Pregel In Graphs

Models & Instances

Chase Zhang

Strikingly

March 16, 2018

Table of Contents

Computing Model

What is Pregel?

- Published by Google at 2009[1]
- Distributed graph computing system at large scale
- Implemented by open source solutions like Spark's GraphX[2]

[1] https://kowshik.github.io/JPregel/pregel_paper.pdf
[2] https://spark.apache.org/graphx/

Computing Model

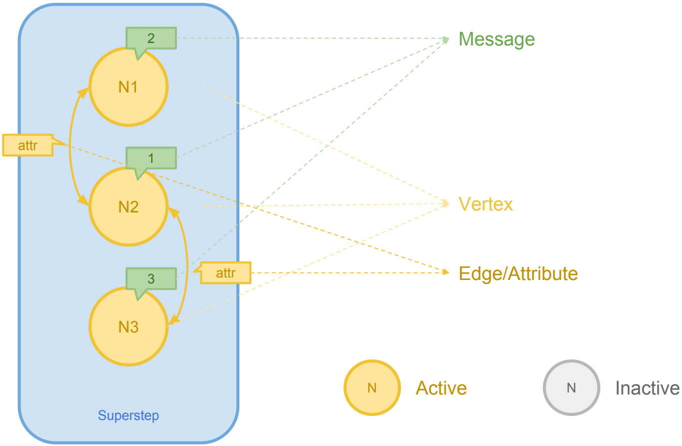

Figure: Basic Model

Computing Model

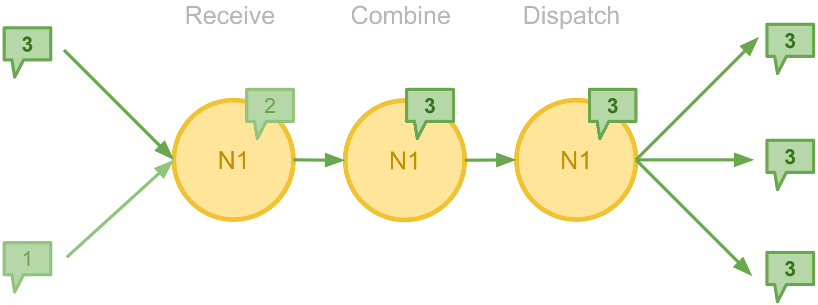

Figure: Pregel is vertex oriented

Computing Model

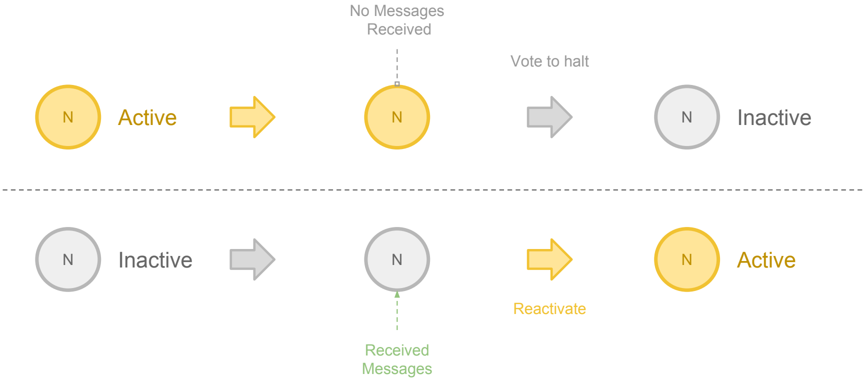

Figure: Vertice have states

Computing Model

- Pregel is a vertex oriented computing model.
  - Map runs upon each vertex (other than edges)
  - Each vertex only knows its out-going edges during each superstep
- Pregel depends on message sending to iteratively computing the result
  - Each vertex receives messages from prior step
  - After one step, vertices send messages to adjacent vertices
- Vertices have states
  - Only active vertices will send message to the others
  - Program terminates once there is no active vertices

Computing Model

Problem

Sending message through network is costly.

