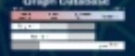
Finding Graph Isomorphisms in GraphX and GraphFrames

Michael Malak

Stack

Graph Processing vs. Graph Database



History

Isomorphisms

Definition

Detection

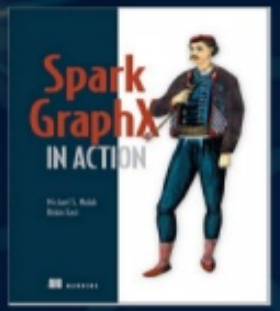
Finding

Summary

Yet Another Great Ontology (YAGO)

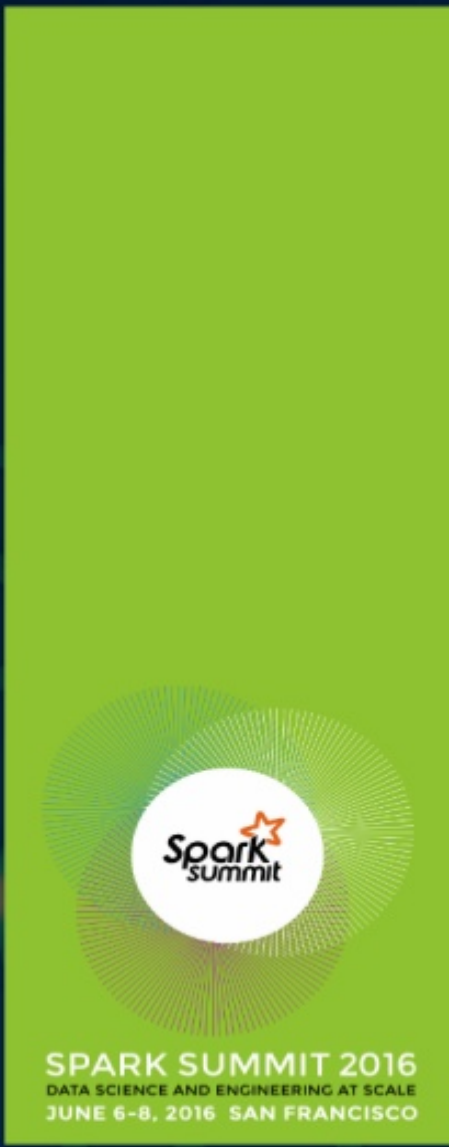
How It's Distributed

Finding Graph Isomorphisms in GraphX and GraphFrames



Michael Malak

Source code:
https://www.manning.com/books/spark-graphx-in-action
https://gist.github.com/michaelmalak



github.com/mlenadkat

ORACLE®

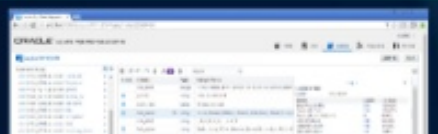
Powered by Spark

PaaS

ORACLE Cloud
Big Data

ORACLE Cloud
GoldenGate

ORACLE Cloud
Big Data Preparation

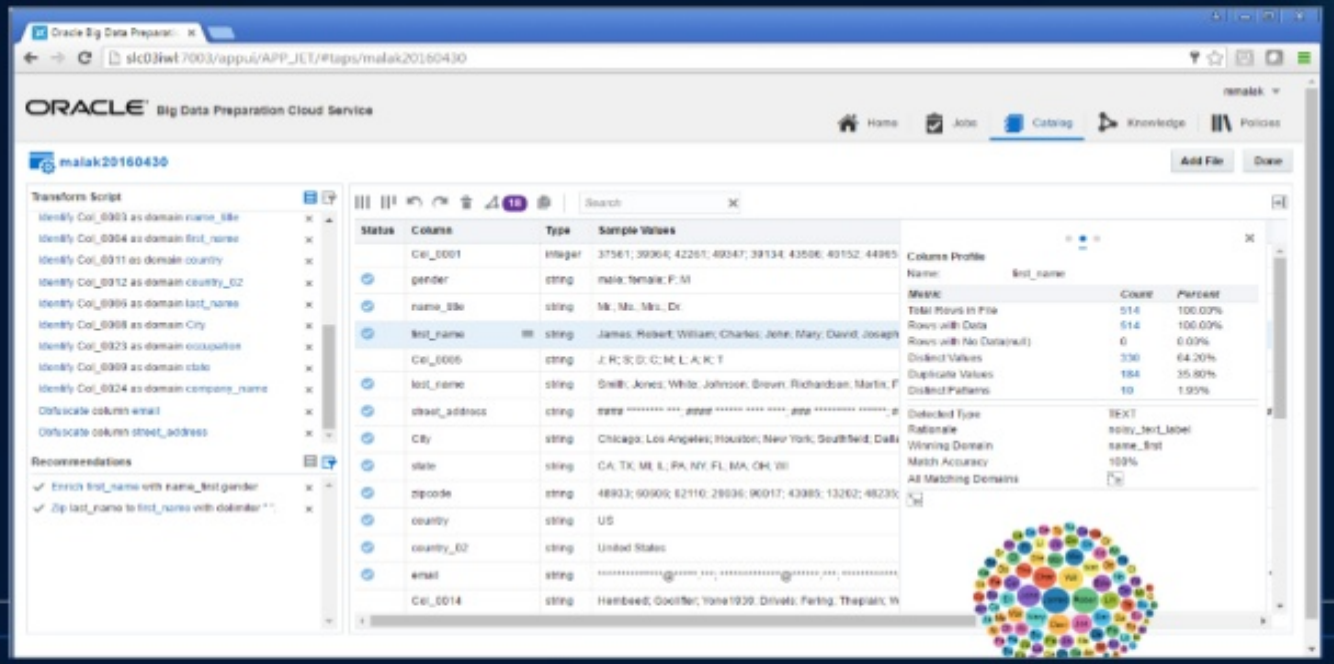


DaaS

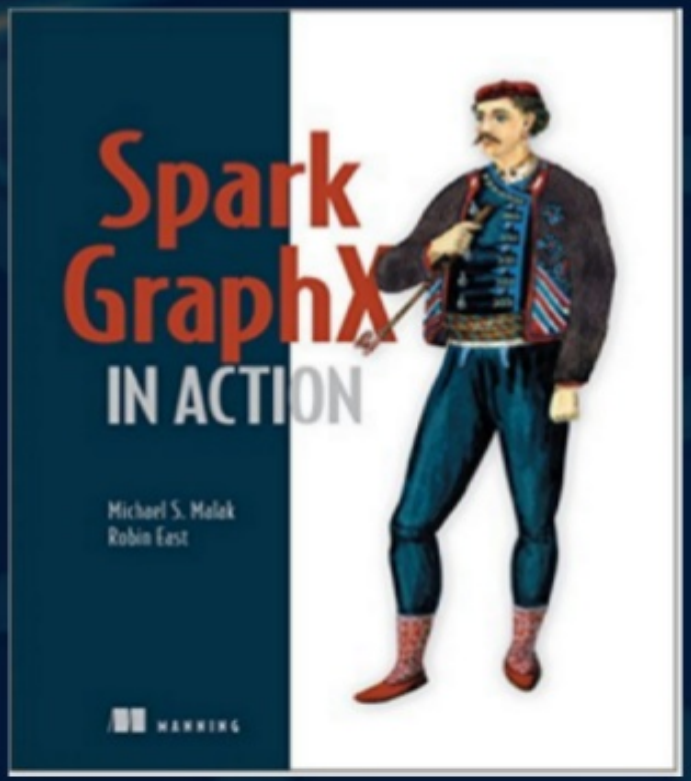
ORACLE Cloud
Data Cloud

ORACLE Cloud

Big Data Preparation



k and GraphFrames



Michael Malak

e code:

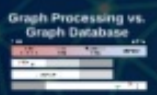
/www.manning.com/books/spark-graphx-in-acti

/gist.github.com/michaelmalak


Finding Graph Isomorphisms in GraphX and GraphFrames
Michael Malak
ORACLE


Spark


Stack

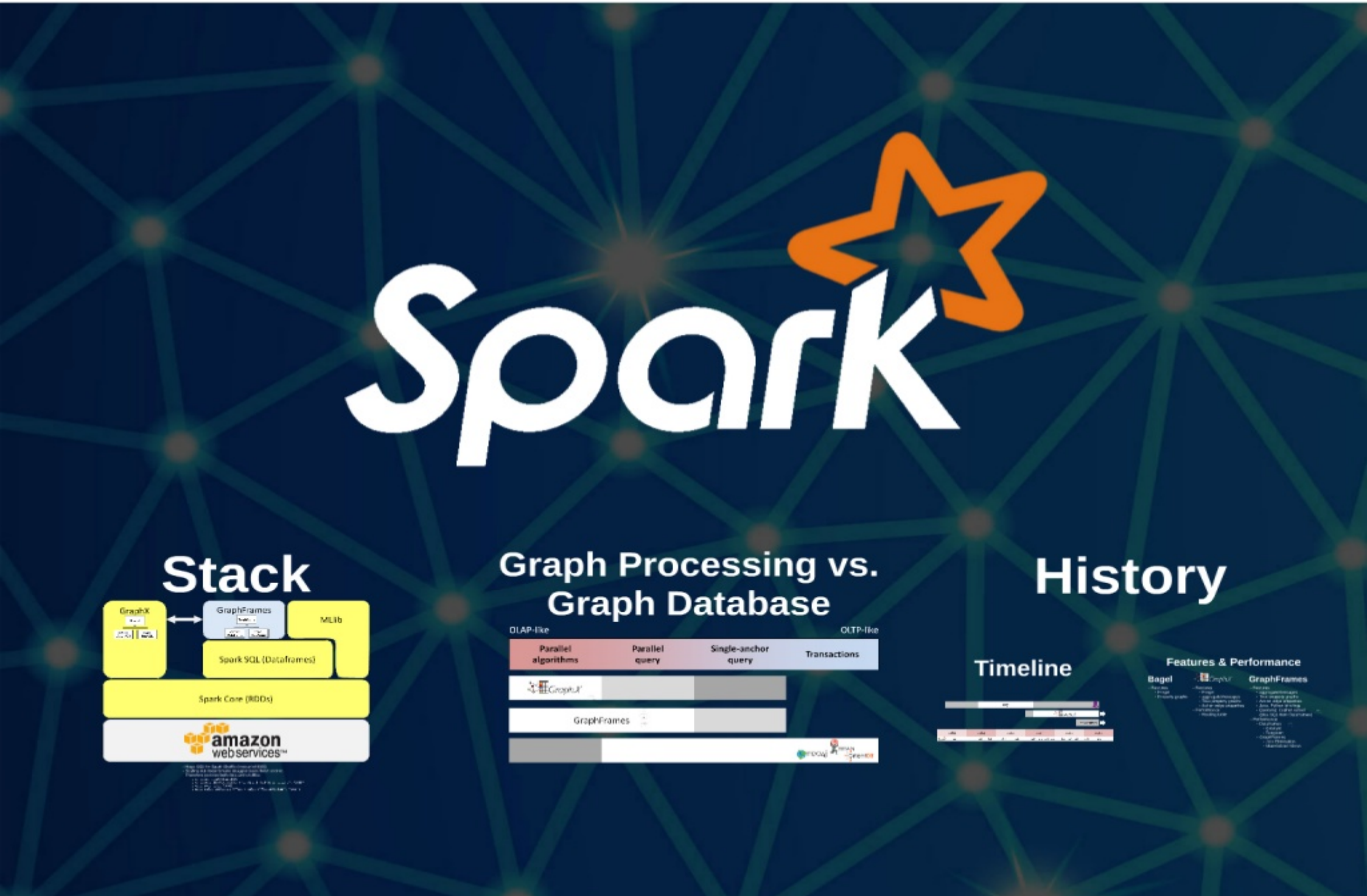

Graph Processing vs. Graph Database


History
Timeline

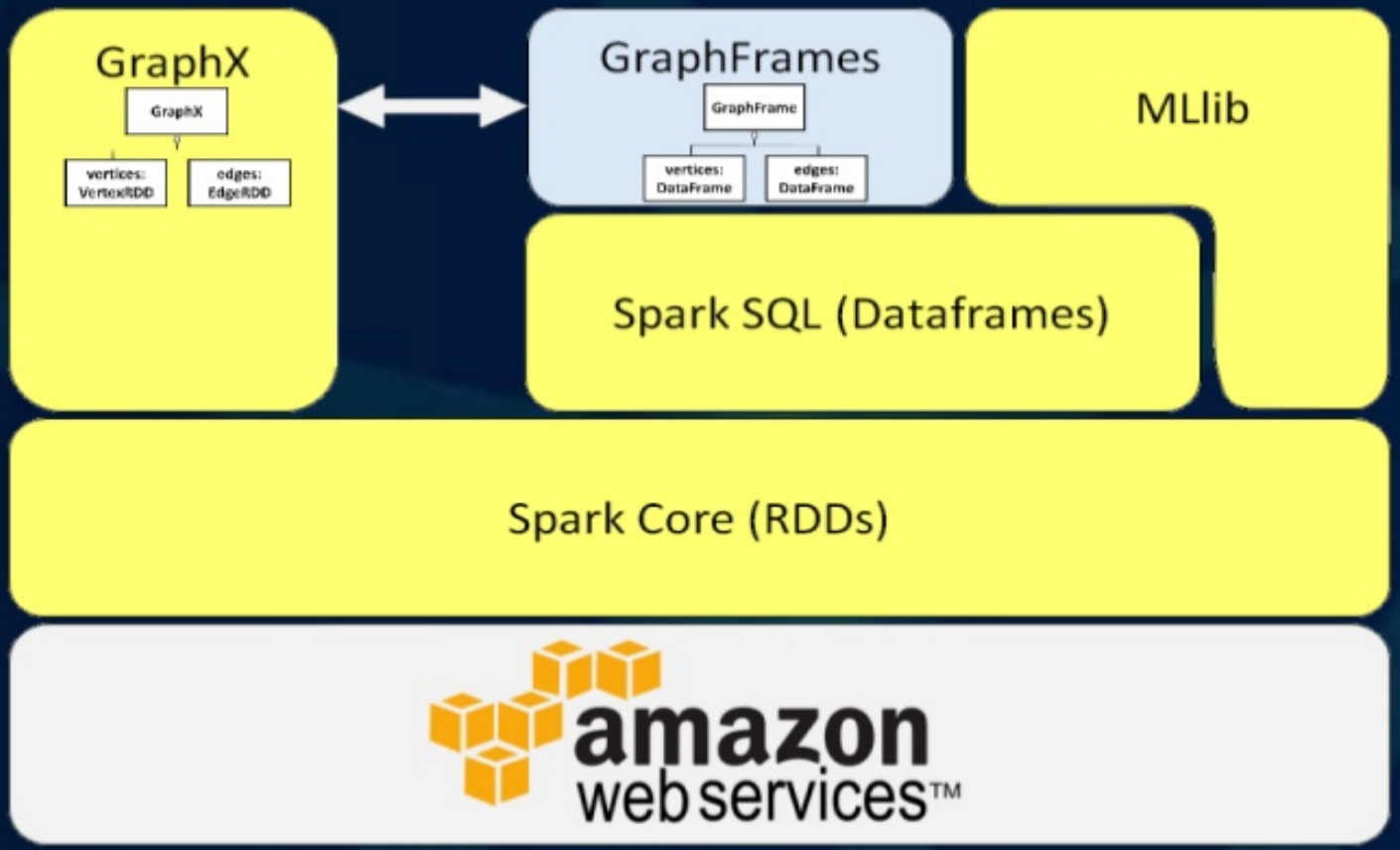

Isomorphisms
Definition
Detection
Finding


yago
select knowledge
Yet Another Great Ontology (YAGO)
How it's Distributed


Summary

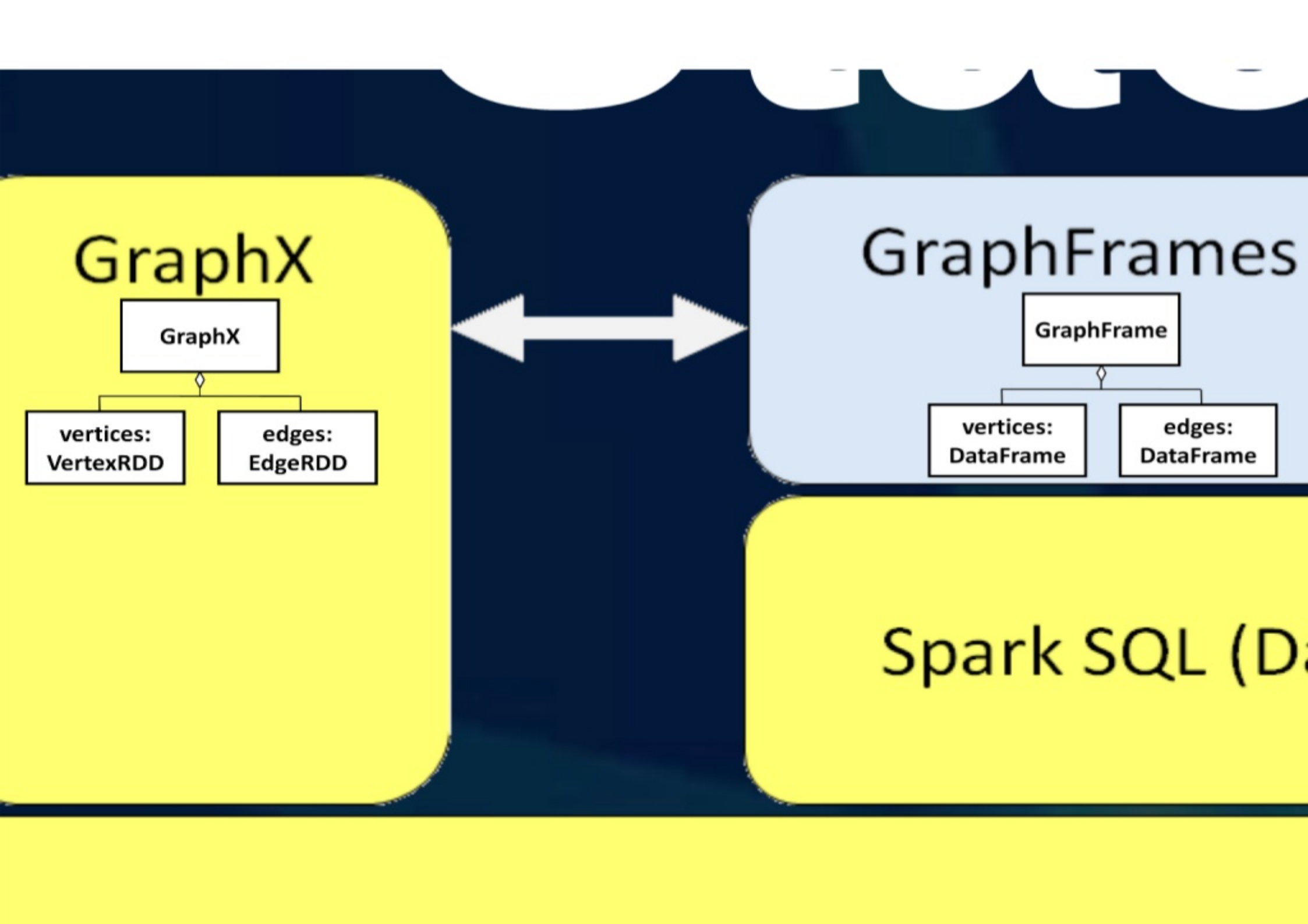

Spark
Stack
GraphX
GraphFrames
MLlib
Spark SQL (Dataframes)
Spark Core (RDDs)
amazon
webservices
Graph Processing vs.
Graph Database
OLAP-like
OLTP-like
Parallel
algorithms
Parallel
query
Single-anchor
query
Transactions
GraphFrames
History
Timeline
Features & Performance
Bagel
GraphFrames

