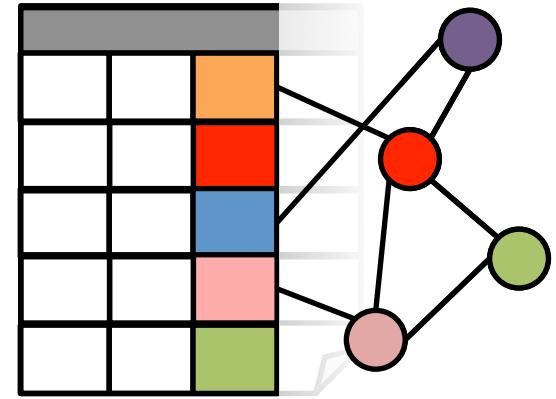


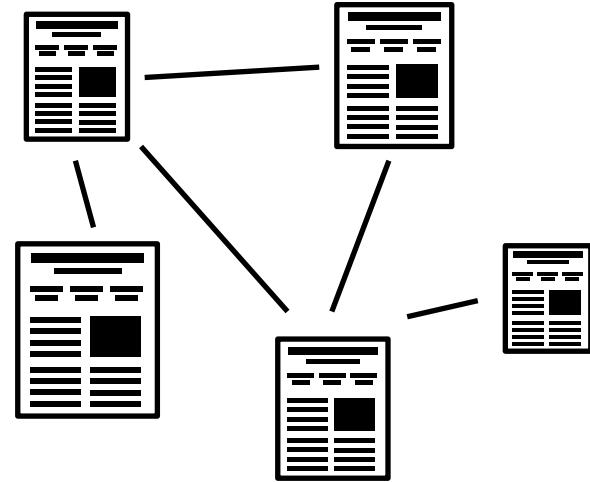
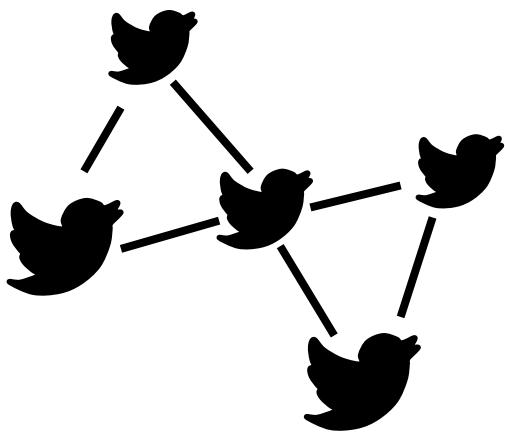
# GraphX

Graph Analytics in Spark

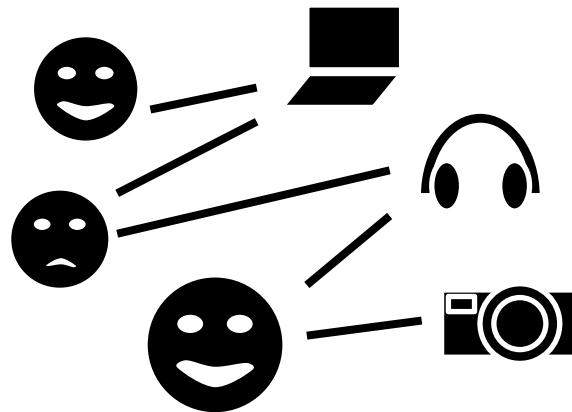
Ankur Dave  
Graduate Student, UC Berkeley AMPLab

