Big Data Systems

Djellel Difallah

Spring 2023

Lecture 14 – Graph Processing

Outline

- Introduction
- The PageRank Algorithm
- Google Pregel
- GraphX

Introduction

- Graphs are data structures used to represent relationships between entities.
- Many real-world computing problems have to deal with extremely large graphs:
  - Web graph, social networks, Urban networks
- Graph processing: Analyzing and processing large-scale graphs to find structural patterns and metrics

Graph Analytics
- Shortest Path
- PageRank
- Graph Coloring
- Minimum Cut
- Connected Components

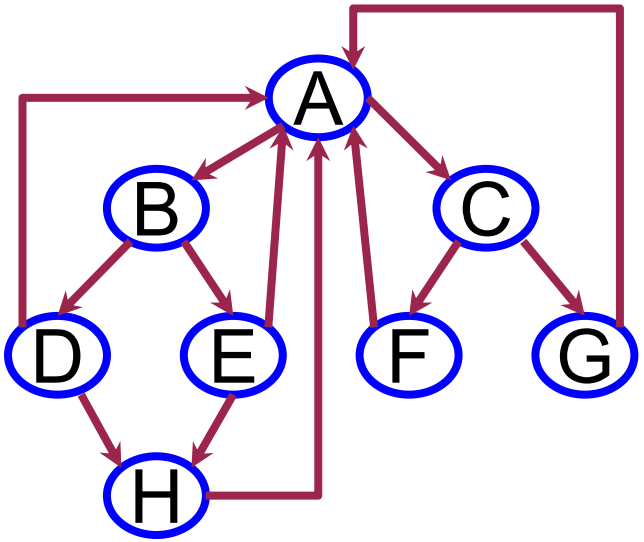
Machine Learning
- Structured Prediction (Graphical Models)
- Collaborative Filtering
- Community Detection
- Classification
- Clustering, etc.

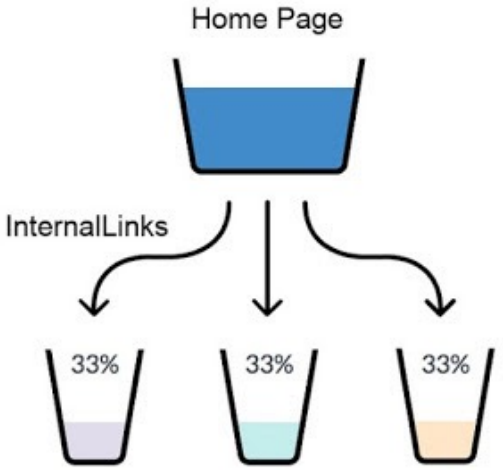
Web Information Retrieval

- Term frequency is one of many signals for ranking documents
- What if you have spam content? e.g., pages with all keywords, or fake content.
- Another important factor to consider is page ”authority” which indicates the reliability of the source
  - Equivalent to citation analysis, and impact factor in scientific publications
- Google introduced the PageRank Algorithm
  - Brin, Sergey, and Lawrence Page. "The anatomy of a large-scale hypertextual web search engine." *WWW 1998*
  - Leverages the hyperlink structure of the Web
  - In-link voting

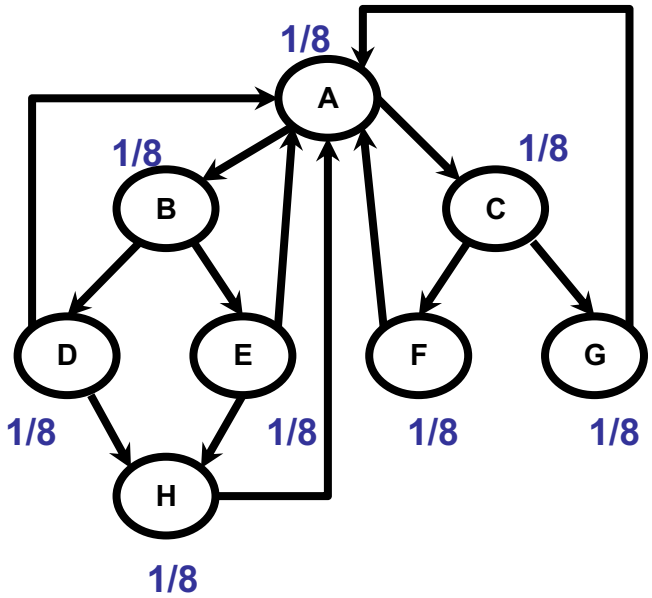
A Graph Representation of the Web

- ***Basic PageRank Update Rule:*** Each page divides its current PageRank ("flow") equally across its outgoing links and passes these equal shares to the pages it points to. (If a page has no outgoing links, it passes all its current PageRank to itself.) Notice that the total PageRank ("flow") in the network is unchanged.





