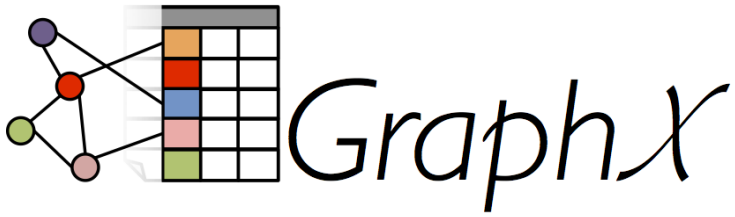


Apache Spark GraphX

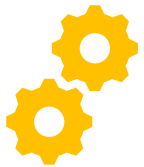
Bc. Jana Vlčková, Bc. Marek Kačmár,

Bc. Branislav Prokop, Bc. Adam Samko

Obsah

Apache Spark Framework

Úvod do Spark GraphX



Spark GraphX a load balancing

Demo aplikácie

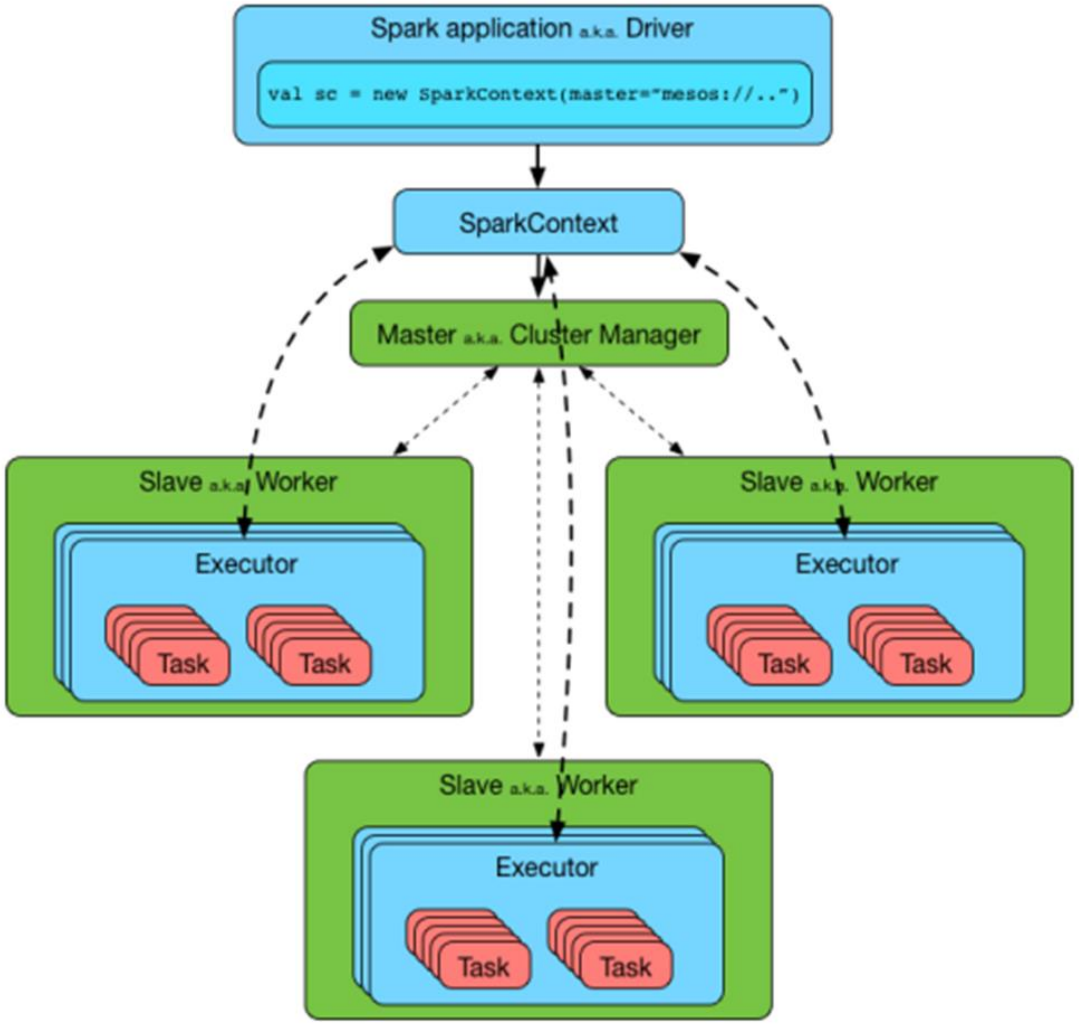
Čo je to Apache Spark?

- platforma na distribuované výpočty v clustroch
- rýchle a všeobecné použitie
- nasledovník MapReduce
- dávkové spracovanie, interaktívne algoritmy a query, streamové spracovanie

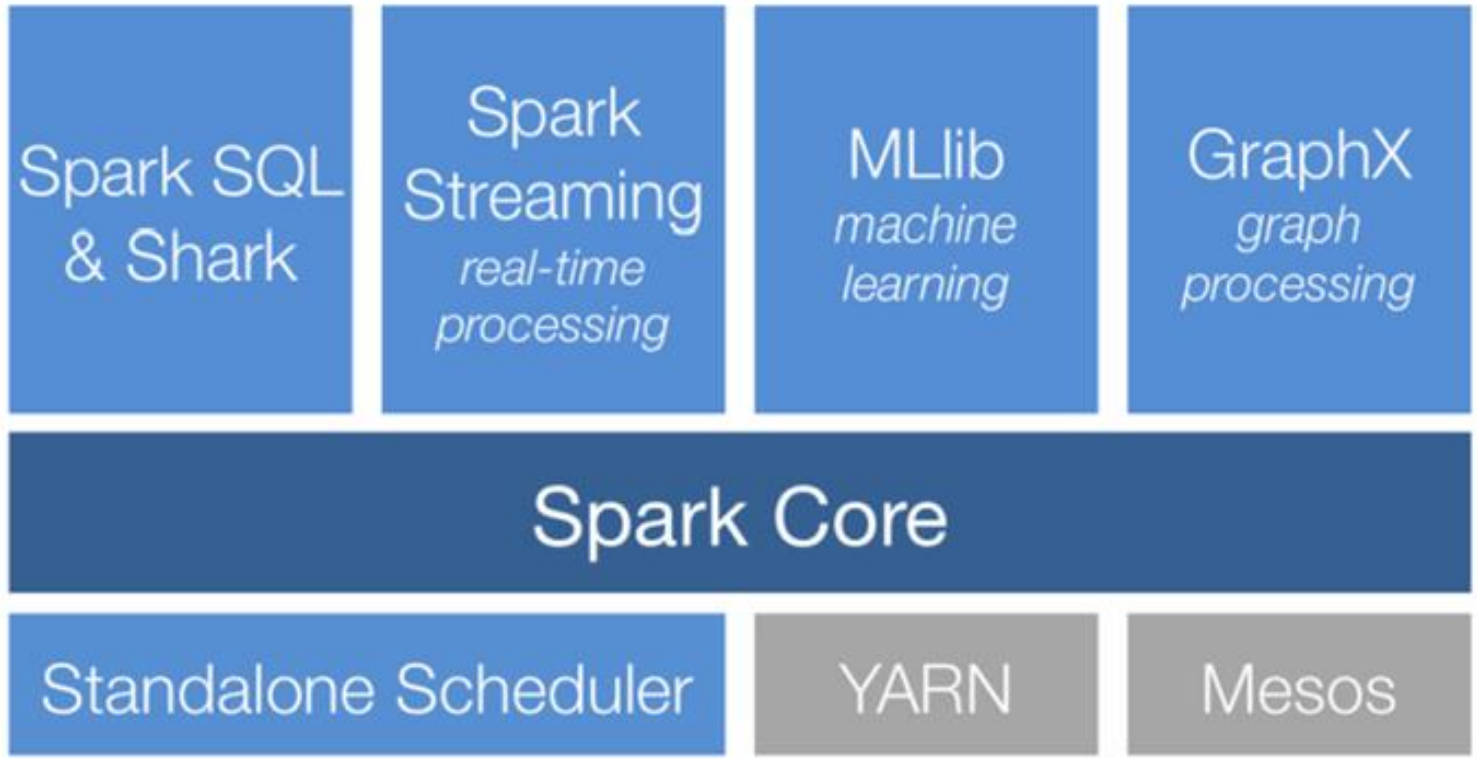
Komponenty prostredia

- jadro Sparku je "výpočtový engine"
- výhody jednotného stack-u:
  - zlepšenie nižších vrstiev = zlepšenie knižníc
  - minimálne náklady
  - kombinovanie rôznych modelov spracovania