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

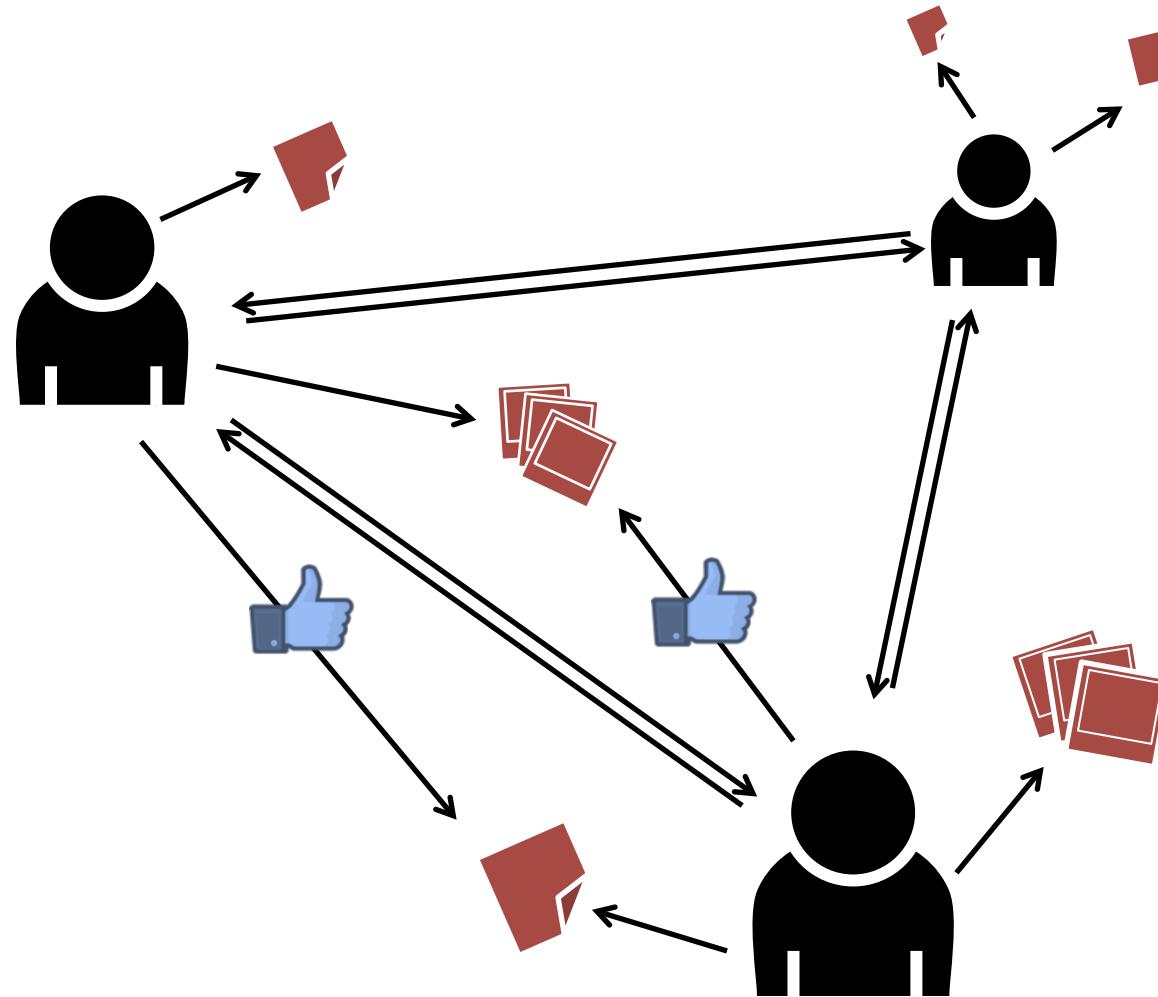




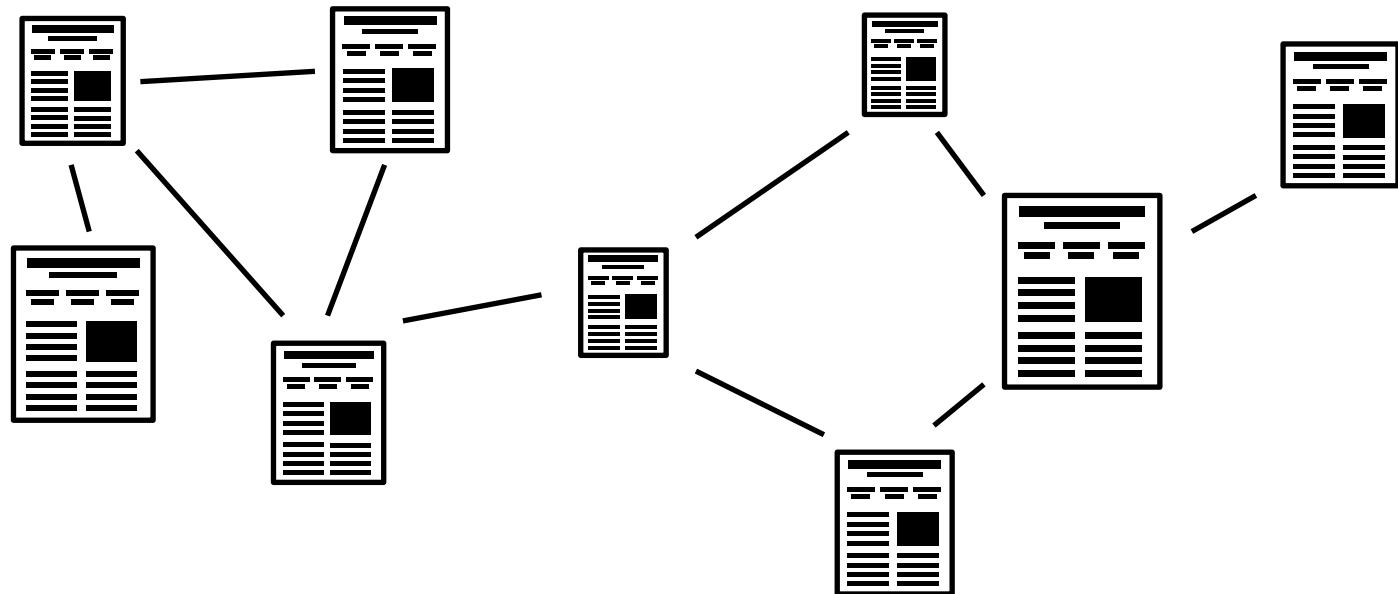
# Graphs are Everywhere



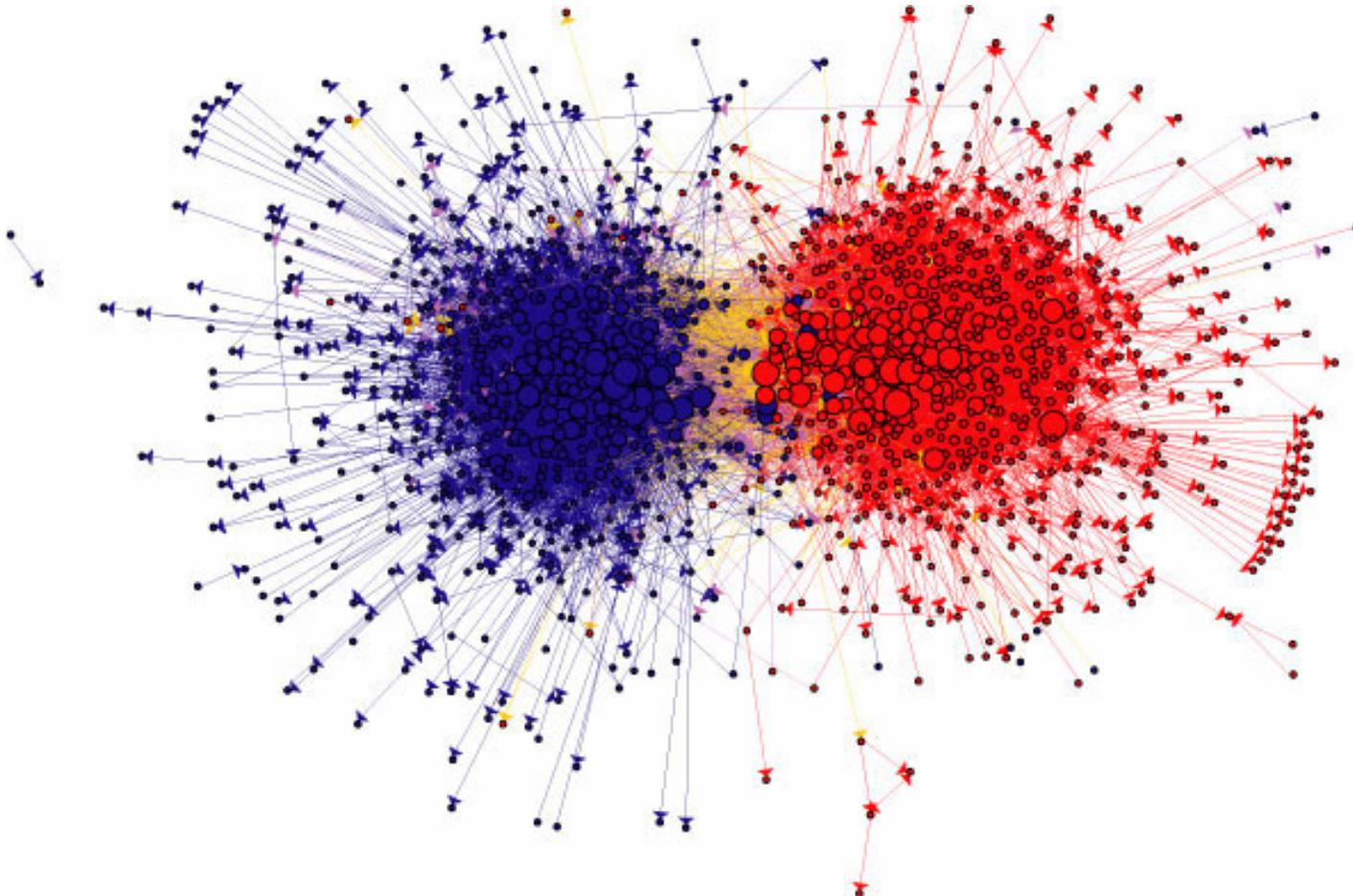
# Social Networks



# Web Graphs

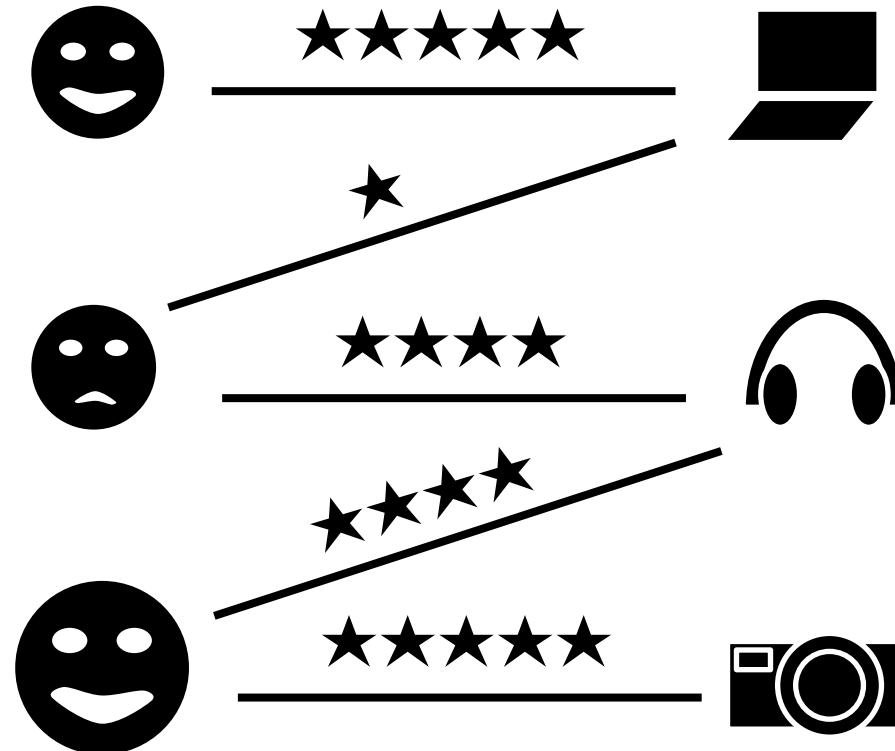


# Web Graphs



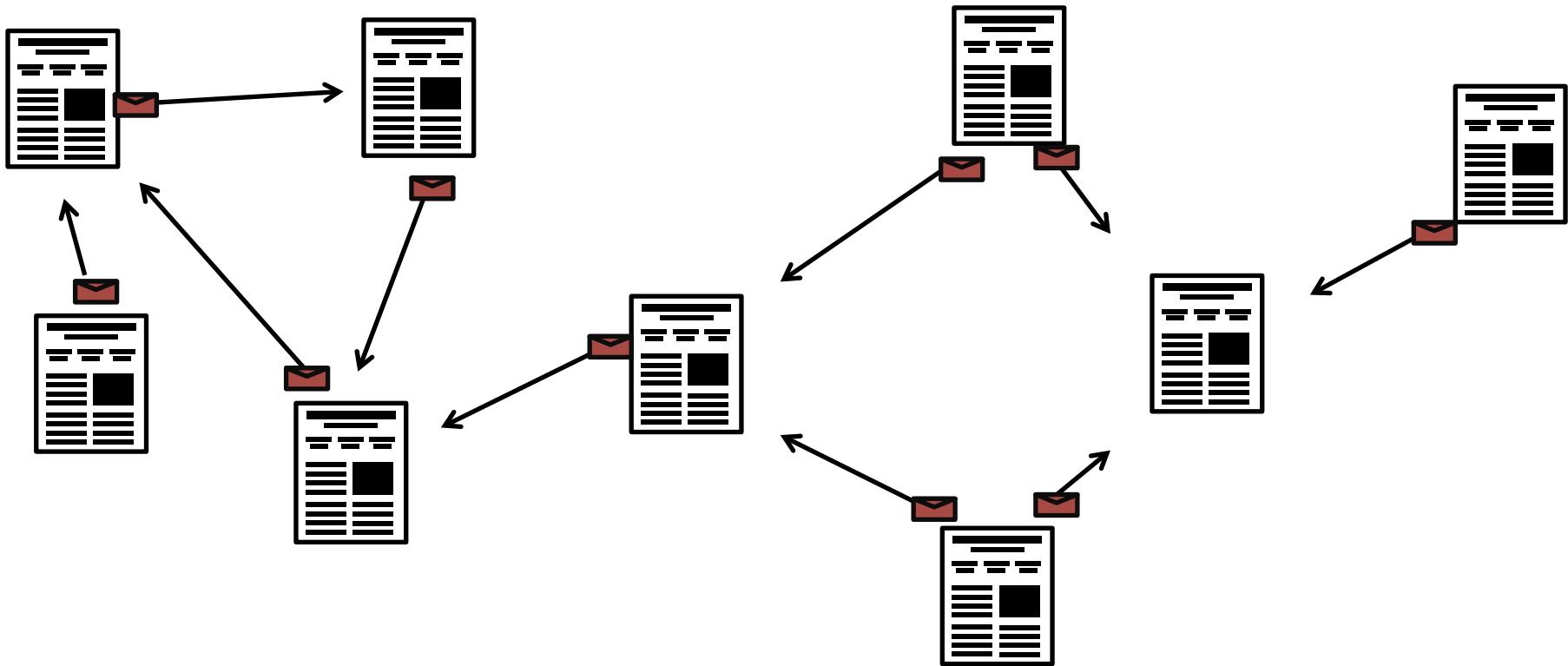
“The political blogosphere and the 2004 U.S. election: divided they blog.”  
Adamic and Glance, 2005.