Stack

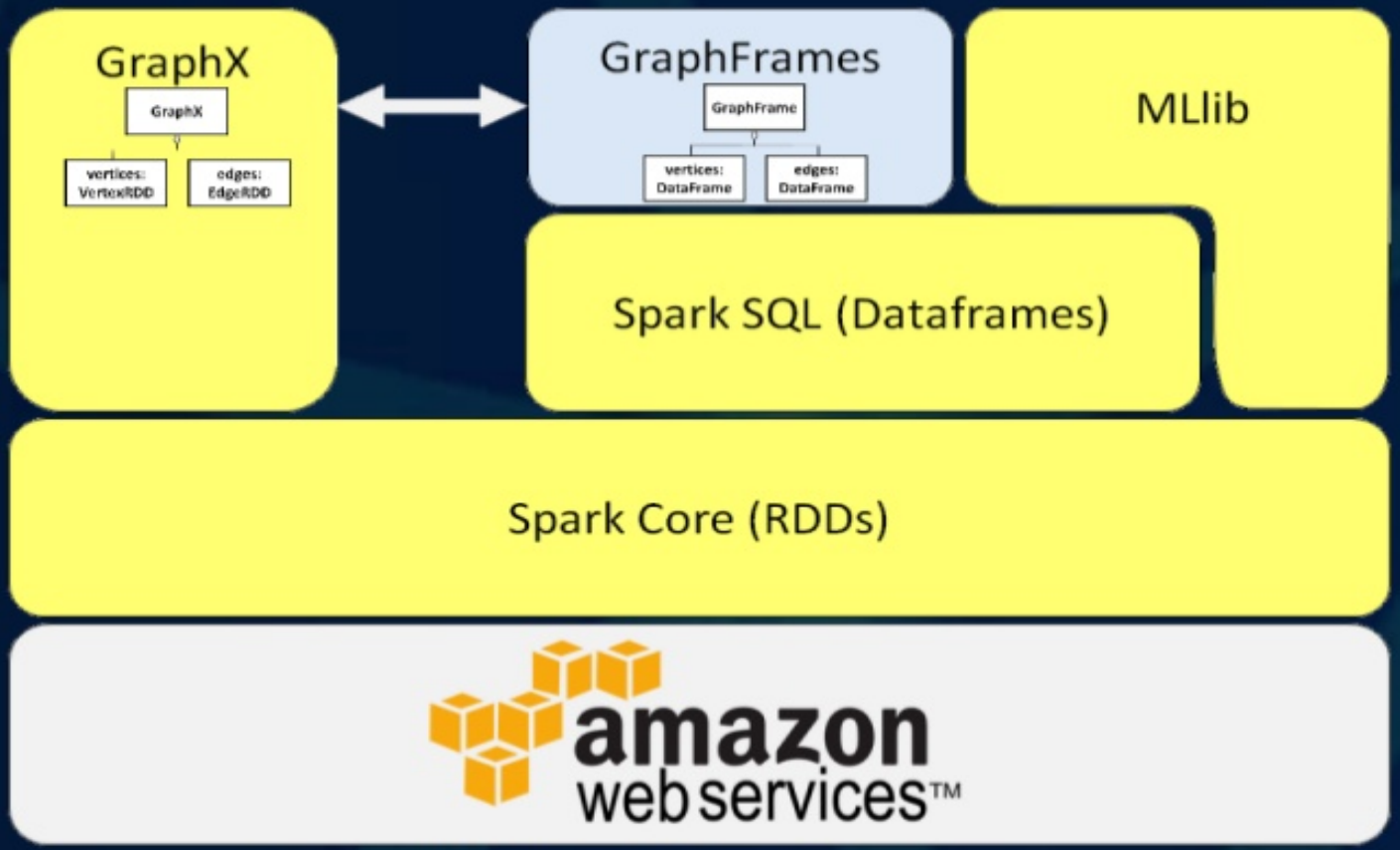


- Huge SSD for Spark Shuffle (instead of EBS)
- Scaling out doesn't make straggler tasks finish sooner.
  Therefore partition both data and shuffles:

```
g.vertices.repartition(1000)
g.vertices.sqlContext.setConf("spark.sql.shuffle.partitions", "1000")
g.edges.repartition(1000)
```


GraphX
GraphX
vertices:
VertexRDD
edges:
EdgeRDD
GraphFrames
GraphFrame
vertices:
DataFrame
edges:
DataFrame
Spark SQL (Da

Stack



- Huge SSD for Spark Shuffle (instead of EBS)
- Scaling out doesn't make straggler tasks finish sooner.
  Therefore partition both data and shuffles:

```
g.vertices.repartition(1000)
g.vertices.sqlContext.setConf("spark.sql.shuffle.partitions", "1000")
g.edges.repartition(1000)
```

Graph Processing vs. Graph Database

OLAP-like

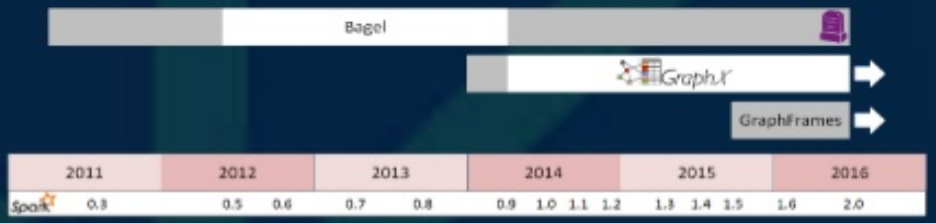
OLTP-like

Parallel algorithms	Parallel query	Single-anchor query	Transactions

GraphFrames

History

Timeline



Features & Performance

Bagel

- Features
  - Pregel
  - Property graphs

GraphX

- Features
  - Pregel
  - aggregateMessages
  - True property graphs
  - Act on edge properties
- Performance
  - Routing table

GraphFrames

- Features
  - aggregateMessages
  - True property graphs
  - Act on edge properties
  - Java, Python bindings
  - Querying: Cypher subset (plus SQL from Dataframes)
- Performance
  - Dataframes
    - Catalyst
    - Tungsten
  - GraphFrames
    - Join Elimination
    - Materialized Views

Timeline

2011	2012	2013	2014	2015	2016
Spark 0.3	0.5 0.6	0.7 0.8	0.9 1.0 1.1 1.2	1.3 1.4 1.5	1.6 2.0

Features & Performance

Bagel

- Features
  - Pregel
  - Property graphs

- Features
  - Pregel
  - aggregateMessages
  - True property graphs
  - Act on edge properties
- Performance
  - Routing table

GraphFrames

- Features
  - aggregateMessages
  - True property graphs
  - Act on edge properties
  - Java, Python bindings
  - Querying: Cypher subset (plus SQL from Dataframes)
- Performance
  - Dataframes
    - Catalyst
    - Tungsten
  - GraphFrames
    - Join Elimination
    - Materialized Views

Performance

Expected from Dataframes (Catalyst & Tungsten)

2012
Giraph 0.1

10x

2014
0.9 RDDs

10x?

2015
1.5 Dataframes

10x

2016
2.0 Dataframes

100x?

1000x?

Actual (anecdotal)

1.5 GraphX
(RDDs)
`readRDF()`

8x

1.5 Graphframes
(Dataframes)
`readRDF()`

Before:
- Spark 2.0 improvements
- Upcoming Graphframes improvements

Features & Performance

Bagel

- Features
  - Pregel
  - Property graphs

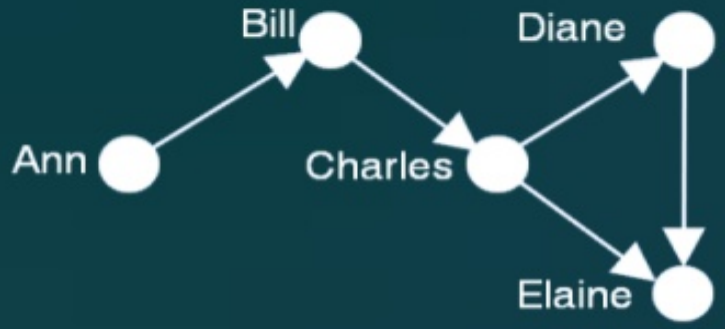
- Features
  - Pregel
  - aggregateMessages
  - True property graphs
  - Act on edge properties
- Performance
  - Routing table

GraphFrames

- Features
  - aggregateMessages
  - True property graphs
  - Act on edge properties
  - Java, Python bindings
  - Querying: Cypher subset (plus SQL from Dataframes)
- Performance
  - Dataframes
    - Catalyst
    - Tungsten
  - GraphFrames
    - Join Elimination
    - Materialized Views

Querying

Who are the friends of friends of Ann?



GraphX

```
val g2 = g.outerJoinVertices(g.aggregateMessages[Int](
  ctx => if (ctx.srcAttr == "Ann" &&
             ctx.attr == "knows") ctx.sendToDst(1),
  math.max(_,_)))((vid, vname, d) =>
    (vname, d.getOrElse(0)))
g2.outerJoinVertices(g2.aggregateMessages[Int](
  ctx => if (ctx.srcAttr._2 == 1 &&
             ctx.attr == "knows") ctx.sendToDst(2),
  math.max(_,_)))((vid, vname, d) =>
    (vname, d.getOrElse(0))).
  vertices.map(_._2).filter(_._2 == 2)
          .map(_._1._1).collect
```

Neo4j

```
MATCH (ann { name: 'Ann' })-[:knows*2..2]-(p)
RETURN p
```

GraphFrames

```
val gf = GraphFrame.fromGraphX(myGraph)
gf.find("(u)-[e1]->(v); (v)-[e2]->(w)")
  .filter("e1.attr = 'is-friends-with' AND " +
          "e2.attr = 'is-friends-with' AND " +
          "u.attr='Ann'")
  .select("w.attr")
  .collect
  .map(_(0).toString)
```

Neo4j

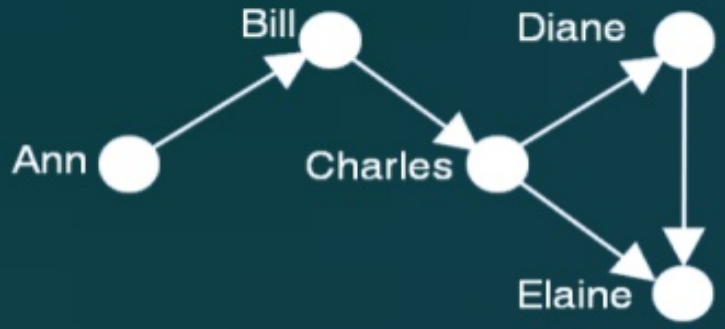
```
MATCH (ann { name: 'Ann' })-[:knows*2..2]-(p)
RETURN p
```

GraphFrames

```
val gf = GraphFrame.fromGraphX(myGraph)
gf.find("(u)-[e1]->(v); (v)-[e2]->(w)")
  .filter("e1.attr = 'is-friends-with' AND " +
          "e2.attr = 'is-friends-with' AND " +
          "u.attr='Ann'")
  .select("w.attr")
  .collect
  .map(_(0).toString)
```

Querying

Who are the friends of friends of Ann?



GraphX

```
val g2 = g.outerJoinVertices(g.aggregateMessages[Int](
  ctx => if (ctx.srcAttr == "Ann" &&
             ctx.attr == "knows") ctx.sendToDst(1),
  math.max(_,_)))((vid, vname, d) =>
    (vname, d.getOrElse(0)))
g2.outerJoinVertices(g2.aggregateMessages[Int](
  ctx => if (ctx.srcAttr._2 == 1 &&
             ctx.attr == "knows") ctx.sendToDst(2),
  math.max(_,_)))((vid, vname, d) =>
    (vname, d.getOrElse(0))).
  vertices.map(_._2).filter(_._2 == 2)
          .map(_._1._1).collect
```

Neo4j

```
MATCH (ann { name: 'Ann' })-[:knows*2..2]-(p)
RETURN p
```

GraphFrames

```
val gf = GraphFrame.fromGraphX(myGraph)
gf.find("(u)-[e1]->(v); (v)-[e2]->(w)")
  .filter("e1.attr = 'is-friends-with' AND " +
          "e2.attr = 'is-friends-with' AND " +
          "u.attr='Ann'")
  .select("w.attr")
  .collect
  .map(_(0).toString)
```

Features & Performance

Bagel

- Features
  - Pregel
  - Property graphs

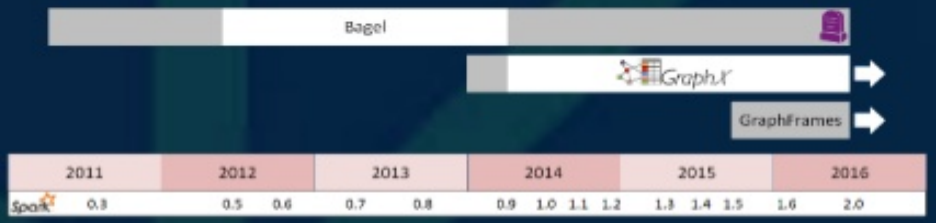
- Features
  - Pregel
  - aggregateMessages
  - True property graphs
  - Act on edge properties
- Performance
  - Routing table

GraphFrames

- Features
  - aggregateMessages
  - True property graphs
  - Act on edge properties
  - Java, Python bindings
  - Querying: Cypher subset (plus SQL from Dataframes)
- Performance
  - Dataframes
    - Catalyst
    - Tungsten
  - GraphFrames
    - Join Elimination
    - Materialized Views

History

Timeline



Features & Performance

Bagel

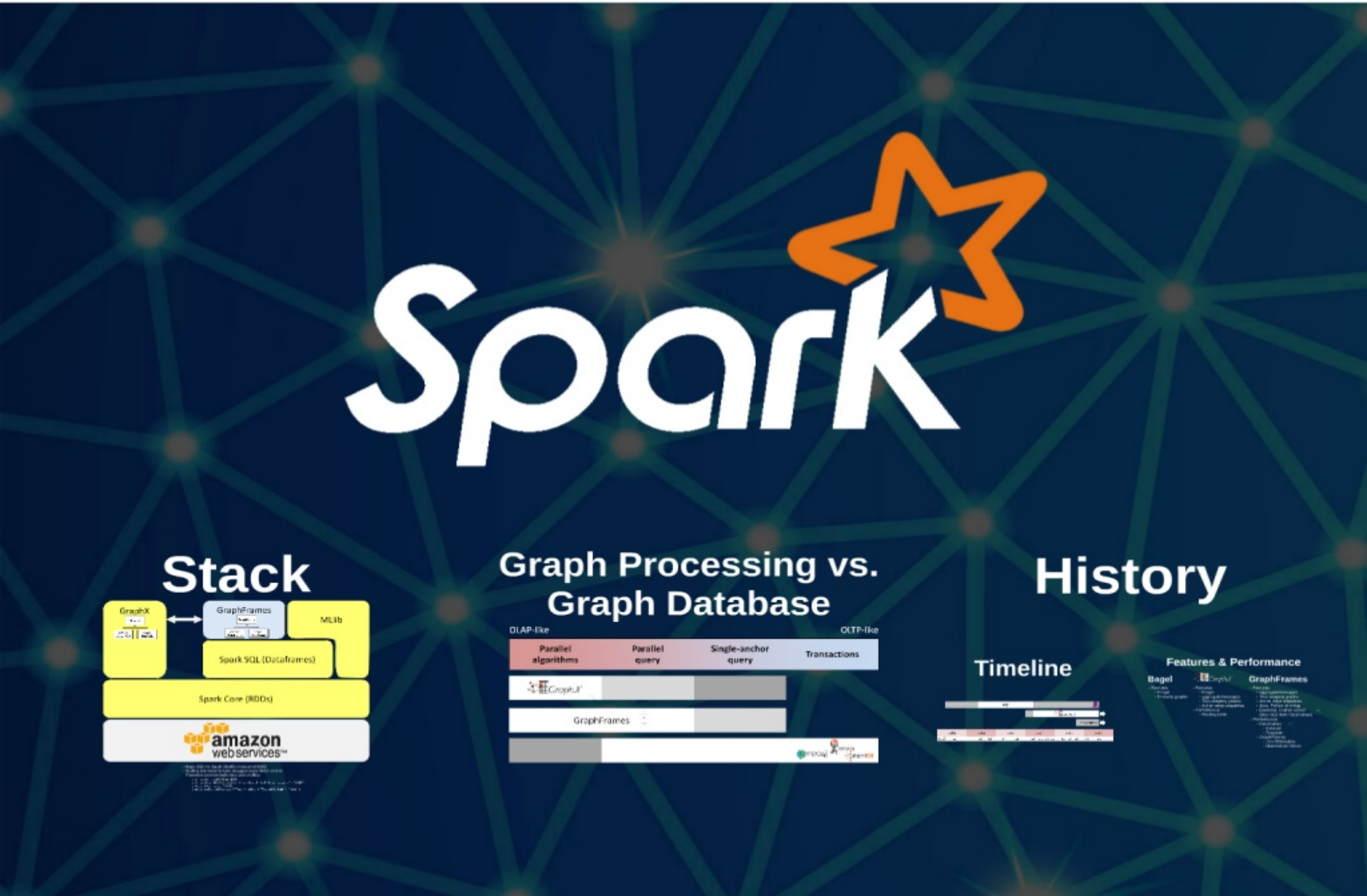
- Features
  - Pregel
  - Property graphs

