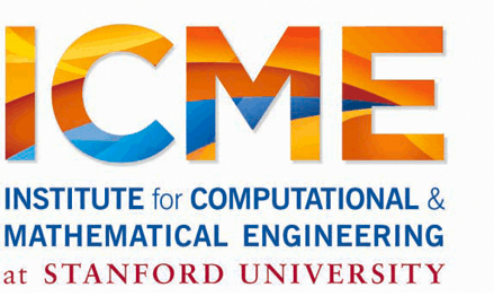
Pregel and GraphX

Reza Zadeh



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

Overview

Graph Computations and Pregel

Introduction to Matrix Computations

Graph Computations and Pregel

Data Flow Models

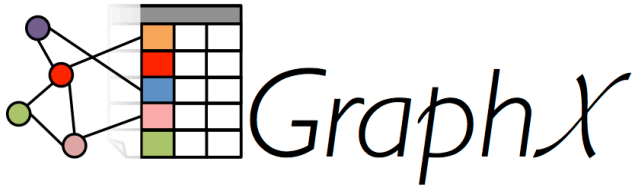
Restrict the programming interface so that the system can do more automatically

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)

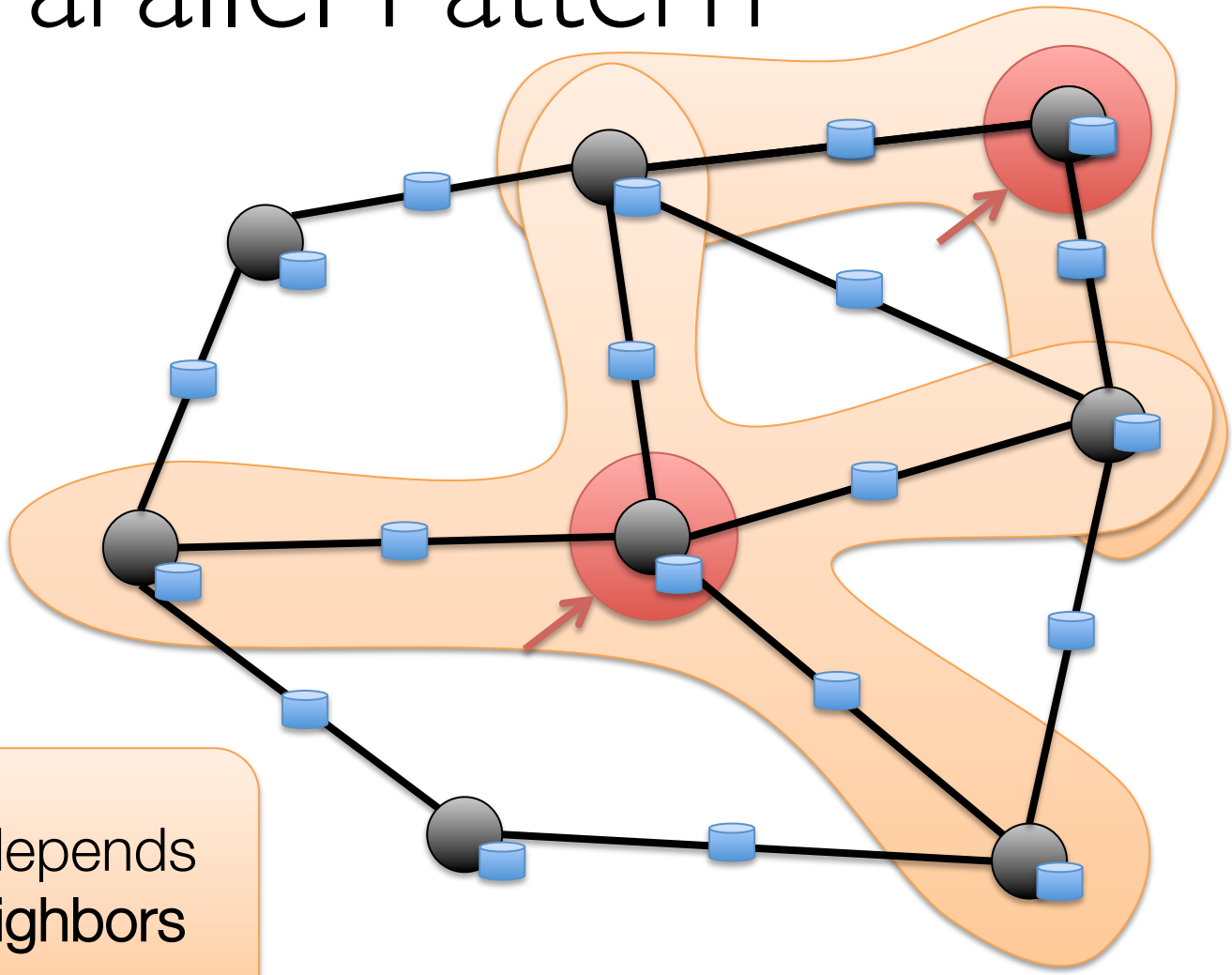
Pregel



Expose specialized APIs to simplify graph programming.

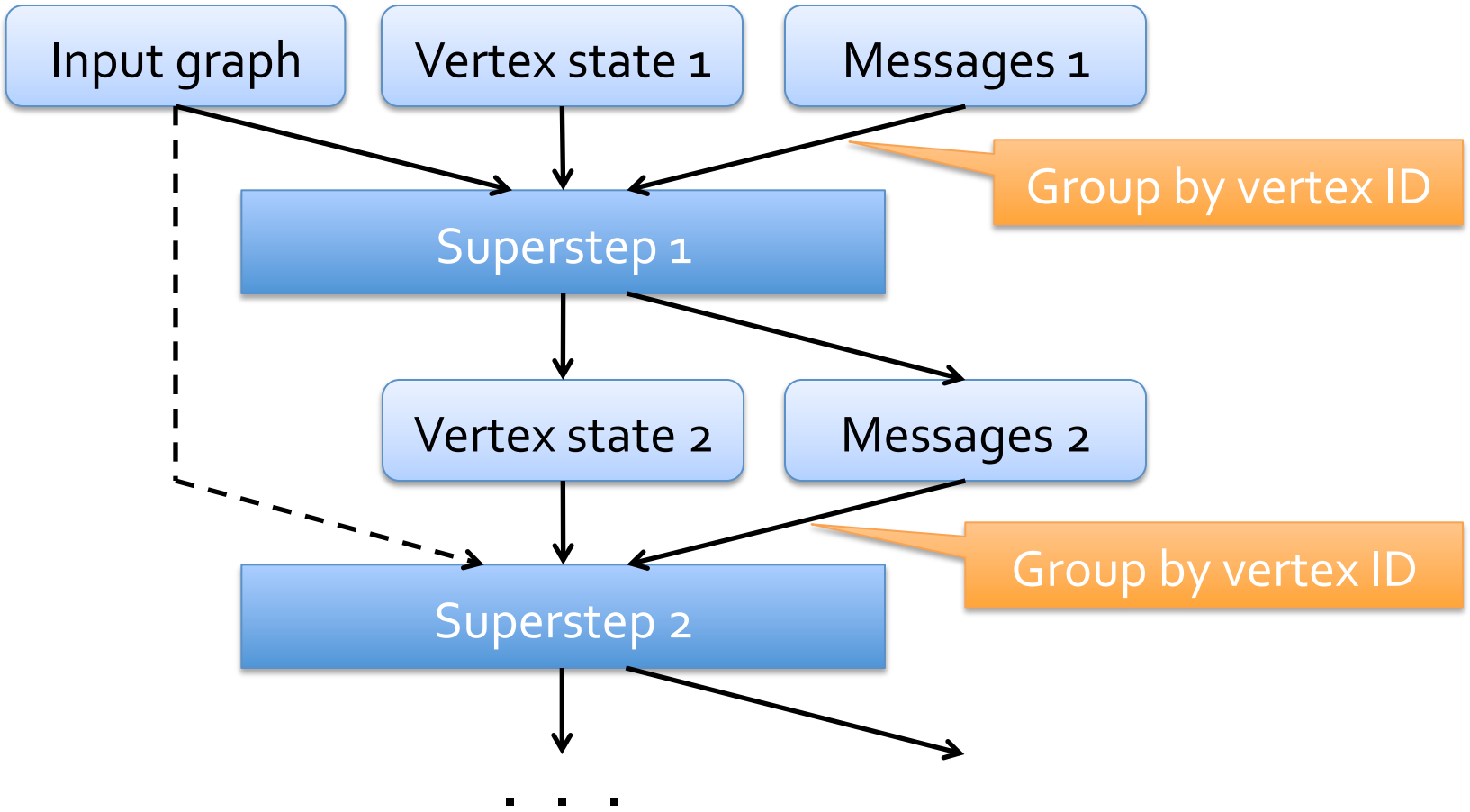
"Think like a vertex"