Computing Model

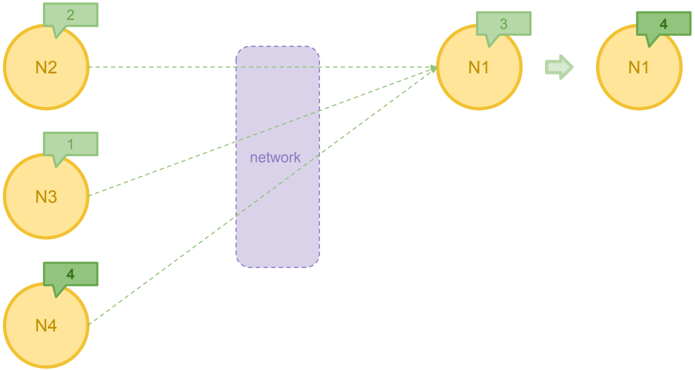

Figure: Passing messages

Computing Model

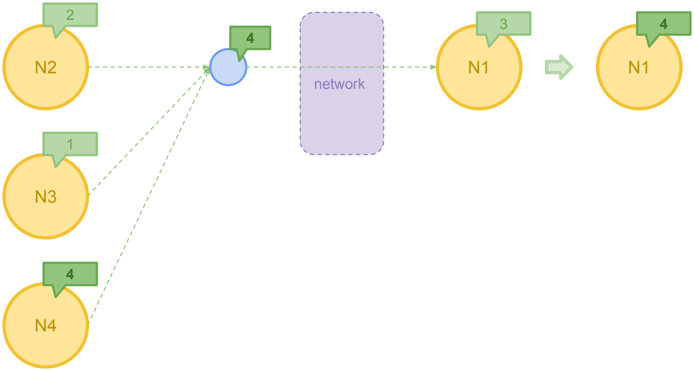

Figure: Solution: Combiner

Computing Model

Problem

Some iterative graph algrithms require graph-wide results and metrics.

Computing Model

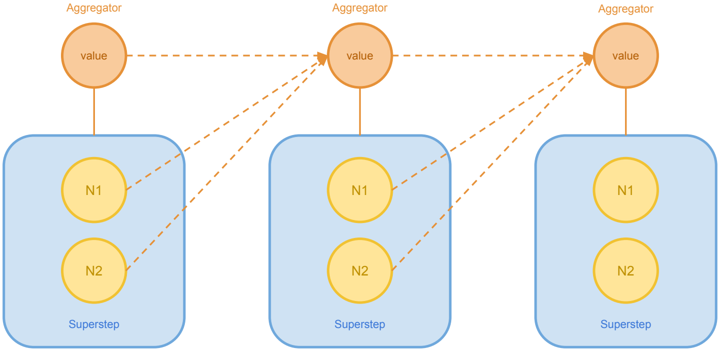

Figure: Solution: Aggregator

Computing Model

Problem

Some graph algorithms require modifying topology during iteration.

Computing Model

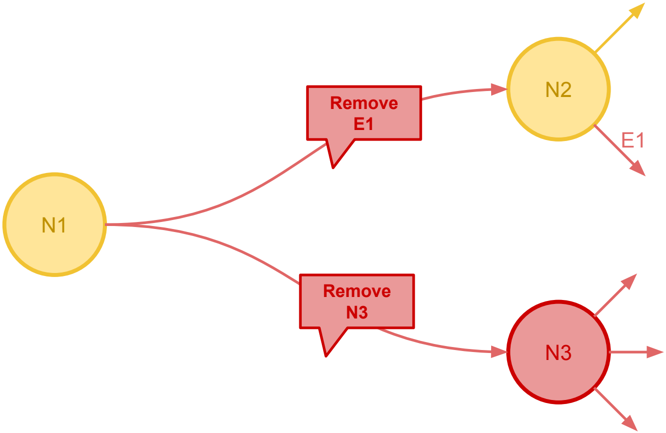

Figure: Solution: Sending modification messages

Computing Model

- Pregel provides combiner for reducing network traffic
- Pregel provides aggregator for recording graph-wide values and metrics
- Vertices in Pregel can send topology modification requests as messages to target vertices

Examples

Problem

Find connected components of a directed graph.

Examples

Solution

(Single machine) Union-Find[3].

[3] https://en.wikipedia.org/wiki/Disjoint-set_data_structure

Examples

Solution

(Pregel) Emit max(min) message received until no active vertex.