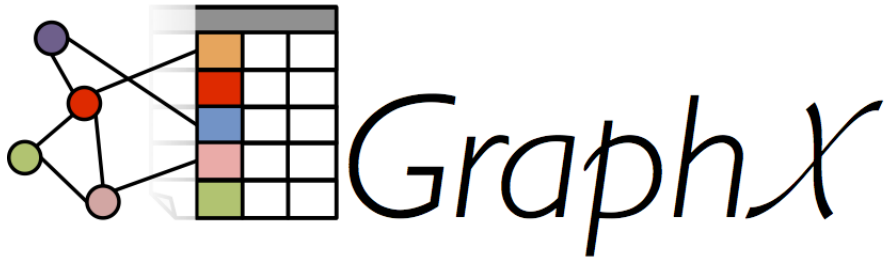
Architektúra Apache Spark



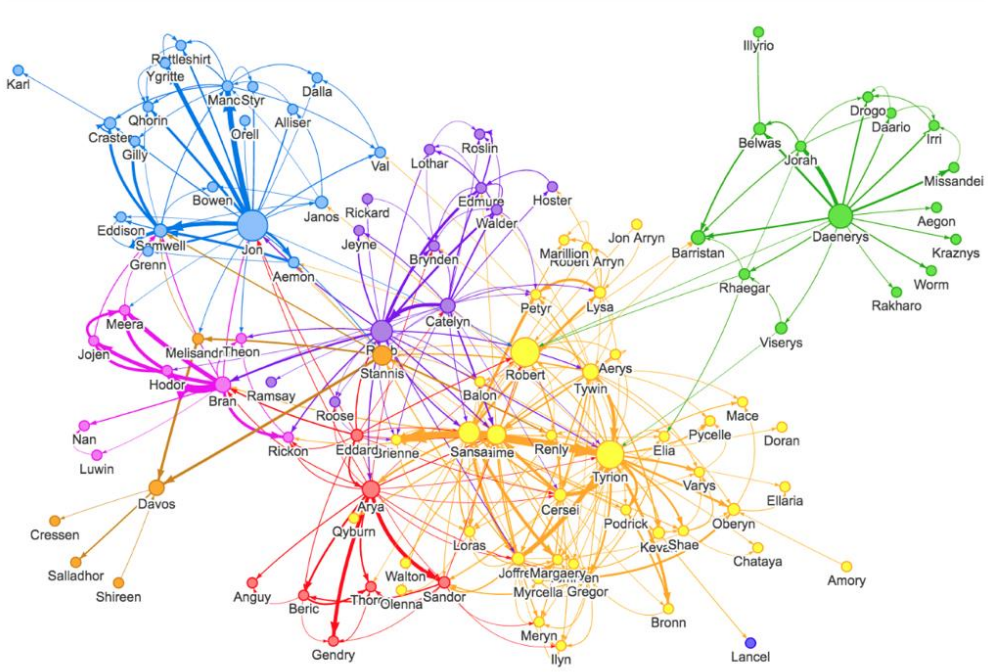
Stack Apache Spark



Spark GraphX



- komponenta rozširujúca Spark o manipuláciu s grafmi a vykonávanie paralelných výpočtov nad nimi
- rozširuje RDD o novú abstrakciu
- podpora výpočtov prostredníctvom základnej sady operácii a Pregel API
- rýchlo rastúca kolekcia grafových algoritmov a builderov



Graf vlastností

- orientovaný graf s viacerými hranami a s používateľom definovanými objektami
- viacero paralelených hrán
- zjednodušenie scénarov s viacerými vzťahmi medzi rovnakými vrcholmi
- parametrizovaný typami vrchol(VD) a hrana(ED)
- nemenné, distribuované a odolné voči chybám
- zmeny vykonávané vytvorením nového grafu

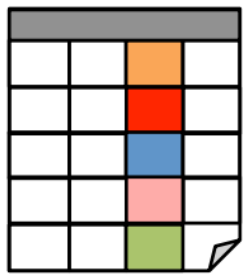
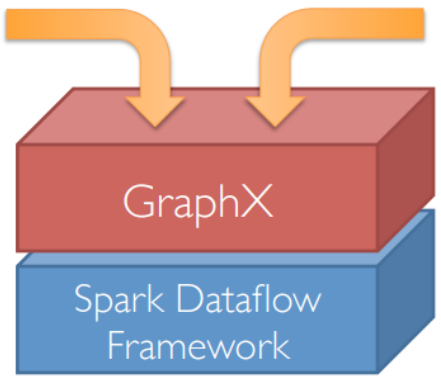
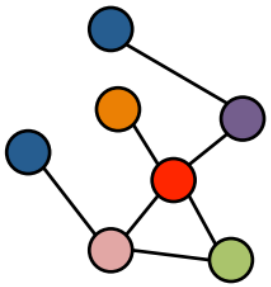


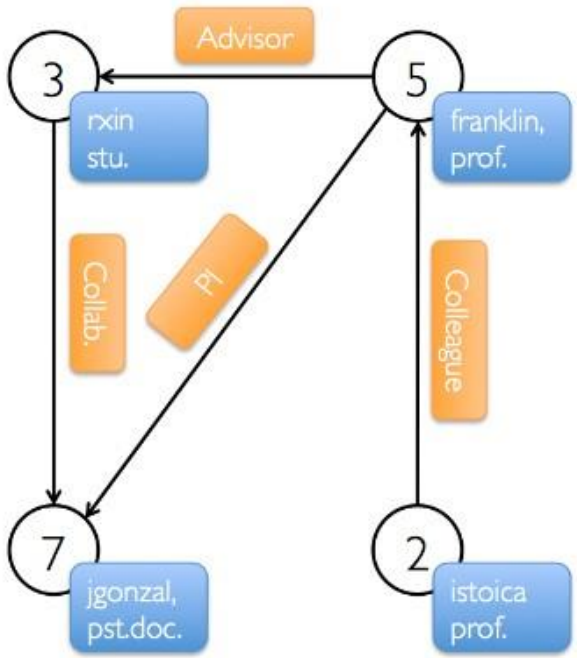
Table View





Graph View

Property Graph



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

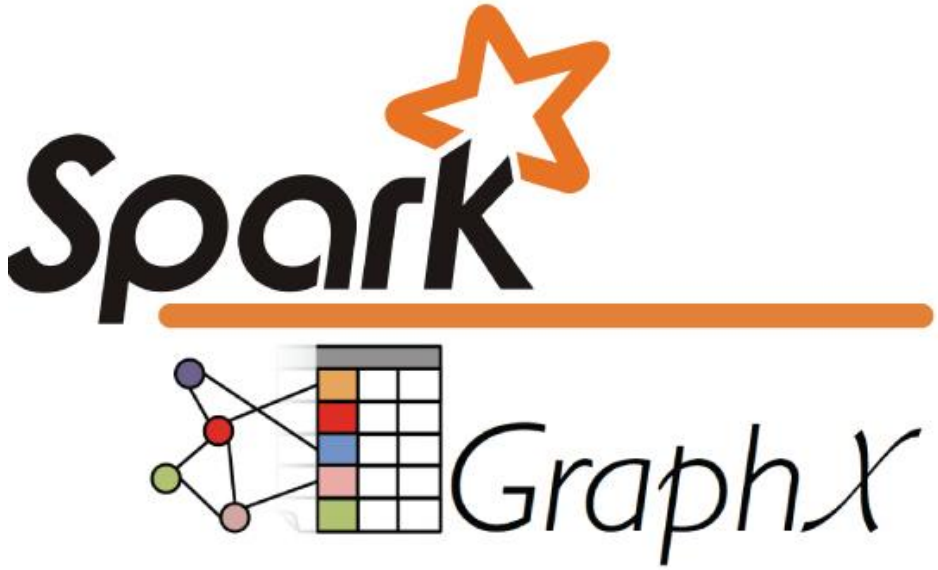
```scala
// Assume the SparkContext has already been constructed
val sc: SparkContext
// Create an RDD for the vertices
val users: RDD[(VertexId, (String, String))] =
  sc.parallelize(Seq((3L, ("rxin", "student")), (7L, ("jgonzal", "postdoc")),
                     (5L, ("franklin", "prof")), (2L, ("istoica", "prof"))))
// Create an RDD for edges
val relationships: RDD[Edge[String]] =
  sc.parallelize(Seq(Edge(3L, 7L, "collab"),    Edge(5L, 3L, "advisor"),
                     Edge(2L, 5L, "colleague"), Edge(5L, 7L, "pi")))
// Define a default user in case there are relationship with missing user
val defaultUser = ("John Doe", "Missing")
// Build the initial Graph
val graph = Graph(users, relationships, defaultUser)
```

Grafové operátory

- základné operátory *map*, *filter* a *reduceByKey*
- kolekcie základných operátorov
- 3 skupiny: operátory vlastností, štrukturálne operátory a spájacie operátory
- jadro všetkých operátorov definované v Graph, ich kompozície v GraphOps
- využívanie všetkých operátorov v rámci objektu Graph - Scala

Grafové buildery

- viacero spôsobov
- **Graph.groupEdges**
- **GraphLoader.edgeListFile**
- **Graph.apply**
- **Graph.fromEdges**
- **Graph.fromEdgeTuples**



Plusy a mínusy

Plusy

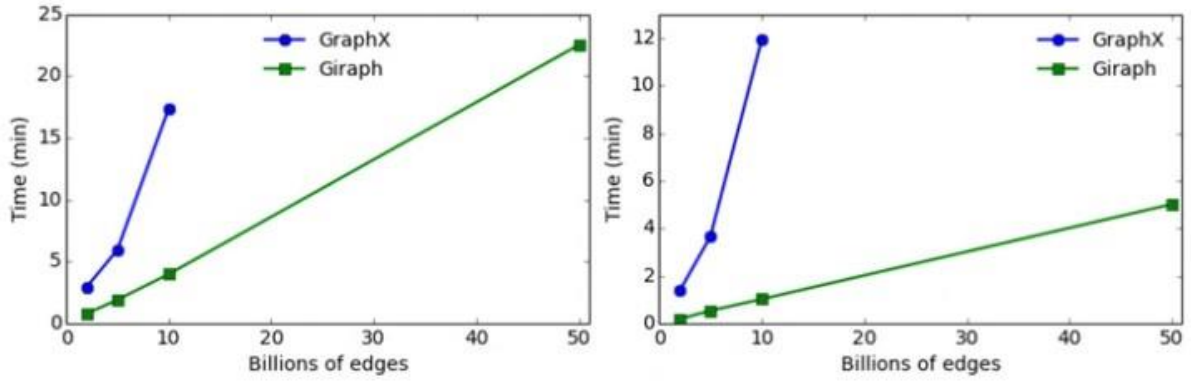
✓**Flexibilita** - práca s grafmi a výpočtami
- zjednocuje ETL, prieskumnú analýzu a iteratívne grafové výpočty
- dáta dokáže zobraziť ako grafy a zároveň ako kolekcie

