Extreme Computing

Distributed Graph Processing

Amir Shaikhha, Fall 2023
Graphs are everywhere

Social networks

Web graph/search engine

E-commerce

Maps

Computational biology
Example: PageRank

Web graph
Graph Parallel Algorithms

Dependency Graph

Local Updates

Iterative Computation

My Interests

Friends Interests
Many graph algorithms

• Collaborative Filtering
  – Alternating Least Squares
  – Stochastic Gradient Descent
  – Tensor Factorization

• Structured Prediction
  – Loopy Belief Propagation
  – Max-Product Linear Programs
  – Gibbs Sampling

• Semi-supervised ML
  – Graph SSL
  – CoEM

• Community Detection
  – Triangle-Counting
  – K-core Decomposition
  – K-Truss

• Graph Analytics
  – PageRank
  – Personalized PageRank
  – Shortest Path
  – Graph Coloring

• Classification
  – Neural Networks
Graph processing framework
Graph processing frameworks

- Application
- Graph-parallel
- Library
- Parallelization
- Communication
- Fault-tolerance
- Synchronization
- Load balancing
- Scheduling
Why do we need a new framework?

- Why don’t we just MapReduce?

- How would you implement Graph processing in MapReduce?
Data Dependencies are Difficult

- Difficult to express dependent data in MR
  - Substantial data transformations
  - User managed graph structure
  - Costly data replication
Iterative Computation is Difficult

• System is not optimized for iteration:
Pregel: Bulk Synchronous Parallel

Compute

Communicate

http://dl.acm.org/citation.cfm?id=1807184
Vertex centric API

**Active**

**Inactive**

*Vote to halt*

*Message received*
Programming API

Class Vertex{
    //Main methods
    Compute(MessageIterator *msgs);
    SendMsgTo(dest, msg);
    VoteToHalt();

    //Auxiliary methods
    GetValue();
    MutableValue();
    GetOutEdgeIterator();
    SuperStep();
}

Example: maximum value

Superstep 0

Superstep 1

Superstep 2

Superstep 3
Example: PageRank

\[
\text{PageRank of site} = \sum \frac{\text{PageRank of inbound link}}{\text{Number of links on that page}}
\]

OR

\[
PR(u) = (1 - d) + d \times \sum \frac{PR(v)}{N(v)}
\]

Iterate until it converges
Example: PageRank

```cpp
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();
        }
    }
};
```
Additional features

• Combiners
• Aggregators
• Topology mutations
  • Partial ordering
    • Removal first (Edge → vertex removal)
    • Addition (vertex → edge addition)
• Handlers
  • User-defined functions to resolve conflicts
• Input/output
  • File, GFS, BigTable, etc.
Implementation

Master

Job tracker
(Pregel)

Worker

Worker

Worker

Worker
Fault tolerance

• Achieved through checkpointing
• At the beginning of a super-step, master instructs the workers to take a check-point
• When a worker fails --- the master re-assigns the partition to a new worker, and restarts from the latest checkpoint
Distributed Collections Views

Table View

Graph View

Table

Row

Row

Row

Row

Dependency Graph

Result

hadoop

Spark

Pregel
Spark Software Stack

Apache Spark

Spark SQL
(SQL)

MLlib
(Machine Learning)

GraphX
(Graph processing)

Spark Streaming
(Streaming)

Other Spark libraries

Spark Core
Processing Engine

Mesos / YARN / Standalone
Cluster Resource Management

HDFS / Amazon S3 / OpenStack Swift / Cassandra
Distributed File System & Storage

Even though in some classes of applications Spark is considered a better alternative to Hadoop, in many others it has limitations that make it complementary to Hadoop. The main limitation of Spark is that datasets should fit in RAM memory. In addition, it does not provide its own distributed storage system, which is a fundamental requirement for Big Data applications. To overcome this lack, Spark has been designed to run on top of several data sources, such as distributed file systems (e.g., HDFS), Cloud object storages (e.g., Amazon S3, OpenStack Swift) and NoSQL databases (e.g., Cassandra).

Spark's real-time processing capability is increasingly being used into applications that requires to extract insights quickly from data, such as recommendation and monitoring systems. For this reason, several big companies exploit Spark for data analysis purpose: SK Telecom analyzes mobile usage patterns of customers, Ebay uses Spark for log aggregation, and Kelkoo for product recommendations.

1. https://spark.apache.org

GraphX

- Tables and Graphs are composable views of the same physical data

- Each view has its own operators that exploit the semantics of the view to achieve efficient execution
View a Graph as a Table

Property Graph

Vertex Property Table

<table>
<thead>
<tr>
<th>Id</th>
<th>Property (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(P1, P4)</td>
</tr>
<tr>
<td>M</td>
<td>(P2, P4)</td>
</tr>
<tr>
<td>I</td>
<td>(P3, P4)</td>
</tr>
<tr>
<td>R</td>
<td>(P3, P4)</td>
</tr>
</tbody>
</table>

Edge Property Table

<table>
<thead>
<tr>
<th>SrcId</th>
<th>DstId</th>
<th>Property (E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>M</td>
<td>P5</td>
</tr>
<tr>
<td>I</td>
<td>M</td>
<td>P6</td>
</tr>
<tr>
<td>I</td>
<td>A</td>
<td>P7</td>
</tr>
<tr>
<td>R</td>
<td>I</td>
<td>P8</td>
</tr>
</tbody>
</table>
class Graph [ V, E ] {
    def Graph(vertices: Table[(Id, V)],
              edges: Table[(Id, Id, E)]) {

        // Table Views ---------------------
        def vertices: Table[(Id, V)]
        def edges: Table[(Id, Id, E)]

        // Transformations -------------------
        def reverse: Graph[V, E]
        def subgraph(pV: (Id, V) => Boolean,
                      pE: Edge[V,E] => Boolean): Graph[V,E]
        def mapV(m: (Id, V) => T): Graph[T,E]
        def mapE(m: Edge[V,E] => T): Graph[V,T]

        // Joins -------------------------------
        def joinV(tbl: Table[(Id, T)]): Graph[(V, T), E]
        def joinE(tbl: Table[(Id, Id, T)]): Graph[V, (E, T)]

        // Computation -----------------------
        def mrTriplets(mapF: (Edge[V,E]) => List[(Id, T)],
                        reduceF: (T, T) => T): Graph[T, E]
    }
}
Distributed Graphs as RDDs

**Property Graph**

**Part. 1**

**Part. 2**

2D Vertex Cut Heuristic

**Vertex Table (RDD)**

**Routing Table (RDD)**

**Edge Table (RDD)**

Distributed Graphs as RDDs
Summary

• Graph processing with Pregel/Giraph
  • Bulk Synchronous Programming (BSP) model
• Graph processing on Spark with GraphX

Resources:
• Giraph: http://giraph.apache.org/
• GraphX: https://spark.apache.org/graphx/
• GraphLab: http://graphlablab.org/
• Okapi: http://grafos.ml/
Resources

• Compulsory reading:
  • Pregel [SIGMOD’10]

• Recommended reading
  – GraphX [OSDI’14]
    • Graph processing framework built on top of Spark
  – GraphLab [OSDI’12]
    • Edge-centric graph processing framework
QUESTIONS?