Pregel and GraphX

Reza Zadeh

@Reza_Zadeh | http://reza-zadeh.com
Overview

Graph Computations and Pregel

Introduction to Matrix Computations
Graph Computations and Pregel
Data Flow Models

Restrict the programming interface so that the system can do more automatically

Express jobs as graphs of high-level operators
  » System picks how to split each operator into tasks and where to run each task
  » Run parts twice fault recovery

New example: Pregel (parallel graph google)
Pregel

Exposé specialized APIs to simplify graph programming.

“Think like a vertex”
Graph-Parallel Pattern

Model / Alg. State

Computation depends only on the neighbors

6
Pregel Data Flow

- Input graph
- Vertex state 1
- Messages 1
- Group by vertex ID
- Superstep 1
- Vertex state 2
- Messages 2
- Group by vertex ID
- Superstep 2
- ...
Simple Pregel in Spark

Separate RDDs for immutable graph state and for vertex states and messages at each iteration

Use groupByKey to perform each step

Cache the resulting vertex and message RDDs

Optimization: co-partition input graph and vertex state RDDs to reduce communication
Example: PageRank

\[ R[i] = 0.15 + \sum_{j \in \text{Nbrs}(i)} w_{ji} R[j] \]

- Rank of user \( i \)
- Weighted sum of neighbors’ ranks

Update ranks in parallel
Iterate until convergence
PageRank in Pregel

Input graph → Vertex ranks 1 → Contributions 1 → Superstep 1 (add contribs) → Vertex ranks 2 → Contributions 2 → Superstep 2 (add contribs) → ...
GraphX

class Graph[VD, ED] {
  val vertices: VertexRDD[VD]
  val edges: EdgeRDD[ED]
}

Provides Pregel message-passing and other operators on top of RDDs
GraphX: Properties

**Property Graph**

- Node 3: Advisor
  - txin
  - stu.
- Node 4: Collab.
  - franklin
  - prof.
- Node 5: Colleague
  - jgonzal
  - pst.doc.
- Node 6: Colleague
  - istoica
  - prof.

**Vertex Table**

<table>
<thead>
<tr>
<th>Id</th>
<th>Property (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>(txin, student)</td>
</tr>
<tr>
<td>7</td>
<td>(jgonzal, postdoc)</td>
</tr>
<tr>
<td>5</td>
<td>(franklin, professor)</td>
</tr>
<tr>
<td>2</td>
<td>(istoica, professor)</td>
</tr>
</tbody>
</table>

**Edge Table**

<table>
<thead>
<tr>
<th>SrcId</th>
<th>DstId</th>
<th>Property (E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>7</td>
<td>Collaborator</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>Advisor</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>Colleague</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>PI</td>
</tr>
</tbody>
</table>
GraphX: Triplets

The *triplets* operator joins vertices and edges:
Map Reduce Triplets

Map-Reduce for each vertex

mapF(\( \langle A, B \rangle \)) → \( A_1 \)

mapF(\( \langle A, C \rangle \)) → \( A_2 \)

reduceF(\( A_1, A_2 \)) → \( A \)
Example: Oldest Follower

What is the age of the oldest follower for each user?

```scala
val oldestFollowerAge = graph
  .mrTriplets(
    e => (e.dst.id, e.src.age), //Map
    (a, b) => max(a, b) //Reduce
  ).vertices
```

![Diagram](image-url)
Summary of Operators

All operations:

https://spark.apache.org/docs/latest/graphx-programming-guide.html#summary-list-of-operators

Pregel API:

https://spark.apache.org/docs/latest/graphx-programming-guide.html#pregel-api
The GraphX Stack
(Lines of Code)

PageRank (5)
Connected Comp. (10)
Shortest Path (10)
SVD (40)
ALS (40)
K-core (51)
Triangle Count (45)
LDA (120)

Pregel (28) + GraphLab (50)

GraphX (3575)

Spark
Optimizations

Overloaded vertices have their work distributed

Edge Cut

Vertex Cut
Optimizations

Property Graph

Vertex Table (RDD)

Routing Table (RDD)

Edge Table (RDD)
More examples

In your HW: Single-Source-Shortest Paths using Pregel
Distributing Matrix Computations
Distributing Matrices

How to distribute a matrix across machines?

» By Entries (CoordinateMatrix)

» By Rows (RowMatrix)

» By Blocks (BlockMatrix)  As of version 1.3

All of Linear Algebra to be rebuilt using these partitioning schemes
Distributing Matrices

Even the simplest operations require thinking about communication e.g. multiplication

How many different matrix multiplies needed?

» At least one per pair of \{Coordinate, Row, Block, LocalDense, LocalSparse\} = 10

» More because multiplies not commutative
Block Matrix Multiplication

Let’s look at Block Matrix Multiplication
(on the board and on GitHub)