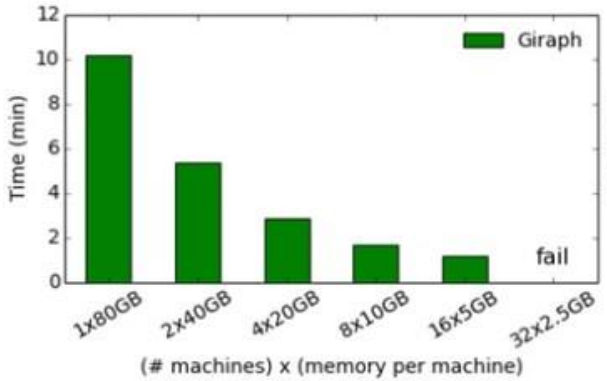
✓**Rýchlosť** - porovnateľný výkon s inými systémami na spracovanie grafov

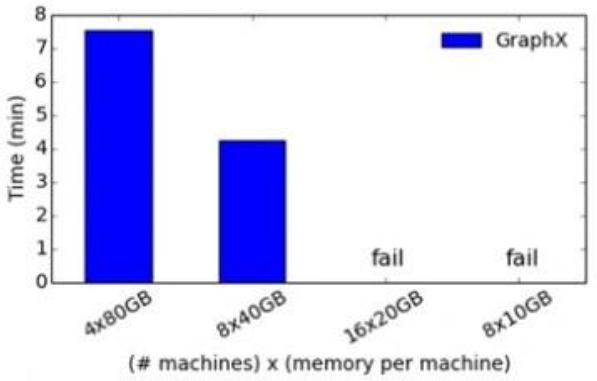
✓**Rastúca knižnica algoritmov** - široká škála grafových algoritmov
- page rank, prepojené komponenty, label propagation,...

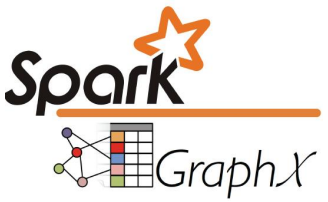
Mínusy

- **Škálovateľnosť** - veľká veľkosť -> strata výkonu
- **Efektivita využitia pamäte** - náročnejšie využitie pamäte