# User-Item Graphs

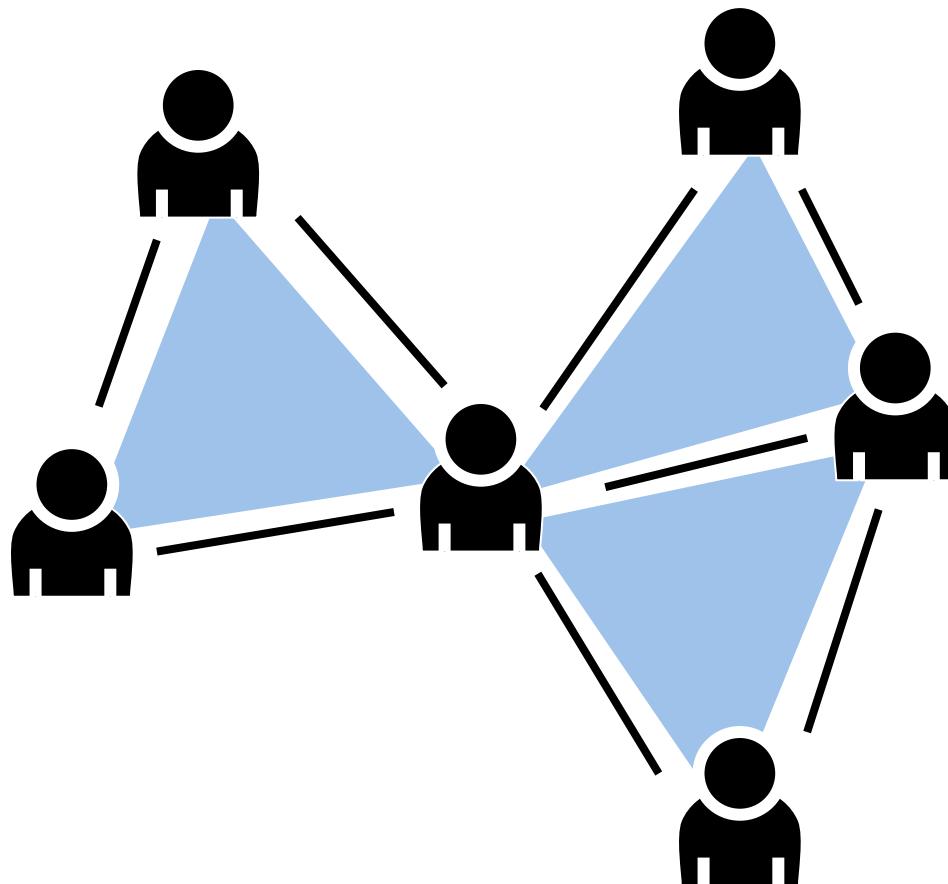


# Graph Algorithms

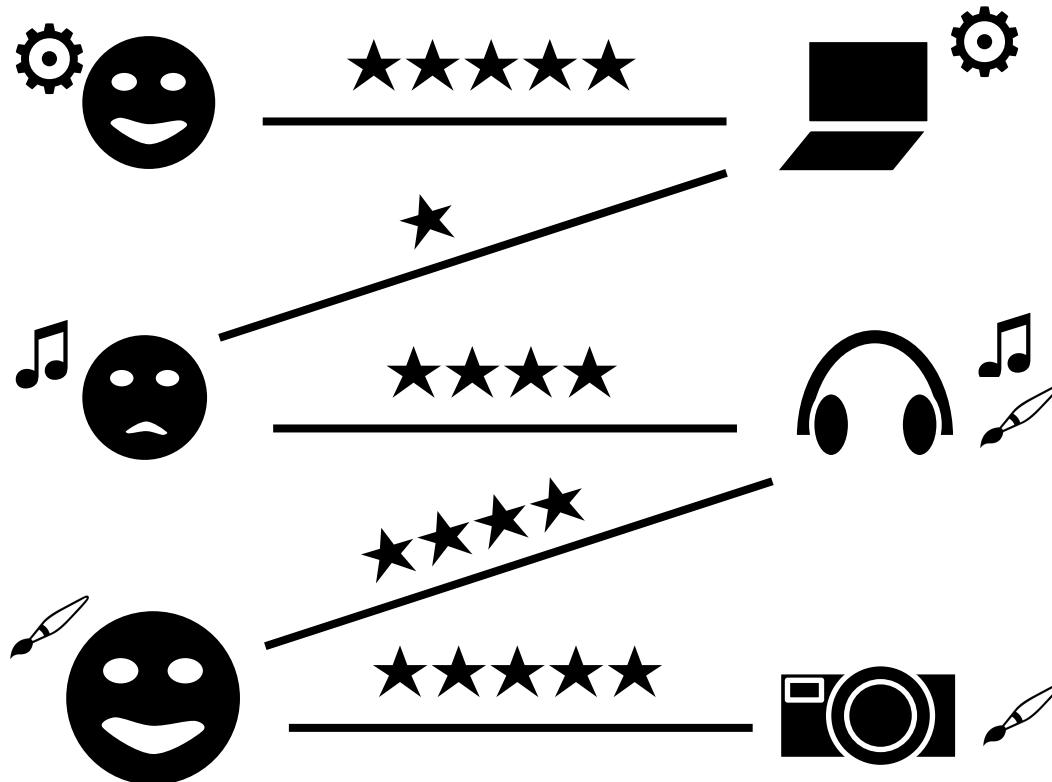
# PageRank



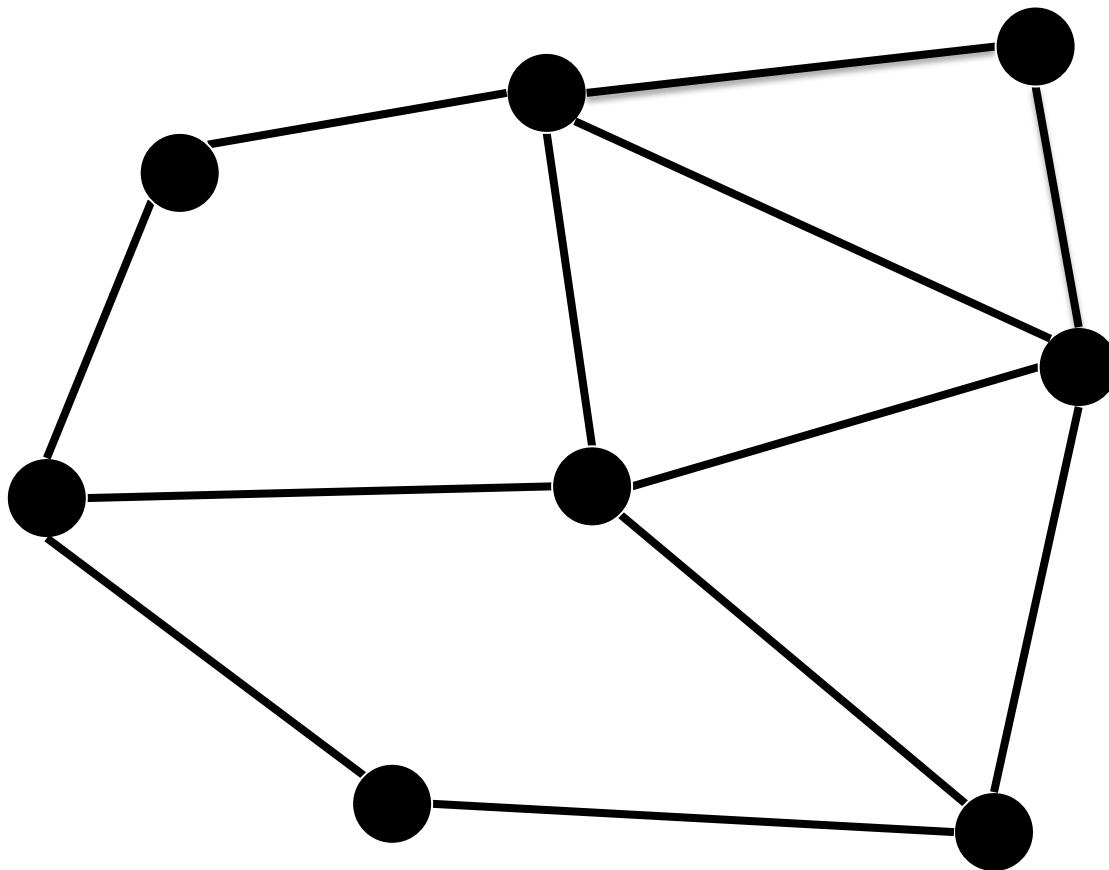
# Triangle Counting



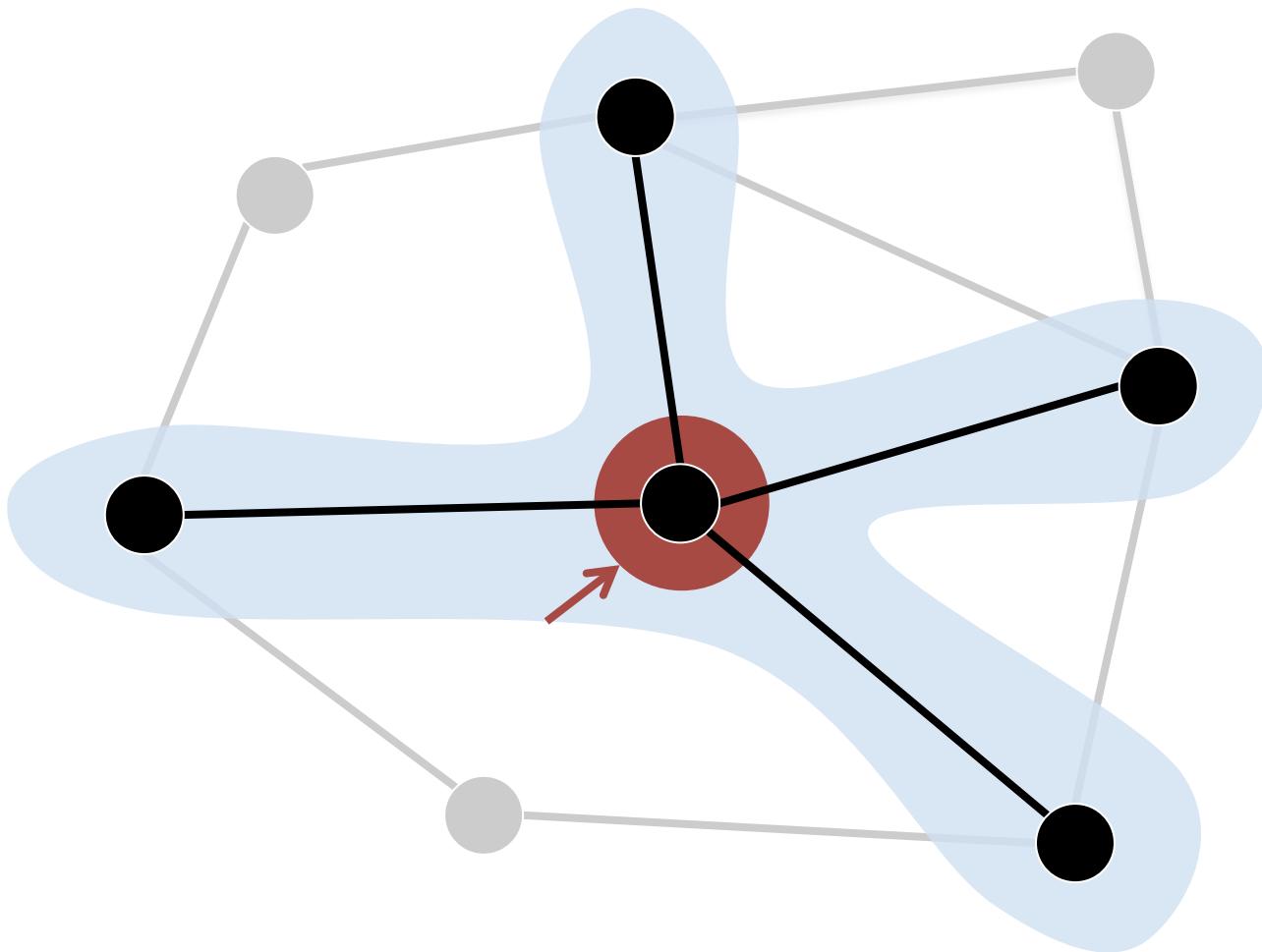
# Collaborative Filtering



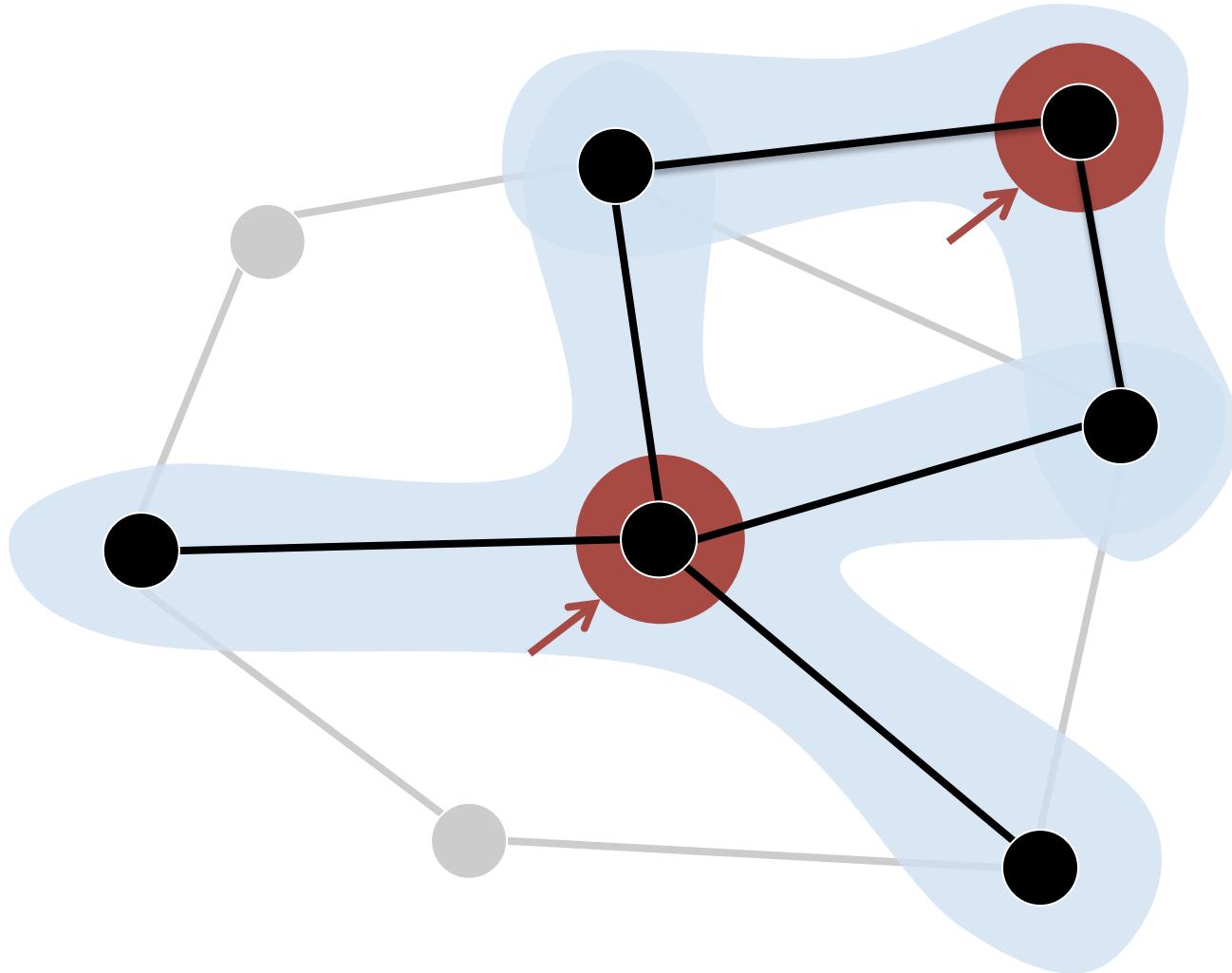