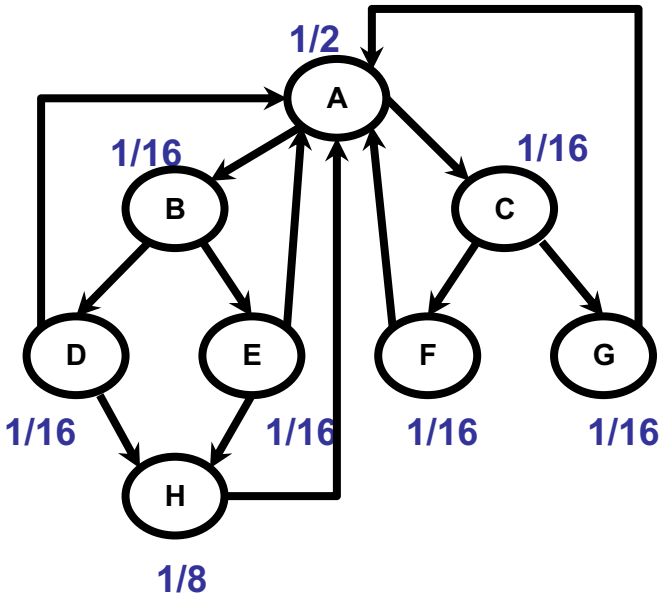
PageRank



1. In a network with *n* nodes, we assign all nodes the same initial PageRank, ***1/n***
2. We perform a sequence of *k* updates to the PageRank: Each page's PageRank: **sum of the shares it receives**

Step	A	B	C	D	E	F	G	H
0	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8

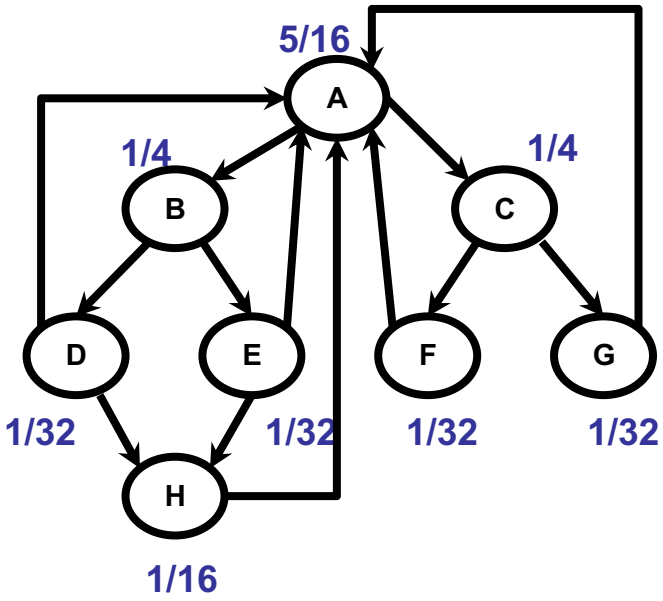
PageRank



1. In a network with *n* nodes, we assign all nodes the same initial PageRank, *1/n*
2. We perform a sequence of ***k* updates** to the PageRank: Each page's PageRank: **sum of the shares it receives**

Step	A	B	C	D	E	F	G	H
0	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
1	1/2	1/16	1/16	1/16	1/16	1/16	1/16	1/8

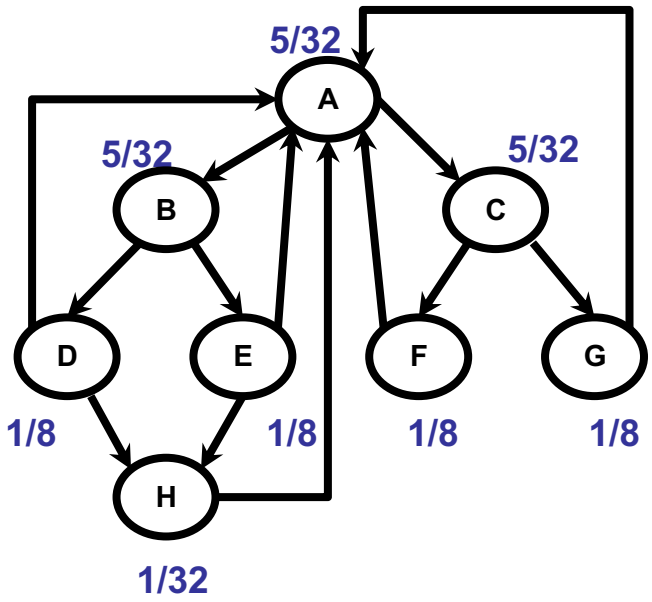
PageRank



1. In a network with *n* nodes, we assign all nodes the same initial PageRank, *1/n*
2. We perform a sequence of *k* updates to the PageRank: Each page's PageRank: **sum of the shares it receives**

Step	A	B	C	D	E	F	G	H
0	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
1	1/2	1/16	1/16	1/16	1/16	1/16	1/16	1/8
2	5/16	1/4	1/4	1/32	1/32	1/32	1/32	1/16

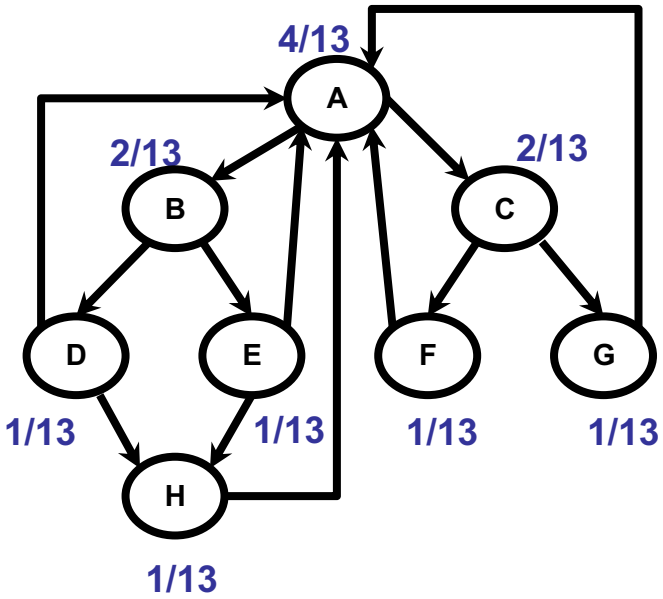
PageRank



1. In a network with *n* nodes, we assign all nodes the same initial PageRank, *1/n*
2. We perform a sequence of *k* updates to the PageRank: Each page's PageRank: **sum of the shares it receives**