Graph-Parallel Pattern



Computation depends only on the **neighbors**

Pregel Data Flow



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

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

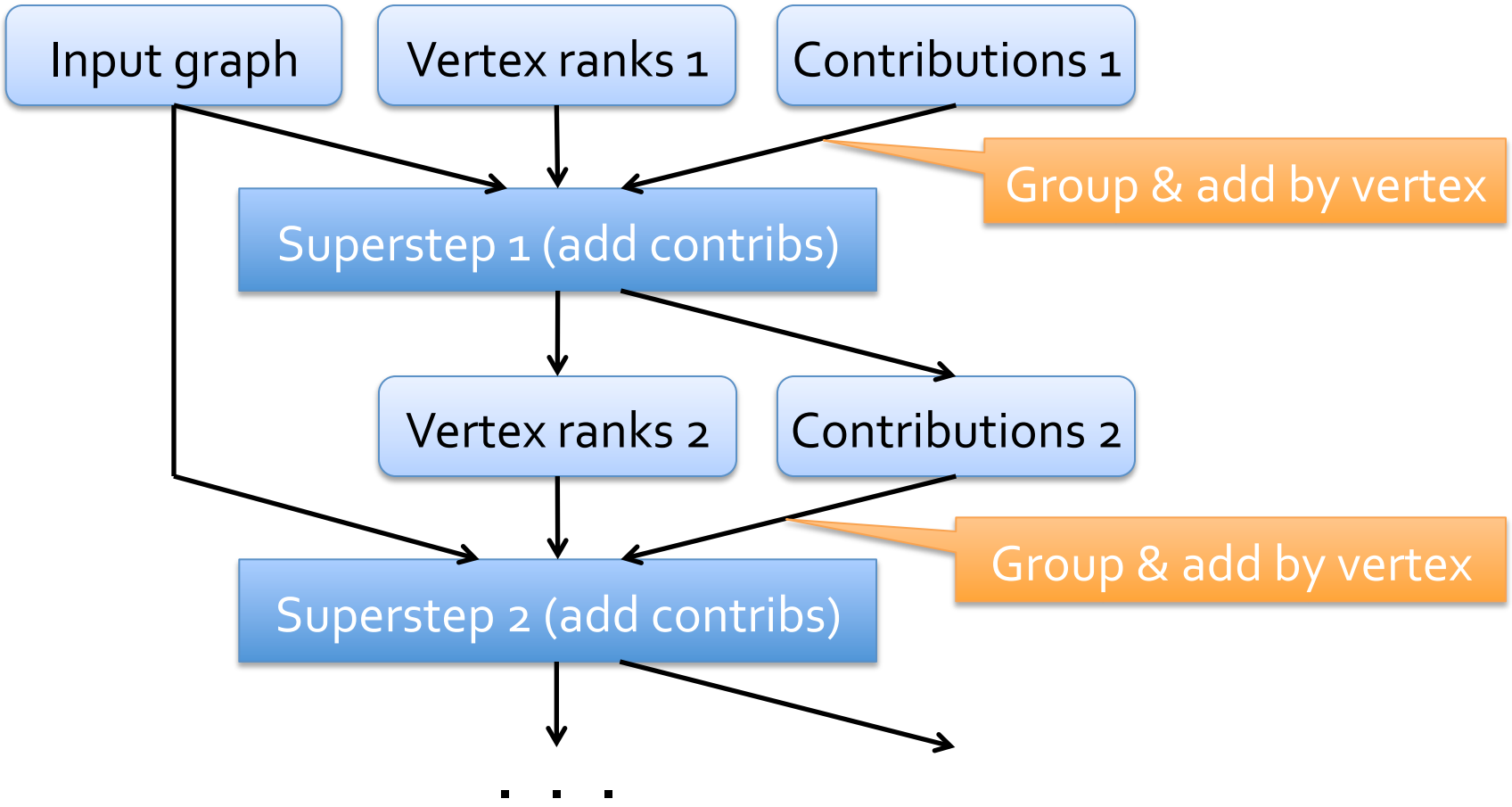
Rank of user *i*

Weighted sum of neighbors' ranks

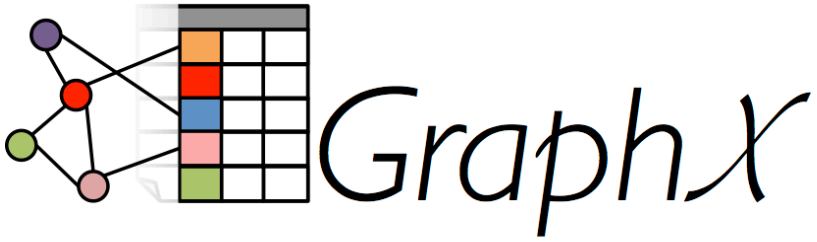
Update ranks in parallel

Iterate until convergence

PageRank in Pregel