GraphX

- Features
  - Pregel
  - aggregateMessages
  - True property graphs
  - Act on edge properties
- Performance
  - Routing table

GraphFrames

- Features
  - aggregateMessages
  - True property graphs
  - Act on edge properties
  - Java, Python bindings
  - Querying: Cypher subset (plus SQL from Dataframes)
- Performance
  - Dataframes
    - Catalyst
    - Tungsten
  - GraphFrames
    - Join Elimination
    - Materialized Views


Spark
Stack
GraphX
GraphFrames
MLlib
Spark SQL (Dataframes)
Spark Core (RDDs)
amazon
webservices
Graph Processing vs.
Graph Database
OLAP-like
OLTP-like
Parallel
algorithms
Parallel
query
Single-anchor
query
Transactions
GraphFrames
History
Timeline
Features & Performance
Bagel
GraphFrames

Isomorphisms

Definition

from cs.stonybrook.edu

Detection

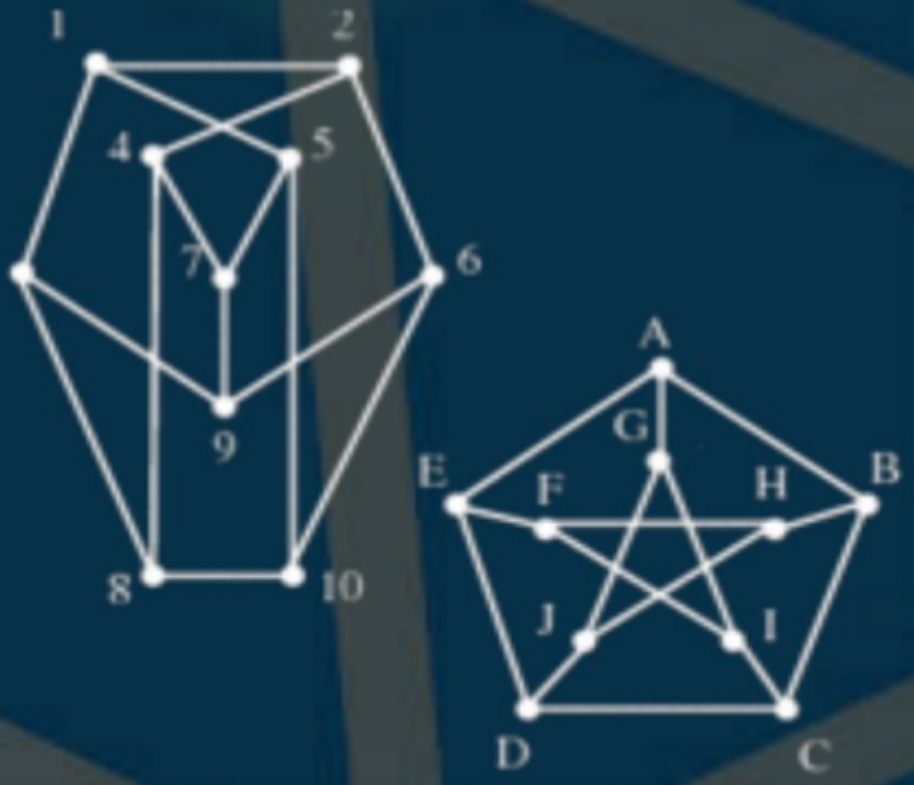
(Not the problem we're (I'm) trying to solve)

Finding

Rule Mining

- Inductive Logic Programming
- Network Analysis
- Social Network Analysis
- Social Link Analysis

Definition

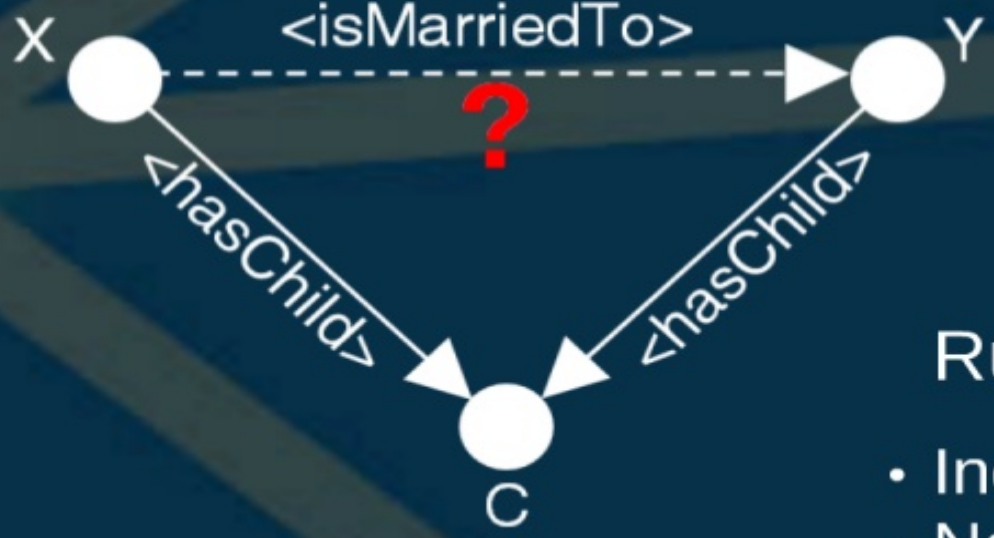


from cs.stonybrook.edu

Detection

(Not the problem we're (I'm) trying to solve)

Finding



Rule Mining

- Inductive Logic Programming
- Network Analysis
- Social Network Analysis
- Social Link Analysis

Isomorphisms

Definition

from cs.stonybrook.edu

Detection

(Not the problem we're (I'm) trying to solve)

Finding

Rule Mining

- Inductive Logic Programming
- Network Analysis
- Social Network Analysis
- Social Link Analysis

YAGO

select knowledge

Yet Another Great Ontology (YAGO)

GeoNames

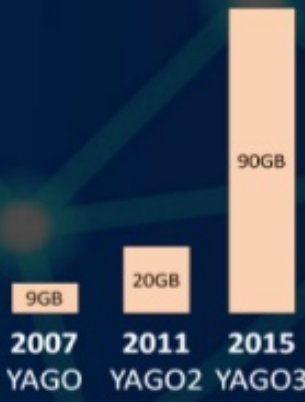
500 million edges

DBpedia

WIKIPEDIA
The Free Encyclopedia

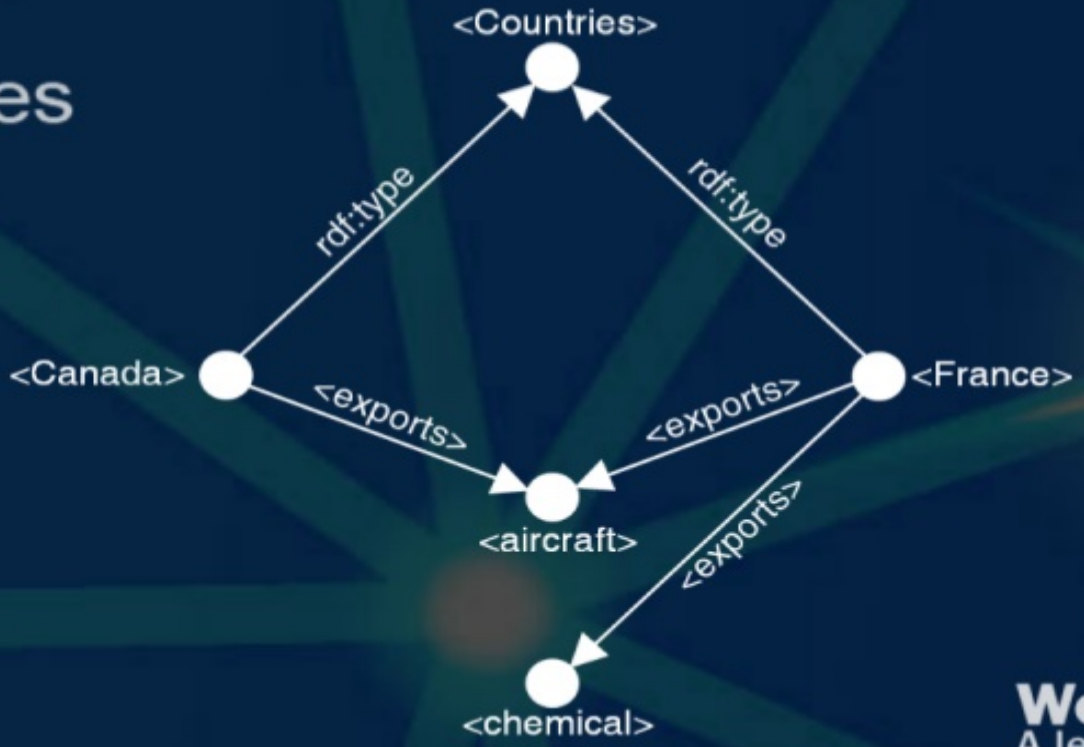
WordNet
A lexical database for English

How It's Distributed



Yet Another Great Ontology (YAGO)

GeoNames



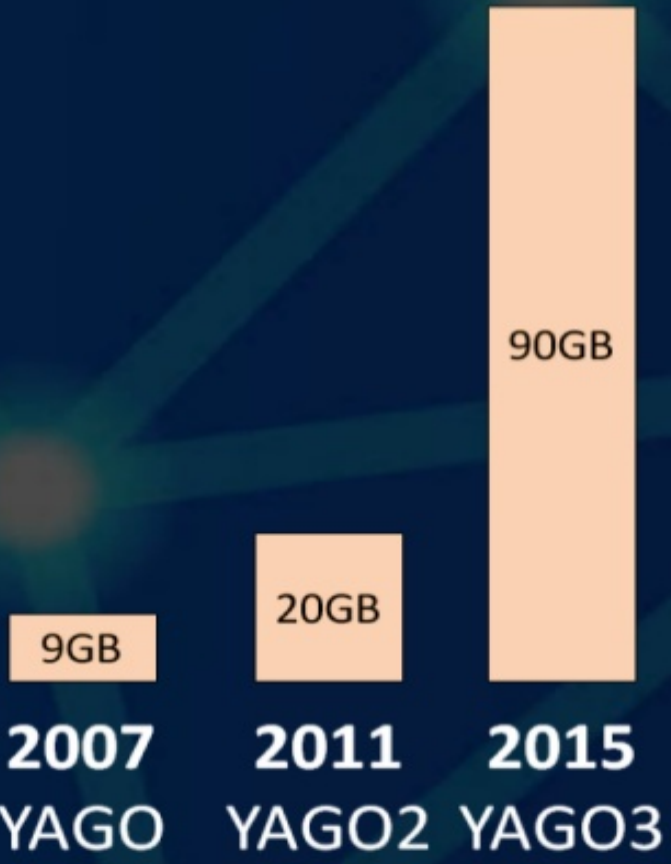
WIKIPEDIA
The Free Encyclopedia

DBpedia

WordNet
A lexical database for English

500 million edges

How It's Distributed



76 .tsv Files in YAGO3

yagoFacts.tsv: 425 MB

yagoLabels.tsv: 3348 MB

yagoTaxonomy.tsv: 62 MB

Functions Needed To Read YAGO

76 .tsv Files in YAGO3

yagoFacts.tsv: 425 MB

```
<Friedrich_Wilhelm_Joseph_Schelling>  <influences>  <Alexander_von_Humboldt>
<Jefferson_County,_Texas>             <owns>        <Jack_Brooks_Regional_Airport>
```

yagoLabels.tsv: 3348 MB

```
<Francesco_I_Crispo>    rdfs:label      "Francesco Ier Crispo"@fra
<Monroe_City_Schools>   skos:prefLabel  "Monroe City Schools"@eng
```

yagoTaxonomy.tsv: 62 MB

```
<wikicat_Birds_of_Tunisia>  rdfs:subClassOf  <wordnet_bird_101503061>
<wordnet_knot_113885836>    rdfs:subClassOf  <wordnet_distorted_shape_113867276>
```

Functions Needed To Read YAGO

readRdf()

1. ZipWithIndex: Number the vertices
2. Join: convert source vertex name to ID
3. Join: convert destination vertex name to ID

GraphX

```
def readRdf(sc:org.apache.spark.SparkContext, filename:String) = {
  val r = sc.textFile(filename).map(_.split("\t"))
  val v = r.map(_(1)).union(r.map(_(3))).distinct.zipWithIndex
  Graph(v.map(_.swap),
        r.map(x => (x(1),(x(2),x(3))))
         .join(v)
         .map(x => (x._2._1._2,(x._2._2,x._2._1._1)))
         .join(v)
         .map(x => new Edge(x._2._1._1, x._2._2, x._2._1._2)))
}
```

GraphFrames

```
def readRdf(sc:org.apache.spark.SparkContext, filename:String) = {
  val r = sc.textFile(filename).map(_.split("\t"))
  val v = r.map(_(1)).union(r.map(_(3))).distinct.zipWithIndex.map(
                        x => Row(x._2,x._1)).cache
  val stv = StructType(StructField("id",LongType) ::
                       StructField("attr",StringType) :: Nil)
  val vdf = sqlContext.createDataFrame(v,stv).cache
  val str = StructType(StructField("edfId",StringType) ::
                       StructField("subject",StringType) ::
                       StructField("predicate",StringType) ::
                       StructField("object",StringType) :: Nil)
  val edf = sqlContext.createDataFrame(r.map(Row.fromSeq(_)),str)
                      .join(vdf, $"subject" === $"attr")
                      .selectExpr("id AS src", "predicate", "object")
                      .join(vdf, $"object" === $"attr")
                      .selectExpr("src", "id AS dst", "predicate AS attr")
  v.unpersist(false)
  GraphFrame(vdf,edf)
}
```

mergeGraphs()

GraphX

```
def mergeGraphs(g1:Graph[String,String], g2:Graph[String,String]) = {
  val v = g1.vertices.map(_._2).union(g2.vertices.map(_._2)).distinct
                    .zipWithUniqueId
  def edgeWithNewVertexIds(g:Graph[String,String]) =
    g.triplets
     .map(et => (et.srcAttr, (et.attr,et.dstAttr)))
     .join(v)
     .map(x => (x._2._1._2, (x._2._1._1,x._2._2)))
     .join(v)
     .map(x => new Edge(x._2._1._1,x._2._2,x._2._1._2))
  Graph(v.map(_.swap),
        edgeWithNewVertexIds(g1).union(edgeWithNewVertexIds(g2)))
}
```

GraphFrames

```
def mergeGraphs(g1:GraphFrame, g2:GraphFrame) = {
  val vertices = g1.vertices.sqlContext.createDataFrame(
    g1.vertices.select($"attr").unionAll(g2.vertices.select($"attr"))
      .distinct.rdd.zip(1L to Long.MaxValue)
      .map(r => Row.fromSeq(r._1.toSeq :+ r._2)),
    new StructType(Array(StructField("id",LongType),
                         StructField("attr",StringType))))
  .cache

  def edgesWithNewVertexIds(g:GraphFrame) = {
    g.triplets
     .join(vertices, $"src.attr" === $"attr")
     .selectExpr("id AS src","edge.attr AS edgeAttr",
                 "dst.attr AS dstAttr")
     .join(vertices, $"dstAttr" === $"attr")
     .selectExpr("src","id AS dst","edgeAttr AS attr")
  }

  GraphFrame(vertices, edgesWithNewVertexIds(g1).unionAll(
                       edgesWithNewVertexIds(g2)))
}
```

readRdf()

1. ZipWithIndex: Number the vertices
2. Join: convert source vertex name to ID
3. Join: convert destination vertex name to ID

GraphX

```
def readRdf(sc:org.apache.spark.SparkContext, filename:String) = {
  val r = sc.textFile(filename).map(_.split("\t"))
  val v = r.map(_(1)).union(r.map(_(3))).distinct.zipWithIndex
  Graph(v.map(_.swap),
        r.map(x => (x(1),(x(2),x(3))))
         .join(v)
         .map(x => (x._2._1._2,(x._2._2,x._2._1._1)))
         .join(v)
         .map(x => new Edge(x._2._1._1, x._2._2, x._2._1._2)))
}
```

GraphFrames

```
def readRdfDf(sc:org.apache.spark.SparkContext, filename:String) = {
  val r = sc.textFile(filename).map(_.split("\t"))
  val v = r.map(_(1)).union(r.map(_(3))).distinct.zipWithIndex.map(
                  x => Row(x._2,x._1)).cache
  val stv = StructType(StructField("id",LongType) ::
                       StructField("attr",StringType) :: Nil)
  val vdf = sqlContext.createDataFrame(v,stv).cache
  val str = StructType(StructField("rdfId",StringType) ::
                       StructField("subject",StringType) ::
                       StructField("predicate",StringType) ::
                       StructField("object",StringType) :: Nil)
  val edf = sqlContext.createDataFrame(r.map(Row.fromSeq(_)),str)
                      .join(vdf, $"subject" === $"attr")
                      .selectExpr("id AS src", "predicate", "object")
                      .join(vdf, $"object" === $"attr")
                      .selectExpr("src", "id AS dst", "predicate AS attr")
  v.unpersist(false)
  GraphFrame(vdf,edf)
}
```

mergeGraphs()

GraphX

```
def mergeGraphs(g1:Graph[String,String], g2:Graph[String,String]) = {
  val v = g1.vertices.map(_._2).union(g2.vertices.map(_._2)).distinct
                     .zipWithIndex
  def edgesWithNewVertexIds(g:Graph[String,String]) =
    g.triplets
     .map(et => (et.srcAttr, (et.attr,et.dstAttr)))
     .join(v)
     .map(x => (x._2._1._2, (x._2._2,x._2._1._1)))
     .join(v)
     .map(x => new Edge(x._2._1._1,x._2._2,x._2._1._2))
  Graph(v.map(_.swap),
        edgesWithNewVertexIds(g1).union(edgesWithNewVertexIds(g2)))
}
```