Examples

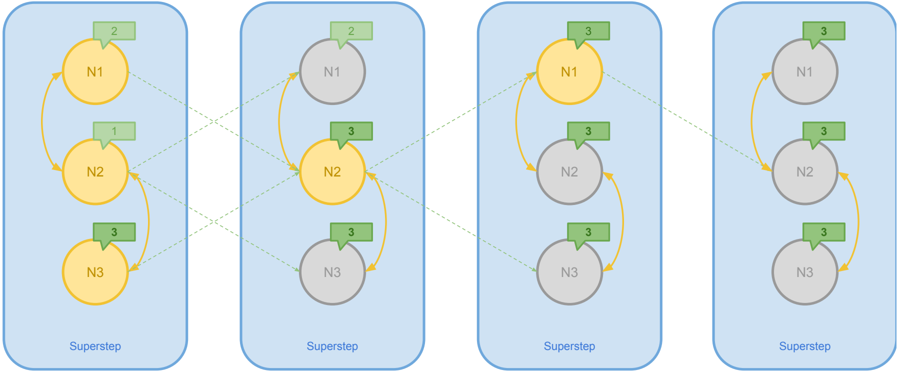

Figure: Connected Components in Pregel

Examples

Problem

Find shortest path from a single source to the other vertices.

Examples

Solution

(Single machine) Dijkstra[4], Bellman-Ford[5], A*[6].

[4] https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
[5] https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm
[6] https://en.wikipedia.org/wiki/A*_search_algorithm

Examples

Solution

(Pregel) BFS, emitting current shortest cost from source vertex to current vertex plus edge cost.