Step	A	B	C	D	E	F	G	H
0	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
1	1/2	1/16	1/16	1/16	1/16	1/16	1/16	1/8
2	5/16	1/4	1/4	1/32	1/32	1/32	1/32	1/16
3	5/32	5/32	5/32	1/8	1/8	1/8	1/8	1/32

PageRank



1. In a network with *n* nodes, we assign all nodes the same initial PageRank, *1/n*
2. We perform a sequence of *k* updates to the PageRank: Each page's PageRank: **sum of the shares it receives**

Step	A	B	C	D	E	F	G	H
0	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
1	1/2	1/16	1/16	1/16	1/16	1/16	1/16	1/8
2	5/16	1/4	1/4	1/32	1/32	1/32	1/32	1/16
3	5/32	5/32	5/32	1/8	1/8	1/8	1/8	1/32
…								
∞	4/13	2/13	2/13	1/13	1/13	1/13	1/13	1/13

Solution exists? Assumption:
Strongly connected network: each node can reach each other node by a directed path

PREGEL

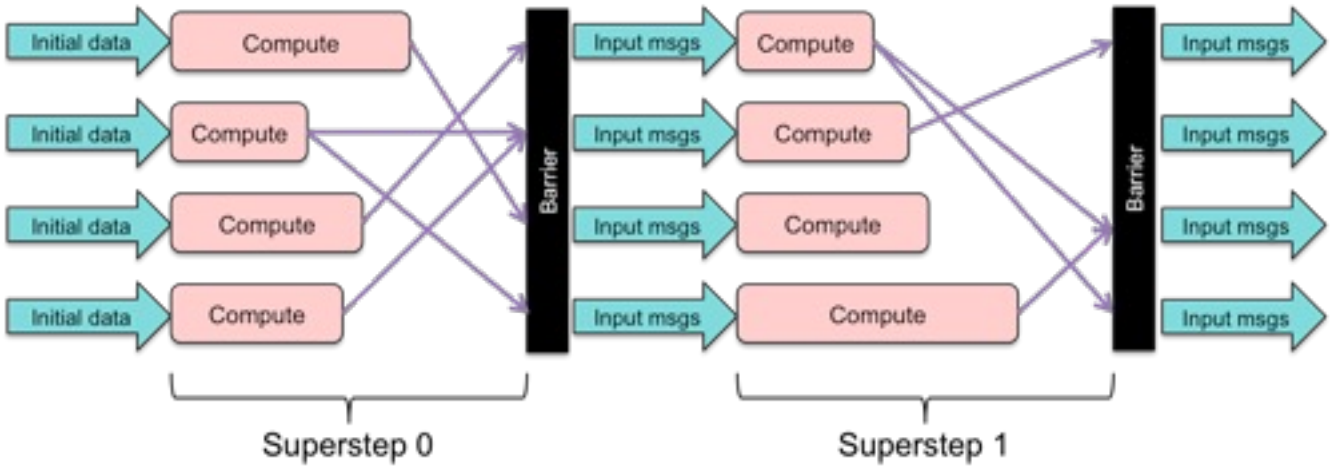
Google Pregel

- Developed by Google to address large-scale graph processing challenges
- Pregel is a scalable and fault-tolerant platform with an API that is sufficiently flexible to express arbitrary graph algorithms
  - Malewicz, Grzegorz, et al. "Pregel: a system for large-scale graph processing." *SIGMOD 2010*.
- Inspired by Bulk Synchronous Parallel (BSP) model

- Hadoop open source alternative: Apache Giraph
  - Exensively used by Facebook

Bulk Synchronous Parallel (BSP)

- BSP is programming model and computation framework for parallel computing.
  - [Paper] Valiant, L. G. "A bridging model for parallel computation." CACM (1990)
- The Computation is divided into a sequence of **supersteps**
  - Processes run concurrently, execute the same code, and create messages for other processes during a superstep
  - Barrier synchronization ensures all messages have been transmitted before the next superstep begins
  - Messages are delivered at the start of the next superstep
  - Restriction on sending and receiving messages within a superstep ensures deadlock-free execution