GraphFrames

```
def mergeGraphs(g1:GraphFrame, g2:GraphFrame) = {
  val vunion = g1.vertices.sqlContext.createDataFrame(
    g1.vertices.select($"attr").unionAll(g2.vertices.select($"attr"))
      .distinct.rdd.map(_(0))
      .zipWithIndex.map(t => Row.fromTuple(t.swap)),
    new StructType(Array(StructField("id",LongType),
                         StructField("attr",StringType)))
  ).cache

  def edgesWithNewVertexIds(gf:GraphFrame) = {
    gf.triplets
      .join(vunion, $"src.attr" === $"attr")
      .selectExpr("id AS src","edge.attr AS edgeattr",
                  "dst.attr AS dstattr")
      .join(vunion, $"dstattr" === $"attr")
      .selectExpr("src","id AS dst","edgeattr AS attr")
  }

  GraphFrame(vunion, edgesWithNewVertexIds(g1).unionAll(
                       edgesWithNewVertexIds(g2)))
}
```

Functions Needed To Read YAGO

readRdf()

1. ZipWithIndex: Number the vertices
2. Join: convert source vertex name to ID
3. Join: convert destination vertex name to ID

GraphX

```
def readRdf(sc:org.apache.spark.SparkContext, filename:String) = {
  val r = sc.textFile(filename).map(_.split("\t"))
  val v = r.map(_(1)).union(r.map(_(3))).distinct.zipWithIndex
  Graph(v.map(_.swap),
        r.map(x => (x(1),(x(2),x(3))))
         .join(v)
         .map(x => (x._2._1._2,(x._2._2,x._2._1._1)))
         .join(v)
         .map(x => new Edge(x._2._1._1, x._2._2, x._2._1._2)))
}
```

GraphFrames

```
def readRdf(sc:org.apache.spark.SparkContext, filename:String) = {
  val r = sc.textFile(filename).map(_.split("\t"))
  val v = r.map(_(1)).union(r.map(_(3))).distinct.zipWithIndex.map(
                        x => Row(x._2,x._1)).cache
  val stv = StructType(StructField("id",LongType) ::
                       StructField("attr",StringType) :: Nil)
  val vdf = sqlContext.createDataFrame(v,stv).cache
  val str = StructType(StructField("edfId",StringType) ::
                       StructField("subject",StringType) ::
                       StructField("predicate",StringType) ::
                       StructField("object",StringType) :: Nil)
  val edf = sqlContext.createDataFrame(r.map(Row.fromSeq(_)),str)
                      .join(vdf, $"subject" === $"attr")
                      .selectExpr("id AS src", "predicate", "object")
                      .join(vdf, $"object" === $"attr")
                      .selectExpr("src", "id AS dst", "predicate AS attr")
  v.unpersist(false)
  GraphFrame(vdf,edf)
}
```

mergeGraphs()

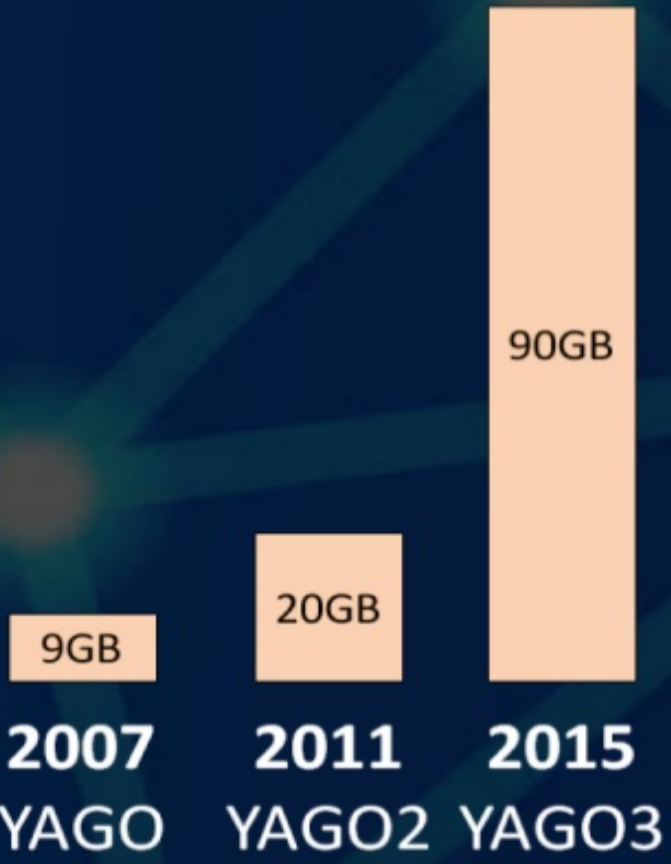
GraphX

```
def mergeGraphs(g1:Graph[String,String], g2:Graph[String,String]) = {
  val v = g1.vertices.map(_._2).union(g2.vertices.map(_._2)).distinct
                    .zipWithUniqueId
  def edgeWithNewVertexIds(g:Graph[String,String]) =
    g.triplets
     .map(et => (et.srcAttr, (et.attr,et.dstAttr)))
     .join(v)
     .map(x => (x._2._1._2, (x._2._1._1,x._2._2)))
     .join(v)
     .map(x => new Edge(x._2._1._1,x._2._2,x._2._1._2))
  Graph(v.map(_.swap),
        edgeWithNewVertexIds(g1).union(edgeWithNewVertexIds(g2)))
}
```

GraphFrames

```
def mergeGraphs(g1:GraphFrame, g2:GraphFrame) = {
  val vertices = g1.vertices.sqlContext.createDataFrame(
    g1.vertices.select($"attr").unionAll(g2.vertices.select($"attr"))
      .distinct.rdd.zip(1L to Long.MaxValue)
      .map(r => Row.fromSeq(r._1.toSeq :+ r._2)),
    new StructType(Array(StructField("id",LongType),
                         StructField("attr",StringType))))
  .cache

  def edgesWithNewVertexIds(g:GraphFrame) = {
    g.triplets
     .join(vertices, $"src.attr" === $"attr")
     .selectExpr("id AS src","edge.attr AS edgeAttr",
                 "dst.attr AS dstAttr")
     .join(vertices, $"dstAttr" === $"attr")
     .selectExpr("src","id AS dst","edgeAttr AS attr")
  }

  GraphFrame(vertices, edgesWithNewVertexIds(g1).unionAll(
                       edgesWithNewVertexIds(g2)))
}
```

How It's Distributed



76 .tsv Files in YAGO3

yagoFacts.tsv: 425 MB

yagoLabels.tsv: 3348 MB

yagoTaxonomy.tsv: 62 MB

Functions Needed To Read YAGO

YAGO

select knowledge

Yet Another Great Ontology (YAGO)

GeoNames

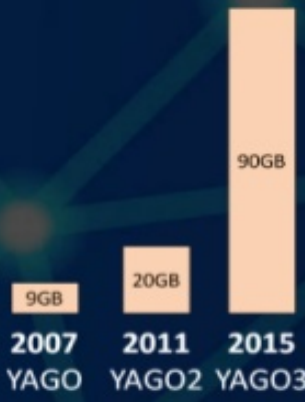
500 million edges

DBpedia

WIKIPEDIA
The Free Encyclopedia

WordNet
A lexical database for English

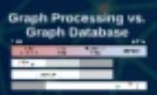
How It's Distributed




Finding Graph Isomorphisms in GraphX and GraphFrames
Michael Malak
ORACLE


Spark


Stack

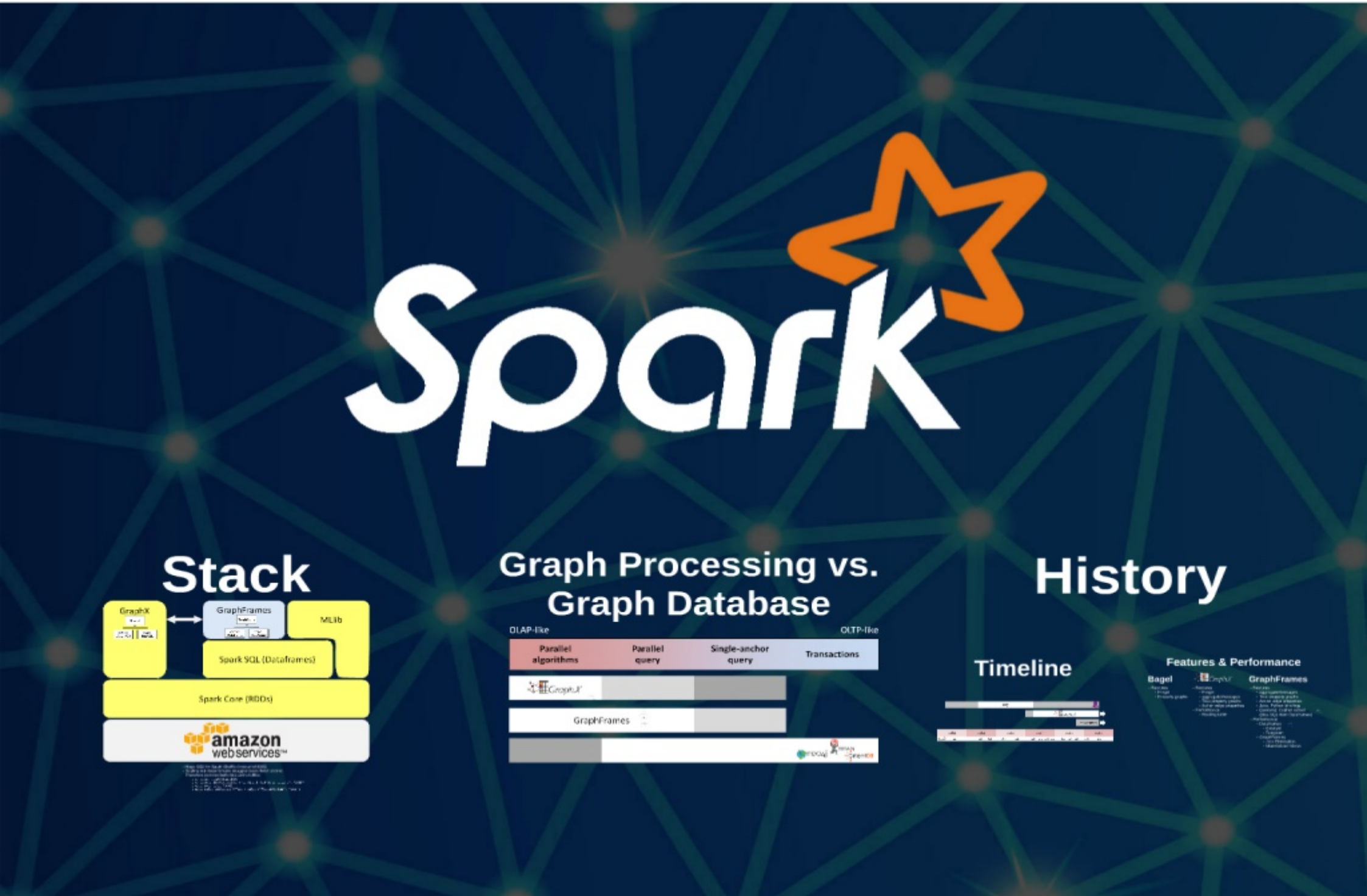

Graph Processing vs. Graph Database


History
Timeline


Isomorphisms
Definition
Detection
Finding


yago
select knowledge
Yet Another Great Ontology (YAGO)
How it's Distributed


Summary


Spark
Stack
GraphX
GraphFrames
MLlib
Spark SQL (Dataframes)
Spark Core (RDDs)
amazon
webservices
Graph Processing vs.
Graph Database
OLAP-like
OLTP-like
Parallel
algorithms
Parallel
query
Single-anchor
query
Transactions
GraphFrames
History
Timeline
Features & Performance
Bagel
GraphFrames

Graph Processing vs. Graph Database

OLAP-like

OLTP-like

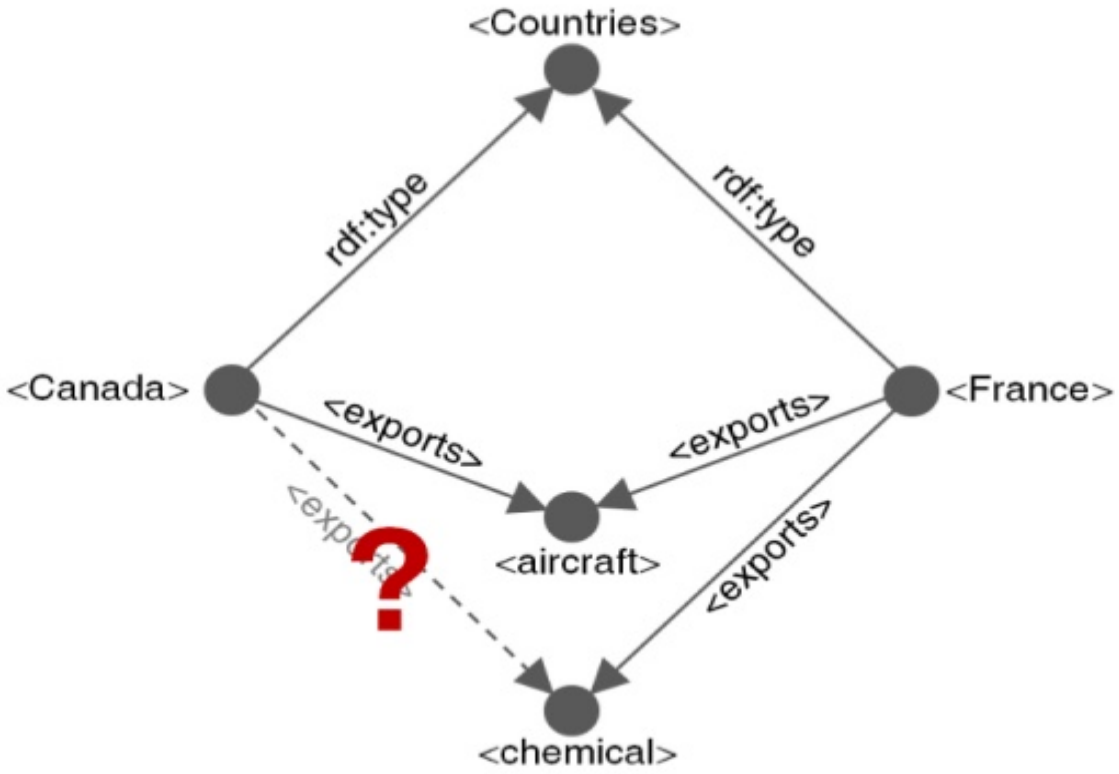
Parallel algorithms	Parallel query	Single-anchor query	Transactions

