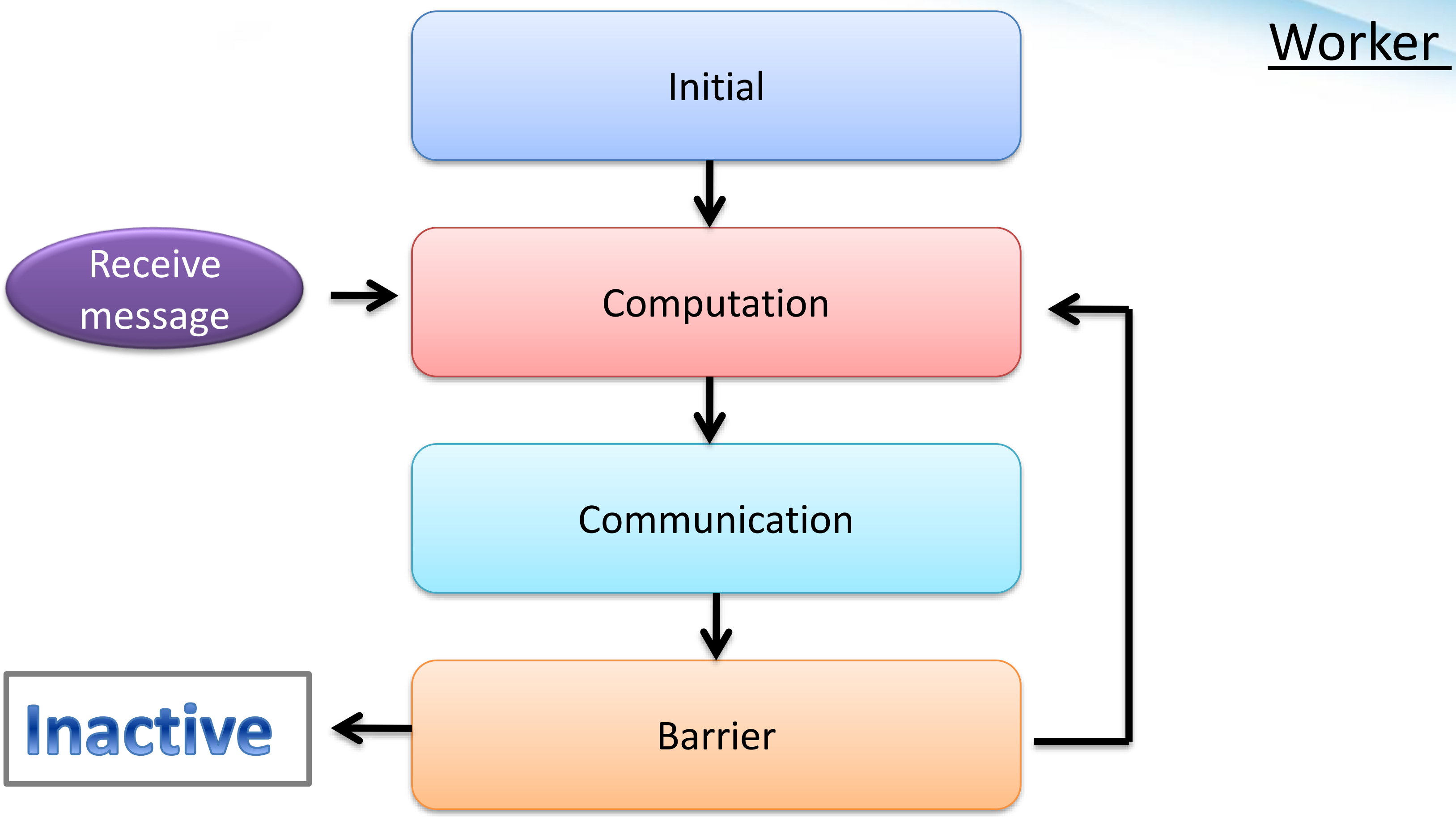
- Pregel provides a default assignment where partition function is *hash*(nodeID) mod *N*, where *N* is the number of partitions, but user can overwrite this assignment algorithm.
- In general, it is a good idea to put close-neighbor nodes into the same partition so that message between these nodes can reduce overhead.

