Big Data Hadoop and Spark Developer

simplilearn

Spark GraphX

Learning Objectives

By the end of this lesson, you will be able to:

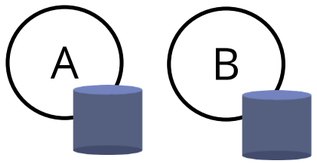
- Define graph and identify the types of graph
- Describe GraphX in Spark
- Identify different operators in GraphX
- Examine PageRank algorithm with social media data

DATA AND
ARTIFICIAL INTELLIGENCE

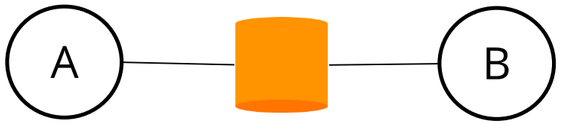
Introduction to Graph

What Is a Graph?

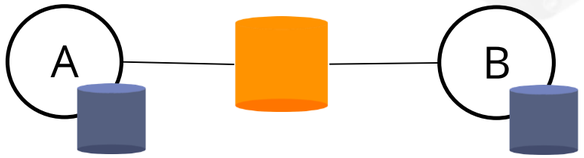
> A graph is a structure which results to a set of objects which are related to each other. The relation between them is represented using edges and vertices.



Vertices



Edges



Triplets

Use Cases of Graph Computation

Fraud Detection System

Page Rank

Disaster Detection System

Business Analysis

Geographic Information System

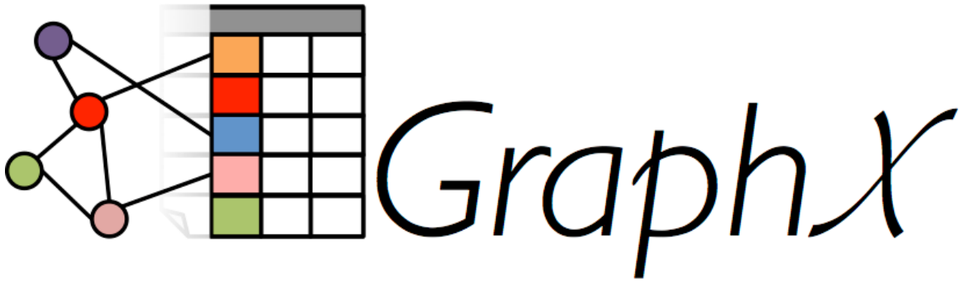
Google Pregel

Types of Graphs

DATA AND
ARTIFICIAL INTELLIGENCE

GraphX in Spark

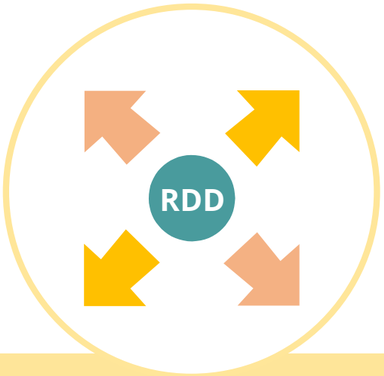
Spark GraphX



GraphX is a graph computation system that runs on data-parallel system framework. It is a new component in Spark for graphs and graph-parallel computation.

Features of Spark GraphX

GraphX is more of a real-time processing framework.



