GraphX

Graph Analytics in Spark

Ankur Dave
Graduate Student, UC Berkeley AMPLab

Joint work with Joseph Gonzalez, Reynold Xin, Daniel Crankshaw, Michael Franklin, and Ion Stoica
Graphs
Social Networks
Web Graphs
User-Item Graphs
Graph Algorithms
PageRank
Triangle Counting
Collaborative Filtering

\[ \text{Users} \approx f(i) \times \text{Products} \]

Ratings

Products
Collaborative Filtering

\[ f[i] = \arg \min_{w \in \mathbb{R}^d} \sum_{j \in \text{Nbrs}(i)} (r_{ij} - w^T f[j])^2 + \lambda \|w\|_2^2 \]
The Graph-Parallel Pattern
The Graph-Parallel Pattern
The Graph-Parallel Pattern
| Many Graph-Parallel Algorithms |
|-----------------|-----------------|
| Collaborative Filtering | Community Detection |
| › Alternating Least Squares | › Triangle-Counting |
| › Stochastic Gradient Descent | › K-core Decomposition |
| › Tensor Factorization | › K-Truss |
| Structured Prediction | Graph Analytics |
| › Loopy Belief Propagation | › PageRank |
| › Max-Product Linear Programs | › Personalized PageRank |
| › Gibbs Sampling | › Shortest Path |
| | › Graph Coloring |
| Semi-supervised ML | Classification |
| › Graph SSL | › Neural Networks |
| › CoEM | |
High-Degree Vertices

Challenges:

1. Storage: How to store a graph where one vertex’s edges don’t fit on a machine?

2. API: How to expose parallelism within vertex neighborhoods?
Complex Pipelines

Raw Wikipedia

Link Table

Hyperlinks

PageRank

Top 20 Pages

Editor Table

Editor Graph

Community Detection

User Community

Top Communities
Complex Pipelines

Solution: Embed graph processing within a table-oriented system (Spark)

Challenges:

1. Storage: How to store graphs as tables?
2. Computation: How to express graph ops as table ops (map, reduce, join, etc.)?
3. API: How to present the two views to the user?
The GraphX API
Property Graphs

Vertex Property:
- User Profile
- Current PageRank Value

Edge Property:
- Weights
- Relationships
- Timestamps
Creating a Graph (Scala)

type VertexId = Long

val vertices: RDD[(VertexId, String)] = 
sc.parallelize(List(
    (1L, “Alice”),
    (2L, “Bob”),
    (3L, “Charlie”)))

class Edge[ED](
    val srcId: VertexId,
    val dstId: VertexId,
    val attr: ED)

val edges: RDD[Edge[String]] = 
sc.parallelize(List(
    Edge(1L, 2L, “coworker”),
    Edge(2L, 3L, “friend”)))

val graph = Graph(vertices, edges)
class Graph[VD, ED] {

  // Table Views --------------------------------------
  def vertices: RDD[(VertexId, VD)]
  def edges: RDD[Edge[ED]]
  def triplets: RDD[EdgeTriplet[VD, ED]]

  // Transformations --------------------------------
  def mapVertices[VD2](f: (VertexId, VD) => VD2): Graph[VD2, ED]
  def mapEdges[ED2](f: Edge[ED] => ED2): Graph[VD2, ED]
  def reverse: Graph[VD, ED]
  def subgraph(epred: EdgeTriplet[VD, ED] => Boolean,
                  vpred: (VertexId, VD) => Boolean): Graph[VD, ED]

  // Joins ------------------------------------------
  def outerJoinVertices[U, VD2](tbl: RDD[(VertexId, U))
      (f: (VertexId, VD, Option[U]) => VD2): Graph[VD2, ED]

  // Computation -----------------------------------
  def aggregateMessages[A](
      sendMsg: EdgeContext[VD, ED, A] => Unit,
      mergeMsg: (A, A) => A): RDD[(VertexId, A)]
// Continued from previous slide
def pageRank(tol: Double): Graph[Double, Double]
def triangleCount(): Graph[Int, ED]
def connectedComponents(): Graph[VertexId, ED]
// ...and more: org.apache.spark.graphx.lib

PageRank

Triangle Count

Connected Components
The triplets view

class Graph[VD, ED] {
   def triplets: RDD[EdgeTriplet[VD, ED]]
}

class EdgeTriplet[VD, ED](
   val srcId: VertexId, val dstId: VertexId, val attr: ED,
   val srcAttr: VD, val dstAttr: VD)