Worker Model

- There are two status types for each vertex
  - Active
  - Inactive
- The algorithm as a whole terminates when all vertices are simultaneously inactive and there are no messages in transit.
- Every vertex is in the active state in superstep 0.



Worker Model



Model

Implement

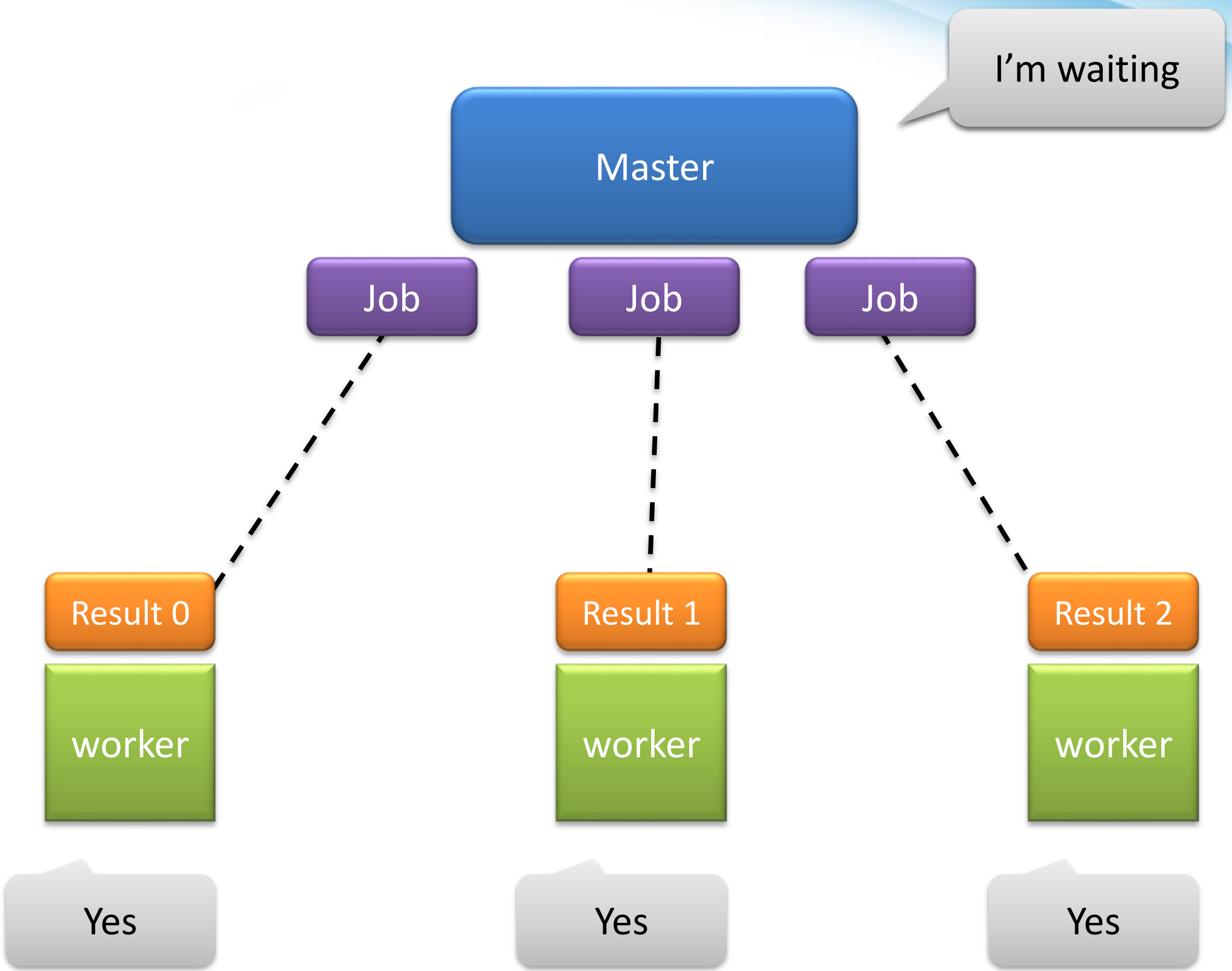
Communication

MODEL

Master

- The master is primarily responsible for coordinating the activities of workers.
- Master sends the same request to every worker that was known to be alive at beginning, and waits for a response from every worker.
- If any worker fails, the master enters recovery mode.