# The Graph-Parallel Pattern



# 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

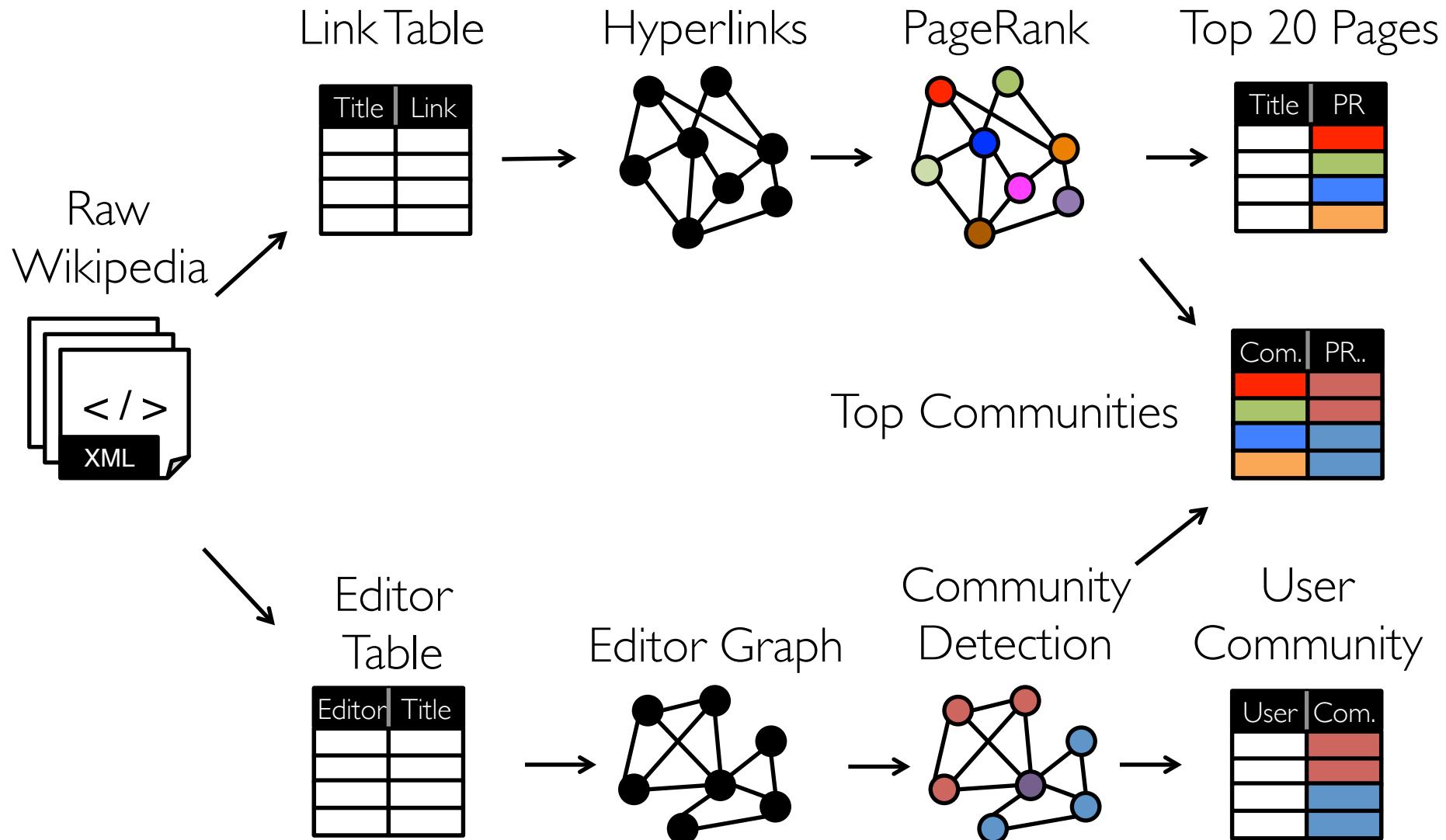
## Graph Analytics

- » PageRank
- » Personalized PageRank
- » Shortest Path
- » Graph Coloring

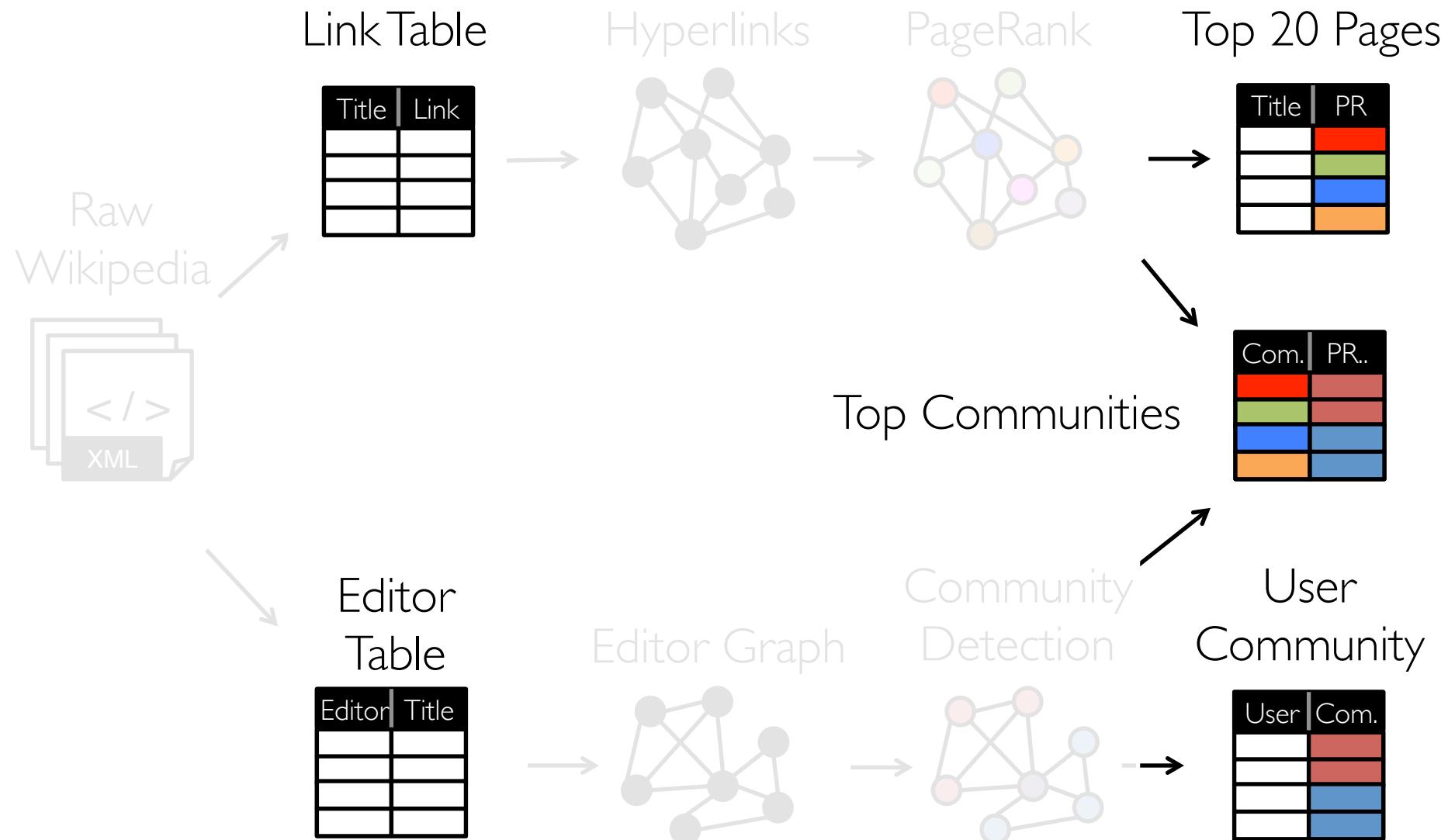