GraphX

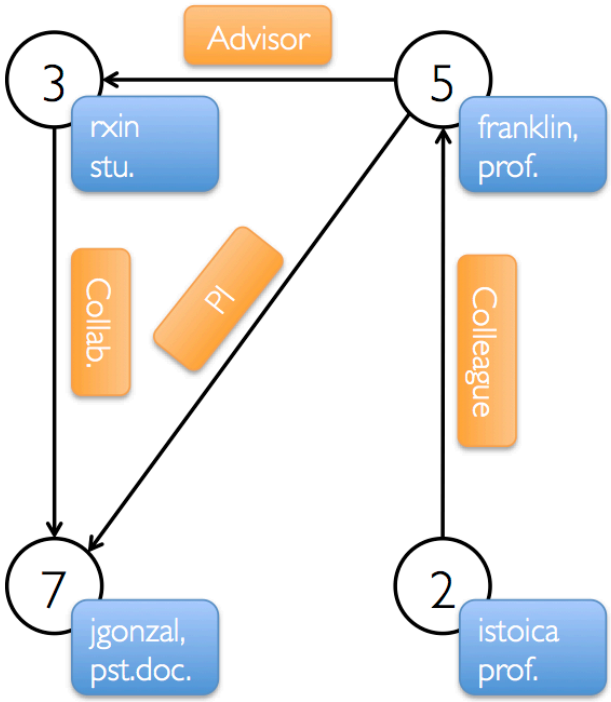



```
class Graph[VD, ED] {
  val vertices: VertexRDD[VD]
  val edges: EdgeRDD[ED]
}
```

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

GraphX: Properties

Property Graph



Vertex Table

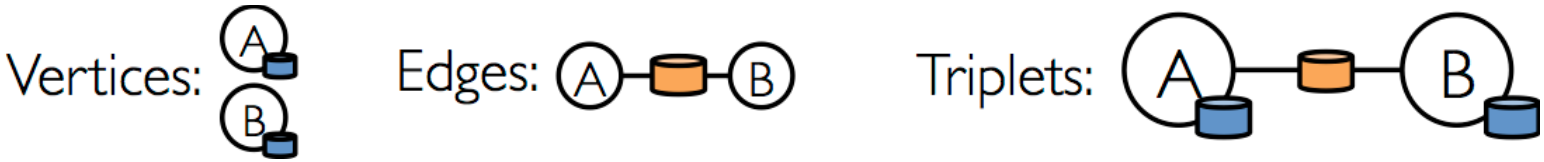
Id	Property (V)
3	(rxin, student)
7	(jgonzal, postdoc)
5	(franklin, professor)
2	(istoica, professor)

Edge Table

SrcId	DstId	Property (E)
3	7	Collaborator
5	3	Advisor
2	5	Colleague
5	7	PI

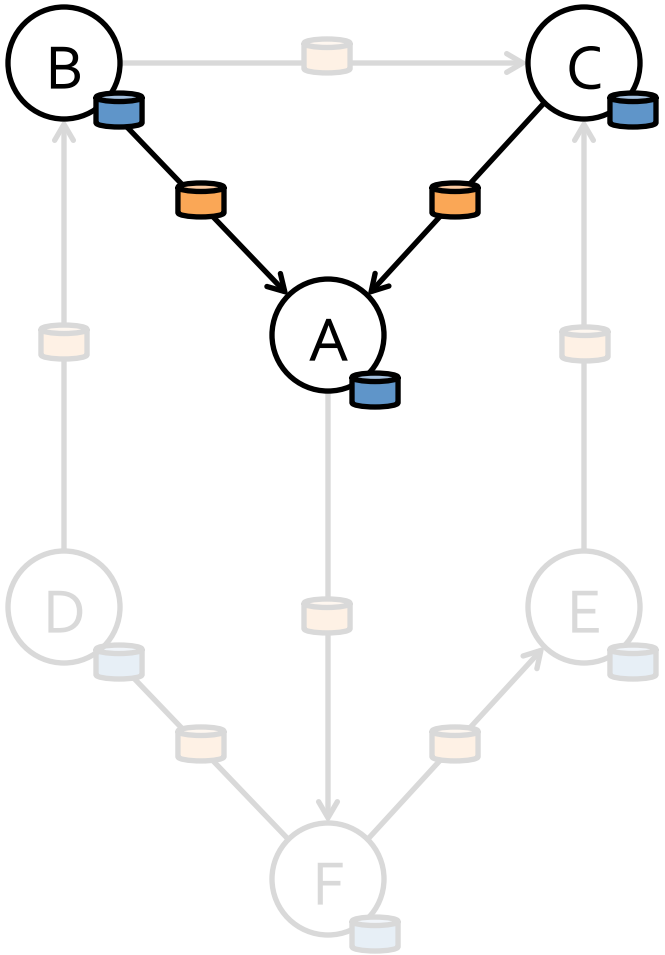
GraphX: Triplets

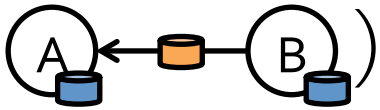
The *triplets* operator joins vertices and edges:

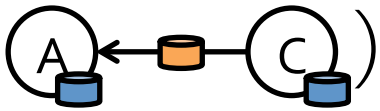


Map Reduce Triplets

Map-Reduce for each vertex



mapF() → 

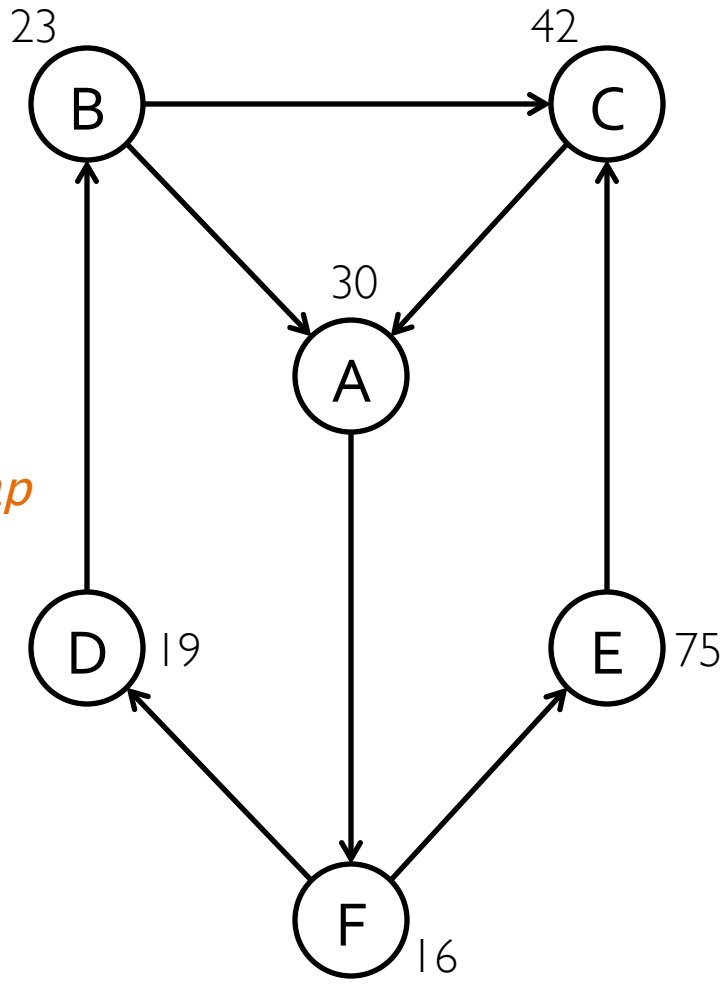
mapF() → 

reduceF() → 

Example: Oldest Follower

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

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



Summary of Operators

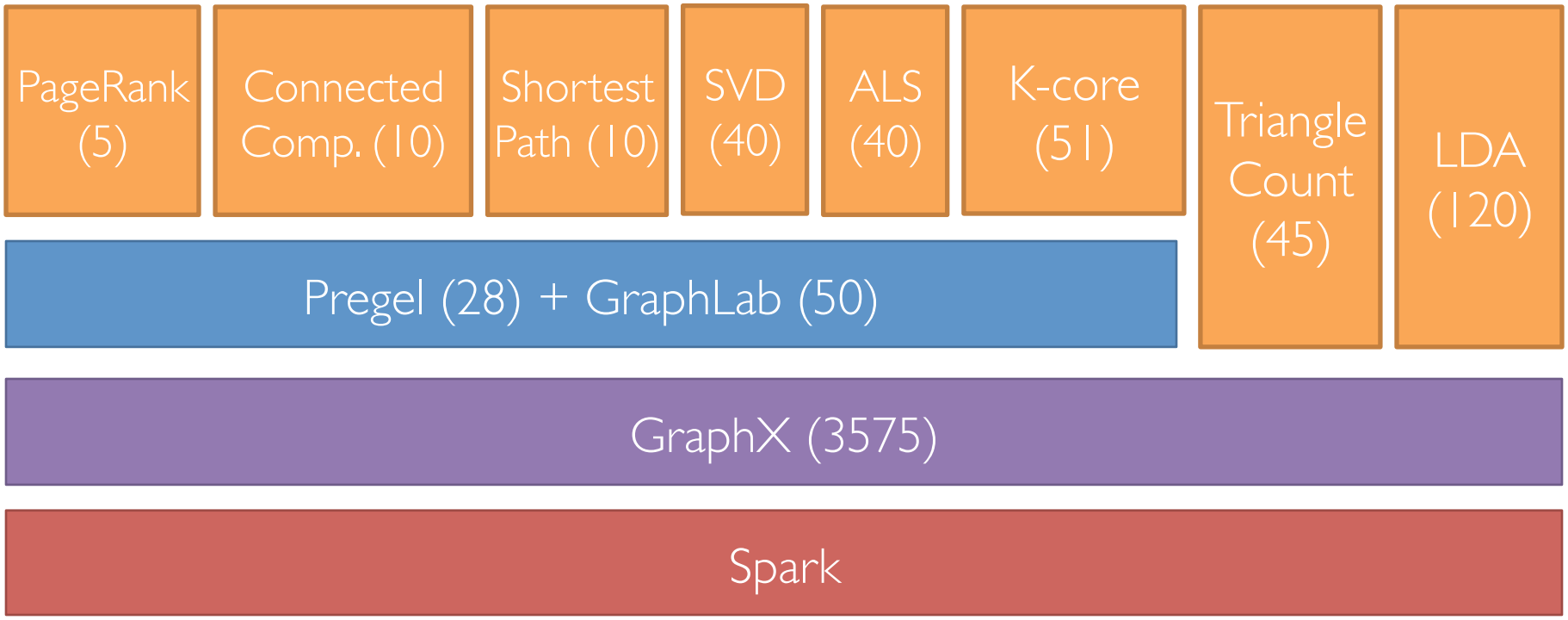
All operations:

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

Pregel API:

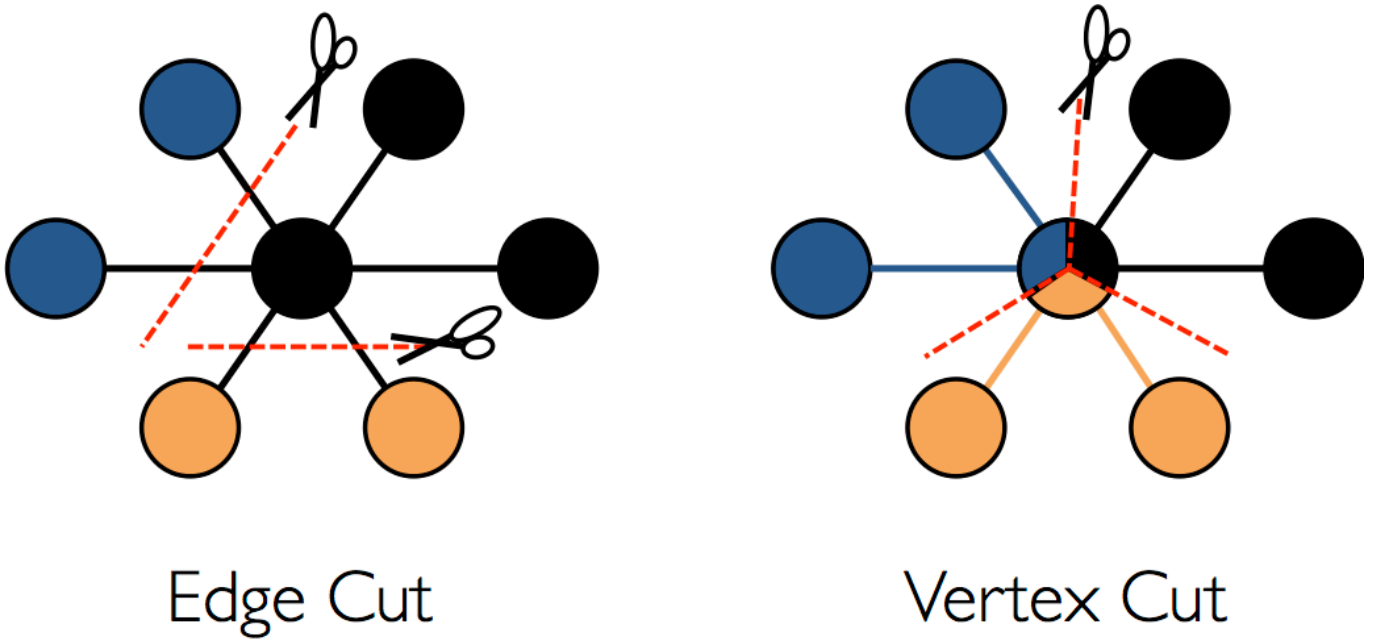
https://spark.apache.org/docs/latest/graphx-programming-guide.html#pregel-api

The GraphX Stack (Lines of Code)

