CLOUD COMPUTING APPLICATIONS

Spark GraphX

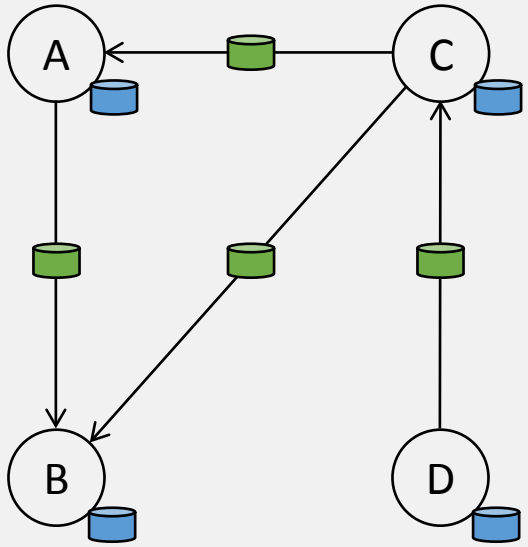
Roy Campbell & Reza Farivar

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

View a Graph as a Table

Property Graph



Vertex Property Table

Id	Property (V)
A	(P1., P2.)
B	(P3, P4.)
C	(P5., P2)
D	(P7., P4)

Edge Property Table

SrcId	DstId	Property (E)
A	B	P8
C	A	P9
D	C	P10
C	B	P11

Constructing the Graph

```
// Assume the SparkContext has already been constructed
val sc: SparkContext
// Create an RDD for the vertices
val users: RDD[(VertexId, (String, String))] =
  sc.parallelize(Array((1L, (”A", "student")), (2L, (”B", "P1")),
                       (3L, ("C", "P2")), (4L, ("D", "P3"))))
// Create an RDD for edges
val relationships: RDD[Edge[String]] =
  sc.parallelize(Array(Edge(1L, 2L, "P8"),    Edge(1L, 3L, "P2"),
                       Edge(2L, 4L, "P3"), Edge(3L, 4L, "P4")))
// Build the initial Graph
val graph = Graph(users, relationships)
```

Some Graph Operators

```
class Graph [ V, E ] {
  def Graph(vertices: Table[ (Id, V) ],
            edges: Table[ (Id, Id, E) ])
    // Table Views -----------------
    def vertices: Table[ (Id, V) ]
    def edges: Table[ (Id, Id, E) ]
    // Transformations -----------------------------
    def reverse: Graph[V, E]
    def subgraph(pV: (Id, V) => Boolean,
                 pE: Edge[V,E] => Boolean): Graph[V,E]
    def mapVertices(m: (Id, V) => T ): Graph[T,E]
    def mapEdges(m: Edge[V,E] => T ): Graph[V,T]
    // Joins ---------------------------------------
    def joinVertices(tbl: Table [(Id, Id, T)]): Graph[V, (E,
T)]
    // Computation ---------------------------------
    def connectedComponents(): Graph[V, E]
}
```