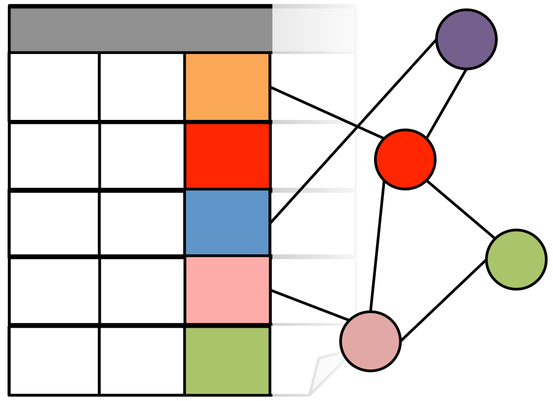
GraphX



Graph Analytics in Spark

Ankur Dave
Graduate Student, UC Berkeley AMPLab

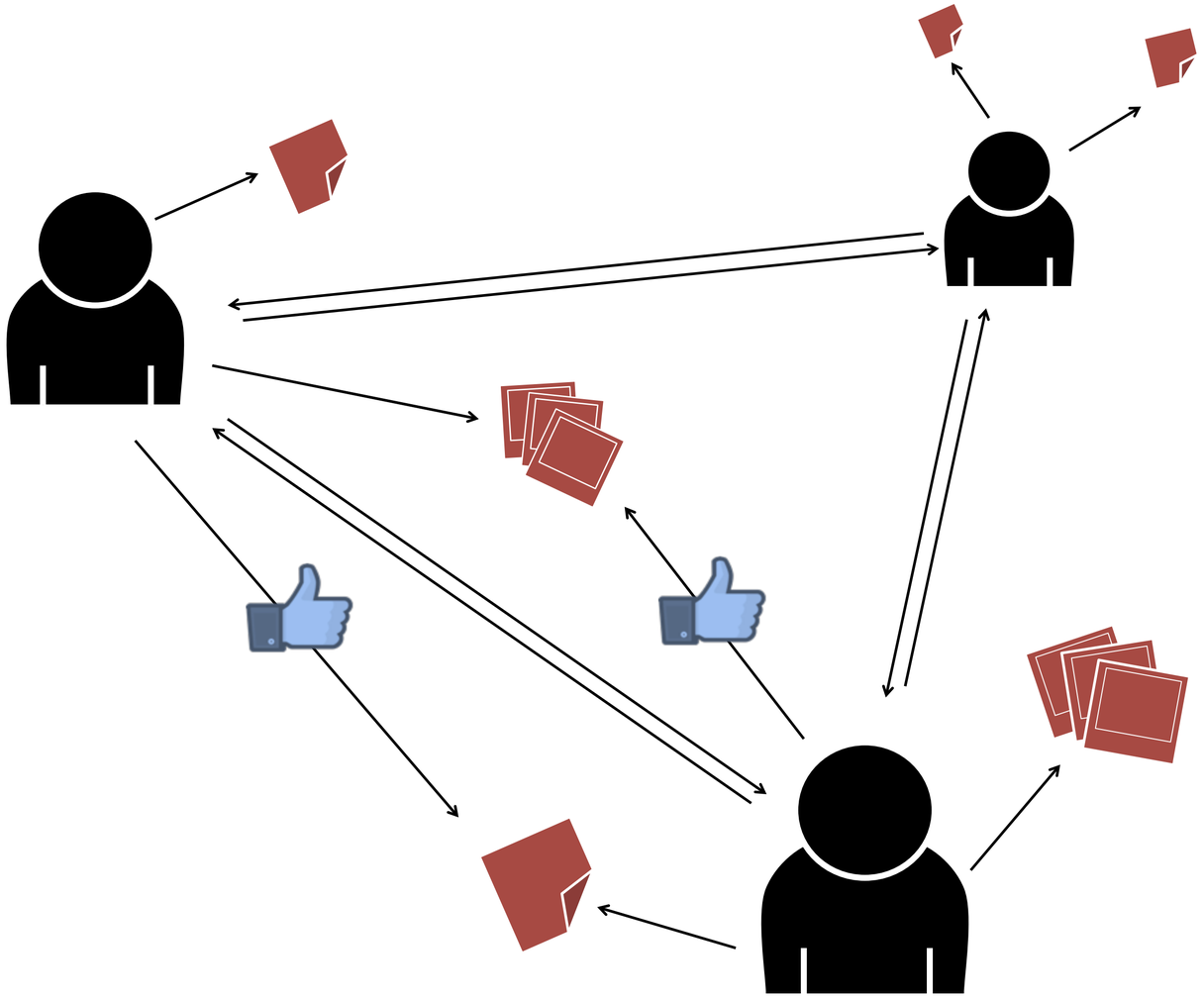
Joint work with Joseph Gonzalez, Reynold Xin, Daniel Crankshaw, Michael Franklin, and Ion Stoica