## Classification

- » Neural Networks

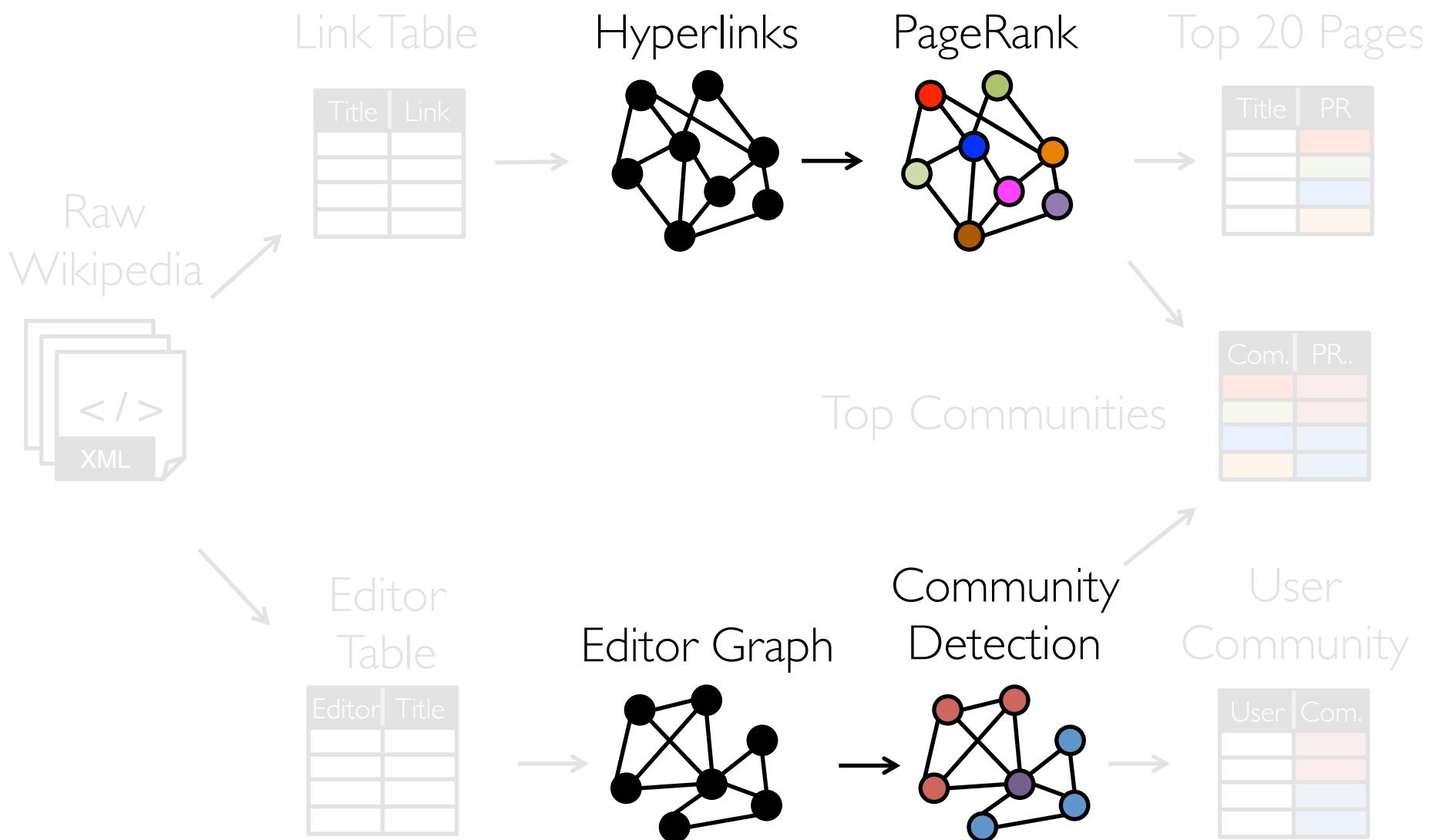
# Modern Analytics



# Tables

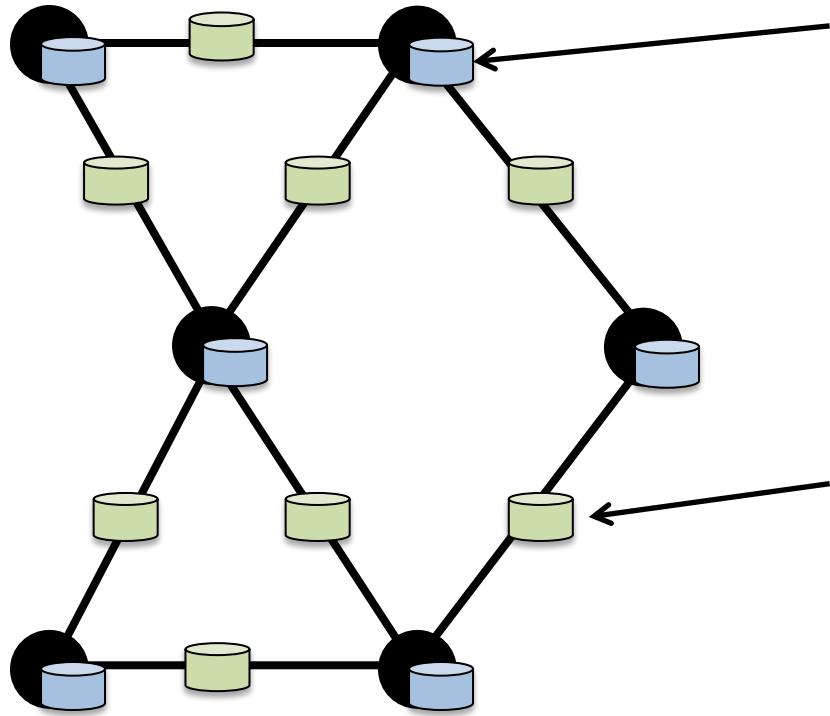


# Graphs



# The GraphX API

# Property Graphs



Vertex Property:

- User Profile
- Current PageRank Value

Edge Property:

- Weights
- Relationships
- Timestamps

# Creating a Graph (Scala)

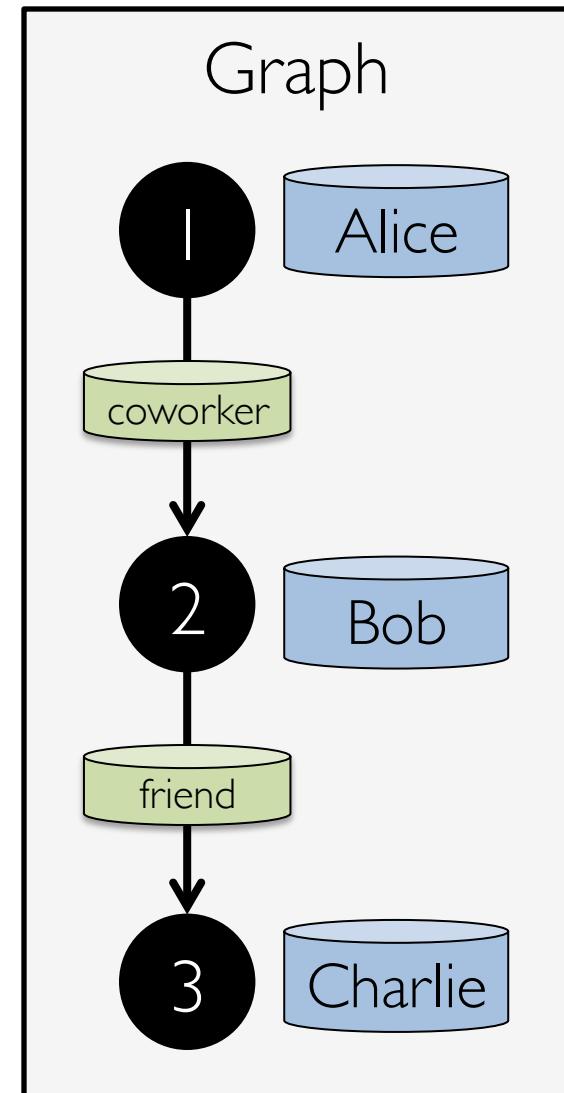
```
type VertexId = Long
```

```
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)
```



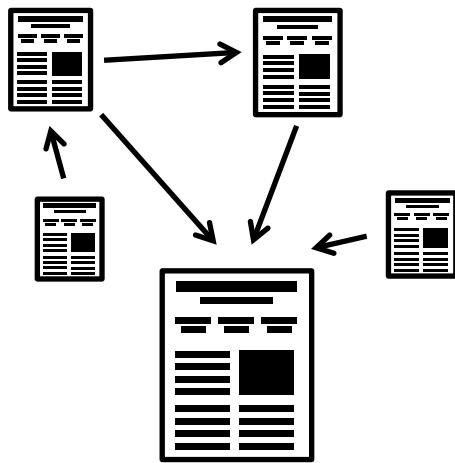
# Graph Operations (Scala)

```
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[VD, ED2]  
    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 mapReduceTriplets[A] (  
        sendMsg: EdgeTriplet[VD, ED] => Iterator[(VertexId, A)],  
        mergeMsg: (A, A) => A): RDD[(VertexId, A)]
```

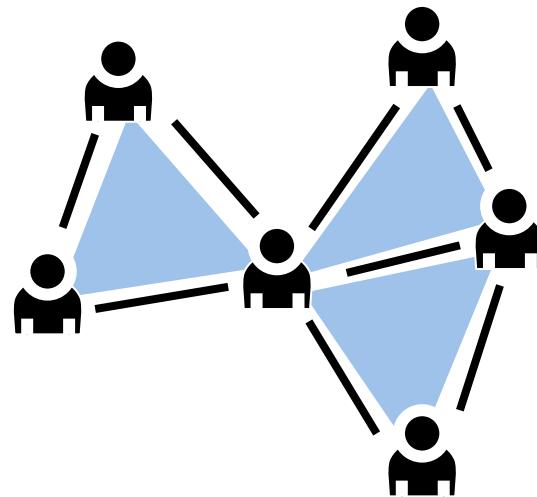
# Built-in Algorithms (Scala)

```
// 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
}
```

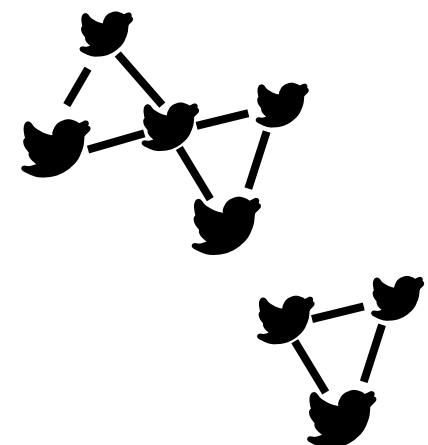
PageRank



Triangle Count

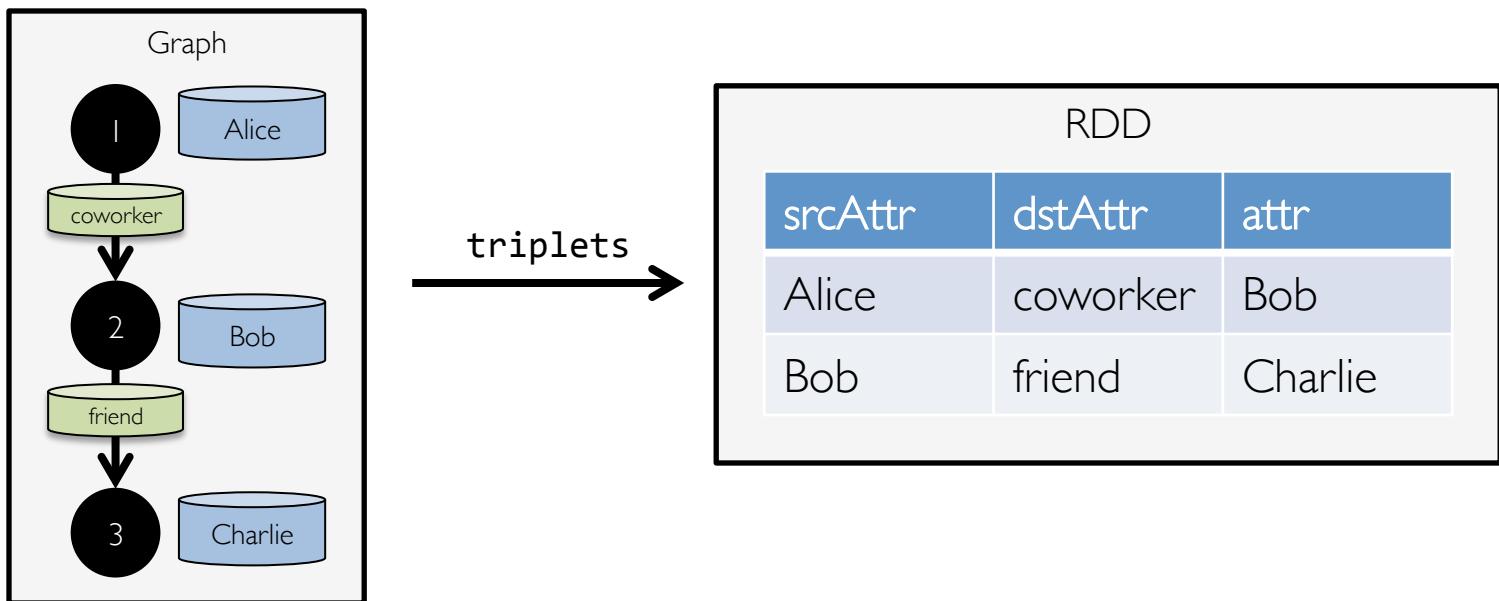


Connected Components



# The triplets view

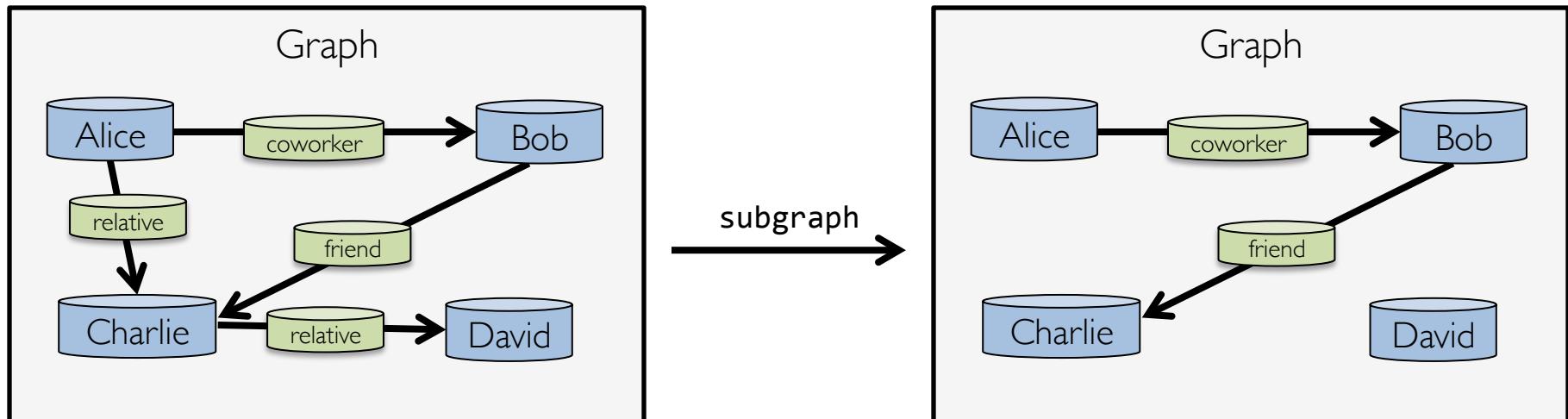
```
class Graph[VD, ED] {  
    def triplets: RDD[EdgeTriplet[VD, ED]]  
}  
  
class EdgeTriplet[VD, ED](  
    val srcId: VertexId, val dstId: VertexId, val attr: ED,  
    val srcAttr: VD, val dstAttr: VD)
```



