Data-Intensive Distributed Computing
CS 431/631 451/651 (Winter 2021)

Part 10a: Analyzing Graphs, Redux

Ali Abedi

These slides are available at https://www.student.cs.uwaterloo.ca/~cs451

This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States
See http://creativecommons.org/licenses/by-nc-sa/3.0/us/ for details
Graph Algorithms, again?
(srsly?)
HDFS
Adjacency Lists

Cache!

join
flatMap
reduceByKey
PageRank vector

join
flatMap
reduceByKey
PageRank vector

join
PageRank vector
Characteristics of Graph Algorithms

Parallel graph traversals
- Local computations
- Message passing along graph edges

Iterations

Even faster?
Big Data Processing in a Nutshell

Partition

Replicate

Reduce cross-partition communication

Let’s be smarter about this!
Simple Partitioning Techniques

Hash partitioning

Range partitioning on some underlying linearization

Web pages: lexicographic sort of domain-reversed URLs
How much difference does it make?

PageRank over webgraph
(40m vertices, 1.4b edges)

Simple Partitioning Techniques

Hash partitioning

Range partitioning on some underlying linearization
Web pages: lexicographic sort of domain-reversed URLs
Social networks: sort by demographic characteristics
Country Structure in Facebook

Analysis of 721 million active users (May 2011)

54 countries w/ >1m active users, >50% penetration

Ugander et al. (2011) The Anatomy of the Facebook Social Graph.
Simple Partitioning Techniques

Hash partitioning

Range partitioning on some underlying linearization
- Web pages: lexicographic sort of domain-reversed URLs
- Social networks: sort by demographic characteristics

But what about graphs in general?
Big Data Processing in a Nutshell

Partition
Replicate
Reduce cross-partition communication

Industry solution?
Partition
Partition

What’s the fundamental issue?
State-of-the-Art Distributed Graph Algorithms

Periodic synchronization

Fast asynchronous iterations

Fast asynchronous iterations
Graph Processing Frameworks
Graph Processing Frameworks

- Pregel
  - Google

- Apache Giraph
  - Based on Pregel
  - On Hadoop

- Spark GraphX
What is Apache Giraph

• Giraph performs iterative calculation on top of an existing Hadoop cluster
Bulk-Synchronous Parallel (BSP) Programming Model

Vertex-centric model

Iteration $i$ Iteration $i+1$
Vertex Centric Programming

- **Vertex Centric Programming Model**
  - Logic written from perspective on a single vertex.
  - Executed on all vertices.

- **Vertices know about**
  - Their own value(s)
  - Their outgoing edges
“Often expensive and should be used as sparingly as possible”
In superstep 0, every vertex is in the active state.
A vertex deactivates itself by voting to halt.
It can be reactivated by receiving an (external) message.
Algorithm termination is based on every vertex voting to halt.
Finding the Largest Value in a Graph

Superstep 0
- Worker
  - 3
  - 6
  - 2
  - 1

Superstep 1
- 6
- 6
- 2
- 6

Superstep 2
- 6
- 6
- 6
- 6

Superstep 3
- 6
- 6
- 6
- 6
Finding the Largest Value in a Graph

```java
public class MaxComputation extends BasicComputation<IntWritable, IntWritable, NullWritable, IntWritable> {
    @Override
    public void compute(Vertex<IntWritable, IntWritable, NullWritable> vertex, Iterable<IntWritable> messages) throws IOException {
        boolean changed = false;
        for (IntWritable message : messages) {
            if (vertex.getValue().get() < message.get()) {
                vertex.setValue(message);
                changed = true;
            }
        }
        if (getSuperstep() == 0 || changed) {
            sendMessageToAllEdges(vertex, vertex.getValue());
        }
        vertex.voteToHalt();
    }
}
```
Advantages

▪ Makes distributed programming easy
  ▪ No locks, semaphores, race conditions
  ▪ Separates computing from communication phase

▪ Vertex-level parallelization
  ▪ Bulk message passing for efficiency

▪ Stateful (in-memory)
  ▪ Only messages & checkpoints hit disk
Giraph Architecture

Master – Application coordinator
- Synchronizes supersteps
- Assigns partitions to workers before superstep begins

Workers – Computation & messaging
- Handle I/O – reading and writing the graph
- Computation/messaging of assigned partitions

ZooKeeper
- Maintains global application state
Lifecycle of a Giraph Program

Apache Giraph, Claudio Martella, Hadoop Summit, Amsterdam, April 2014
More Applications

Single Source Shortest path (SSSP)
SSSP (2/6)

Superstep 0

Worker 1

Worker 2

Worker 3

Worker 4

32
SSSP (5/6)

Superstep 3

Worker 1

Worker 2

Worker 3

Worker 4
SSSP (6/6)

Algorithm has converged
public void compute(Iterable<DoubleWritable> messages) {
    double minDist = Double.MAX_VALUE;
    for (DoubleWritable message : messages) {
        minDist = Math.min(minDist, message.get());
    }
    if (minDist < getValue().get()) {
        setValue(new DoubleWritable(minDist));
        for (Edge<LongWritable, FloatWritable> edge : getEdges()) {
            double distance = minDist + edge.getValue().get();
            sendMessage(edge.getTargetVertexId(), new DoubleWritable(distance));
        }
    }
    voteToHalt();
}