Master

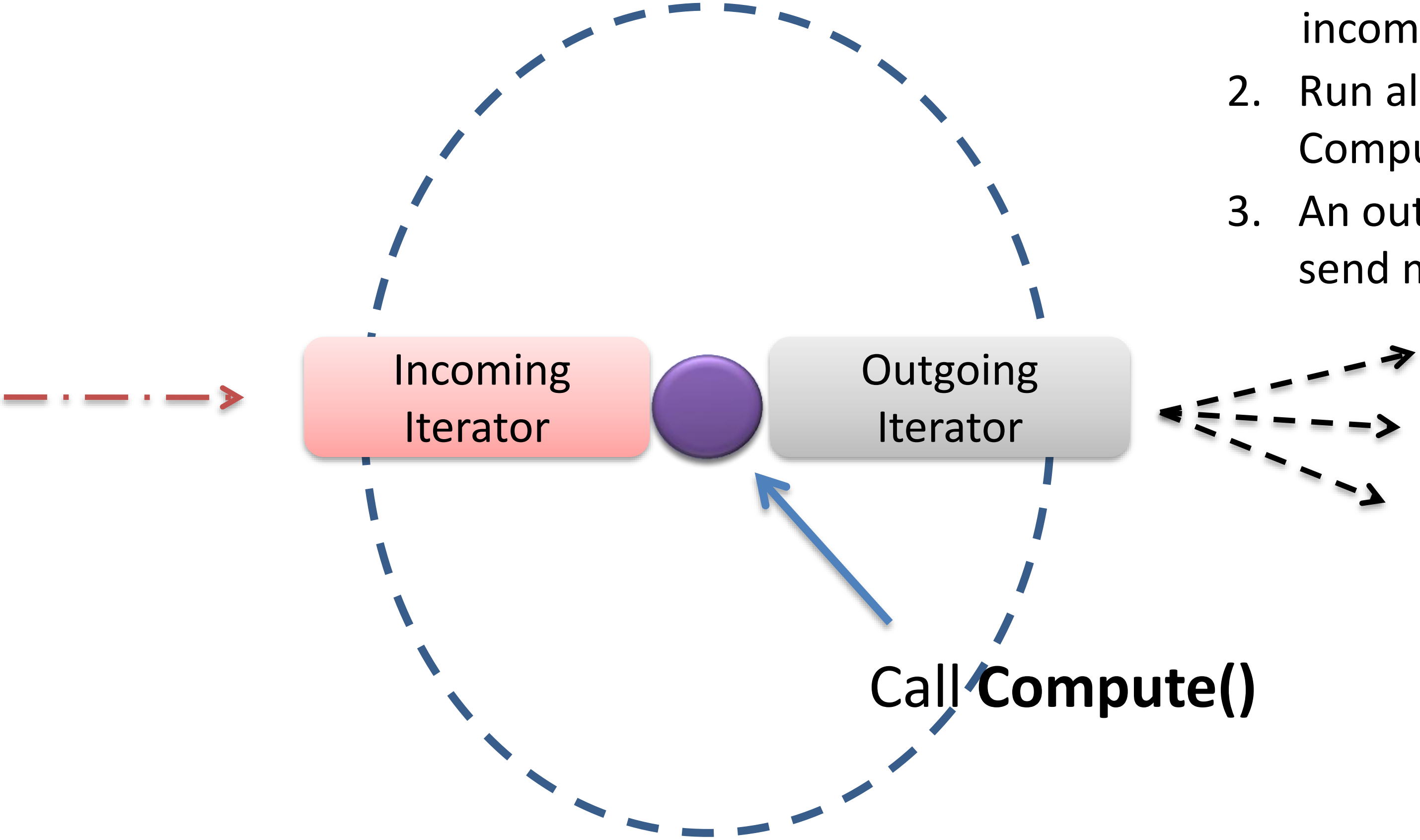

I’m waiting
Master
Job
Job
Job
Result 0
Result 1
Result 2
worker
worker
worker
Yes
Yes
Yes

Worker

- A worker machine maintains the state of its portion of the graph in memory.
- Worker performs a superstep that loops through all vertices and calls **Compute()**.
  - During a superstep the framework invokes a user-defined function for each vertex, conceptually in parallel. The function specifies behavior at a single vertex V and a single superstep S: receive, compute, send out
- Worker has no access to incoming edges because each incoming edge is part of a list owned by the source vertex.