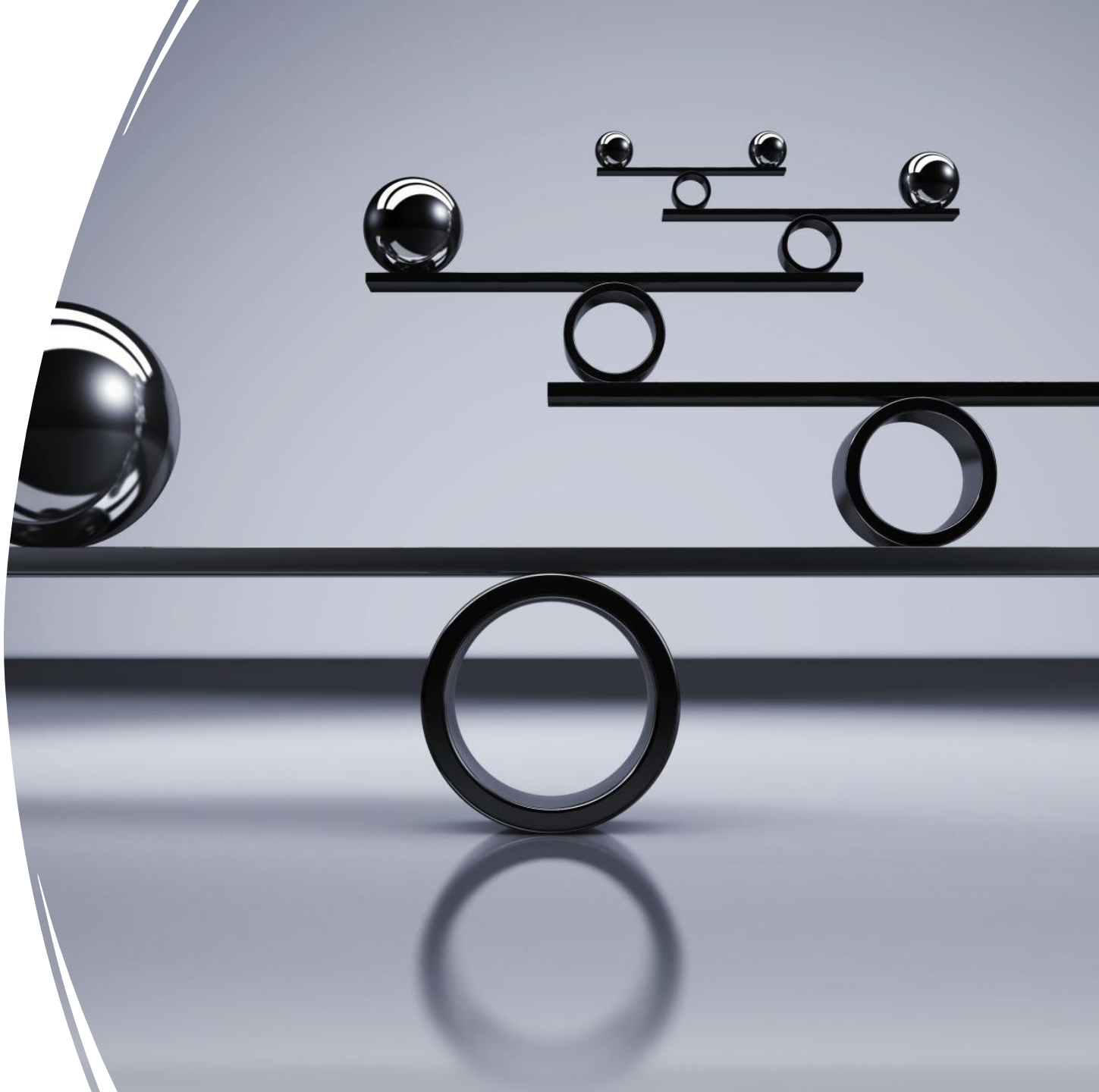


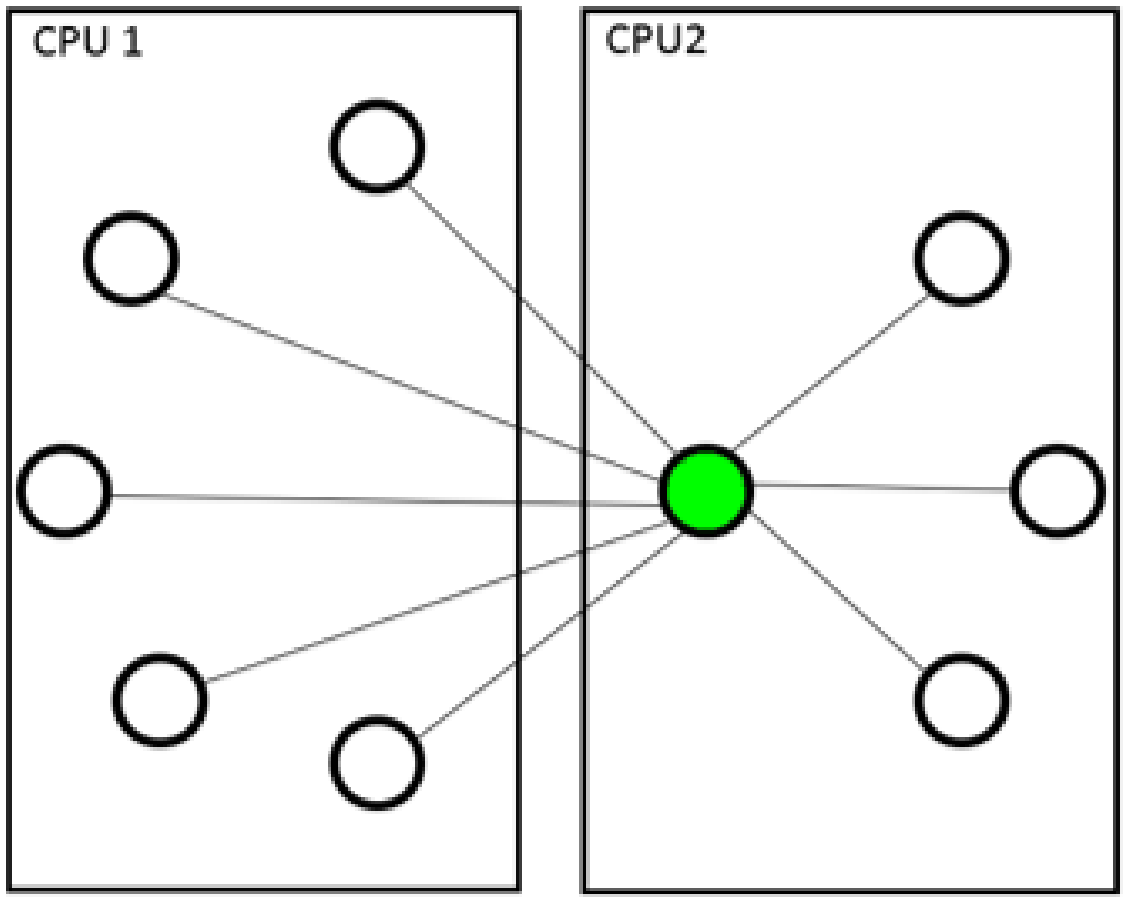






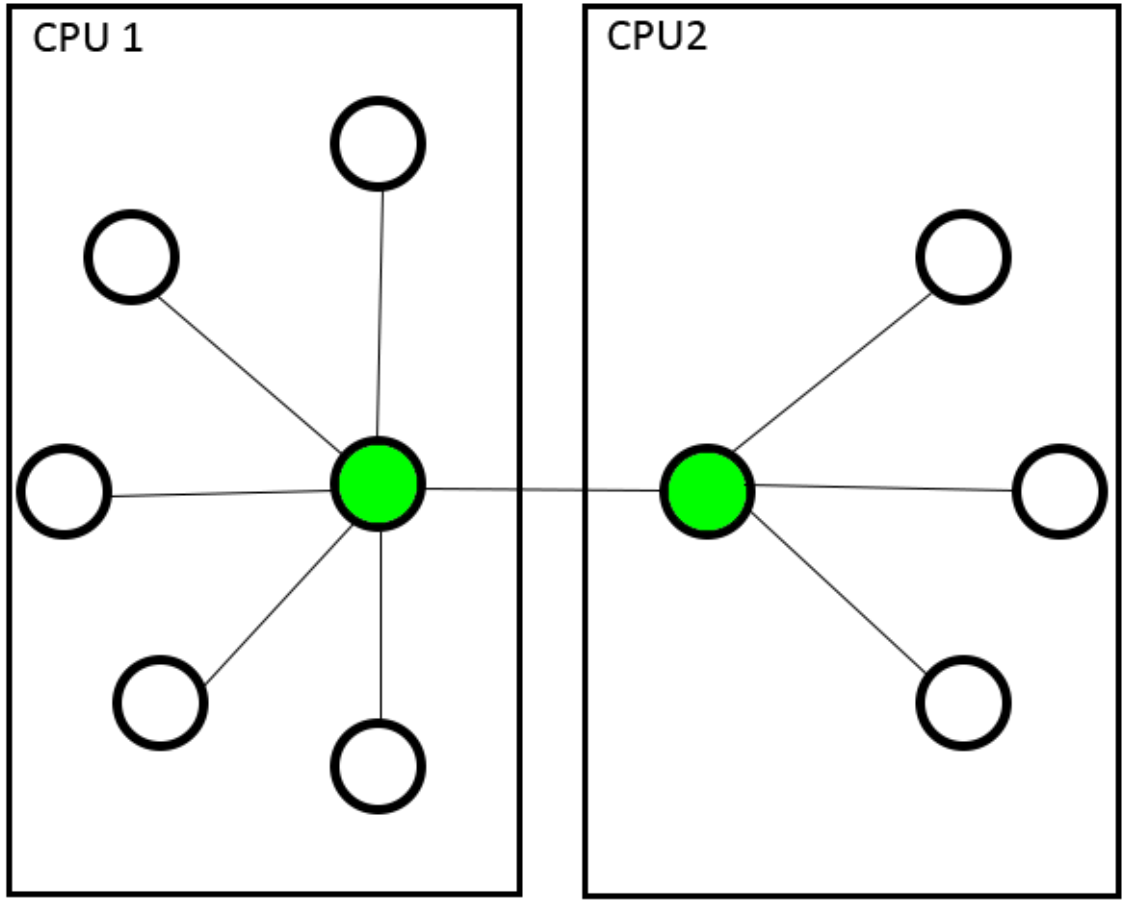
Load balancing





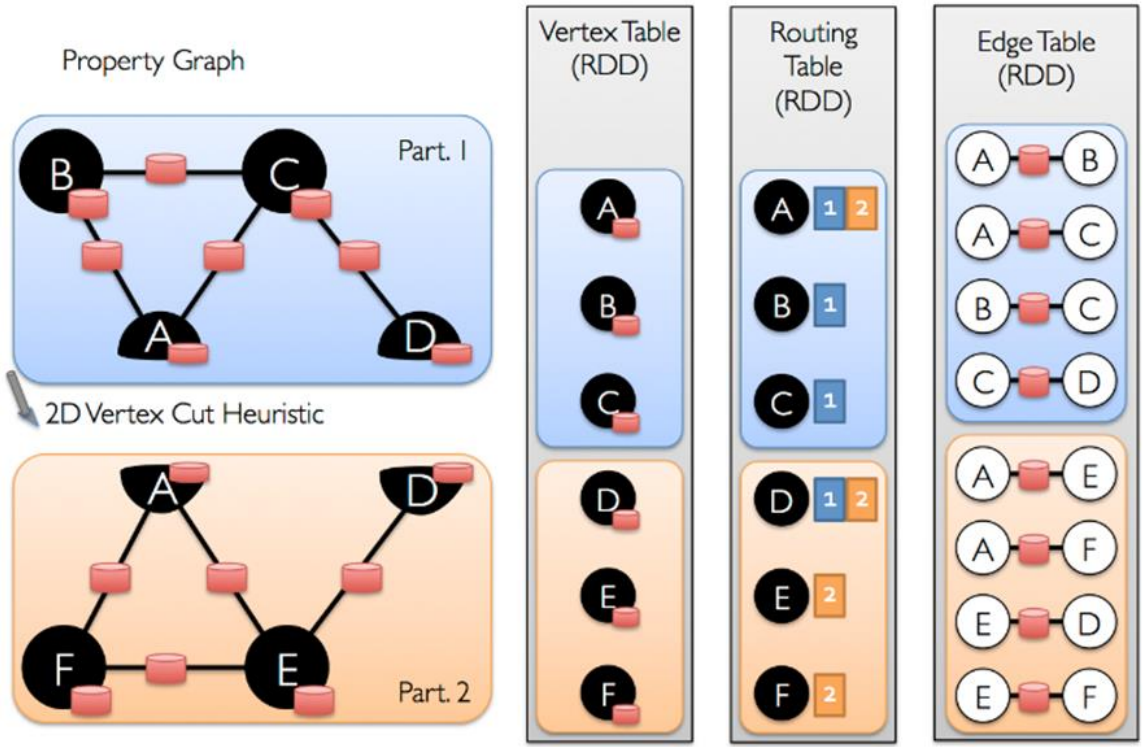
Edge cut

- Úspora úložného priestoru
- Zvýšená sieťová prevádzka



Vertex cut

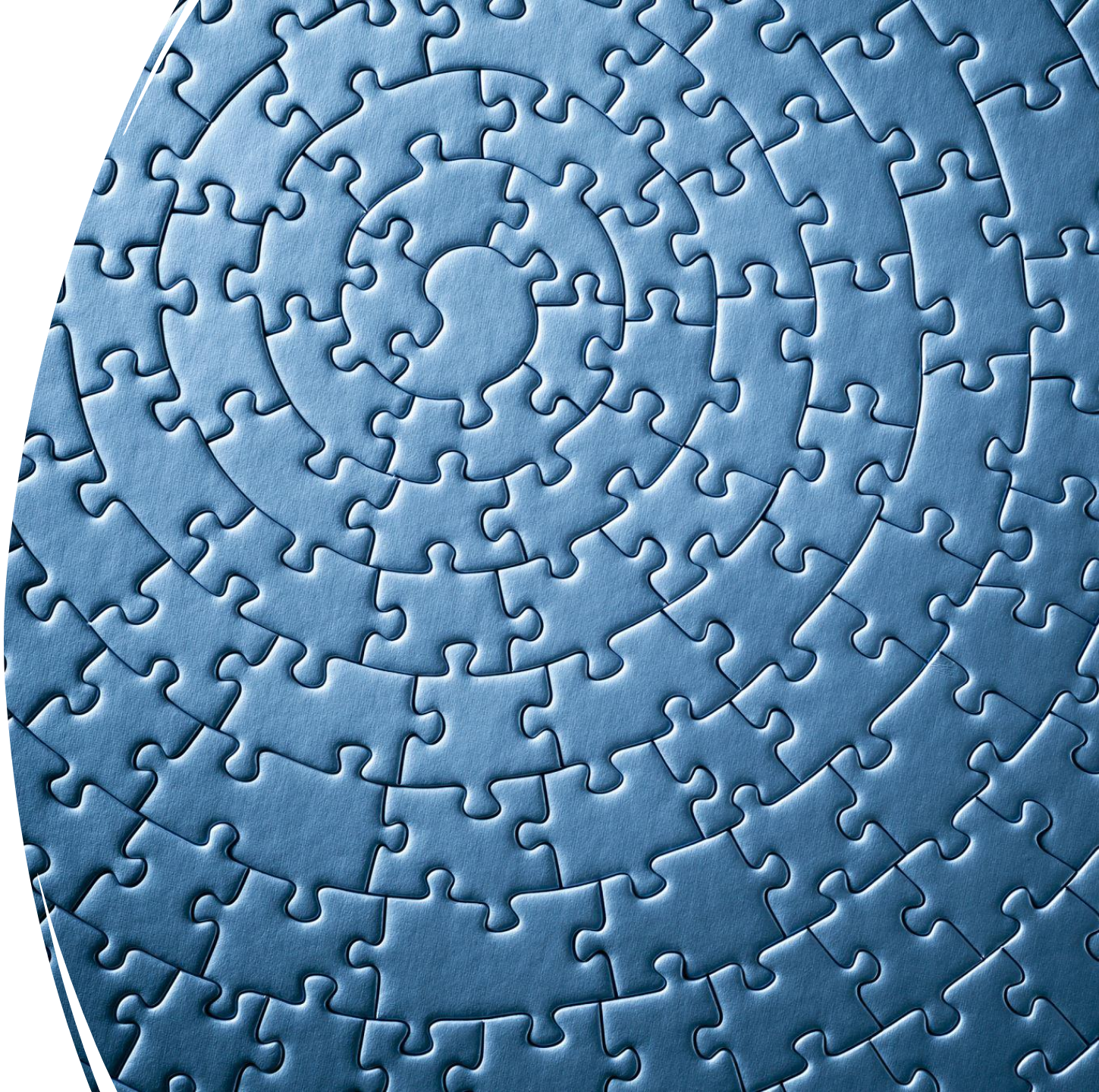
- Znížená sieťová prevádzka
- Zvýšené nároky na úložný priestor