Examples

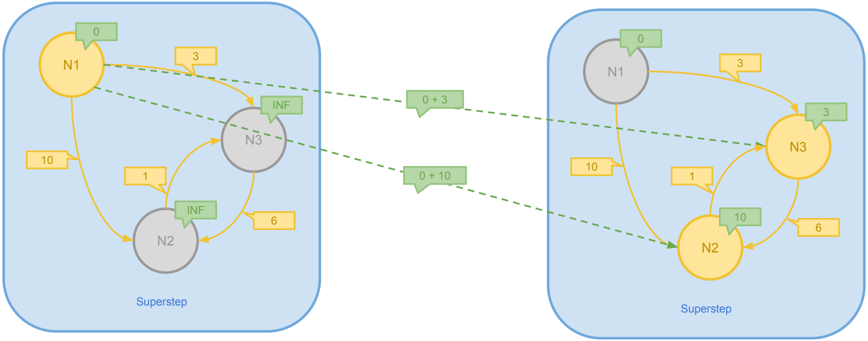

Figure: Shortest Path: Step 1

Examples

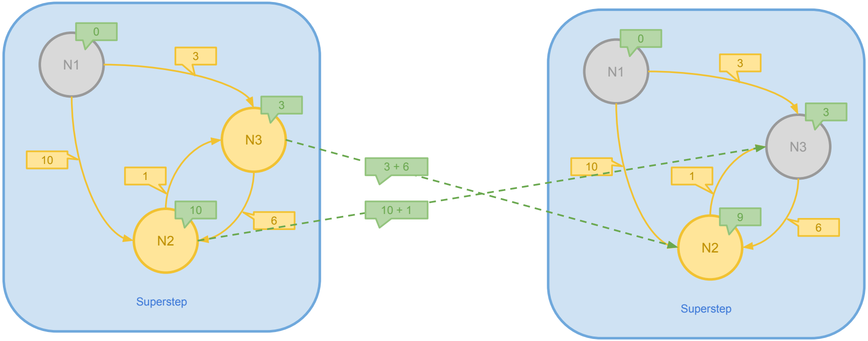

Figure: Shortest Path: Step 2

Examples

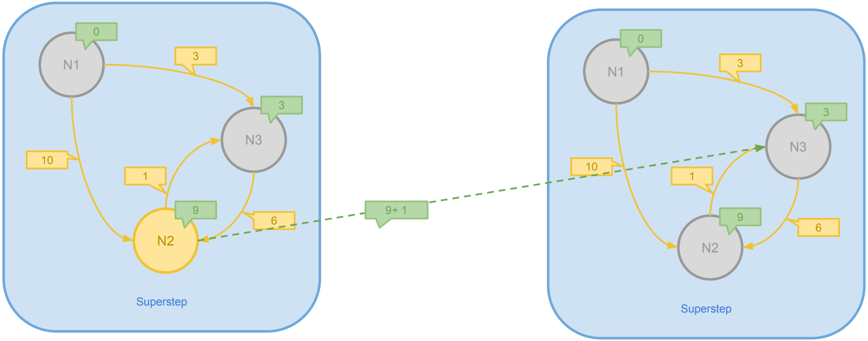

Figure: Shortest Path: Step 3

Examples

Problem

PageRank[7]

[7] https://en.wikipedia.org/wiki/PageRank

Examples

$\lvert V \rvert$	vertices number
$\lvert E(v) \rvert$	edges number of vertex v
$val(v)$	current message value of v
$sum(v)$	sum of messages received of v in last superstep

- Initial messages: $\frac{1}{\lvert V \rvert}$
- Emit messages: $\frac{val(v)}{\lvert E(v) \rvert}$
- Update value: $sum(v) \cdot 0.85 + \frac{0.15}{\lvert V \rvert}$

Examples

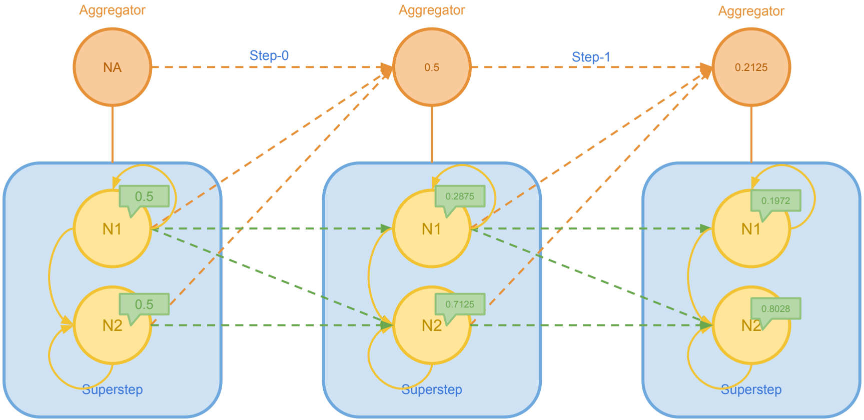

Figure: PageRank

Examples

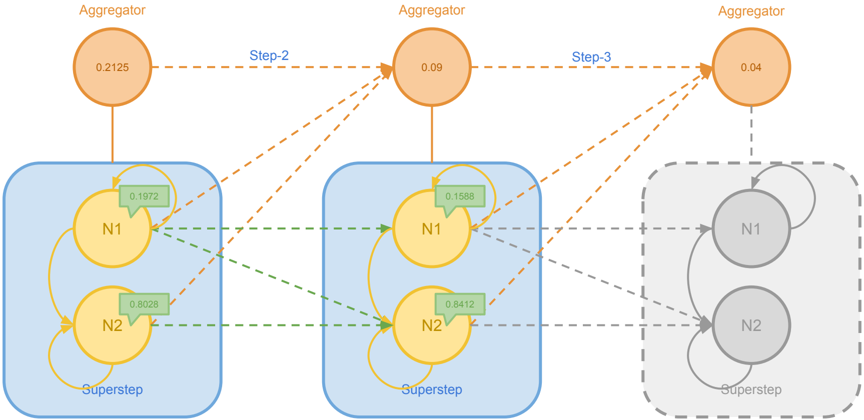

Figure: PageRank

System Design

- Graph partitioning
- Master-worker system model
- Message buffer for better performance
- Checkpoint and confined recovery

System Design

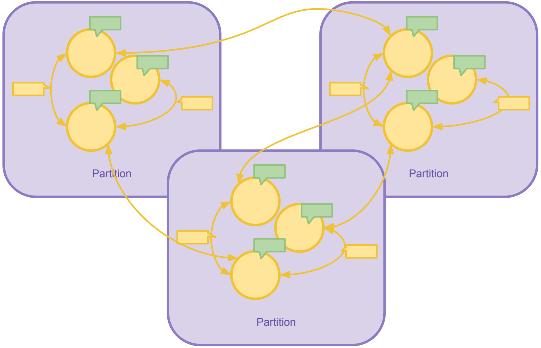

Figure: Graph Partitioning

System Design

- Graph is partitioned by $\text{hash}(\text{vertexId})$ by default
- User partitioning function can be provided for locality

