Intro to Pregel

Justus Hibshman
9/27/18

Background

Pregel was developed by Google in 2010 as a system to speed up their graph computations.

Original Paper: “Pregel: A System for Large Scale Graph Processing”

https://dl.acm.org/citation.cfm?id=1807184

Some open source versions:

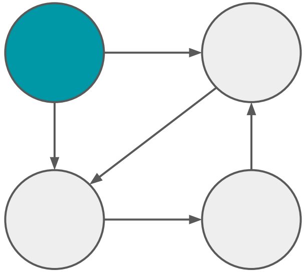
Apache Giraphe: http://giraph.apache.org/

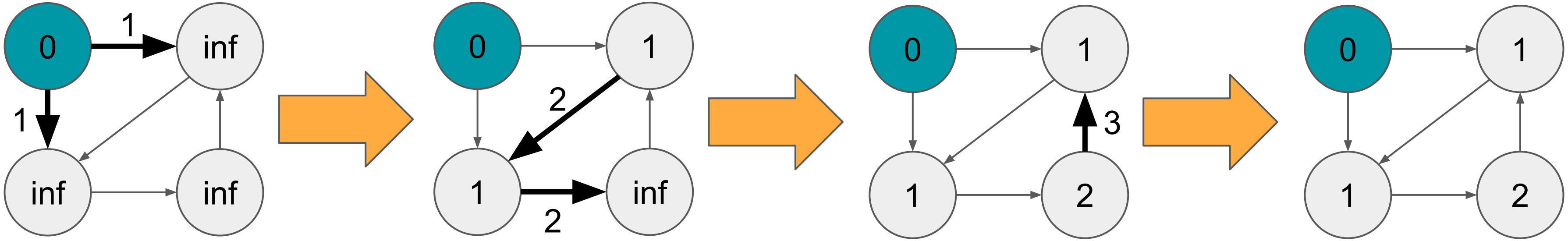
Phoebus: https://github.com/xslogic/phoebus

Basic Idea: Perform Computation “At” Vertices

- Graphs are directed.
- All vertices have a function computed for them in a “superstep.”
- Vertices can pass messages to each other to be used in the following superstep - only have explicit knowledge of their outgoing edges.
- Computation stops when all vertices signal that they’re done.

Example: Finding Distance from Start Node





Some More Details

- A superstep happens (conceptually) in parallel over the nodes.
- The function that operates per-node is the same for every node.

- Nodes may send any number of messages in a given superstep.
- Nodes can sent messages to any node provided they have that node’s id.
- Typically nodes just send messages via outgoing edges.
- Order of message reception is undefined.

- Once vertices vote to halt they don’t do any computation until they “awaken” by receiving another message.

Use

```
Template <typename VertexValue, typename EdgeValue, typename MessageValue>
class Vertex {
 public:
  virtual void Compute(MessageIterator* msgs) = 0;
  const string& vertex_id() const;
  int64 superstep() const;
  const VertexValue& GetValue();
  VertexValue* MutableValue();
  OutEdgeIterator GetOutEdgeIterator();
  void SendMessageTo(const string& dest_vertex, const MessageValue& message);
  void VoteToHalt();
};
```

Additional Features

Combiners

- Used to improve performance
- Collapse multiple messages into one (e.g. take a sum of integer messages)
- No guarantees about which messages will be combined
- No guarantees about what order they’ll be combined in

Additional Features

Aggregators

- Used for “global communication, monitoring, and data”
- Nodes can provide a single value to an aggregator at each superstep.
- Values are combined via a “reduction operator.”
- The Result of superstep S’s aggregation is accessible in superstep S+1.

Additional Features

Topology Mutations

- Nodes can request the creation/removal of edges and nodes.
- Node removals precede node additions, which precede...
- Users define handlers for conflicting requests, such as multiple additions of the same node.

Simplistic PageRank Implementation

```
class PageRankVertex : public Vertex<double, void, double> {
 public:
  virtual void Compute(MessageIterator* msgs) {
    if (superstep() >= 1) {
      double sum = 0;
      for (; !msgs->Done(); msgs->Next())
        sum += msgs->Value();
      *MutableValue() = 0.15 / NumVertices() + 0.85 * sum;
    }
    if (superstep() < 30) {  // In practice would use an aggregator to detect convergence.
      const int64 n = GetOutIterator().size();
      SendMessageToAllNeighbors(GetValue() / n);
    } else {
      VoteToHalt();
    }
  }
};
```