GraphFrames

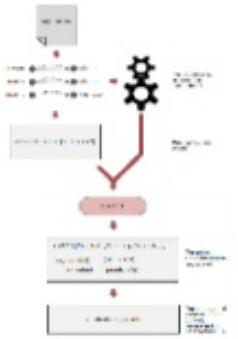

GraphX


Country Exports Example

Country Exports Example

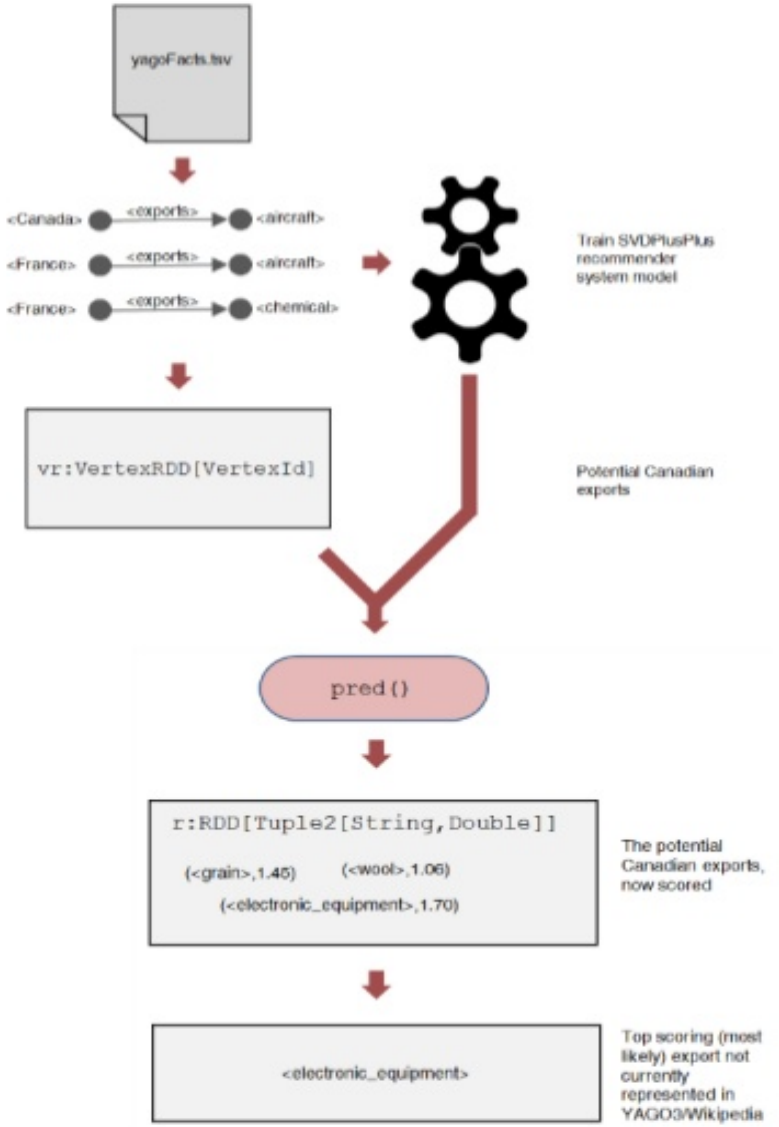


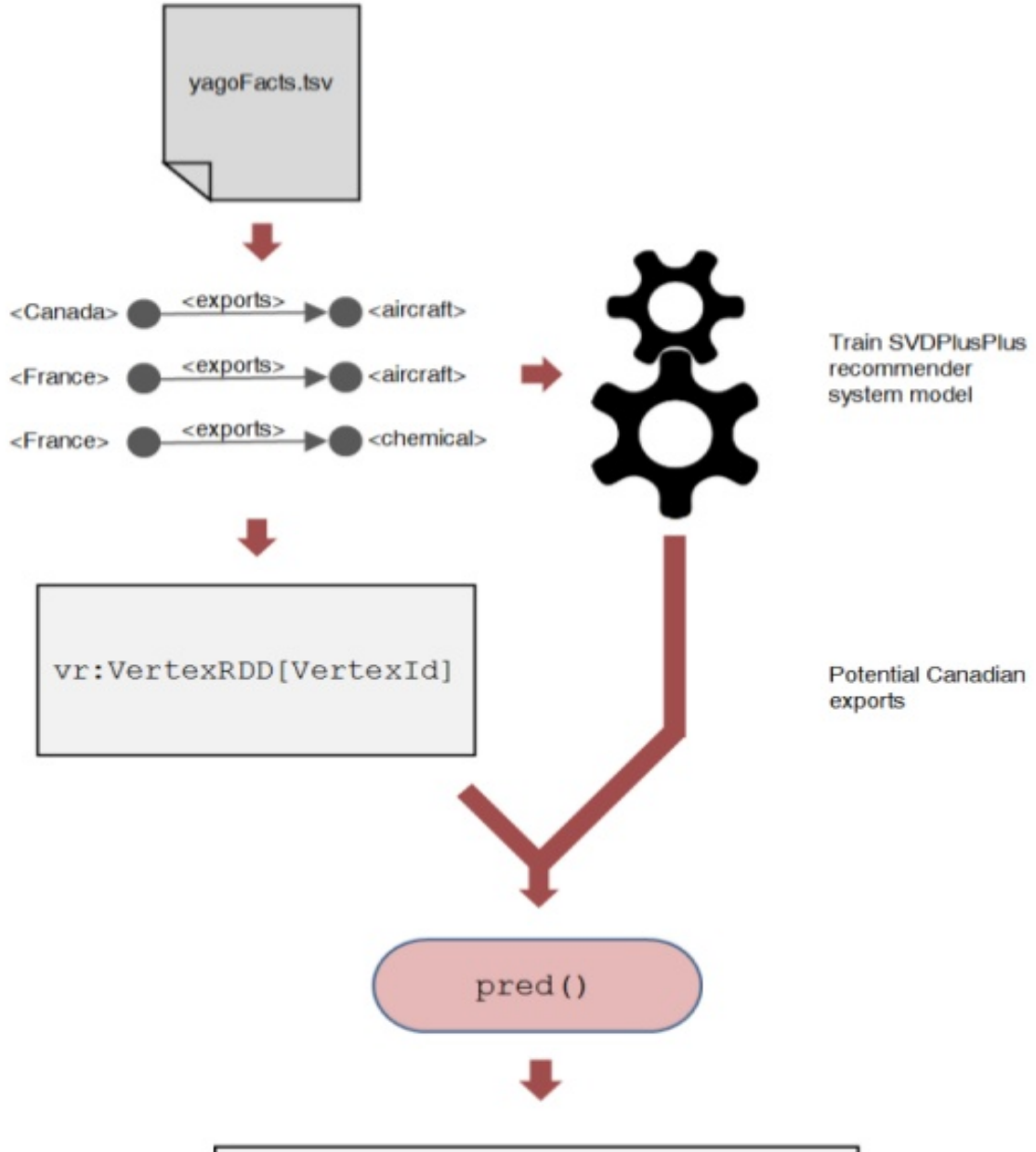
Dataflow

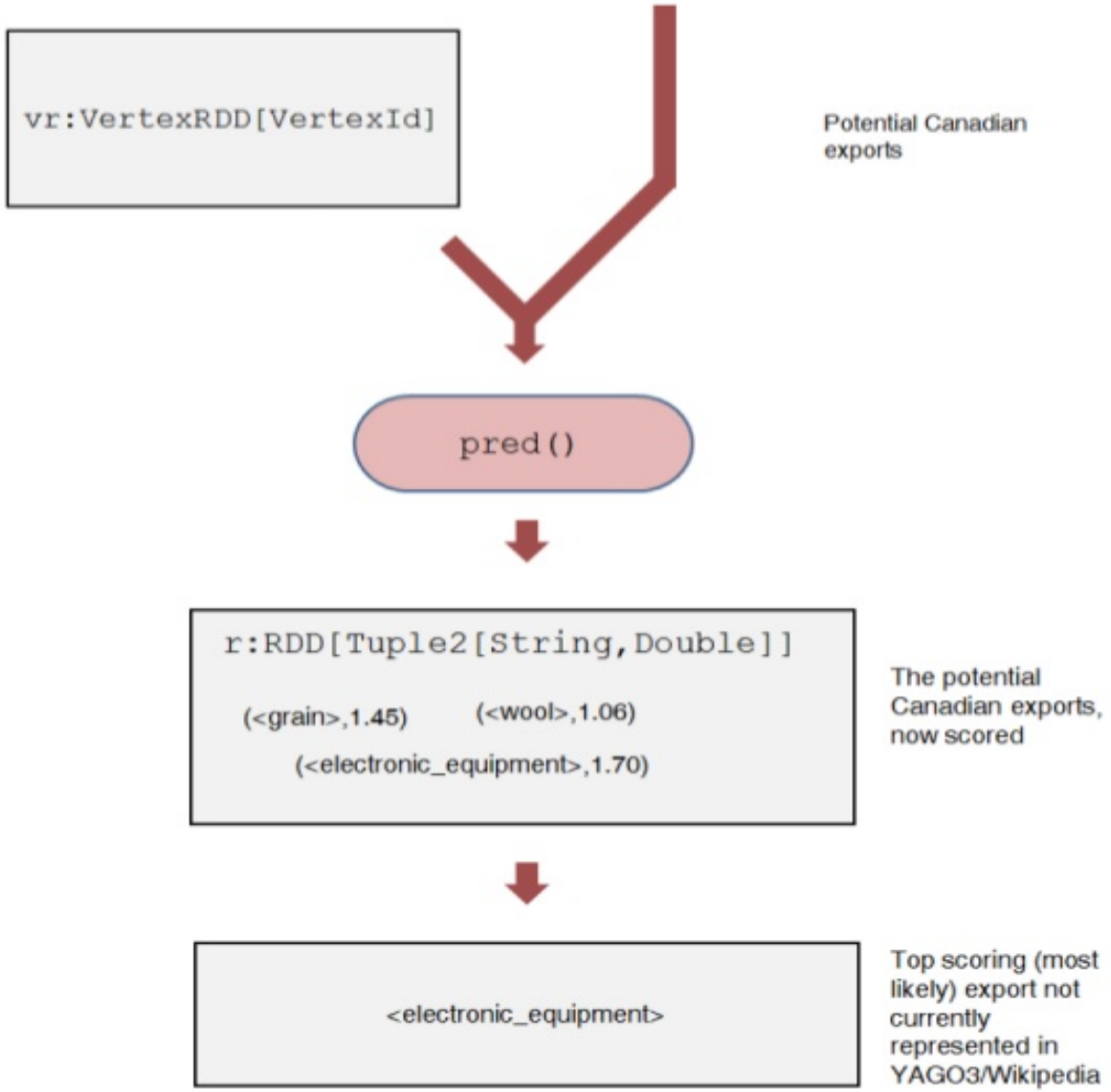


Results

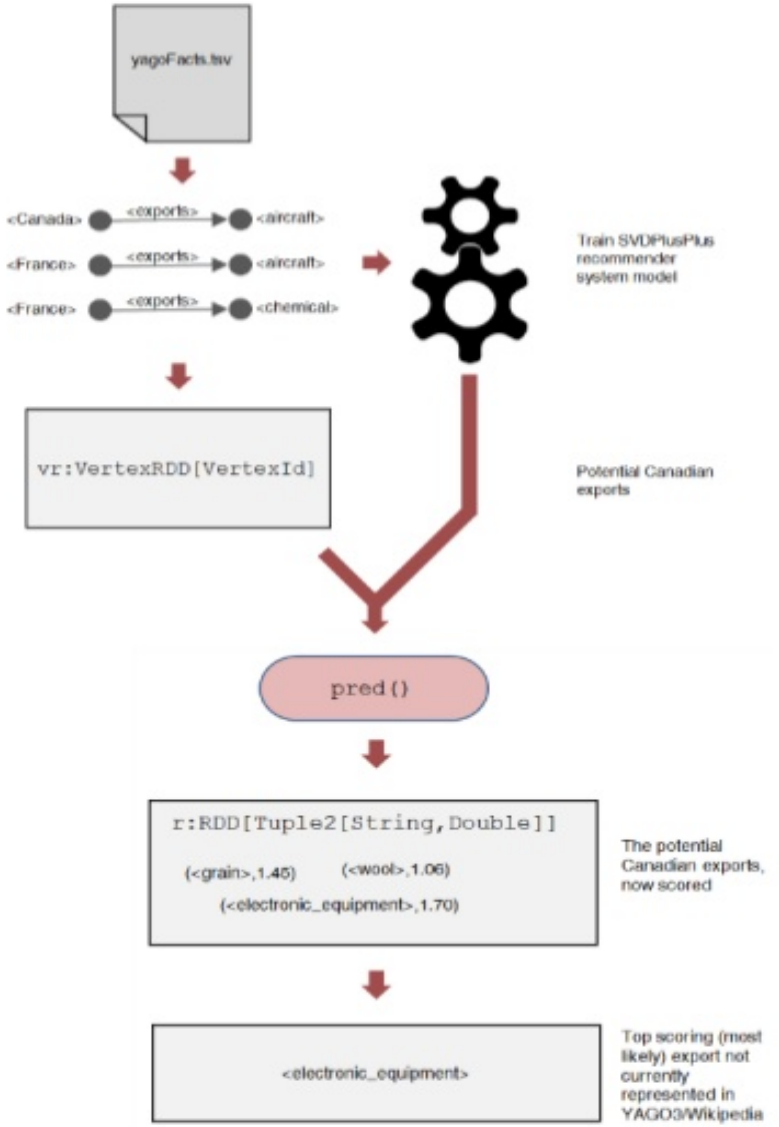
Dataflow


yagoFacts.tsv
<Canada> <exports> <aircraft>
<France> <exports> <aircraft>
<France> <exports> <chemical>
Train SVDPlusPlus recommender system model
vr:VertexRDD[VertexId]
Potential Canadian exports
pred()
r:RDD[Tuple2[String,Double]]
(<grain>,1.45) (<wool>,1.06)
(<electronic_equipment>,1.70)
The potential Canadian exports, now scored
<electronic_equipment>
Top scoring (most likely) export not currently represented in YAGO3/Wikipedia


yagoFacts.tsv
<Canada> <exports> <aircraft>
<France> <exports> <aircraft>
<France> <exports> <chemical>
Train SVDPlusPlus recommender system model
vr:VertexRDD[VertexId]
Potential Canadian exports
pred()


vr:VertexRDD[VertexId]
Potential Canadian exports
pred()
r:RDD[Tuple2[String,Double]]
(<grain>,1.45)
(<wool>,1.06)
(<electronic_equipment>,1.70)
The potential Canadian exports, now scored
<electronic_equipment>
Top scoring (most likely) export not currently represented in YAGO3/Wikipedia

Dataflow


yagoFacts.tsv
<Canada> <exports> <aircraft>
<France> <exports> <aircraft>
<France> <exports> <chemical>
Train SVDPlusPlus recommender system model
vr:VertexRDD[VertexId]
Potential Canadian exports
pred()
r:RDD[Tuple2[String,Double]]
(<grain>,1.45) (<wool>,1.06)
(<electronic_equipment>,1.70)
The potential Canadian exports, now scored
<electronic_equipment>
Top scoring (most likely) export not currently represented in YAGO3/Wikipedia

Results

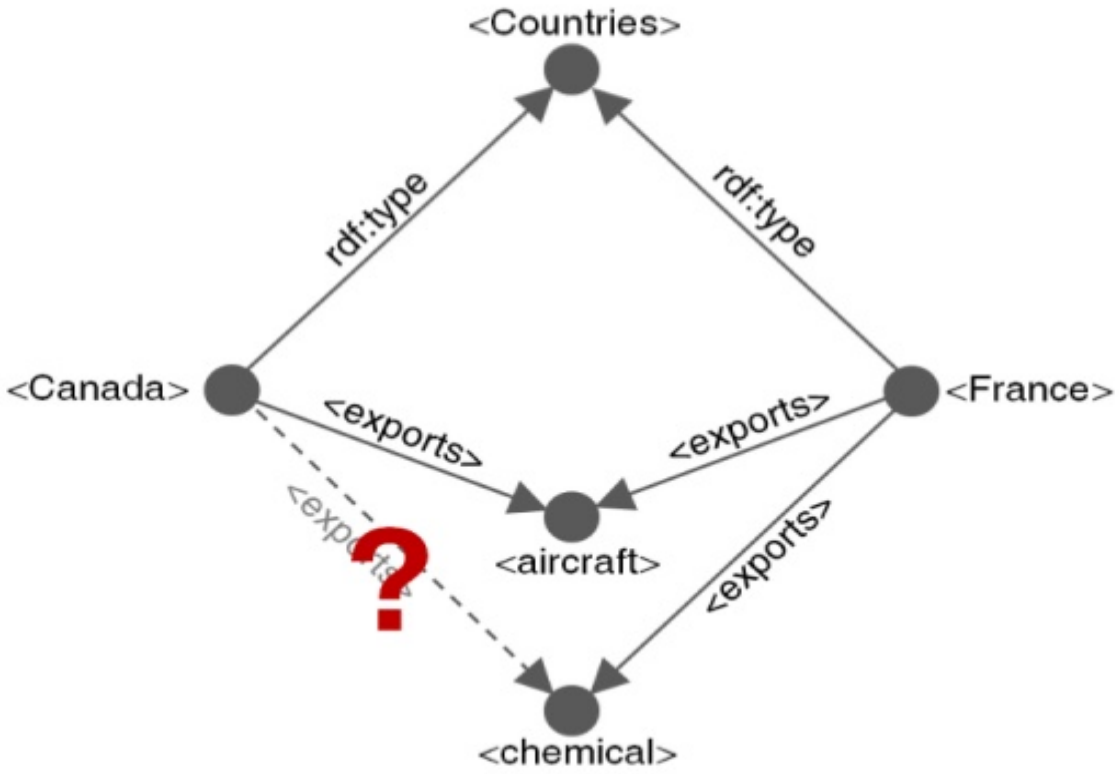
```
val cid = gf.vertices.filter(_._2 == "<Canada>").first._1
val r = vr.map(v => (v,pred(vm,mean,cid,v)))

val maxKey = r.max()(new Ordering[Tuple2[VertexId, Double]]() {
  override def compare(x: (VertexId, Double), y: (VertexId, Double)): Int =
      Ordering[Double].compare(x._2, y._2)
}).1

gf.vertices.filter(_._1 == maxKey).collect

res0: Array[(org.apache.spark.graphx.VertexId, String)] =
 Array((1721488,<wordnet_electronic_equipment_103278248>))
```

```
grep "<exports>" yagoFacts.tsv | grep "<Canada>"
<id_1wrx1wu_dv6_1pgb7a4> <Canada> <exports> <wordnet_aluminum_114627820>
<id_1wrx1wu_dv6_j2l8e6>  <Canada> <exports> <wordnet_electricity_111449907>
<id_1wrx1wu_dv6_t6wmo1>  <Canada> <exports> <wordnet_lumber_114943580>
<id_1wrx1wu_dv6_jhowo0>  <Canada> <exports> <wordnet_natural_gas_114960090>
<id_1wrx1wu_dv6_s9bzqx>  <Canada> <exports> <wordnet_aircraft_102686568>
<id_1wrx1wu_dv6_12fzkgg> <Canada> <exports> <wordnet_plastic_114592610>
```

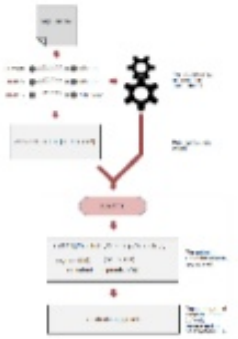
worldstopexports.com

1. Oil: US$77.8 billion (19% of total exports)
2. Vehicles: $60 billion (14.7%)
3. Machines, engines, pumps: $31.1 billion (7.6%)
4. Gems, precious metals: $19 billion (4.7%)
5. Electronic equipment: $13.2 billion (3.2%)
6. Plastics: $12.5 billion (3.1%)
7. Aircraft, spacecraft: $12.3 billion (3%)
8. Wood: $11.8 billion (2.9%)
9. Aluminum: $8.2 billion (2%)
10. Paper: $7.7 billion (1.9%)

Country Exports Example



Dataflow



Results


GraphX


Country Exports Example

Graph Processing vs. Graph Database

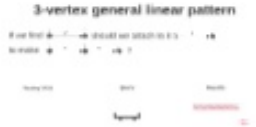
OLAP-like

OLTP-like

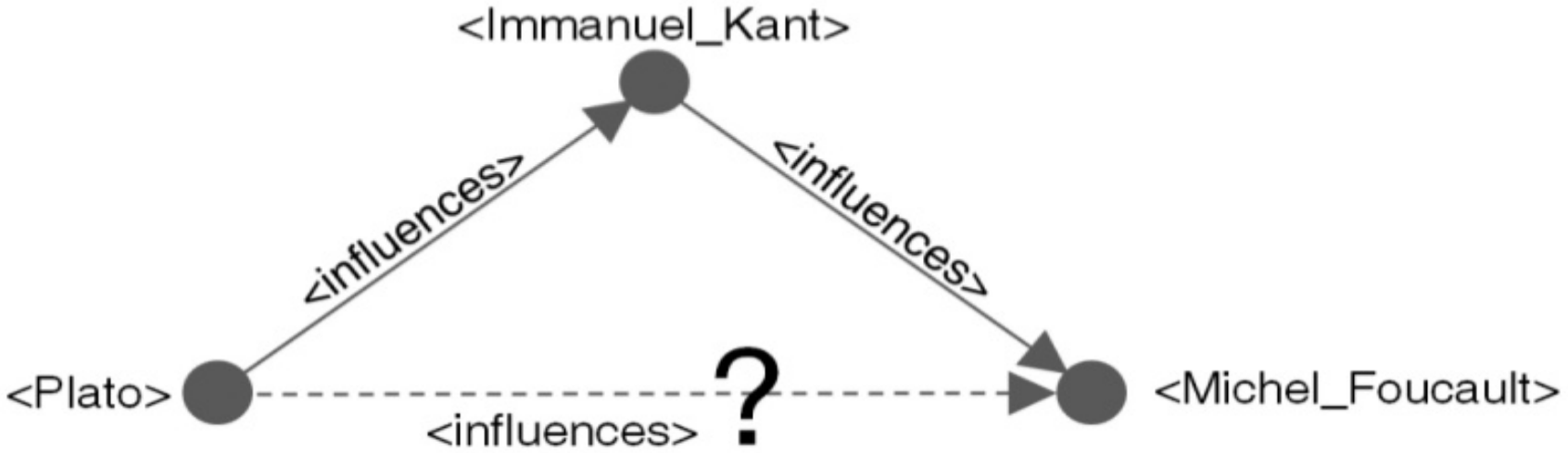
Parallel algorithms	Parallel query	Single-anchor query	Transactions