System Design

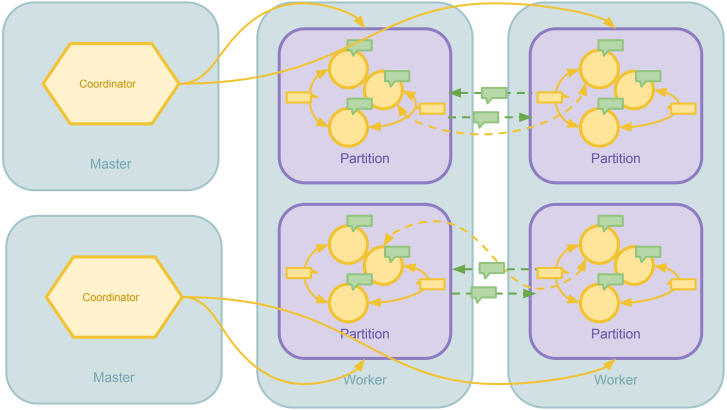

Figure: Master-Worker Model

System Design

- Master coodinates supersteps, it holds no partition of graph
- Master calculate and hold values of aggregators
- Master send RPC to every participated worker and wait for task to finish
- Workers hold partitions of graph. Given a vertexId, a worker can know which worker is holding it without querying master

System Design

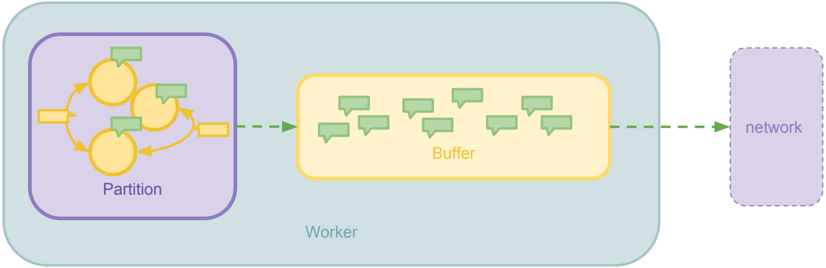

Figure: Message Buffer

System Design

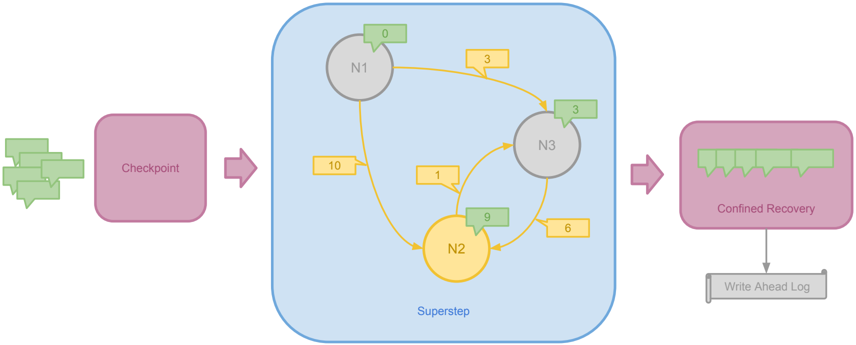

Figure: Checkpoint & Confined Recovery

System Design

- A worker buffers message locally and send them as a batch as to reduce network overhead
- Before and after a superstep, a worker checkpoint and perform confined recovery
- Confined recovery logs every outgoing messages, so that only lost partitions need to be recalculated

