Spark GraphX

- Apache Spark's API for graphs and graph-parallel computation
- GraphX unifies ETL (Extract, Transform & Load) process
- Exploratory analysis and iterative graph computation within a single system

Use Cases

- Facebook's friends, LinkedIn's connections
- Internet's routers
- Relationships between galaxies and stars in astrophysics and Google's Maps
- Disaster detection, banking, stock market

RDD on GraphX

- GraphX extends the Spark RDD with a Resilient Distributed Property Graph
- The property graph is a directed multigraph which can have multiple edges in parallel
- The parallel edges allow multiple relationships between the same vertices

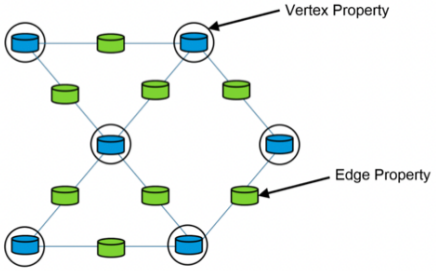


Figure: *Property Graph*

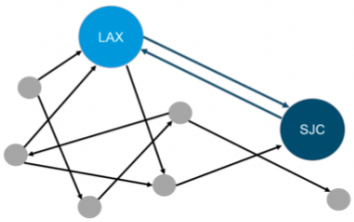


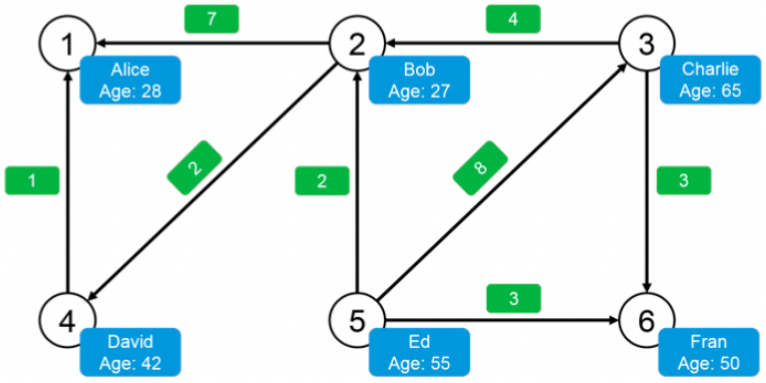
Figure: *An example of property graph*

Spark GraphX Features

- **Flexibility**
  - Spark GraphX works with both graphs and computations
  - GraphX unifies ETL (Extract, Transform & Load), exploratory analysis and iterative graph computation
- **Speed**
  - The fastest specialized graph processing systems
- **Growing Algorithm Library**
  - Page rank, connected components, label propagation, SVD++, strongly connected components and triangle count

GraphX with Examples

- The graph here represents the Twitter users and whom they follow on Twitter. For e.g. Bob follows Davide and Alice on Twitter
- Looking at the graph, we can extract information about the people (vertices) and the relations between them (edges)



Source code

```
//Importing the necessary classes
import org.apache.spark._
import org.apache.spark.rdd.RDD
import org.apache.spark.util.IntParam
import org.apache.spark.graphx._
import org.apache.spark.graphx.util.GraphGenerators
```

Displaying Vertices: Further, we will now display all the names and ages of the users (vertices).

```
val vertexRDD: RDD[(Long, (String, Int))] = sc.parallelize(vertexArray)
val edgeRDD: RDD[Edge[Int]] = sc.parallelize(edgeArray)
val graph: Graph[(String, Int), Int] = Graph(vertexRDD, edgeRDD)
graph.vertices.filter { case (id, (name, age)) => age > 30 }
  .collect.foreach { case (id, (name, age)) => println(s"$name is $age")}
```

The output for the above code is as below:

```
David is 42
Fran is 50
Ed is 55
Charlie is 65
```

Displaying Edges: Let us look at which person likes whom on Twitter.

```
for (triplet <- graph.triplets.collect)
{
println(s"${triplet.srcAttr._1} likes ${triplet.dstAttr._1}")
}
```

The output for the above code is as below:

```
Bob likes Alice
Bob likes David
Charlie likes Bob
Charlie likes Fran
David likes Alice
Ed likes Bob
Ed likes Charlie
Ed likes Fran
```

Other Example

```
## pyspark --packages graphframes:graphframes:0.6.0-spark2.2-s_2.11
from graphframes import *
from pyspark import *
from pyspark.sql import *
spark = SparkSession.builder.appName('fun').getOrCreate()
vertices = spark.createDataFrame([('1', 'Carter', 'Derrick', 50),
                                  ('2', 'May', 'Derrick', 26),
                                 ('3', 'Mills', 'Jeff', 80),
                                  ('4', 'Hood', 'Robert', 65),
                                  ('5', 'Banks', 'Mike', 93),
                                 ('98', 'Berg', 'Tim', 28),
                                 ('99', 'Page', 'Allan', 16)],
                                 ['id', 'name', 'firstname', 'age'])
edges = spark.createDataFrame([('1', '2', 'friend'),
                               ('2', '1', 'friend'),
                              ('3', '1', 'friend'),
                              ('1', '3', 'friend'),
                               ('2', '3', 'follows'),
                               ('3', '4', 'friend'),
                               ('4', '3', 'friend'),
                               ('5', '3', 'friend'),
                               ('3', '5', 'friend'),
                               ('4', '5', 'follows'),
                              ('98', '99', 'friend'),
                              ('99', '98', 'friend')],
                              ['src', 'dst', 'type'])
g = GraphFrame(vertices, edges)
## Take a look at the DataFrames
g.vertices.show()
g.edges.show()
## Check the number of edges of each vertex
g.degrees.show()
```

Spark Knowledge Graph

https://github.com/spoddutur/graph-knowledge-browser