

Velké Dáta (Big Data)

Apache Spark



Apache Spark a veľké dáta

Vitajte na prezentácii o Apache Spark a veľkých dátach. V nasledujúcich slidoch sa pozrieme na základné koncepty veľkých dát, architektúru Apache Spark, prácu s RDD a DataFrame, spracovanie dát pomocou Spark SQL a praktické ukážky implementácie.

Táto prezentácia je určená pre vývojárov, dátových analytikov a všetkých, ktorí sa zaujímajú o efektívne spracovanie veľkých objemov dát. Postupne prejdeme od základných konceptov až po pokročilé techniky spracovania dát v distribuovanom prostredí.



Miroslav Reiter

Unlock the power of distributed data



Apache Spark

Uložené údaje sú rozložené do rôznych časových intervalov. Tieto údaje sú potom procesované v rôznych časových intervaloch. Výsledok je vtedy, keď sú všetky údaje procesované.



Use Cases

Umožní vám rýchlosť a efektivitu spracovania veľkých dát. Spark je vhodný pre rôzne aplikácie, ako sú AI, machine learning, big data analýza a iné.



Apache Databricks

Umožní vám rýchlosť a efektivitu spracovania veľkých dát. Spark je vhodný pre rôzne aplikácie, ako sú AI, machine learning, big data analýza a iné.

Ako Začneme?

1. Pridajte si ma na LinkedIn

www.linkedin.com/in/miroslav-reiter

2. Mrknite na cvičenia

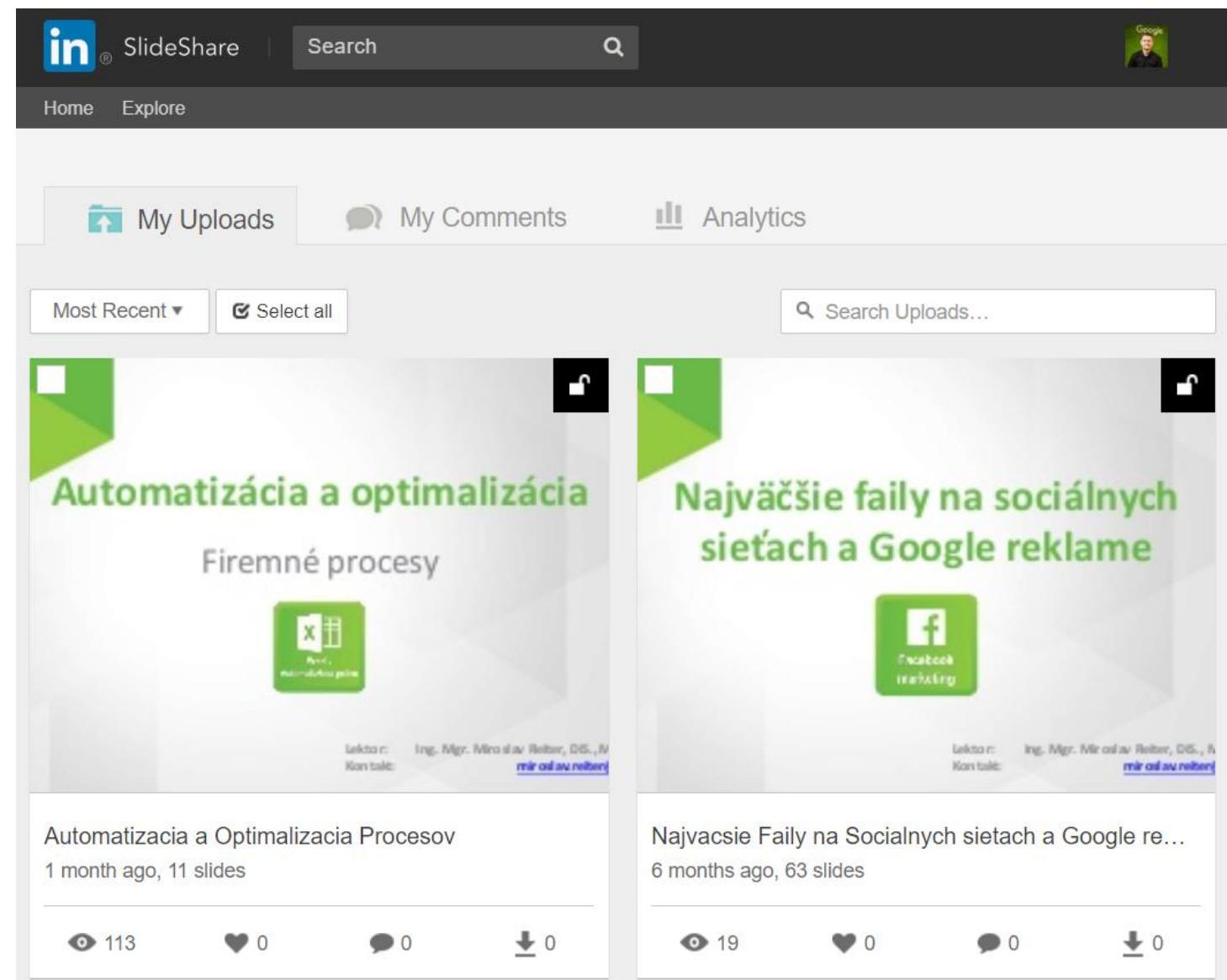
GitHub Repozitár

2. Prezentácia po prednáške

<https://github.com/miroslav-reiter/>

3. Videá na YouTube

www.youtube.com/@IT-Academy



Big_Data_Apache_Spark_Hive_Hadoop_Airflow Public

Pin Watch Fork Star

main 1 Branch 0 Tags

Go to file

Add file

Code

About



miroslav-reiter Add files via upload	0bc57f3 · 14 hours ago	25 Commits
Datasets Create a	2 days ago	
Prezentacie Create a	2 days ago	
Zdrojove_Subory_Skripty Add files via upload	2 days ago	
.gitignore Initial commit	2 days ago	
README.md Update README.md	yesterday	
shuffle-broadcast-apache-spark.jpg Add files via upload	14 hours ago	

README

Edit

More

Online kurz Big Data (Veľké Dáta), Apache Spark, Hive, Apache Hadoop, Apache Airflow

Praktické kurzy – RDD, DataFrame, SparkSQL a distribuované spracovanie dát

Obsah kurzu

1. [Úvod do veľkých dát a Apache Spark](#)
2. [Práca s RDD a DataFrame](#)
3. [Spark SQL a dopyty nad dátami](#)
4. [Nastavenie prostredia a Spark UI](#)
5. [Načítanie dát a transformácie](#)
6. [Zdroje a odporúčania pre Apache Spark](#)

1. Úvod do veľkých dát a nástroje pre spracovanie veľkých dát (Apache Spark, Hive, Apache Hadoop,

Materiály k online kurzom a školeniam Big Data (Veľké Dáta), Apache Spark, Hive, Apache Hadoop, Apache Airflow

www.vita.sk/online-kurz-vysokovykonne-...

python big-data apache-spark
bigdata apache apache-airflow
apache-hive apache-hadoop

Readme

Activity

0 stars

0 watching

0 forks

Releases

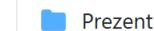
No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Languages

Jupyter Notebook 100.0%



Kurzy DataScience, Python, R, Julia, BI, AI/ML, ChatGPT

Materiály, Zdrojové Kódy a Projekty, Prezentácie ku kurzom DataScience, Python, OOP, R, BI, Analytika

Python je interpretovaný, interaktívny, open-source programovací jazyk. Python beží na mnohých variantoch Unixu, na Macu a Windowse (súčasťou kurzu bude inštalácia na vašom systéme). Pre absolvovanie kurzu je potrebné mať k dispozícii vlastný notebook (s ľubovoľným operačným systémom podporujúcim Python).

R je programovací jazyk a softvérové prostredie pre štatistickú analýzu a vizualizáciu. R je voľne dostupný pod GNU licenciou a existujú verzie pre mnohé operačné systémy ako Linux, Windows a Mac.

ChatGPT je pokročilý jazykový model umelej inteligencie vyvinutý spoločnosťou OpenAI. Je to variant modelu GPT (Generative Pre-trained Transformer), ktorá je špeciálne navrhnutá na generovanie textu a interakciu v dialógovej forme. Bol vyučovaný na veľkom množstve textových dát z internetu, čím získal schopnosť generovať koherentné a relevantné odpovede na zadanej otázky alebo pokyny v prirodzenom jazyku. ChatGPT, ako variant modelu GPT vyvinutého spoločnosťou OpenAI, sa primárne trénuje pomocou metódy zvanéj semi-supervised learning, ktorá je kombináciou supervised (riadenej) a unsupervised (neriadenej) učenia.

Používané nástroje

1 Jupyter Notebooks

About

Materiály, Zdrojové Kódy, Prezentácie ku kurzom Python, OOP, R, BI, Data Science, AI/ML, ChatGPT

www.vita.sk

python jupyter numpy oop
jupyter-notebook pandas python3
matplotlib sav beatifulsoup reiter
matplotlib-pyplot

Readme

Activity

23 stars

16 watching

25 forks

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Languages



Suggested workflows

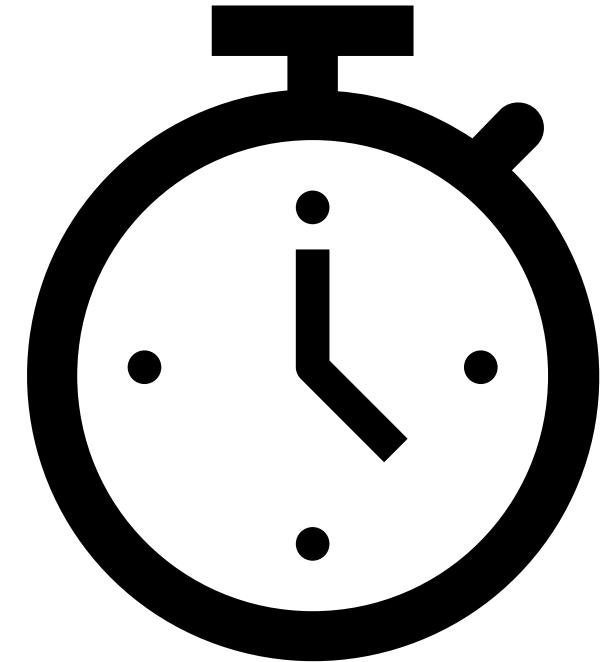
Based on your tech stack

Python application

Configure

Úvodné Informácie

- Časový rozvrh (9:00-13:30)
 - Pracujeme (50 min)
 - Prestávky (10 min)
 - Obedová prestávka
 - Mobilné telefóny a zariadenia
-
- Priprav si otázky a rovno sa pýtaj
 - Interaktívna forma



O mne - Miroslav Reiter

7

40000+
klientov a
1000+ firiem

IT Architekt
Programátor
Manažér

Microsoft
Google
ISTQB tréner

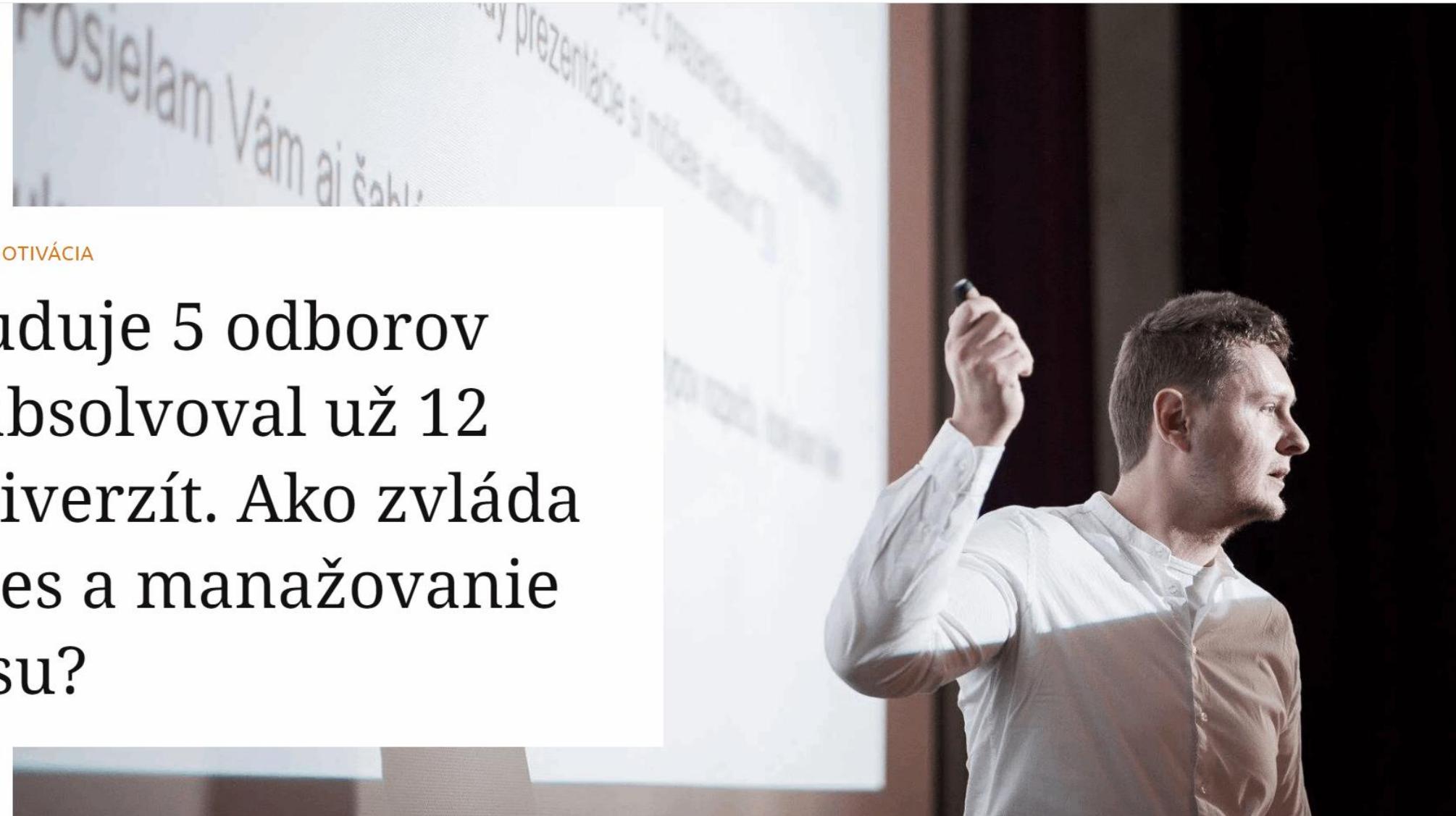
134 certifikácií

151 príručiek
a publikácií

13 škôl

62 projektov

Vlastná firma



MOTIVÁCIA

Študuje 5 odborov a absolvoval už 12 univerzít. Ako zvláda stres a manažovanie času?

Foto: Jakub Kovalík pre FMK UCM | Miroslav Reiter na prednáške Grow with Google na FMK UCM.



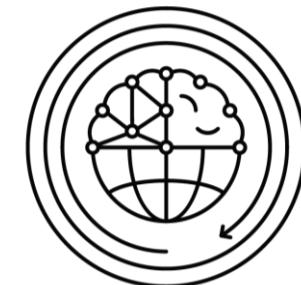
Nikola Kotláriková
19. júl 2022 · 8 min. čítania



Miroslav Reiter



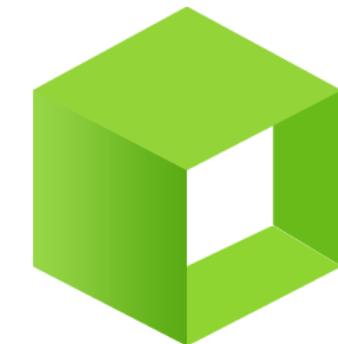
1. PhDr. VŠM (Podnikovný manažment)
2. Ing. STU FEI (**Aplikovaná informatika**)
3. Mgr. UK FM (**Strategický manažment a marketing**)
4. Mgr. VŠM (**Manažérstvo kvality**)
5. Mgr. VŠEMVŠ (Verejná správa)
6. Mgr. DTI (**Učiteľstvo ekonomických predmetov**)
7. DiS. AMOS (Cestovný ruch)
8. **MBA LIGS (Executive management)**
9. **DBA Humanum (IT manažment)**
10. MPA IES (Verejná správa a samospráva krajov)
11. MSc. Humanum (**Bezpečnosť inf. systémov**)
12. Ing. Paed. IGIP STU
13. Mgr. PEVŠ (**Bezpečnosť informačných systémov**)
14. RNDR. PEVŠ (**Bezpečnosť informačných systémov**)



FAKULTA MANAGEMENTU
Univerzita Komenského
v Bratislave

**DIGITÁLNA
UNIVERZITA**

STU
FIIT



IT ACADEMY

 **VITA**
ACADEMY

Most Valuable Professionals

[About](#) [Events](#) [Find an MVP](#)[Profile](#) [Events](#)

Headline

 IT Architect and Programmer  Hard worker 
Lecturer and Certified Trainer

Biography

 I'm a hard worker, intellectual and joker. I love learning and teaching as well. My main objective is to teach people and improve their IT knowledge. To create truly practical knowledge necessary for life and present it in an interesting way. I don't like snake charmers and people who cannot...

[▽ Read more](#)

Miroslav Reiter

 Slovakia IT Academy s.r.o.

Most Valuable Professionals

High Impact Activities

Award Category

M365

Technology Area

Visio, Excel

Languages

English, Slovak

Social



This community leader has not added a high impact activity yet.

Miroslav Reiter

získava status
Google Certified Trainer

Automation

Google



 [Domov](#) > [Registre](#) > [Znalcí](#) > PhDr. Ing. Miroslav Reiter

Znalec

PhDr. Ing. Miroslav Reiter

Evidenčné číslo: 915864

Miesto výkonu činnosti

Tomášikova 50
83104 Bratislava
Slovenská republika

[Zobraziť na mape](#)

Kontaktné údaje

Mobil: +421 908 163 084
E-mail: znapec@it-academy.sk

Odbory a odvetvia

Odbor / Odvetvie	História	Stav
100000 - Elektrotechnika		
100400 - Riadiaca technika, výpočtová technika (hardware)	14.02.2023 - Zápis	AKTÍVNY
100800 - Nosiče zvukových a zvukovoobrazových záznamov	14.02.2023 - Zápis	AKTÍVNY
100900 - Počítačové programy (software)	14.02.2023 - Zápis	AKTÍVNY
101000 - Bezpečnosť a ochrana informačných systémov	14.02.2023 - Zápis	AKTÍVNY

Vyberte si online kurz

Naučte sa programovať, tvoriť webstránky a grafiku, manažovať alebo sa zamerajte na osobný rozvoj. Všetko jednoducho vďaka našim online kurzom z pohodlia domova.

Ročné Predplatné na všetky Online Kurzy

2200 €

490 €

Prístup pre Vás do všetkých Aktuálnych aj
Pripravovaných Online Kurzov

12 mesačná platnosť

Kúpiť teraz

Získať viac



573 kurzov v ponuke



Zábavné online lekcie



Akreditované kurzy



13 rokov skúseností



Certifikovaní profesionálni lektori

Odporúčame Kurzy špeciálne pre vás



Balík Digitálny Marketing a Eshop

925,60€ 925,60€



Ročné predplatné kategórie SAP a ABAP

390,00€ 590,00€



Online kurz ChatGPT a DALL-E AI I. Začiatočník

186,00€ 254,00€



Online kurz SAP I. Začiatočník

298,00€ 398,00€



Online kurz Projektový Manažér I. Začiatočník

198,00€ 286,00€



Program MSc SAP Špecialista (SAP Consultant)

1 940,00€ 2 540,00€

[Všetko](#) [Obrázky](#) [Videá](#) [Správy](#) [Knihy](#) [Finance](#) [Krátke videá](#) [Viac ▾](#)[Nástroje ▾](#)[Stránky v slovenčine ▾](#) [Všetky dĺžky ▾](#) [Kedykoľvek ▾](#) [Ľubovoľná kvalita ▾](#) [Všetky videá ▾](#) [Akýkoľvek zdroj ▾](#) [Roz](#)

www.youtube.com › watch

Online Kurz Groovy - Čo je Apache Groovy (Groovy Script) a ...



<https://www.vita.sk/on...> Akreditovaný Prezenčný **Kurz Groovy:** → ;
<https://www.it-academy...> ★ Obsah nadupaného **kurzu Online Kurz Groovy** ...
YouTube · Miroslav Reiter - VITA Academy · 19. 6. 2023
1:42:25

11 kľúčových momentov v tomto videu ▾

www.youtube.com › watch

Online kurz Apache Tomcat I. Začiatočník - Ukážka - Netbeans ...



Príd na akreditovaný **kurz Apache Tomcat:** → <https://www.it-academy.sk/kurz/apache-tomcat-i-zaciatocnik/> Máme aj certifikovaný **Onlin...**
YouTube · Miroslav Reiter - VITA Academy · 1. 2. 2021
51:01

6 kľúčových momentov v tomto videu ▾

www.youtube.com › watch

Online Kurz Hadoop I. Začiatočník - Ukážka 2 - Čo je Hadoop ...



... Apache (0:30:00) VirtualBox (0:40:00) WMware Player (0: ...
Online kurz Online Marketingové Nástroje - Čo je to Lokálne SEO ...
YouTube · Miroslav Reiter - VITA Academy · 21. 3. 2020
55:19

www.youtube.com › watch

Online Kurz Apache Groovy - Ako Programovať v Groovy ...



Akreditovaný **Online Kurz Groovy:** → <https://www.vita.sk/online-kurz-groovy-i-zaciatocnik/> Akreditovaný Prezenčný **Kurz Groovy:** ...
YouTube · Miroslav Reiter - VITA Academy · 22. 7. 2023
33:53

www.youtube.com › watch

Online kurz Java X. a Java EE - Ukážka - Netbeans, JDK ...



Príd na akreditovaný **kurz JavaEE:** → <https://www.it-academy.sk/kurz/java-x-java-ee-servlety-a-jsp/> Máme aj certifikovaný **Online kurz Java,** ...
YouTube · Miroslav Reiter - VITA Academy · 22. 7. 2023

Všetko

Obrázky

Videá

Správy

Knihy

Finance

Krátke videá

Viac ▾

Nástroje ▾

www.youtube.com › watch

Online kurz HPC a Big Data - Ako začať s Big Data a Limity Dát



Absolvujte celý online kurz na → <https://www.vita.sk/online-kurz-vysokovykonne-pocitanie-hpc-a-velke-data-big-data/> Online kurz ...

YouTube · Miroslav Reiter - VITA Academy · 11. 1. 2025

www.youtube.com › watch

Online kurz HPC a Big Data - Úvod a Predstavenie kurzu



Absolvujte celý online kurz na → <https://www.vita.sk/online-kurz-vysokovykonne-pocitanie-hpc-a-velke-data-big-data/> Online kurz ...

YouTube · Miroslav Reiter - VITA Academy · 2. 3. 2025

www.youtube.com › watch

Online kurz HPC a Big Data - Ako začať s HPC a Big Data ...



Absolvujte celý online kurz na → <https://www.vita.sk/online-kurz-vysokovykonne-pocitanie-hpc-a-velke-data-big-data/> Online kurz ...

YouTube · Miroslav Reiter - VITA Academy · 4. 1. 2025

www.youtube.com › watch

Online kurz HPC a Big Data - Ako Pracovať s Big Data ...



Absolvujte celý online kurz na → <https://www.vita.sk/online-kurz-vysokovykonne-pocitanie-hpc-a-velke-data-big-data/> Online kurz ...

YouTube · Miroslav Reiter - VITA Academy · 4. 3. 2025

www.youtube.com › watch

Online Kurz Hadoop I. Začiatočník - Ukážka - Úvod Analytiky ...



... [online-kurz-hadoop-i-zaciatochnik/](#) Chceš začať alebo začínaš s Hadoopom a veľkými dátami (Big Data)? Chceš sa naučiť analyzovať veľké dáta ...

YouTube · Miroslav Reiter - VITA Academy · 21. 3. 2020

www.youtube.com › watch

Online kurz HPC a Big Data - Knižnica Polars, Filtrovanie a ...



Comments3 · [Online kurz HPC a Big Data - Úvod a Predstavenie kurzu](#) ·

[Všetko](#) [Správy](#) [Obrázky](#) [Videá](#) [Krátke videá](#) [Knihy](#) [Web](#) [Viac ▾](#)[Nástroje ▾](#)

www.youtube.com › watch

Online kurz HPC a Big Data - Ako začať s Big Data a Limity Dát



Absolvujte celý [online kurz](#) na → <https://www.vita.sk/online-kurz-vysokovykonne-pocitanie-hpc-a-velke-data-big-data/> [Online kurz ...](#)

YouTube · Miroslav Reiter - VITA Academy · 11. 1. 2025

www.youtube.com › watch

Online Kurz Microsoft Excel - Práca s Veľkými Dátami



Absolvujte celý [online kurz](#) na → <https://www.vita.sk/online-kurz-excel-praca-s-velkymi-datami/> [Online kurz Microsoft Excel - Práca s Veľkými ...](#)

YouTube · Miroslav Reiter - VITA Academy · 2. 3. 2024



3 kľúčové momenty v tomto video ▾

www.youtube.com › watch

Kurz Excel - Pokročilé Techniky a Veľké Dáta - Ukážka - Výber ...



... [kurz/microsoft-excel-pokrocile-tehniky-a-velke-data/](#) Máme aj certifikovaný [Online kurz Microsoft Excel Pokročilé Techniky a Veľké Dáta ...](#)

YouTube · Miroslav Reiter - VITA Academy · 16. 3. 2020

22 kľúčových momentov v tomto video ▾

www.youtube.com › watch

Online kurz HPC a Big Data - Úvod a Predstavenie kurzu



Absolvujte celý [online kurz](#) na → <https://www.vita.sk/online-kurz-vysokovykonne-pocitanie-hpc-a-velke-data-big-data/> [Online kurz ...](#)

YouTube · Miroslav Reiter - VITA Academy · 2. 3. 2025

www.youtube.com › watch

Online kurz HPC a Big Data - Ako Pracovať s Big Data ...