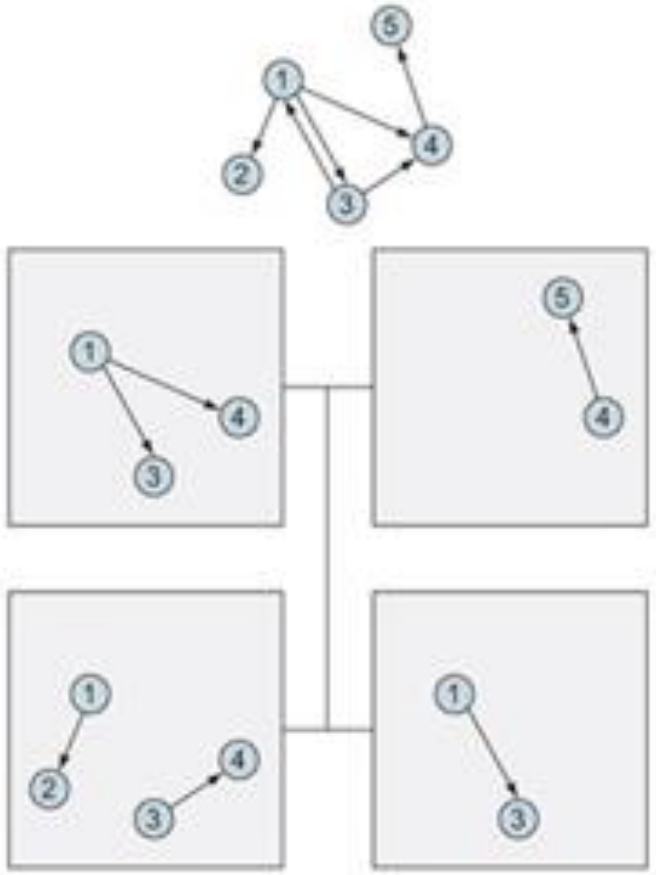


Vertex cut Only

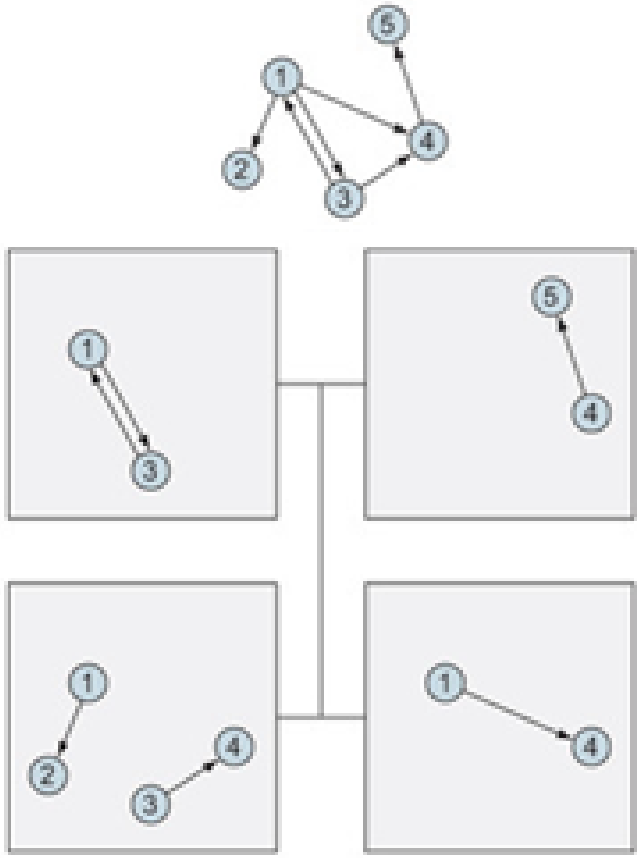
- Znížené sieťová prevádzka
- Znížené nároky na úložný priestor
- VertexTable (id, data)
- EdgeTable (pid, src, dst, data)
- RoutingTable (id, pid)