Worker

1. An incoming iterator to the incoming message.
2. Run algorithm by calling Compute()
3. An outgoing iterator to send message



Aggregators

- An aggregator computes a single global value by applying an aggregation function to a set of values that the user supplies.
- Worker combines all of the values supplied to an aggregator instance when executes a superstep.
- An aggregator is partially reduced over all of the worker's vertices in the partition.
- At the end of superstep workers form a tree to reduce partially reduced aggregator into global values and deliver them to the master.

Failure Recover

- Worker failure are detected using regular ‘ping’ messages that master issues to workers.
- If a worker does not receive a ping message after a special interval, the worker process terminates.
- If the master does not hear back from a worker, the master marks the worker process as failed.

Failure Recover (cont.)

- If one or more workers fail, the master reassigns graph partitions, these workers performed, to the currently available set of workers.
- Workers reload their partition state from the most recent available checkpoint at the beginning of a superstep.

Model

Implement

Communication

MODEL

Communication

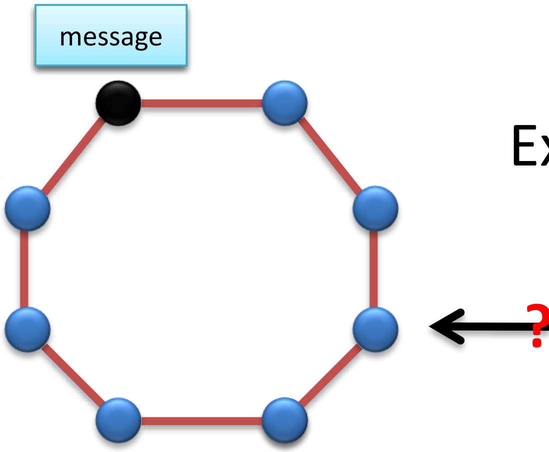
- Vertices communicate directly with one another by sending message.
- In Pregel, there are many virtual functions that can be overridden by programmer.
  - Compute — Algorithm
  - Combiners
  - Aggregators — Communication for some purpose

Communication

- A vertex can send any number of messages in a superstep.
- All messages sent to vertex V in superstep S are available, via an iterator, but not guaranteed order of messages in the iterator.
- Vertex V sent message to destination vertex, which may not be a neighbor of V.

Communication(cont.)

- A vertex could learn the identifier of a non-neighbor from a message received earlier, or could be known implicitly.
- When destination vertex does not exist, pregel executes user-defined handles, like create the missing vertex or remove the dangling edge.