Absolvujte celý [online kurz](#) na → <https://www.vita.sk/online-kurz-vysokovykonne-pocitanie-hpc-a-velke-data-big-data/> [Online kurz ...](#)

YouTube · Miroslav Reiter - VITA Academy · 4. 3. 2025

AKREDITOVANÉ VZDELÁVACIE PROGRAMY



Vyhľadávanie akreditovaných vzdelávacích programov
(vyplňte vstupné polia, podľa ktorých chcete vyhľadávať)

Názov programu / modulu

python

Číslo akreditácie / štvorčíslo**Mesto / okres / kraj****Vzdelávacia inštitúcia****Vstupné vzdelanie****Organizačná forma****Vyhľadať**[Jednoduché vyhľadávanie](#)**Počet nájdených záznamov: 15**

Názov	Dátum akreditácie	Rozsah	Inštitúcia	Mesto
Programovacie jazyky / Python I. - Základy_(POA: 3428/2020/122/5)	10. 12. 2020	10,0 hod.	IT Academy s. r. o.	Bratislava
Programovacie jazyky / Python II. - Mierne pokročilé techniky_(POA: 3428/2020/122/5)	10. 12. 2020	10,0 hod.	IT Academy s. r. o.	Bratislava
Programovacie jazyky / Python - Automatizácia Práce (POA: 3428/2020/122/5)	10. 12. 2020	10,0 hod.	IT Academy s. r. o.	Bratislava
Programovacie jazyky / Python - Knižnice_(POA: 3428/2020/122/5)	10. 12. 2020	10,0 hod.	IT Academy s. r. o.	Bratislava
Programovacie jazyky / Python - Pandas_(POA: 3428/2020/122/5)	10. 12. 2020	10,0 hod.	IT Academy s. r. o.	Bratislava
Programovacie jazyky / Python Django I. - Základy_(POA: 3428/2020/122/5)	10. 12. 2020	10,0 hod.	IT Academy s. r. o.	Bratislava
Programovacie jazyky / Python Django II. - Mierne pokročilé techniky_(POA: 3428/2020/122/5)	10. 12. 2020	10,0 hod.	IT Academy s. r. o.	Bratislava
Programovacie jazyky / Python Django III. - Pokročilé techniky_(POA: 3428/2020/122/5)	10. 12. 2020	10,0 hod.	IT Academy s. r. o.	Bratislava
Programovacie jazyky / Python III. - Pokročilé techniky_(POA: 3428/2020/122/5)	10. 12. 2020	10,0 hod.	IT Academy s. r. o.	Bratislava
Programovacie jazyky / Python IV. - Návrhové Vzory_(POA: 3428/2020/122/5)	10. 12. 2020	10,0 hod.	IT Academy s. r. o.	Bratislava

1 2

Adresa pracoviska:

Ministerstvo školstva, výskumu, vývoja a mládeže Slovenskej republiky

Sekcia stredných škôl a celoživotného vzdelávania

Odbor vzdelávania dospelých

Stromová 1

813 30 Bratislava 1

Informačný systém d'álšieho vzdelávania

Detail vzdelávacieho programu

Vzdelávací program

Názov vzdelávacej inštitúcie	IT Academy s. r. o.
Adresa vzdelávacej inštitúcie	Tomášikova 12570/50A , 831 04 Bratislava
Názov vzdelávacieho programu	Operačné systémy, databázy a analyтика
Názov modulu	Veľké Dáta Big Data
Celkový rozsah	10,0
Číslo potvrdenia o akreditácii	3428/2020/122/2
Dátum vydania potvrdenia	10. 12. 2020
Cieľová skupina	Osoby so záujmom získať odborné vedomosti a zručnosti v oblasti operačných systémov a databáz, ktoré si chcú rozšíriť znalosti o fungovaní, správu a konfiguráciu systémových a databázových riešení, a ktoré chcú pracovať alebo pracujú ako administrátori operačných systémov a aplikácií, databázoví administrátori, softvéroví architekti, databázoví konzultanti a špecialisti.
Profil absolventa	Absolvent modulu dokáže budovať Big Data platformy, ukladať a spracovať dátu do užitočnej podoby. Vie, ako sledovať rôzne situácie okamžite aj za dlhú história, nazerať na rôzne udalosti z rôznych pohľadov (napr. cez rôzne oddelenia) a získavať komplexný pohľad, ktorý pomáha pochopiť situáciu na trhu.
Odborný garant	Ing. Mgr. Miroslav Reiter MBA, DBA
Lektori	Ing. Ľudovít Kovačovič , Ing. Mário Kašuba , Ing. Mgr. Miroslav Reiter MBA, DBA

Adresa pracoviska:

Ministerstvo školstva, výskumu, vývoja a mládeže Slovenskej republiky
Sekcia stredných škôl a celoživotného vzdelávania
Odbor vzdelávania dospelých
Stromová 1
813 30 Bratislava 1

©2025

Mapa stránky**Občania**[Zákon o celoživotnom vzdelávaní](#)**Inštitúcie**[Pokyn k vypracovaniu žiadosti o akreditáciu vzdelávacieho programu](#)**Stredné a vysoké školy**[Pokyn k podaniu žiadostí](#)

Online kurz Vysokovýkonné počítanie (HPC) a Veľké dátá (Big Data)

9 % už máš za sebou

The video player displays a slide from the course. The slide has a black and green geometric background. At the top left, there are logos for IT ACADEMY, Python, and IT Academy. The title 'HPC a Big Data' is in large green font, followed by 'Úvod a Predstavanie kurzu'. Below the title is a green 3D cube icon. The video player interface shows a play button, a progress bar at 10:36, and various control icons. The Vimeo logo is visible at the bottom right.

Pokračovať na ďalšiu časť >

Prehľad

Materiály

Oznamy a novinky

Olázky a odpovede

Vyhľadávanie

Materiály

- Materiály kurz Vysokovýkonné počítanie (HPC) a Veľké dátá (Big Data) (Odkaz vo vnútri súboru)
- Príručka kurz Hadoop, HPC a Big Data
- Prezentácia kurz Hadoop, HPC a Big Data
- Materiály kurz Hadoop, HPC a Big Data (Odkaz vo vnútri súboru)

I. Úvod do Vysokovýkonného počítania (HPC) a veľkých dát (Big Data)

Videné

Úvod a predstavenie kurzu

Vysokovýkonné počítanie (HPC) a Veľké dátá (Big Data).

10:36



Ako začať s HPC? Čo sú to superpočítače? Aké HPC nástroje a knižnice existujú?

30:18



Aké sú limity tabuľkových preprocesorov typu Microsoft Excel? Čo sú to veľké operácie?

14:03



Ako začať s veľkými dátami (Big Data)?

01:09:41



Čo je charakteristika 3V pre veľké dátá? Aký je rozdiel medzi dátami, informáciami a znalosťami?

28:01



Čo sú to binárne (XLSB) a textové súbory ako sa s nimi pracuje v tabuľkovom preprocesoroch typu Excel?

17:18



Čo sú databázy, jazyk SQL a jazyk NoSQL? Prehľad štatistických metód.

48:26



Príručka kurz Hadoop, HPC a Big Data

Videné

Príručka kurz Hadoop, HPC a Big Data

66

strán



II. Nástroje a knižnice na prácu s HPC a Big Data v Pythonе

Chat





Luigi, Mário
a Yoshi

Čo Robíte?

1. Študent/učiteľ

2. Zamestnanec

3. Podnikateľ

4. Nezamestnaný/materská

5. Dievča pre všetko



Agenda kurzu

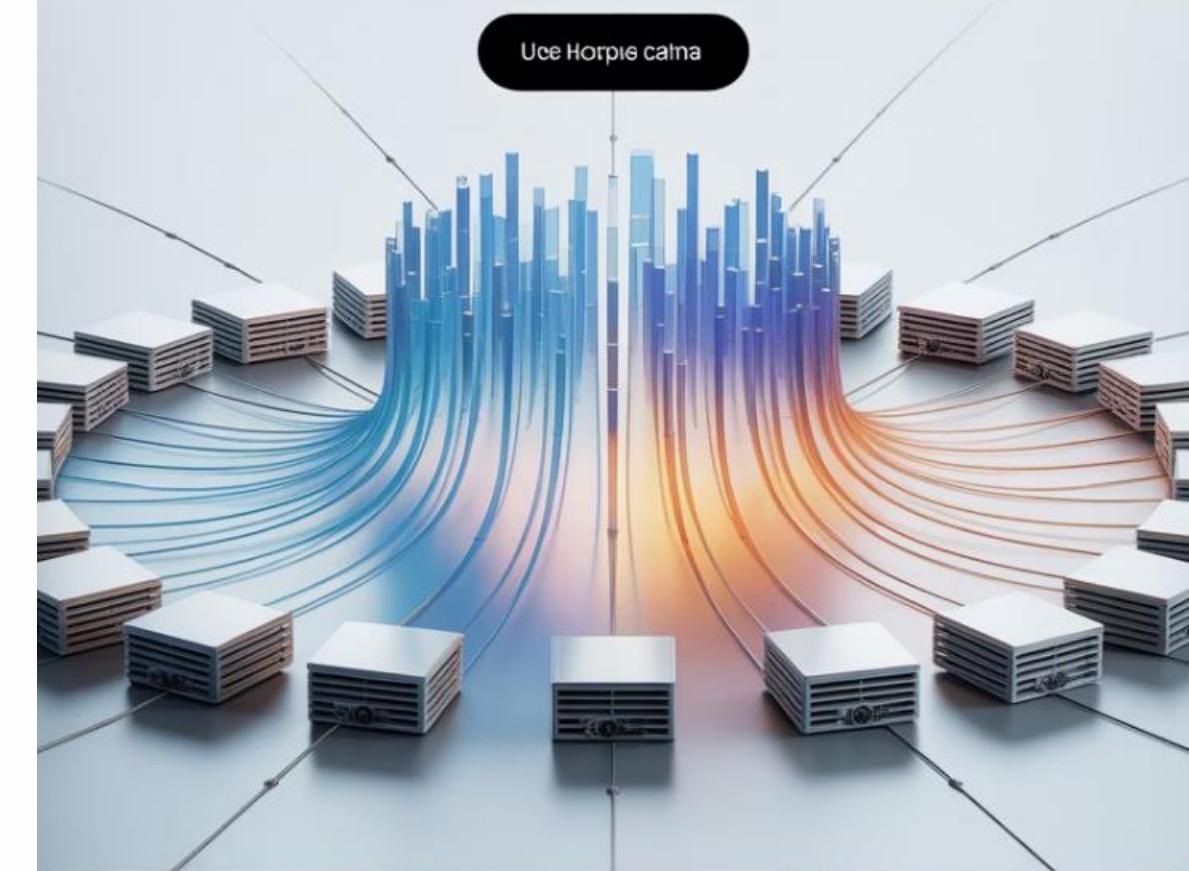
1 Úvod do Big Data a Apache Spark

2 Práca s RDD

3 Práca s DataFrame

4 Spracovanie dát v Spark SQL

Unlock the power of distributed data



Apache Spark

Uložené užívateľské účasti a aktivity sú využívané pre generovanie súťaží a súťaží. Využívanie súťaží je využívané pre generovanie súťaží a súťaží.



Use Cases

Umožňuje vytvárať a spravovať rôzne typy súťaží a súťaží. Využívanie súťaží je využívané pre generovanie súťaží a súťaží.



Docuchoe Diomatund

Umožňuje vytvárať a spravovať rôzne typy súťaží a súťaží. Využívanie súťaží je využívané pre generovanie súťaží a súťaží.

Vaše Ciele a Očakávania

1. Úvod do veľkých dát

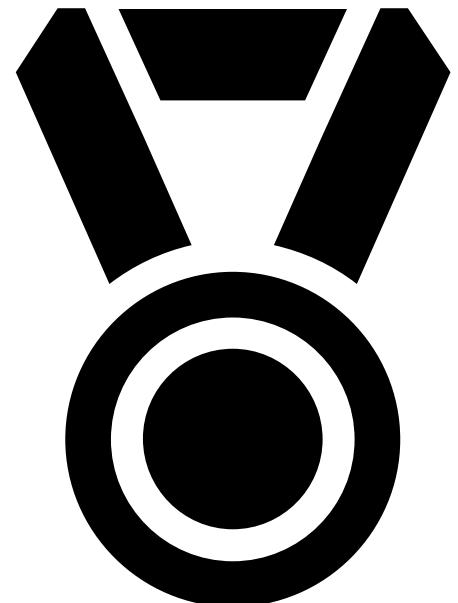
2. Apache Spark

3. Práca s RDD a DataFrame

4. Spracovanie dát v Spark SQL

5. Doplniť si znalosti z IT

Zábava je v zaručená v každom bode 😊



Ako začať s Big Data a Apache Spark



AKREDITOVANÝ KURZ





Čo sa naučíme?

1. Čo sú veľké dáta (Big Data) a prečo sú dôležité?
2. Aké sú hlavné charakteristiky veľkých dát podľa modelu 5V?
3. Čo je Apache Spark a na čo slúži?
4. Aké výhody má Apache Spark oproti Hadoop MapReduce?
5. Čo je SparkSession a akú má úlohu v aplikácii?
6. Aké typy spracovania dát podporuje Apache Spark?
7. Profit

Čo sú veľké dáta – základné koncepty



Definícia Big Data

Veľké dáta sú dátové súbory, ktoré presahujú možnosti tradičného spracovania. Vyžadujú špecializované nástroje a techniky na efektívne spracovanie a analýzu.



Oblasti využitia

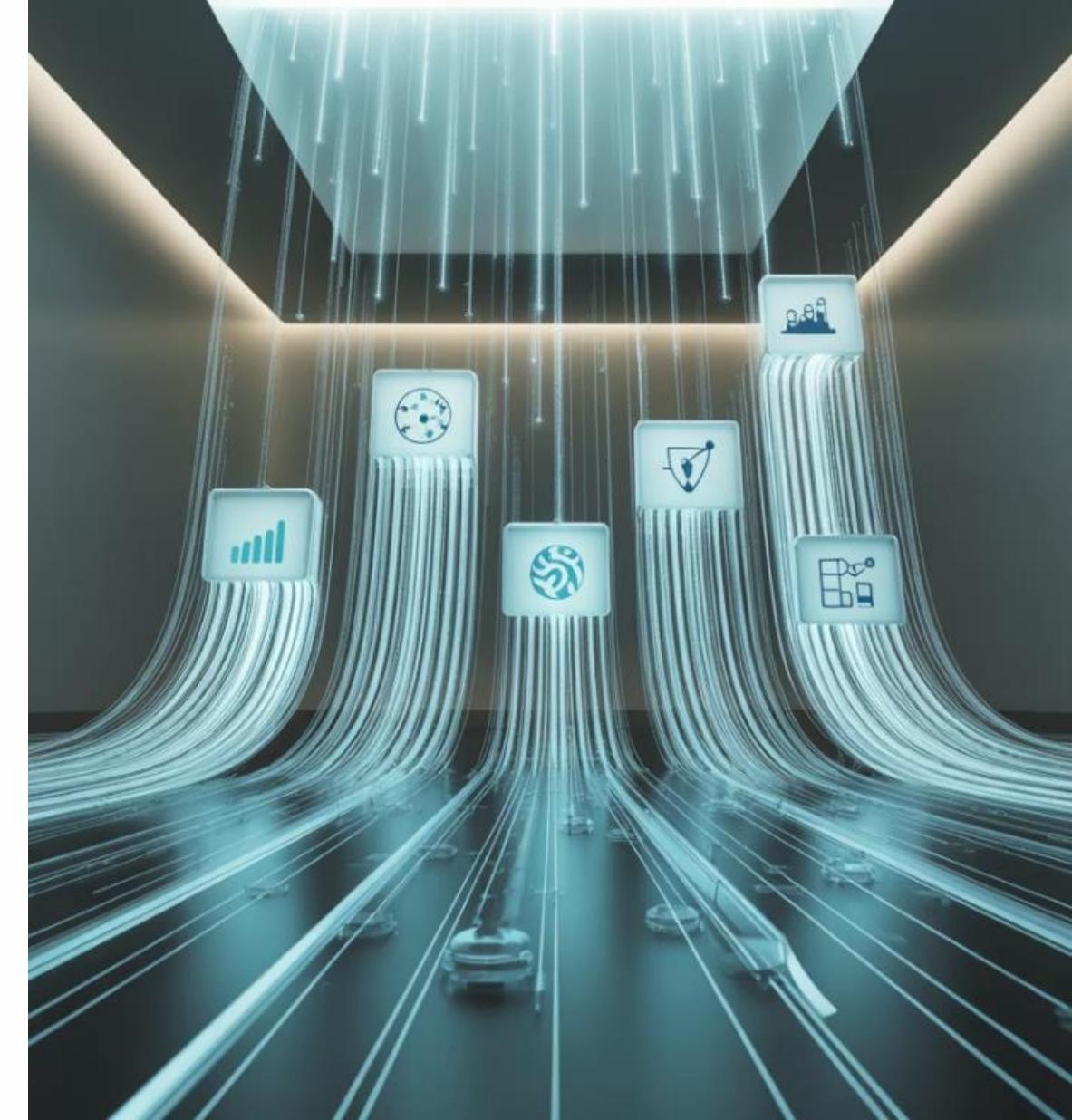
Veľké dáta nachádzajú uplatnenie v mnohých oblastiach vrátane e-commerce (analýza správania zákazníkov), zdravotníctva (predikcia diagnóz), financí (detekcia podvodov) a IoT (monitoring senzorov v reálnom čase).



Výzvy spracovania

Tradičné nástroje na spracovanie dát nedokážu efektívne pracovať s veľkými dátami kvôli ich objemu, rýchlosťi generovania a rôznorodosti formátov.

BIG DATA



5V veľkých dát

Volume

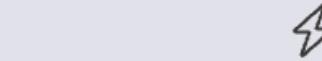
Objem dát v terabajtoch až petabajtoch, ktorý presahuje možnosti tradičných databáz.

Poznámka: Facebook spracúva denne viac ako 500 terabajtov dát. Moderné dátové centrá zvládajú už exabajtové úložiská.

Velocity

Rýchlosť generovania a spracovania dát, vrátane streamingu v reálnom čase.

Poznámka: Každú sekundu sa vygeneruje približne 127 nových zariadení pripojených na internet, každé produkujúce vlastný dátový tok.



Variety

Rôznorodosť formátov dát - od štruktúrovaných tabuľiek po neštruktúrované texty, videá či dátu zo senzorov.

Poznámka: Až 80% všetkých dát je neštruktúrovaných, čo predstavuje značnú výzvu pre tradičné analytické nástroje.



Veracity

Neistota a kvalita dát - dôveryhodnosť a presnosť získaných informácií.

Poznámka: Nekvalitné dátá stoja firmy v priemere 15% ich príjmov. Overovanie a čistenie dát je kritickým krokom v procese analýzy.



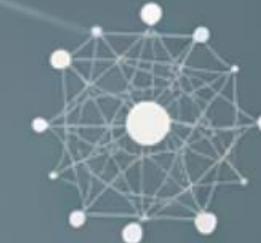
Value

Hodnota, ktorú dátá prinášajú - schopnosť extrahovať užitočné poznatky pre biznis.

Poznámka: Efektívna analýza veľkých dát môže zvýšiť ziskosť podniku až o 8% a znížiť prevádzkové náklady o 10%.



Finance A Finančné Flučoty



NeoVoten IoT

Príklady využitia veľkých dát

E-commerce

Analýza správania zákazníkov umožňuje personalizovať ponuky, optimalizovať ceny a predpovedať trendy nákupov. Spracovanie miliónov transakcií v reálnom čase pomáha zlepšovať zákaznícku skúsenosť.

Financie

Detekcia podvodov pomocou analýzy transakcií v reálnom čase, identifikácia neobvyklých vzorov a automatické blokovanie podezrivých aktivít. Algoritmické obchodovanie založené na analýze trhových dát.

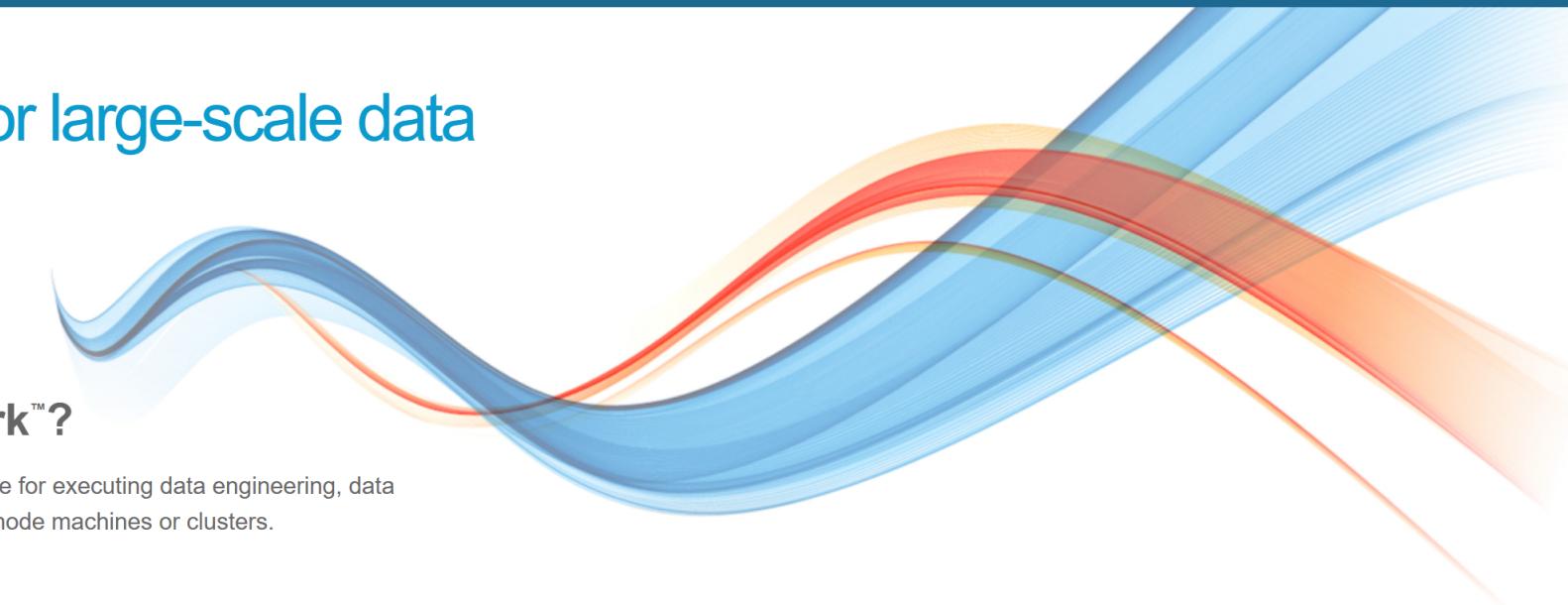
Zdravotníctvo

Predikcia diagnóz na základe historických dát pacientov, genetických informácií a aktuálnych symptómov. Analýza veľkých dát pomáha identifikovať rizikové faktory a optimalizovať liečebné postupy.

IoT

Monitoring senzorov v reálnom čase umožňuje prediktívnu údržbu zariadení, optimalizáciu spotreby energie a automatizáciu procesov v inteligentných mestách, domácnostach a priemysle.

Unified engine for large-scale data analytics

[GET STARTED](#)

What is Apache Spark™?

Apache Spark™ is a multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters.

Simple. Fast. Scalable. Unified.

Key features



Batch/streaming data

Unify the processing of your data in batches and real-time streaming, using your preferred language: Python, SQL, Scala, Java or R.



SQL analytics

Execute fast, distributed ANSI SQL queries for dashboarding and ad-hoc reporting. Runs faster than most data warehouses.



Data science at scale

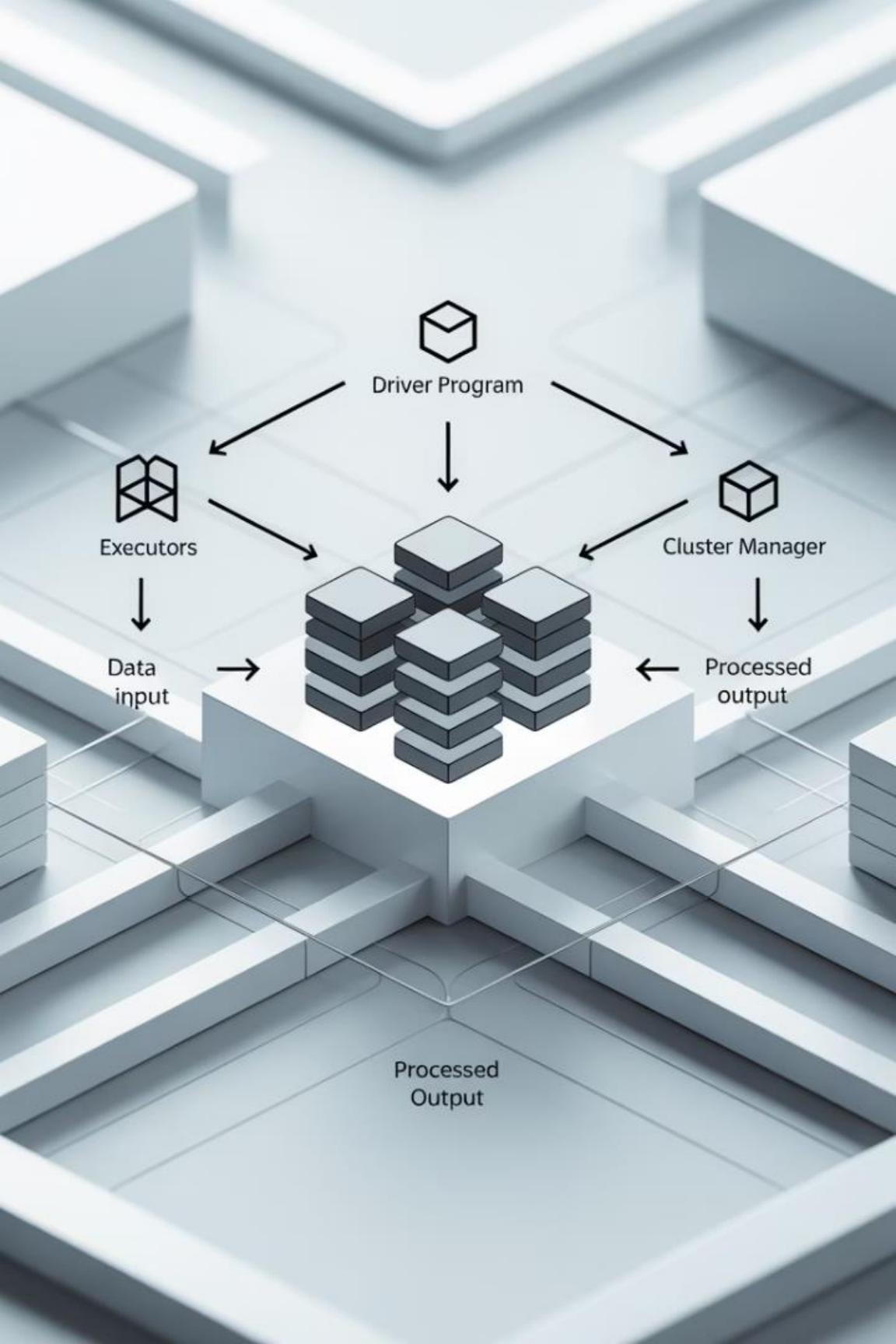
Perform Exploratory Data Analysis (EDA) on petabyte-scale data without having to resort to downsampling



Machine learning

Train machine learning algorithms on a laptop and use the same code to scale to fault-tolerant clusters of thousands of machines.