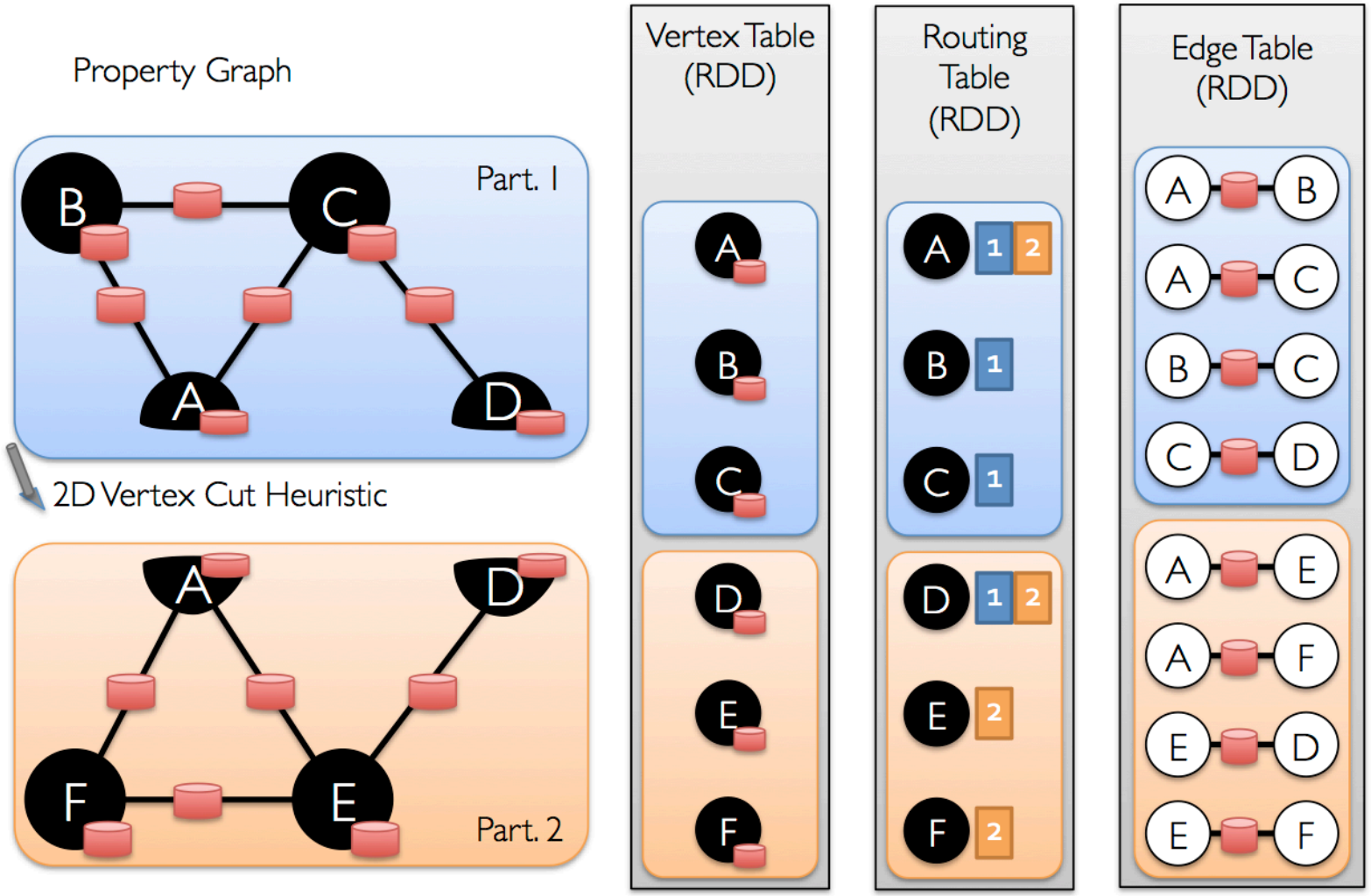


Optimizations

Overloaded vertices have their work distributed



Optimizations



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)