Execute exception handle

1. Add vertex
2. Remove edge

Combiners

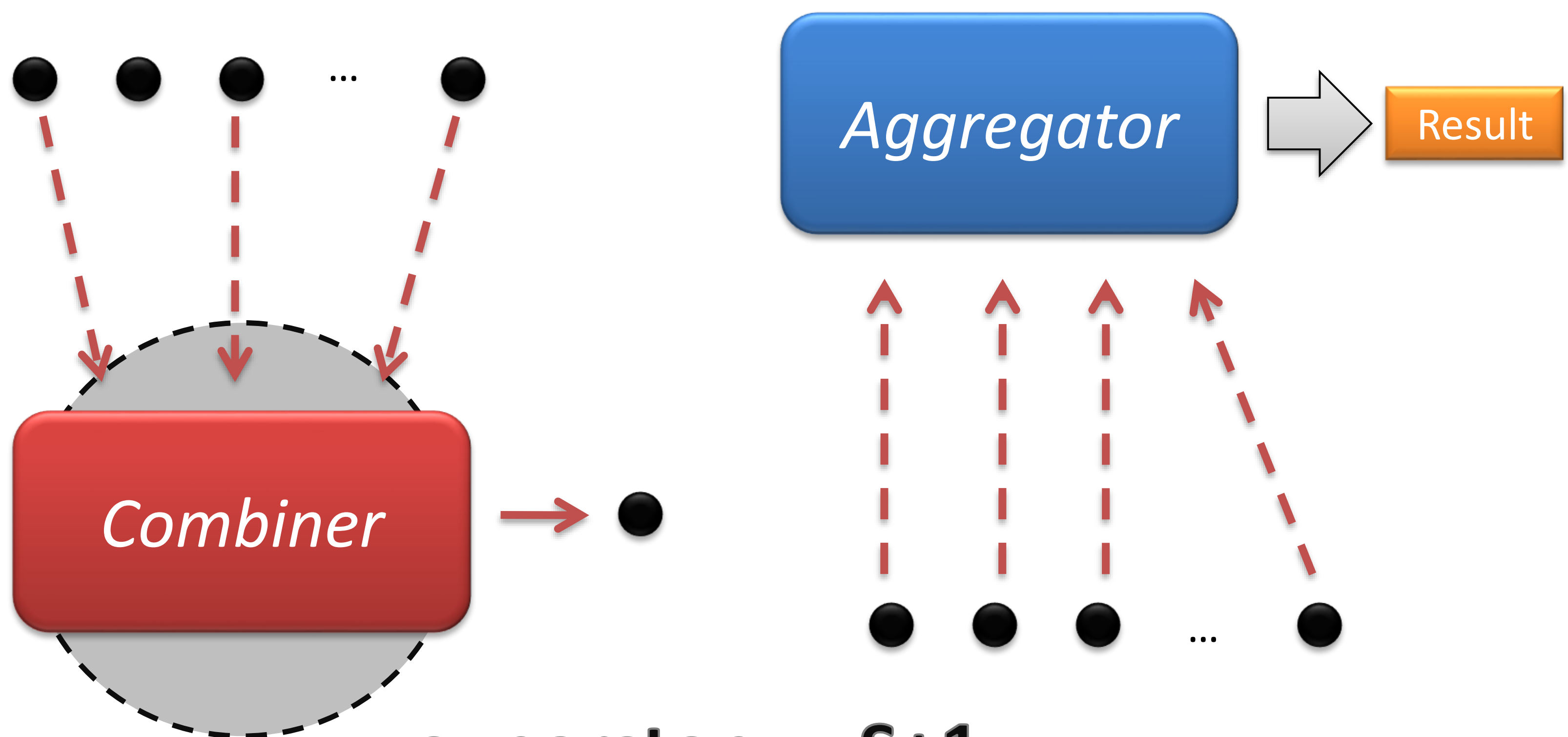
- Combiners can combine several messages into a single message.
- Combiners are not enabled by a default, because there is no mechanical way to find a useful combining function that is consistent.
- Combiners do not guarantee about which messages are combined, the groupings presented to the combiner, or the order of combining.
  - Combiner should only be enabled for commutative and associative operator.

Aggregators

- Pregel aggregators are a mechanism for global communication, monitoring, and data.
- Each vertex can provide a value to an aggregator in superstep S, the system combines those values using a reduction operator, and the resulting value is make available to all vertices in superstep S+1.
    - Minimum
    - Summary
    - …etc

Communication(cont.)

- Sending a message, especially to a vertex on another machine, incurs some overhead.