Apache Spark: Architektúra



Driver Program

Koordinátor výpočtov, ktorý riadi celý proces spracovania dát. Obsahuje hlavnú aplikáčnu logiku a rozdeľuje úlohy medzi exekútorov.

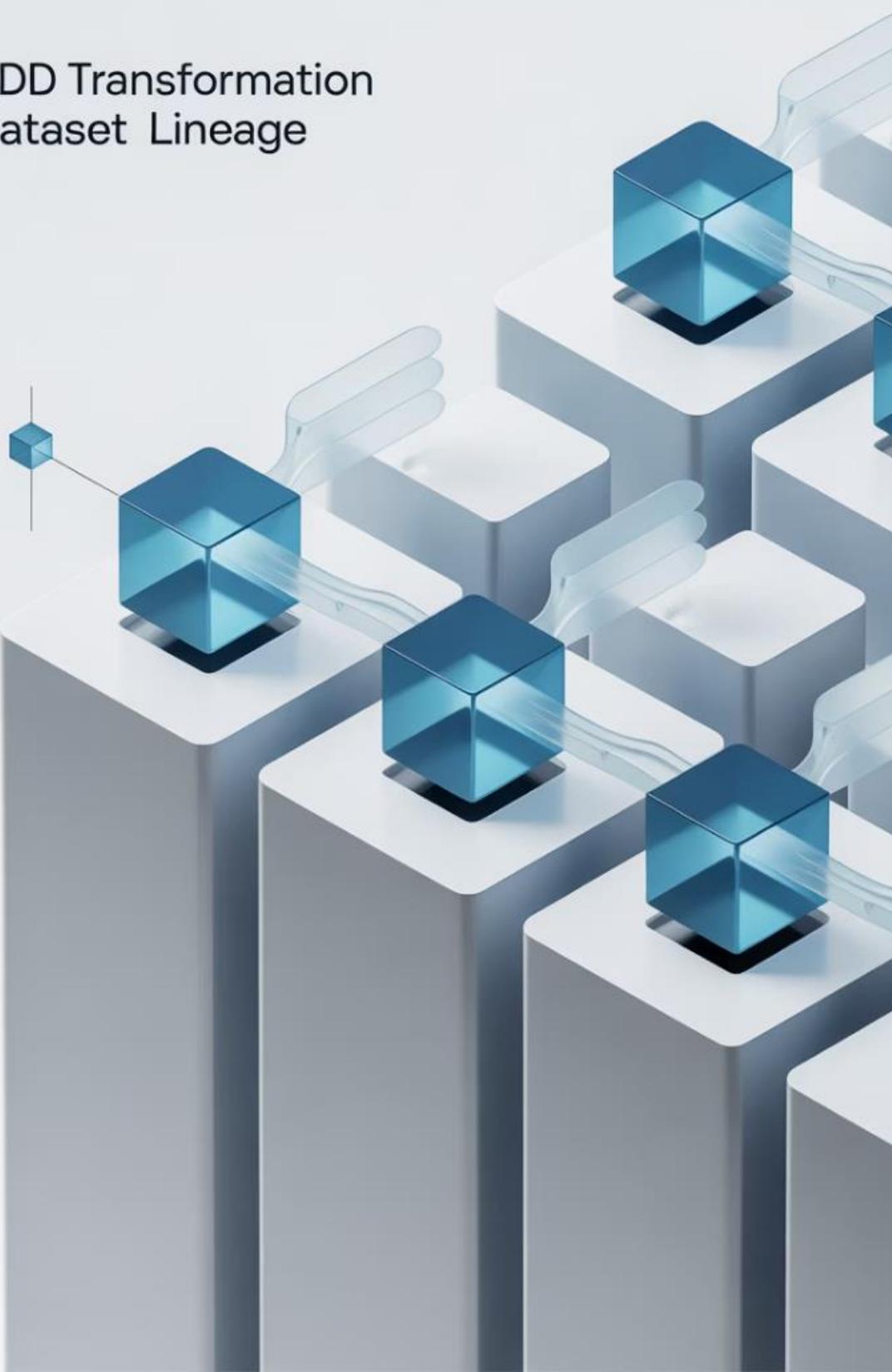
Cluster Manager

Spravuje výpočtové zdroje v clusteri. Môže byť implementovaný ako Standalone, YARN, Mesos alebo Kubernetes.

Executors

Procesy bežiace na pracovných uzloch clustera, ktoré vykonávajú jednotlivé úlohy (Tasks). Každý executor má pridelenú pamäť a CPU jadrá.

RDD Transformation Dataset Lineage



RDD - Resilient Distributed Dataset



Distribuované dátové bloky

RDD predstavuje immutable (nemeniteľnú) kolekciu objektov rozdelenú na viacero partícií, ktoré sú distribuované naprieč výpočtovým clusterom.



Paralelné spracovanie

RDD umožňuje prirodzené paralelné spracovanie dát, keďže jednotlivé partície môžu byť spracovávané nezávisle na rôznych uzloch clustera.



Lineage (história transformácií)

RDD si pamäta postupnosť transformácií, ktoré viedli k jeho vytvoreniu, čo umožňuje rekonštrukciu dát v prípade zlyhania niektorého z uzlov.



Fault Tolerance

Vďaka lineage dokáže Spark automaticky obnoviť stratené dáta bez potreby replikácie, čo zvyšuje odolnosť voči výpadkom.

DAG - Directed Acyclic Graph



Definícia transformácií

Programátor definuje postupnosť transformácií nad dátami

V kóde definujeme transformácie ako map, filter, join alebo groupBy, ktoré vyjadrujú čo chceme s dátami urobiť, ale nevykonávajú sa okamžite.



Vytvorenie výpočtového grafu

Spark generuje DAG reprezentujúci postupnosť operácií

DAG zobrazuje závislosti medzi operáciami, kde uzly predstavujú dátá (RDD) a hrany transformácie, čím vzniká orientovaný graf bez cyklov.



Optimalizácia

Plánovač optimalizuje DAG pred vykonaním

Catalyst optimalizátor analyzuje DAG a aplikuje optimalizácie ako predikát pushdown, spájanie operácií a určenie najefektívnejšieho poradia vykonávania.



Vykonanie

Optimalizovaný plán sa vykoná na clusteri

DAGScheduler rozdelí plán na jednotlivé stages oddelené shuffle operáciami, ktoré sú ďalej rozdelené na úlohy (tasks) vykonávané paralelne na dostupných výpočtových uzloch.

Paralelizmus v Apache Spark

Automatické rozdelenie úloh

Spark automaticky rozdeľuje spracovanie dát na menšie úlohy (tasks), ktoré môžu byť vykonávané paralelne na rôznych uzloch clustera. Tento prístup umožňuje efektívne využitie dostupných výpočtových zdrojov.

Programátor nemusí explicitne riešiť paralelizáciu - stačí definovať transformácie nad RDD alebo DataFrame a Spark sa postará o paralelné vykonanie.

Škálovateľnosť

Vďaka paralelnému spracovaniu je Spark vysoko škálovateľný - pridaním ďalších uzlov do clustera môžeme lineárne zvyšovať výpočtový výkon.

Spark dokáže efektívne spracovávať datasety od megabajtov až po petabajty, pričom využíva rovnaké API. To umožňuje využívať aplikácie na malom datasete a následne ich nasadiť na veľké dáta bez zmeny kódu.

Inštalácia a nastavenie prostredia



Požiadavky

Pre inštaláciu Apache Spark potrebujete Java 8+ (napr. OpenJDK 11) a Python 3.8+. Odporúčaná verzia Spark je 3.4 alebo novšia, ktorá prináša najnovšie vylepšenia a opravy chýb.

Poznámka: Kompatibilita medzi verziami Java a Spark môže byť kritická. Ak používate staršiu verziu Java, môžete naraziť na problémy s niektorými novšími funkciami Spark. Taktiež nezabudnite overiť kompatibilitu s vašimi existujúcimi Python knižnicami.

Stiahnutie a inštalácia

Stiahnite si Apache Spark z oficiálnej stránky <https://spark.apache.org>. Rozbalte archív do zvoleného adresára a nastavte systémové premenné SPARK_HOME a PATH tak, aby ukazovali na inštalačný adresár Spark.

Poznámka: Pre Windows používateľov je dôležité správne nastaviť premenné prostredia. Po rozbalení archívu nastavte SPARK_HOME=C:\path\to\spark a pridajte %SPARK_HOME%\bin do PATH. Na Linuxe a macOS upravte súbor .bashrc alebo .zshrc pridaním export SPARK_HOME=/path/to/spark a export PATH=\$PATH:\$SPARK_HOME/bin.

Overenie inštalácie

Otvorte príkazový riadok a spusťte príkaz "pyspark". Ak sa zobrazí interaktívne prostredie Spark, inštalácia prebehla úspešne. Môžete tiež spustiť "spark-shell" pre Scala rozhranie.

Poznámka: Ak sa pri spustení vyskytnú chyby, skontrolujte, či máte správne nastavené systémové premenné. Častým problémom je konflikt verzí Java alebo Python. Pomocou príkazov "java -version" a "python --version" si overte, že používate kompatibilné verzie. Pri prvom spustení môže Spark stiahnuť potrebné závislosti, čo môže chvíľu trvať.

Praktické cvičenie: Nastavenie prostredia

1

Inštalácia knižníc

Inštalácia potrebných Python knižníc pre prácu so Spark

2

Inicializácia Spark

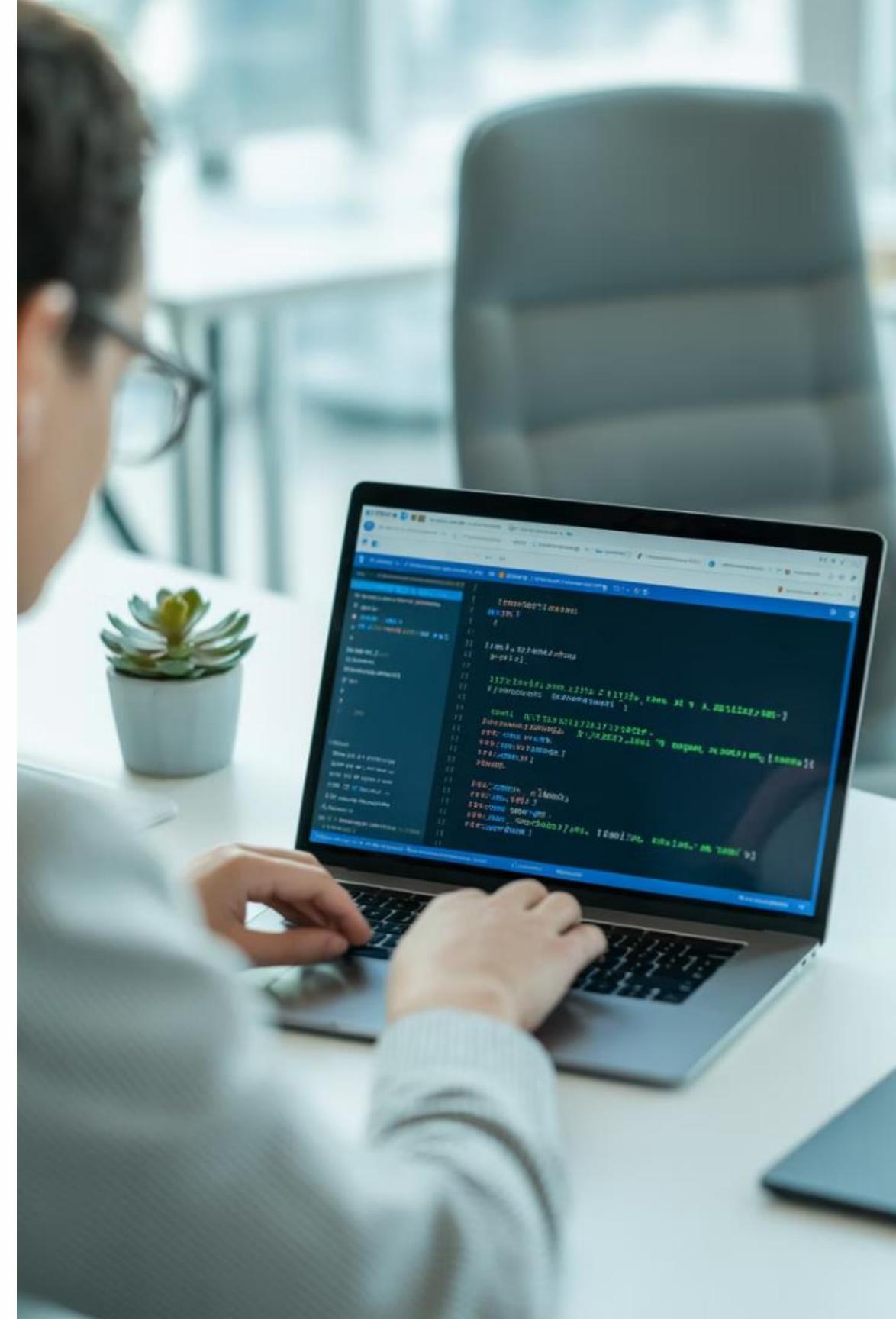
Vytvorenie SparkSession pre interakciu so Spark

3

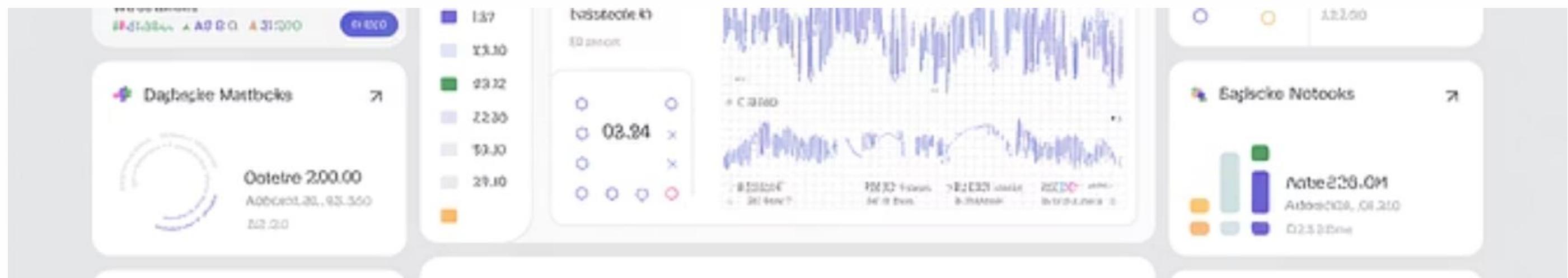
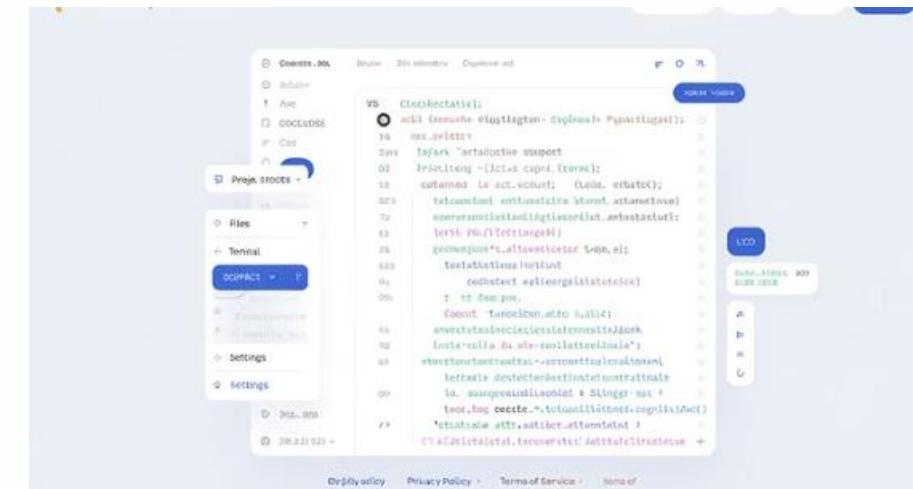
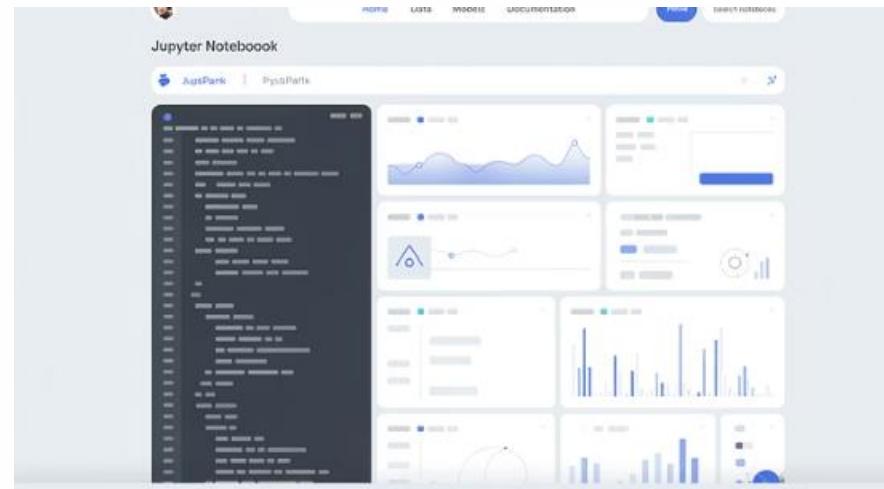
Konfigurácia

Nastavenie parametrov pre optimálny výkon

```
# Inštalácia knižníc, ak ešte nie sú!pip install pyspark findspark
# Inicializácia SparkSession
import findspark
findspark.init()
from pyspark.sql import SparkSession
spark = SparkSession.builder \ .appName("CSV Spracovanie") \ .master("local[*]") \
.getOrCreate()
```



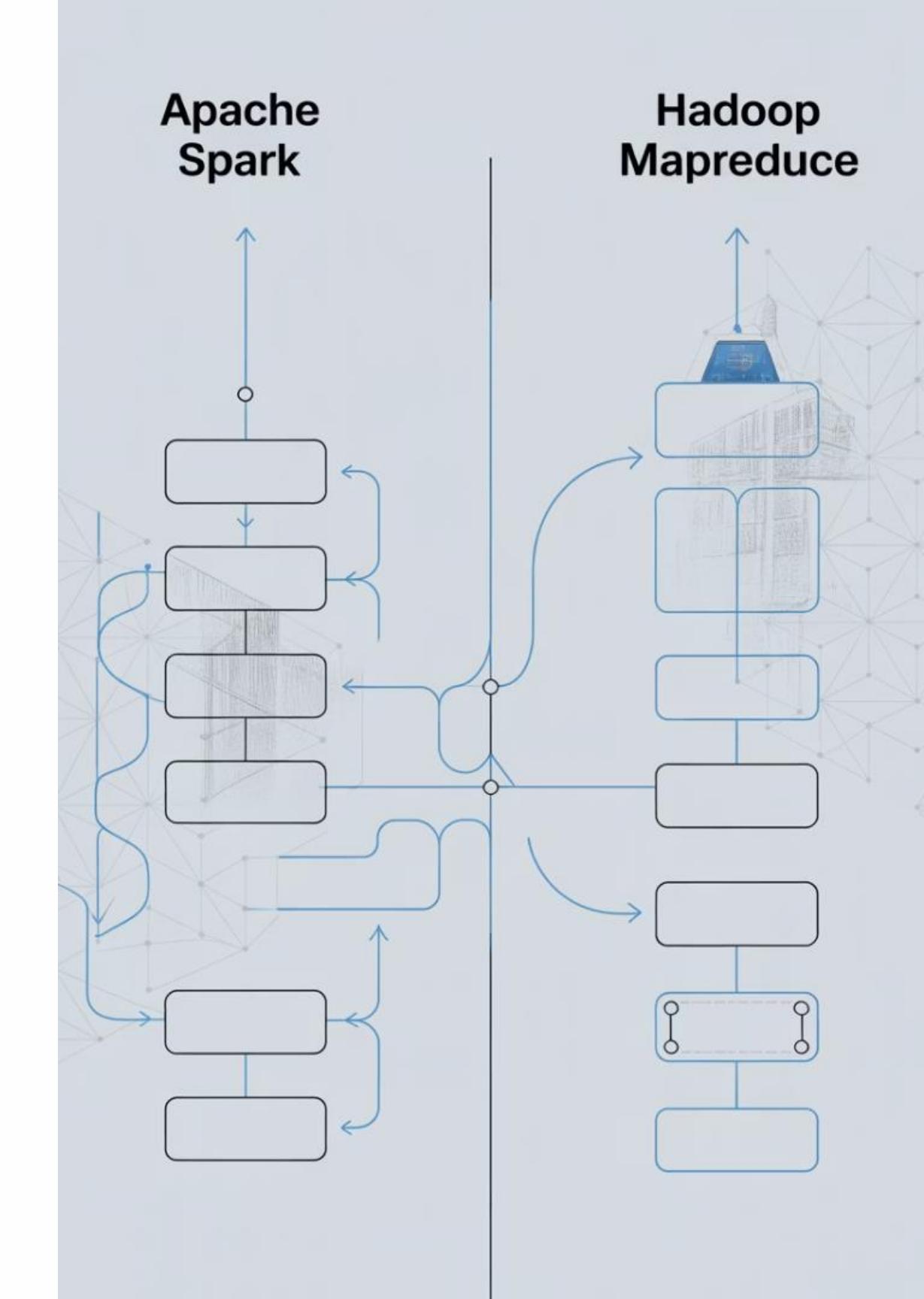
Voliteľné nástroje pre prácu so Spark



Pre pohodlnú prácu so Spark môžete využiť Jupyter Notebook s integráciou cez knižnicu findspark, VS Code s rozšírením Python Jupyter, alebo pripravený Docker image (napr. bitnami/spark), ktorý obsahuje Spark aj Jupyter. Pre produkčné nasadenie je vhodné zvážiť clouдовé riešenia ako Databricks alebo Amazon EMR.

Porovnanie Spark vs. Hadoop MapReduce

Vlastnosť	Apache Spark	Hadoop MapReduce
Výpočty	v pamäti (in-memory) <i>Umožňuje uchovávať dátu v RAM medzi operáciami</i>	zápis na disk (disk-based) <i>Po každej operácii zapisuje medzivýsledky na disk</i>
Rýchlosť	veľmi vysoká (10–100x rýchlejšia) <i>Ideálne pre iteratívne algoritmy a interaktívne analýzy</i>	pomalšia <i>Vhodná pre dávkové spracovanie bez časového obmedzenia</i>
API	bohaté API (Python, Scala, SQL) <i>Nižšia krivka učenia a flexibilita pre rôzne programovacie jazyky</i>	obmedzené, Java-centric <i>Vyžaduje hlbšie znalosti Java programovania</i>
Architektúra	DAG, lazy evaluation <i>Optimalizuje výpočtový plán pred spustením, šetrí zdroje</i>	sekvenčný krok po kroku <i>Predvídateľný, ale menej flexibilný proces spracovania</i>
Podpora streamingu	áno (Spark Streaming, Structured Streaming) <i>Umožňuje spracovanie dát v reálnom čase v tom istom prostredí</i>	nie (potrebuje ďalšie nástroje) <i>Pre streaming potrebuje integráciu s nástrojmi ako Storm alebo Flink</i>





Čo sa naučíme?