Partition Strategy

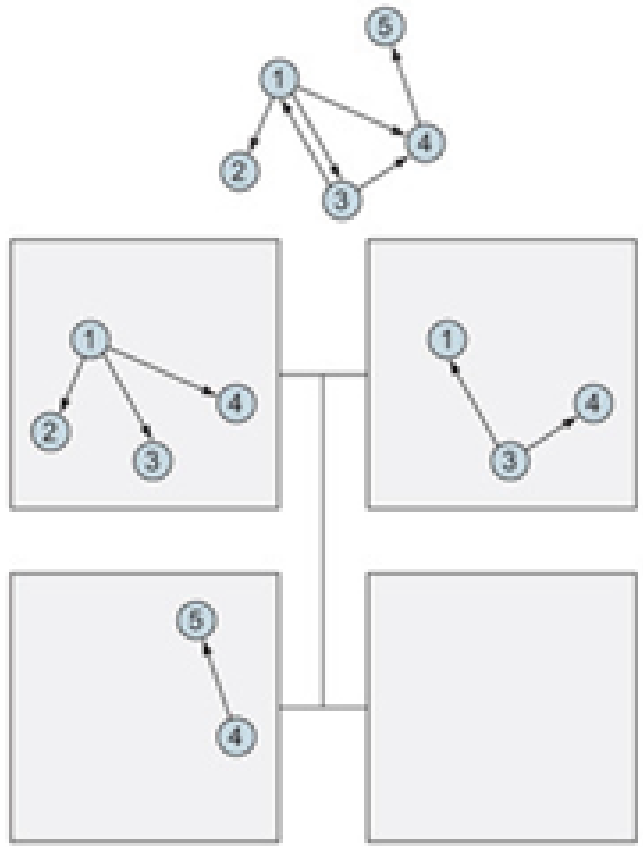




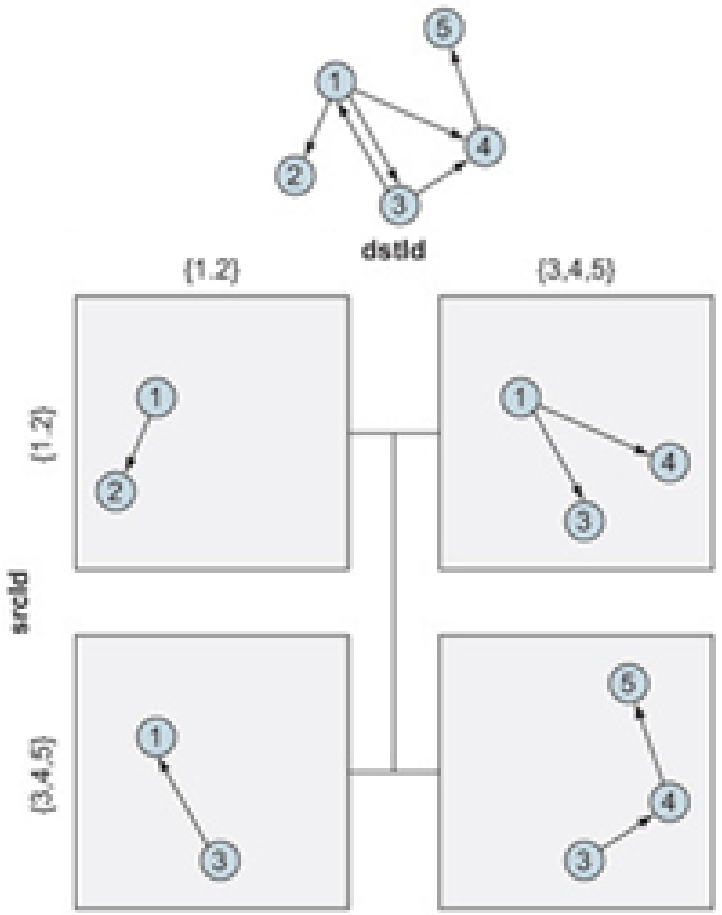
Random vertex cut



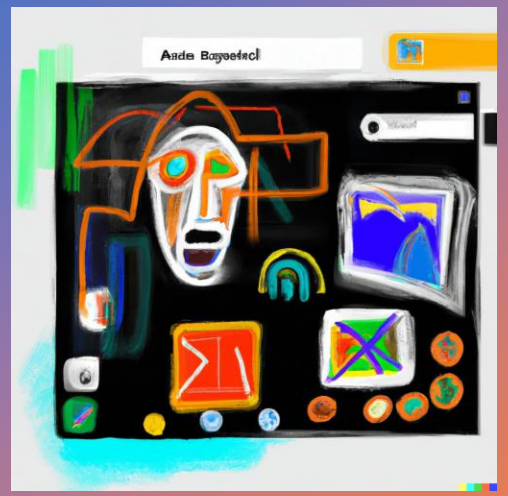
Canonical Random vertex cut



Edge Partition 1D



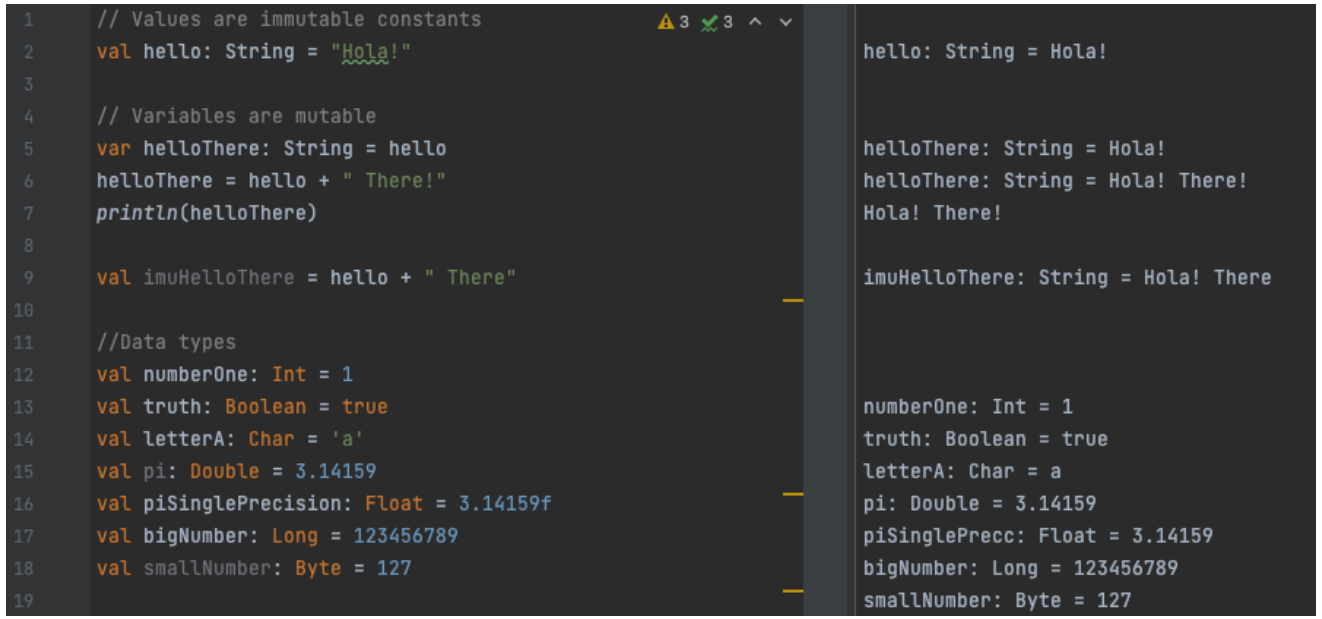
Edge Partition 2D



Demo aplikácie

Scala

- 'Scalable Language' - kombinácia objektovo-orientovaného a funkcionálneho prístupu
- staticky typovaná
- Beží na JVM ale je spustiteľná aj v Javascript Runtime Environment-e (Node.js)
- Bezproblémová interoperabilita s Javou
- Prečo Scala?
  - Spark je napisaný v Scale (Pyspark)
  - Menej ťažkopádna a prehľadnejšia ako Java
  - Popularita
  - Funkcionalny prístup
  - Paralelizmus a konkurencia
  - Rýchlosť, paralelizmus, type-safety (oproti Python-u - PySpark)



```
// Values are immutable constants
val hello: String = "Hola!"

// Variables are mutable
var helloThere: String = hello
helloThere = hello + " There!"
println(helloThere)

val imuHelloThere = hello + " There"

//Data types
val numberOne: Int = 1
val truth: Boolean = true
val letterA: Char = 'a'
val pi: Double = 3.14159
val piSinglePrecision: Float = 3.14159f
val bigNumber: Long = 123456789
val smallNumber: Byte = 127
```

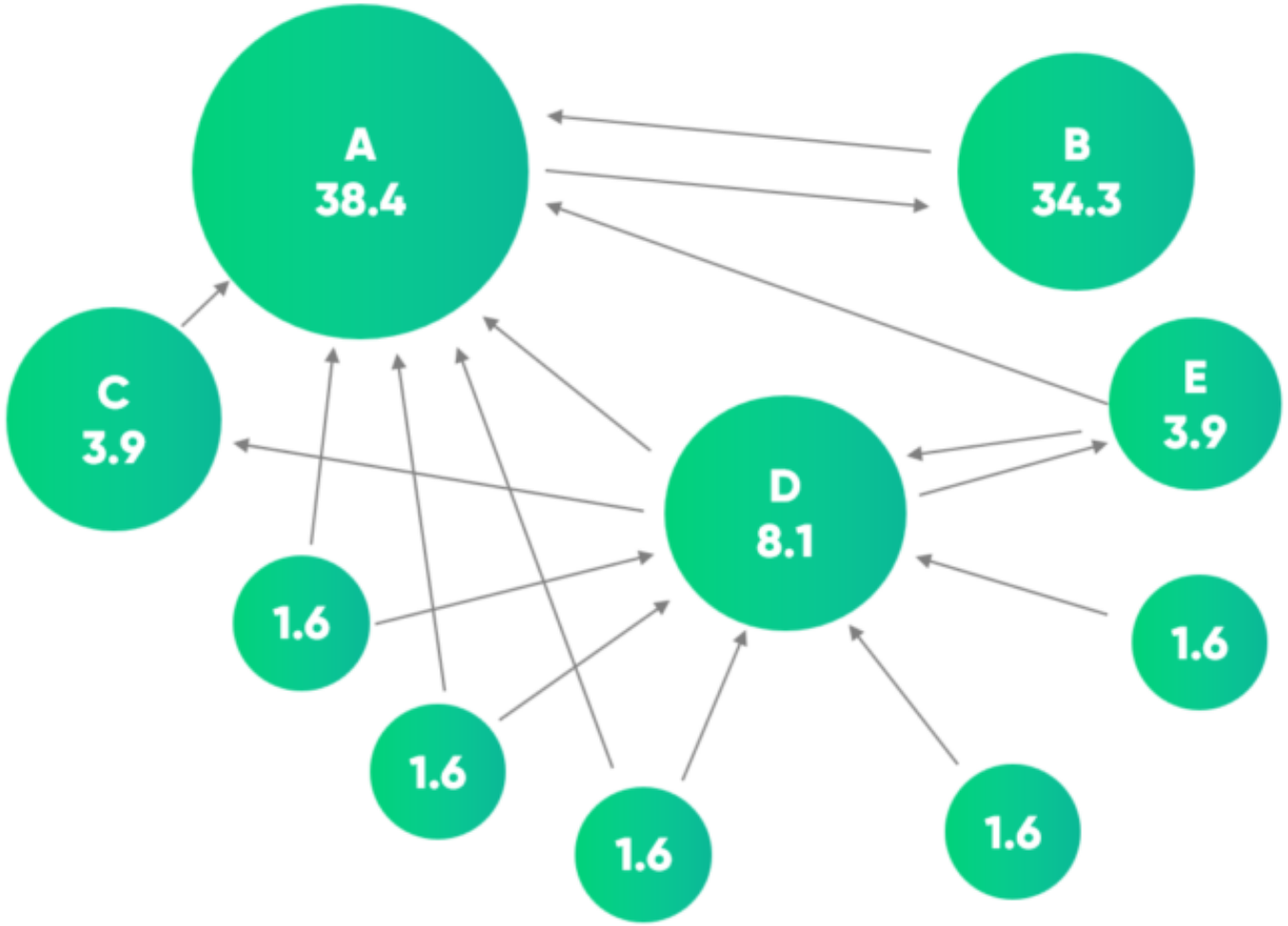
```
hello: String = Hola!

helloThere: String = Hola!
helloThere: String = Hola! There!
Hola! There!

imuHelloThere: String = Hola! There

numberOne: Int = 1
truth: Boolean = true
letterA: Char = a
pi: Double = 3.14159
piSinglePrecc: Float = 3.14159
bigNumber: Long = 123456789
smallNumber: Byte = 127
```