superstep:= S+1

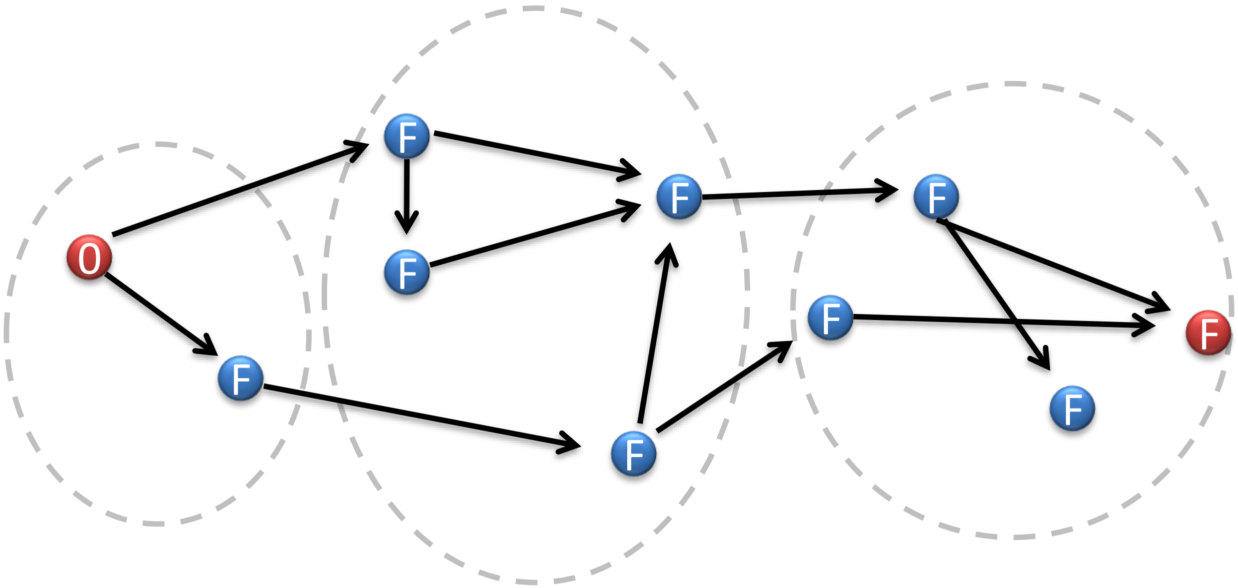
Shortest Paths –

The shortest path problem is the best well-know problem in graph theory

SAMPLE CASE

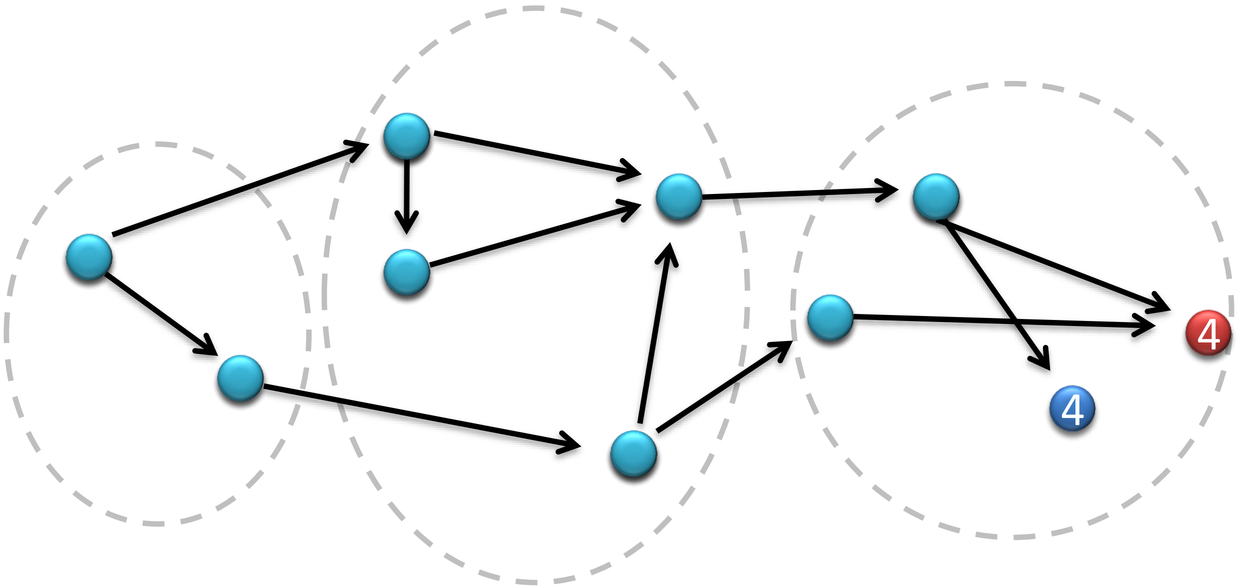
Shortest Paths

- Phase 0
  - Assume the value associated with each vertex is initialized to **INF** (a constant larger than any distance in the graph).
  - Only the source vertex updates its value (from INT to 0).