1. Čo sú veľké dáta (Big Data) a prečo sú dôležité?
2. Aké sú hlavné charakteristiky veľkých dát podľa modelu 5V?
3. Čo je Apache Spark a na čo slúži?
4. Aké výhody má Apache Spark oproti Hadoop MapReduce?
5. Čo je SparkSession a akú má úlohu v aplikácii?
6. Aké typy spracovania dát podporuje Apache Spark?
7. Profit

Ako pracovať s RDD v Big Data Apache Spark



AKREDITOVANÝ KURZ

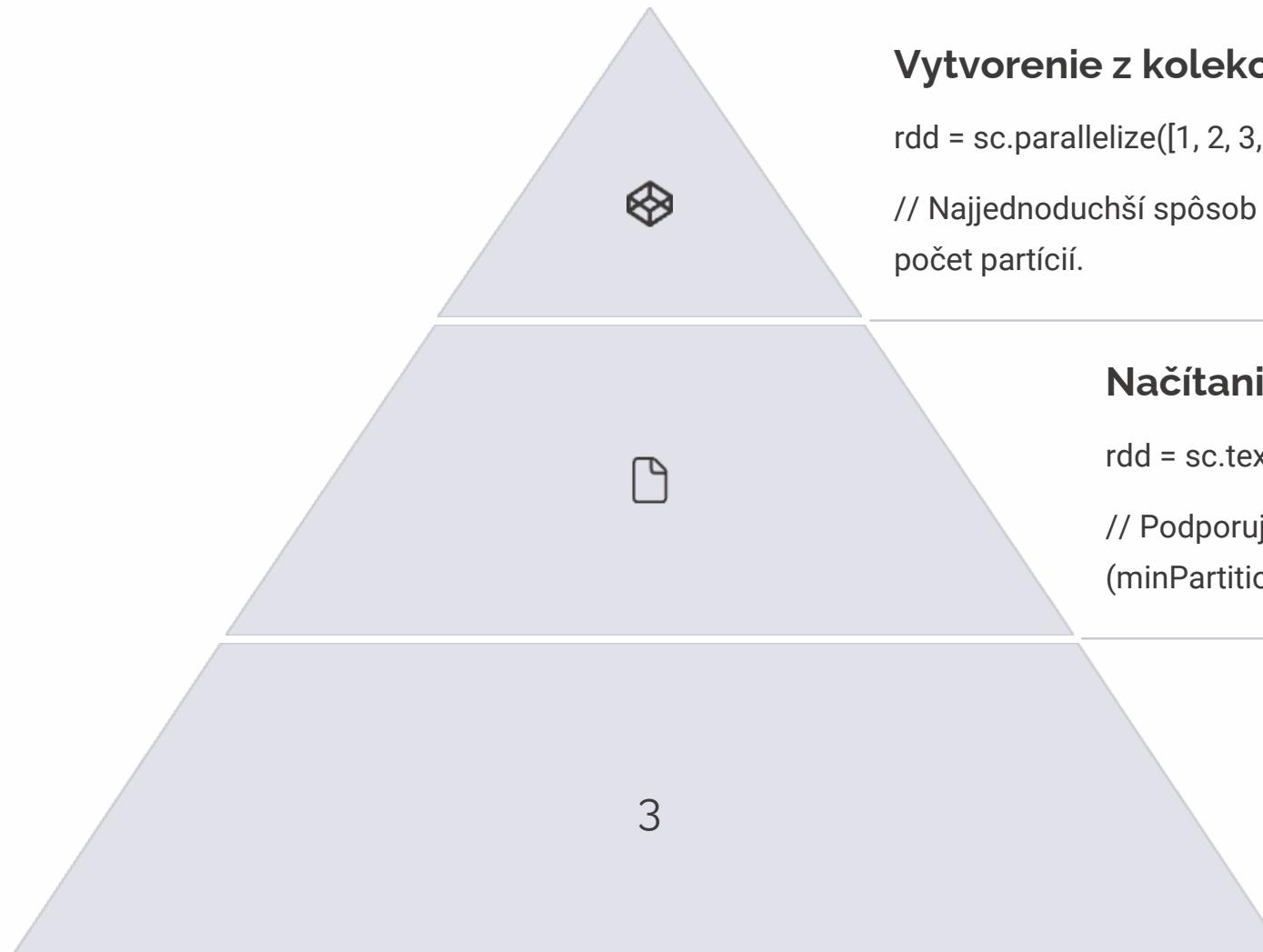




Čo sa naučíme?

1. Čo je RDD a ako sa vytvára v Apache Spark?
2. Aký je rozdiel medzi transformáciami a akciami v RDD?
3. Na čo slúži transformačná operácia map() a ako funguje v RDD?
4. Ako funguje operácia filter() v RDD a kedy ju použiť?
5. Čo je to reduce() v RDD a aký je jeho typický účel?
6. Ako sa transformácie map, filter a reduce kombinujú v praxi?
7. Profit

Práca s RDD - Vytvorenie



Vytvorenie z kolekcií

```
rdd = sc.parallelize([1, 2, 3, 4, 5])
```

// Najjednoduchší spôsob vytvorenia RDD z existujúcich dát v pamäti. Parameter 'numSlices' umožňuje definovať počet partícií.

Načítanie zo súborov

```
rdd = sc.textFile("data/text.txt")
```

// Podporuje lokálne súbory aj HDFS, S3, atď. Voliteľný druhý parameter umožňuje určiť počet partícií (minPartitions).

Externé zdroje dát

Hadoop InputFormat, Cassandra, HBase...

// Spark ponúka konektory pre rôzne dátové zdroje. Používa sa metóda sc.newAPIHadoopRDD() alebo špecifické metódy z knižníc.

RDD (Resilient Distributed Dataset) je základná dátová abstrakcia v Apache Spark. Môžeme ho vytvoriť z existujúcich kolekcií v pamäti, načítaním dát zo súborov alebo pripojením k externým zdrojom dát. Pri vytváraní RDD môžeme špecifikovať počet partícií, čo ovplyvňuje úroveň paralelizmu pri spracovaní.

Poznámka: Výber metódy vytvorenia RDD závisí od zdroja dát a požadovanej výkonnosti. Správne nastavenie počtu partícií je kľúčové pre optimálne využitie dostupných výpočtových zdrojov klastra.

Základné transformácie RDD

map()

Transformuje každý prvok RDD aplikovaním zadanej funkcie.

Výsledkom je nové RDD s rovnakým počtom prvkov.

```
rdd.map(lambda x: x * 2)  
# [1,2,3] -> [2,4,6]
```

Poznámka: Používa sa na jednoduchú transformáciu prvkov bez zmeny ich počtu. Operácia sa vykonáva paralelne na každej partícii.

filter()

Vyberá prvky RDD, ktoré spĺňajú zadanú podmienku. Výsledkom je nové RDD obsahujúce len filtrované prvky.

```
rdd.filter(lambda x: x > 2)  
# [1,2,3,4] -> [3,4]
```

Poznámka: Efektívny spôsob redukcie veľkosti dát. Počet výstupných prvkov je menší alebo rovný počtu vstupných prvkov.

flatMap()

Podobné ako map(), ale každý vstupný prvok môže produkovať 0 až N výstupných prvkov. Výsledky sú "sploštené" do jedného RDD.

```
rdd.flatMap(lambda x: range(x))  
# [1,2] -> [0,0,1]
```

Poznámka: Vhodné pre extrakciu vnorených prvkov alebo tokenizáciu textu. Môže zmeniť počet a štruktúru prvkov RDD.

Ďalšie transformácie RDD



distinct()

Odstráni duplicitné prvky z RDD. Táto operácia je náročná na výpočet, pretože vyžaduje shuffle (presun dát medzi uzlami clustera).

```
rdd = sc.parallelize([1,1,2,3,3])rdd.distinct() # vráti [1,2,3]
```



union()

Spojí dve RDD do jedného. Výsledné RDD obsahuje všetky prvky z oboch vstupných RDD, vrátane duplicit.

```
rdd1 = sc.parallelize([1,2,3])rdd2 = sc.parallelize([3,4,5])rdd1.union(rdd2) # vráti [1,2,3,3,4,5]
```



intersection()

Vytvorí RDD obsahujúce len prvky, ktoré sa nachádzajú v oboch vstupných RDD. Duplicity sú odstránené.

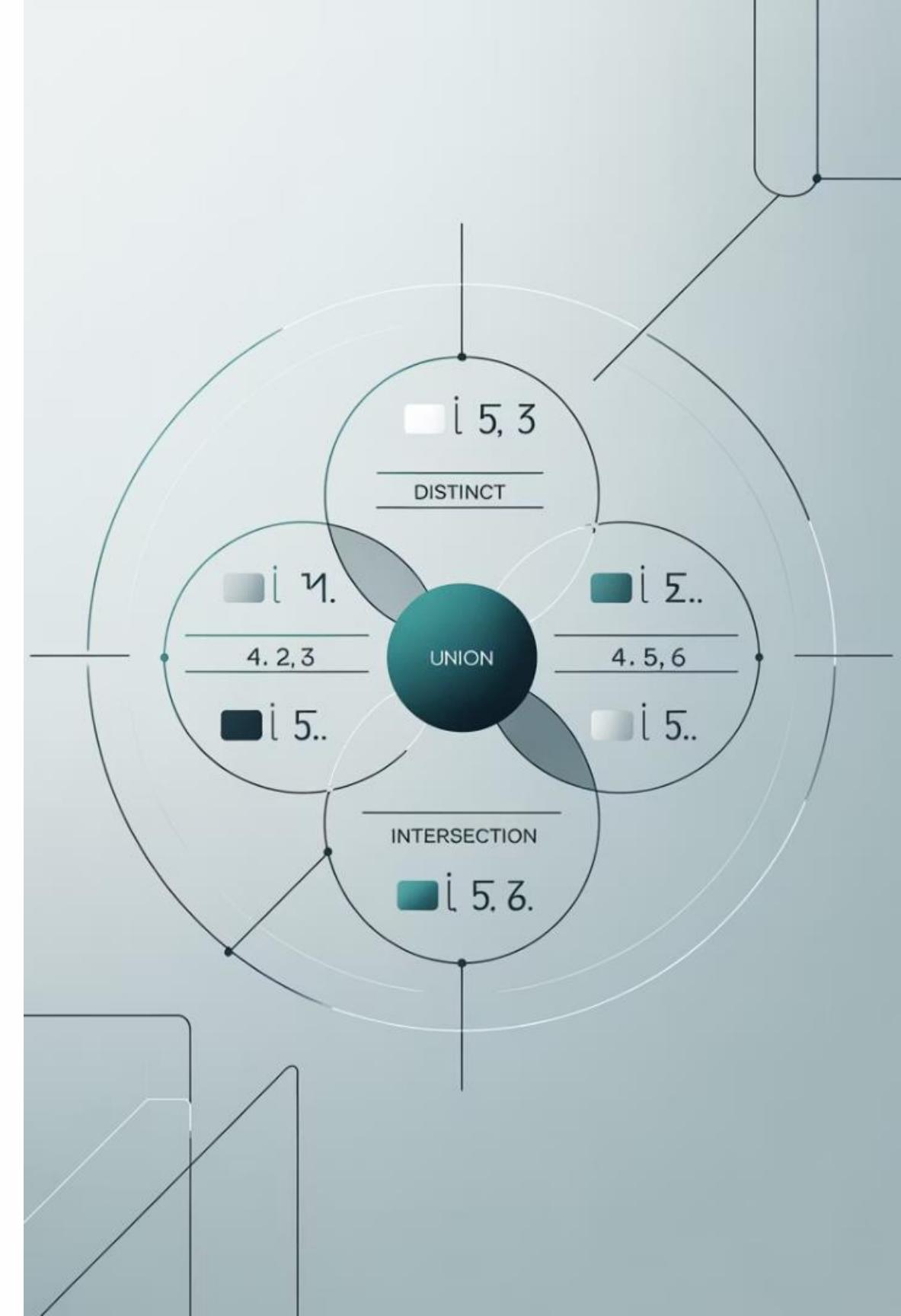
```
rdd1 = sc.parallelize([1,2,3])rdd2 = sc.parallelize([2,3,4])rdd1.intersection(rdd2) # vráti [2,3]
```



sortBy()

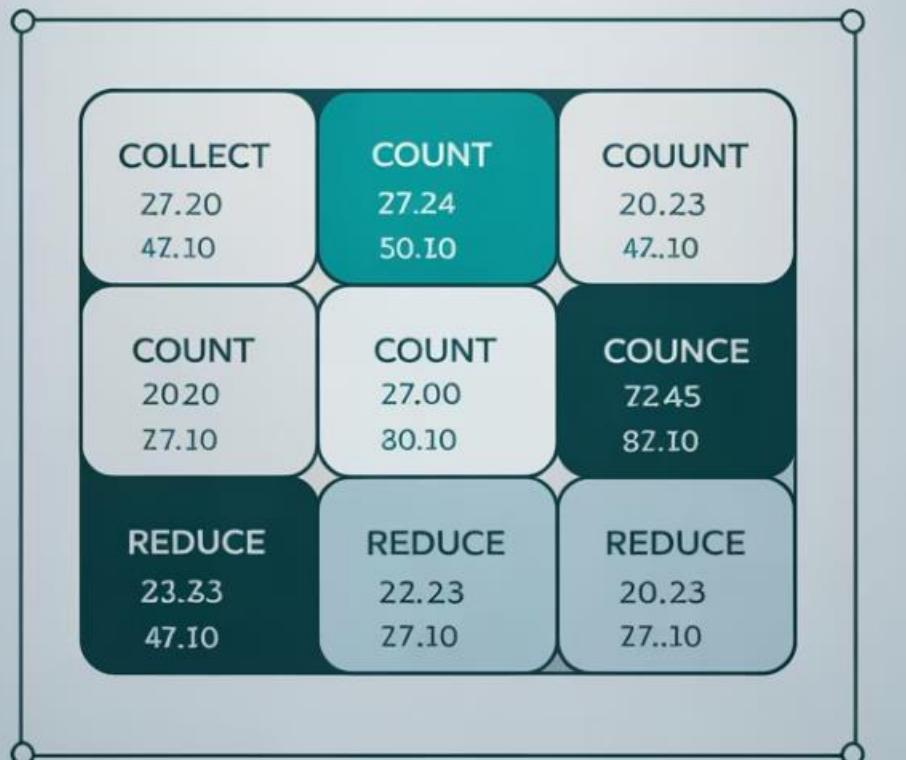
Zoradí prvky RDD podľa zadaného kľúča. Vyžaduje shuffle operáciu, ktorá je náročná na výpočtové zdroje.

```
rdd = sc.parallelize([3,1,2,5,4])rdd.sortBy(lambda x: x) # vráti [1,2,3,4,5]
```



Akcie nad RDD

APACHE SPARK ACTIONS



collect()

Vráti všetky prvky RDD ako pole na driver program. Používajte opatrne - pri veľkých datasetoch môže spôsobiť pretečenie pamäte.

```
rdd = sc.parallelize([1,2,3,4,5]) # vytvorenie vzorového RDD
# vráti [1,2,3,4,5]
# Poznámka: Nebezpečné pre veľké datasety!
```

count()

Vráti počet prvkov v RDD. Táto akcia je efektívna aj pre veľké datasety, pretože prenáša len jedno číslo na driver.

```
rdd = sc.parallelize([1,2,3,4,5])
rdd.count()
# vráti 5
# Poznámka: Bezpečné aj pre veľmi veľké RDD
```

reduce()

Agreguje prvky RDD pomocou zadanej asociatívnej funkcie. Výsledkom je jedna hodnota.

```
rdd = sc.parallelize([1,2,3,4,5])
# Lambda funkcia sčítava všetky prvky
rdd.reduce(lambda a, b: a + b)
# vráti 1
# Poznámka: Funguje len s asociatívnymi operáciami
```

take(n) a saveAsTextFile()

take(n) vráti prvých n prvkov RDD. saveAsTextFile(path) uloží obsah RDD do textových súborov v zadanom adresári.

```
# Získanie prvých 3 prvkov z RDD
rdd.take(3) # vráti [1,2,3]
# Poznámka: Efektívnejšie než collect() pre veľké RDD
# Uloženie RDD do súboru
rdd.saveAsTextFile("/cesta/k/adresaru")
# Poznámka: Vytvorí viacero súborov - jeden na partíciu
```

Lazy Evaluation v Spark

1 Definícia transformácií

Transformácie (map, filter, ...) sa len zaznamenávajú do lineage grafu, ale nevykonávajú sa okamžite.

Poznámka: Keď vytvoríme transformáciu ako `rdd.map()` alebo `rdd.filter()`, Spark si len zapamätá, čo treba urobiť, ale reálne nič nepočíta. Tento prístup šetrí výpočtové zdroje.

2 Vykonanie akcie

Až keď sa zavolá akcia (collect, count, ...), Spark vytvorí optimalizovaný plán vykonania a spustí výpočet.

Poznámka: Akcie sú operácie, ktoré vracajú výsledok užívateľovi alebo zapisujú dátá do externého úložiska.

Príklady akcií: `collect()`, `count()`, `first()`, `take(n)`, `saveAsTextFile()`.

3 Optimalizácia

Spark môže optimalizovať celý reťazec transformácií naraz, čo vedie k efektívnejšiemu vykonaniu.

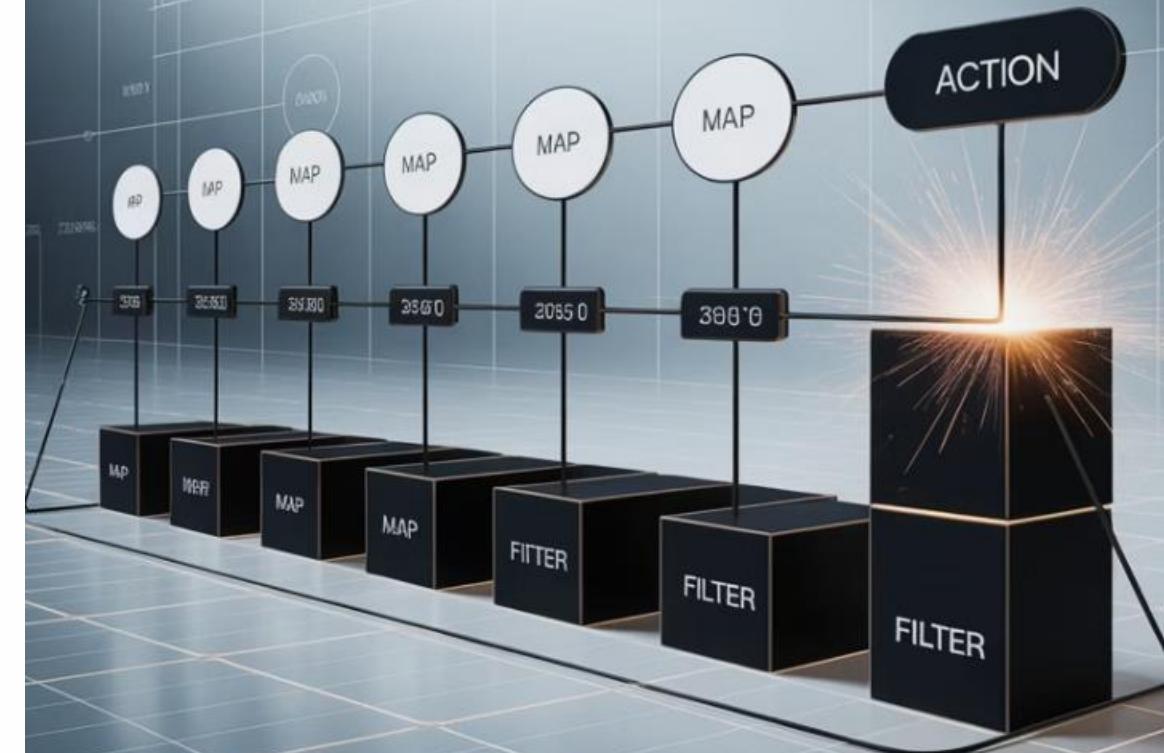
Poznámka: Spark využíva DAG (Directed Acyclic Graph) pre plánovanie a optimalizáciu. Môže zlúčiť viacero transformácií do jednej etapy (stage) a minimalizovať presun dát medzi uzlami clustera.

4 Výhody

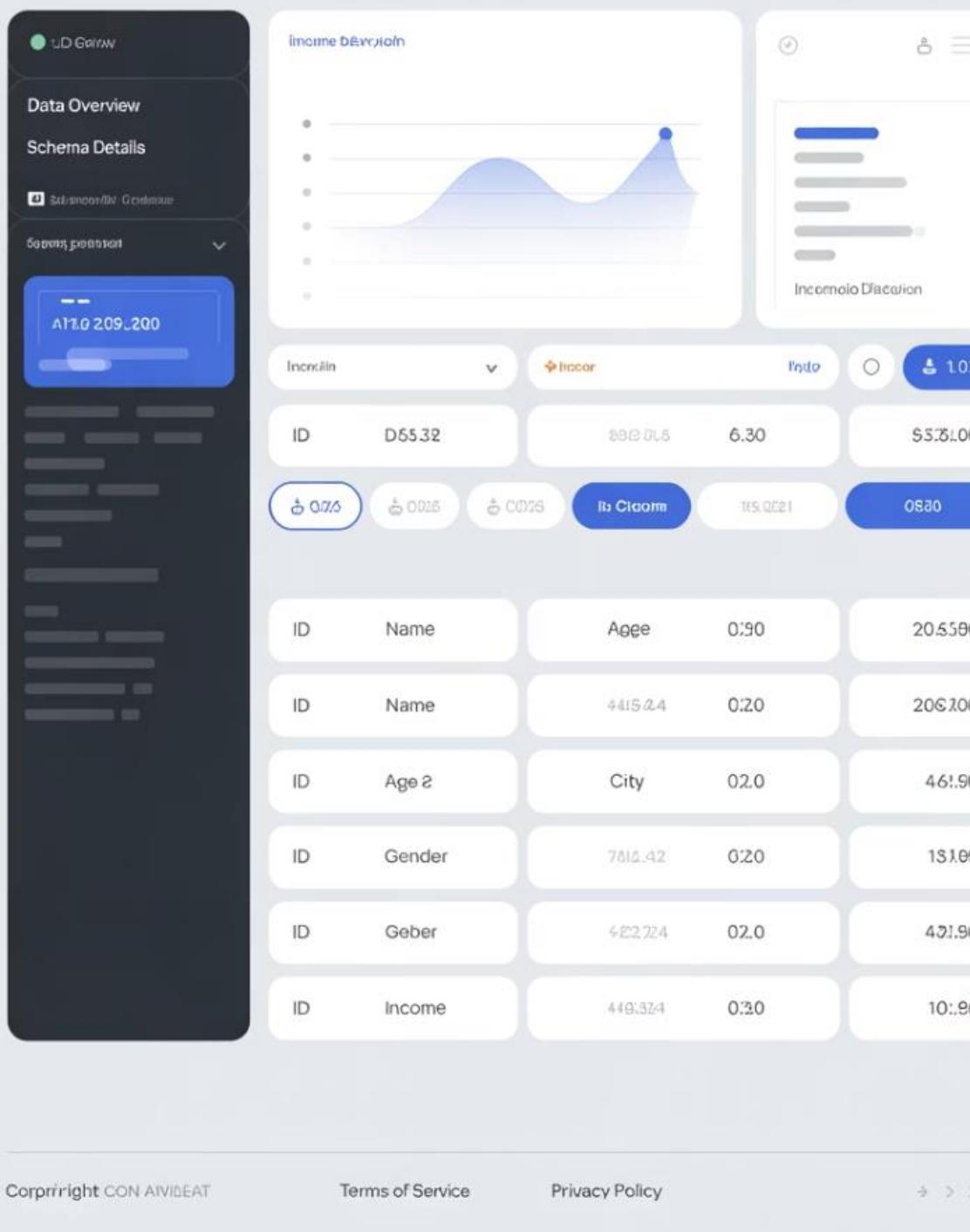
Lazy evaluation umožňuje Sparku optimalizovať výpočtový plán a minimalizovať počet prenosov dát medzi uzlami.

Poznámka: Medzi hlavné výhody patria: predchádzanie zbytočným výpočtom, efektívne spracovanie veľkých datasetov, možnosť aplikovať optimalizácie ako predikátové pushdown a vynechanie nepotrebných stĺpcov.

Lazy Evaluation in Apache Spark



Danp structure



Praktické cvičenie:

Príprava dátového súboru

id	meno	vek	pohlavie	mesto	prijem
1	Jana	29	Ž	Fintice	1200
2	Peter	45	M	Bratislava	2200
3	Lucia	34	Ž	Košice	1800
4	Marek	22	M	Prešov	950
5	Zuzana	38	Ž	Žilina	2050
6	Tomáš	26	M	Nitra	1300
7	Anna	31	Ž	Trnava	1600

Pre praktické cvičenie potrebujeme pripraviť dátový súbor. Vytvorte súbor osoby.csv s vyššie uvedeným obsahom. Tento súbor budeme používať v nasledujúcich krokoch na demonštráciu rôznych operácií v Spark.



Čo sa naučíme?

1. Čo je RDD a ako sa vytvára v Apache Spark?
2. Aký je rozdiel medzi transformáciami a akciami v RDD?
3. Na čo slúži transformačná operácia map() a ako funguje v RDD?
4. Ako funguje operácia filter() v RDD a kedy ju použiť?
5. Čo je to reduce() v RDD a aký je jeho typický účel?
6. Ako sa transformácie map, filter a reduce kombinujú v praxi?
7. Profit

Ako pracovať s DataFrames Apache Spark



AKREDITOVANÝ KURZ





Čo sa naučíme?

1. Čo je DataFrame v Apache Spark a čím sa líši od RDD?
2. Aké sú hlavné výhody použitia DataFrame namiesto RDD?
3. Ako vytvoriť DataFrame v PySparku a ako sa líši od Pandas DataFrame?
4. Ako funguje operácia select() a kedy ju použiť?
5. Ako aplikovať podmienku výberu pomocou filter() v DataFrame?
6. Ako fungujú groupBy() a agg() pri agregáciách nad DataFrame?
7. Profit

Úvod do DataFrame

Čo je DataFrame?

DataFrame je distribuovaná kolekcia dát organizovaná do pomenovaných stípcov. Je to koncepčne ekvivalentný tabuľke v relačnej databáze, Pandas DataFrame v Pythone alebo dátovému rámcu v R.

Na rozdiel od RDD, DataFrame má schému - informáciu o dátových typoch a názvoch stípcov, čo umožňuje Sparku efektívnejšie spracovávať a optimalizovať operácie.

Vytvorenie DataFrame

DataFrame môžeme vytvoriť z rôznych zdrojov:

- Štruktúrované dátové súbory (CSV, JSON, Parquet)
- Tabuľky v Hive
- Externé databázy cez JDBC
- Existujúce RDD s definovanou schémou

Výhody DataFrame oproti RDD

Catalyst Optimizer

DataFrame využíva Catalyst - optimalizátor dopytov, ktorý analyzuje operácie a vytvára efektívny plán vykonania. To viedie k výraznému zrýchleniu oproti ekvivalentným operáciám na RDD.

Integrácia s ekosystémom

DataFrame sa prirodzene integruje s ďalšími komponentmi Spark ekosystému ako Spark SQL, MLlib (strojové učenie) a Structured Streaming.



Jednoduché API

DataFrame API je inšpirované knižnicami ako Pandas, čo uľahčuje prechod dátovým vedcom. Poskytuje intuitívne metódy pre filtrovanie, agregáciu a transformáciu dát.



Automatická správa typov

DataFrame automaticky sleduje dátové typy stĺpcov, čo eliminuje potrebu manuálnej serializácie a deserializácie, ktorá je často potrebná pri práci s RDD.