Page Rank



Delante

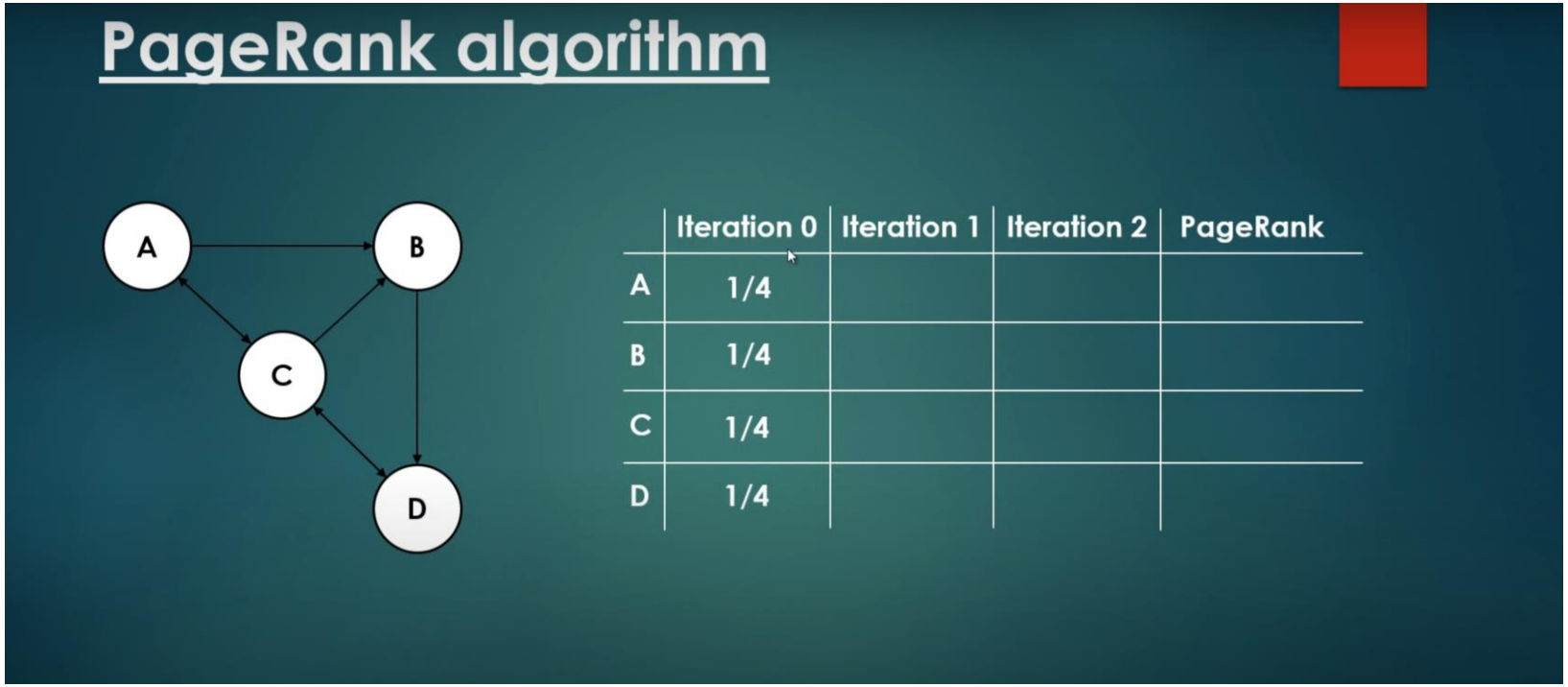
PageRank

- Formula pre iterácie

$$PR_{t+1}(P_i) = \sum_{P_j} \frac{PR_t(P_j)}{C(P_j)}$$

- Pagerank vrchola v daľšej iterácii

= Súčet Pagerankov vrcholov, kt. ukazujú na náš vrchol, vydelený počtom vecholov, na kt. daný vrchol ukazuje



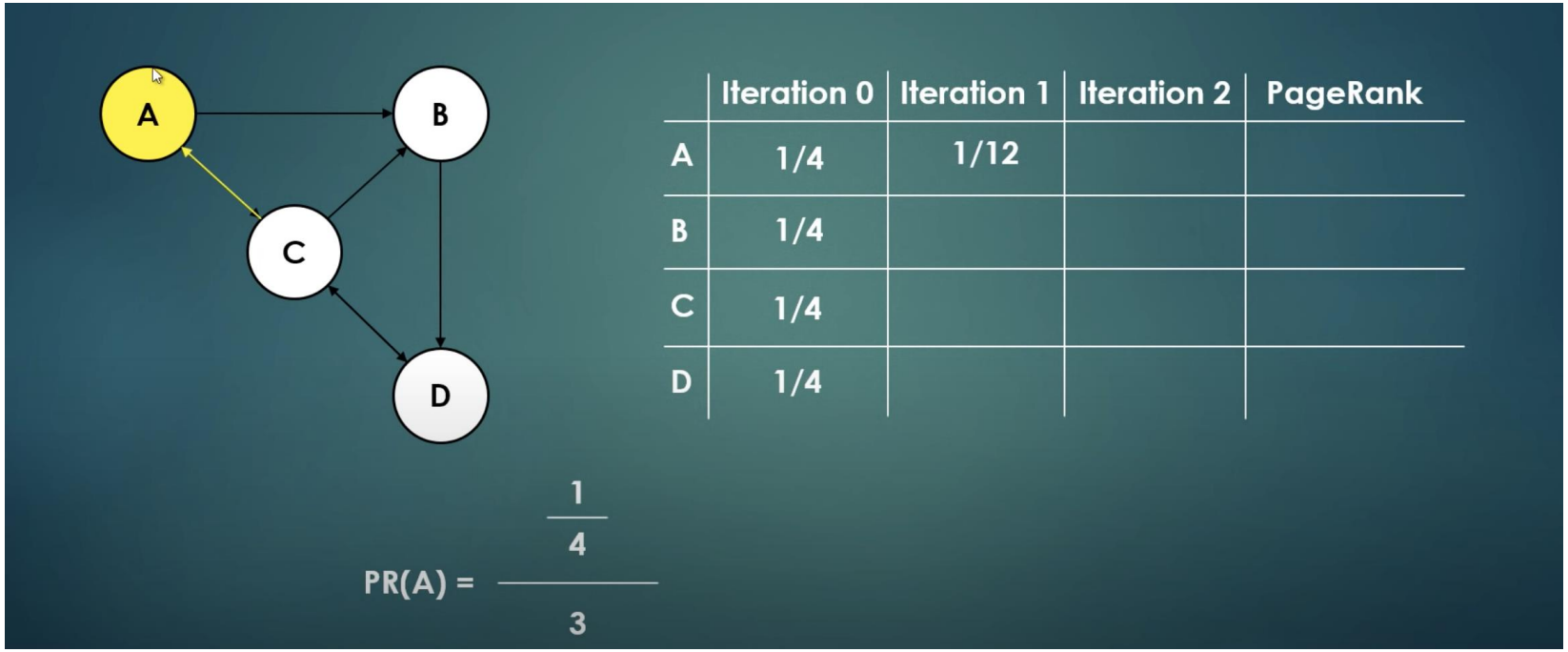
PageRank

- Formula pre iterácie

$$PR_{t+1}(P_i) = \sum_{P_j} \frac{PR_t(P_j)}{C(P_j)}$$

- Pagerank vrchola v daľšej iterácii

= Súčet Pagerankov vrcholov, kt. ukazujú na náš vrchol, vydelený počtom vecholov, na kt. daný vrchol ukazuje



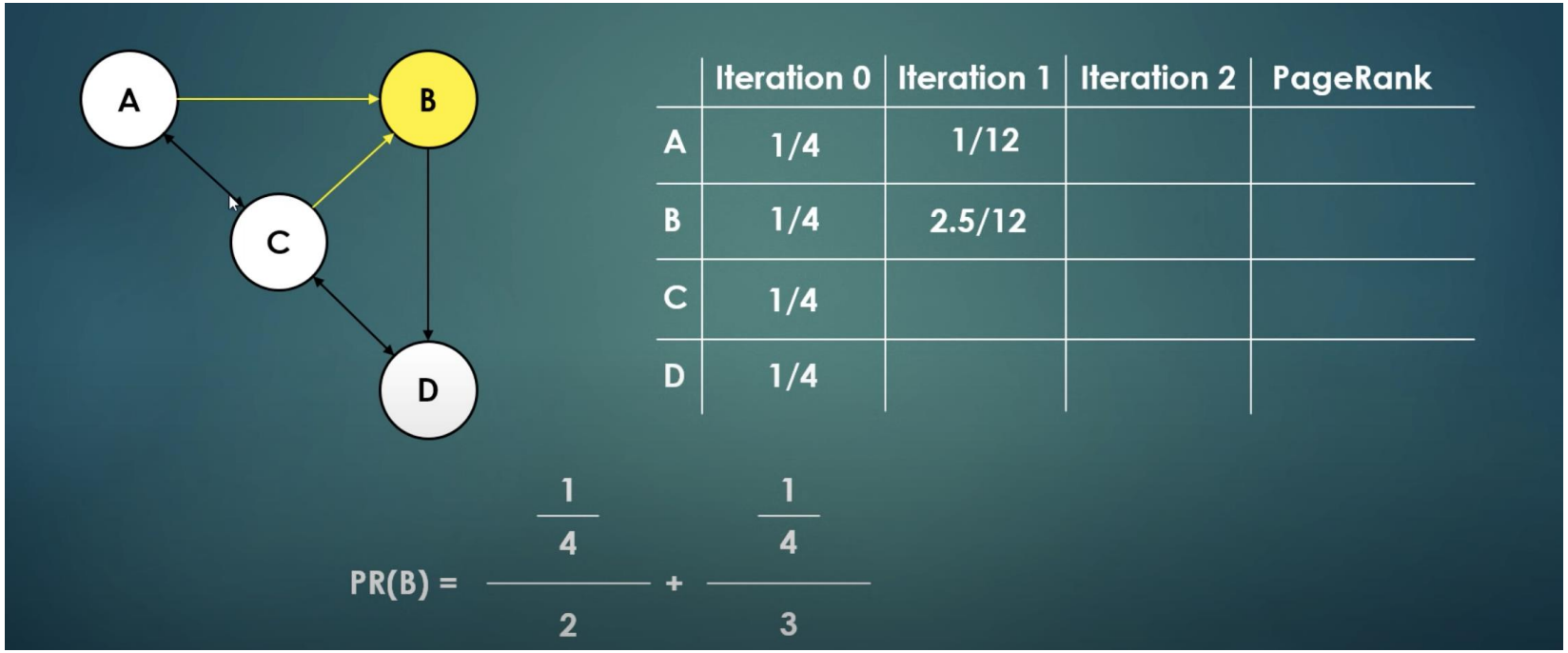
	Iteration 0	Iteration 1	Iteration 2	PageRank
A	1/4	1/12		
B	1/4			
C	1/4			
D	1/4			