GraphFrames

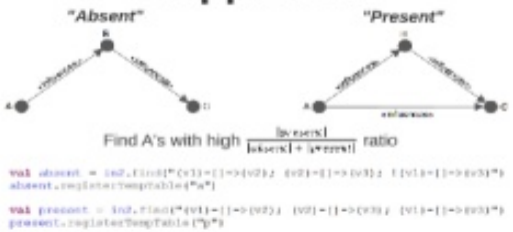
GraphFrames



3-vertex <influences>



Approach

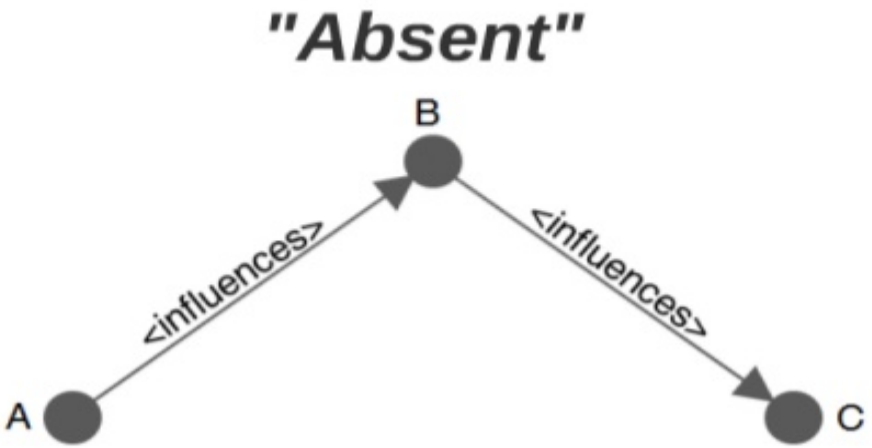


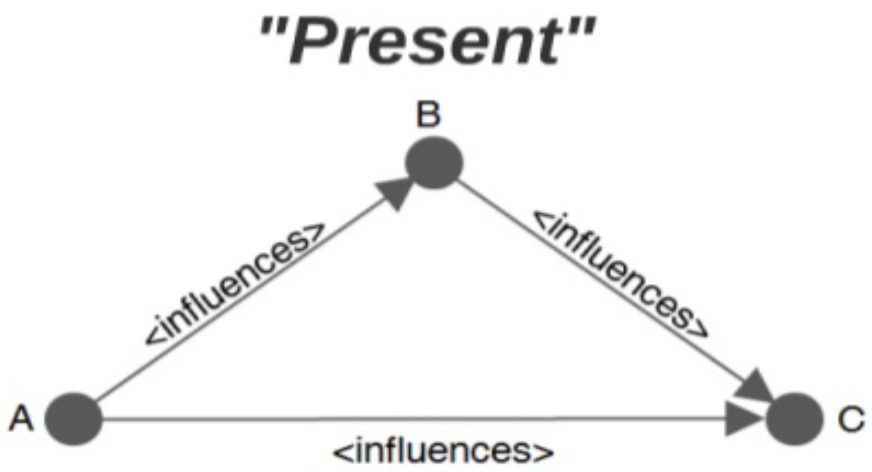
Outdegree in SQL

Score the A's

For Plato as A, score the C's

Approach





Find A's with high $\frac{|present|}{|absent| + |present|}$ ratio

```
val absent = in2.find("(v1)-[]->(v2); (v2)-[]->(v3); !(v1)-[]->(v3)")
absent.registerTempTable("a")

val present = in2.find("(v1)-[]->(v2); (v2)-[]->(v3); (v1)-[]->(v3)")
present.registerTempTable("p")
```

Outdegree in SQL

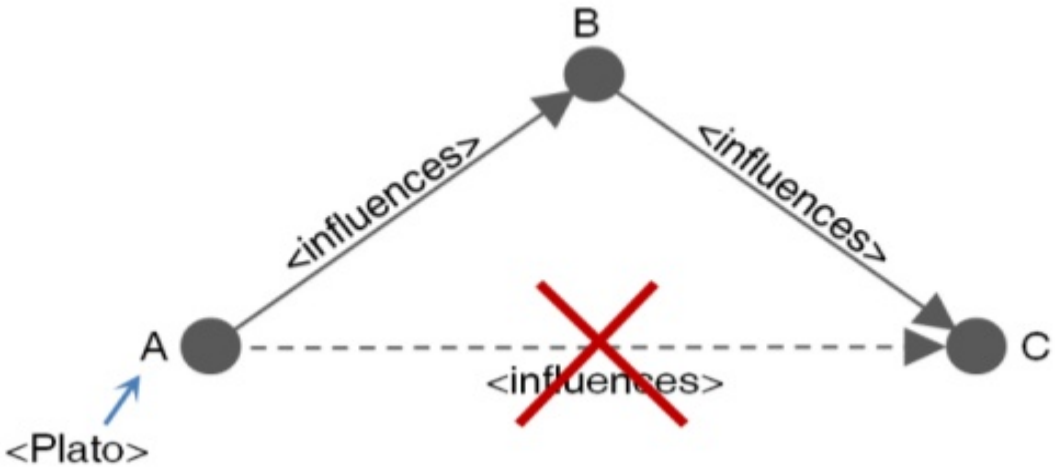
```
grep "<influences>" yagoFacts.tsv >yagoFactsInfluences.tsv

val in = readRdfDf(sc, "yagoFactsInfluences.tsv")

in.edges.registerTempTable("e")
in.vertices.registerTempTable("v")

val in2 = GraphFrame(in.vertices.sqlContext.sql(
                        "SELECT v.id," +
                        "       FIRST(v.attr) AS attr," +
                        "       COUNT(*) AS outdegree " +
                        "FROM   v " +
                        "JOIN   e " +
                        "  ON   v.id=e.src " +
                        "GROUP BY v.id").cache,
                     in.edges)
```

Score the A's

```
absent.sqlContext.sql(
  "SELECT v1 an," +
  "       SUM(v1.outdegree * v2.outdegree * v3.outdegree) AS ac " +
  "FROM   a " +
  "GROUP BY v1").registerTempTable("aa")

present.sqlContext.sql(
  "SELECT v1 pn," +
  "       SUM(v1.outdegree * v2.outdegree * v3.outdegree) AS pc " +
  "FROM   p " +
  "GROUP BY v1").registerTempTable("pa")

absent.sqlContext.sql("SELECT an," +
                      "       ac * pc/(ac+pc) AS score " +
                      "FROM   aa " +
                      "JOIN   pa" +
                      "  ON   an=pn " +
                      "ORDER BY score DESC").show
```

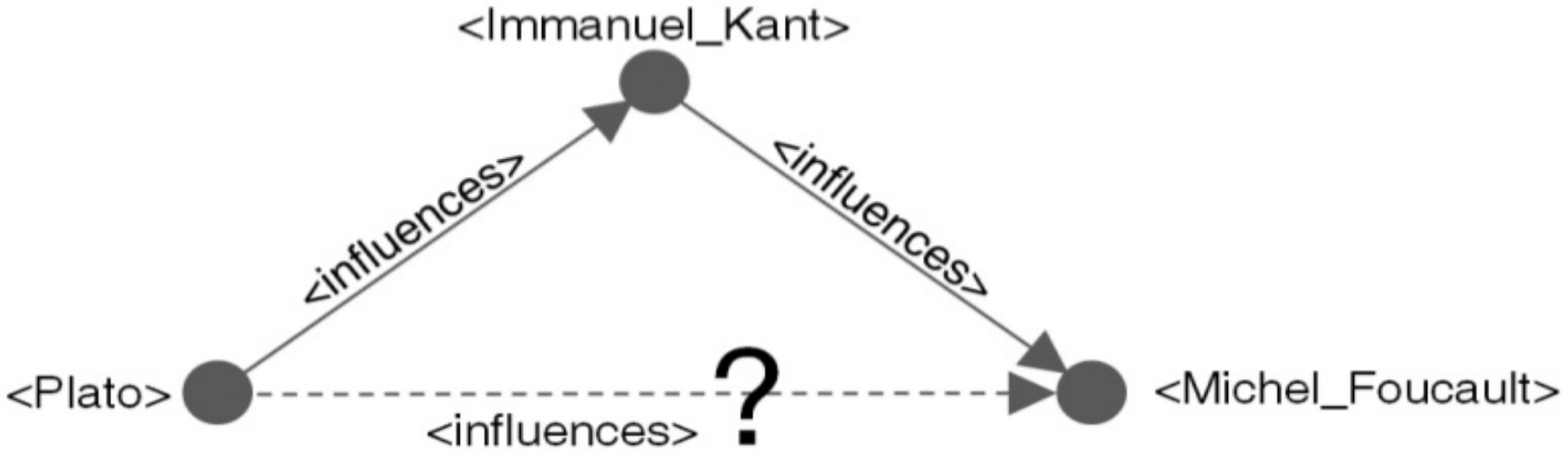
```
+--------------------+--------------------+
|                  an|               score|
+--------------------+--------------------+
|  [7662,<Plato>,102]| 3.822406412297308E7|
|[10648,<Aristotle...|3.2961326121938106E7|
|[4959,<Immanuel_K...|2.6445857520978764E7|
|[2961,<Georg_Wilh...|2.1092802441273782E7|
|[9304,<Baruch_Spi...|1.4513392385496272E7|
|[12217,<René_Desc...|1.2407118036818413E7|
|[12660,<Johann_Wo...|1.0109121178397963E7|
|[11895,<Jean-Jacq...|   9081581.748842742|
|[11615,<Gottfried...|   7146037.710399863|
|[2025,<Friedrich_...|  6897244.1896990575|
|[1082,<William_Sh...|   4168778.144288711|
|[11034,<Adam_Smit...|  4100936.5022027283|
|[1121,<John_Locke...|   3868447.819527024|
|[1566,<Heraclitus...|  3616900.3025887734|
|[3746,<Karl_Marx>...|   3575419.671920321|
|[10954,<Søren_Kie...|   3143375.914849735|
|[7322,<David_Hume...|  3122089.3473657905|
|[8540,<Arthur_Sch...|   2978239.727690162|
|[3186,<Ibn_Tufail...|   2234249.031615453|
|[8267,<Epicurus>,24]|  1812594.4073720106|
+--------------------+--------------------+
only showing top 20 rows
```

For Plato as A, score the C's

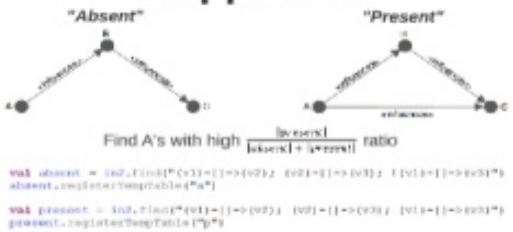



```
absent.sqlContext.sql(
  "SELECT v1.attr, " +
  "       v3.attr, " +
  "       SUM(v1.outdegree * v2.outdegree * v3.outdegree) AS score " +
  "FROM   a " +
            "WHERE  v1.id=7662 " +
            "GROUP BY v1.attr, v3.attr " +
            "ORDER BY score DESC").collect
res24: Array[org.apache.spark.sql.Row] =
Array([<Plato>,<Karl_Marx>,7139388],
[<Plato>,<Jean-Paul_Sartre>,3143640], [<Plato>,<Michel_Foucault>,2871606],
[<Plato>,<Gilles_Deleuze>,2689128], [<Plato>,<Henri_Bergson>,2179128],
[<Plato>,<Maurice_Merleau-Ponty>,2088450]...
```

3-vertex <influences>



Approach

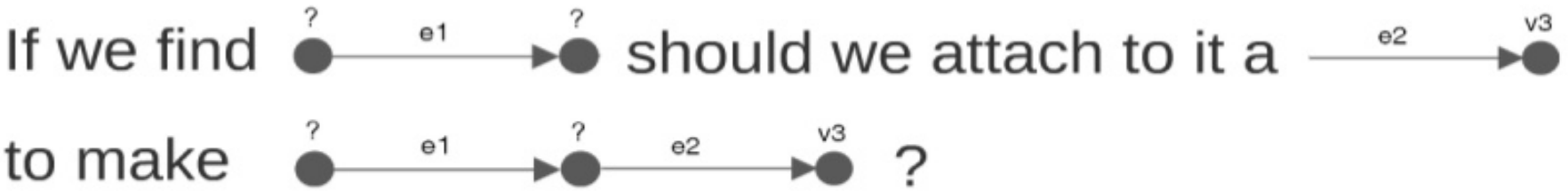


Outdegree in SQL

Score the A's

For Plato as A, score the C's

3-vertex general linear pattern

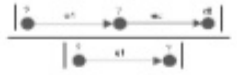


Reading YAGO

```
var yago = readNdfDf(sc, "/home/ec2-user/yago/yagoFacts.tsv")

def mergeIntoYago(filename:String):Unit = {
  val r = readNdfDf(sc, filename)
  r.vertices.cache
  r.edges.cache
  val g = mergeGraphs(yago, r)
  val g2 = GraphFrame(g.vertices, g.edges.distinct)
  g2.vertices.cache
  g2.vertices.count
  g2.edges.cache
  g2.edges.count
  yago.vertices.unpersist
  yago.edges.unpersist
  r.vertices.unpersist
  r.edges.unpersist
  yago = g2
}

mergeIntoYago("/home/ec2-user/yago/yagoLabels.tsv")
mergeIntoYago("/home/ec2-user/yago/yagoTaxonomy.tsv")
```

Query

```
yago.find("(v1) -[e1]-> (v2); (v2) -[e2]-> (v3)")
    .distinct
    .selectExpr("e1.attr AS e1attr", "e2.attr AS e2attr",
                "v2.id AS v2id", "v3.attr AS v3attr")
    .groupBy("e1attr", "e2attr", "v2id", "v3attr")
    .count
    .join(yago.edges.groupBy("attr")
                    .count
                    .selectExpr("attr", "count AS ccount"),
          $"e1attr" === $"attr")
    .selectExpr("e1attr", "e2attr", "v3attr", "count/ccount AS ratio")
    .orderBy($"ratio".desc)
    .show(50)
```



Results

Discovered bug in YAGO 3.02 (not in YAGO 3.00 or forthcoming YAGO 3.03)

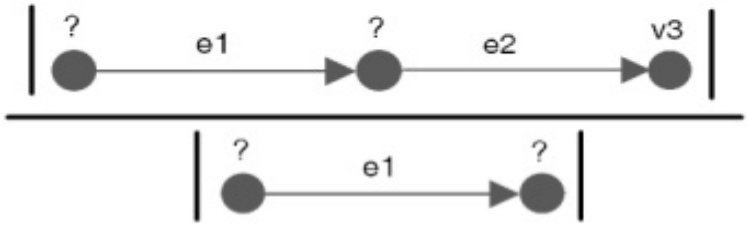
←What?
←What?

Reading YAGO

```
var yago = readRdfDf(sc, "/home/ec2-user/yago/yagoFacts.tsv")

def mergeIntoYago(filename:String):Unit = {
  val r = readRdfDf(sc, filename)
  r.vertices.cache
  r.edges.cache
  val g = mergeGraphs(yago, r)
  val g2 = GraphFrame(g.vertices, g.edges.distinct)
  g2.vertices.cache
  g2.vertices.count
  g2.edges.cache
  g2.edges.count
  yago.vertices.unpersist
  yago.edges.unpersist
  r.vertices.unpersist
  r.edges.unpersist
  yago = g2
}

mergeIntoYago("/home/ec2-user/yago/yagoLabels.tsv")
mergeIntoYago("/home/ec2-user/yago/yagoTaxonomy.tsv")
```

Query

```
yago.find("()-[e1]->(v2); (v2)-[e2]->(v3)")
    .distinct
    .selectExpr("e1.attr AS e1attr", "e2.attr AS e2attr",
                "v3.id AS v3id", "v3.attr AS v3attr")
    .groupBy("e1attr", "e2attr", "v3id", "v3attr")
    .count
    .join(yago.edges.groupBy("attr")
                    .count
                    .selectExpr("attr", "count AS e1count"),
          $"e1attr" === $"attr")
    .selectExpr("e1attr", "e2attr", "v3attr", "count/e1count AS ratio")
    .orderBy($"ratio".desc)
    .show(40)
```



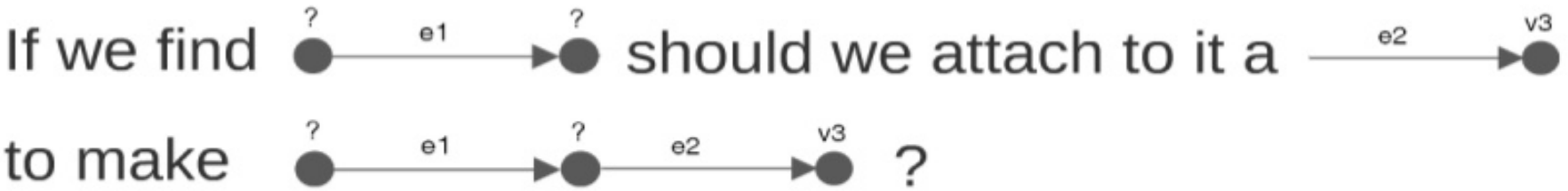
Results

e1attr	e2attr	v3attr	ratio
<hasAcademicAdvisor>	<hasGender>	<male>	0.9746811345897582
<influences>	<hasGender>	<male>	0.8912262249101587
<hasGender>	skos:prefLabel	"male"@eng	0.8428451872279868
<hasGender>	rdfs:label	"male"@eng	0.8428451872279868
<isCitizenOf>	<participatedIn>	<War_on_Terror>	0.835011170256778
<dealsWith>	<participatedIn>	<War_on_Terror>	0.772226304188097
<isCitizenOf>	<participatedIn>	<Operation_Enduri...	0.7614529608075682
<isCitizenOf>	<participatedIn>	<War_in_Afghanist...	0.757205505143834 3
<isCitizenOf>	<participatedIn>	<Operation_Enduri...	0.7488484982210332
<isCitizenOf>	<dealsWith>	<China>	0.742863447058499
<isCitizenOf>	<participatedIn>	<Korean_War>	0.7089389635104946
<isCitizenOf>	<participatedIn>	<Unified_Task_Force>	0.6842540750751579
<dealsWith>	<participatedIn>	<Operation_Enduri...	0.6796473181484203
<hasNeighbor>	<dealsWith>	<China>	0.6774193548387096

Discovered bug in YAGO 3.02 (not in YAGO 3.00 or forthcoming YAGO 3.03)

<dealsWith>	<dealsWith>	<United_States>	0.6223365172667157
<dealsWith>	<dealsWith>	<China>	0.6193975018368847
<dealsWith>	<participatedIn>	<Operation_Enduri...	0.6186627479794269
<isCitizenOf>	<participatedIn>	<Operation_Desert...	0.6018424028463469
<isCitizenOf>	<participatedIn>	<Operation_Active...	0.587031469785145
<isCitizenOf>	<participatedIn>	<Operation_Eagle'...	0.586865984499545
<isCitizenOf>	<participatedIn>	<Operation_Enduri...	0.5862316242380782
<isCitizenOf>	<participatedIn>	<Iraqi_no-fly_zones>	0.5859282345478115
<isCitizenOf>	<participatedIn>	<Colombian_confli...	0.5689935736547426
<isCitizenOf>	<participatedIn>	<Operation_Provid...	0.5657390297046088
<isCitizenOf>	<participatedIn>	<European_theatre...	0.5540447361888738
<isCitizenOf>	<participatedIn>	<Western_Front_(W...	0.5523071406900737
<hasOfficialLangu...	rdfs:label	"Joseph"@eng	0.5515832482124617 ← What?
<isCitizenOf>	<participatedIn>	<Helmand_province...	0.5459635380754061
<isCitizenOf>	<participatedIn>	<People's_General>	0.5350139283448714
<isCitizenOf>	<participatedIn>	<Congo_Crisis>	0.5346277960118047
<hasOfficialLangu...	rdfs:label	"Francis"@eng	0.5342185903983657 ← What?

