

DECOM / UFOP
BCC362 - SISTEMAS DISTRIBUÍDOS
PROF. JOUBERT DE CASTRO LIMA

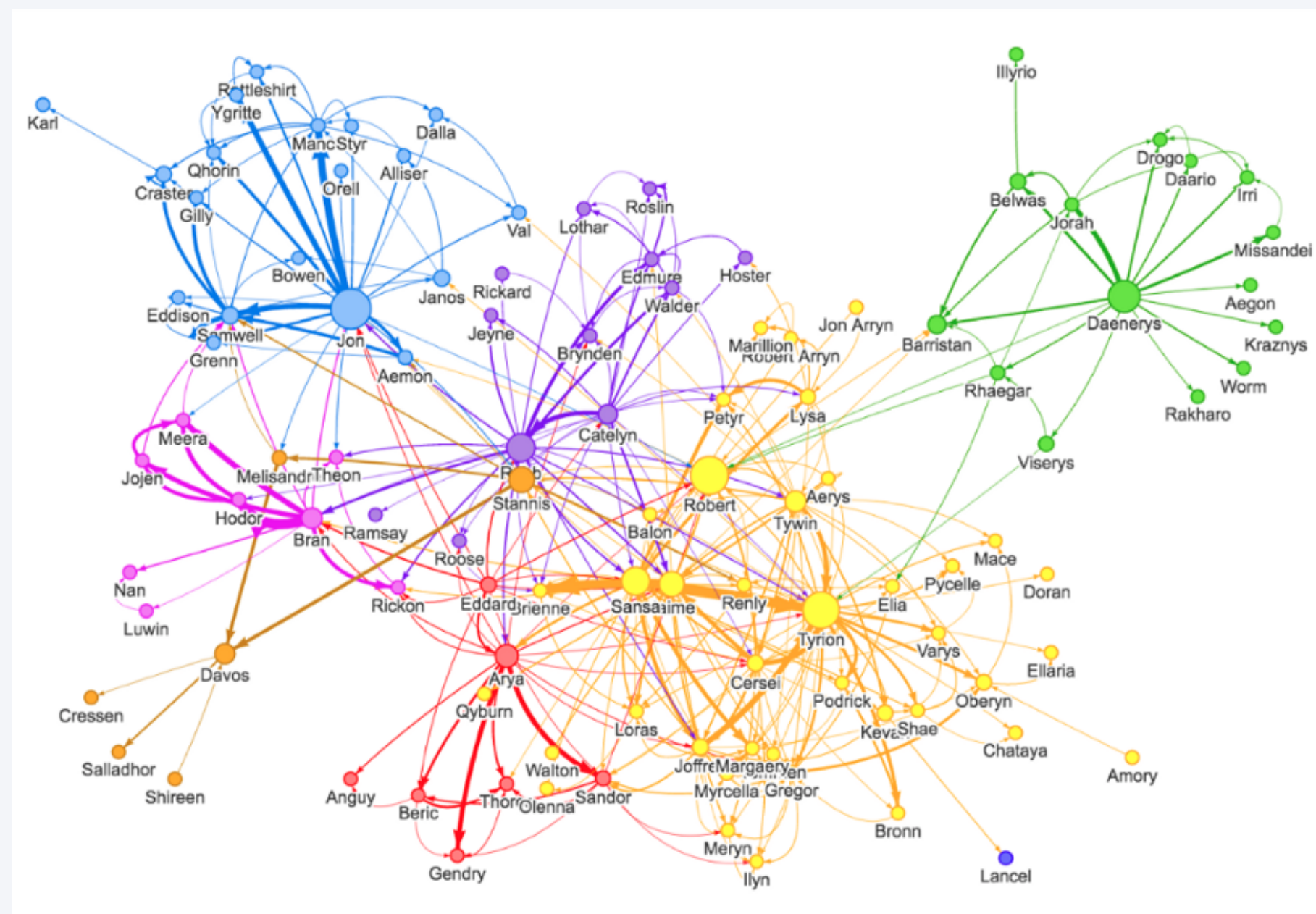
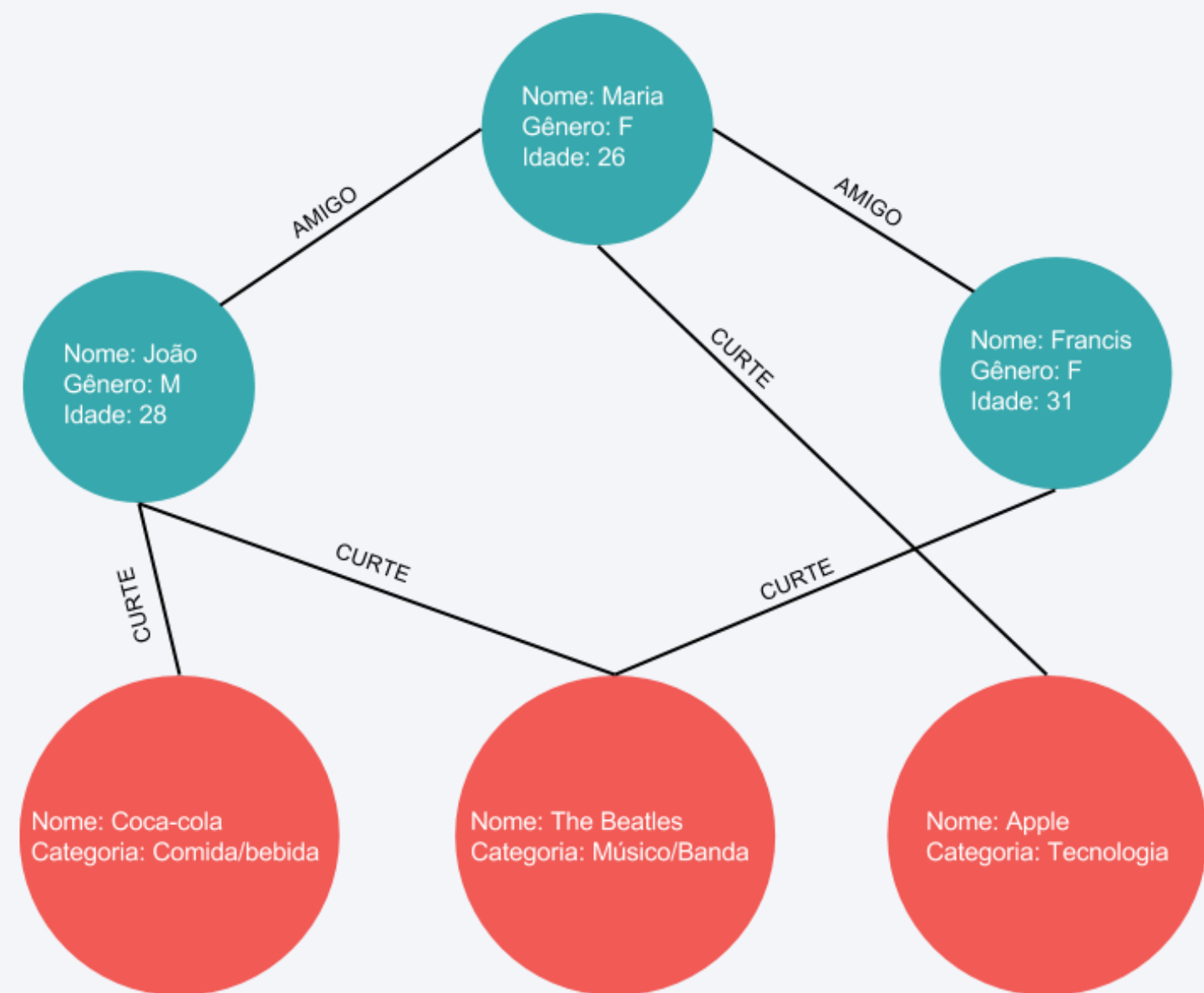
Apache Spark GraphX

Gustavo Presoti Sales Brito
Vinícius Samy Santana Souza

SPARK

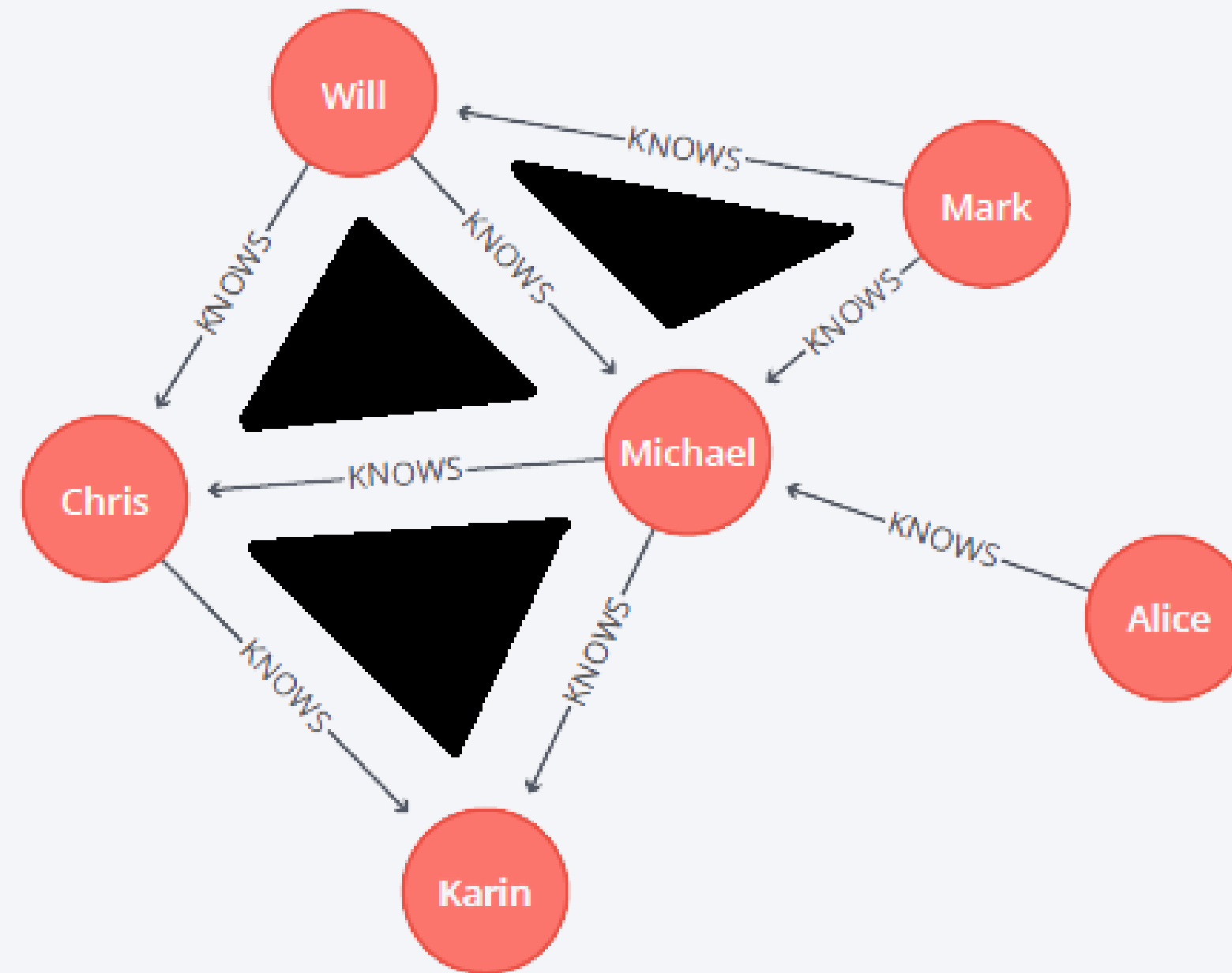
- Framework para processamento de Big Data
- Projeto da fundação Apache
- Unificado e de fácil compreensão
- Suporte a Java, Scala e Python

GRAFOS



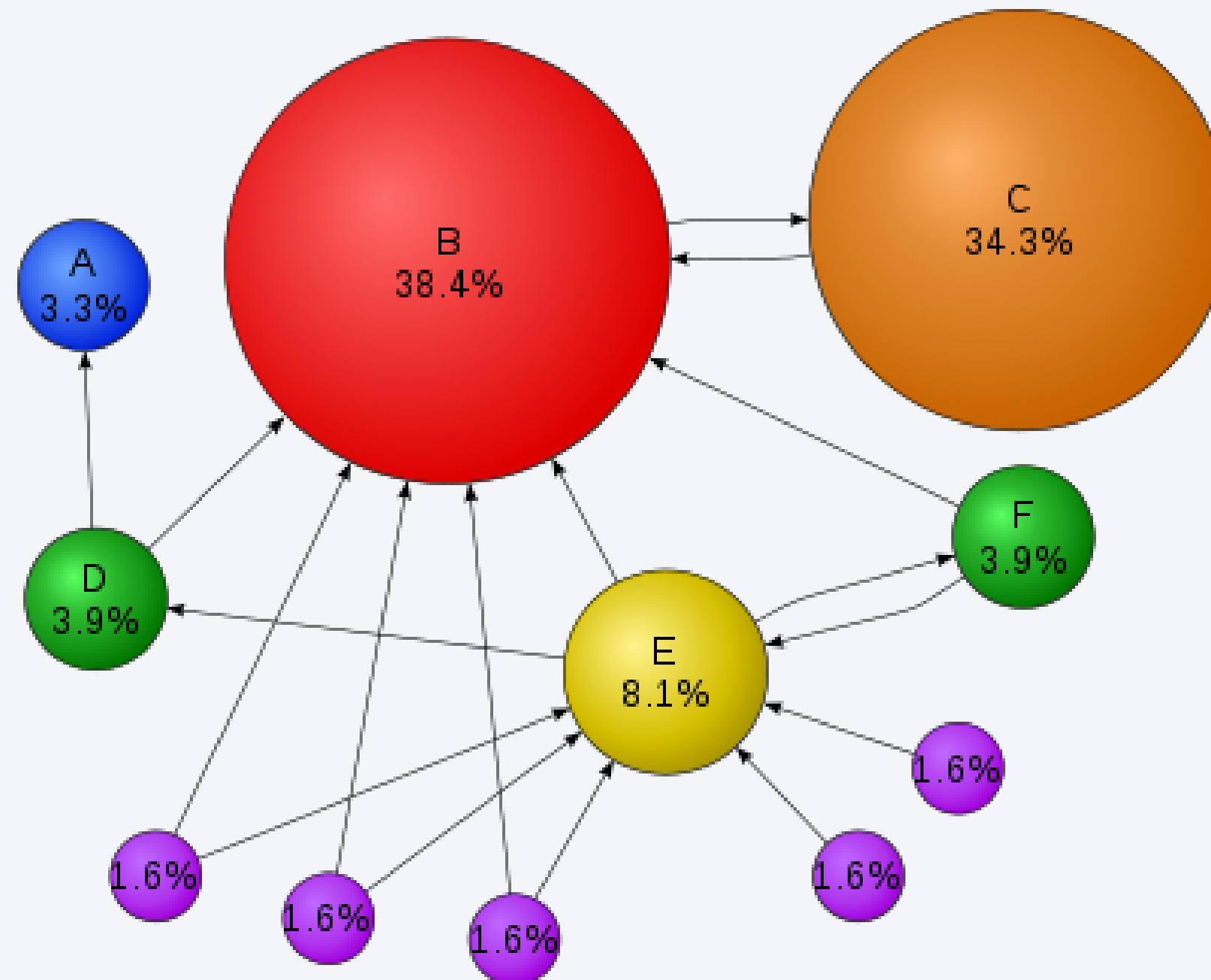
ALGORITMOS EM GRAFOS

Contagem de Triângulos



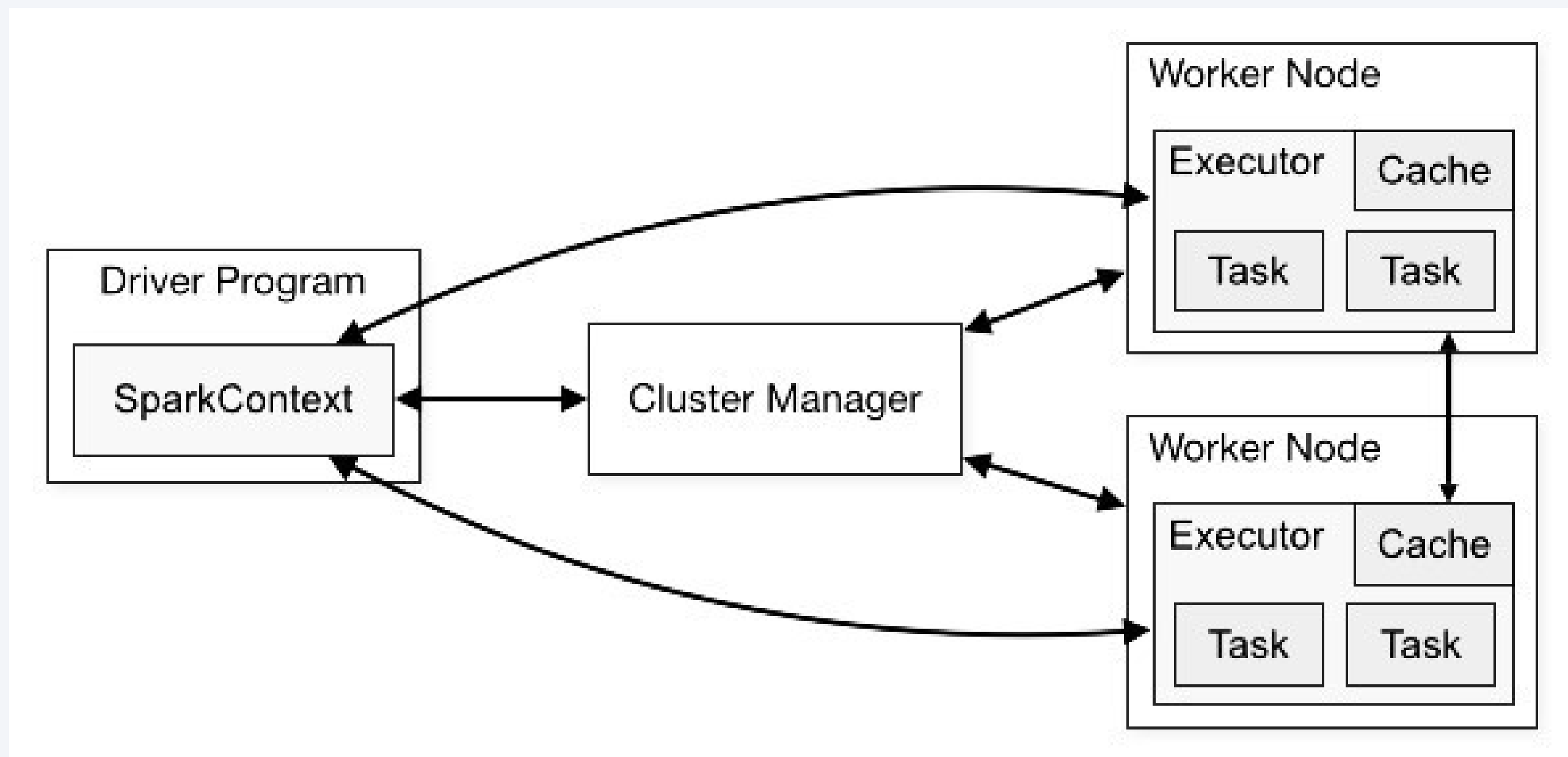
ALGORITMOS EM GRAFOS

Page Rank



SPARK

Arquitetura

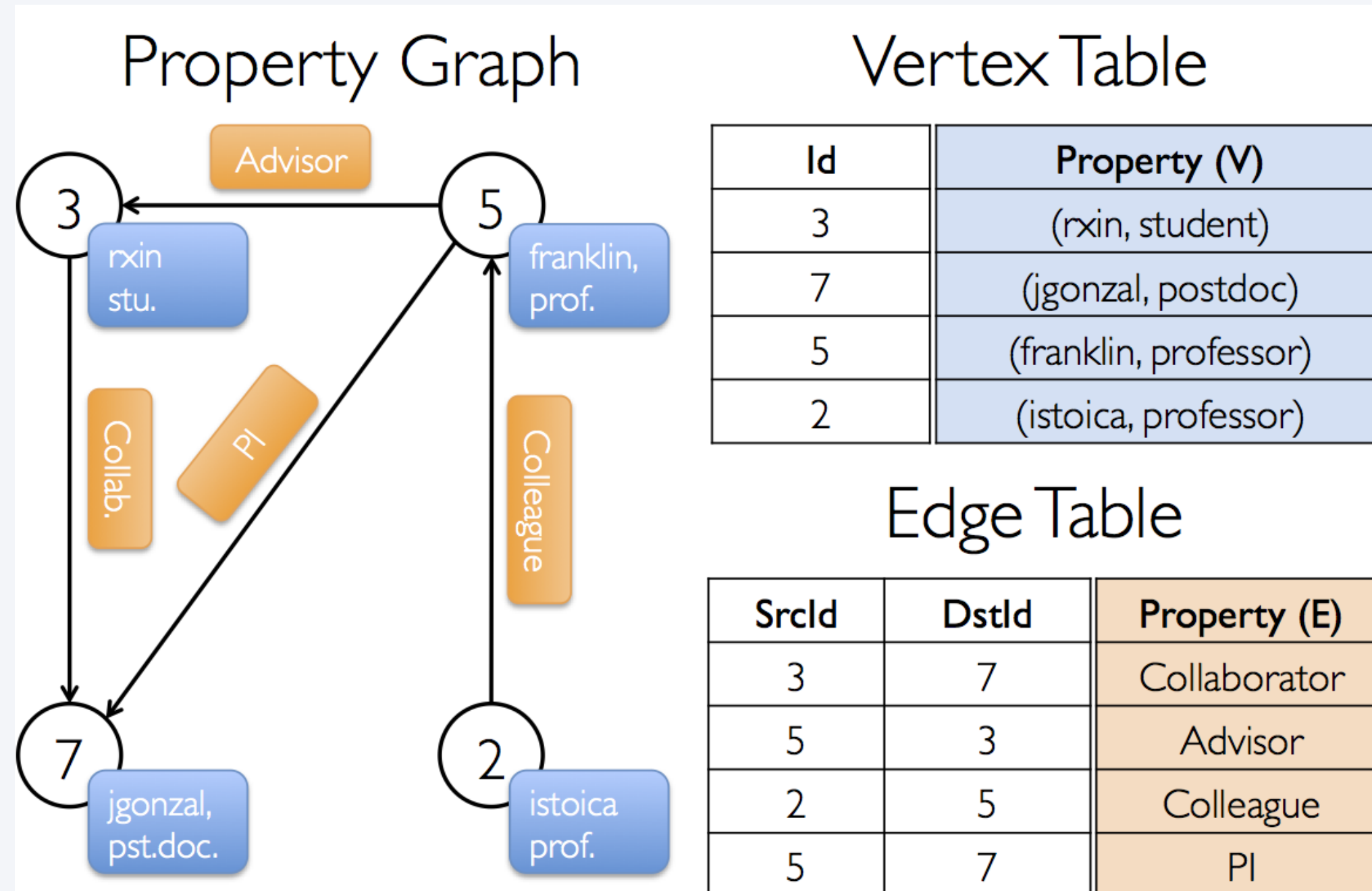


SPARK GRAPHX

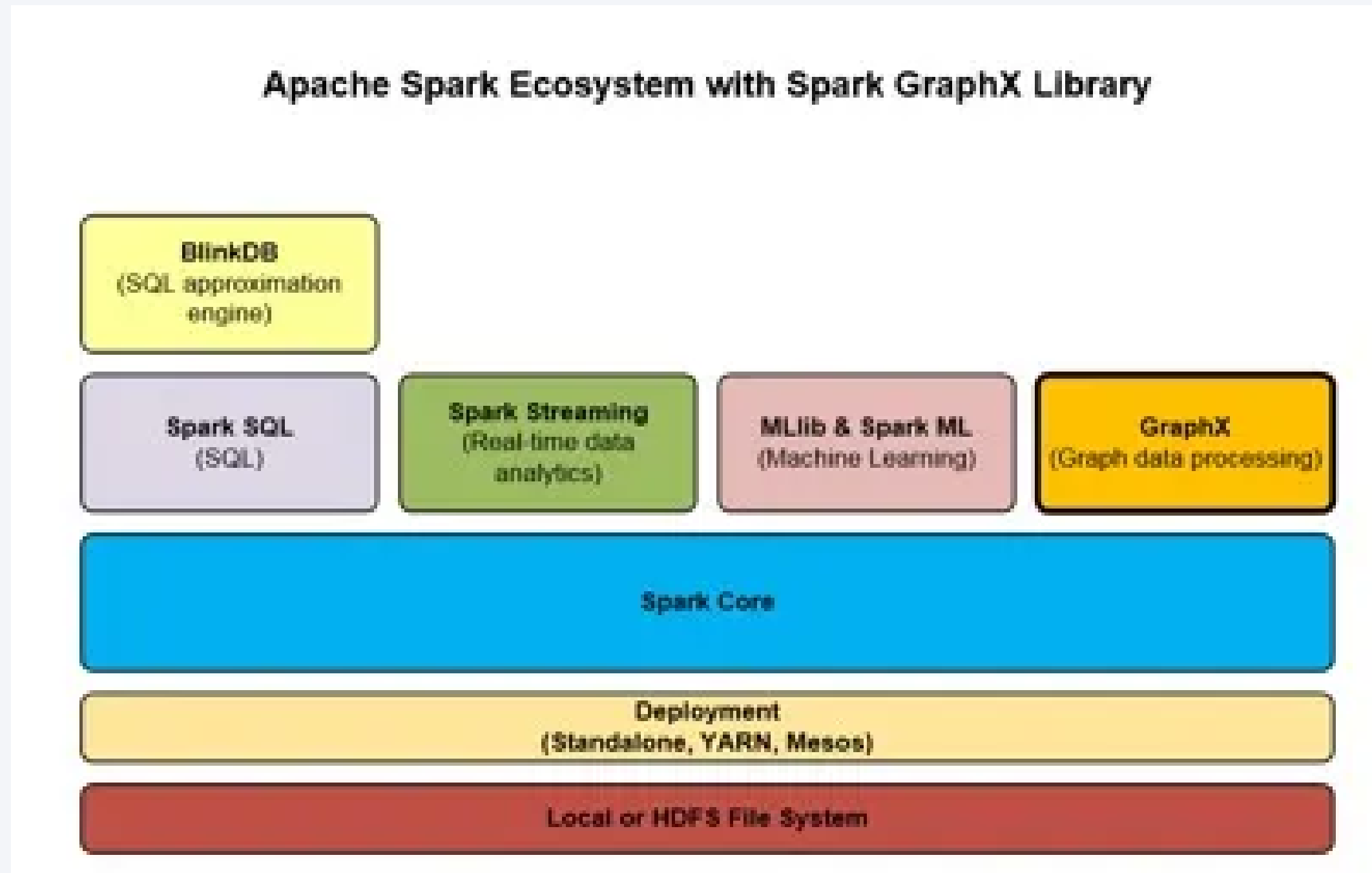
- Computação paralela de grafos
- Multigrafo direcionado
- Operadores e algoritmos integrados
- Facilidade de análise

SPARK GRAPHX

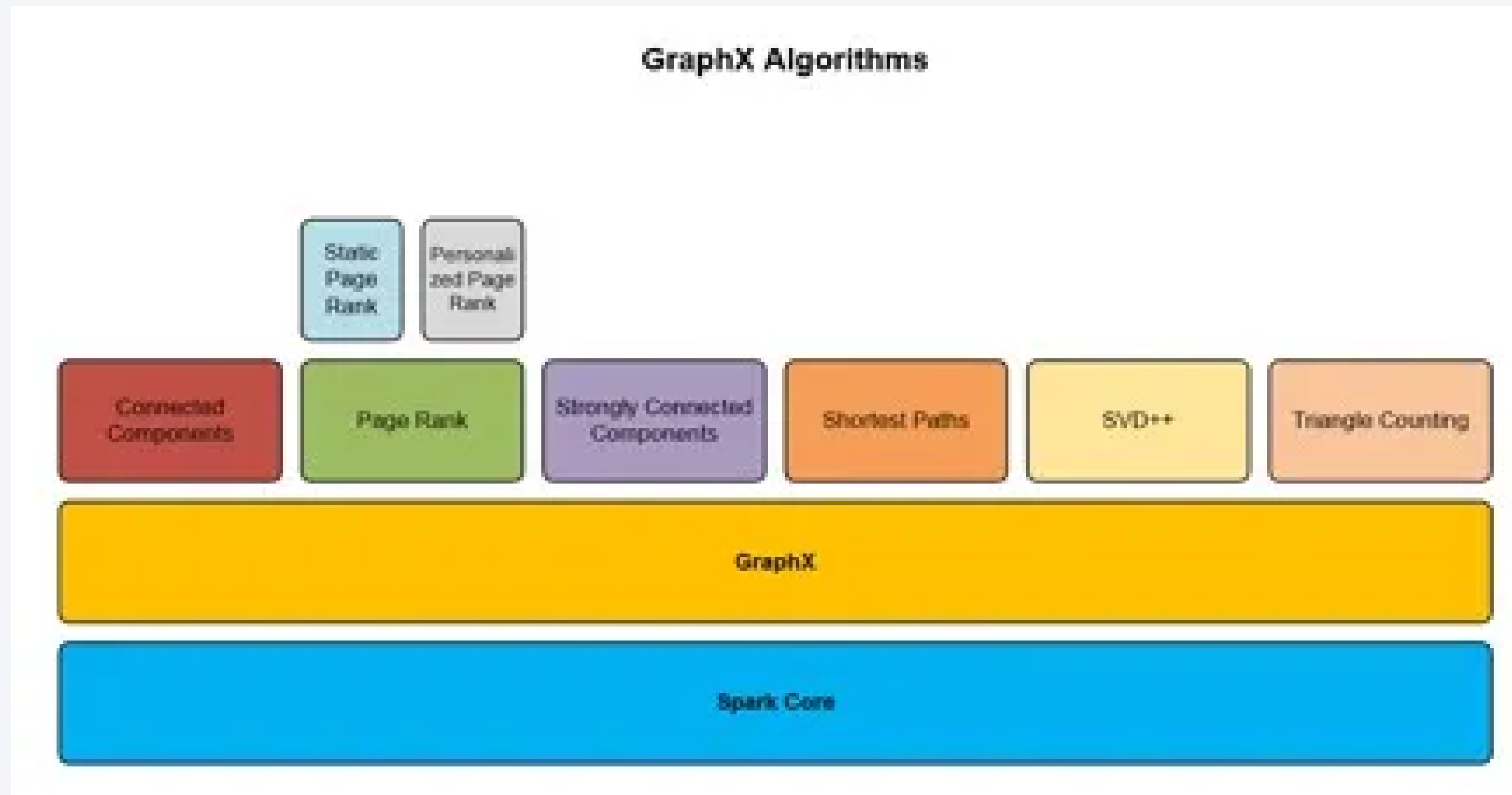
Grafo de propiedades



ECOSSISTEMA SPARK



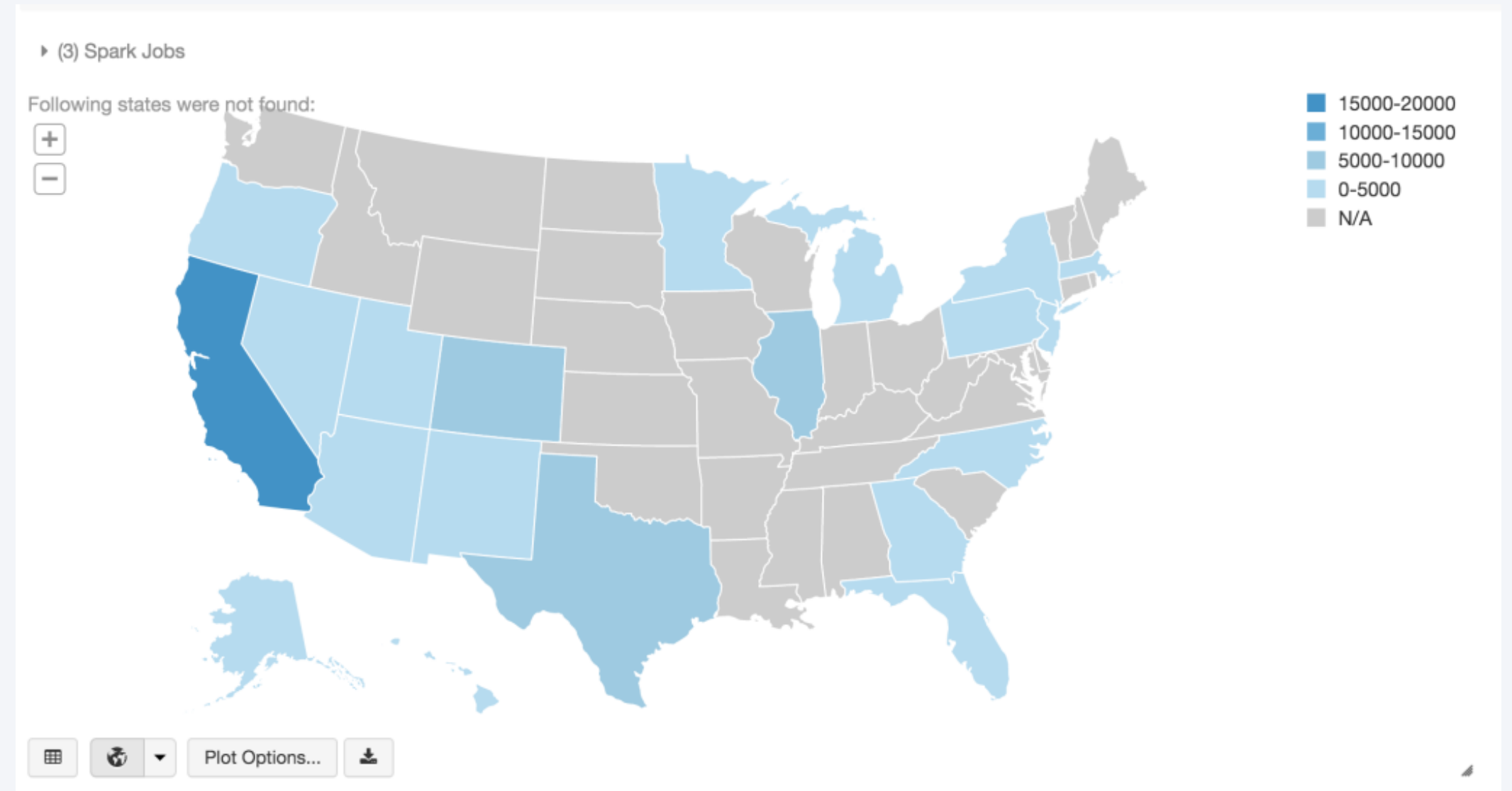
SPARK GRAPHX



SPARK GRAPHX

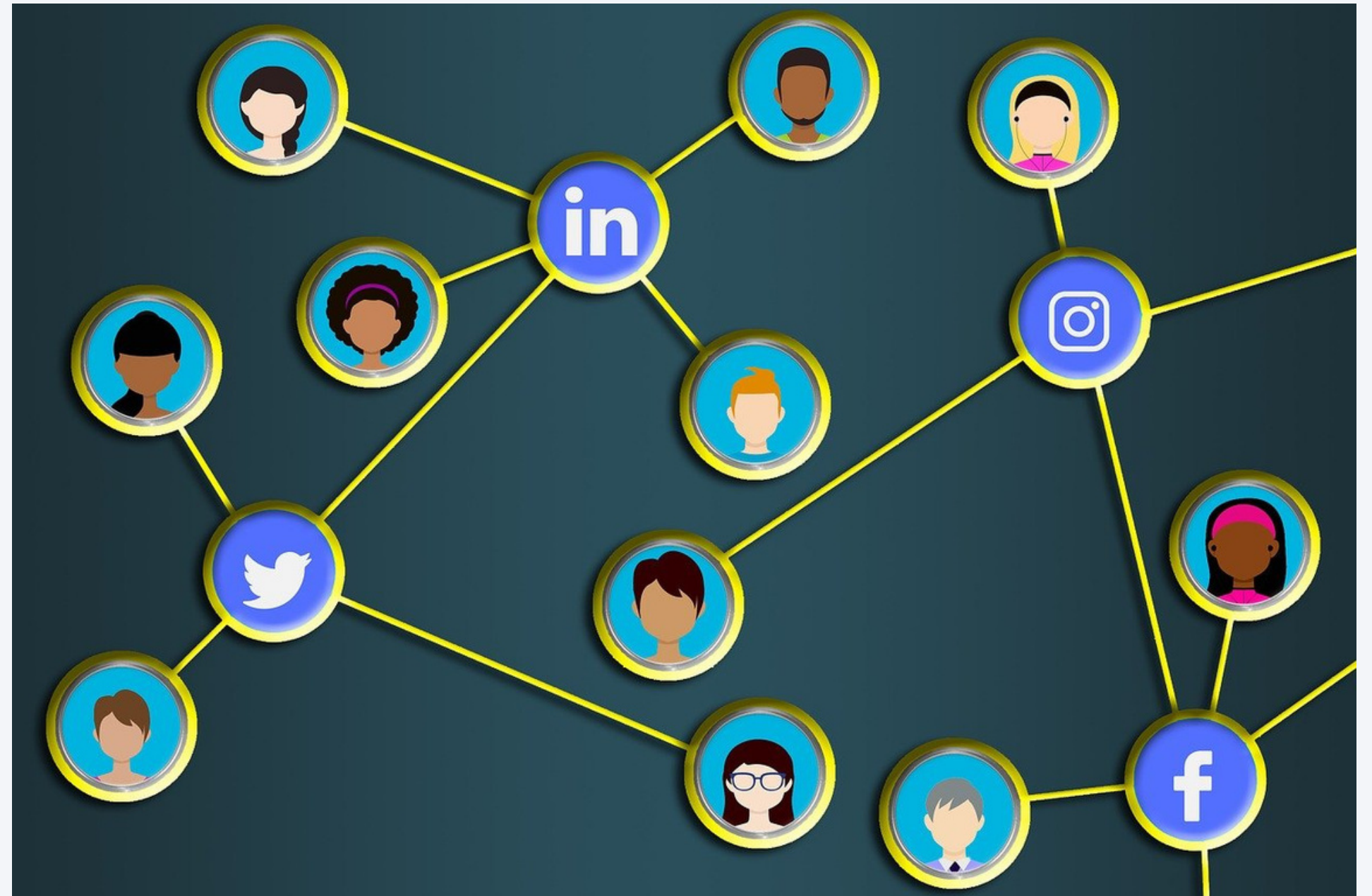
Casos de uso

- Detecção de fraude
- Desempenho de voo
- Distância mais curta



APLICACAO

- Redes sociais como grafos
- Instância gerada
 - Vértices: 7.000
 - Arestas: 19.597.201
- Pagerank
 - Encontrar os perfis mais influentes



APLICAÇÃO

Profiles.csv

id	name
0	Melanie
1	Timothy
2	Michael
3	Idella
4	LueLLa
5	Stephen
6	Ronda
7	Jerome

Connections.csv

profile1	profile2
0	3726
0	5807
0	4191
0	2364
0	1705
0	3181
0	5134
0	4247

ROTINA

```
//Importando Libs
import org.apache.spark._
import org.apache.spark.graphx._
import org.apache.spark.rdd.RDD
import org.apache.spark.graphx.GraphLoader
```

ROTINA

```
//Lendo arestas a partir de um csv ( csv => String )  
val edges_1 = spark.sparkContext.textFile("gs://dataproc-bucket-1/spark-example/connections.csv")  
//Separando as colunas pelo delimitador ',' ( String => Array[String] )  
val edges_2 = edges_1.map(f=>f.split(','))  
//Descartando Header ( Array[String] => Array[String] )  
val edges_3 = edges_2.mapPartitionsWithIndex{ (idx, itr) => if (idx == 0) itr.drop(1) else itr }  
//Representando arestas como Edge() e armazenando em um RDD (Array[String] => RDD[Edge[Long]] )  
val edges : RDD[Edge[Long]] = edges_3.map( arr=> Edge(arr(0).toLong, arr(1).toLong, 0L) )
```

ROTINA

```
//Mesmo processo para os vertices
//... ( csv => String )
val vertices_1 = spark.sparkContext.textFile("gs://dataproc-bucket-1/spark-example/profiles.csv")
//... ( String => Array[String] )
val vertices_2 = vertices_1.map(f=>f.split(','))
//... ( Array[String] => Array[String] )
val vertices_3 = vertices_2.mapPartitionsWithIndex{ (idx, itr) => if (idx ==0) itr.drop(1) else itr }
//... (Array[String] => RDD[(VertexId, String)] )
val vertices: RDD[(VertexId, String)] = vertices_3.map( arr=> (arr(0).toLong, arr(1)) )
```


ROTINA

```
//Criando grafo
val graph = Graph(vertices, edges)

//Realizando pageRank
val ranks = graph.pageRank(0.0001).vertices
//Associando Ranks aos nomes de perfil
val ranksByName = vertices.join(ranks).map({ case (id, (username, rank)) => (rank, username) })
//Printando top 10 mais bem rankeados
ranksByName.top(10).foreach(println(_))
```

FORMAS DE IMPLANTAÇÃO

- Dataproc
- Dataproc no Google Kubernetes Engine

DATAPROC

Nome

Nome do cluster *

cluster-fdf3



Local

Região *

us-central1



Zona *

us-central1-a



Tipo de cluster

☒ Padrão (1 mestre, N workers)

☐ Nó único (1 mestre, 0 worker)

Fornecer um nó que atua como mestre e worker. É bom para prova de conceito ou processamento em pequena escala

☐ Alta disponibilidade (3 mestres, N workers)

O modo de alta disponibilidade do Hadoop oferece operações YARN e HDFS ininterruptas, independentemente de qualquer falha ou reinicialização de nó único.

DATAPROC

Nó mestre ^

Contém o YARN Resource Manager, HDFS NameNode e todos os drivers do job.

Família de máquinas

PROPÓSITO GERAL OTIMIZADO PARA COMPUTAÇÃO OTIMIZADO PARA MEMÓRIA

Tipos de máquinas para cargas de trabalho comuns, otimizadas para custo e flexibilidade

Série

E2

Seleção de plataforma de CPU com base na disponibilidade

Tipo de máquina

e2-standard-2 (2 vCPU, 8 GB de memória)



vCPU
2

Memory
8 GB

✓ PLATAFORMA DE CPU E GPU

Tamanho do disco principal (...)

100 GB ?

Primary disk type

Standard Persistent Disk ?

Número de SSDs locais *

x 375GB ?

Nós de trabalho ^

Cada um contém um YARN NodeManager e um HDFS DataNode. O fator de replicação de HDFS é 2.

Família de máquinas

PROPÓSITO GERAL OTIMIZADO PARA COMPUTAÇÃO OTIMIZADO PARA MEMÓRIA

Tipos de máquinas para cargas de trabalho comuns, otimizadas para custo e flexibilidade

Série

E2

Seleção de plataforma de CPU com base na disponibilidade

Tipo de máquina

e2-standard-2 (2 vCPU, 8 GB de memória)



vCPU
2

Memory
8 GB

✓ PLATAFORMA DE CPU E GPU

Number of worker nodes

3 ?

Tamanho do disco principal (...)

50 GB ?

Primary disk type

Standard Persistent Disk ?

DATAPROC

Criação do Dataproc

Bucket de preparação do Cloud Storage

Bucket de preparação do Cloud Storage

 dataproc-bucket-1

BROWSE

Google Storage

←

Detalhes do bucket

↻ ATUALIZAR

🎓 SAIBA MAIS

dataproc-bucket-1

Local

us-central1 (Iowa)

Classe de armazenamento

Standard

Acesso público

Não público

Proteção

Nenhum

OBJETOS

CONFIGURAÇÃO

PERMISSÕES

PROTEÇÃO

CICLO DE VIDA

Intervalos > dataproc-bucket-1 > spark-example 📄

FAZER UPLOAD DE ARQUIVOS

CARREGAR PASTA

CRIAR PASTA

GERENCIAR RETENÇÕES

FAZER O DOWNLOAD

EXCLUIR

Filtrar apenas pelo prefixo do nome ▾

≡ Filtro

Filtrar objetos e pastas

🔴

Mostrar dados excluídos

⌵

<input type="checkbox"/>	Nome	Tamanho	Tipo	Criado ?	Classe de armazenamento	Última modificação	Acesso público ?	Histórico	
<input type="checkbox"/>	📁 .git/	—	Pasta	—	—	—	—	—	⋮
<input type="checkbox"/>	📄 connections.csv	199,7 MB	text/csv	13 de ou...	Standard	13 de out. de 20...	Não público	—	⬇ ⋮
<input type="checkbox"/>	📁 misc/	—	Pasta	—	—	—	—	—	⋮
<input type="checkbox"/>	📄 profiles.csv	86,4 KB	text/csv	13 de ou...	Standard	13 de out. de 20...	Não público	—	⬆ ⋮

DATAPROC NO GKE

1 - Criando cluster GKE

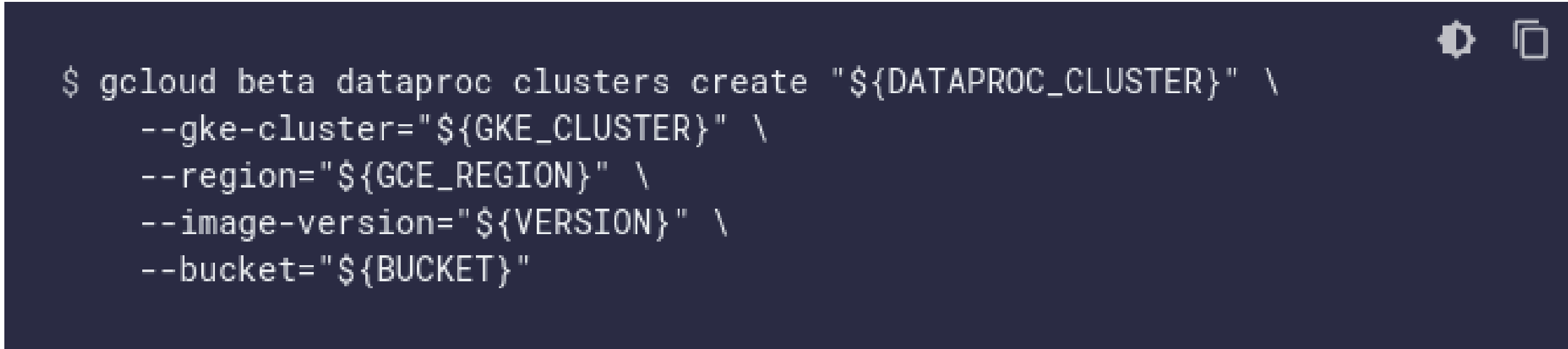
Um cluster do GKE em execução é necessário como plataforma de implantação para componentes do Dataproc.

```
$ gcloud beta container clusters create "${GKE_CLUSTER}" \
  --scopes=cloud-platform \
  --workload-metadata=GCE_METADATA \
  --machine-type=n1-standard-4 \
  --region="${GCE_REGION}"
```

DATAPROC NO GKE

2 - Criar um cluster do Dataproc-on-GKE

Esta etapa aloca um cluster do Dataproc em um cluster do GKE existente, implantando componentes que vinculam o cluster do GKE ao serviço do Dataproc para permitir o envio de jobs do Spark.



```
$ gcloud beta dataproc clusters create "${DATAPROC_CLUSTER}" \  
  --gke-cluster="${GKE_CLUSTER}" \  
  --region="${GCE_REGION}" \  
  --image-version="${VERSION}" \  
  --bucket="${BUCKET}"
```

DATAPROC NO GKE

3 - Enviar um job do Spark

```
$ gcloud dataproc jobs submit spark \
  --cluster="${DATAPROC_CLUSTER}" \
  --region="${GCE_REGION}" \
  --class=org.apache.spark.examples.SparkPi \
  --jars=file:///usr/lib/spark/examples/jars/spark-examples.jar
```


REFERÊNCIA BIBLIOGRÁFICA

- AMMAR, Khaled; OZSU, Tamer. Experimental analysis of distributed graph systems. arXiv preprint arXiv:1806.08082, 2018.
- XIN, Reynold S. et al. Graphx: Unifying data-parallel and graph-parallel analytics. arXiv preprint arXiv:1402.2394, 2014.
- <https://databricks.com/blog/2016/03/16/on-time-flight-performance-with-graphframes-for-apache-spark.html>
- <https://www.linkedin.com/pulse/conhecendo-o-mundo-dos-grafos-com-spark-graphx-gleber-teixeira-phd/?originalSubdomain=pt>

REFERÊNCIA BIBLIOGRÁFICA

- <http://spark.apache.org/docs/latest/graphx-programming-guide.html>
- <https://www.infoq.com/br/articles/apache-spark-introduction/>
- <https://www.infoq.com/br/articles/apache-spark-graphx/>
- <https://www.devmedia.com.br/introducao-ao-apache-spark/34178>
- <https://lorenadesouza.medium.com/bootcamp-de-dados-na-tw-spark-6633275480e4>

REFERÊNCIA BIBLIOGRÁFICA

- <https://laptrinhx.com/getting-started-with-apache-spark-graphx-part-1-3634131423/>
- <https://www.slideshare.net/JenAman/credit-fraud-prevention-with-spark-and-graph-analysis>
- <https://pixabay.com/pt/illustrations/rede-social-comunica%C3%A7%C3%A3o-internet-4095262/>
- <https://cloud.google.com/dataproc/docs/concepts/jobs/dataproc-gke>