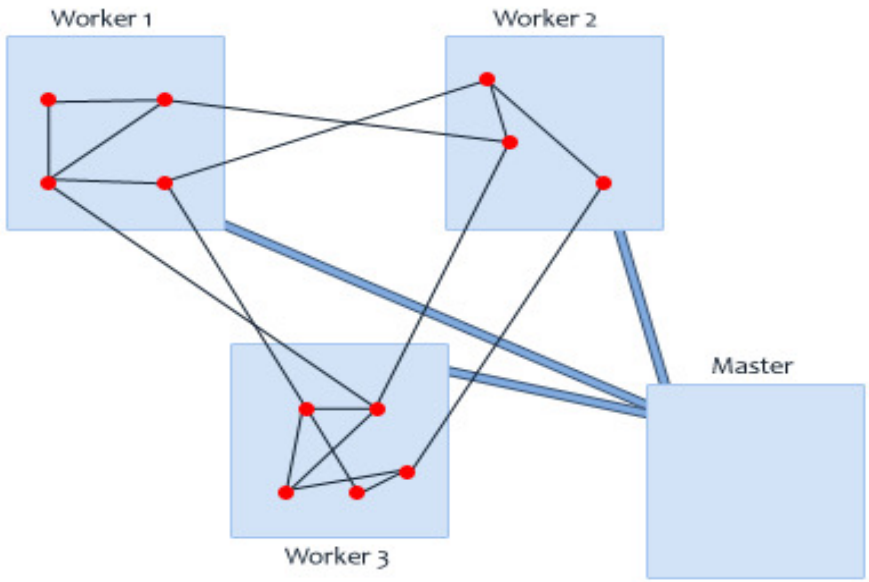


Why is a new Paradigm Needed?

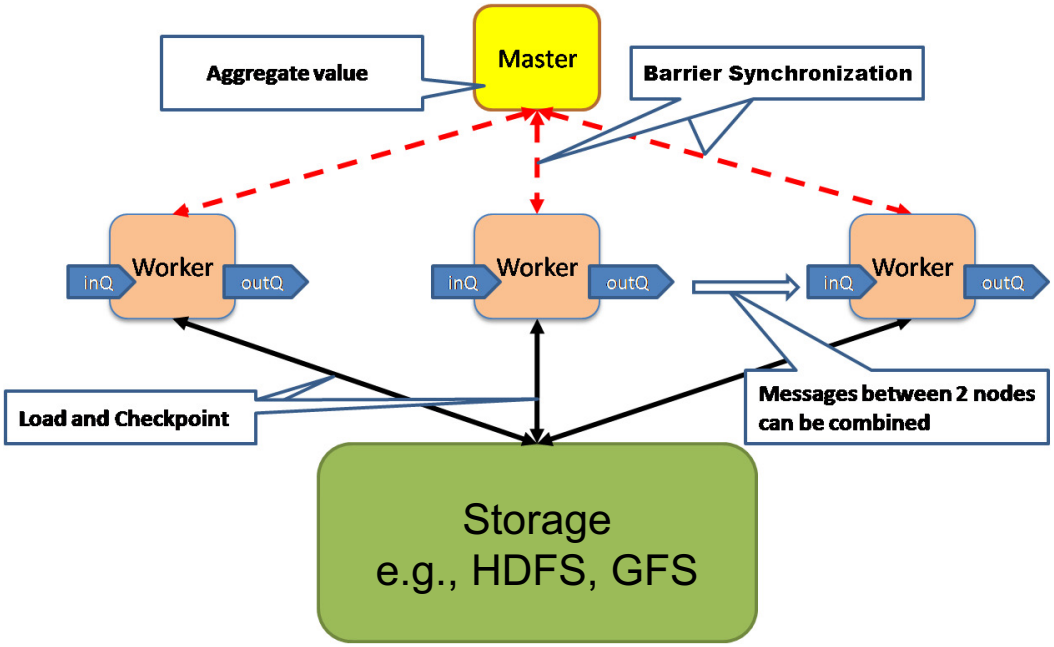
- Graph algorithms can be written as a series of chained MapReduce invocation
- MapReduce shortcomings:
  - Passes the entire state of the graph from one stage to the next
  - Needs to coordinate the steps of a chained MapReduce
- Pregel
  - Keeps vertices & edges on the machine that performs computation
  - Uses network transfers only for messages

Pregel Computation Model

- Pregel is also known as the *think-like-a-vertex model*



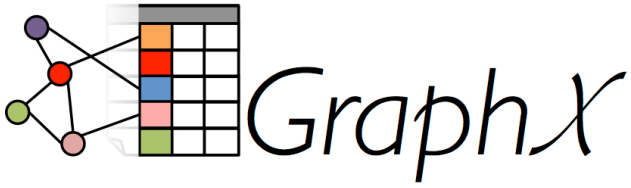
Pregel Architecture



From: http://java.dzone.com/news/google-pregel-graph-processing

PageRank in Pregel's API

```
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) { // In practice would use an aggregator to detect convergence.
      const int64 n = GetOutIterator().size();
      SendMessageToAllNeighbors(GetValue() / n);
    } else {
      VoteToHalt();
    }
  }
};
```