# The subgraph transformation

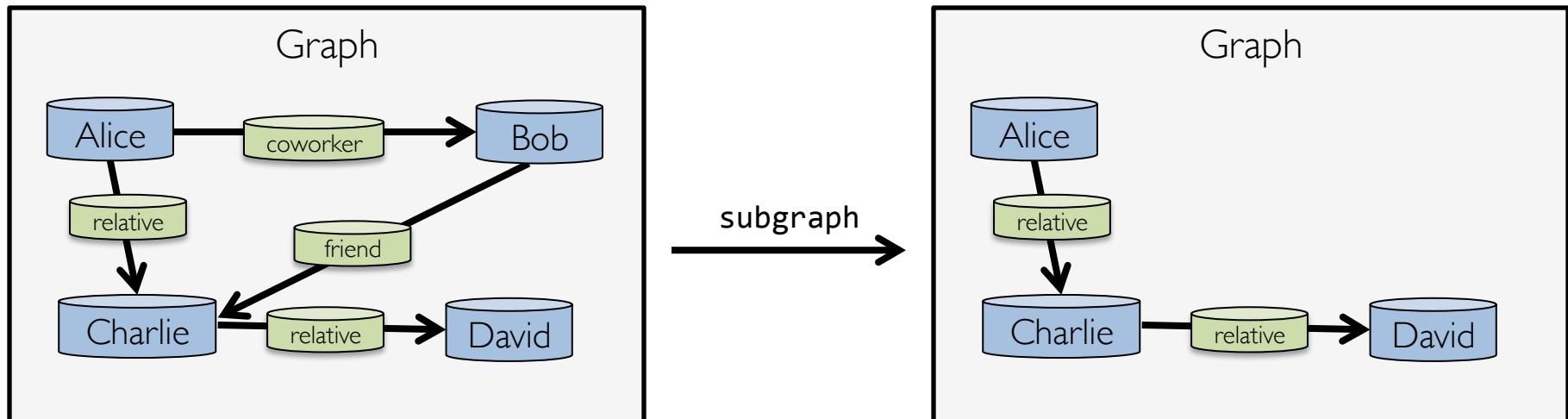
```
class Graph[VD, ED] {  
    def subgraph(epred: EdgeTriplet[VD, ED] => Boolean,  
                vpred: (VertexId, VD) => Boolean): Graph[VD, ED]  
}
```

```
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")
```

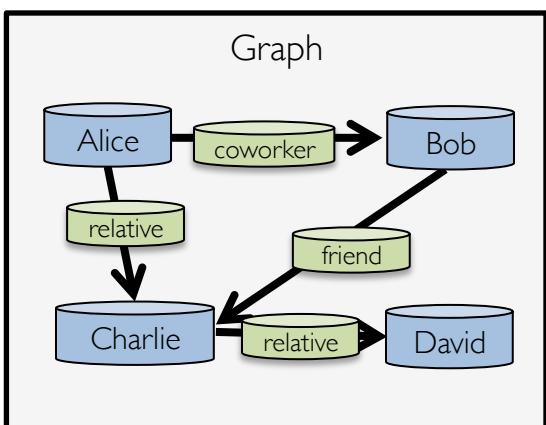


# Computation with mapReduceTriplets

```
class Graph[VD, ED] {  
    def mapReduceTriplets[A](  
        sending: Sender[VD, ED, A],  
        merging: (A, A) => A  
    )  
}
```

**⚠ upgrade to aggregateMessages in Spark 1.2.0**

```
graph.mapReduceTriplets(  
    edge => Iterator(  
        (edge.srcId, 1),  
        (edge.dstId, 1)),  
    _ + _)
```



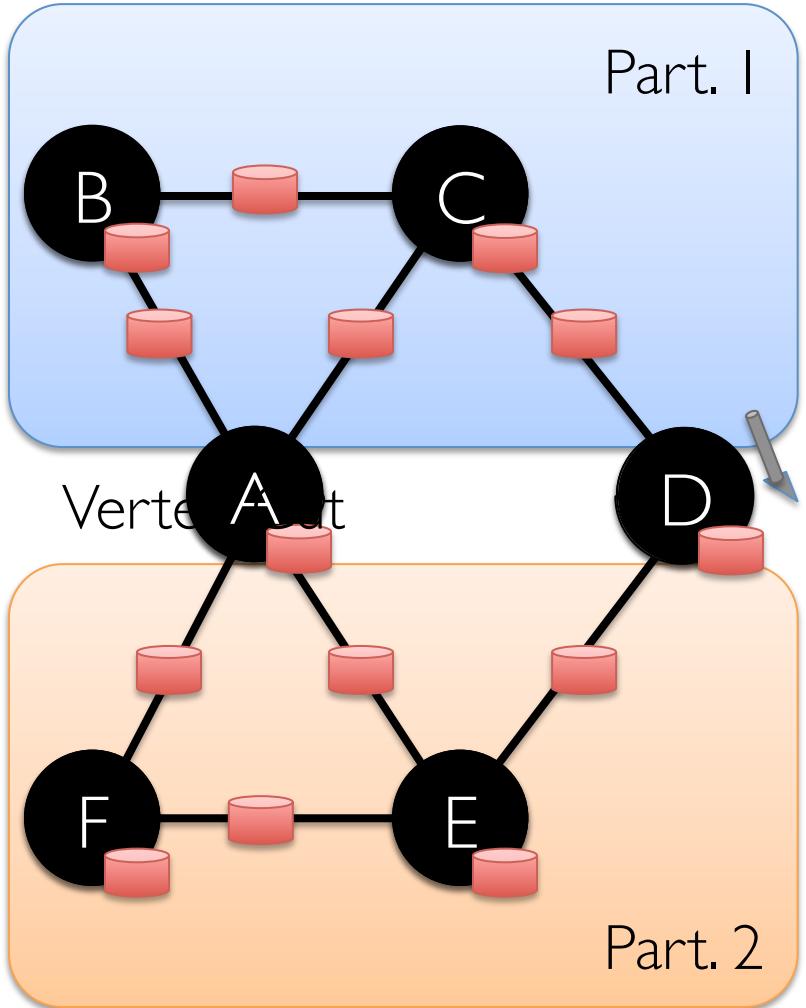
mapReduceTriplets →

RDD	
vertex id	degree
Alice	2
Bob	2
Charlie	3
David	1

# How GraphX Works

# Encoding Property Graphs as RDDs

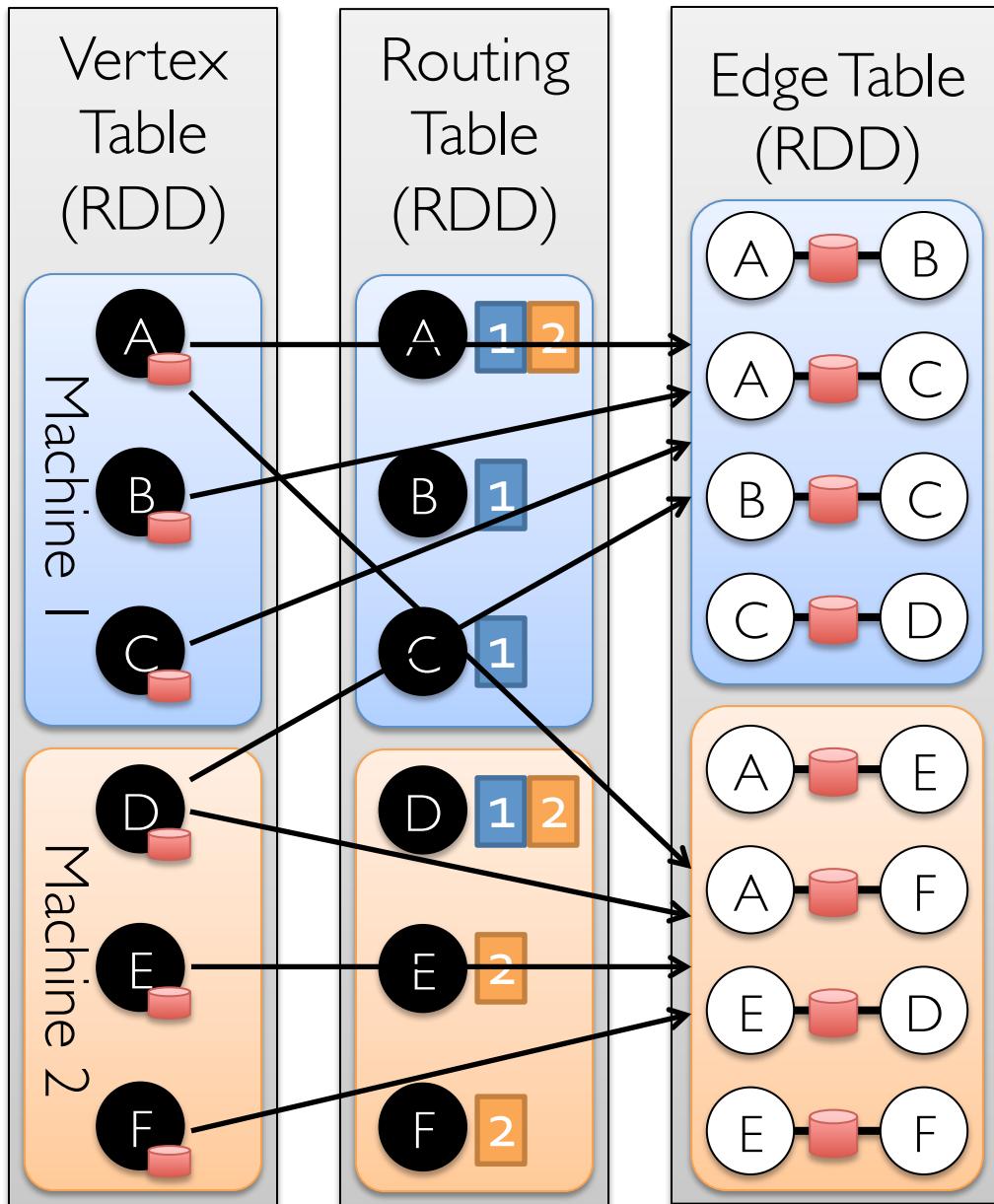
Property Graph



Part. 1

D

Part. 2



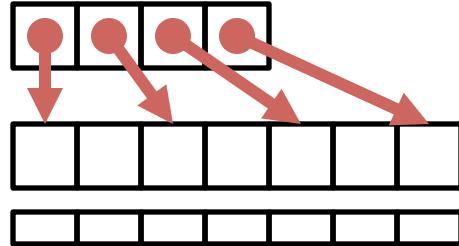
Vertex  
Table  
(RDD)