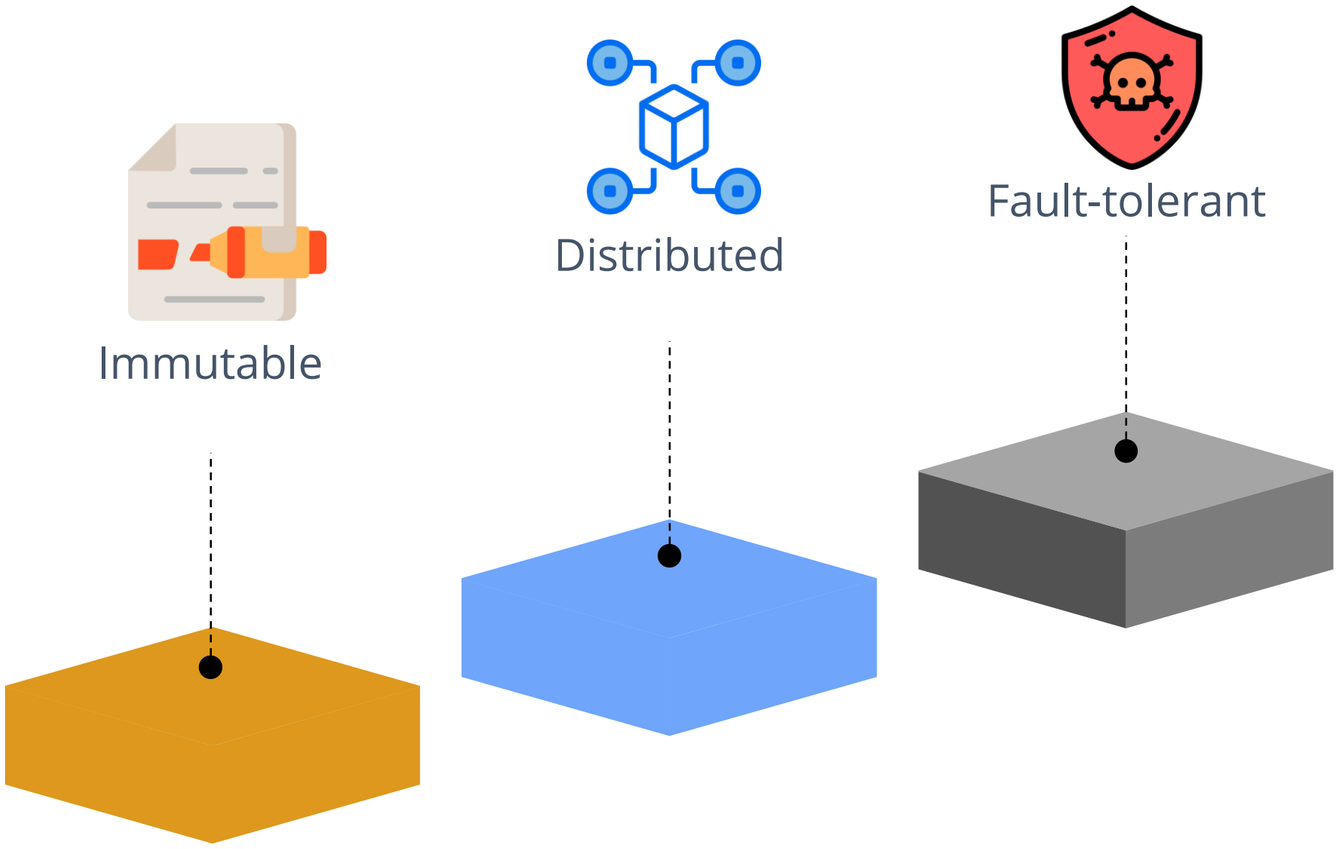
GraphX extends the RDD abstraction and introduces RDG.

GraphX simplifies the graph ETL and analysis process substantially.