<table>
<thead>
<tr>
<th>srcAttr</th>
<th>dstAttr</th>
<th>attr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>coworker</td>
<td>Bob</td>
</tr>
<tr>
<td>Bob</td>
<td>friend</td>
<td>Charlie</td>
</tr>
</tbody>
</table>
The subgraph transformation

class Graph[VD, ED] {
    def subgraph(epred: EdgeTriplet[VD, ED] => Boolean,
                 vpred: (VertexId, VD) => Boolean): Graph[VD, ED]
}

graph.subgraph(epred = (edge) => edge.attr != "relative")
The subgraph transformation

class Graph[VD, ED] {
    def subgraph(epred: EdgeTriplet[VD, ED] => Boolean,
                 vpred: (VertexId, VD) => Boolean): Graph[VD, ED]
}

graph.subgraph(vpred = (id, name) => name != "Bob")
Computation with `aggregateMessages`

class `Graph[VD, ED]`
{
    def aggregateMessages[A](
        sendMsg: `EdgeContext[VD, ED, A]` => Unit,
        mergeMsg: (A, A) => A): RDD[(`VertexId`, A)]
    }

class `EdgeContext[VD, ED, A]`
{
    val srcId: `VertexId`, val dstId: `VertexId`, val attr: ED,
    val srcAttr: VD, val dstAttr: VD) {
    def sendToSrc(msg: A)
    def sendToDst(msg: A)
    }

    graph.aggregateMessages(
        ctx => {
            ctx.sendToSrc(1)
            ctx.sendToDst(1)
        },
        _ + _)
Computation with `aggregateMessages`
Example: Graph Coarsening

Web Pages → Intra-Domain Links → Pages by Domain

Graph constructor

Domains

Connected Components

Spark group-by

Domain Graph
How GraphX Works
Storing Graphs as Tables

Property Graph

Vertex Table (RDD)
- Machine 1
  - A
  - B
- Machine 2
  - C
  - D

Edge Table (RDD)
- Machine 2
  - A
  - E
- Machine 1
  - F
  - D
  - C
  - B
  - A

Spark Summit East
Simple Operations

Reuse vertices or edges across multiple graphs

Input Graph

Transformed Graph

Vertex Table

Edge Table

Vertex Table

Edge Table

Transform Vertex Properties
Implementing **triplets**

**Vertex Table (RDD)**
- Machine 1:
  - A
  - B
  - C
- Machine 2:
  - D
  - E
  - F

**Routing Table (RDD)**
- A 1
- B 1
- C 1
- D 1
- E 2
- F 2

**Edge Table (RDD)**
- A
  - B
  - C
- B
  - C
  - D
- C
  - D
  - E
- D
  - E
  - F
- E
  - D
  - F
- F
  - E
  - F
Implementing **triplets**
Reduction in Communication Due to Cached Updates

**Connected Components on Twitter Graph**
(1.4B edges)

![Graph showing network communication (MB) vs iteration](chart.png)
Most vertices are within 8 hops of all vertices in their component
Implementing `aggregateMessages`

- **Edge Table (RDD)**
  - Mirror Cache
    - A
    - B
    - C
    - D
  - A
  - B
  - C
  - D
  - A
  - B
  - C
  - D

- **Mirror Cache**
  - A
  - D
  - E
  - F
  - A
  - A
  - E
  - A
  - A
  - F
  - E
  - E
  - F

- **Scan**
- **Result Vertex Table**
  - A
  - B
  - C
  - D
  - E
  - F
Speedup Due To Access Method Selection

**Connected Components on Twitter (1.4B edges)**

- **Scan All Edges**
- **Index of "Active" Edges**

**Runtime (Seconds)**

**Iteration**

Scan

Indexed
GraphX performs comparably to state-of-the-art graph processing systems.
Open Source

• Alpha release with Spark 0.9.0 in Feb 2014
• Stable release with Spark 1.2.0 in Dec 2014
Future of GraphX

1. Language support
   a) Java API: PR #3234
   b) Python API: collaborating with Intel, SPARK-3789

2. More algorithms
   a) LDA (topic modeling): PR #2388
   b) Correlation clustering

3. Research
   a) Local graphs
   b) Streaming/time-varying graphs
   c) Graph database–like queries
IndexedRDD

Support efficient updates to immutable RDDs using purely functional data structures

https://github.com/amplab/spark-indexedrdd
Thanks!

http://spark.apache.org/graphx

ankurd@eecs.berkeley.edu

jegonzal@eecs.berkeley.edu

rxin@eecs.berkeley.edu

crankshaw@eecs.berkeley.edu