amplab UC BERKELEY

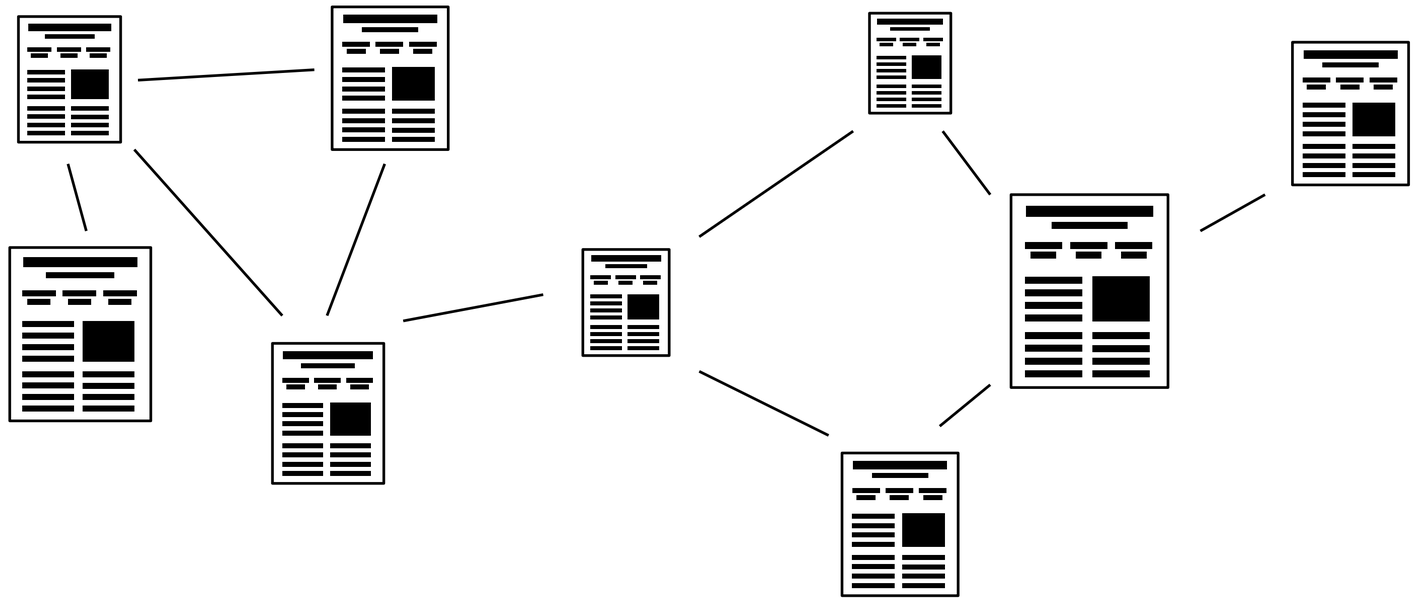
Spark SUMMIT EAST

Graphs

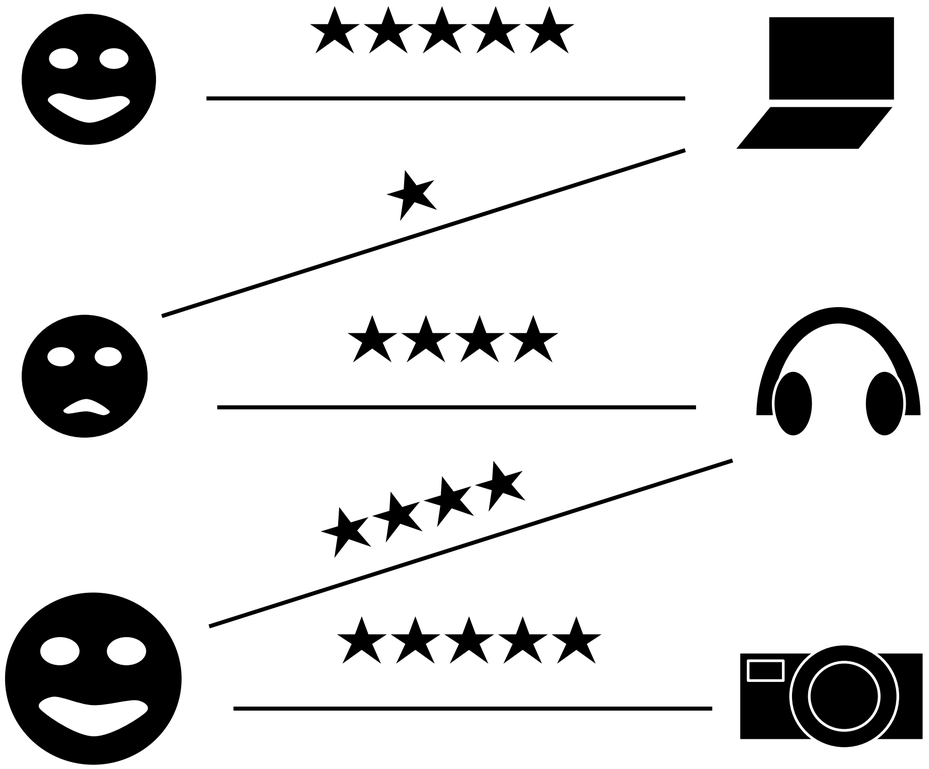
Social Networks



Web Graphs

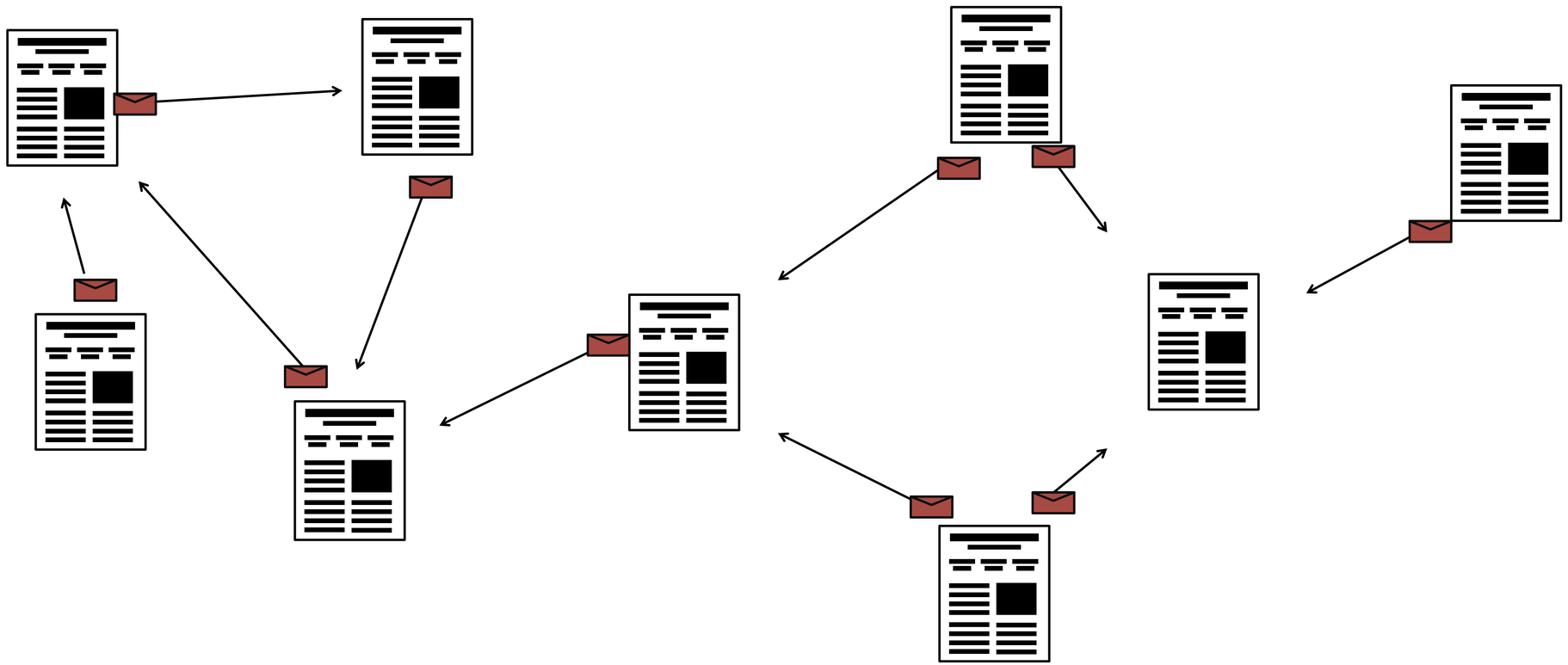


User-Item Graphs

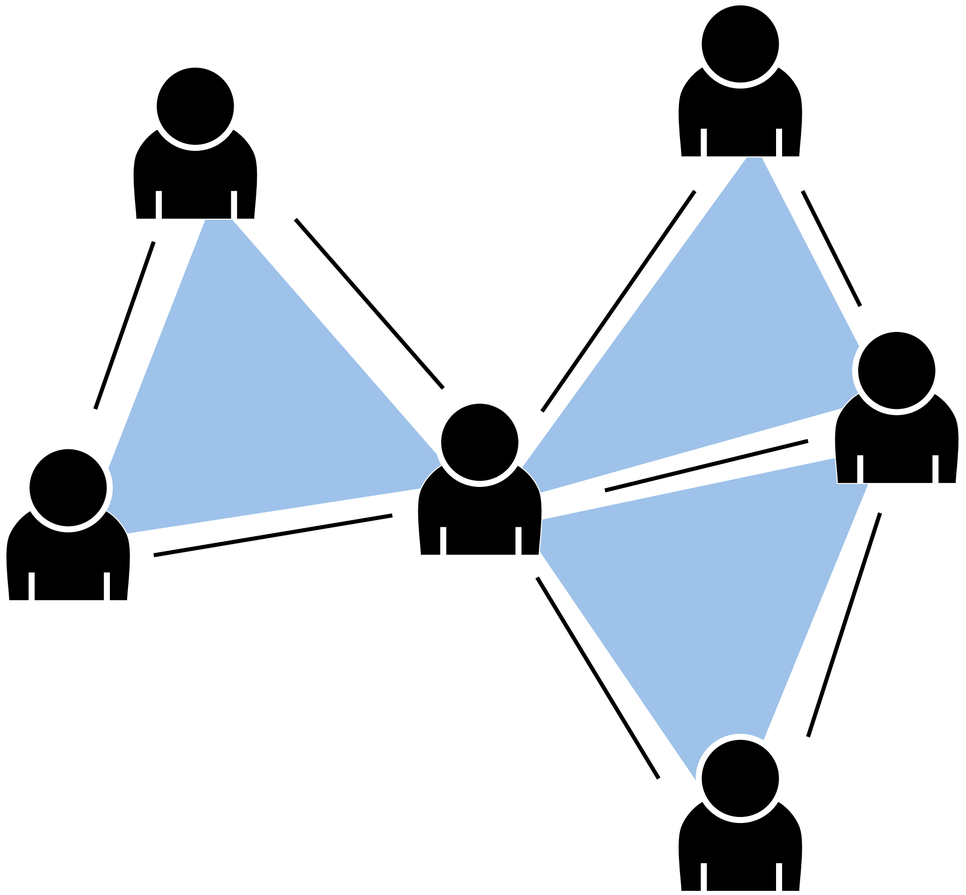


Graph Algorithms

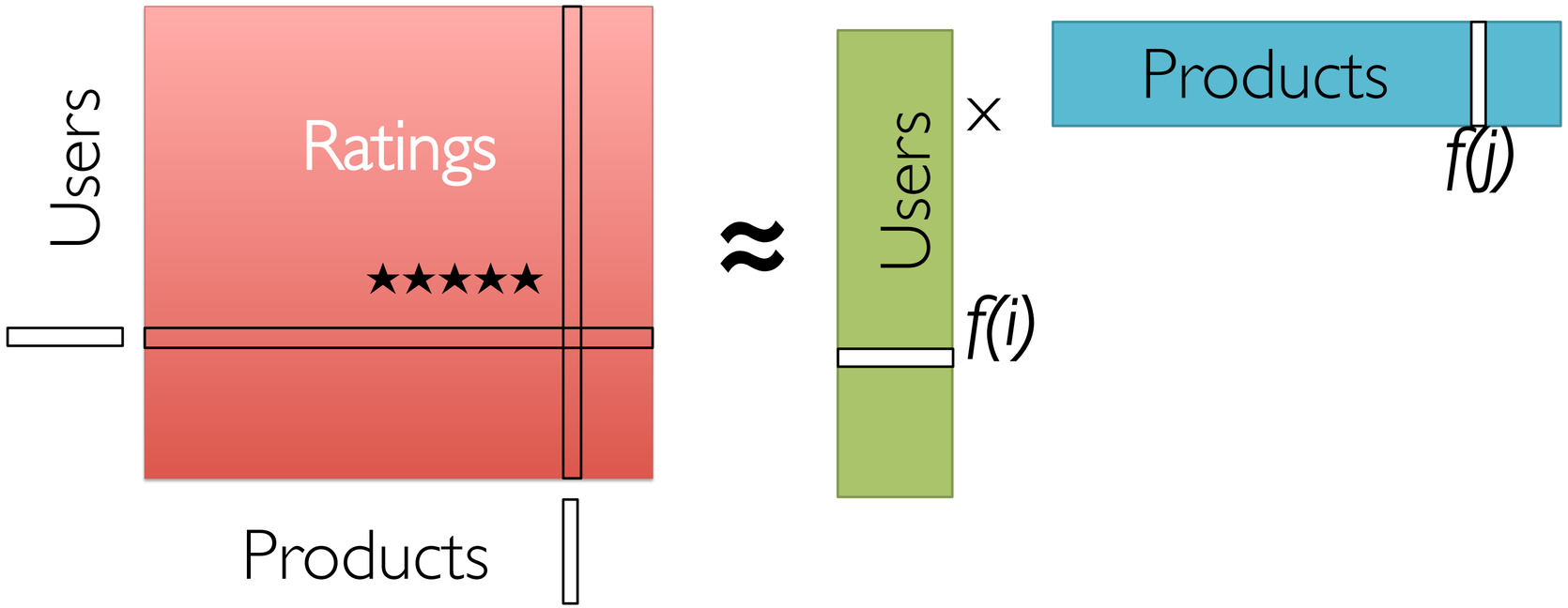
