A comparison of graph processing systems:
A deeper look & Learnings so far
Overview

Graph Computing

Frameworks and APIs

- Pregel
- Apache Giraph
- Apache GraphX
- Pregelix

Project Status
Graph Computing - Why?

Because sometimes... Relational isn’t enough.
Graph Computing - How?

There are three categories of software present:

- Graph Databases
Graph Computing - How?

There are three categories of software present:

- Graph Analysis
Graph Computing - How?

- Bridge Frameworks
  - Apache Tinkerpop

```
graph = TitanFactory.build().set('storage.backend', 'inmemory').open()
g = graph.traversal()

cvd = graph.addVertex(label, "actor", "name", "jean claude van damme")
kick = graph.addVertex(label, "movie", "name", "Kickboxer", "year", 1989)
blood = graph.addVertex(label, "movie", "name", "Bloodsport", "year", 1988)

cvd.addEdge("acted_in", kick)
cvd.addEdge("acted_in", blood)
```

The Gremlin Query Language used by Apache Tinkerpop:
https://academy.datastax.com/resources/getting-started-graph-databases

```
to other vertices g.V().has("name", "jean claude van damme").out()

to edges g.V().has("name", "jean claude van damme").outE()

filtering with traversals g.V().has("name", "jean claude van damme").out().has("year", between(1980, 1990))
```
Overview

Graph Computing

Frameworks and APIs

- Pregel
- Apache Giraph
- Apache GraphX
- Pregelix

Project Status
Pregel

● The first dedicated system for iterative graph algorithms.
● Uses the Bulk-Synchronous-Parallel approach by Valiant et al.
● “Think like a vertex.”
● API:
  Compute()
  Combiners
  Aggregators
  Mutations
  I/O

```cpp
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();
};
```

Figure 3: The Vertex API foundations.
Apache Giraph - PageRank

```java
public class MyPageRankComputation extends BasicComputation<
    LongWritable, DoubleWritable, FloatWritable, DoubleWritable> {

    public static final int MAX_SUPERSTEPS = 2;

    @Override
    public void compute(Vertex<LongWritable, DoubleWritable, FloatWritable>
        iterable<DoubleWritable> messages) throws IOException {
        if (getSuperstep() >= 1) {
            double sum = 0;
            for (DoubleWritable message : messages) {
                sum += message.get();
            }
            vertex.setValue(new DoubleWritable(sum));
        }

        if (getSuperstep() < MAX_SUPERSTEPS) {
            int numEdges = vertex.getNumEdges();
            DoubleWritable message = new DoubleWritable(vertex.getValue().get
                for (Edge<LongWritable, FloatWritable> edge: vertex.getEdges()) {
                    sendMessage(edge.getTargetVertexId(), message);
                }
            //send message to all edges
            vertex.voteToHalt();
        }
    }
}
```

```
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) {
            const int64 n = GetOutEdgeIterator().size();
            SendMessageToAllNeighbors(GetValue() / n);
        } else {
            VoteToHalt();
        }
    }
```

Figure 4: PageRank implemented in Pregel.

From original Pregel paper by Malewicz et al., SIGMOD ’10
GraphX

- Extends the Spark RDD by introducing a new Graph abstraction.
- Adds more functionality to the only-graph-analysis frameworks.

Source: http://spark.apache.org/graphx/
import org.apache.spark.graphx.GraphLoader

// Load the edges as a graph
val graph = GraphLoader.edgeListFile(sc, "data/graphx/followers.txt")

// Run PageRank
val ranks = graph.pageRank(0.0001).vertices

// Join the ranks with the usernames
val users = sc.textFile("data/graphx/users.txt").map { line =>
  val fields = line.split("\,"
  (fields(0).toLong, fields(1))
}
val ranksByUsername = users.join(ranks).map {
  case (id, (username, rank)) => (username, rank)
}

// Print the result
println(ranksByUsername.collect().mkString("\n"))

Source: http://spark.apache.org/graphx/
Pregelix

- Uses a set-oriented, dataflow approach to implement BSP model.
- Treat messages and vertex states like tuples in a schema.
- Message exchanges now are the joins of schemas.
- Uses existing distributed dataflow execution engines: Hyracks

<table>
<thead>
<tr>
<th>Relation</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertex</td>
<td>(vid, halt, value, edges)</td>
</tr>
<tr>
<td>Msg</td>
<td>(vid, payload)</td>
</tr>
<tr>
<td>GS</td>
<td>(halt, aggregate, superstep)</td>
</tr>
</tbody>
</table>

Source: author's slides
The PageRank algorithm, with uniform transition probabilities on the edges http://en.wikipedia.org/wiki/PageRank

```java
package edu.uci.ics.pregelix.benchmark.vertex;

import org.apache.giraph.edge.Edge;
import org.apache.giraph.examples.RandomWalkVertex;
import org.apache.giraph.utils.MathUtils;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;

public class PageRankVertex extends RandomWalkVertex<LongWritable, NullWritable> {

    @Override
    protected double transitionProbability(double stateProbability, Edge<LongWritable, NullWritable> edge) {
        return stateProbability / getNumEdges();
    }

    @Override
    protected double recompute(Iterable<DoubleWritable> partialRanks, double teleportationProbability) {
        // rank contribution from incident neighbors
        double rankFromNeighbors = MathUtils.sum(partialRanks);
        // rank contribution from dangling vertices
        double danglingContribution = getDanglingProbability() / getTotalNumVertices();
        // recompute rank
        return (1d - teleportationProbability) * (rankFromNeighbors + danglingContribution) + teleportationProbability / getTotalNumVertices();
    }
}
```
Overview

Graph Computing

Frameworks and APIs

- Pregel
- Apache Giraph
- Apache GraphX
- Pregelix

Project Status
Benchmarks

**Single machine tests**

Tentative Datasets:
- Wikipedia citations (~300 MB)
- Web graph of Berkeley and Stanford (7600595 edges, 685230 nodes)

Tentative Algorithms:
- PageRank
- Triangle Count/SSSP
- Strongly Connected Components

**Cluster tests**

Tentative Datasets:
- Wikipedia Authorship (1.67 GB)
- Web graph of Berkeley and Stanford
- Twitter (7 GB)

Tentative Algorithms:
- PageRank
- Triangle Count/SSSP
- Strongly Connected Components


An experimental comparison of pregel-like graph processing systems, Han et al., VLDB, August 2014
Stanford Large Network Dataset Collection

- Social networks: online social networks, edges represent interactions between people
- Networks with ground-truth communities: ground-truth network communities in social and information networks
- Communication networks: email communication networks with edges representing communication
- Citation networks: nodes represent papers, edges represent citations
- Collaboration networks: nodes represent scientists, edges represent collaborations (co-authoring a paper)
- Web graphs: nodes represent webpages and edges are hyperlinks
- Amazon networks: nodes represent products and edges link commonly co-purchased products
- Internet networks: nodes represent computers and edges communication
- Road networks: nodes represent intersections and edges roads connecting the intersections
- Autonomous systems: graphs of the internet
- Signed networks: networks with positive and negative edges (friend/foe, trust/distrust)
- Location-based online social networks: Social networks with geographic check-ins
- Wikipedia networks, articles, and metadata: Talk, editing, voting, and article data from Wikipedia
- Twitter and Memetracker: Memetracker phrases, links and 467 million Tweets
- Online communities: Data from online communities such as Reddit and Flickr
- Online reviews: Data from online review systems such as BeerAdvocate and Amazon

SNAP networks are also available from UF Sparse Matrix collection. Visualizations of SNAP networks by Tim Davis.
Thank You :)