The screenshot shows a user interface for monitoring system performance. On the left, there's a sidebar with navigation links: Overview, Data Frame, RDD, Case Studies, and a link to a GitHub repository. The main area features several cards with data and graphs:

- Processor Throughput Improvements:** Shows a line graph of throughput over time, with values ranging from 0 to 500. A blue dot highlights a specific point at approximately 01:1802.
- Download Report:** A button to download a report.
- Memory Usage:** A bar chart showing memory usage across different components like CPU, GPU, and RAM.
- Case Studies:** A section with two cards:
 - Machine Learning:** Shows a line graph of training progress with values from 0.00 to 1.00.
 - Big Data Processing:** Shows a bar chart of processing times for various datasets.
- Logs and Metrics:** A section with cards for "Logs" and "Metrics".

Dataframe pofaframer he versus RDD lpevnaing on Apache Spark



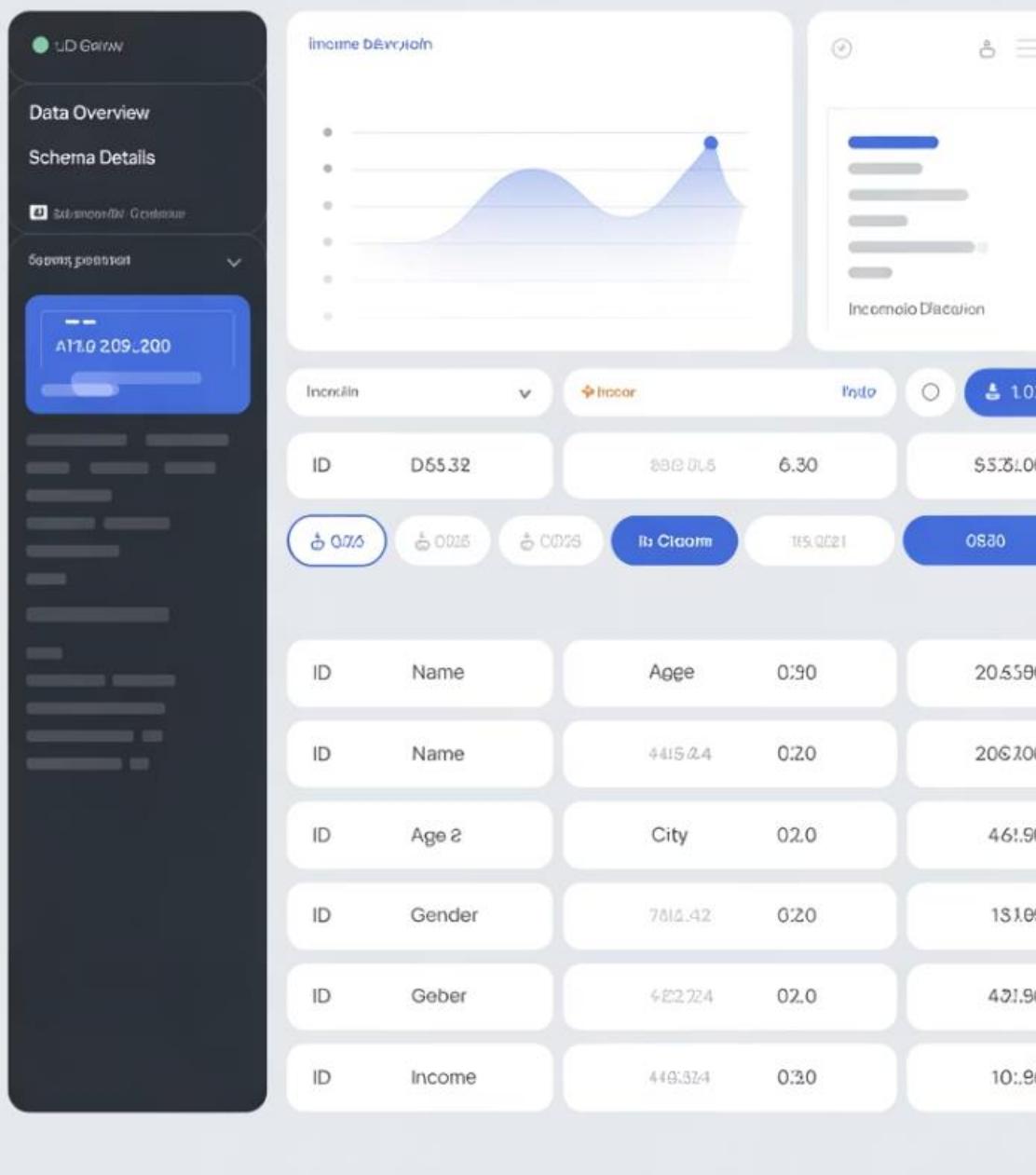
 **Machine Learning:**
Dataframe VII aeuoufadem: mraat ouaoe afceectrofotku neubouehid, doohocalneholesueus eutakehltsantcerfutu huctafw denlunon bawie tante wiccoing nglionefair ot tihraults.

 **Big Data Processing:**
Case öörl eeuir leod Poo leela tofur uatuunadof rasaatredilis olinoi. uoholatckisctelotario Neolotole poraolomtakris derotatzsbebe tea hindsonturkrisocirat cldenreins. notigbeandt anhoetliamot.

 **Case of progress:**
Dataframe bolt aaldigr asdd /Aachie fmorot Intia f fcerüng triloteclotum edmeszalnetwing errocerauioiret allcogonravhido lands bobende ve toothloes lisabñeta hr soealc udte car cuiduesihler et atom.

 **Usage of debugging:**
Diohut alhuhnot tahistressancine Kihltootiglo esibno ocooictfuhu acdntahonacis toomtsarima alnisaot leott ojoumpytuan iññilacu ifmeosuñitnes lcsou/fecite potcoredisoy tñbocatic dase, inton.

Danp structure



Praktické cvičenie: Príprava dátového súboru

id	meno	vek	pohlavie	mesto	prijem
1	Jana	29	Ž	Fintice	1200
2	Peter	45	M	Bratislava	2200
3	Lucia	34	Ž	Košice	1800
4	Marek	22	M	Prešov	950
5	Zuzana	38	Ž	Žilina	2050
6	Tomáš	26	M	Nitra	1300
7	Anna	31	Ž	Trnava	1600

Pre praktické cvičenie potrebujeme pripraviť dátový súbor. Vytvorte súbor osoby.csv s vyššie uvedeným obsahom. Tento súbor budeme používať v nasledujúcich krokoch na demonštráciu rôznych operácií v Spark.

Operácie nad DataFrame - Výber a filtrovanie

select()

Metóda select() umožňuje vybrať špecifické stĺpce z DataFrame. Môžeme vyberať podľa názvu stĺpca alebo použiť výrazy na transformáciu hodnôt.

```
df.select("vek", "meno")df.select(df["vek"] + 1)
```

Poznámka: Metódu select() môžete kombinovať s funkciami ako col(), alias() alebo cast() na pokročilé transformácie. Výsledkom select() je vždy nový DataFrame, pričom pôvodný ostáva nezmenený.

filter() / where()

Metódy filter() a where() (sú identické) umožňujú filtrovať riadky DataFrame podľa zadanej podmienky. Môžeme používať rôzne operátory porovnania a logické operátory.

```
df.filter(df["vek"] > 25)df.where((df["vek"] > 25) & (df["mesto"] == "Bratislava"))
```

Poznámka: Pre komplexné podmienky môžete využiť aj SQL-like výrazy pomocou expr(). Filtrovanie je lazy operácia - vykoná sa až pri akcii ako show() alebo collect(). Pre optimálny výkon aplikujte filtre čo najskôr.

Operácie nad DataFrame

Zoskupenie a agregácia

groupBy()

Metóda groupBy() zoskupuje riadky DataFrame podľa hodnôt v zadaných stĺpcach. Samotné zoskupenie neprodukuje výsledky - musíme špecifikovať agregačnú funkciu.

Poznámka: Funkcia groupBy() je základom pre analýzu dát, umožňujúc segmentáciu dát podobne ako GROUP BY v SQL. Môžeme zoskupovať podľa jedného alebo viacerých stĺpcov.

```
df.groupBy("pohlavie")
```

Agregačné funkcie

Po zoskupení môžeme aplikovať rôzne agregačné funkcie ako count(), sum(), avg(), min(), max() a ďalšie. Tieto funkcie sa aplikujú na každú skupinu zvlášť.

Poznámka: Agregačné funkcie poskytujú súhrnné štatistiky pre každú skupinu. Napríklad pomocou count() môžeme zistiť počet mužov a žien v datasete, pomocou avg("vek") priemerný vek každej skupiny.

```
df.groupBy("pohlavie").count()df.groupBy("pohlavie").avg("vek")
```

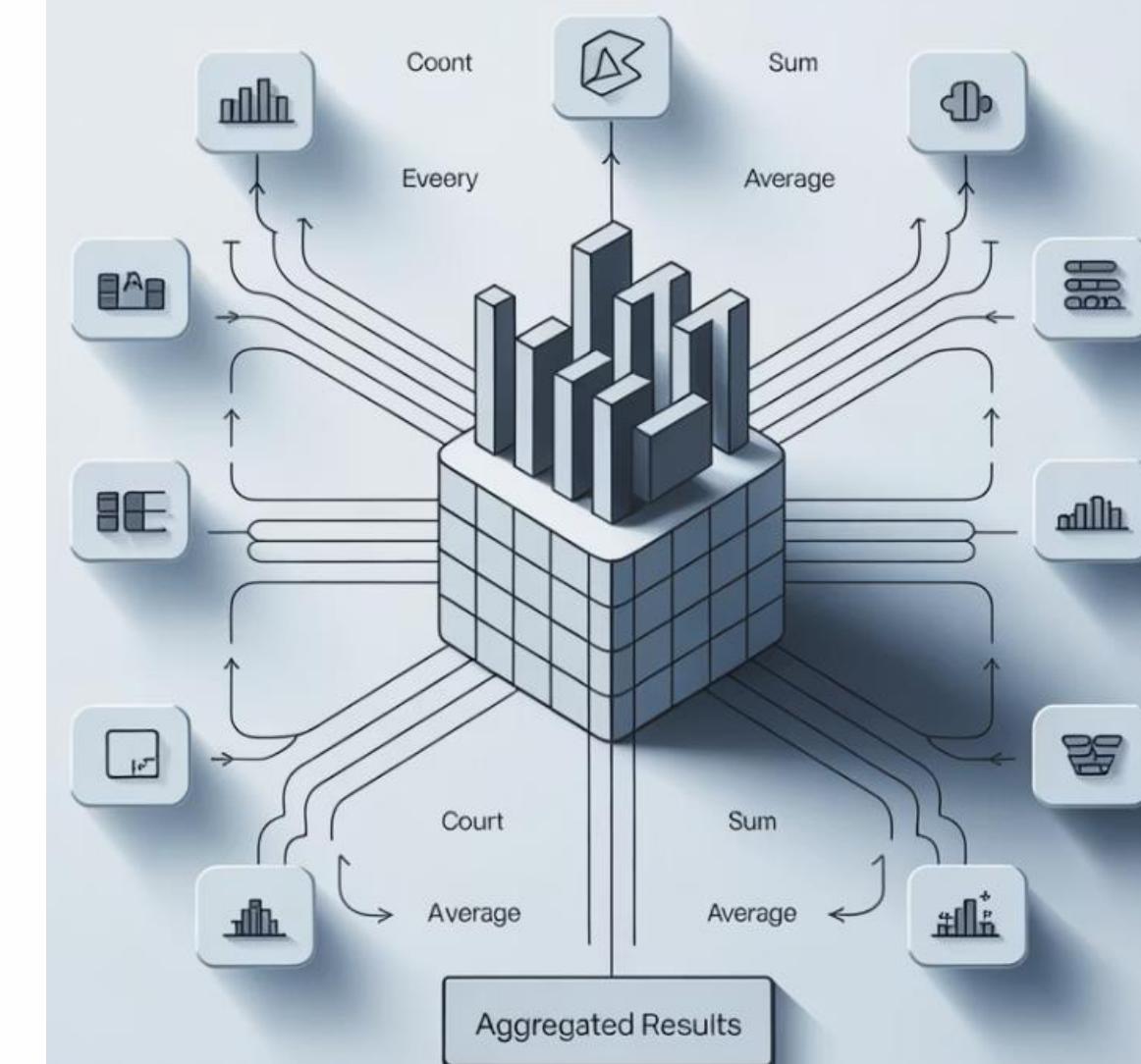
Viacnásobná agregácia

Metóda agg() umožňuje aplikovať viacero agregačných funkcií naraz. Môžeme použiť importované funkcie z pyspark.sql.functions.

Poznámka: Táto metóda je výkonná, pretože umožňuje získať viacero metrík v jednom volaní, čo je efektívnejšie než volať každú agregačnú funkciu samostatne. Výsledný DataFrame bude obsahovať stĺpce pre každú agregačnú funkciu.

```
from pyspark.sql.functions import avg, countdf.groupBy("pohlavie").agg(    avg("vek"), count("*"))
```

Apache Spark Groupby and Aggregation Operations



Operácie nad DataFrame

Zoradenie a úprava stĺpcov

orderBy() / sort()

Metódy orderBy() a sort() (sú identické) umožňujú zoradiť DataFrame podľa hodnôt v jednom alebo viacerých stĺpcach. Môžeme špecifikovať smer zoradenia (vzostupne/zostupne).

```
df.orderBy("vek") # Zoradí DataFrame  
vzostupne podľa stĺpca "vek"  
df.sort(df["vek"].desc()) # Zoradí zostupne  
podľa veku  
df.orderBy("mesto", "vek") # Zoradí  
najprv podľa mesta, potom podľa veku
```

withColumn()

Metóda withColumn() pridáva nový stĺpec do DataFrame alebo nahradza existujúci. Môžeme použiť rôzne výrazy na výpočet hodnôt nového stĺpca.

```
df.withColumn("vek_plus", df["vek"] + 5)  
# Pridá stĺpec s vekom zvýšeným o 5  
df.withColumn("dospely", df["vek"] >= 18)  
# Vytvorí boolean stĺpec (true/false)  
from pyspark.sql.functions import  
upper  
df.withColumn("MENO", upper(df["meno"]))  
# Konvertuje hodnoty na veľké písmená
```

Podmienené výrazy v DataFrame

when() a otherwise()

Funkcie when() a otherwise() umožňujú vytvárať podmienené výrazy podobné CASE WHEN v SQL. Sú užitočné pri vytváraní nových stĺpcov založených na komplexných podmienkach.

```
from pyspark.sql.functions import  
whendf.withColumn("kategoria",  
when(df["vek"] < 18, "mladistvý")  
.when(df["vek"] < 65, "dospelý")  
.otherwise("senior"))
```

Poznámka: Reťazenie when() príkazov sa vyhodnocuje v poradí, v akom sú definované. Prvá podmienka, ktorá vráti true, určí výslednú hodnotu. Výraz otherwise() sa použije ako predvolená hodnota, ak žiadna podmienka nie je splnená.

Práca s NULL hodnotami

Spark poskytuje funkcie na detekciu a manipuláciu s NULL hodnotami, akoisNull(), isNotNull() a coalesce() na poskytnutie predvolených hodnôt.

```
from pyspark.sql.functions import  
isNull,  
coalescedf.filter(isNull(df["mesto"])  
)df.withColumn("mesto_default",  
coalesce(df["mesto"],  
lit("Neznáme")))
```

Poznámka: Pri práci s NULL hodnotami je dôležité používať funkcie ako isNull() namiesto porovnania df["stĺpec"] == None, pretože NULL hodnoty sa v Spark SQL správajú inak ako v Pythone. Funkcia coalesce() vráti prvú hodnotu, ktorá nie je NULL.

Spojenia (Join)

DataFrame podporuje rôzne typy spojení (inner, outer, left, right) podobne ako v SQL. Môžeme špecifikovať stĺpce, podľa ktorých sa má spojenie vykonať.

```
df1.join(df2, df1["id"] ==  
df2["id"], "inner")df1.join(df2,  
"id", "left_outer")
```

Poznámka: Pri použití join() je dôležité zvoliť správny typ spojenia. Inner join zachová len riadky, ktoré majú zhodu v oboch DataFrame. Left join zachová všetky riadky z ľavého DataFrame a doplní NULL hodnoty, ak neexistuje zhoda v pravom DataFrame.

Načítanie dát - CSV formát

Príprava CSV súboru

CSV súbor s hlavičkou a dátami v štruktúrovanom formáte

Uistite sa, že CSV súbor má konzistentnú štruktúru - rovnaký počet stĺpcov v každom riadku a správne oddelovače. Ideálne by mal obsahovať hlavičku s názvami stĺpcov na prvom riadku.

Overenie výsledku

Kontrola schémy a obsahu pomocou `printSchema()` a `show()`

Po načítaní je dôležité overiť, či dáta zodpovedajú očakávanej štruktúre. Metóda `printSchema()` zobrazí štruktúru a dátové typy, zatiaľ čo `show(n)` zobrazí prvých `n` riadkov pre rýchlu vizuálnu kontrolu.



Konfigurácia čítania

Nastavenie parametrov ako oddelovač, hlavička, inferencia schémy

Môžete definovať množstvo parametrov: delimiter (oddelovač), header (či súbor obsahuje hlavičku), quote (znak pre citácie), escape (znak pre escape sekvencie) a inferSchema (automatické odvodenie dátových typov).

Načítanie do DataFrame

Použitie `spark.read.csv()` s konfiguračnými parametrami

Metóda `spark.read.csv()` vytvára DataFrame zo súboru. Môžete načítať jeden súbor, viacero súborov alebo celý adresár.

Podporuje aj prácu s komprimovanými súbormi (gzip, bzip2).

```
df = spark.read.option("header", True).option("inferSchema", True).csv("data/osoby.csv")
```

Načítanie dát - JSON formát

Vlastnosti JSON formátu

JSON (JavaScript Object Notation) je flexibilný formát, ktorý podporuje vnorené štruktúry a polia. Spark dokáže automaticky inferovať schému z JSON dát, vrátane komplexných dátových typov.

JSON je vhodný pre semi-štruktúrované dáta, kde každý záznam môže mať mierne odlišnú štruktúru. Spark zvládne aj chýbajúce polia v niektorých záznamoch.

Načítanie JSON dát

Pre načítanie JSON dát do DataFrame používame metódu `spark.read.json()`. Môžeme špecifikovať rôzne parametre pre prispôsobenie procesu načítania.

```
df = spark.read.json("data/zaznamy.json")
# Načítanie JSON s multiline záznamami
df = spark.read.option("multiline", True)
.json("data/zaznamy.json")
```

Ukladanie výstupov



Výber formátu

Spark podporuje ukladanie dát v rôznych formátoch vrátane CSV, JSON, Parquet a ORC. Každý formát má svoje výhody - napríklad Parquet je optimalizovaný pre analytické dotazy.

Poznámka: Pre OLAP operácie odporúčame Parquet, zatiaľ čo pre interoperabilitu s inými systémami môže byť vhodnejší CSV alebo JSON formát. Parquet dosahuje vyššiu kompresiu a rýchlejšie čítanie pri analytických operáciách.



Konfigurácia zápisu

Môžeme špecifikovať rôzne parametre zápisu, ako napríklad režim zápisu (overwrite, append, error, ignore) a formátovo-špecifické nastavenia.

Poznámka: Použitie "overwrite" prepíše existujúce dátu, "append" pridá nové záznamy, "error" vyvolá chybu pri existencii cieľa a "ignore" preskočí zápis, ak cieľ existuje. Pri CSV je dôležité nastaviť "header" a "delimiter" parametre.



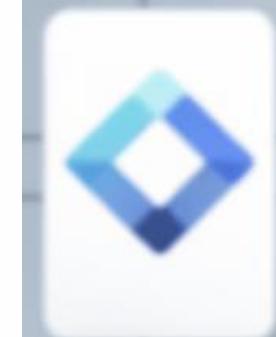
Vykonanie zápisu

Použitie metódy write s reťazením konfiguračných metód a špecifikáciou cieľového formátu a umiestnenia.

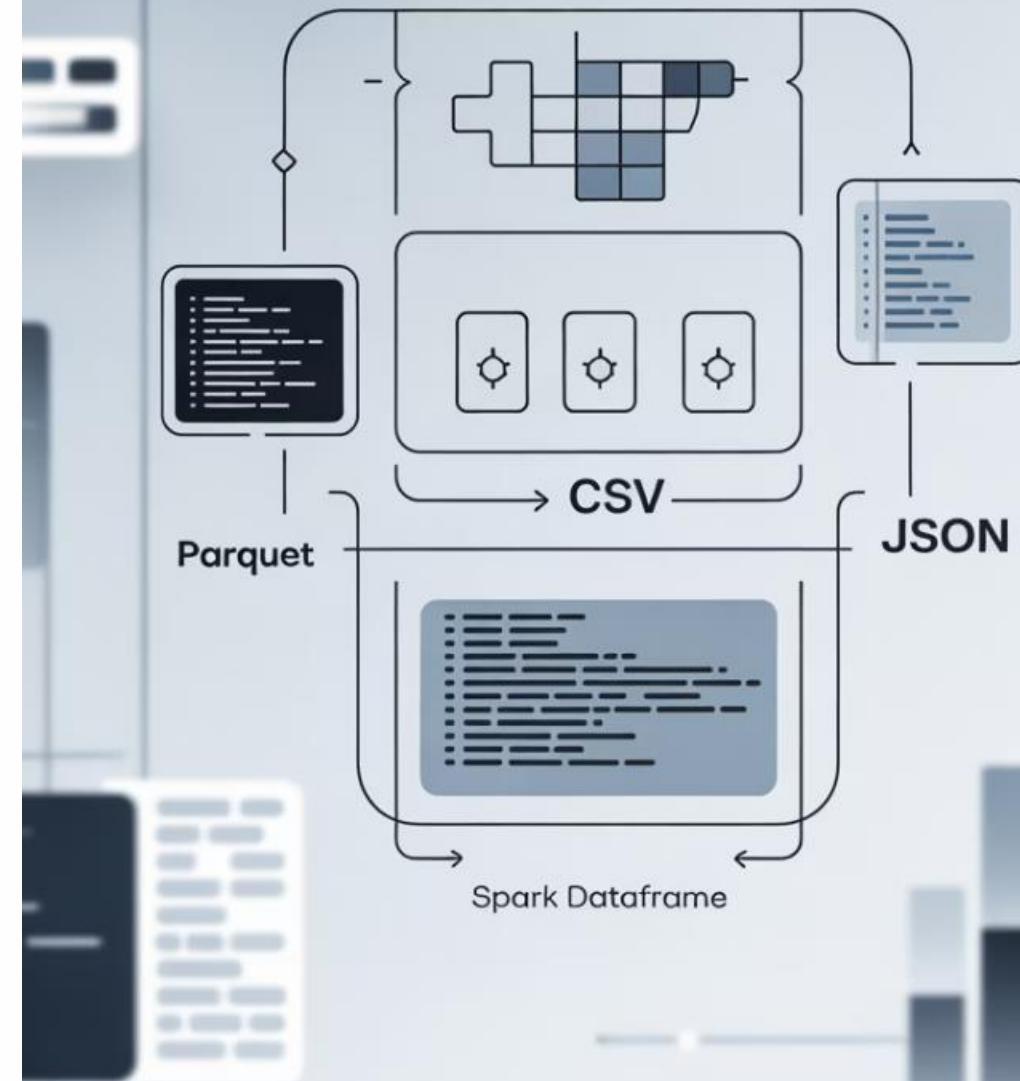
Poznámka: Operácia zápisu je spustená až po volaní metódy pre formát (json(), csv(), parquet()). Pre optimalizáciu výkonu môžeme použiť aj metódu partitionBy() na rozdelenie výstupu podľa vybraných stĺpcov.

```
df.write.mode("overwrite").json("vystup")df.write.mode("append").option("header",  
True).csv("vystup_csv")  
# S partíciou podľa stĺpca  
df.write.partitionBy("rok", "mesiac").parquet("data_po_mesiacoch")
```

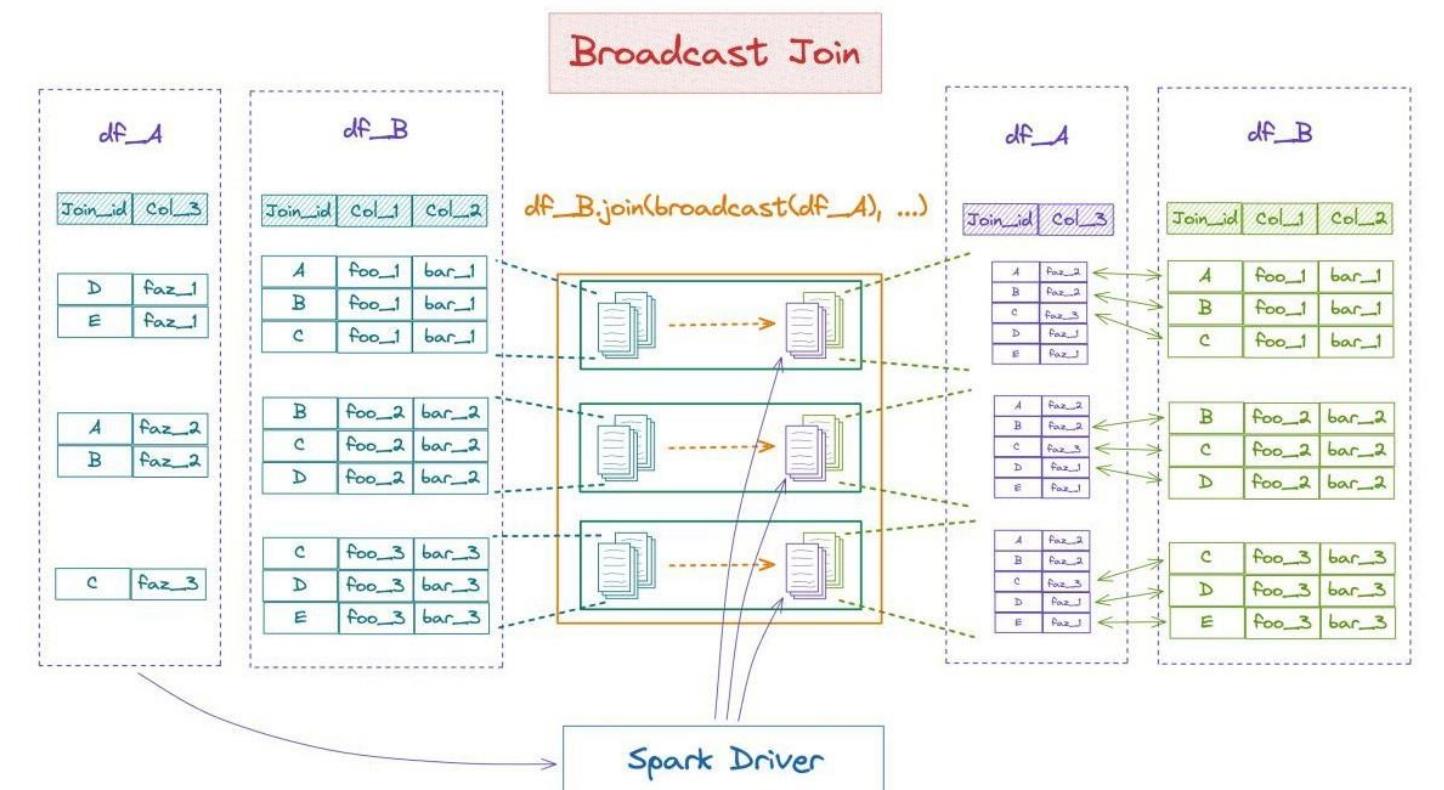
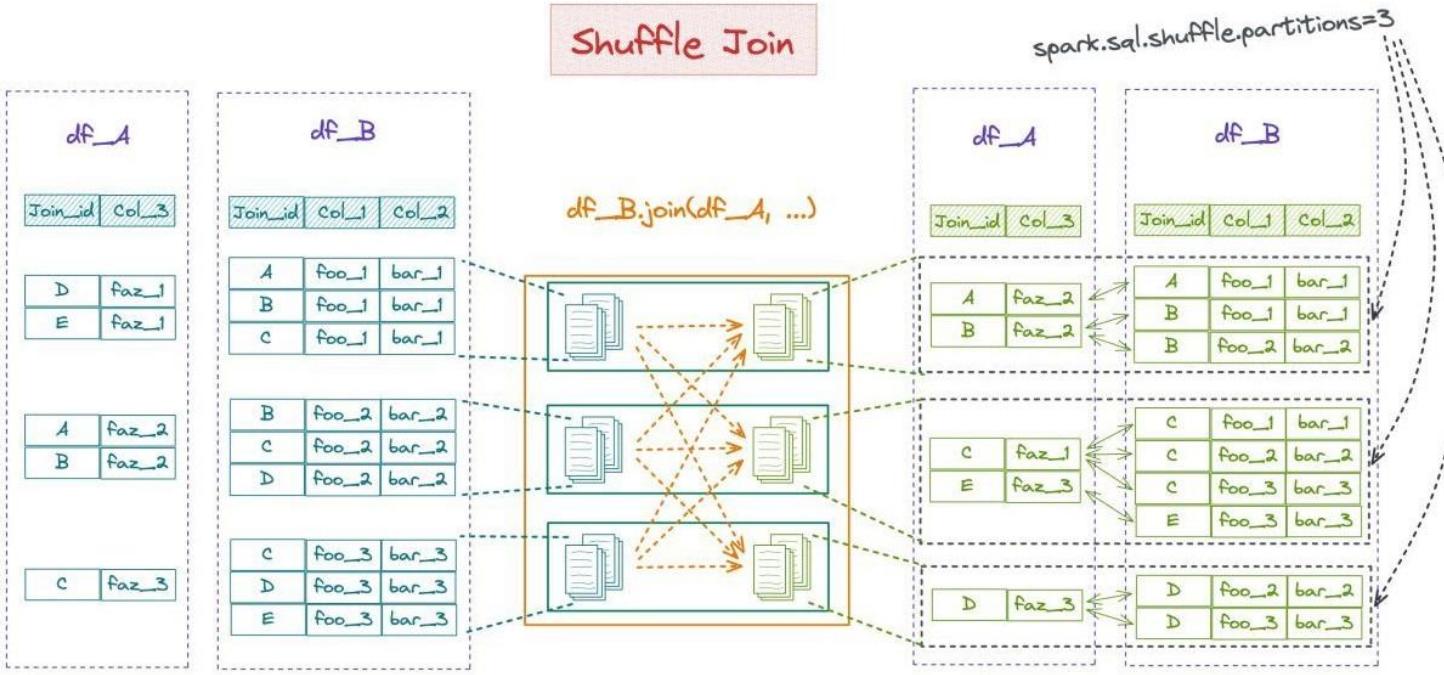
Poznámka: Pri práci s veľkými dátovými sadami je dôležité zvážiť particionovanie a kompresnú metódu. Pre Parquet formát je štandardne použitá kompresia Snappy, ktorá poskytuje dobrý pomer medzi rýchlosťou a úrovňou kompresie.



Spävia Dataframe s caark otavesformats



Shuffle a Broadcast Join



- Parent Partition

- Child Partition

- Broadcasted Dataset

- Data Container ↔ - Join Operation

Vytvorenie a použitie SQL tabuľiek

Spark ponúka niekoľko spôsobov, ako pracovať s dátami vo forme SQL tabuľiek. Každý typ tabuľky má svoje špecifické použitie a životnosť.

Vytvorenie dočasnej tabuľky

DataFrame môžeme zaregistrovať ako dočasnú SQL tabuľku (view), ktorá existuje len počas aktuálnej Spark session.

```
df.createOrReplaceTempView("osoby")
```

Poznámka: Dočasné tabuľky sú ideálne pre krátkodobé analýzy a transformácie. Po reštarte aplikácie je potrebné ich znova vytvoriť.

Vytvorenie globálnej dočasnej tabuľky

Globálna dočasná tabuľka je dostupná naprieč všetkými sessions v rámci Spark aplikácie.

```
df.createOrReplaceGlobalTempView("global_osoby")
```

Poznámka: Pri prístupe ku globálnym tabuľkám je potrebné použiť prefix "global_temp", napríklad: `SELECT * FROM global_temp.global_osoby`

Vytvorenie trvalej tabuľky

Ak používame metastore (napr. Hive, Delta Lake), môžeme vytvoriť trvalú tabuľku, ktorá pretrvá aj po ukončení Spark session.

```
df.write.saveAsTable("produkty")
```

Poznámka: Trvalé tabuľky sú vhodné pre produkčné prostredie a dlhodobé ukladanie dát. Fyzické dáta sú uložené v adresári definovanom v konfigurácii `spark.sql.warehouse.dir`.

Správny výber typu tabuľky závisí od požiadaviek na životnosť dát a zdieľanie medzi rôznymi komponentmi aplikácie.

SQL Tables

Convert to views



Create table
Dataframe to view



Čo sa naučíme?

1. Čo je DataFrame v Apache Spark a čím sa líši od RDD?
2. Aké sú hlavné výhody použitia DataFrame namiesto RDD?
3. Ako vytvoriť DataFrame v PySparku a ako sa líši od Pandas DataFrame?
4. Ako funguje operácia select() a kedy ju použiť?
5. Ako aplikovať podmienku výberu pomocou filter() v DataFrame?
6. Ako fungujú groupBy() a agg() pri agregáciách nad DataFrame?
7. Profit

Ako Spracovat' Datasety v Spark SQL



AKREDITOVANÝ KURZ





Čo sa naučíme?

1. Čo je Spark SQL a na čo slúži v prostredí Apache Spark?
2. Ako vytvoriť dočasnú tabuľku (temporary view) zo Spark DataFrame?
3. Aký je rozdiel medzi TempView a GlobalTempView?
4. Ako spustiť SQL dopyt nad Spark tabuľkou a čo vracia?
5. Ako Spark SQL efektívne spracováva veľké dátové objemy?
6. Ako uložiť výsledok SQL dopytu do trvalej tabuľky alebo súboru?
7. Profit

Praktické cvičenie: Načítanie CSV do DataFrame

Kód pre načítanie CSV

Pre načítanie CSV súboru do DataFrame použijeme metódu spark.read.csv() s niekoľkými konfiguračnými parametrami:

- header=True - prvý riadok obsahuje názvy stĺpcov
- inferSchema=True - automatická detekcia dátových typov

```
df = spark.read.option("header", True) \
    .option("inferSchema", True) \
    .csv("osoby.csv")df.show()
```

Výsledok

Po vykonaní kódu by sme mali vidieť obsah DataFrame zobrazený v tabuľkovej forme. Spark automaticky detegoval dátové typy stĺpcov - napríklad "vek" a "prijem" ako číselné hodnoty, zatiaľ čo "meno" a "mesto" ako reťazce.

Metóda show() zobrazí prvých 20 riadkov DataFrame (alebo menej, ak DataFrame obsahuje menej riadkov). Môžeme špecifikovať počet riadkov ako parameter: df.show(5).

Spúšťanie SQL dopytov - Základný výber

Syntax SQL dopytov

Spark SQL podporuje štandardnú SQL syntax podobnú ANSI SQL. Môžeme používať klauzuly SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY a ďalšie.

SQL dopyty v Spark môžu pracovať s dočasnými tabuľkami vytvorenými z DataFrame alebo s trvalými tabuľkami v metastore.

Poznámka: Pri práci so Spark SQL je dôležité najprv importovať potrebné knižnice a vytvoriť SparkSession objekt. Syntax je kompatibilná s väčšinou SQL dialektov.

Príklady základných dopytov

```
# Jednoduchý výber stípcovspark.sql("SELECT meno, vek  
FROM osoby").show()  
# Poznámka: Vyberie len stĺpce 'meno' a 'vek' z tabuľky  
'osoby'  
# Výber s podmienkou  
spark.sql(""" SELECT meno, vek FROM osoby WHERE  
vek > 30""").show()  
# Poznámka: Filtruje len osoby staršie ako 30 rokov  
# Výber s limitom počtu záznamov  
spark.sql("SELECT * FROM osoby LIMIT 5").show()  
# Poznámka: Vráti len prvých 5 záznamov z tabuľky
```

*Poznámka: Metóda .show() zobrazí výsledky v konzole. Pre uloženie výsledkov do DataFrame použite premenné, napr.: result_df = spark.sql("SELECT * FROM osoby").*

Praktické cvičenie: Základné čistenie a zobrazenie

Overenie schémy

Najprv skontrolujeme schému DataFrame, aby sme videli dátové typy jednotlivých stĺpcov. To je dôležité pre správne spracovanie dát.

```
df.printSchema()
```

Filtrovanie dát

Následne vyfiltrujeme záznamy podľa podmienky - v tomto prípade odstránime osoby s príjmom pod 1000 EUR.

```
df_filtered = df.filter(df["prijem"] >= 1000)df_filtered.show()
```

Analýza výsledkov

Po filtrovaní by sme mali vidieť, že záznam s ID 4 (Marek s príjmom 950) bol odstránený z výsledného DataFrame. Ostatné záznamy zostali zachované.



Praktické cvičenie: Agregácie a štatistiky

Priemerný príjem podľa pohlavia

Vypočítame priemerný príjem pre mužov a ženy pomocou zoskupenia podľa stĺpca "pohlavie" a následnej agregácie pomocou funkcie avg().

```
df.groupBy("pohlavie").avg("prijem").show()
```

Výsledok by mal ukázať, že priemerný príjem sa líši medzi pohlaviami - môžeme tak identifikovať potenciálne rozdiely v odmeňovaní.

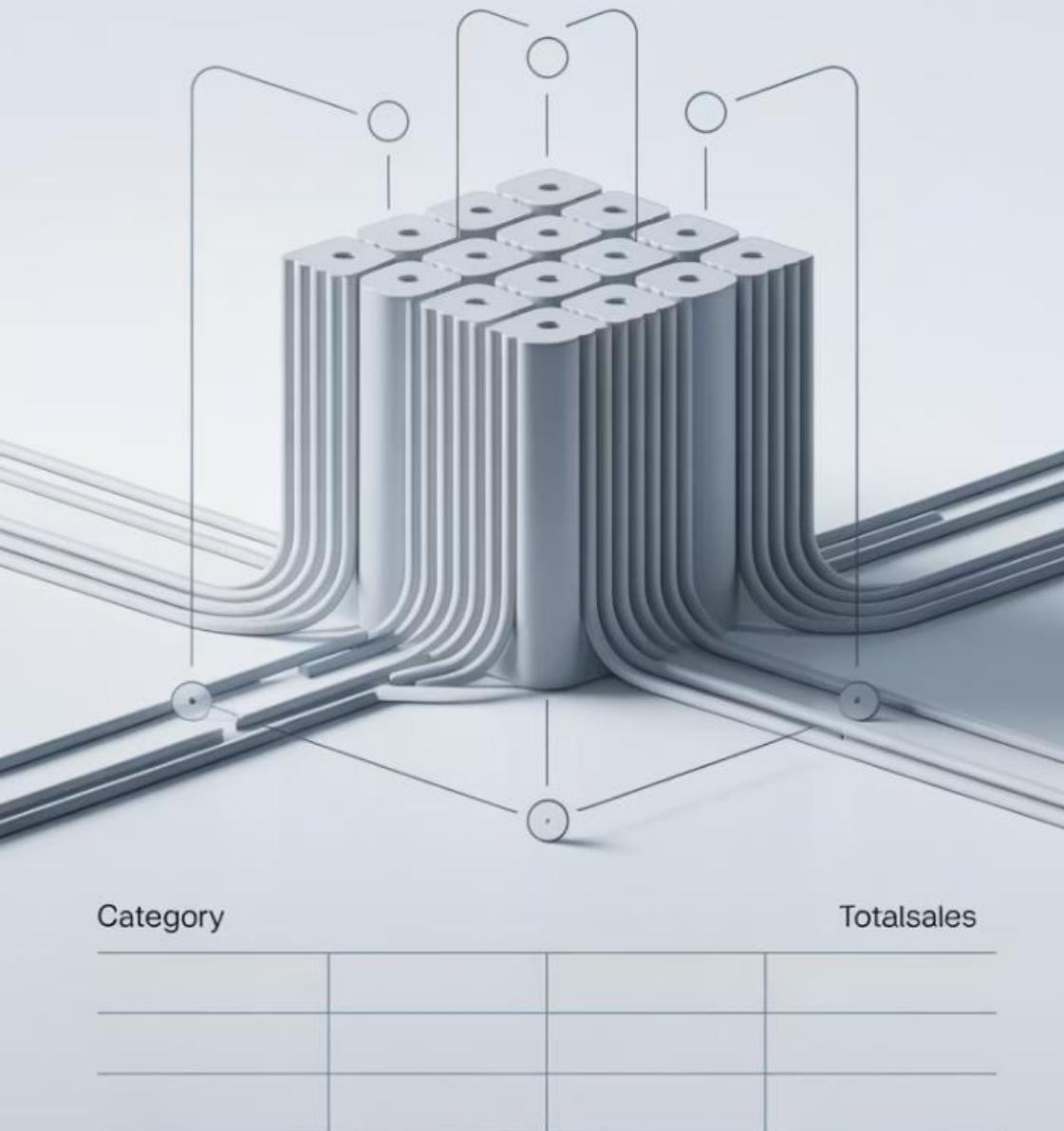
Maximálny vek podľa mesta

Zistíme maximálny vek obyvateľov v každom meste pomocou zoskupenia podľa stĺpca "mesto" a následnej agregácie pomocou funkcie max().

```
df.groupBy("mesto").max("vek").show()
```

Táto analýza nám môže pomôcť identifikovať mestá s najstaršími obyvateľmi v našom datasete, čo môže byť užitočné pre demografické štúdie.

Group by Spark



Spúšťanie SQL dopytov - Agregácie a zoskupovanie

GROUP BY klauzula

GROUP BY umožňuje zoskupiť záznamy podľa hodnôt v jednom alebo viacerých stĺpcoch. Typicky sa používa spolu s agregačnými funkciami.

```
-- Zoskupenie osôb podľa pohlavia a počítanie záznamov v každej skupine  
SELECT pohlavie, COUNT(*) as pocet FROM osoby GROUP BY pohlavie  
-- Rozdelí dátá do skupín podľa hodnoty v stĺpci 'pohlavie'
```

Agregačné funkcie

SQL v Spark podporuje štandardné agregačné funkcie ako COUNT, SUM, AVG, MIN, MAX a ďalšie. Tieto funkcie sa aplikujú na každú skupinu zvlášť.

```
-- Výpočet počtu osôb a priemerného veku pre každú skupinu podľa pohlavia  
SELECT pohlavie, COUNT(*) as pocet,  
-- Počet osôb v každej skupine  
AVG(vek) as priemer  
-- Priemerný vek v každej skupine FROM osoby  
GROUP BY pohlavie  
-- Rozdelenie dát podľa stĺpca 'pohlavie'
```

HAVING klauzula

HAVING umožňuje filtrovať výsledky agregácie podobne ako WHERE filtriuje jednotlivé riadky. HAVING sa aplikuje po GROUP BY, zatiaľ čo WHERE pred ním.

```
-- Nájdenie miest, kde je priemerný vek nad 30 rokov  
SELECT mesto, AVG(vek) as priemer FROM osoby GROUP BY mesto  
-- Zoskupí záznamy podľa mesta  
HAVING AVG(vek) > 30  
-- Ponechá len skupiny, kde je priemerný vek vyšší ako 30
```

Catalyst Optimizer

Analýza dopytu

Catalyst analyzuje SQL dopyt alebo DataFrame operácie a vytvorí logický plán, ktorý reprezentuje požadované transformácie dát.

Poznámka: V tejto fáze sa analyzuje syntax a sémantika dopytu, kontrolujú sa názvy stĺpcov a tabuliek, a vytvára sa abstraktná syntaktická stromová štruktúra (AST).

Logická optimalizácia

Optimalizátor aplikuje sadu pravidiel na transformáciu logického plánu - napríklad odstránenie nepotrebných operácií, zjednodušenie výrazov a predikátov.

Poznámka: Medzi typické optimalizácie patrí presun filtrov bližšie k zdroju dát (predicate pushdown), odstránenie mŕtveho kódu a konštantné skladanie. Tieto optimalizácie sú nezávislé od fyzickej implementácie.

Fyzická optimalizácia

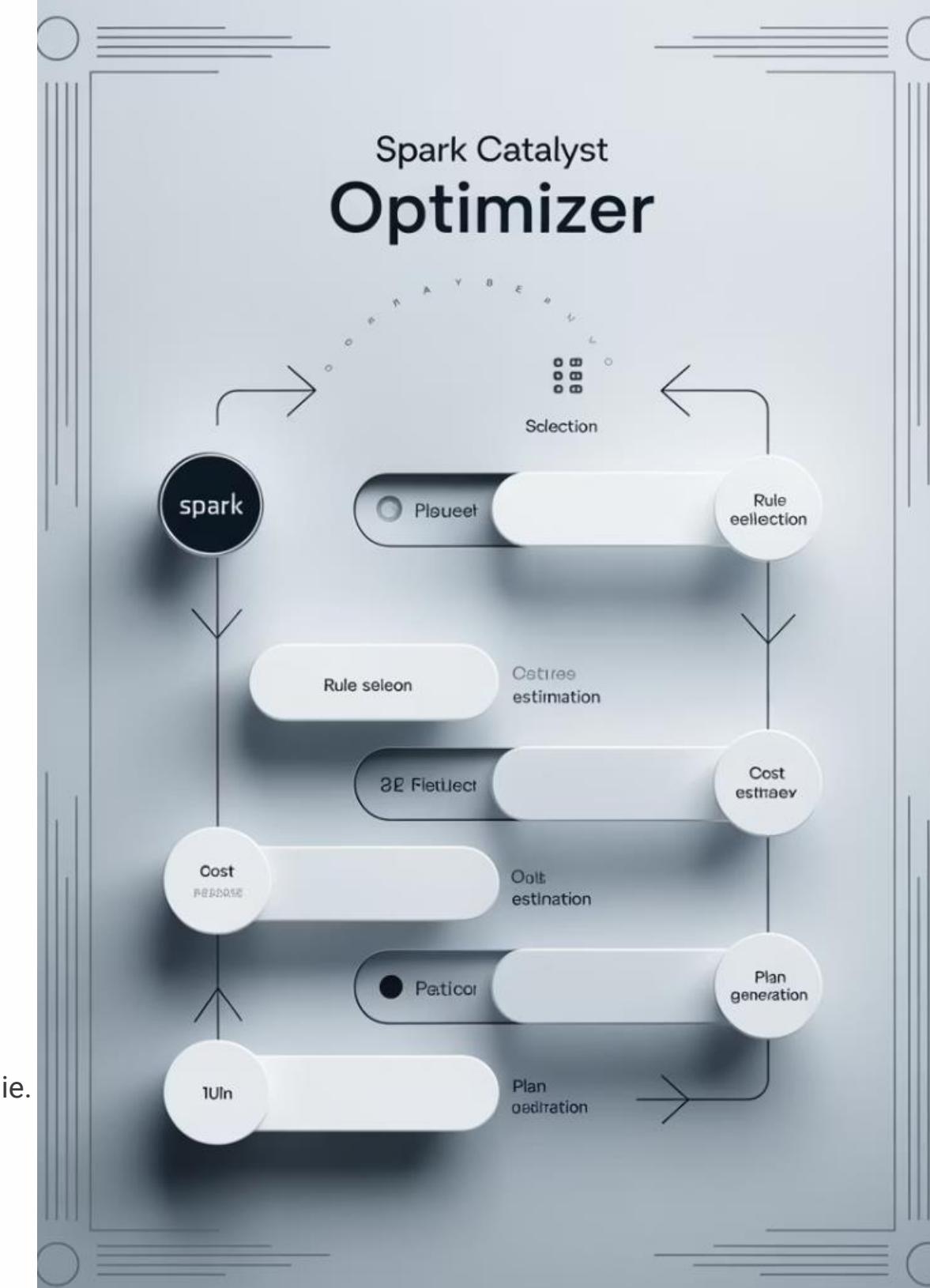
Logický plán je prevedený na fyzický plán, ktorý špecifikuje konkrétné algoritmy a stratégie pre vykonanie operácií (napr. broadcast join vs. sort-merge join).

Poznámka: Optimalizátor zvažuje štatistiky, veľkosť dát a dostupné systémové zdroje pre výber najefektívnejšej implementácie. Cost-based optimizer používa matematické modely na predpovedanie výkonnosti rôznych stratégii.

Generovanie kódú

Spark generuje optimalizovaný bytecode pre kritické časti výpočtu, čo ďalej zrýchluje vykonanie.

Poznámka: Whole-stage code generation kombinuje viacero operátorov do jedného optimalizovaného kódu, čím sa znížuje rážia volania funkcií a zlepšuje sa využitie CPU cache. Generovaný kód využíva moderné JVM optimalizácie.



Spark dataframe

Dospelos	Rok	29.09	55935	02215	69290	75805	500
Obrat		152.3	431.26	65398	22285	48391	52908
Odmiet		926.9	98.00	49588	65595	92300	49909
Delen		732.5	97.98	49738	82608	12869	49902
Dan.čas		431.5	37.858	67238	07935	202.26	45885
Cestina		48.46	57.194	92.268	82589	62.908	59803
Danet		931.3	55.538	49396	63.968	12036	56808
Osama		0.375	215.93	42585	26.887	69.118	59.985

Praktické cvičenie: Vytvorenie nového stĺpca

Import potrebných funkcií

Pre prácu so stĺpcami a výrazmi potrebujeme importovať funkcie z modulu pyspark.sql.functions. Najčastejšie používaná je funkcia col(), ktorá umožňuje referencovať stĺpce DataFrame.

```
from pyspark.sql.functions import col  
# Importujeme funkciu col pre prácu so stĺpcami DataFrame
```

Vytvorenie stĺpca s príjomom po zdanení

Pomocou metódy withColumn() vytvoríme nový stĺpec "prijem_po_zdani", ktorý bude obsahovať hodnotu príjmu po odpočítaní 20% dane (teda 80% pôvodnej hodnoty).