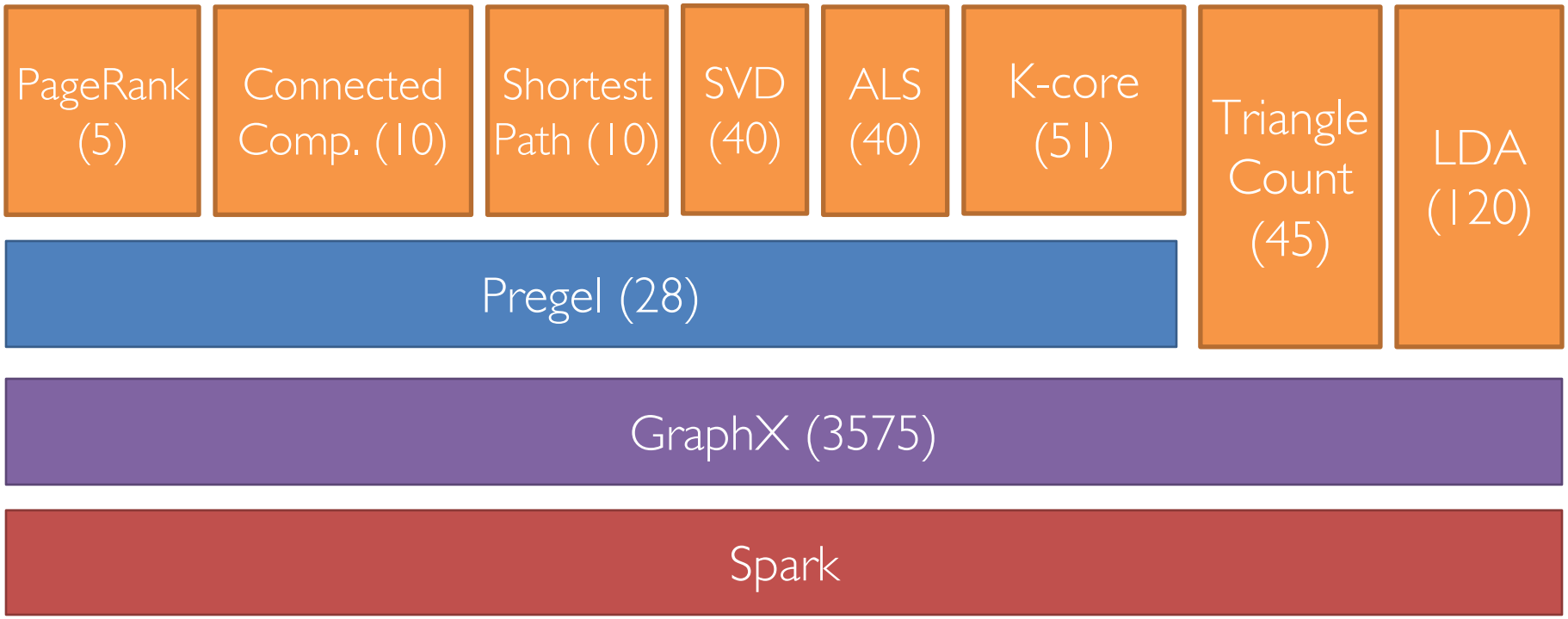


SPARK GRAPHX

Introduction

- GraphX: a component in Spark for graphs and graph-parallel computation
  - [Paper] Gonzalez, Joseph E., et al. "Graphx: Graph processing in a distributed dataflow framework." OSDI 2014
- Extends Spark RDD with a Graph abstraction: directed multigraph with properties on vertices and edges
- Pregel API: GraphX offers a Pregel-like API for iterative graph processing based on the Bulk Synchronous Parallel (BSP) model
- Built-in collection of graph algorithms and builders for easy graph analytics
- Doesn't support PySpark
  - Python: Newer version *GraphFrame* is in pre-release

Spark GraphX Stack



PageRank in GraphX

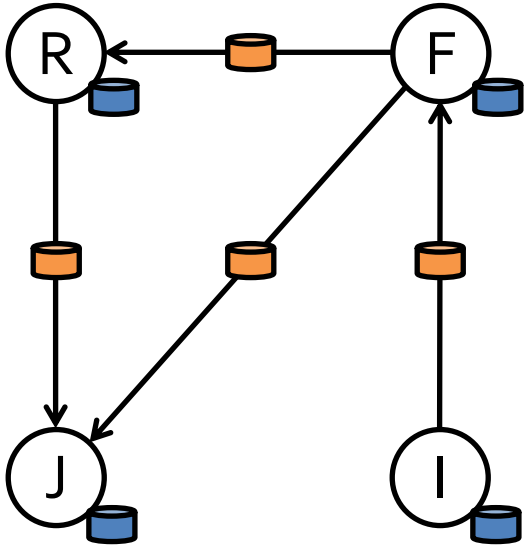
```
import org.apache.spark.graphx.GraphLoader

// 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"))
```

https://github.com/apache/spark/blob/master/data/graphx/followers.txt
https://github.com/apache/spark/blob/master/data/graphx/users.txt