PageRank



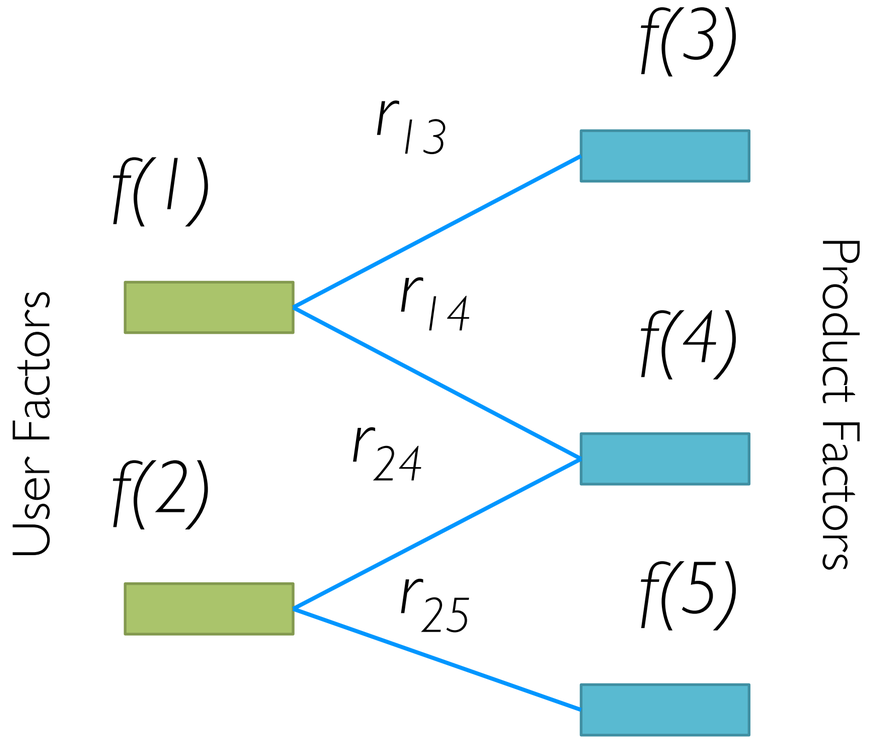
Triangle Counting



Collaborative Filtering

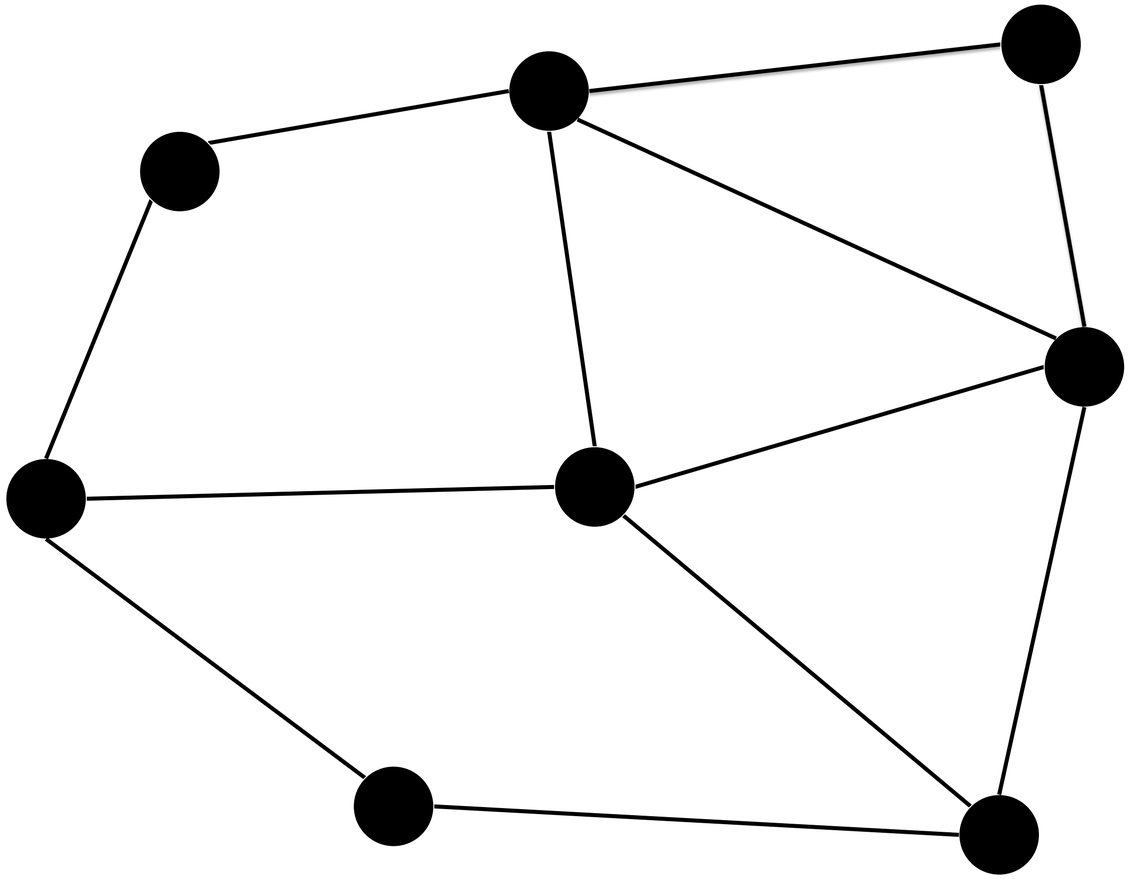


Collaborative Filtering

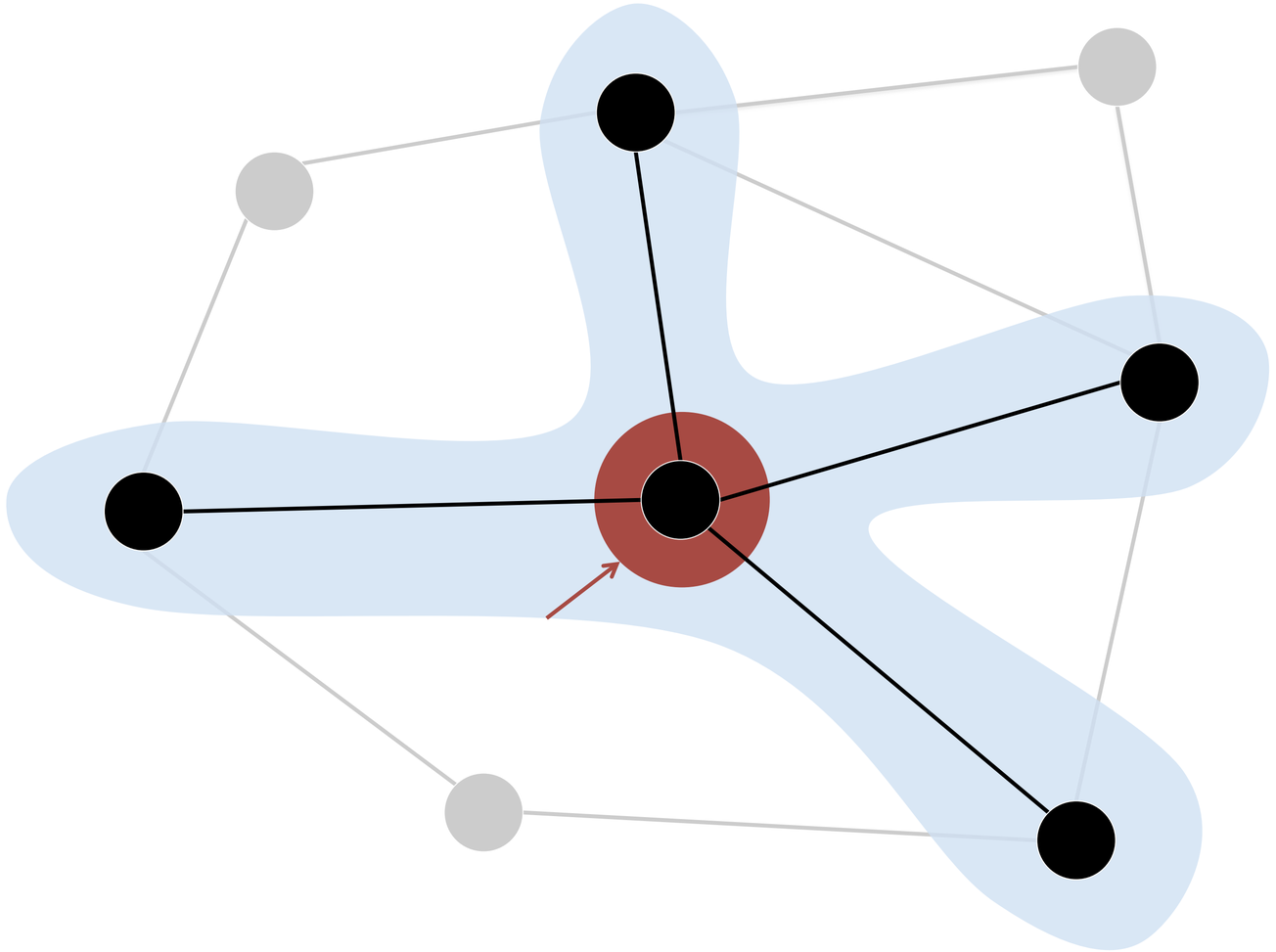


$$f[i] = \arg\min_{w \in \mathbb{R}^d} \sum_{j \in \mathrm{Nbrs}(i)} \left(r_{ij} - w^T f[j]\right)^2 + \lambda ||w||_2^2$$