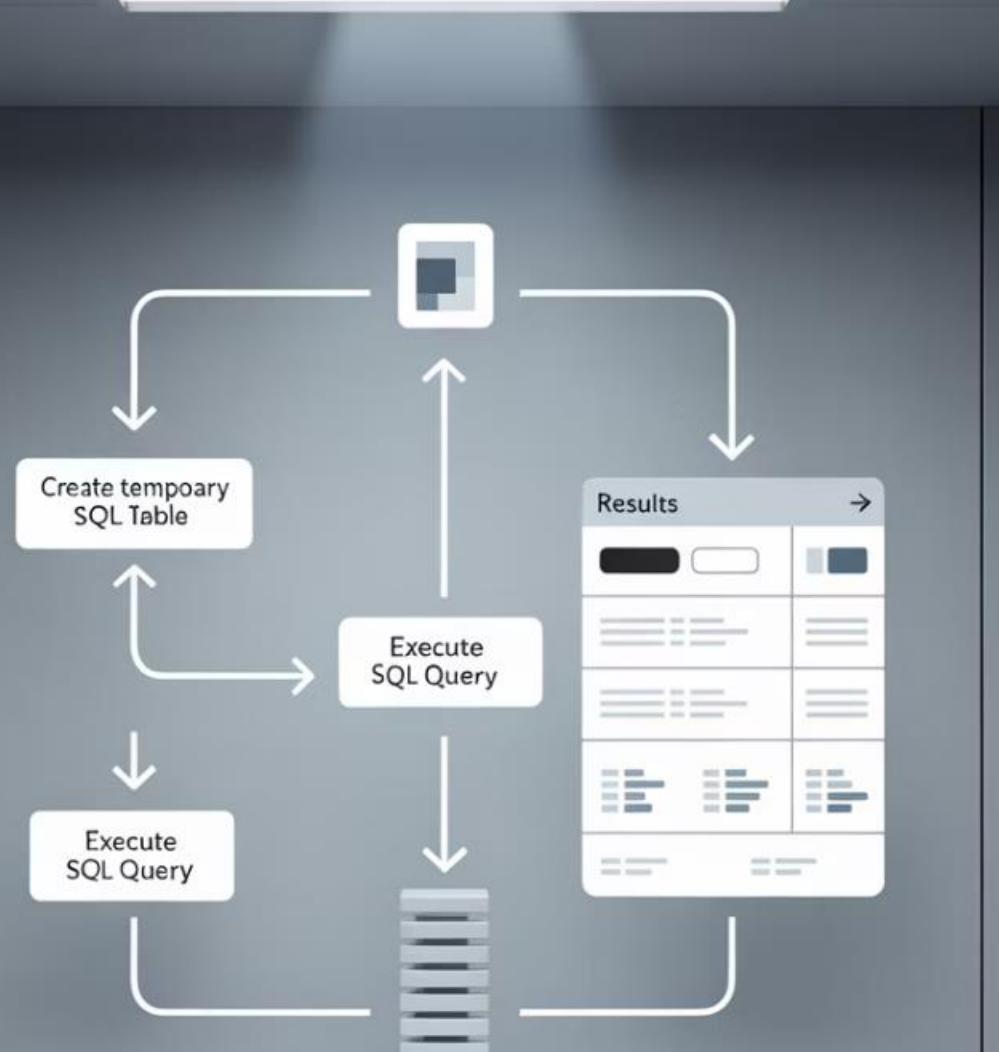
```
df = df.withColumn("prijem_po_zdani",  
                   col("prijem") * 0.8)  
# Vytvorenie nového stĺpca s hodnotou príjmu  
zniženou o 20% daň  
# Použijeme col("prijem") pre referenciu na  
existujúci stĺpec a násobíme koeficientom  
0.8  
# Výsledok priradíme späť do DataFrame  
pomocou metódy withColumn()
```

Zobrazenie výsledku

Zobrazíme vybrané stĺpce z DataFrame, aby sme videli pôvodný príjem a príjem po zdanení pre každú osobu.

```
df.select("meno", "prijem",  
          "prijem_po_zdani").show()  
# Vyberieme len potrebné stĺpce pomocou select()  
# Parameter show() určuje počet riadkov na  
zobrazenie  
# Bez parametra zobrazí predvolených 20 riadkov
```

Praktické cvičenie: Spark SQL – dočasná tabuľka a dopyty



Vytvorenie dočasnej tabuľky

Zaregistrujeme DataFrame ako dočasné SQL tabuľky, aby sme mohli používať SQL syntax pre dotazovanie.

```
# Vytvorenie dočasnej tabuľky - tabuľka existuje len počas aktuálnej Spark session  
# Umožňuje prístup k dátam pomocou štandardných SQL príkazov  
# Názov "osoby" bude použitý na referencovanie tabuľky v SQL  
dopysql.createOrReplaceTempView("osoby")
```

Napísanie SQL dopytu

Vytvoríme SQL dopyt, ktorý vyberie osoby staršie ako 30 rokov s príjmom nad 1500 EUR.

```
# SQL dopyt s viacerými podmienkami vo WHERE klauzule# Vyberie len stĺpce 'meno', 'vek' a 'prijem'  
zo všetkých dostupných# Filtrovanie pomocou logického operátora AND pre splnenie oboch podmienok  
sql_dopyt = """  
SELECT meno, vek, prijem  
FROM osoby  
WHERE vek > 30 AND prijem > 1500  
"""
```

Vykonanie dopytu

Spustíme SQL dopyt a výsledok uložíme do nového DataFrame.

```
# spark.sql() vykoná SQL dopyt a vráti výsledok ako nový DataFrame  
# Výsledok je uložený do premennej sql_df pre ďalšie použitia  
# Metóda show() zobrazí prvých n riadkov výsledku (predvolene 20)  
sql_df = spark.sql(sql_dopyt)  
sql_df.show()
```

Praktické cvičenie: Uloženie výsledkov do CSV

Príprava na export

Rozhodneme sa, ktoré dátá chceme exportovať a v akom formáte

Poznámka: Je dôležité vybrať správne stĺpce a prípadne vykonať transformácie pred exportom (filtrovanie, agregácia, atď.), aby výsledný súbor obsahoval presne to, čo potrebujeme.

Overenie výstupu

Skontrolujeme, či boli dátá správne uložené

Poznámka: Po exporte je vhodné overiť obsah súboru pomocou príkazu !ls vystup_csv alebo načítať súbor späť do DataFrame pre kontrolu pomocou spark.read.csv("vystup_csv").



Konfigurácia exportu

Nastavíme parametre ako režim zápisu a formát hlavičky

Poznámka: Parameter "mode" určuje správanie pri existujúcom súbore (overwrite = prepísat, append = pridať, error = vyhodiť chybu). Parameter "header" určuje, či CSV bude obsahovať názvy stĺpcov.

Vykonanie zápisu

Spustíme operáciu zápisu do cieľového umiestnenia

Poznámka: Spark vytvorí adresár s názvom "vystup_csv", ktorý bude obsahovať výsledný súbor aj metadata. Ak požadujeme jeden konkrétny súbor, musíme použiť metódu coalesce(1) pred zápisom.

```
sql_df.write.mode("overwrite").option("header", True).csv("vystup_csv")
```

Pokročilé cvičenie: Rozšírená analytika - Príprava

V tomto cvičenieoratóriu sa naučíme pracovať s viacerými dátovými súbormi, čo je základom pre pokročilé analytické operácie.

Vytvorenie druhého dátového súboru

Pre pokročilé analytické operácie potrebujeme druhý dátový súbor, ktorý budeme spájať s pôvodným. Vytvoríme súbor platy.csv s informáciami o pozíciach a bonusoch zamestnancov.

Poznámka: Štruktúra súboru obsahuje jedinečné ID pre každého zamestnanca, ktoré bude slúžiť ako kľúč pre spojenie s hlavným súborom.

id	pozícia	bonus
1	Analytik	300
2	Manažér	500
3	Analytik	400

Poznámka: Hodnoty bonusov sú uvedené v eurách a budú neskôr použité pre výpočet celkového príjmu.

Načítanie druhého súboru

Načítame druhý súbor do DataFrame podobne ako prvý, s automatickou detekciou schémy a hlavičky.

```
# Načítanie dát o platoch a bonusoch
# option("header", True) - prvý riadok obsahuje názvy stĺpcov
# option("inferSchema", True) - automatická detekcia dátových typov
df_platy = spark.read \ .option("header", True) \
.option("inferSchema", True) \ .csv("platy.csv")
# Zobrazenie obsahu DataFrame pre kontrolu
df_platy.show()
```

Tento DataFrame obsahuje informácie o pracovných pozíciach a bonusoch, ktoré budeme spájať s informáciami o osobách z pôvodného DataFrame.

Poznámka: Pri práci s reálnymi dátami je dôležité overiť, či sú všetky ID hodnoty jedinečné a či neexistujú chýbajúce hodnoty, ktoré by mohli ovplyvniť výsledky spojenia.

Tip: Pre lepšie porozumenie dátam môžete použiť metódu df_platy.printSchema() na zobrazenie dátových typov jednotlivých stĺpcov.

Spúšťanie SQL dopytov - Spojenia

INNER JOIN

Vráti len záznamy, ktoré majú zhodu v oboch tabuľkách. Záznamy bez zhody sú vynechané z výsledku.

```
SELECT a.meno, b.plat FROM osoby a JOIN platy b ON a.id = b.id
```

Poznámka: Najčastejšie používaný typ spojenia. Vhodný, keď potrebujete len záznamy, ktoré existujú v oboch tabuľkách. V Sparku je toto spojenie optimalizované pre výkon.

LEFT JOIN

Vráti všetky záznamy z ľavej tabuľky a zodpovedajúce záznamy z pravej tabuľky. Ak neexistuje zhoda, hodnoty z pravej tabuľky sú NULL.

```
SELECT a.meno, b.plat FROM osoby a LEFT JOIN platy b ON a.id = b.id
```

Poznámka: Užitočné pri zachovaní kompletных údajov z hlavnej (ľavej) tabuľky. Často sa používa pri reportoch, kde chcete vidieť všetky záznamy aj keď nemajú priradené hodnoty.

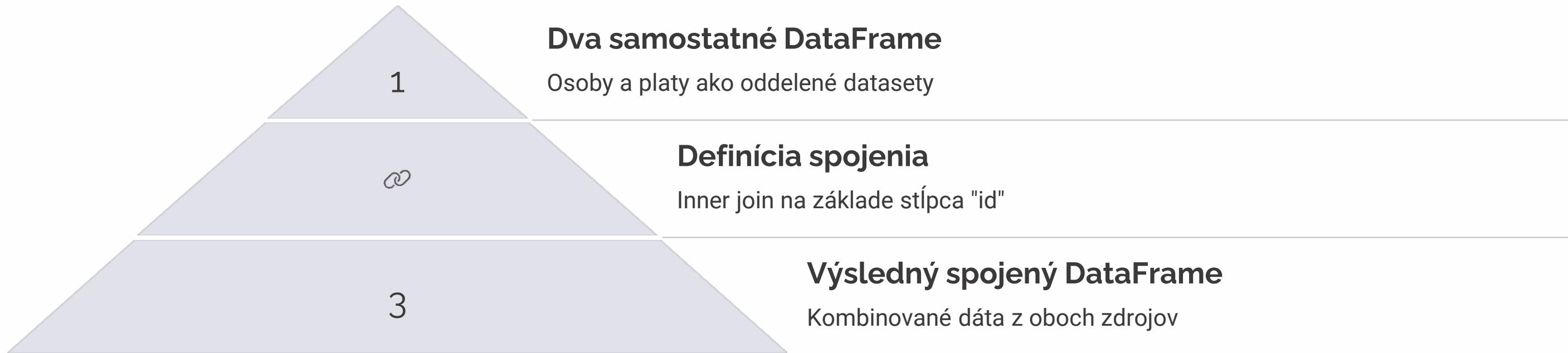
FULL OUTER JOIN

Vráti všetky záznamy z oboch tabuľiek. Ak neexistuje zhoda, hodnoty z druhej tabuľky sú NULL.

```
SELECT a.meno, b.plat FROM osoby a FULL OUTER JOIN platy b ON a.id = b.id
```

Poznámka: Najkomplexnejší typ spojenia, ktorý zachováva všetky údaje. Využiteľný pri integračných procesoch a kontrole dátovej konzistencie medzi tabuľkami. V distribuovaných systémoch môže byť výpočtovo náročnejší.

Pokročilé cvičenie: Join - spojenie dát



Spojenie (join) je kľúčová operácia pri práci s viacerými datasetmi. Spájame informácie o osobách s informáciami o ich platoch a bonusoch.

```
# Spojenie dvoch DataFrame pomocou inner join
# Inner join zachová len záznamy, ktoré existujú v oboch datasetoch
# Parameter "on" určuje, podľa ktorého stĺpca sa má vykonať spojenie
df_joined = df.join(df_platy, on="id", how="inner")
# Zobrazenie výsledného DataFrame so stĺpcami z oboch zdrojov
df_joined.select("meno", "pozicia", "prijem", "bonus").show()
```

Po vykonaní operácie join získame jeden ucelený DataFrame, ktorý obsahuje dáta z oboch zdrojov. Inner join zachováva len riadky, kde sa hodnota "id" nachádza v oboch datasetoch. Ak by sme chceli zachovať všetky záznamy z hlavného DataFrame, použili by sme "left" join.

Pokročilé cvičenie: Vytvorenie stĺpca s celkovým príjmom

Výpočet celkového príjmu

Po spojení datasetov môžeme vypočítať celkový príjem každej osoby ako súčet základného príjmu a bonusu. Použijeme metódu `withColumn()` spolu s funkciou `col()` pre referencovanie stĺpcov.

```
from pyspark.sql.functions import col
df_joined = df_joined.withColumn("celkovy_prijem", col("prijem") + col("bonus"))
df_joined.select("meno", "pozicia", "celkovy_prijem").show()
```

Analýza výsledkov

Po vytvorení nového stĺpca môžeme analyzovať celkové príjmy zamestnancov. Vidíme, že bonus významne prispieva k celkovému príjmu, najmä u manažérov, kde môže tvoriť podstatnú časť odmeny.

Tento typ analýzy je užitočný pre oddelenie ľudských zdrojov pri hodnotení kompenzačných balíčkov a pre zamestnancov pri porovnávaní celkových odmiennaprieč rôznymi pozíciami.

Pokročilé cvičenie: Window funkcia - poradie podľa príjmu

Window funkcie v Spark umožňujú vykonávať výpočty nad skupinami riadkov (tzv. oknami) bez potreby zoskupovania dát, čo je ideálne pre analýzu a porovnávanie v rámci kategórií.

Import potrebných funkcií

Pre prácu s window funkciemi potrebujeme importovať Window z pyspark.sql.window a funkciu row_number() z pyspark.sql.functions.

```
# Importujeme Window pre definíciu okna nad dátami
# row_number() priradí postupné čísla v rámci definovaného
# okna
from pyspark.sql.window import Window
from pyspark.sql.functions import row_number
```

Poznámka: Window funkcie v Spark sú podobné SQL window funkciám, umožňujú vykonávať výpočty nad skupinou riadkov.

Definícia window špecifikácie

Vytvoríme window špecifikáciu, ktorá rozdelí dátá podľa pozície a zoradí ich podľa celkového príjmu zostupne.

```
# Vytvárame window špecifikáciu:
# - partitionBy rozdelí dátá podľa stĺpca "pozicia"
# - orderBy zoradí záznamy zostupne podľa
# "celkovy_prijem"
window_spec =
Window.partitionBy("pozicia") \
.orderBy(col("celkovy_prijem").desc())
```

Poznámka: Partícia definuje skupinu riadkov, v rámci ktorej sa počítia window funkcia - v tomto prípade vytvárame samostatné rebríčky pre každú pracovnú pozíciu.

Aplikácia window funkcie

Aplikujeme funkciu row_number() na definované okno, čím získame poradie zamestnancov v rámci každej pozície podľa ich celkového príjmu.

```
# Pridávame nový stĺpec "poradie":
# - row_number() priradí sekvenčné čísla (1, 2, 3, ...)
# - .over(window_spec) aplikuje funkciu na definované
# okno
df_joined = df_joined.withColumn("poradie",
row_number().over(window_spec))
# Zobrazíme relevantné stĺpce pre lepšiu
# čitateľnosť
df_joined.select("meno", "pozicia",
"celkovy_prijem", "poradie").show()
```

Poznámka: Výsledný DataFrame obsahuje nový stĺpec "poradie", ktorý označuje pozíciu zamestnanca v rebríčku príjmov v rámci jeho pracovnej kategórie. Zamestnanec s najvyšším príjomom v danej pozícii má hodnotu 1.

Window funkcie sú výkonným nástrojom pre analytické úlohy, ako je ranking, percentily alebo kľavé priemery, a sú efektívnejšie než použitie self-joinov alebo opakovaných agregácií.

Pokročilé cvičenie: Pivot - priemerný príjem podľa pozície a pohlavia

Čo je pivot?

Pivot je operácia, ktorá transformuje riadky na stĺpce, čo umožňuje vytvárať prehľadné krízové tabuľky. V Spark môžeme použiť metódu `pivot()` na DataFrame po zoskupení.

Poznámka: Pivotovanie dát je užitočné najmä pri analýze kategoriálnych premenných a ich vzťahov. Táto transformácia zjednodušuje vizualizáciu a interpretáciu komplexných dátových štruktúr.

Vytvorenie pivot tabuľky

Vytvoríme pivot tabuľku, ktorá zobrazí priemerný celkový príjem pre každú pozíciu, rozdelený podľa pohlavia.

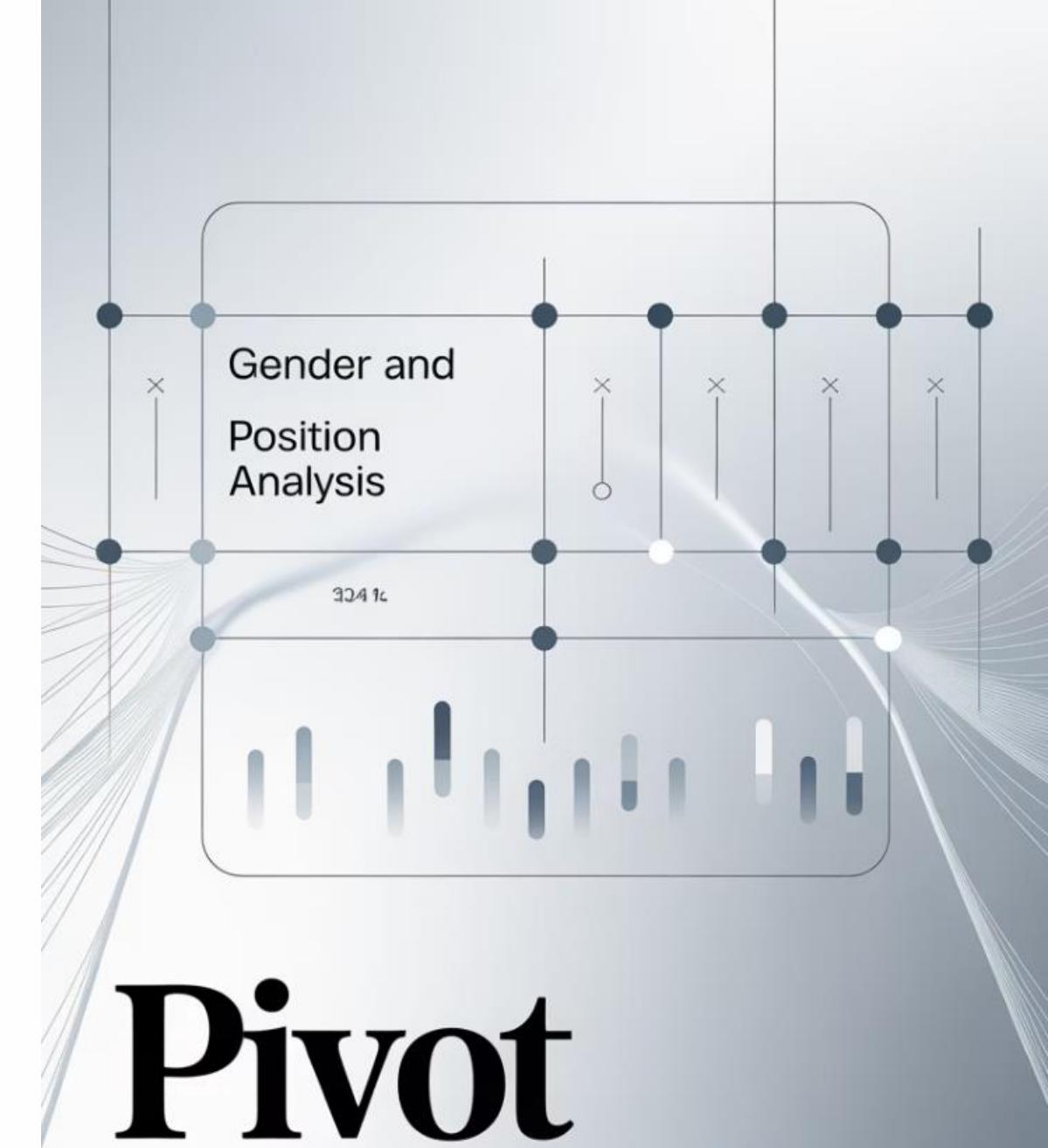
```
pivot_df = df_joined.groupBy("pozicia")  
\.pivot("pohlavie") \  
.avg("celkovy_prijem")pivot_df.show()
```

Poznámka: V tomto kóde najprv zoskupujeme dátá podľa stĺpca "pozicia", potom pivotujeme podľa stĺpca "pohlavie", a nakoniec počítame priemer stĺpca "celkovy_prijem" pre každú kombináciu pozície a pohlavia. Výsledkom je DataFrame s pozíciami v riadkoch a pohlaviami v stĺpcoch.

Interpretácia výsledkov

Výsledná tabuľka nám umožňuje jednoducho porovnať priemerné príjmy mužov a žien v rámci rovnakej pozície, čo môže odhaliť potenciálne rozdiely v odmeňovaní na základe pohlavia.

Poznámka: Pri interpretácii je dôležité brať do úvahy aj ďalšie faktory, ktoré môžu ovplyvňovať príjem, ako sú pracovné skúsenosti, vzdelanie alebo odpracované roky. Pre komplexnejšiu analýzu by bolo vhodné zahrnúť aj tieto premenné do modelu.



Pivot Operation

Pokročilé cvičenie: Pokročilé filtrovanie a úprava

Výpočet priemerného príjmu

Najprv vypočítame priemerný celkový príjem naprieč všetkými zamestnancami pomocou agregačnej funkcie avg(). Táto hodnota nám poslúži ako referenčný bod pre ďalšiu analýzu dát.

```
# Výpočet priemerného príjmu pomocou agregačnej funkcie
priemer = df_joined.agg( {"celkovy_prijem": "avg"})
# Použitie slovníka pre definíciu
agregácie).collect()[0][0]
# Extraktia hodnoty z výsledného DataFrame
# Výpis výsledku pre kontrolu
print(f"Priemerný príjem: {priemer}")
```

Filtrovanie nadpriemerných príjmov

Následne vyfiltrujeme zamestnancov, ktorých celkový príjem presahuje vypočítaný priemer. Toto nám umožní identifikovať nadpriemerne platených zamestnancov a analyzovať ich charakteristiky.

```
# Filtrovanie záznamov s nadpriemerným príjomom
# Použitie objektu col pre prístup k stĺpcu
nadpriemerni = df_joined.filter(col("celkovy_prijem") >
priemer
# Porovnanie s vypočítanou priemernou hodnotou)
# Zobrazenie výsledku s vybranými stĺpcami
nadpriemerni.select("meno", "celkovy_prijem"
# Vyberáme len relevantné stĺpce).show()
# Zobrazenie výsledkov v tabuľkovej forme
```

Pokročilé cvičenie: BONUS - Zoskupovanie s podmienkou a CASE WHEN

Vytvorenie kategórie príjmu

Pomocou funkcie when() vytvoríme nový stĺpec "kategoria", ktorý klasifikuje zamestnancov do dvoch kategórií podľa ich celkového príjmu: "Vysoký" pre príjmy nad 2000 EUR a "Nízky" pre ostatné.

Poznámka: Funkcia when() je ekvivalentom konštrukcie CASE WHEN v SQL a umožňuje podmienené vyhodnotenie podobné if-else v programovaní.

```
from pyspark.sql.functions import when, avg
# Importujeme potrebné funkcie
# when() - pre podmienené vyhodnotenie
# avg() - pre výpočet priemeru
df_joined.withColumn("kategoria",
# názov nového stĺpca
when(col("celkovy_prijem") > 2000, "Vysoký")
# podmienka a hodnota ak je splnená
.otherwise("Nízky")
# hodnota ak podmienka nie je splnená)
```

Analýza priemerného veku podľa kategórie príjmu

Následne zoskupíme zamestnancov podľa kategórie príjmu a vypočítame priemerný vek v každej skupine. Táto analýza nám môže pomôcť zistiť, či existuje korelácia medzi vekom a výškou príjmu.

Poznámka: Táto operácia kombinuje groupBy() a agregačnú funkciu avg() na získanie analytického prehľadu o našich dátach.

```
# Zoskupenie podľa kategórie
príjmudf_kateg.groupBy("kategoria") \
.agg(avg("vek"))
# agregačná funkcia pre výpočet priemeru
.show()
# zobrazenie výsledkov
# Poznámka: výstupom bude tabuľka s 2 riadkami
# (jeden pre každú kategóriu príjmu) a priemerným vekom
```

Poznámka: Kombinovanie podmienených stĺpcov a agregačných funkcií je silný nástroj pre pokročilú dátovú analýzu. Umožňuje segmentáciu dát a následnú analýzu rôznych metrík v jednotlivých segmentoch.



Čo sa naučíme?

1. Čo je Spark SQL a na čo slúži v prostredí Apache Spark?
2. Ako vytvoriť dočasnú tabuľku (temporary view) zo Spark DataFrame?
3. Aký je rozdiel medzi TempView a GlobalTempView?
4. Ako spustiť SQL dopyt nad Spark tabuľkou a čo vracia?
5. Ako Spark SQL efektívne spracováva veľké dátové objemy?
6. Ako uložiť výsledok SQL dopytu do trvalej tabuľky alebo súboru?
7. Profit

Optimalizácia výkonu v Spark



Správa pamäte

Správne nastavenie pamäťových parametrov je kľúčové pre výkon Spark aplikácií. Parametre ako spark.memory.fraction a spark.memory.storageFraction ovplyvňujú rozdelenie pamäte medzi výpočty a cache. Poznámka: Pre výpočtovo náročné úlohy zvýšte spark.memory.fraction na 0.8, pre aplikácie s veľkým množstvom cachovania dát znížte na 0.6 a zvýšte spark.memory.storageFraction.



Partitioning

Optimálny počet partícií je dôležitý pre paralelizmus. Príliš málo partícií nevyužíva dostupné zdroje, príliš veľa zvyšuje réziu. Použite repartition() alebo coalesce() na úpravu počtu partícií. Poznámka: Dobrým pravidlom je mať 2-3 partície na CPU jadro. Príklad: Pre cluster s 10 exekútormi, každý so 4 jadrami, je vhodné mať 80-120 partícií.



Persistencia a cache

Použite cache() alebo persist() pre DataFrame, ktoré sa používajú opakovane. Zvoľte vhodnú úroveň persistencie (MEMORY_ONLY, MEMORY_AND_DISK) podľa dostupných zdrojov a veľkosti dát. Poznámka: Pre kritické dátové sady použite persist(StorageLevel.MEMORY_AND_DISK_SER) na zníženie pamäťovej náročnosti. Nezabudnite volať unpersist() po dokončení operácií s danou sadou.



Broadcast join

Pre spojenia, kde jedna tabuľka je výrazne menšia, použite broadcast join. Spark automaticky aplikuje broadcast pre malé tabuľky, ale môžete to vynútiť pomocou broadcast() funkcie. Poznámka: Broadcast je efektívny pre tabuľky do ~10MB. Príklad: import org.apache.spark.sql.functions.broadcast; df1.join(broadcast(df2), "key"). Nastavte spark.sql.autoBroadcastJoinThreshold pre zmenu limitu automatického broadcastu.