View a Graph as a Table

Property Graph



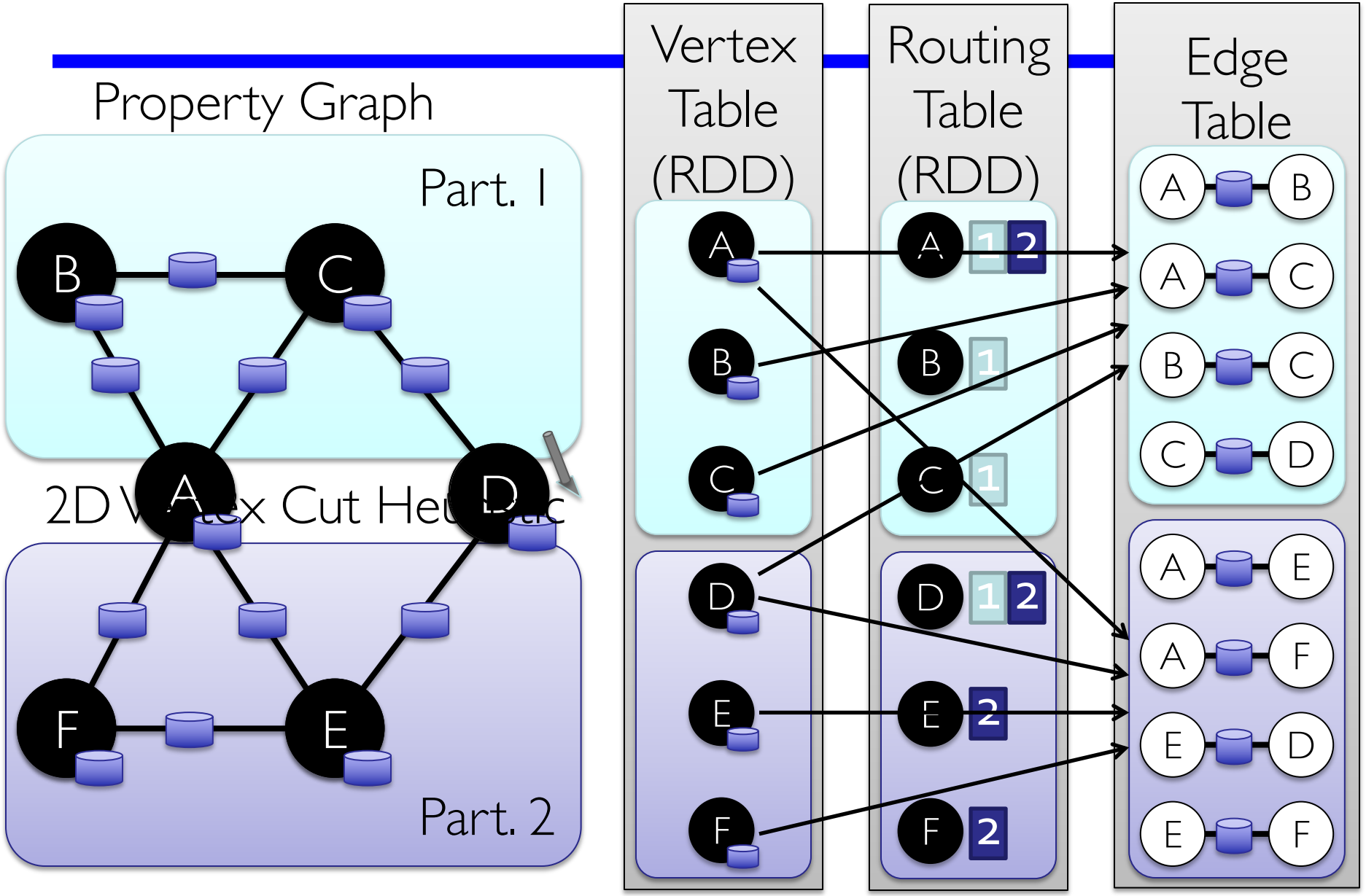
VertexRDD

Id	Property (V)
Rxin	(Stu., Berk.)
Jegonzal	(PstDoc, Berk.)
Franklin	(Prof., Berk)
Istoica	(Prof., Berk)

EdgeRDD

SrcID	DstID	Property (E)
rxin	jegonzal	Friend
franklin	rxin	Advisor
istoica	franklin	Coworker
franklin	jegonzal	PI