Routing  
Table  
(RDD)

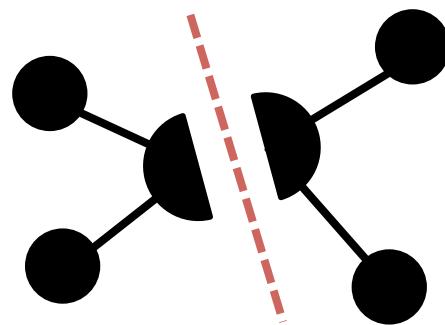
Edge Table  
(RDD)

# Graph System Optimizations

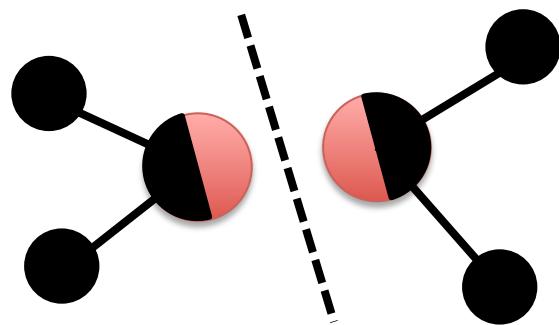
Specialized  
Data-Structures



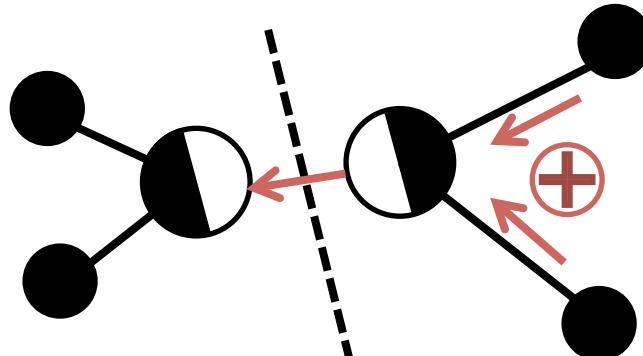
Vertex-Cuts  
Partitioning



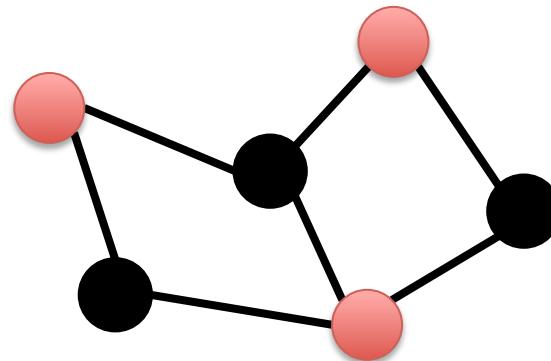
Remote  
Caching / Mirroring



Message Combiners



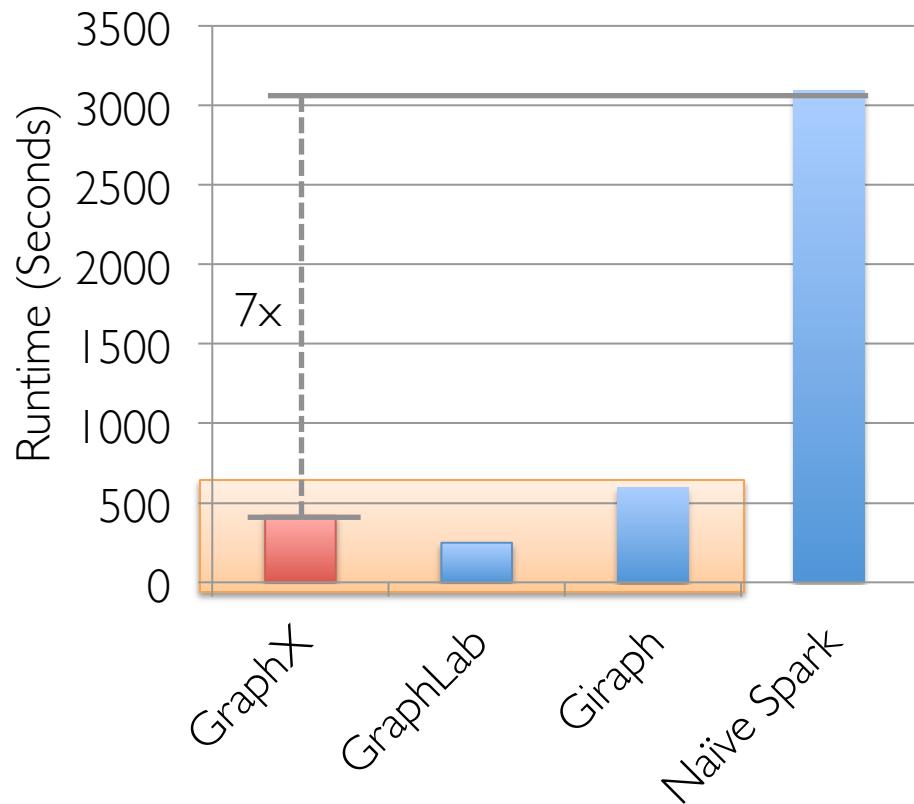
Active Set Tracking



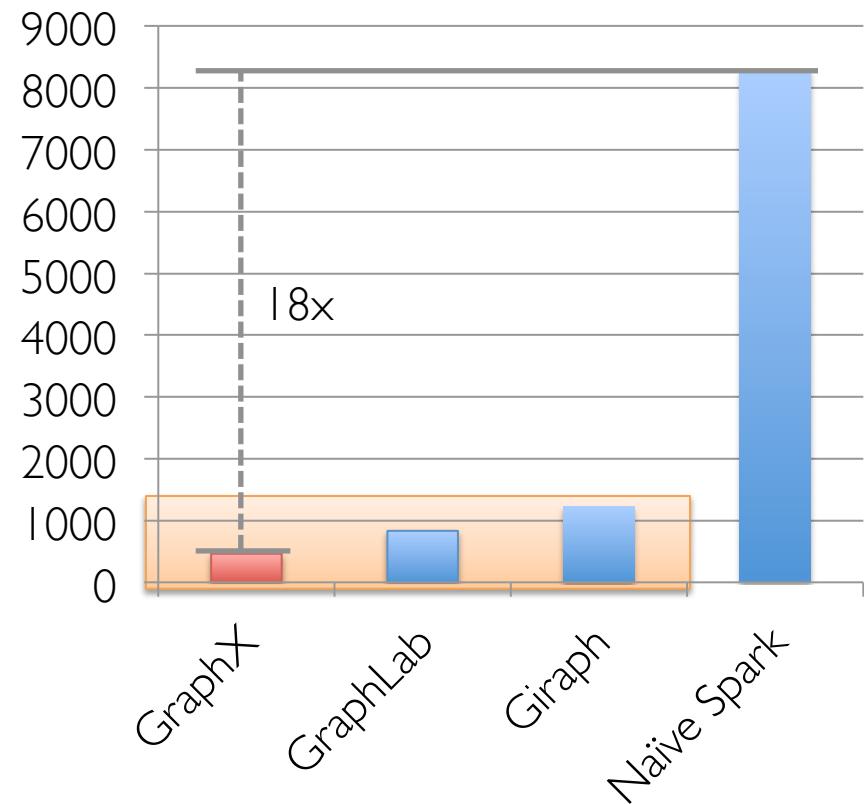
# PageRank Benchmark

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

Twitter Graph (42M Vertices, 1.5B Edges)



UK-Graph (106M Vertices, 3.7B Edges)



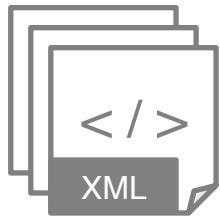
GraphX performs comparably to  
state-of-the-art graph processing systems.

# 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
  - c) Your algorithm here?
3. Speculative
  - a) Streaming/time-varying graphs
  - b) Graph database—like queries

# Try GraphX Today

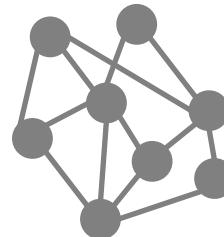
Raw  
Wikipedia



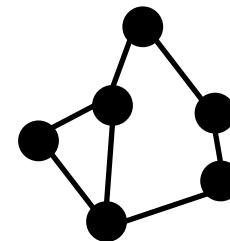
Text  
Table

Title	Body

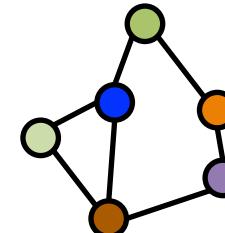
Hyperlinks



Berkeley  
subgraph



PageRank



Top 20 Pages

Title	PR

# Thanks!

<http://spark.apache.org/graphx>

[ankurd@eecs.berkeley.edu](mailto:ankurd@eecs.berkeley.edu)

[jegonzal@eecs.berkeley.edu](mailto:jegonzal@eecs.berkeley.edu)

[rxin@eecs.berkeley.edu](mailto:rxin@eecs.berkeley.edu)

[crankshaw@eecs.berkeley.edu](mailto:crankshaw@eecs.berkeley.edu)