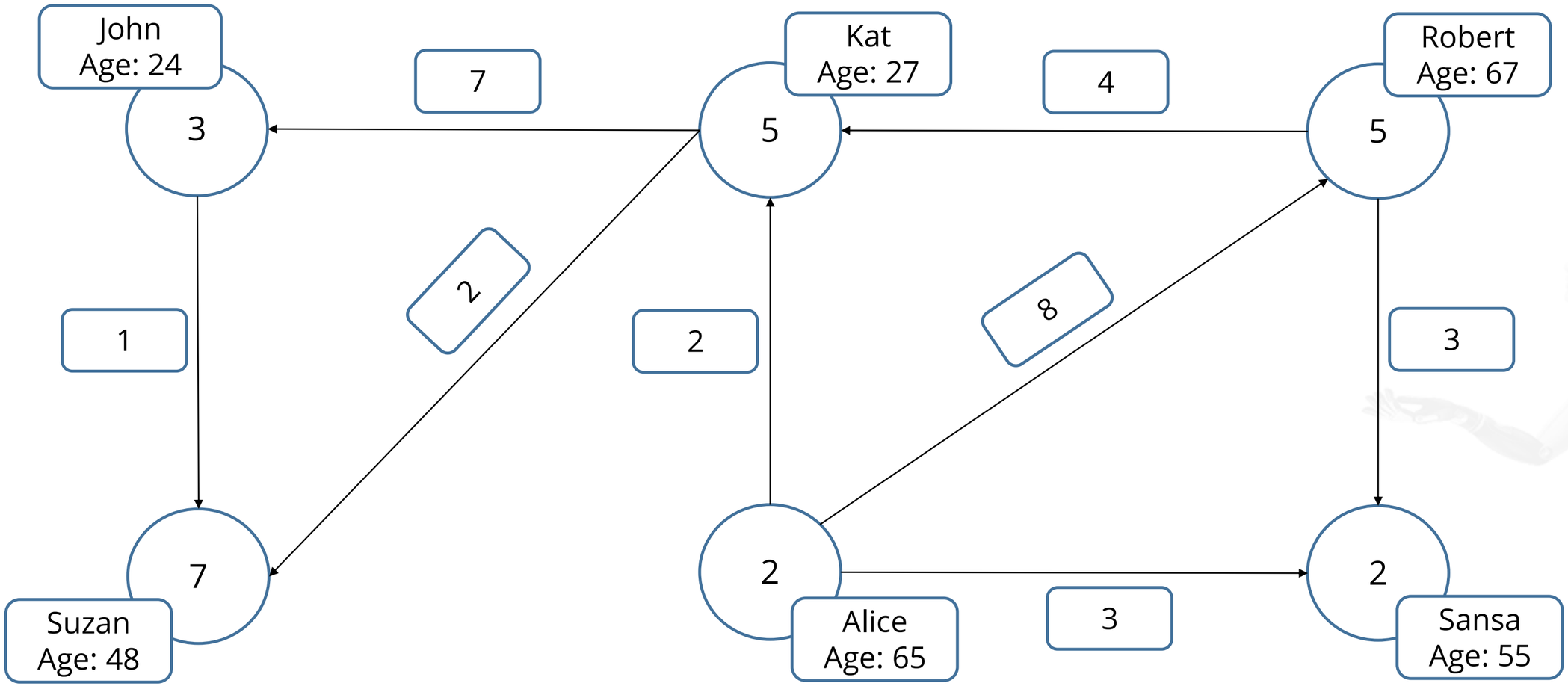
Property Graph

A directed multigraph is a directed graph with potentially multiple parallel edges sharing the same source and destination vertex.

The following are the characteristics of property graph:



GraphX: Example



Implementation of GraphX

Importing classes:

```
//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 names and edges:

```
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")}
```

DATA AND
ARTIFICIAL INTELLIGENCE

Graph Operators

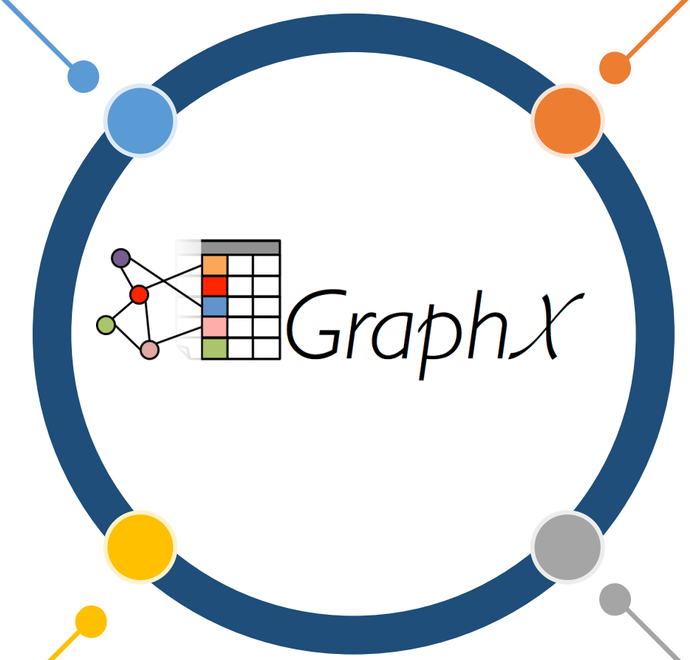
Graph Operators

Property graphs have a collection of basic operators.
These operators take user defined functions and produce new graphs.

Example:

```
val graph: Graph[(String, String), String]
// Use the GraphOps.inDegrees operator
val inDegrees: VertexRDD[Int] = graph.inDegrees
```

Types of Graph Operators



Property Operator

Structural Operator

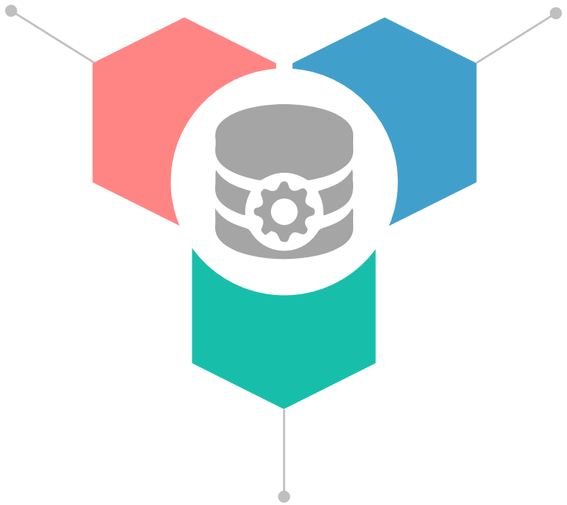
Neighborhood Aggregation

Join Operator

Property Operator

The property operator contains the following operations:

mapVertices



mapEdges

mapTriplets

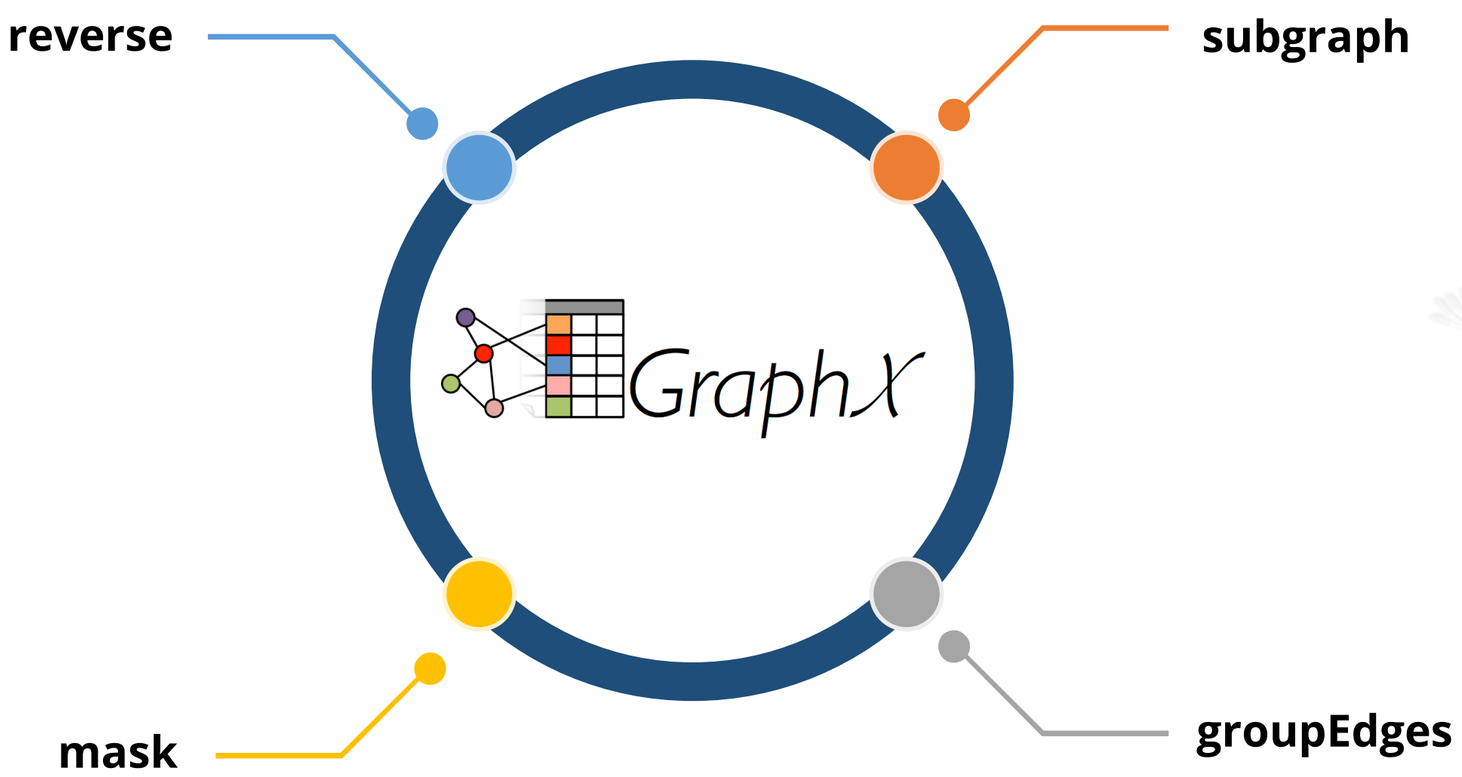
Property Operator

The following are the syntax of property operators:

```
class Graph[VD, ED] {
  def mapVertices[VD2](map: (VertexId, VD) => VD2): Graph[VD2,
ED]
  def mapEdges[ED2](map: Edge[ED] => ED2): Graph[VD, ED2]
  def mapTriplets[ED2](map: EdgeTriplet[VD, ED] => ED2):
Graph[VD, ED2]
}
```

Structural Operators

The following are a few basic structural operators:



Structural Operators

The following are the syntax of structural operators:

```
class Graph[VD, ED] {
  def reverse: Graph[VD, ED]
  def subgraph(epred: EdgeTriplet[VD,ED] => Boolean,
               vpred: (VertexId, VD) => Boolean): Graph[VD, ED]
  def mask[VD2, ED2](other: Graph[VD2, ED2]): Graph[VD, ED]
  def groupEdges(merge: (ED, ED) => ED): Graph[VD,ED]
}
```

Join Operators

The join operators are used to join data from external collections (RDDs) with graph.

joinVertices()

outerJoinVertices()

joinVertices Operator

The joinVertices is an operator that joins the vertices with the input RDD and returns a new graph with the vertex properties.

```
val nonUniqueCosts: RDD[(VertexId, Double)]
val uniqueCosts: VertexRDD[Double] =
graph.vertices.aggregateUsingIndex(nonUnique, (a,b) => a + b)
val joinedGraph = graph.joinVertices(uniqueCosts)(
  (id, oldCost, extraCost) => oldCost + extraCost)
```

outerJoinVertices Operator

In outerJoinVertices operator, the user defined map function is applied to all vertices and can change the vertex property type.

```
val outDegrees: VertexRDD[Int] = graph.outDegrees
val degreeGraph = graph.outerJoinVertices(outDegrees) { (id,
oldAttr, outDegOpt) =>
  outDegOpt match {
    case Some(outDeg) => outDeg
    case None => 0 // No outDegree means zero outDegree
  }
}
```

Neighborhood Aggregation

Neighborhood aggregation is the key task in graph analytics which includes aggregating information about the neighborhood of each vertex.

graph.mapReduceTriplets → graph.AggregateMessages

aggregateMessages is the core aggregation operation in GraphX which applies a user defined sendMsg function to each edge triplet in the graph.

Neighborhood Aggregation

The following is the syntax of aggregateMessage operator:

```
class Graph[VD, ED] {
  def aggregateMessages[Msg: ClassTag](
      sendMsg: EdgeContext[VD, ED, Msg] => Unit,
      mergeMsg: (Msg, Msg) => Msg,
      tripletFields: TripletFields = TripletFields.All)
    : VertexRDD[Msg]
}
```

Graph-Parallel System

Graph-Parallel System

Web Graphs

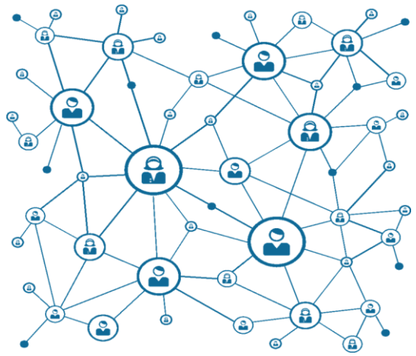
User-Item Graphs

Data Exploding

Target Advertising

Identifying Communities

Deciphering the Meaning of Documents



Web Graphs

User-Item Graphs

Limitations of Graph-Parallel System

1. Each graph-parallel system framework presents a different graph computation.
2. These frameworks depend on different runtimes.
3. These frameworks cannot resolve the data ETL and cannot decipher process issues.

DATA AND
ARTIFICIAL INTELLIGENCE

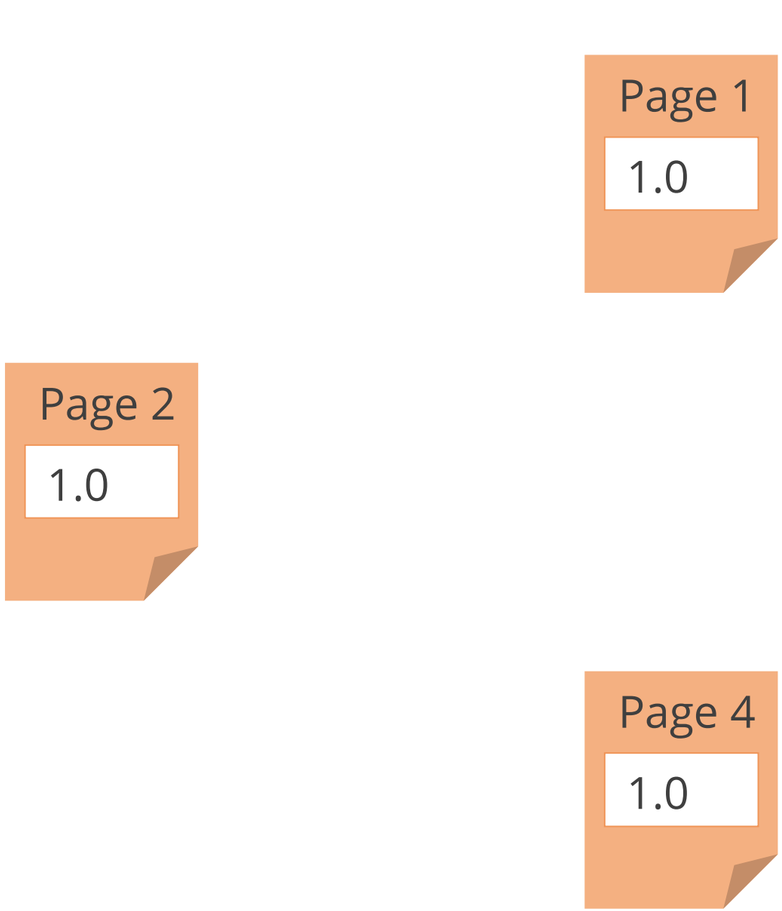
Algorithms in Spark

PageRank Algorithm

- It is an iterative algorithm
- It is used to determine the relevance or importance of a webpage
- It gives web pages a ranking score
- It outputs a probability distribution

PageRank Algorithm

On each iteration, a page contributes to its neighbors its own rank, divided by the number of its neighbors.



contribp = rankp / neighborsp

new-rank = Σcontribs * .85 + .15

PageRank with Social Media Network

GraphX includes a social network dataset on which we can run the PageRank algorithm.

```
import org.apache.spark.graphx.GraphLoader
// Load the edges as a graph
val graph = GraphLoader.edgeListFile(sc, "data/graphx/followers.txt")
// Run PageRank
val ranks = graph.pageRank(0.0001).vertices
// Join the ranks with the usernames
val users = sc.textFile("data/graphx/users.txt").map { line =>
  val fields = line.split(",")
  (fields(0).toLong, fields(1))
}
val ranksByUsername = users.join(ranks).map {
  case (id, (username, rank)) => (username, rank)
}
// Print the result
println(ranksByUsername.collect().mkString("\n"))
```

Connected Components

The connected components is an algorithm that labels each connected component of the graph.

```
import org.apache.spark.graphx.GraphLoader
// Load the graph
val graph = GraphLoader.edgeListFile(sc, "data/graphx/followers.txt")
// Find the connected components
val cc = graph.connectedComponents().vertices
// Join the connected components with the usernames
val users = sc.textFile("data/graphx/users.txt").map { line =>
  val fields = line.split(",")
  (fields(0).toLong, fields(1))
}
val ccByUsername = users.join(cc).map {
  case (id, (username, cc)) => (username, cc)
}
// Print the result
println(ccByUsername.collect().mkString("\n"))
```

Triangle Counting

The triangle counting is an algorithm that determines the number of triangles passing through each vertex, providing a measure of clustering.

```
import org.apache.spark.graphx.{GraphLoader, PartitionStrategy}
// Load the edges in canonical order and partition the graph for triangle count
val graph = GraphLoader.edgeListFile(sc, "data/graphx/followers.txt", true)
  .partitionBy(PartitionStrategy.RandomVertexCut)
// Find the triangle count for each vertex
val triCounts = graph.triangleCount().vertices
// Join the triangle counts with the usernames
val users = sc.textFile("data/graphx/users.txt").map { line =>
  val fields = line.split(",")
  (fields(0).toLong, fields(1))
}
val triCountByUsername = users.join(triCounts).map { case (id, (username, tc)) =>
  (username, tc)
}
// Print the result
println(triCountByUsername.collect().mkString("\n"))
```

Assisted Practice

Working of PageRank Algorithm

Duration: 15 mins

Problem Statement: In this demonstration, you will understand the working of PageRank algorithm.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

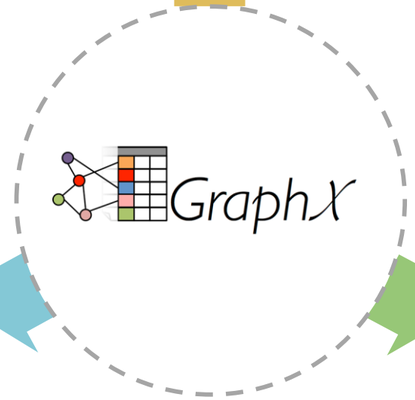
Pregel API

Pregel API

Pregel API is used for developing any vertex-centric algorithm.

It takes a message list as input and also has access to the current state of the vertex attribute and vertex id.

Vertex Program



Send Message Program

It takes the triplet view as the input with all the attributes materialized.

Merge Message Program

It takes two messages meant for the same vertex and combines them into one message.

Pregel API

Pregel API requires the following parameters:

Initial Message

The initial message to start the computation

Max Iteration

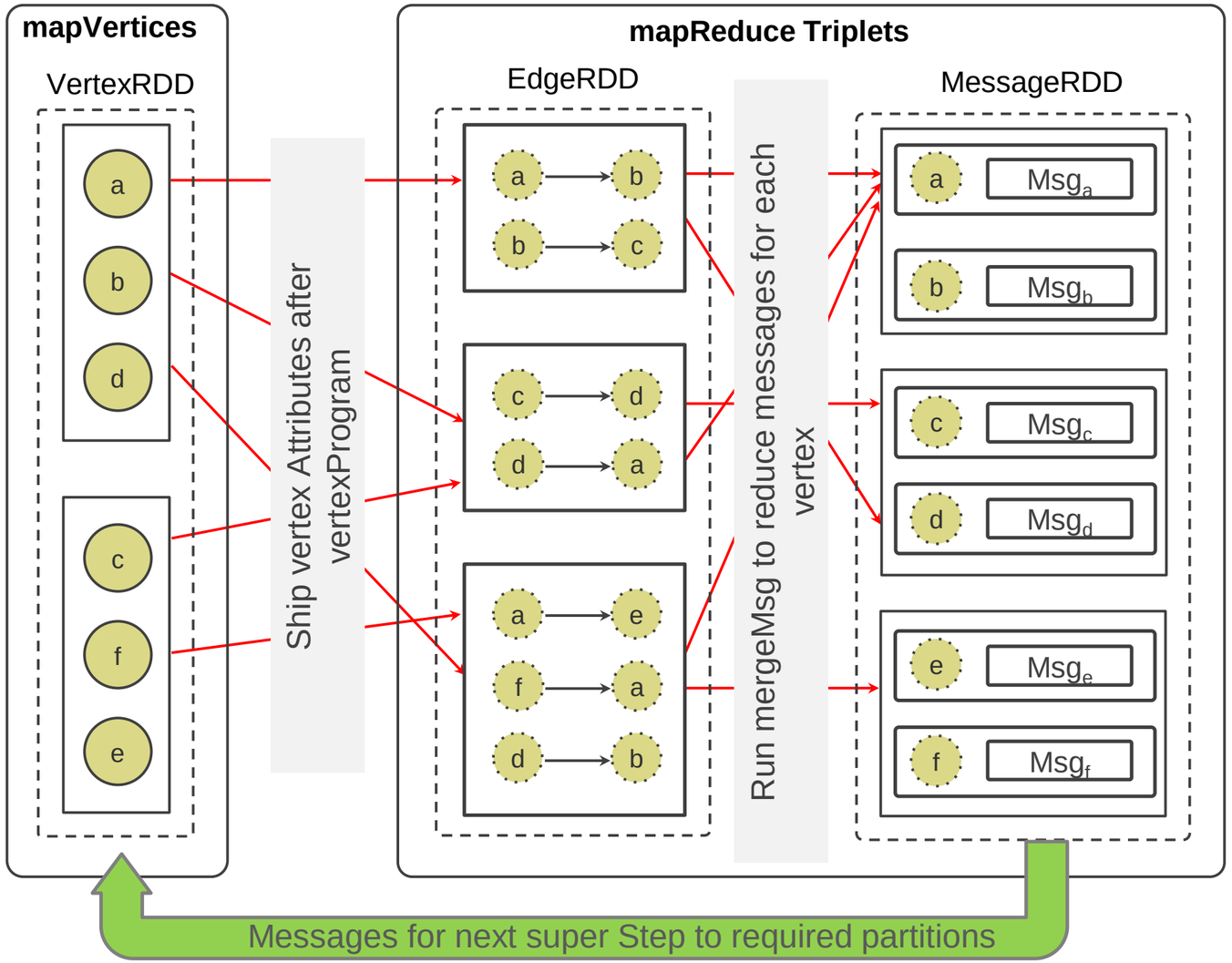
The max number of super steps for the Pregel API

Edge Direction

To filter the edges on which send message function will run

Pregel API

Pregel API in GraphX



DATA AND
ARTIFICIAL INTELLIGENCE

Use Case of GraphX

Use Case of GraphX

Flight Data Analysis Using Spark

Problem

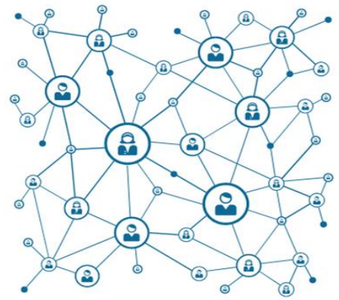
A data analyst wants to analyze the real-time data of flight using Spark GraphX to provide real-time computation results and visualize them.

Use Case of GraphX

Flight data

Database storing real-time flight data

Creating graph using GraphX



GraphX

Query 1

Compute the longest flight route

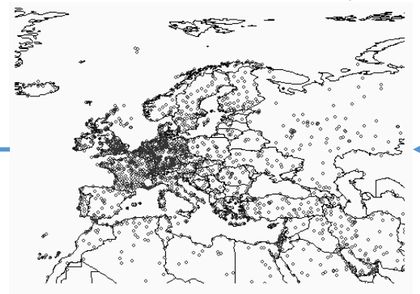
Query 2

Calculate the busiest airports

Query 3

Calculate the routes with lowest flight cost

Visualizing using Google Data Studio



USA Flight Mapping

Results

Assisted Practice

GraphX with Social Media Real-World Problem

Duration: 15 mins

Problem Statement: In this demonstration, you will work on a social medial real-world problem to understand GraphX.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Key Takeaways