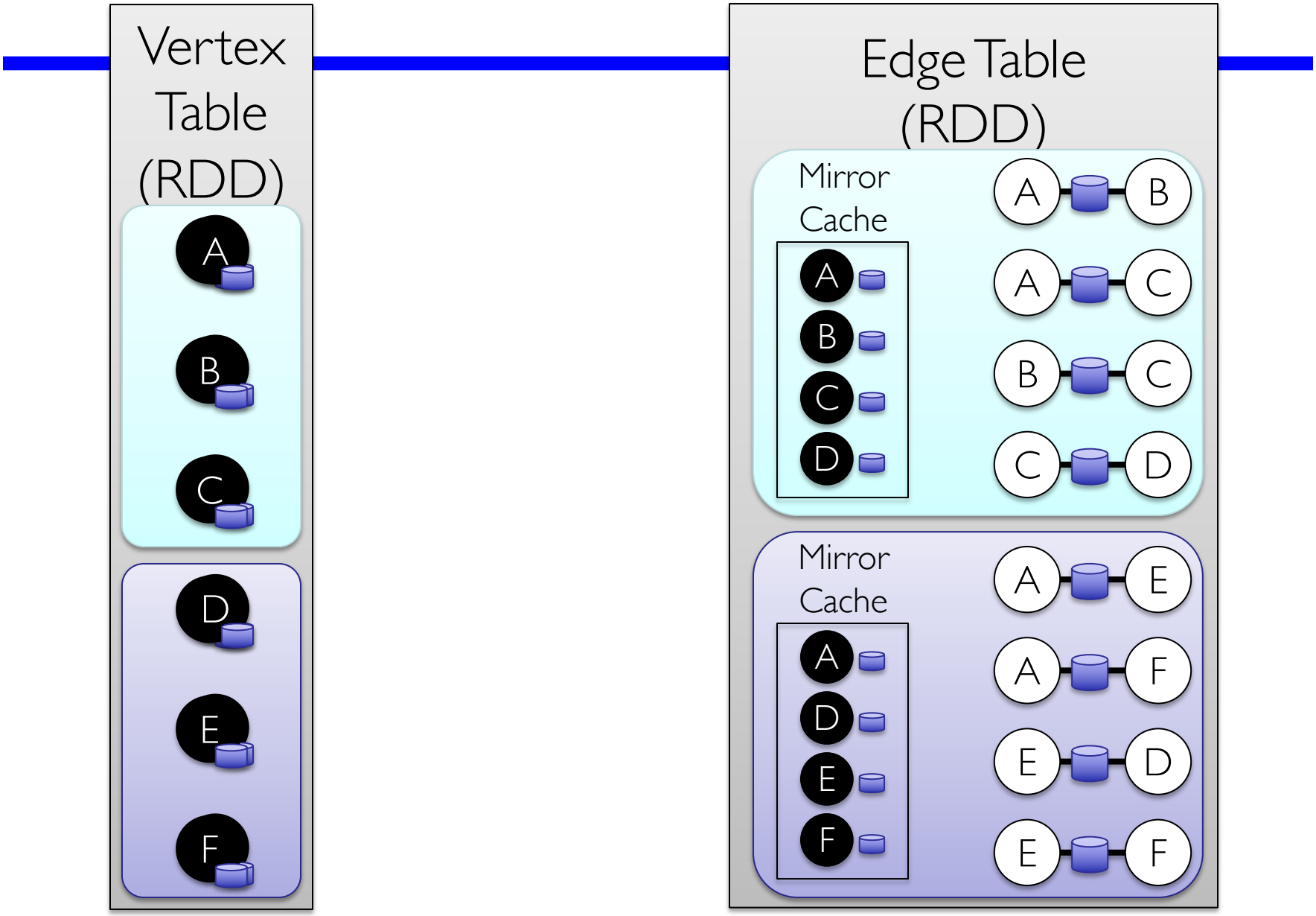
GraphX System Design



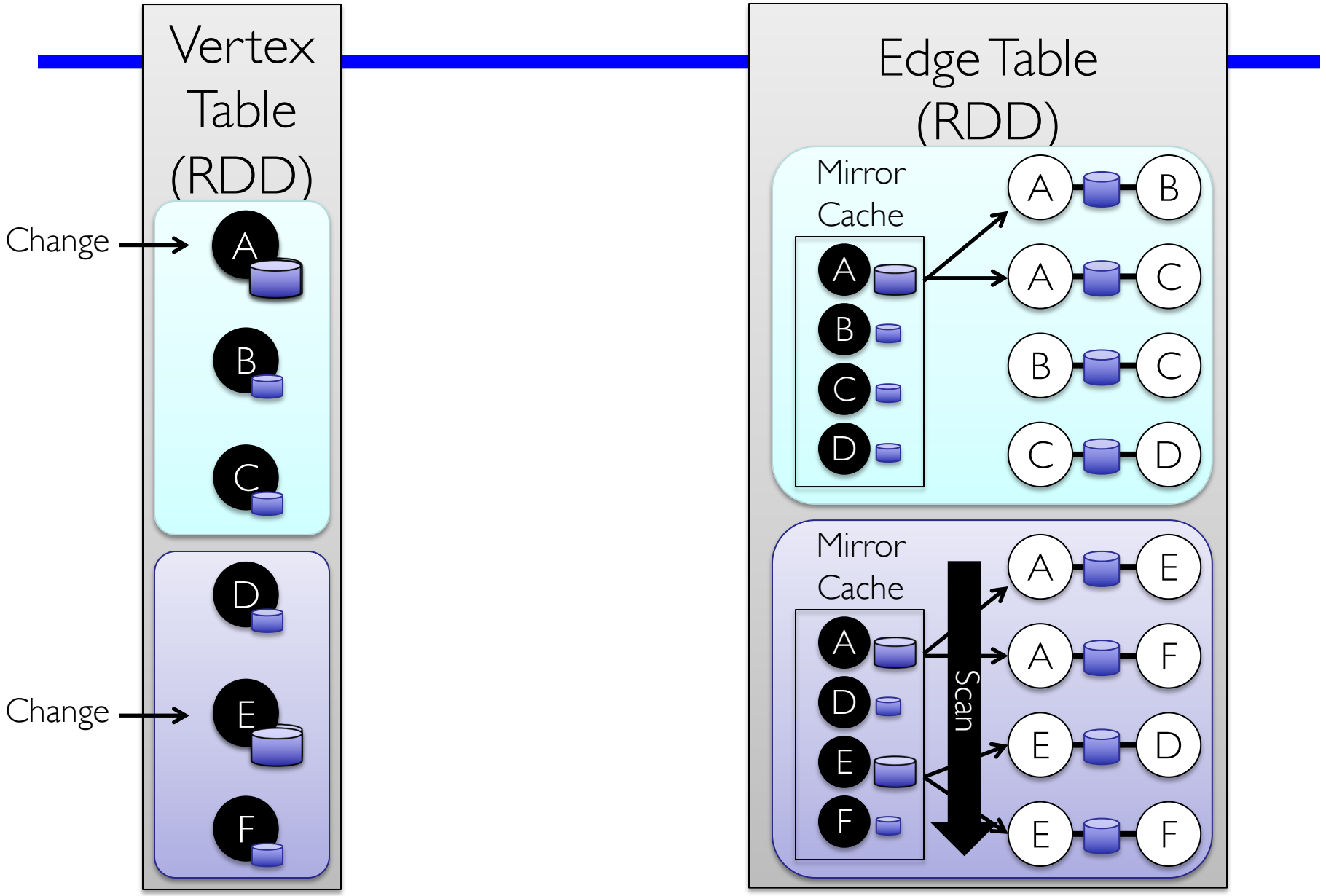
Optimizations

- **Vertex Caching:** To reduce the overhead of data movement, GraphX caches vertex properties that are frequently accessed in memory, enabling faster access in subsequent iterations.
- **Graph Pruning:** GraphX can remove unnecessary vertices and edges that do not contribute to the computation, reducing the overall size of the graph and improving processing efficiency.
- **Incremental Computation:** GraphX supports incremental computation, where only the changes to the graph structure or properties are processed, avoiding redundant computation.