3-vertex general linear pattern

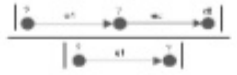


Reading YAGO

```
var yago = readNdfDf(sc, "/home/ec2-user/yago/yagoFacts.tsv")

def mergeIntoYago(filename:String):Unit = {
  val r = readNdfDf(sc, filename)
  r.vertices.cache
  r.edges.cache
  val g = mergeGraphs(yago, r)
  val g2 = GraphFrame(g.vertices, g.edges.distinct)
  g2.vertices.cache
  g2.vertices.count
  g2.edges.cache
  g2.edges.count
  yago.vertices.unpersist
  yago.edges.unpersist
  r.vertices.unpersist
  r.edges.unpersist
  yago = g2
}

mergeIntoYago("/home/ec2-user/yago/yagoLabels.tsv")
mergeIntoYago("/home/ec2-user/yago/yagoTaxonomy.tsv")
```

Query

```
yago.find("(v1) -[e1]-> (v2); (v2) -[e2]-> (v3)")
    .distinct
    .selectExpr("e1.attr AS e1attr", "e2.attr AS e2attr",
                "v2.id AS v2id", "v3.attr AS v3attr")
    .groupBy("e1attr", "e2attr", "v2id", "v3attr")
    .count
    .join(yago.edges.groupBy("attr")
                    .count
                    .selectExpr("attr", "count AS ccount"),
          $"e1attr" === $"attr")
    .selectExpr("e1attr", "e2attr", "v3attr", "count/ccount AS ratio")
    .orderBy($"ratio".desc)
    .show(50)
```



Results

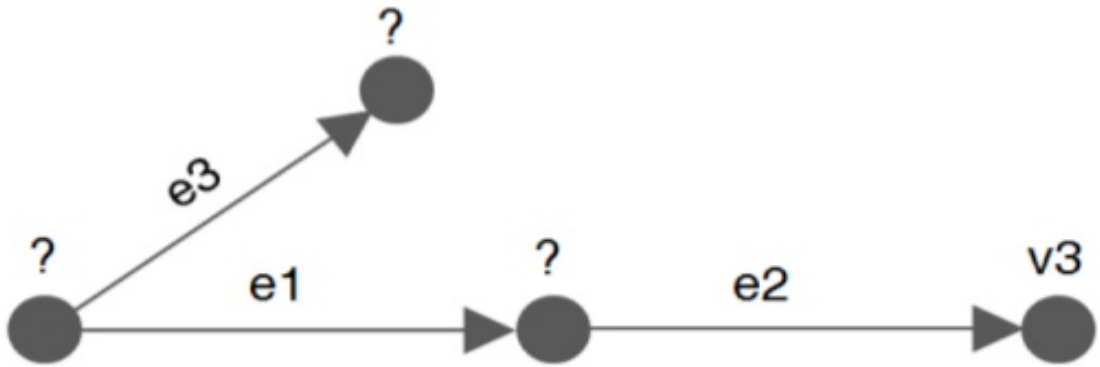
Discovered bug in YAGO 3.02 (not in YAGO 3.00 or forthcoming YAGO 3.03)

←What?
←What?

mergeIntoYago("/home/ec2-user/yago/yagoTaxonomy.tsv")

← What?

4-vertex general Y pattern




```
y1 = yago.find("(v1)-[e3]->(v4); (v1)-[e1]->(v2); (v2)-[e2]->(v3)")
     .distinct
     .selectExpr("e3.attr AS e3attr", "e1.attr AS e1attr", "e2.attr AS e2attr",
                 "v3.id AS v3id", "v3.attr AS v3attr")
     .groupBy("e3attr", "e1attr", "e2attr", "v3id", "v3attr")
     .count
     .cache
```

Out of Memory

AWS 5 nodes of r3.8xlarge, each 244GB RAM, 32 vCPU, 320GB SSD

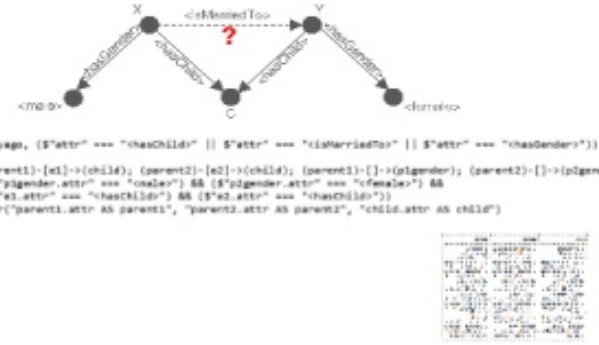
Out of Memory

AWS 5 nodes of r3.8xlarge, each 244GB RAM, 32 vCPU, 320GB SSD

3-vertex <hasChild> <isMarriedTo>

Initial Results

Refinement



Initial Results


```
var yago = readRdfDf(sc, "/yago3/yagoFacts.tsv")

def removeSingletons(g:GraphFrame) =
  GraphFrame(g.vertices.sqlContext.createDataFrame(
               g.triplets.select("src").map(_.getStruct(0))
                         .union(g.triplets.select("dst").map(_.getStruct(0)))
                         .distinct,
               g.vertices.schema),
             g.edges)

def filterEdges(g:GraphFrame, condition:Column) =
  removeSingletons(GraphFrame(g.vertices, g.edges.where(condition)))

val y2 = filterEdges(yago, $"attr" === "<hasChild>" || $"attr" === "<isMarriedTo>")

val df = y2.find("(parent1)-[e1]->(child); (parent2)-[e2]->(child); !(parent1)-[]->(parent2)")
           .filter(($"parent1" !== $"parent2") && ($"e1.attr" === "<hasChild>") &&
                   ($"e2.attr" === "<hasChild>"))
           .selectExpr("parent1.attr AS parent1", "parent2.attr AS parent2",
                       "child.attr AS child")
```

parent1	parent2	child
<Arcadius>	<Aelia_Flaccilla>	<Honorius_(emperor)>
<Victor_Emmanuel_...	<Victor_Emmanuel_...	<Maria_Pia_of_Savoy>
<George_VI_of_the...	<George_VI>	<Elizabeth_II>
<Otto_I,_Duke_of_...	<Otto_I_Wittelsba...	<Louis_I,_Duke_of...
<Jack_Nicholas_Pr...	<Rhoda_Pritzker>	<Nicholas_J._Prit...
<Augusta_Leigh>	<Lord_Byron>	<Elizabeth_Medora...
<Mehmed_the_Conqu...	<Mehmed_II>	<Bayezid_II>
<Jane_Stanhope,_C...	<Charles_Stanhope...	<Charles_Stanhope...
<Eleanor_Beaucham...	<Edmund_Beaufort,...	<Edmund_Beaufort,...
<Empress_Song_Fujin>	<Li_Bian>	<Li_Jing_(Souther...
<Guy_XIV_de_Laval>	<Isabella_of_Brit...	<Jeanne_de_Laval>
<Ronald_Reagan>	<Dorothy_Walker_B...	<George_H._W._Bush>
<Benedict_Swingat...	<Elizabeth_Calvert>	<Eleanor_Calvert>
<Hoapili>	<Kekuanaoa>	<Kamehameha_V>
<Leopold_II_of_Be...	<Prince_Philippe,...	<Albert_I_of_Belg...
<Henry_the_Fowler>	<Henry_I_the_Fowler>	<Gerberga_of_Saxony>
<Lý_Hu?_Tông>	<Tr?n_Th?_Dung>	<Lý_Chiêu_Hoàng>
<Argia_(mythology)>	<Aristodemus>	<Procles>
<William_I_of_Eng...	<William_the_Conq...	<Robert_Curthose>
<Emperor_Mommu>	<Emperor_Monmu>	<Emperor_Shomu>

Refinement


```
val y2 = filterEdges(yago, ($"attr" === "<hasChild>" || $"attr" === "<isMarriedTo>" || $"attr" === "<hasGender>"))

val df = y2.find("(parent1)-[e1]->(child); (parent2)-[e2]->(child); (parent1)-[]->(p1gender); (parent2)-[]->(p2gender)")
           .filter(($"p1gender.attr" === "<male>") && ($"p2gender.attr" === "<female>") &&
                   ($"e1.attr" === "<hasChild>") && ($"e2.attr" === "<hasChild>"))
           .selectExpr("parent1.attr AS parent1", "parent2.attr AS parent2", "child.attr AS child")
```

parent1	parent2	child
<Pirodjom_I>	<Justhathor-Honut...>	<Psusennes_I>
<Pireajem_I>	<Justhathor-Henut...>	<Menkheperre>
<Andromache_T>	<Olivia_Cadiz_F...>	<Hannad_VT>
<Charles_III_of_N...>	<Eleanor_of_Casti...>	<Blanche_I_of_Nav...>
<John_de_Foix,_Is...>	<Margaret_de_le_P...>	<Gaston_de_Foix,_...>
<Prince_Tomohito_...>	<Princess_Tomohit...>	<Princess_Akiko_o...>
<Prince_Tomohito_...>	<Princess_Tomohit...>	<Princess_Yoko_of...>
<Ozzy_Osbourne>	<Sharon_Osbourne>	<Jack_Osbourne>
<Ozzy_Osbourne>	<Sharon_Osbourne>	<Kelly_Osbourne>
<Robert_Sidney,_1...>	<Barbara_Sidney,_...>	<Robert_Sidney,_2...>
<Bruce_Dickinson>	<Dorma_Dunnery>	<Breeklyn_Dickinson>
<Philip_V_of_France>	<Joan_II,_Countes...>	<Blanche_of_Fran...>
<Philip_V_of_France>	<Joan_II,_Countes...>	<Isabella_of_Fran...>
<Philip_V_of_France>	<Joan_II,_Countes...>	<Margaret_I,_Cour...>
<Edmund_I>	<Elfgifu_of_Shaft...>	<Eadwig>
<Edmund_I>	<Elfgifu_of_Shaft...>	<Edgar_the_Peaceful>
<Ludovico_III_Gon...>	<Barbara_of_Brand...>	<Federico_I_Gonza...>
<Thomas_Hardenson,...>	<Mary_Perkins>	<Bailey_Hardenson>
<Caroline,_Margrav...>	<Susanna_of_Bavaria>	<Albert_Alexander...>
<Edward_the_Elder>	<Ælfflæd,_wife_of...>	<Edgyth>

parent1	parent2	child
<Pinedjem_I>	<Duathathor-Henut...	<Psusennes_I>
<Pinedjem_I>	<Duathathor-Henut...	<Menkheperre>
<Abdülmecid_I>	<Gülüstü_Kadin_Ef...	<Mehmed_VI>
<Charles_III_of_N...	<Eleanor_of_Casti...	<Blanche_I_of_Nav...
<John_de_Foix,_1s...	<Margaret_de_la_P...	<Gaston_de_Foix,_...
<Prince_Tomohito_...	<Princess_Tomohit...	<Princess_Akiko_o...
<Prince_Tomohito_...	<Princess_Tomohit...	<Princess_Yoko_of...
<Ozzy_Osbourne>	<Sharon_Osbourne>	<Jack_Osbourne>
<Ozzy_Osbourne>	<Sharon_Osbourne>	<Kelly_Osbourne>
<Robert_Sidney,_1...	<Barbara_Sidney,_...	<Robert_Sidney,_2...
<Bruce_Sudano>	<Donna_Summer>	<Brooklyn_Sudano>
<Philip_V_of_France>	<Joan_II,_Countes...	<Blanche_of_Franc...
<Philip_V_of_France>	<Joan_II,_Countes...	<Isabella_of_Fran...
<Philip_V_of_France>	<Joan_II,_Countes...	<Margaret_I,_Coun...
<Edmund_I>	<Ælfgifu_of_Shaft...	<Eadwig>
<Edmund_I>	<Ælfgifu_of_Shaft...	<Edgar_the_Peaceful>
<Ludovico_III_Gon...	<Barbara_of_Brand...	<Federico_I_Gonza...
<Thomas_Hardeman,...	<Mary_Perkins>	<Bailey_Hardeman>
<Casimir,_Margrav...	<Susanna_of_Bavaria>	<Albert_Alcibiade...
<Edward_the_Elder>	<Ælfflæd,_wife_of...	<Eadgyth>

Refinement


```
val y2 = filterEdges(yago, ($"attr" === "<hasChild>" || $"attr" === "<isMarriedTo>" || $"attr" === "<hasGender>"))

val df = y2.find("(parent1)-[e1]->(child); (parent2)-[e2]->(child); (parent1)-[]->(p1gender); (parent2)-[]->(p2gender)")
           .filter(($"p1gender.attr" === "<male>") && ($"p2gender.attr" === "<female>") &&
                   ($"e1.attr" === "<hasChild>") && ($"e2.attr" === "<hasChild>"))
           .selectExpr("parent1.attr AS parent1", "parent2.attr AS parent2", "child.attr AS child")
```

parent1	parent2	child
<Pirodjom_I>	<Justhathor-Honut...>	<Psusennes_I>
<Pireajem_I>	<Justhathor-Henut...>	<Menkheperre>
<Andromache_T>	<Olivia_Cadiz_F...>	<Hannad_VT>
<Charles_III_of_N...>	<Eleanor_of_Casti...>	<Blanche_I_of_Nav...>
<John_de_Foix,_Is...>	<Margaret_de_le_P...>	<Gaston_de_Foix,_...>
<Prince_Tomohito_...>	<Princess_Tomohit...>	<Princess_Akiko_o...>
<Prince_Tomohito_...>	<Princess_Tomohit...>	<Princess_Yoko_of...>
<Ozzy_Osbourne>	<Sharon_Osbourne>	<Jack_Osbourne>
<Ozzy_Osbourne>	<Sharon_Osbourne>	<Kelly_Osbourne>
<Robert_Sidney,_1...>	<Barbara_Sidney,_...>	<Robert_Sidney,_2...>
<Bruce_Dickinson>	<Dorma_Dunnery>	<Breeklyn_Dickinson>
<Philip_V_of_France>	<Joan_II,_Countes...>	<Blanche_of_Fran...>
<Philip_V_of_France>	<Joan_II,_Countes...>	<Isabella_of_Fran...>
<Philip_V_of_France>	<Joan_II,_Countes...>	<Margaret_I,_Cour...>
<Edmund_I>	<Elfgifu_of_Shaft...>	<Eadwig>
<Edmund_I>	<Elfgifu_of_Shaft...>	<Edgar_the_Peaceful>
<Ludovico_III_Gon...>	<Barbara_of_Brand...>	<Federico_I_Gonza...>
<Thomas_Hardenson,...>	<Mary_Perkins>	<Bailey_Hardenson>
<Caroline,_Margrav...>	<Susanna_of_Bavaria>	<Albert_Alexander...>
<Edward_the_Elder>	<Ælfflæd,_wife_of...>	<Edgyth>