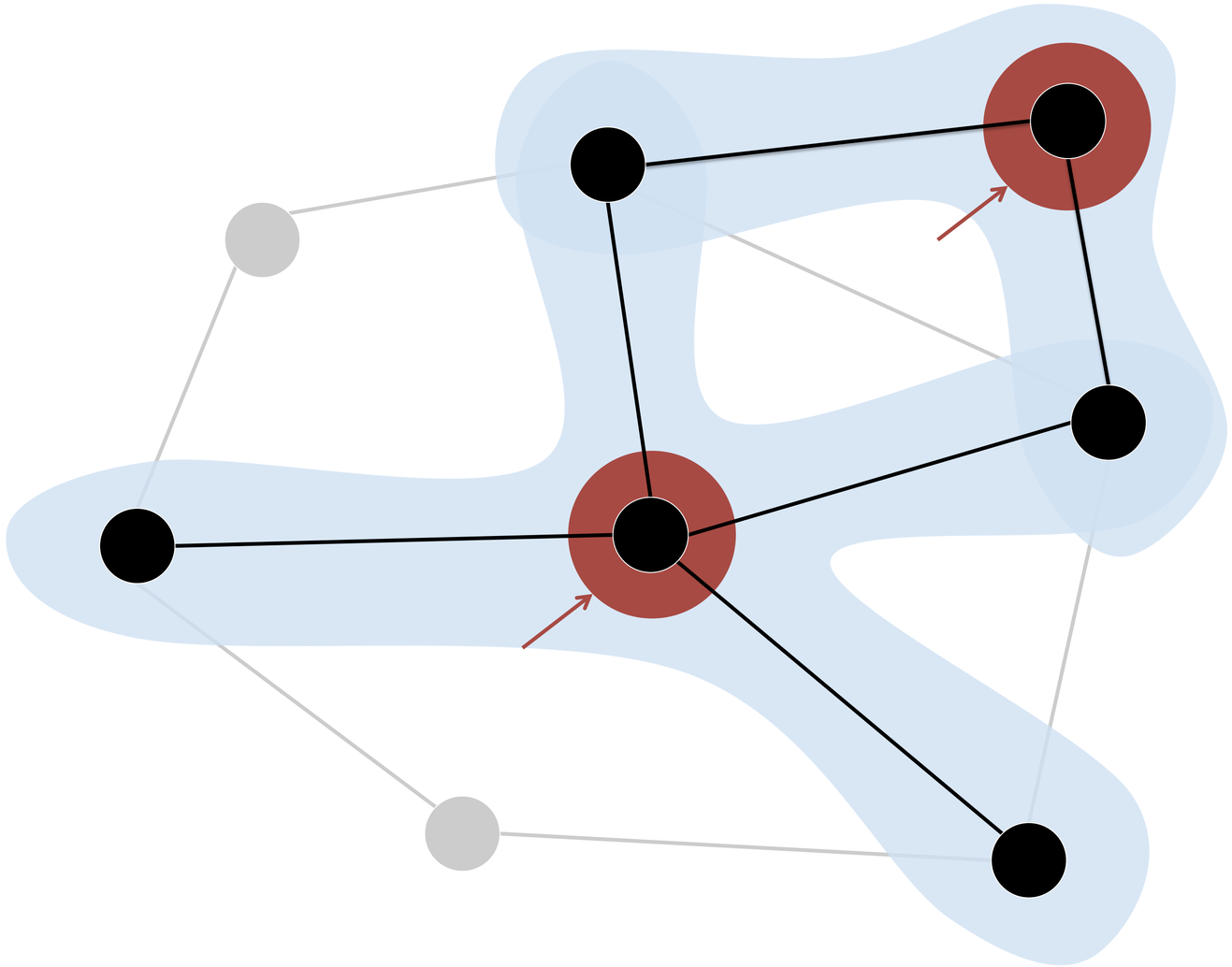
The Graph-Parallel Pattern



The Graph-Parallel Pattern



The Graph-Parallel Pattern



Many Graph-Parallel 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

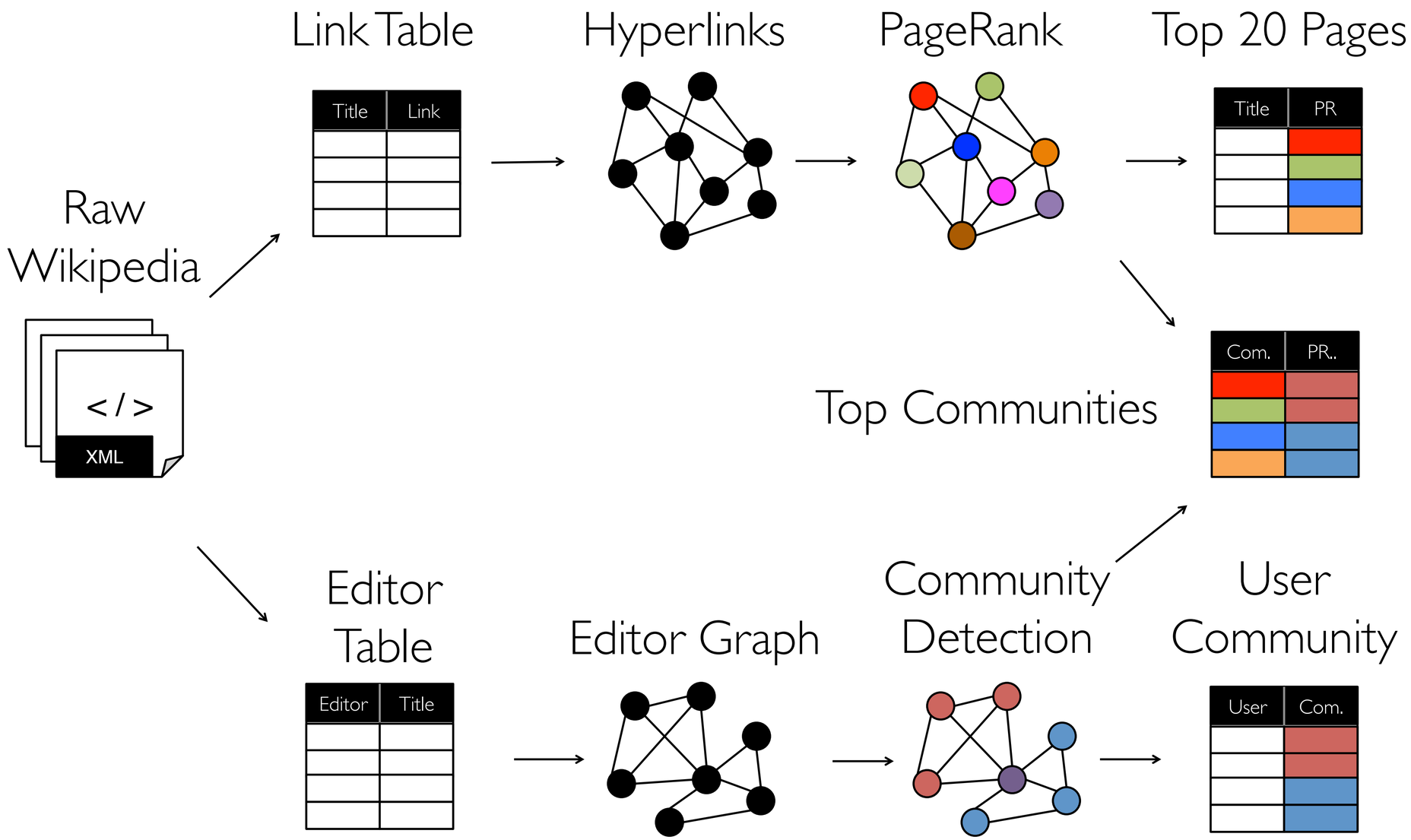
High-Degree Vertices

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



Complex Pipelines

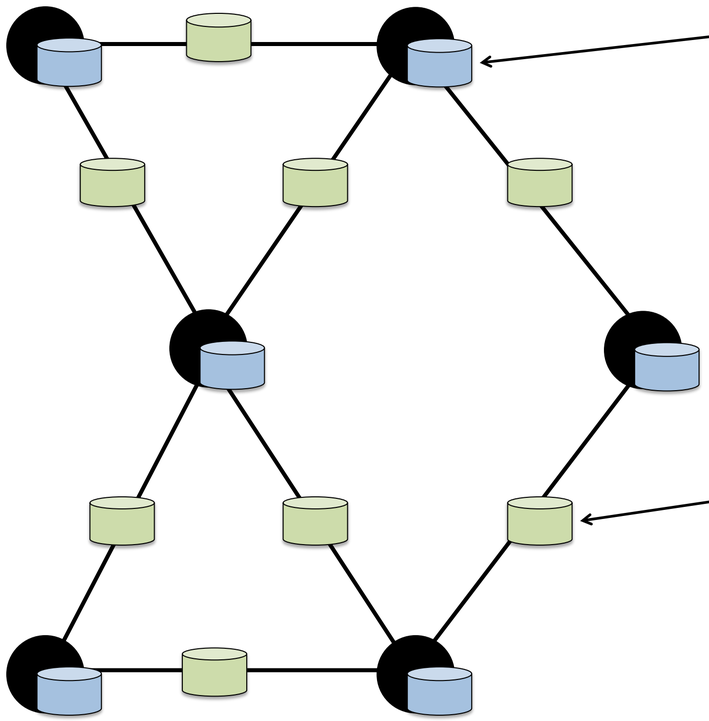
Solution: Embed graph processing within a table-oriented system (Spark)

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:

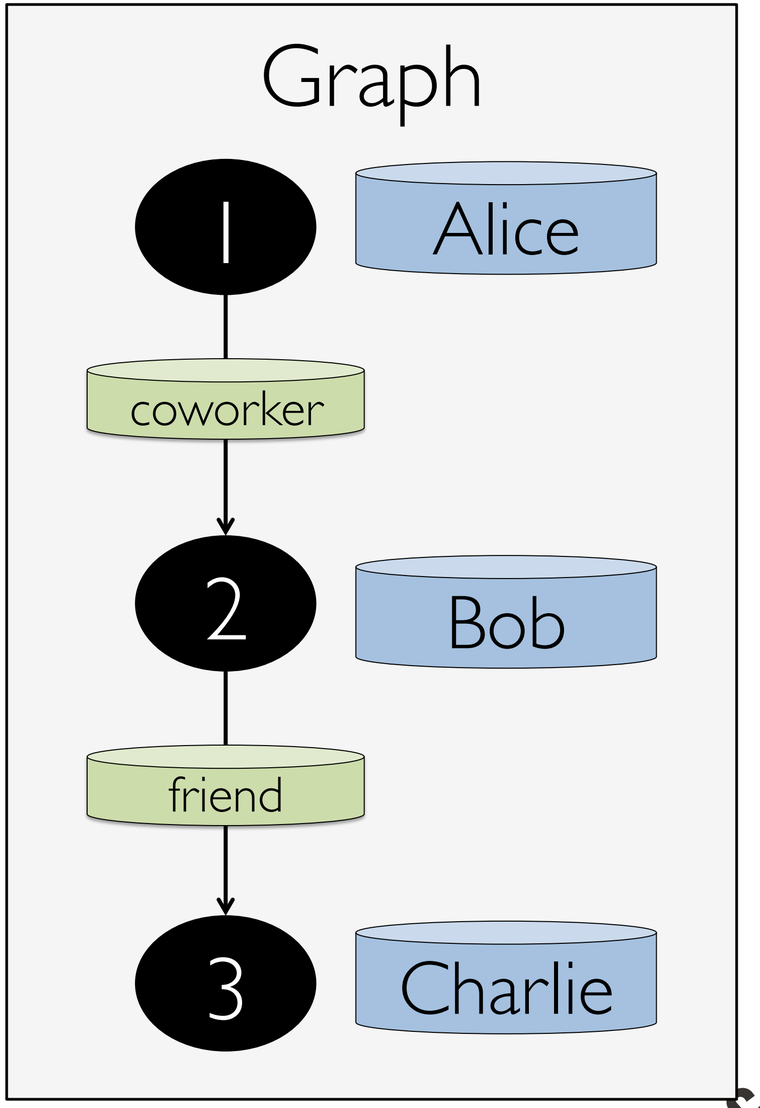
- User Profile
- Current PageRank Value

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)
```



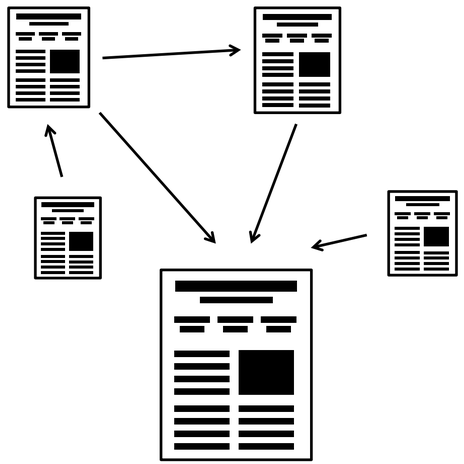
Graph Operations (Scala)

```
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)]
```

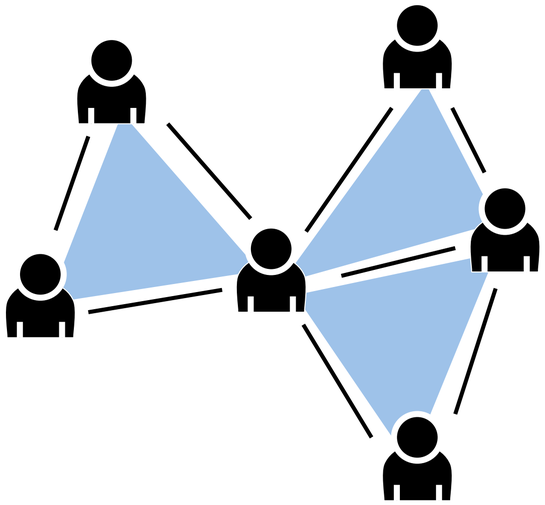
Built-in Algorithms (Scala)