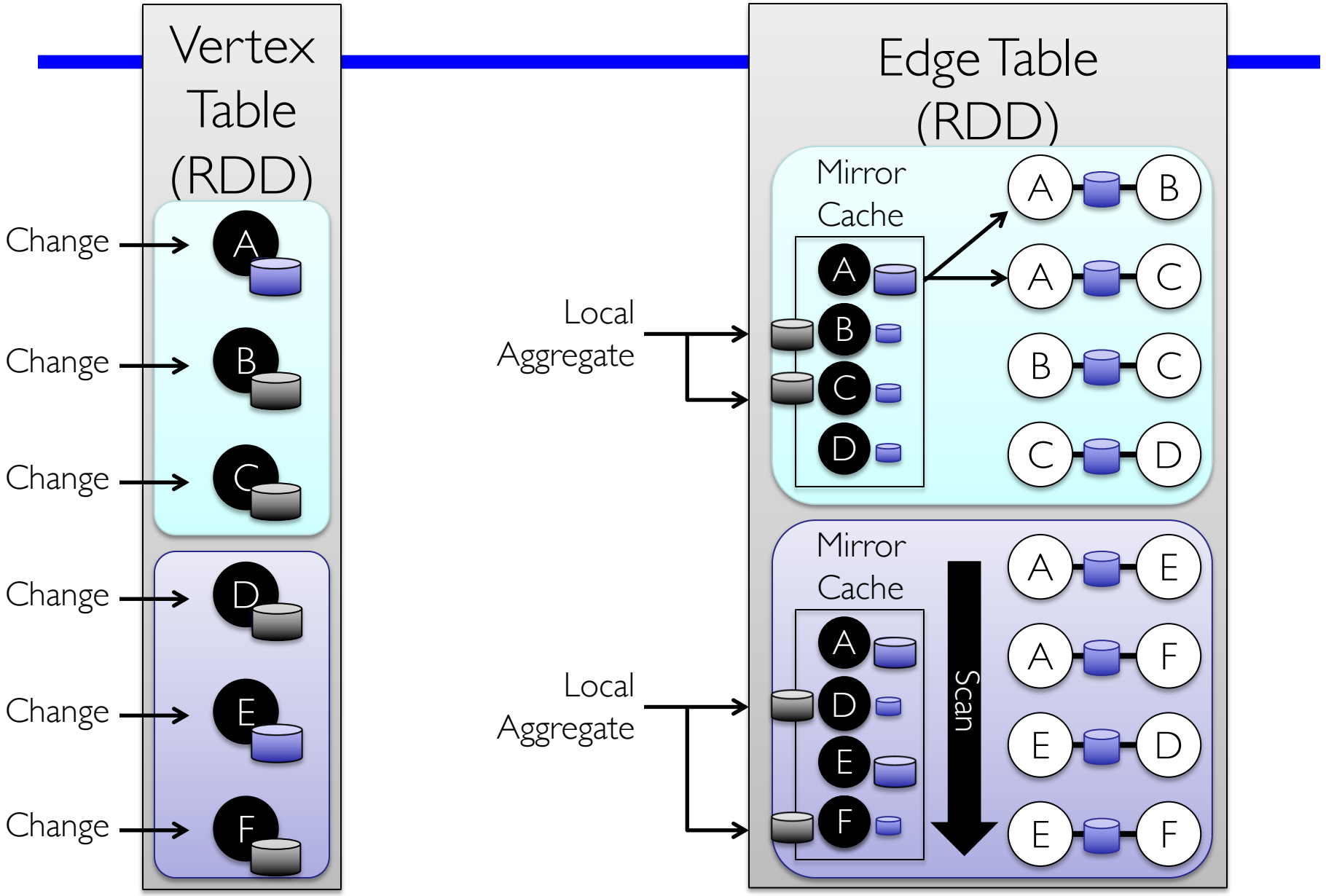
Caching


Vertex Table (RDD)
A
B
C
D
E
F
Edge Table (RDD)
Mirror Cache
A
B
C
D
A B
A C
B C
C D
Mirror Cache
A
D
E
F
A E
A F
E D
E F

Updates


Vertex Table (RDD)
Change
A
B
C
D
Change
E
F
Edge Table (RDD)
Mirror Cache
A
B
C
D
A
B
A
C
B
C
C
D
Mirror Cache
A
D
E
F
Scan
A
E
A
F
E
D
E
F

Updates


Vertex Table (RDD)
Change
A
Change
B
Change
C
Change
D
Change
E
Change
F
Edge Table (RDD)
Local Aggregate
Mirror Cache
A
B
C
D
A
B
A
C
B
C
C
D
Local Aggregate
Mirror Cache
A
D
E
F
Scan
A
E
A
F
E
D
E
F