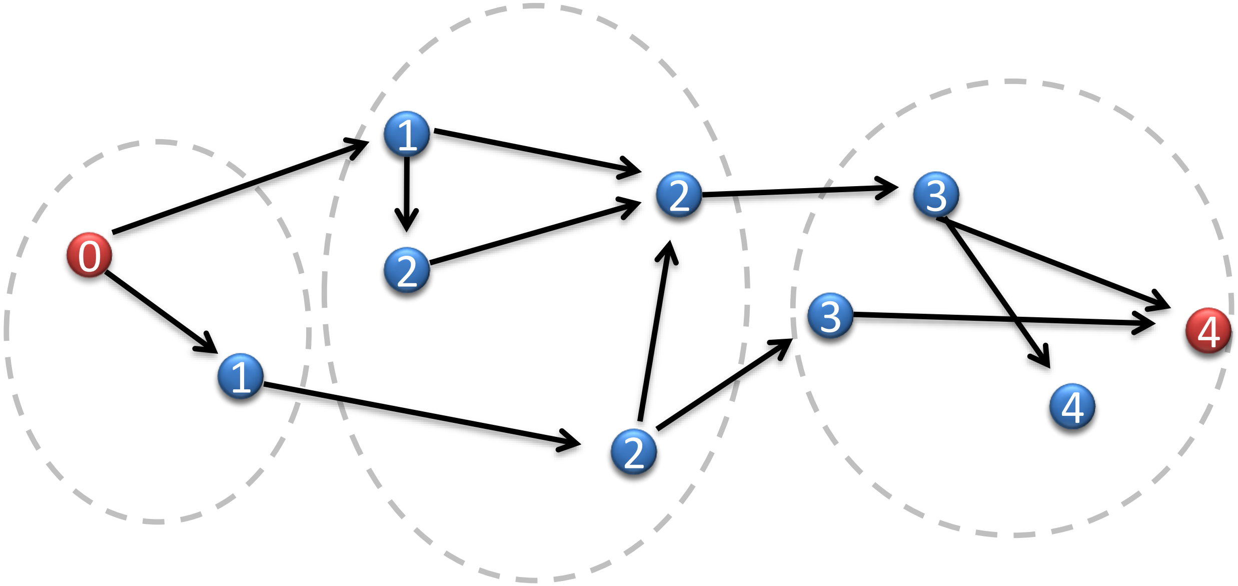
Shortest Paths

- Phase 1
  - For each updated vertex, send its value to neighbors.
  - For each vertex which received one or more messages, update its value to the minimal value in these messages and its value.



Shortest Paths

- Phase 2
  - The algorithm is terminated when no more updates occur.



Summary of Pregel

- Pregel is a model suitable for large-scale graph computing
  - Quality
  - Scalability
  - Fault tolerance
- User switches to the ‘think like a vertex’ mode of programming
  - Designed for sparse graphs where communication occurs mainly over edges.
  - Its performance will suffer when most vertices continuously send messages to most other vertices. Realistic dense graphs are rare.
- Some graph algorithm can be transformed into more Pregel-friendly variants.

Summary

- Scalability
  - Provide the capability of processing very large amounts of data.
- Availability
  - Provide the ability of failure tolerance on machine failure.
- Manageability
  - Provide mechanism for the system to automatically monitor itself and manage the complex job transparently for users.
- Performance
  - Good enough than extra passes over the data.

References

- Jeffrey Dean and Sanjay Ghemawat. "MapReduce: Simplied Data Processing on Large Clusters, " *OSDI* 2004 .
- Grzegorz Malewicz , Matthew H. Austern , Aart J.C. Bik , James C. Dehnert , Ilan Horn , Naty Leiser , Grzegorz Czajkowski. "Pregel: a system for large-scale graph processing," Proceedings of the 28th ACM symposium on Principles of distributed computing, (August 10-12, 2009)
- Hadoop.
  - http://hadoop.apache.org/
- NCHC Cloud Computing Research Group.
  - http://trac.nchc.org.tw/cloud
- Jimmy Lin's course website.
  - http://www.umiacs.umd.edu/~jimmylin/

- https://www.youtube.com/watch?v=X8z_MOU5N00
- 9 12 12 PageRank in MapReduce and Pregel 10 42
- From 4:10