PageRank

- Formula pre iterácie

$$PR_{t+1}(P_i) = \sum_{P_j} \frac{PR_t(P_j)}{C(P_j)}$$

- Pagerank vrchola v daľšej iterácii = Súčet Pagerankov vrcholov, kt. ukazujú na náš vrchol, vydelený počtom vecholov, na kt. daný vrchol ukazuje



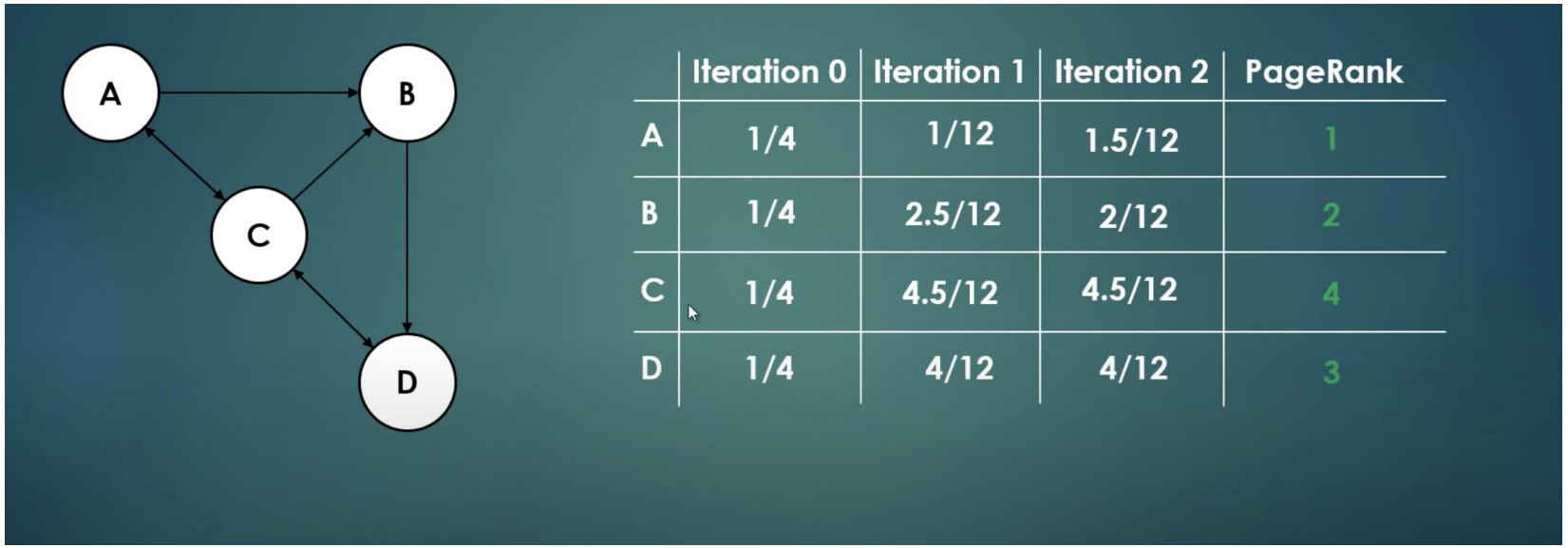
PageRank

- Formula pre iterácie

$$PR_{t+1}(P_i) = \sum_{P_j} \frac{PR_t(P_j)}{C(P_j)}$$

- Pagerank vrchola v daľšej iterácii

= Súčet Pagerankov vrcholov, kt. ukazujú na náš vrchol, vydelený počtom vecholov, na kt. daný vrchol ukazuje



	Iteration 0	Iteration 1	Iteration 2	PageRank
A	1/4	1/12	1.5/12	1
B	1/4	2.5/12	2/12	2
C	1/4	4.5/12	4.5/12	4
D	1/4	4/12	4/12	3

SparkContext a RDD

- **SparkContext** – vstupný bod pre funkcionalitu Spark-u
  - Reprezentuje spojenie so Spark Clusterom
  - Vytvára RDDs
  - Zdieľa premennné do clustera

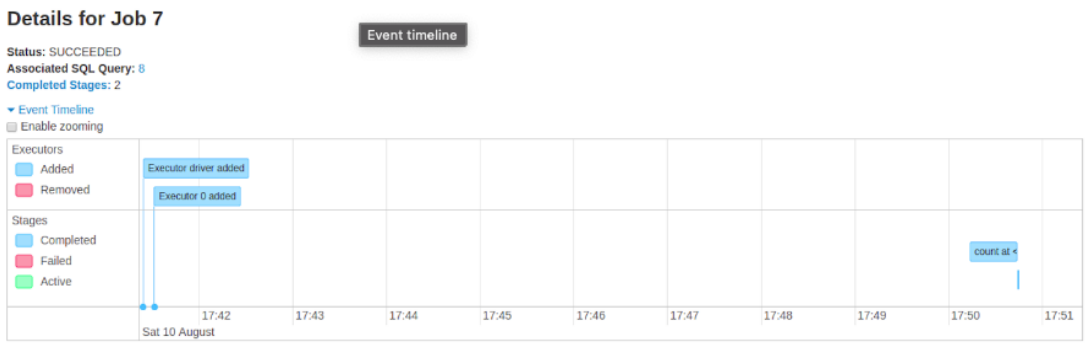
```
new SparkContext(master: String, appName: String, sparkHome: String = null, jars:
    Seq[String] = Nil, environment: Map[String, String] = Map())
    Alternative constructor that allows setting common Spark properties directly
```

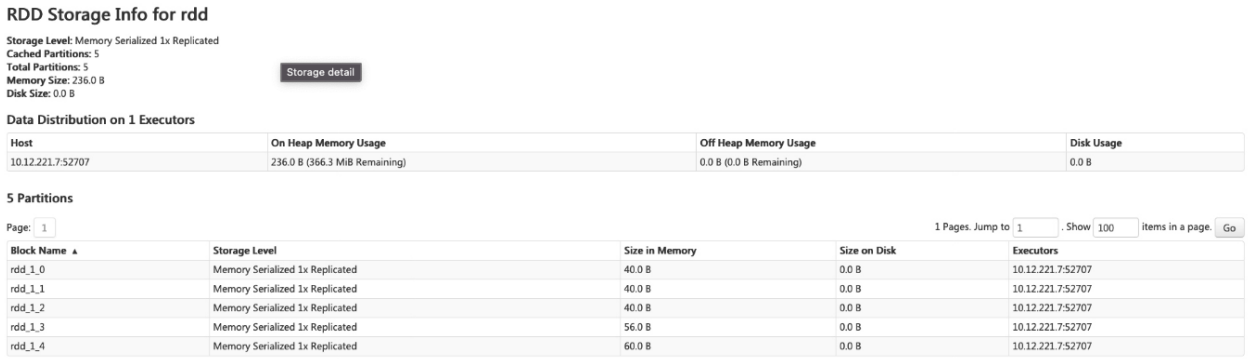
- **RDD** - *resilient distributed dataset*
  - ***Resilient =*** *'odolný' voči chybám*
  - ***Distributed =*** *'distibuovaný', je možné s ním pracovať paralelne*
    - paralelizácia existujúcej kolekcie v programe ovládača
    - odkazovanie na množinu údajov v externom úložnom systéme (HADOOP - HDFS, HBase)
  - *Dataset = 'množina údajov'*

	A	B	C	D	E	F
1	**Order ID**	**Product**	**Category**	**Amount**	**Date**	**Country**
2	1	Carrots	Vegetables	$4,270	1/6/2012	United States
3	2	Broccoli	Vegetables	$8,239	1/7/2012	United Kingdom
4	3	Banana	Fruit	$617	1/8/2012	United States
5	4	Banana	Fruit	$8,384	1/10/2012	Canada
6	5	Beans	Vegetables	$2,626	1/10/2012	Germany
7	6	Orange	Fruit	$3,610	1/11/2012	United States
8	7	Broccoli	Vegetables	$9,062	1/11/2012	Australia
9	8	Banana	Fruit	$6,906	1/16/2012	New Zealand
10	9	Apple	Fruit	$2,417	1/16/2012	France
11	10	Apple	Fruit	$7,431	1/16/2012	Canada
12	11	Banana	Fruit	$8,250	1/16/2012	Germany
13	12	Broccoli	Vegetables	$7,012	1/18/2012	United States
14	13	Carrots	Vegetables	$1,903	1/20/2012	Germany

SparkWebUI

- sada webových používateľských rozhraní (UIs) určených na monitorovanie klastra
- Prehliadanie Job-ov, Úložiska, Exekútorov, Stage-ov
- Job Details
  - Job Status: (running, succeeded, failed)
  - Event timeline
- Storage Tab
  - Zobrazenie použitých úložísk (RDDs, DataFrames)
- Etc





Otázka ku skúške

Ktoré metódy nepatria vo frameworku GraphX do partition strategy?

- RandomVertexCut
- RandomEdgeCut ✓
- EdgePartition1D
- CanonicalRandomVertexCut
- EdgePartition3D ✓

Zdroje

- GraphX Programming Guide
- Spark GraphX Tutorial – Graph Analytics In Apache Spark
- A comparison of state-of-the-art graph processing systems
- Performance and monitoring
- Graph distributions and storage
- Co je Apache Spark?
- Cluster Mode Overview
- What's the Difference Between SPARK 2014 and Apache Spark?
- PageRank Algorithm - Example - YouTube

Otázky?

Ďakujeme za pozornosť