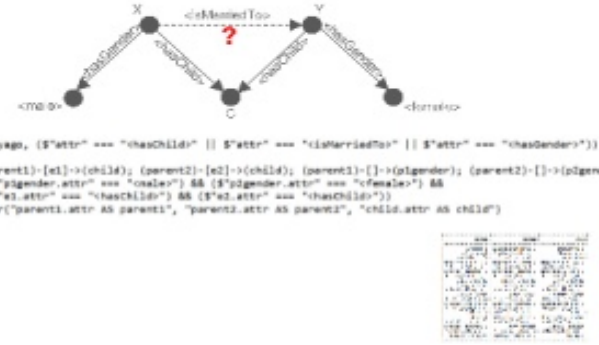
Out of Memory

AWS 5 nodes of r3.8xlarge, each 244GB RAM, 32 vCPU, 320GB SSD

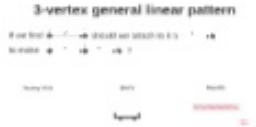
3-vertex <hasChild> <isMarriedTo>

Initial Results

Refinement



GraphFrames



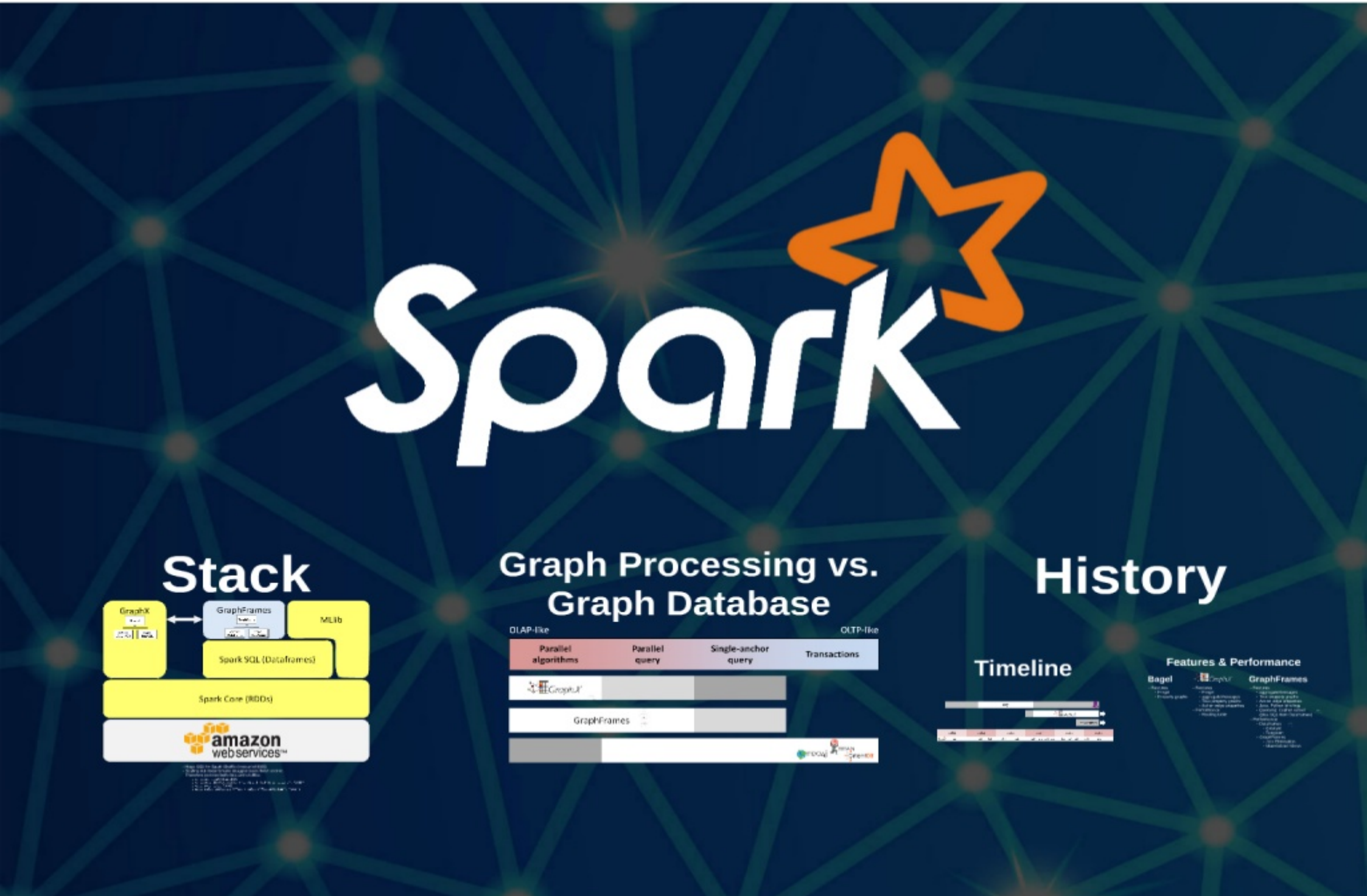
Graph Processing vs. Graph Database

OLAP-like

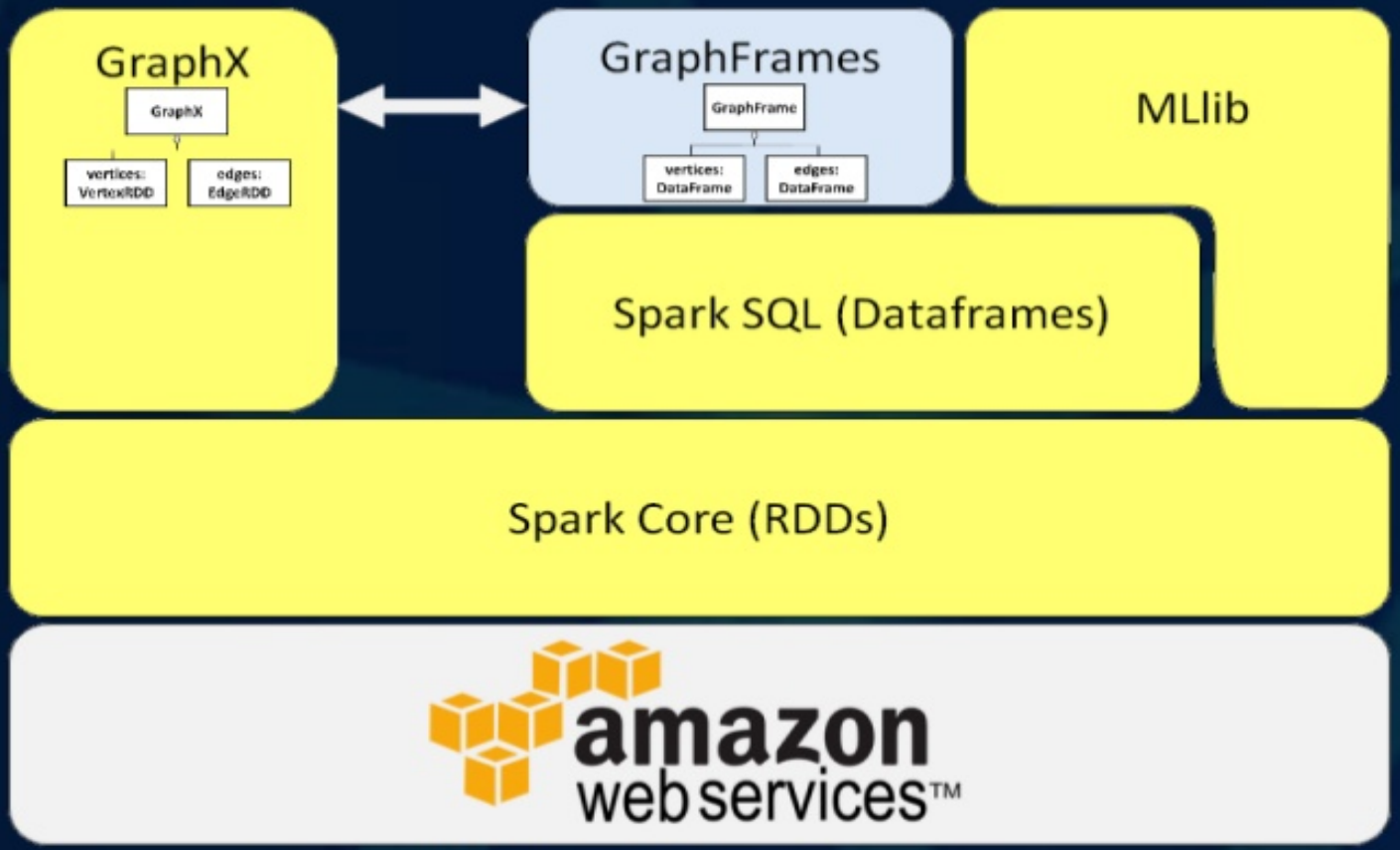
OLTP-like

Parallel algorithms	Parallel query	Single-anchor query	Transactions

GraphFrames


Spark
Stack
GraphX
GraphFrames
MLlib
Spark SQL (Dataframes)
Spark Core (RDDs)
amazon
webservices
Graph Processing vs.
Graph Database
OLAP-like
OLTP-like
Parallel
algorithms
Parallel
query
Single-anchor
query
Transactions
GraphFrames
History
Timeline
Features & Performance
Bagel
GraphFrames

Stack



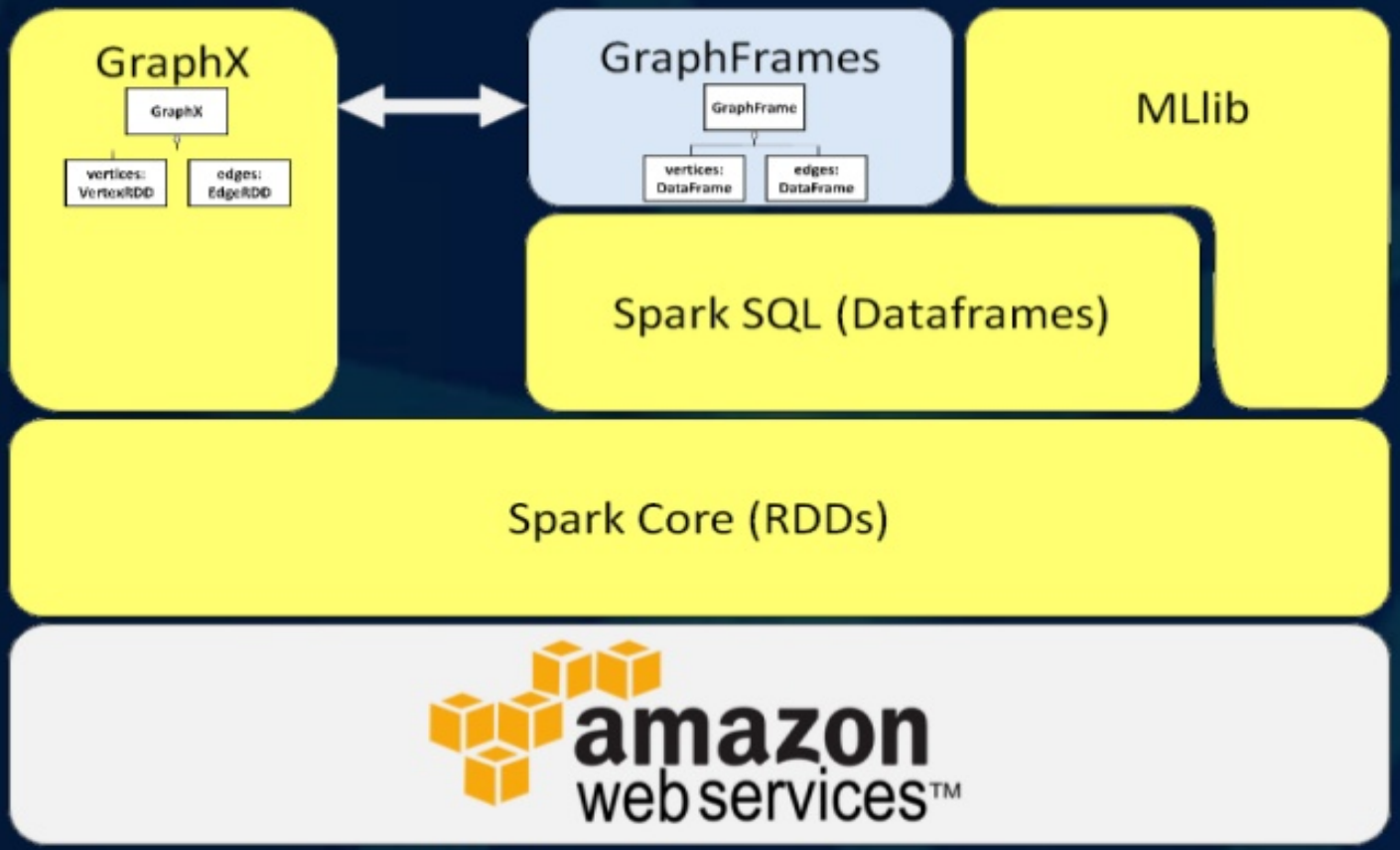
- Huge SSD for Spark Shuffle (instead of EBS)
- Scaling out doesn't make straggler tasks finish sooner.
  Therefore partition both data and shuffles:

```
g.vertices.repartition(1000)
g.vertices.sqlContext.setConf("spark.sql.shuffle.partitions", "1000")
g.edges.repartition(1000)
```

- Huge SSD for Spark Shuffle (instead of EBS)
- Scaling out doesn't make straggler tasks finish sooner.
  Therefore partition both data and shuffles:

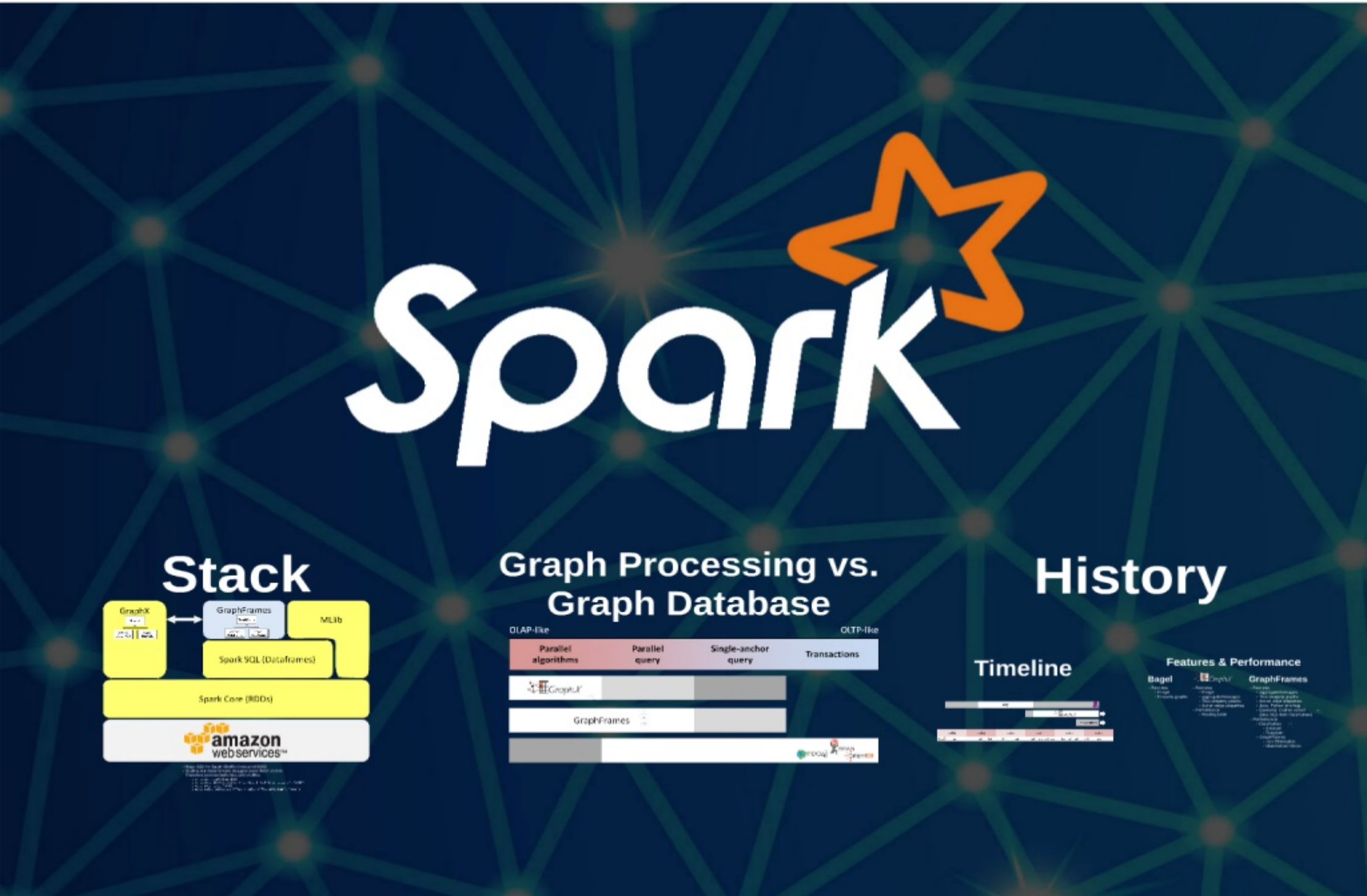
```
g.vertices.repartition(1000)
g.vertices.sqlContext.setConf("spark.sql.shuffle.partitions", "1000")
g.edges.repartition(1000)
g.edges.sqlContext.setConf("spark.sql.shuffle.partitions", "1000")
```

Stack



- Huge SSD for Spark Shuffle (instead of EBS)
- Scaling out doesn't make straggler tasks finish sooner.
  Therefore partition both data and shuffles:

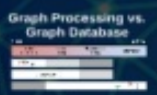
```
g.vertices.repartition(1000)
g.vertices.sqlContext.setConf("spark.sql.shuffle.partitions", "1000")
g.edges.repartition(1000)
```


Spark
Stack
GraphX
GraphFrames
MLlib
Spark SQL (Dataframes)
Spark Core (RDDs)
amazon
webservices
Graph Processing vs.
Graph Database
OLAP-like
OLTP-like
Parallel
algorithms
Parallel
query
Single-anchor
query
Transactions
GraphFrames
History
Timeline
Features & Performance
Bagel
GraphFrames


Finding Graph Isomorphisms in GraphX and GraphFrames
Michael Malak
ORACLE


Spark


Stack


Graph Processing vs. Graph Database


History
Timeline


Isomorphisms
Definition
Detection
Finding


yago
select knowledge
Yet Another Great Ontology (YAGO)
How it's Distributed


Summary

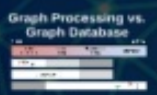
Summary

- Mini-isomorphisms (aka rule mining) good for:
  - Data quality/data cleansing
  - Recommendations
    - Opportunities
    - Fraud alerts
    - Product recommendations
- SVD++ in GraphX useful for tiniest of rules
- Cypher subset in GraphFrames for larger patterns
  - For larger patterns, better to pre-filter


Finding Graph Isomorphisms in GraphX and GraphFrames
Michael Malak
ORACLE


Spark


Stack


Graph Processing vs. Graph Database

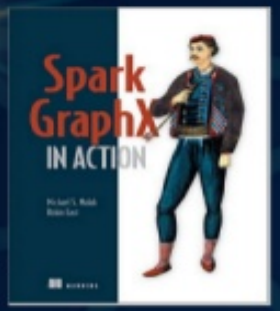

History
Timeline


Isomorphisms
Definition
Detection
Finding


yago
select knowledge
Yet Another Great Ontology (YAGO)
How it's Distributed


Summary

Finding Graph Isomorphisms in GraphX and GraphFrames



Michael Malak

Source code:
https://www.manning.com/books/spark-graphx-in-action
https://gist.github.com/michaelmalak

SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO