Introduction to GraphX

Petro Verkhogliad

November 3rd, 2016

Data Representations

- Unstructured

```
$ grep thing filename
```

- Relational

```
$ select id from table_name
```

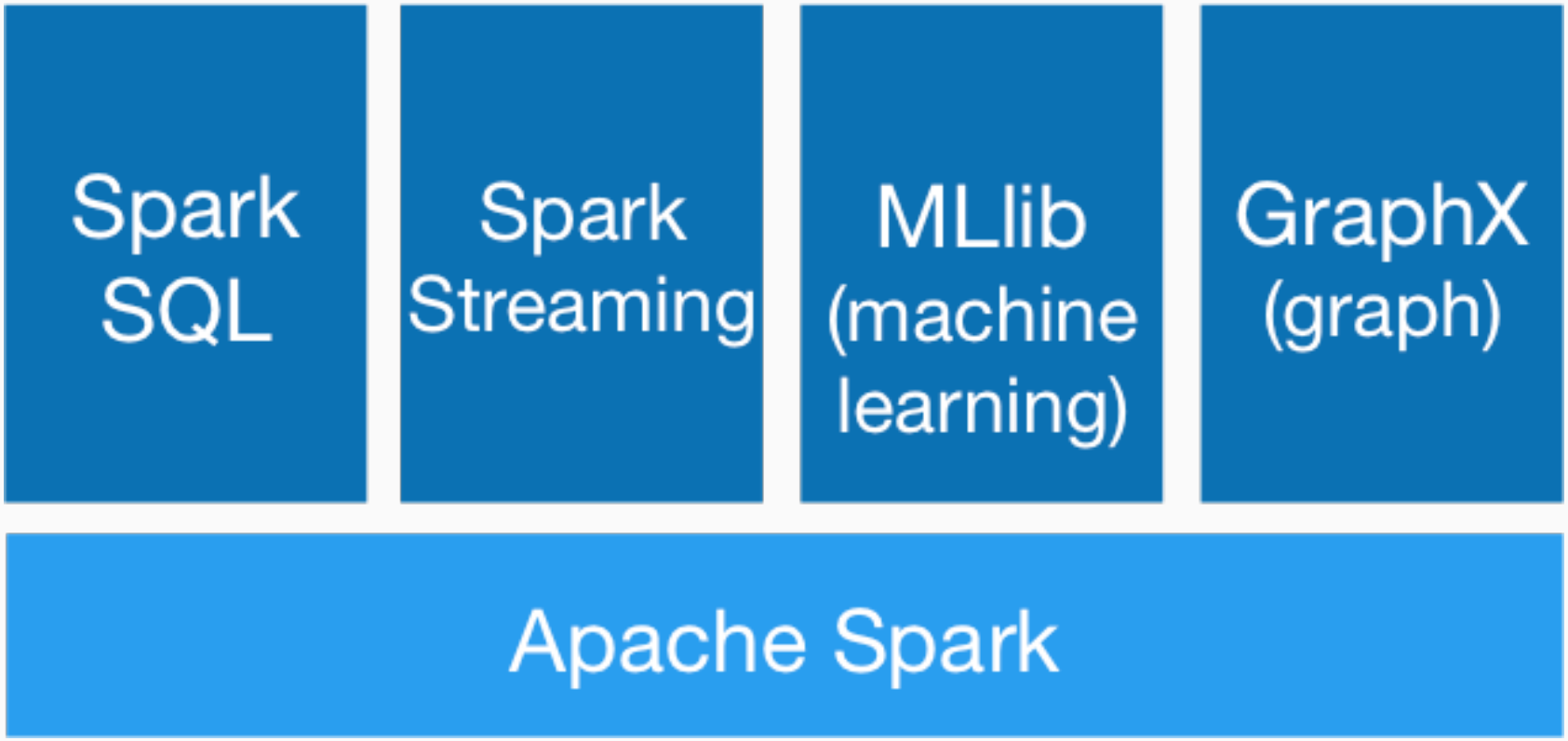
- Document

```
$ db.collection.find()
```

- Graph

```
$ graph.vertices.filter { case (id, _) => id > 2 }.count
```

Apache Spark



Apache Spark

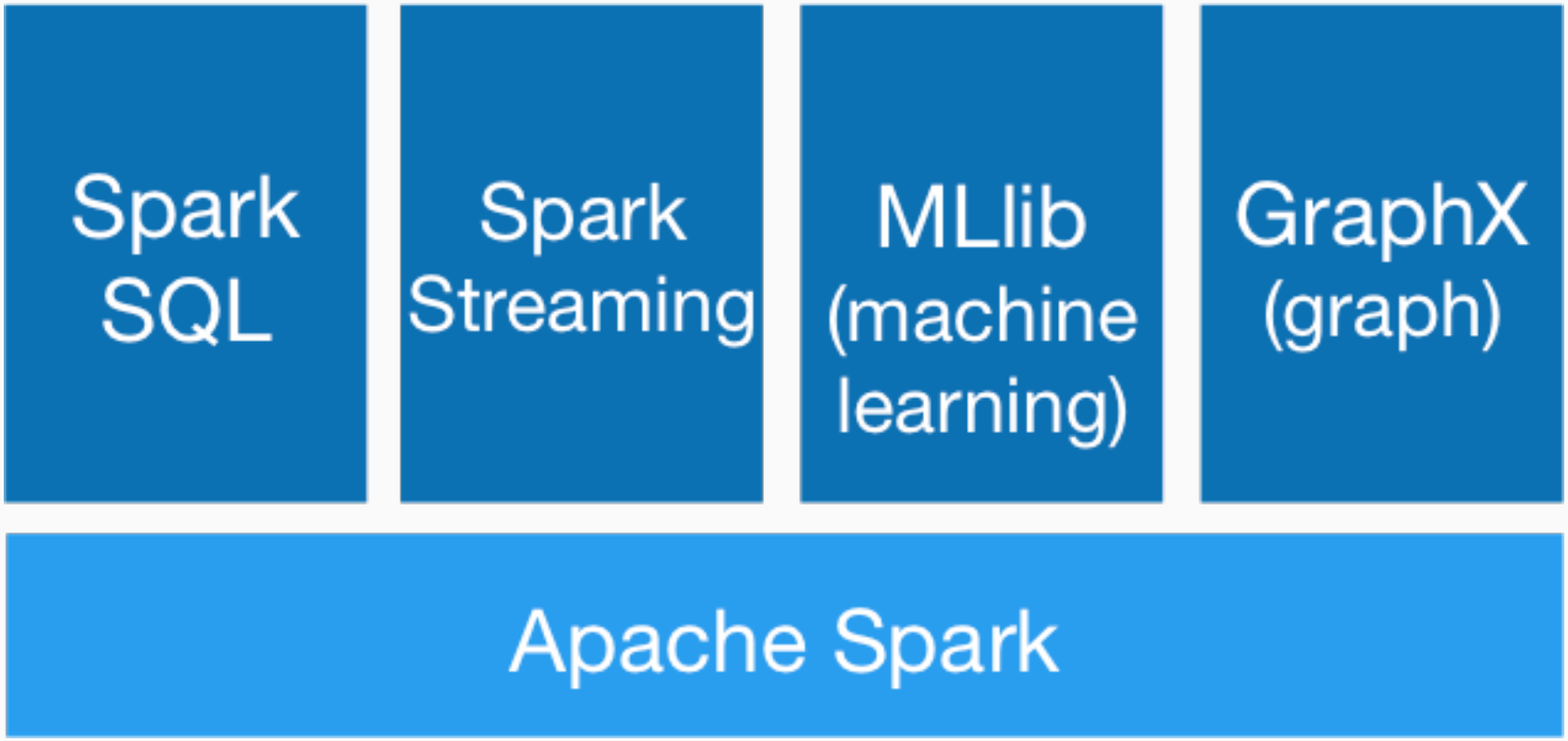
- *RDD* - Resilient Distributed Datasets
- *DataFrame* - distributed collection of data organized into named columns
- *Dataset* - strongly-typed, immutable collection of objects mapped to a relational schema

Word Count with Apache Spark

```
val textFile = sc.textFile("/var/log.txt")
val counts = textFile.flatMap(line => line.split(" "))
                 .map(word => (word, 1))
                 .reduceByKey(_ + _)
counts.saveAsTextFile("/var/log_counts.txt")
```

GraphX

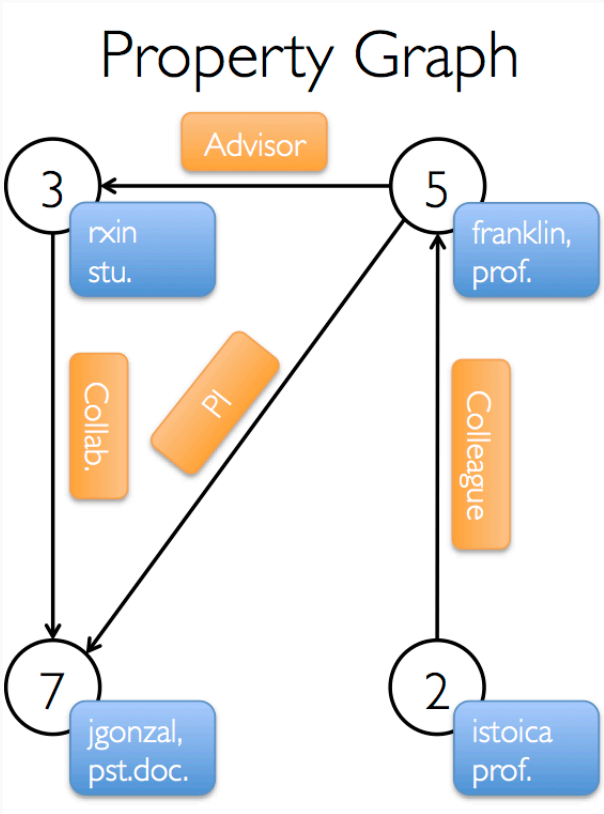
"GraphX is Apache Spark's API for graphs and graph-parallel computation."[1]



[1] https://spark.apache.org/

Property Graph

abstraction over RDDs, implementing a directed multigraph with properties on edges and vertices



Vertex Table

Id	Property (V)
3	(rxin, student)
7	(jgonzal, postdoc)
5	(franklin, professor)
2	(istoica, professor)

Edge Table

SrcId	DstId	Property (E)
3	7	Collaborator
5	3	Advisor
2	5	Colleague
5	7	PI

[1] https://spark.apache.org/

GraphX Example - basics

```
val context: SparkContext
val vs: RDD[(VertexId, String)] = context.makeRDD(
    Array((1L, "alice"), (2L, "bob"),
          (3L, "charles"), (4L, "dean"), (5L, "emma")))
val es: RDD[Edge[String]] = sc.makeRDD(
    Array(Edge(1L, 2L, "friend-of"), Edge(2L, 3L, "adversary-of"),
          Edge(2L, 4L, "friend-of"), Edge(3L, 4L, "friend-of"),
          Edge(2L, 5L, "friend-of")))
val graph: Graph[String, String] = Graph(vs, es)
graph.edges.filter(e => e.srcId == 2).collect


res1: Array[org.apache.spark.graphx.Edge[String]] =
    Array(Edge(2,3,adversary-of),
          Edge(2,4,friend-of), Edge(2,5,friend-of))
```

GraphX Example - aggregateMessages

```
graph.aggregateMessages[Int](_.sendToDst(1), _ + _)
     .join(graph.vertices)
     .map { case (id, (count, name)) => (name, count) }
     .collect


res2: Array[(String, Int)] = Array(
    (dean,2), (emma,1),
    (bob,1), (charles,1))
```

GraphX Future - GraphFrames

- http://graphframes.github.io
- DataFrames for graphs
- continually being improved for API and features
- driving improvements for Catalyst
- motifs

```
val graph: GraphFrame = ...
graph.find("(a)-[e]->(b); (b)-[e2]->(a)")
```

GraphX Future - IndexedRDD

- spark-indexedrdd package
- updatable key-value structure for Spark
- aiming to add support for efficient lookups, updates, deletions
- fundamental to implementing time-varying graphs in Spark

Questions

- What kind of data do you work with?
- What storage do you use for your data?
- What are your data challenges?
- Do you work with graphs?

Thank You!

Come hangout on YowDev Slack

https://yowdev-slackin.herokuapp.com/

Want to give a talk at Ottawa Scala Enthusiasts Group?

https://www.meetup.com/Ottawa-Scala-Enthusiasts/