```
    // 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
}
```

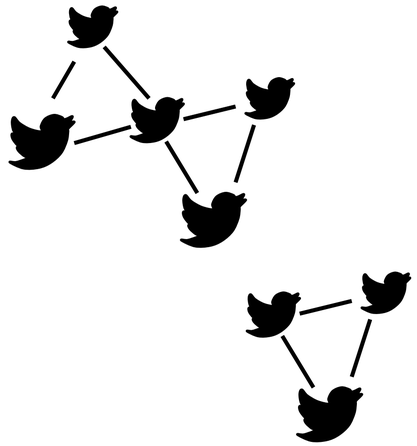
PageRank



Triangle Count

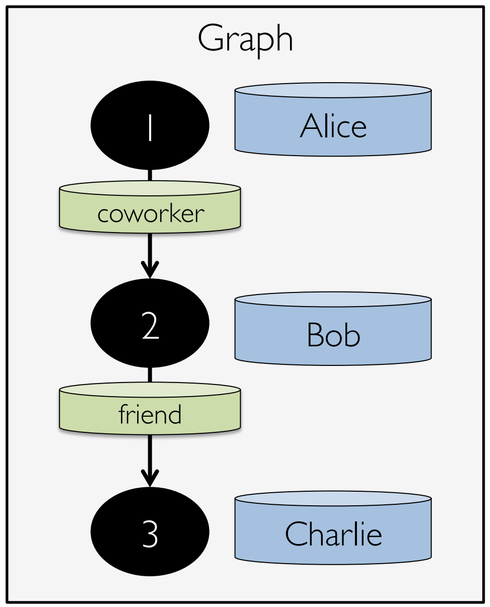


Connected Components



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)
```



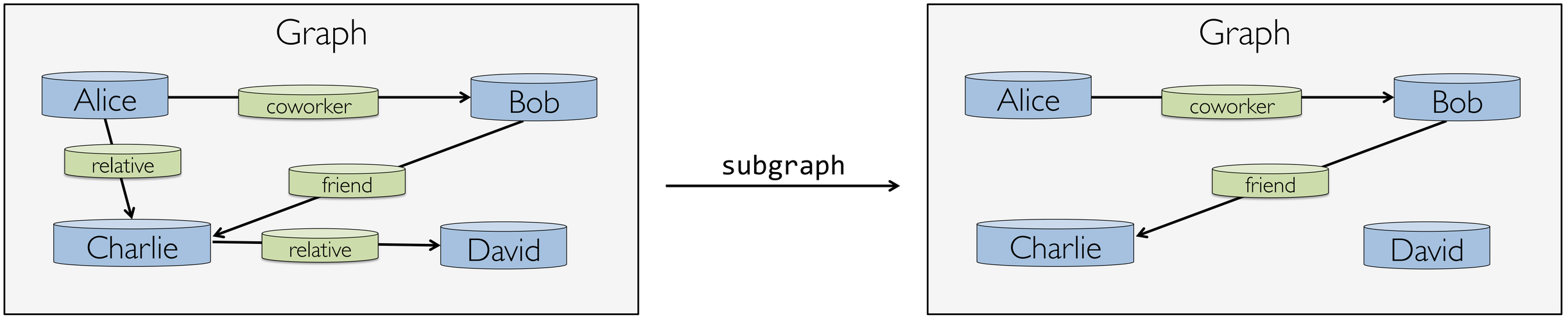
triplets

RDD

srcAttr	dstAttr	attr
Alice	coworker	Bob
Bob	friend	Charlie

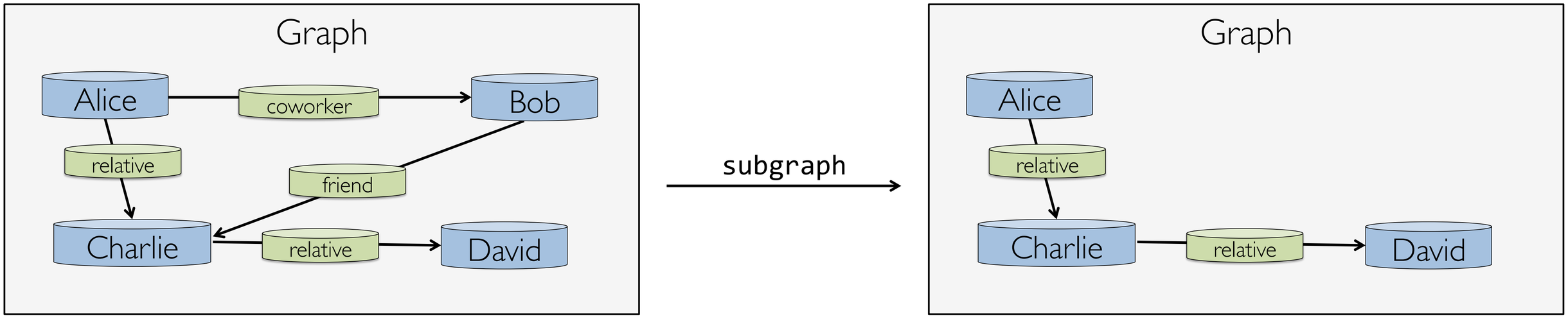
The `subgraph` transformation

```
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

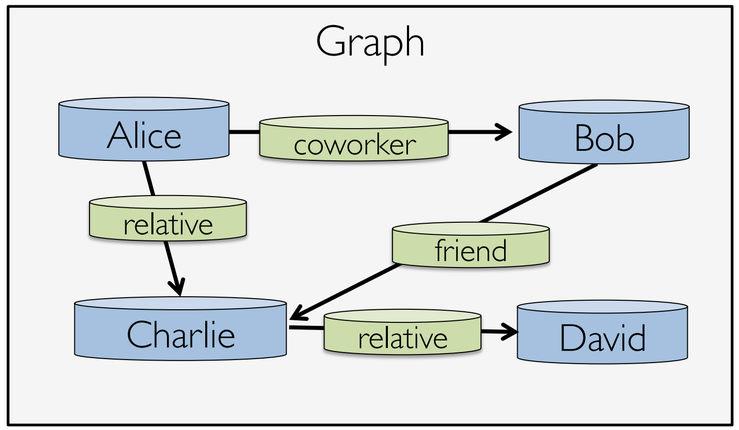
```
class Graph[VD, ED] {
  def subgraph(epred: EdgeTriplet[VD, ED] => Boolean,
               vpred: (VertexId, VD) => Boolean): Graph[VD, ED]
}

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`

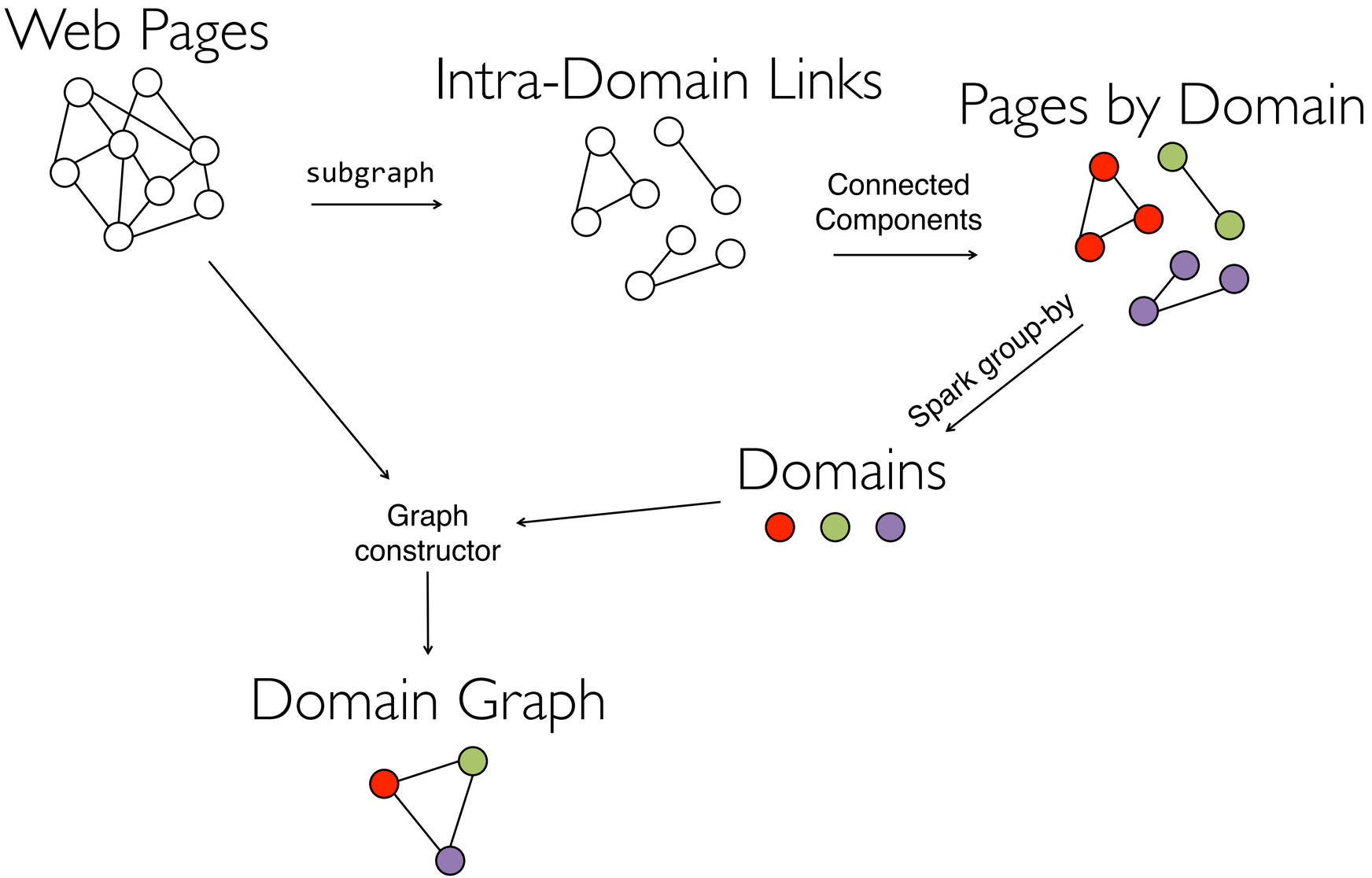


`aggregateMessages`

RDD

vertex id	degree
Alice	2
Bob	2
Charlie	3
David	1

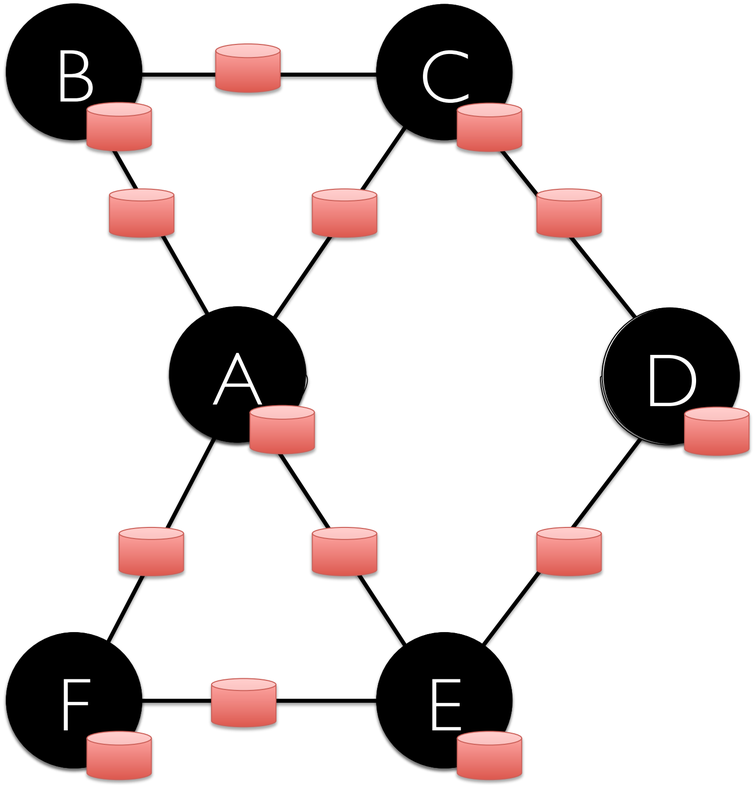
Example: Graph Coarsening



How GraphX Works

Storing Graphs as Tables

Property Graph



Vertex Table (RDD)

Machine 1

A

B

C

Machine 2

D

E

F

Edge Table (RDD)

A B

A C

B C

C D

A E

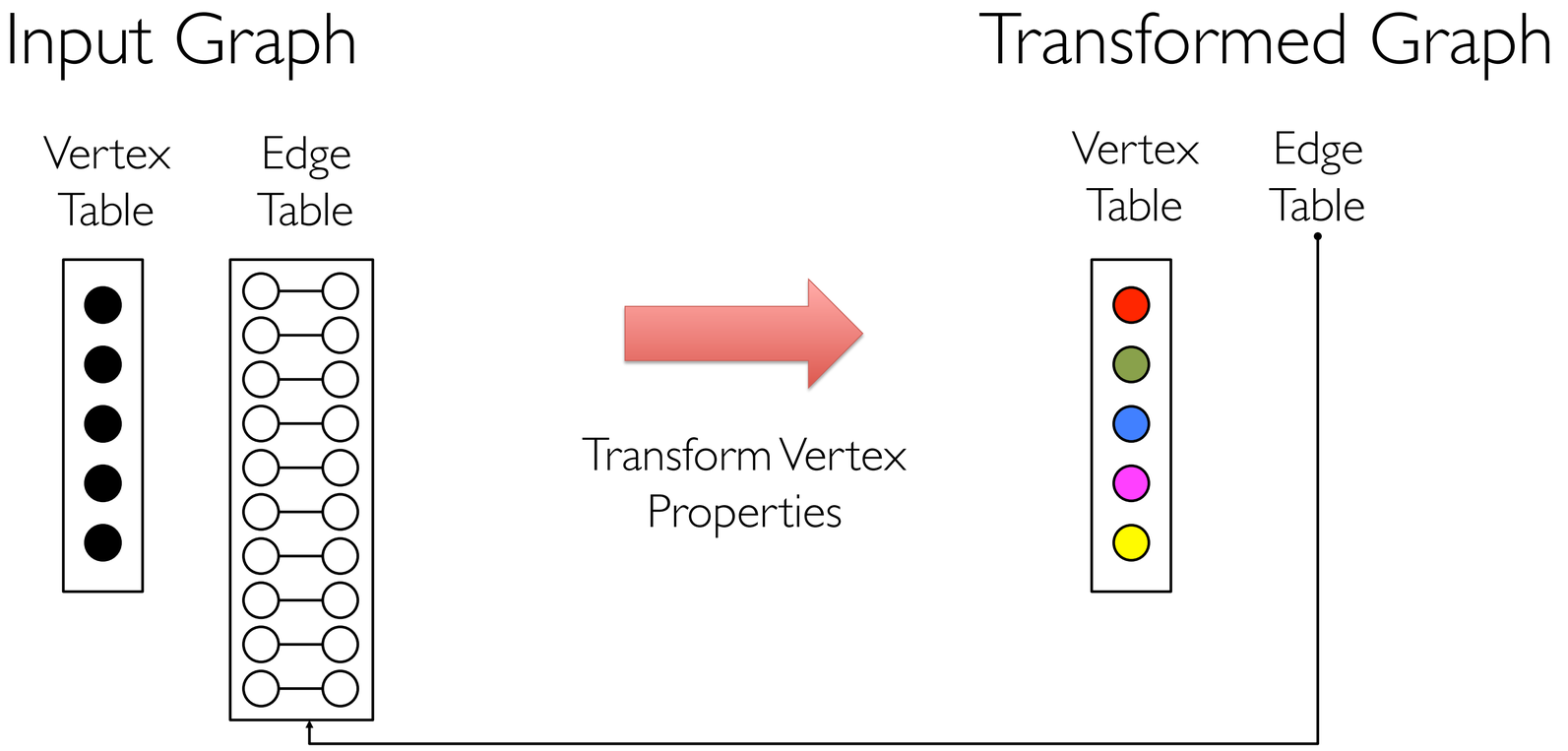
A F

E D

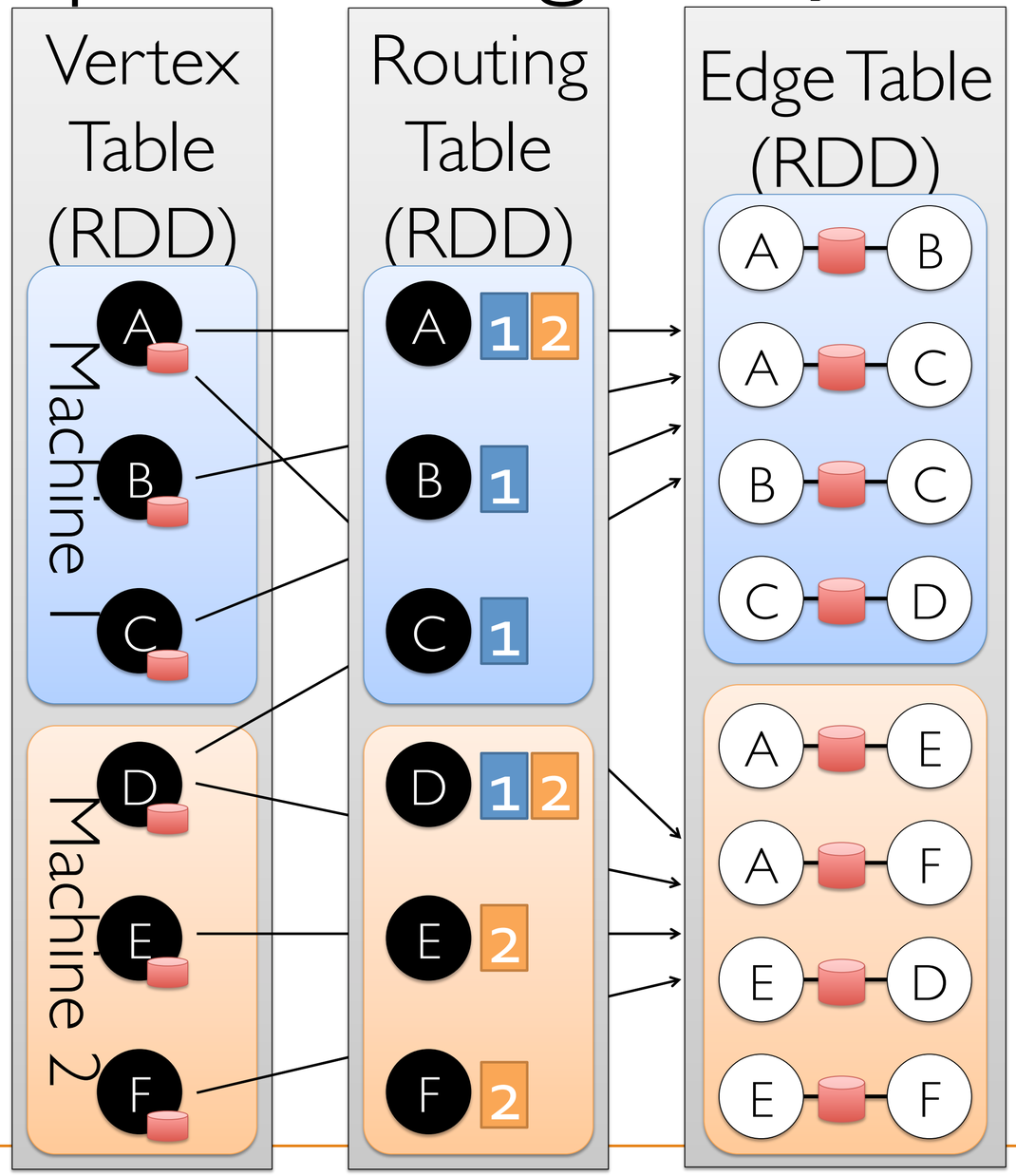
E F

Simple Operations

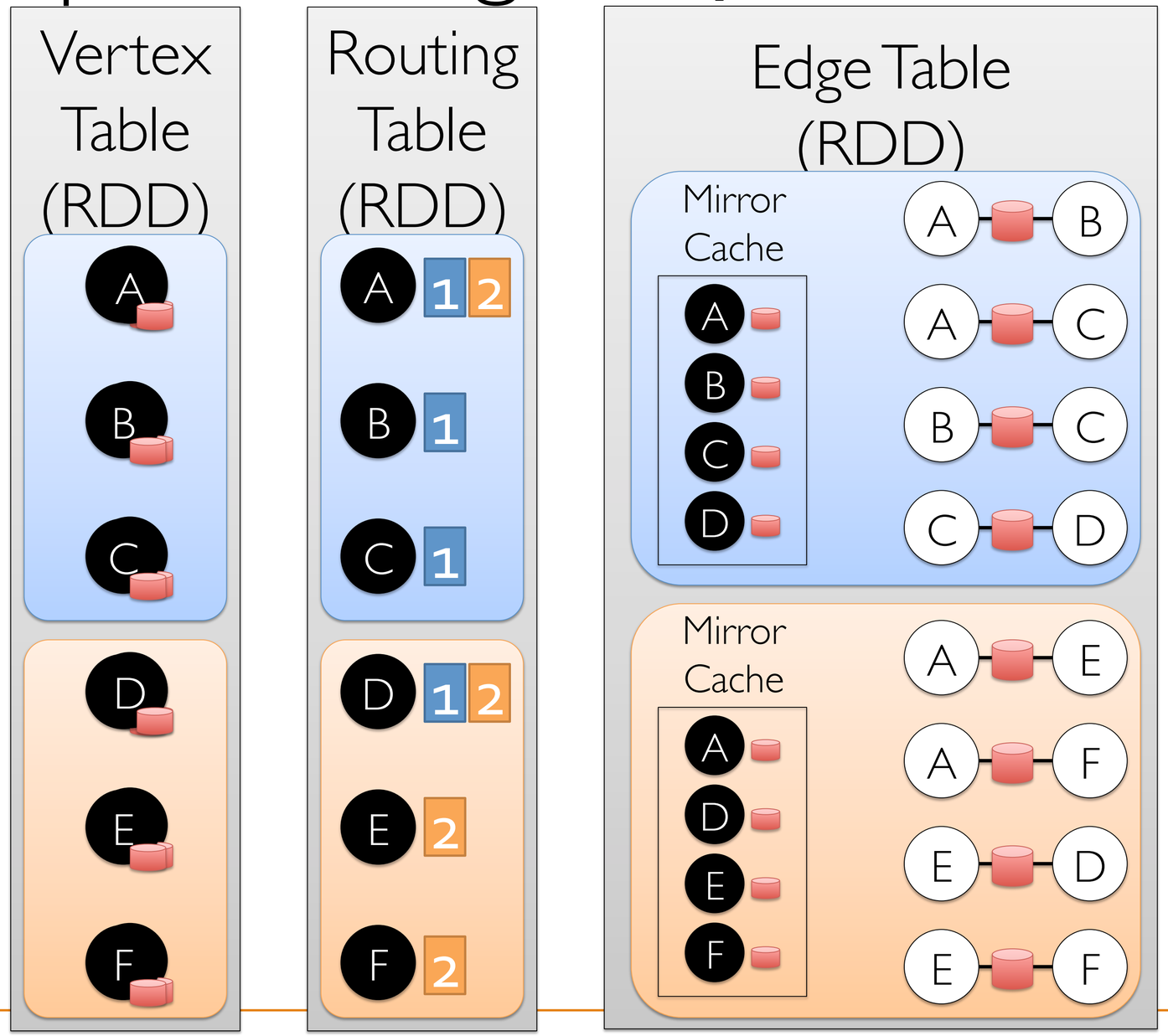
Reuse vertices or edges across multiple graphs



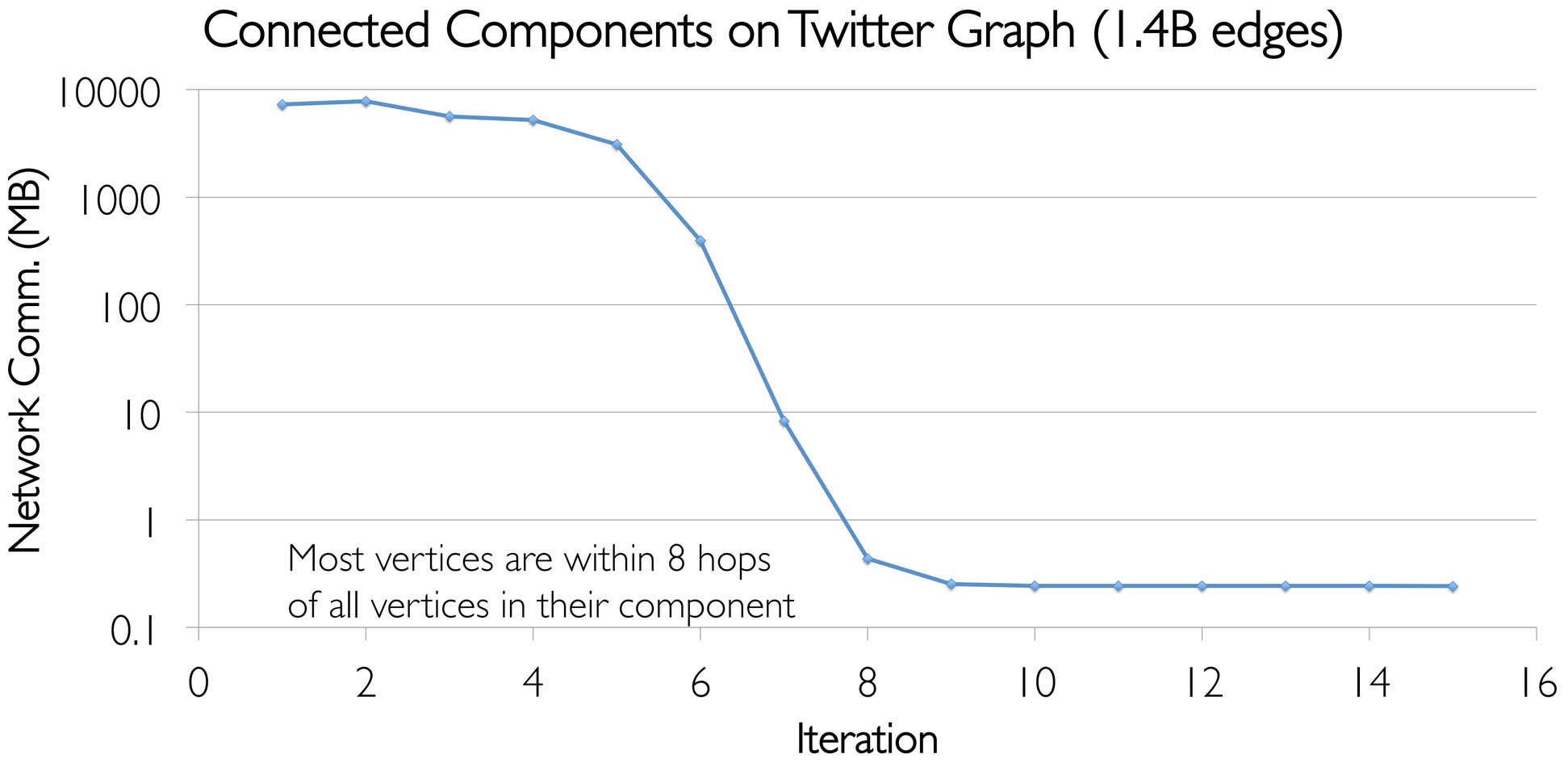
Implementing `triplets`



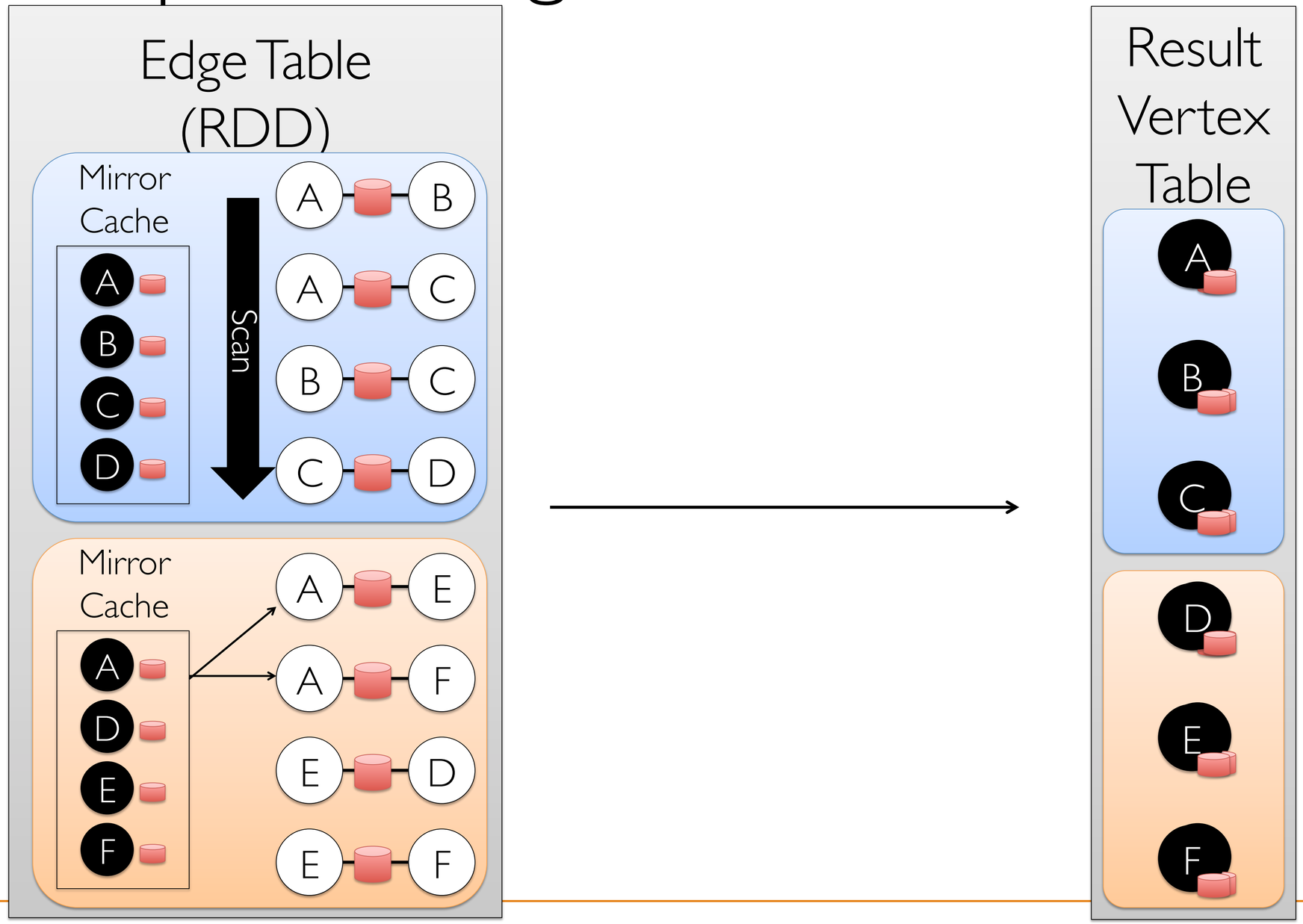
Implementing `triplets`



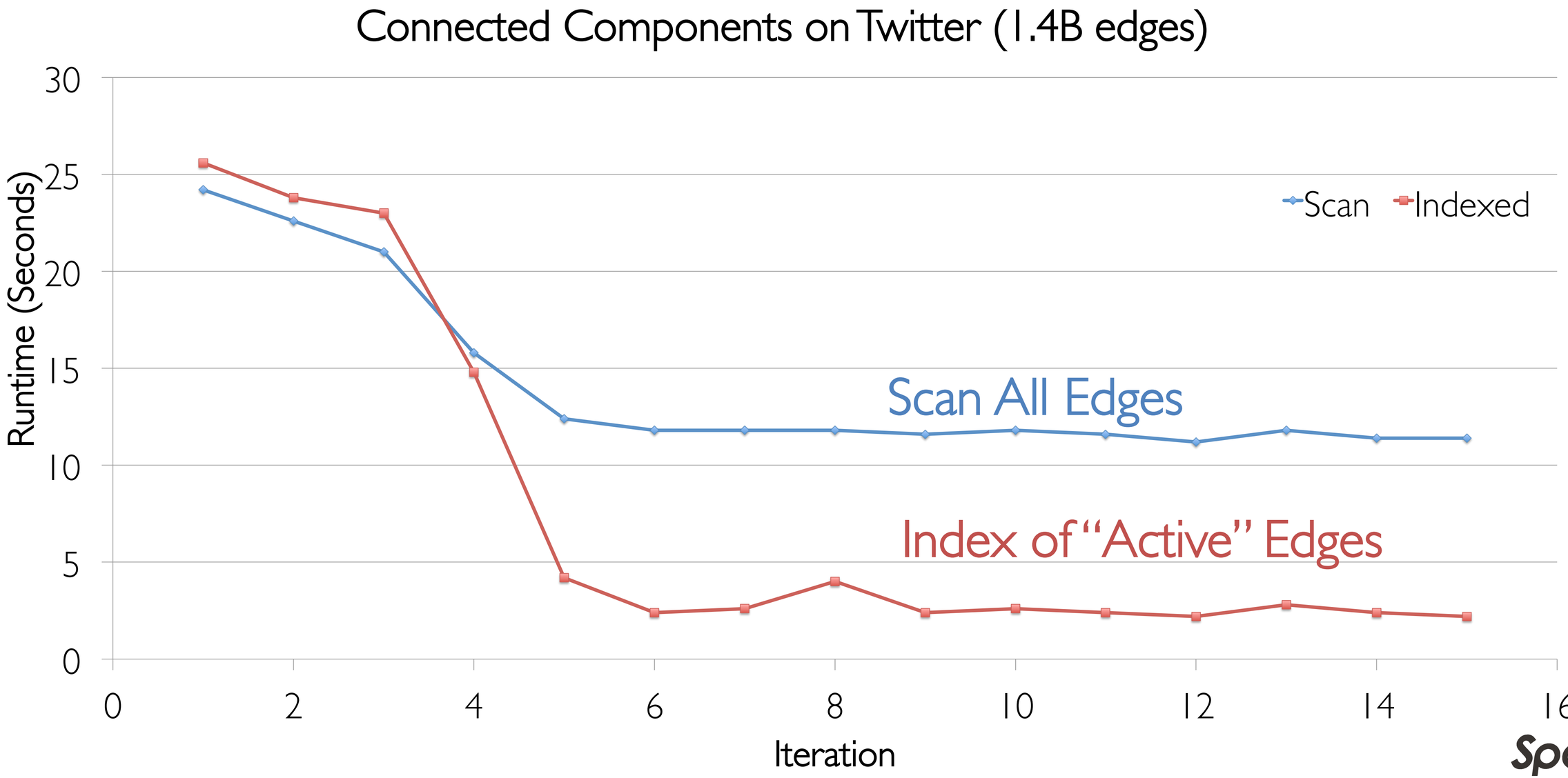
Reduction in Communication Due to Cached Updates



Implementing `aggregateMessages`

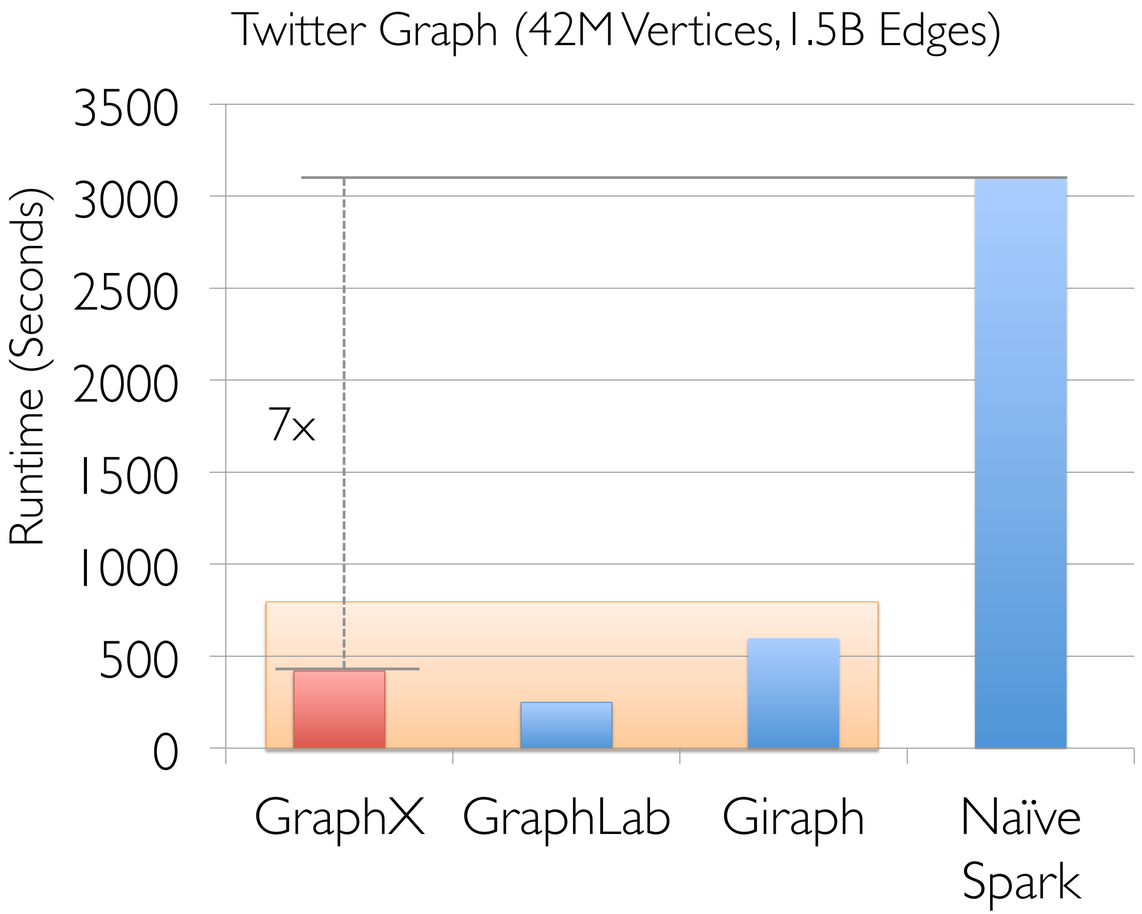


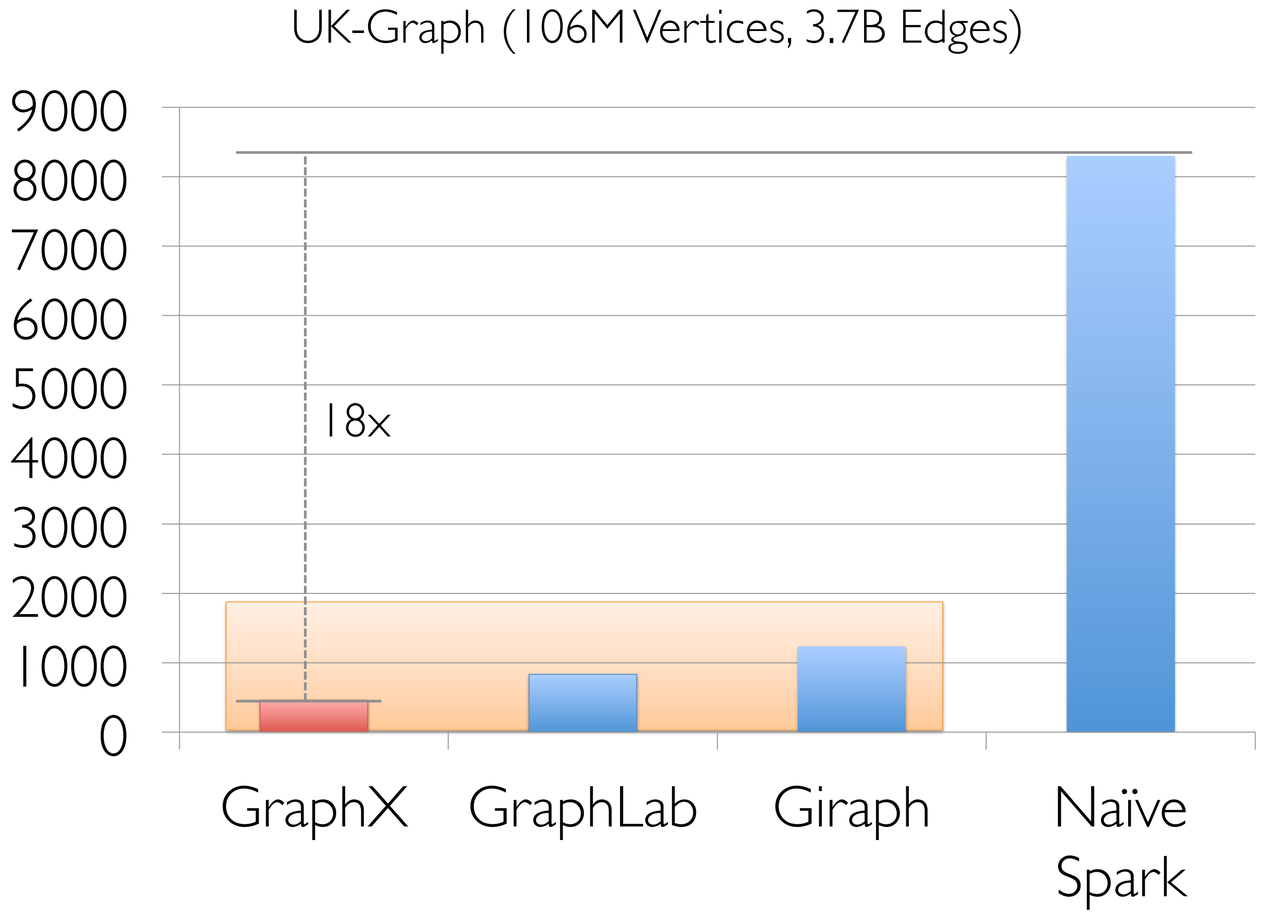
Speedup Due To Access Method Selection



PageRank Benchmark

EC2 Cluster of 16 x m2.4xLarge (8 cores) + 1GigE





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

Spark SUMMIT EAST

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

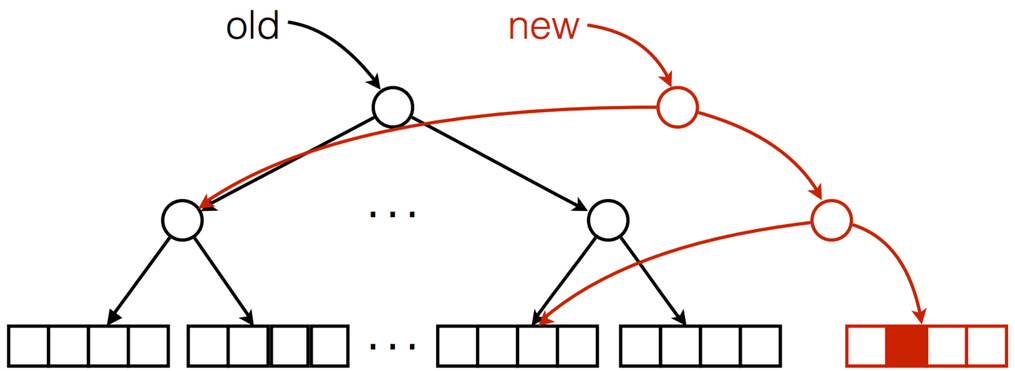
1. Language support
   a) Java API: PR #3234
   b) Python API: collaborating with Intel, SPARK-3789

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