You are now able to:

- Define graph and identify the types of graph
- Describe GraphX in Spark
- Identify different operators in GraphX
- Examine PageRank algorithm with social media data

DATA AND
ARTIFICIAL INTELLIGENCE

Knowledge Check

Knowledge Check

1

Which of the following is a part of a graph?

a. Edges

b. Vertices

c. Triplets

d. All of the above

Knowledge Check

1

Which of the following is a part of a graph?

a. Edges

b. Vertices

c. Triplets

d. All of the above

The correct answer is **d.**

Edges, vertices, and triplets are parts of a graph.

Knowledge Check

2

Which of the following operators joins the vertices with the input RDD and returns a new graph with the vertex properties?

a. joinVertices()

b. outerJoinVertices()

c. Both a and b

d. None of the above

Knowledge Check

2

Which of the following operators joins the vertices with the input RDD and returns a new graph with the vertex properties?

a. joinVertices()

b. outerJoinVertices()

c. Both a and b

d. None of the above

The correct answer is **a.**

joinVertices() joins the vertices with the input RDD and returns a new graph with the vertex properties.

Knowledge Check

3

Which of the following structural operator constructs a subgraph by returning a graph that contains the vertices and edges that are also found in the input graph?

a. reverse

b. subgraph

c. groupEdges

d. mask

Knowledge Check

3

Which of the following structural operator constructs a subgraph by returning a graph that contains the vertices and edges that are also found in the input graph?

a. reverse

b. subgraph

c. groupEdges

d. mask

The correct answer is **d.**

mask operator constructs a subgraph by returning a graph that contains the vertices and edges that are also found in the input graph

Lesson-End Project

Problem Statement: The US Department of transport collects statistics of all the airlines which includes Airline Details, Airport Details and flight journey details.

Collected data is in CSV format (flights_graph.csv) which contains the following fields:

a. Airline
b. Flight_Number
c. Origin_Airport
d. Destination_Airport
e. Distance
f. Arrival_Delay
g. Arrival_Time
h. Diverted
i. Cancelled

You are hired as a big data consultant to provide important insights. You must write Spark job using its graph component and use the above data to provide the following insights:

1. Routes that have distances greater than 1500 km
2. Routes where max trips are canceled
3. Routes where flights' delayed time was greater than 1300 minutes

DATA AND
ARTIFICIAL INTELLIGENCE

Thank You