Instance: User Tracking System

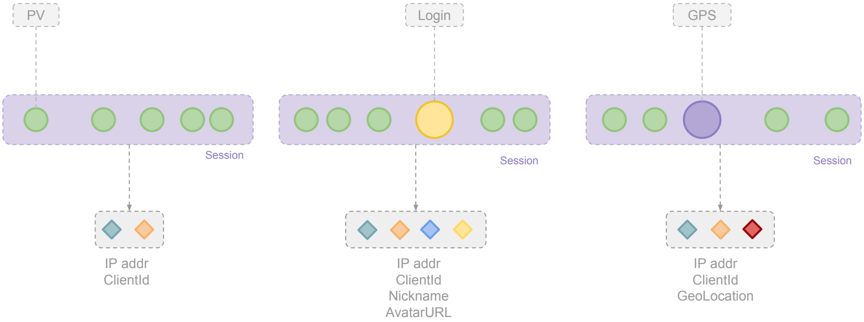

Figure: User Tracking Problem

Instance: User Tracking System

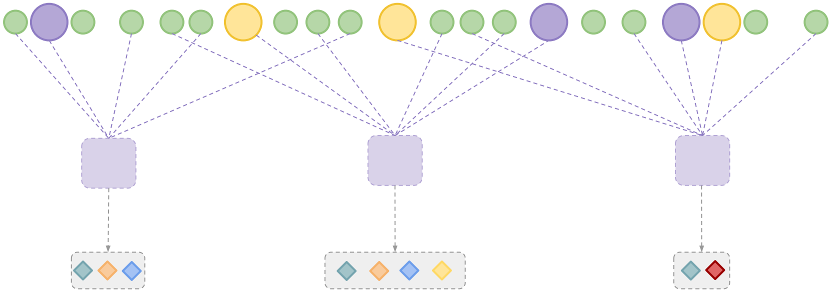

Figure: Sessionize Events & Feature Extraction

Instance: User Tracking System

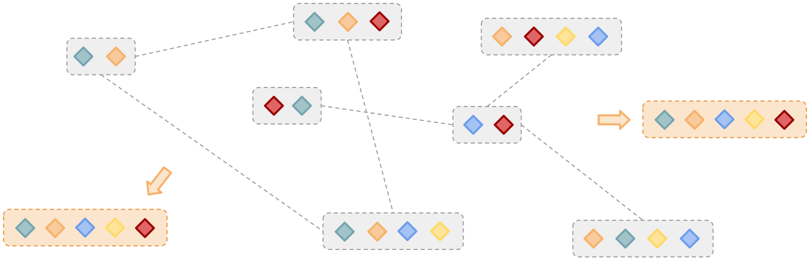

Figure: Find Connected Components & Get User Profile

Instance: User Tracking System

Problem

Calculating edges betwen sessions cost $O(N^2)$ time if we compare each pair. It will be too enormous even for just a million(10^6) sessions.

Instance: User Tracking System

Solution

Five significant features are selected for matching, and we define two sessions are matched if more than two features matched. So we can:

- Enumerate combinations of the five significant features, it will be $\mathrm{C}_5^2 = 10$
- Applying sort or hash method to distribute sessions into matching buckets
- Connect sessions in a same bucket

Instance: User Tracking System

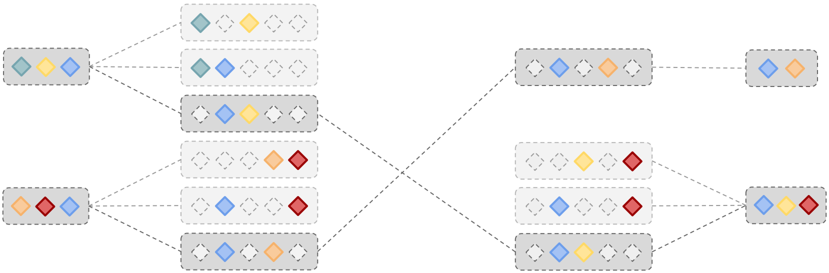

Figure: Match Features

Instance: User Tracking System

Claim

This optimization (using sort method) reduced time cost.
At first, the total elements to match upon each other will increasee to $\mathsf{O}(\mathsf{N} \times 10)$. To sort all elements cost $\mathsf{O}(\log \mathsf{N} \times 10)$. Connect elements in a bucket cost $\mathsf{O}(\mathsf{N} \times 10)$ in total.
The final cost will be

$$\mathsf{O}(\log \mathsf{N} \times 10) \simeq \mathsf{O}(\log \mathsf{N}) \preceq \mathsf{O}(\mathsf{N}^2)$$

Thank you!