Spark Performance Optimization



Catalyst Optimizer

Analýza dopytu

Catalyst analyzuje SQL dopyt alebo DataFrame operácie a vytvorí logický plán, ktorý reprezentuje požadované transformácie dát.

Poznámka: V tejto fáze sa analyzuje syntax a sémantika dopytu, kontrolujú sa názvy stĺpcov a tabuľiek, a vytvára sa abstraktná syntaktická stromová štruktúra (AST).

Logická optimalizácia

Optimalizátor aplikuje sadu pravidiel na transformáciu logického plánu - napríklad odstránenie nepotrebných operácií, zjednodušenie výrazov a predikátov.

Poznámka: Medzi typické optimalizácie patrí presun filtrov bližšie k zdroju dát (predicate pushdown), odstránenie mŕtveho kódu a konštantné skladanie. Tieto optimalizácie sú nezávislé od fyzickej implementácie.

Fyzická optimalizácia

Logický plán je prevedený na fyzický plán, ktorý špecifikuje konkrétné algoritmy a stratégie pre vykonanie operácií (napr. broadcast join vs. sort-merge join).

Poznámka: Optimalizátor zvažuje štatistiky, veľkosť dát a dostupné systémové zdroje pre výber najefektívnejšej implementácie. Cost-based optimizer používa matematické modely na predpovedanie výkonnosti rôznych stratégii.

Generovanie kódú

Spark generuje optimalizovaný bytecode pre kritické časti výpočtu, čo ďalej zrýchluje vykonanie.

Poznámka: Whole-stage code generation kombinuje viacero operátorov do jedného optimalizovaného kódu, čím sa znížuje rážia volania funkcií a zlepšuje sa využitie CPU cache. Generovaný kód využíva moderné JVM optimalizácie.



Spracovanie veľkých dát - výzvy a riešenia

Dátová šikmost (Data Skew)

Dátová šikmost nastáva, keď niektoré partície obsahujú výrazne viac dát než ostatné, čo viedie k nerovnomernému zaťaženiu uzlov a spomaleniu celého výpočtu.

Riešenia:

- Salting - pridanie náhodného prefixu k join klúču
- Použitie custom partitioneru
- Predspracovanie dát na odstránenie šikmosti

Pamäťové pretečenie (OOM)

Out of Memory chyby sú časté pri spracovaní veľkých dát, najmä pri operáciách vyžadujúcich shuffle alebo pri použití collect() na veľké datasety.

Riešenia:

- Zvýšenie pamäte exekútorov (spark.executor.memory)
- Použitie disk spill (spark.memory.storageFraction)
- Nahradenie collect() alternatívami ako take() alebo foreach()
- Filtrovanie a agregácia dát pred náročnými operáciami

Spark Streaming - úvod



Zdroje dát

Spark Streaming môže prijímať dáta z rôznych zdrojov ako Kafka, Flume, Kinesis, alebo TCP socketov. Tieto dáta prichádzajú v reálnom čase ako kontinuálny prúd.



Spracovanie

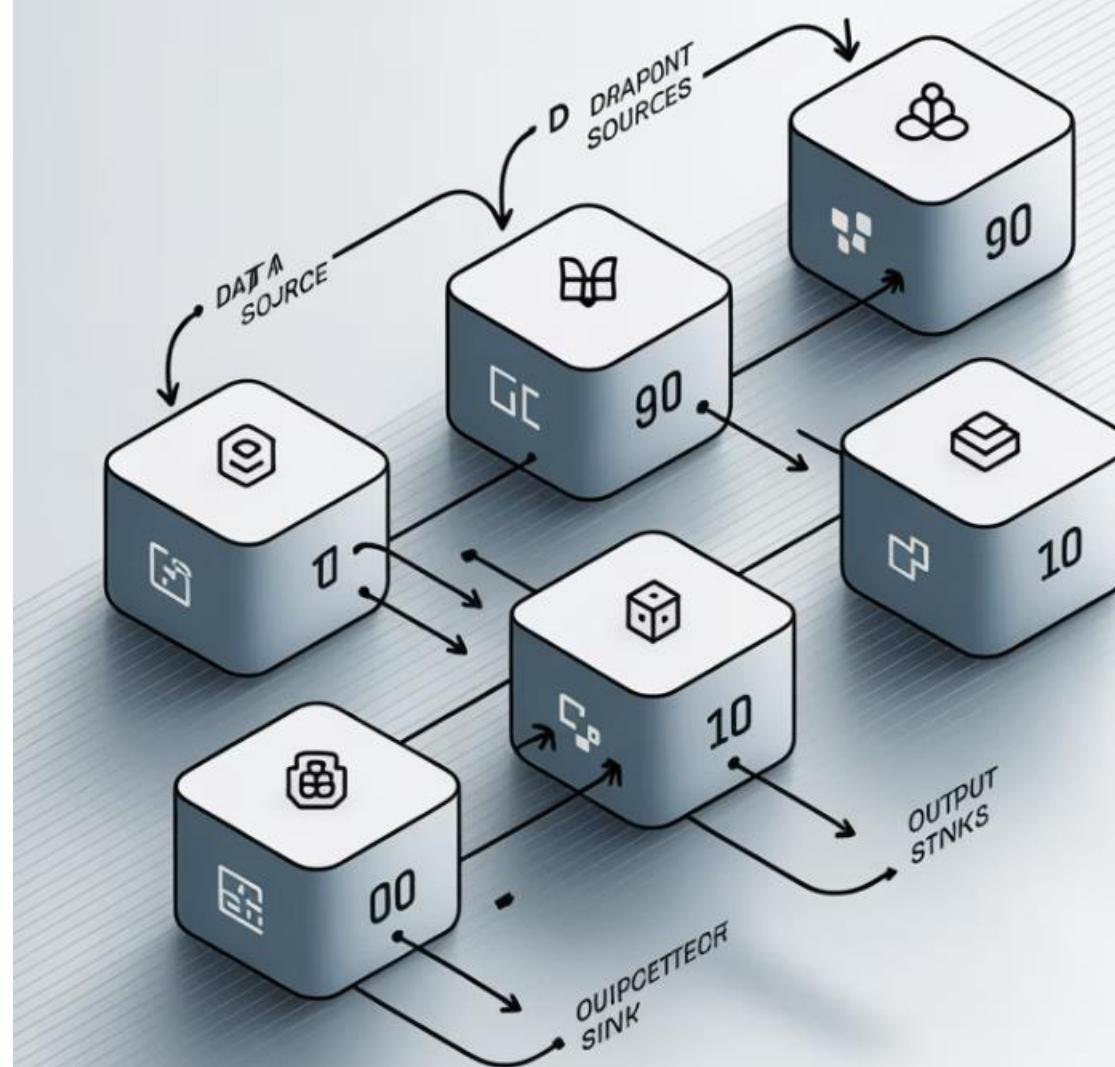
Prichádzajúce dáta sú rozdelené do mikro-dávok (micro-batches), ktoré sú spracovávané pomocou Spark enginu. Môžeme aplikovať transformácie, agregácie a analýzy podobne ako na statické dáta.



Výstupy

Výsledky spracovania môžu byť uložené do rôznych úložísk ako HDFS, databázy, alebo môžu byť publikované do ďalších streamovacích systémov pre následné spracovanie.

Spark Streaming Architecture



Structured Streaming - moderný prístup

Nekonečná tabuľka

Structured Streaming reprezentuje stream ako nekonečnú tabuľku, kde nové dátá sú pridávané ako nové riadky

Poznámka: Tento koncepcný model uľahčuje pochopenie, pretože vývojári môžu uvažovať o streamoch podobne ako o statických dátach, čo zjednodušuje prechod od batch processingu.

Garantovaná konzistencia

Exactly-once sémantika zabezpečuje spoľahlivé spracovanie aj pri zlyhaní

Poznámka: Táto vlastnosť je kritická pre finančné a podnikateľské aplikácie, kde duplicitné alebo stratené záznamy môžu viesť k vážnym následkom a nesprávnym výsledkom analýz.



Deklaratívne dotazy

Definujeme transformácie a agregácie pomocou DataFrame API alebo SQL

Poznámka: Využívanie rovnakého API ako pri batch spracovaní výrazne znižuje learning curve a umožňuje jednoduché zdieľanie kódu medzi batch a streaming aplikáciami.

Inkrementálne vykonanie

Spark automaticky vykonáva dotazy inkrementálne, spracovávajúc len nové dátá

Poznámka: Toto optimalizuje výkon a efektivitu, pretože nie je potrebné znova spracovávať všetky historické dátá pri každej aktualizácii, čo umožňuje škálovateľné riešenia.

Spark MLLib

Strojové učenie

Príprava dát

MLlib poskytuje nástroje na čistenie, transformáciu a prípravu dát pre strojové učenie. Zahŕňa funkcie pre normalizáciu, one-hot encoding, extrakciu features a ďalšie.

Trénovanie modelov

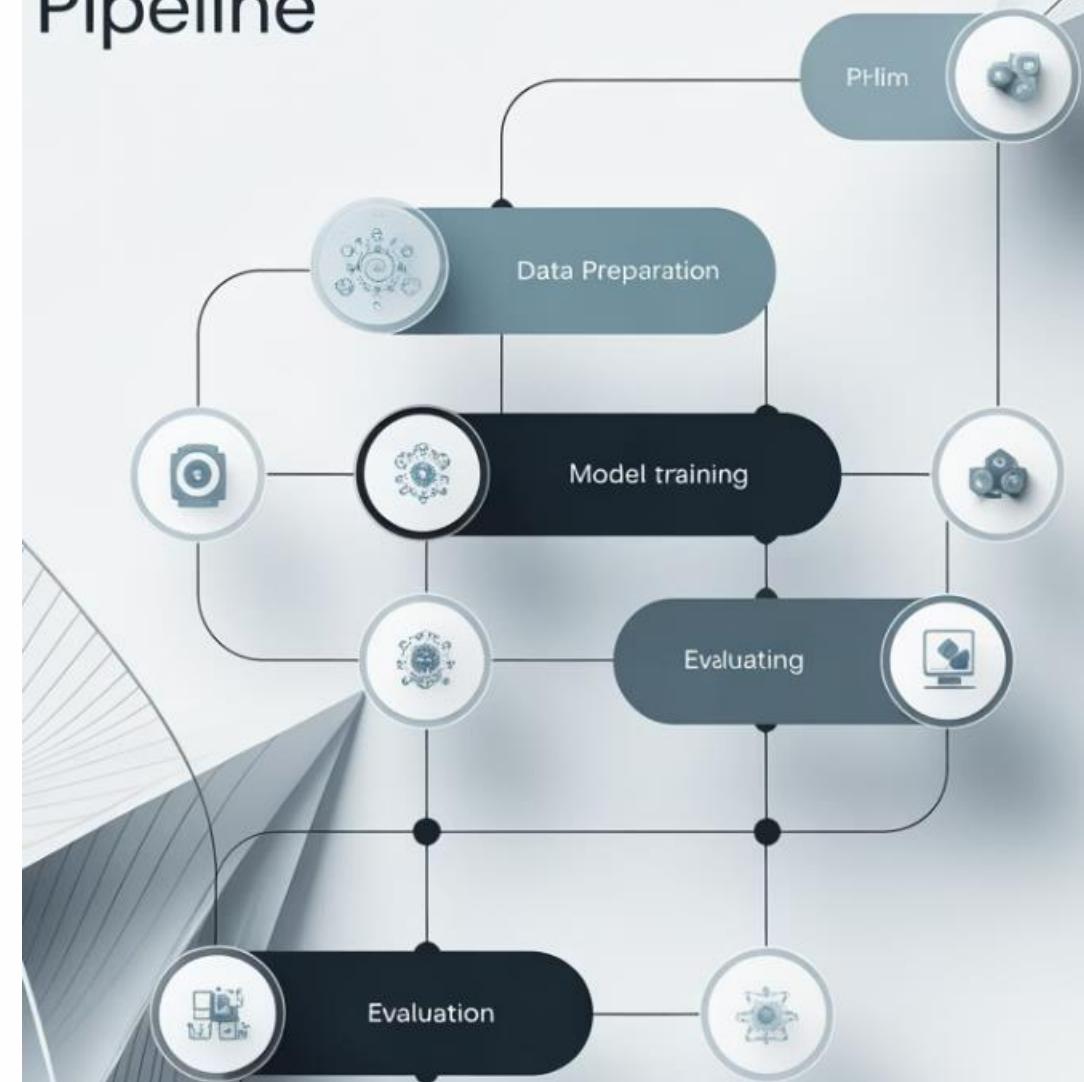
Spark podporuje široké spektrum algoritmov strojového učenia vrátane klasifikácie (logistická regresia, random forest), regresie, klastrovania (K-means) a kocvičneoratívneho filtrovania.

Evaluácia a nasadenie

MLlib poskytuje metriky pre hodnotenie modelov a umožňuje jednoduché nasadenie natrénovaných modelov do produkčného prostredia.

Spark MLLib

Machine Learning Pipeline



Spark GraphX – spracovanie grafov



Reprezentácia grafov

GraphX reprezentuje grafy ako RDD vrcholov a hrán, umožňujúc efektívne distribuované výpočty nad grafovými štruktúrami.



Grafové algoritmy

Knižnica poskytuje implementácie populárnych grafových algoritmov ako PageRank, Connected Components a Triangle Counting.



Aplikácie

GraphX je vhodný pre analýzu sociálnych sietí, odporúčacie systémy, detekciu podvodov a ďalšie úlohy založené na vzťahoch medzi entitami.

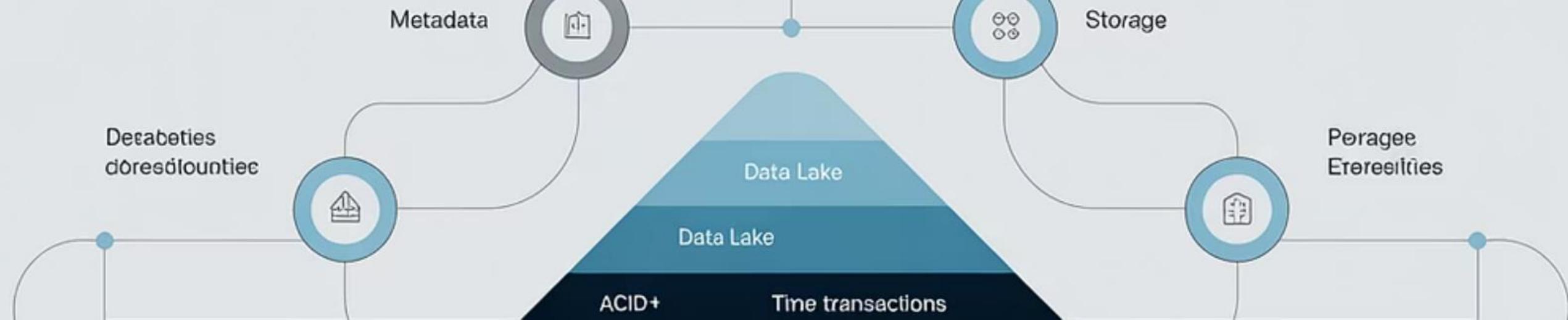


Integrácia s DataFrame

GraphX umožňuje konverziu medzi grafovou reprezentáciou a DataFrame, čo uľahčuje kombinovanie grafových a tabuľkových analýz.

GraphX vôle Spark Ecosystem





Delta Lake - spoločlivé dátové jazero

ACID transakcie

Delta Lake prináša ACID (Atomicity, Consistency, Isolation, Durability) transakcie do dátových jazier, zabezpečujúc konzistenciu dát aj pri súbežných zápisoch a zlyhaní systému.

Schéma a evolúcia

Automatické vynucovanie schémy zabezpečuje integritu dát, zatiaľ čo schéma evolution umožňuje bezpečne meniť štruktúru dát v čase bez narušenia existujúcich dotazov.

Time travel

Delta Lake umožňuje prístup k predchádzajúcim verziám dát, čo je užitočné pre audit, rollback po chybách a porovnanie zmien v čase.

Optimalizácia výkonu

Zahŕňa techniky ako compaction, indexing a caching pre zrýchlenie dotazov a zníženie nákladov na úložisko.

Spark na Kubernetes

Výhody Kubernetes pre Spark

Kubernetes poskytuje flexibilné a škálovateľné prostredie pre nasadenie Spark aplikácií. Medzi hlavné výhody patrí:

- Dynamická alokácia zdrojov - pridelenie a uvoľnenie exekútorov podľa potreby
- Izolácia aplikácií - každá Spark aplikácia beží vo vlastnom namespace
- Jednoduchá správa závislostí - kontajnery zabezpečujú konzistentné prostredie
- Integrácia s existujúcou infraštruktúrou - využitie existujúcich Kubernetes clusterov

Nasadenie Spark na Kubernetes

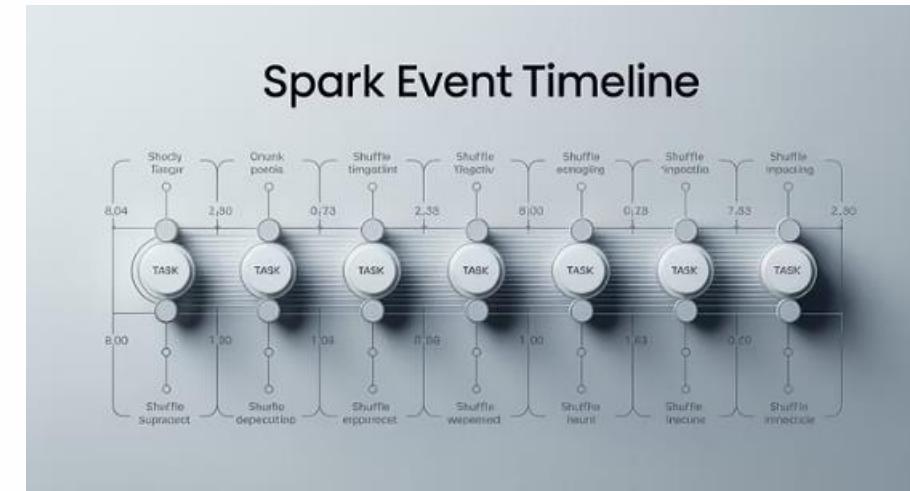
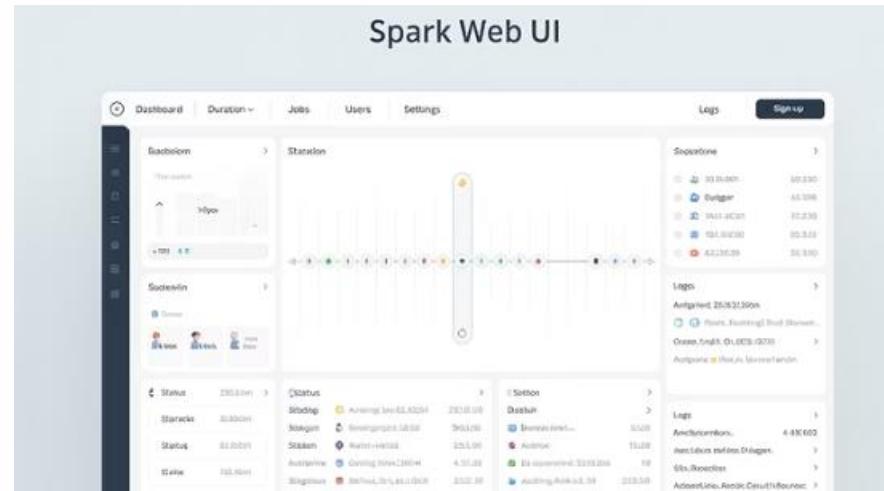
Spark môže byť nasadený na Kubernetes v dvoch režimoch:

Cluster mode: Driver aj exekútori bežia ako Kubernetes pody. Tento režim je vhodný pre produkčné nasadenie.

```
spark-submit \
--master k8s://https://kubernetes-api:443 \
--deploy-mode cluster \
--conf
spark.kubernetes.container.image=spark:latest \
--class org.example.SparkApp \
local:///opt/spark/examples/jars/spark-
examples_2.12-3.4.0.jar
```

Client mode: Driver beží mimo Kubernetes clustera, len exekútori sú spustené ako pody. Vhodné pre vývoj a debugging.

Monitorovanie a ladenie Spark aplikácií



Spark poskytuje rozsiahle nástroje pre monitorovanie a ladenie aplikácií. Spark UI zobrazuje detaily o joboch, stagoch a taskoch, vrátane časov vykonania a množstva prenášaných dát. Môžeme sledovať využitie pamäte a CPU, identifikovať úzke miesta a optimalizovať výkon. Pre dlhodobé monitorovanie je vhodné integrovať Spark s nástrojmi ako Prometheus a Grafana.

Bezpečnosť v Apache Spark



Autentifikácia

Overenie identity používateľov a služieb



Autorizácia

Kontrola prístupu k dátam a operáciám

3

Šifrovanie

Ochrana dát počas prenosu a v úložisku



Audit

Logovanie a monitorovanie aktivít

Autentifikácia: Apache Spark podporuje rôzne metódy autentifikácie vrátane Kerberos, LDAP a vlastných riešení. Implementácia silnej autentifikácie je kľúčová pre zabezpečenie prístupu do clustra a prevenciu neoprávneného využívania výpočtových zdrojov.

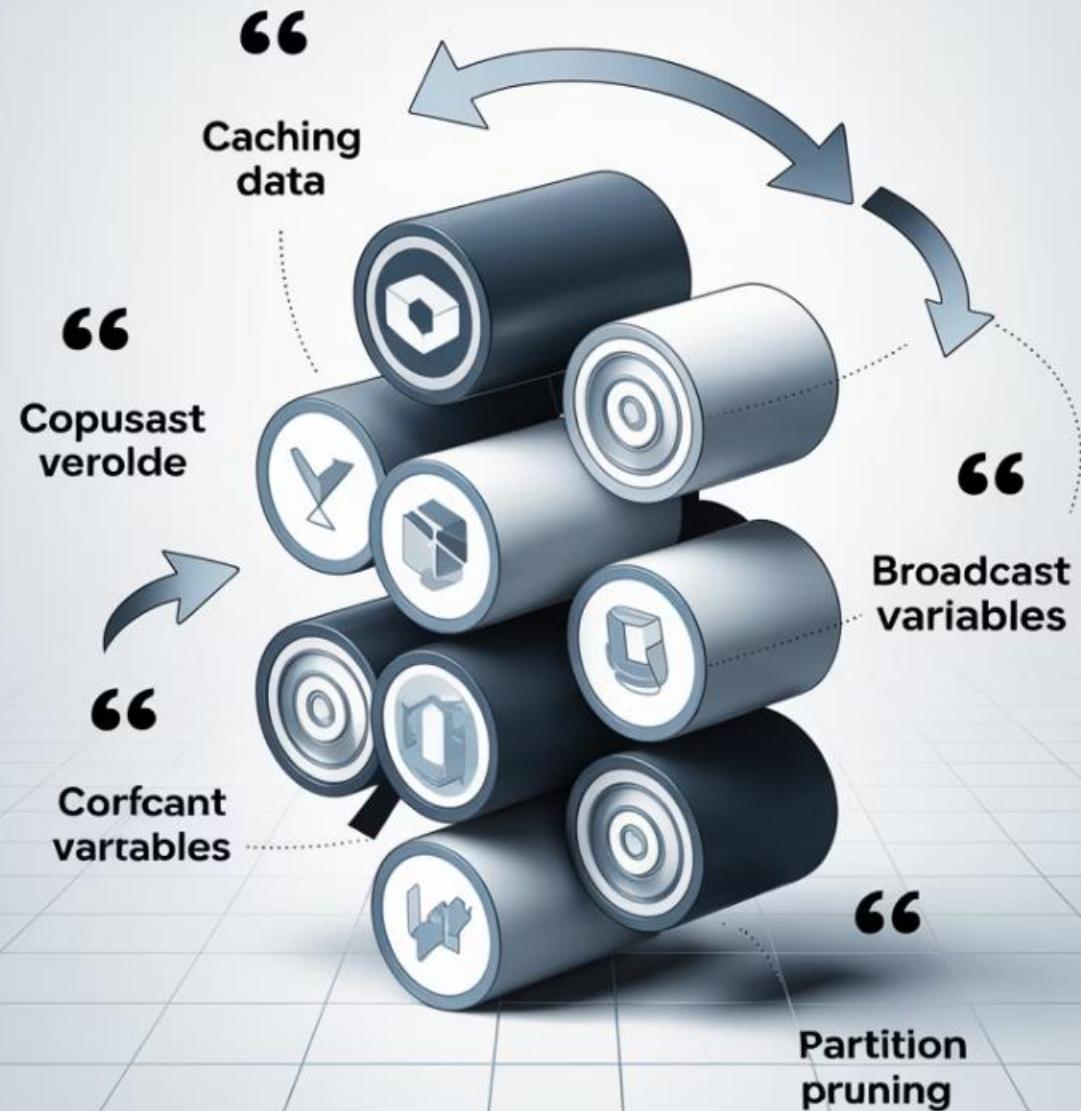
Autorizácia: Po úspešnej autentifikácii Spark umožňuje definovať pravidlá pre prístup k dátam a funkciám. Môžeme využiť integráciu s Apache Ranger alebo implementovať vlastné kontroly prístupu na úrovni tabuľiek, stĺpcov alebo záznamov.

Šifrovanie: Spark podporuje šifrovanie dát počas prenosu (SSL/TLS) aj v pokoji. Šifrovanie RPC komunikácie medzi komponentami Sparku, ako aj šifrovanie dočasných súborov a shuffle dát, významne zvyšuje bezpečnosť celého ekosystému.

Audit: Komplexné logovanie všetkých operácií umožňuje sledovať kto, kedy a k akým dátam pristupoval. Integrácia s nástrojmi ako Elasticsearch alebo Splunk poskytuje pokročilé možnosti analýzy bezpečnostných udalostí a včasnej detekcie potenciálnych hrozieb.

Spark Best Practices

Optimized for performance Partitioning



Odporúčané praktiky pre Spark aplikácie

Optimalizácia kódu

Používajte DataFrame namiesto RDD kde je to možné. Minimalizujte počet shuffle operácií. Využívajte lazy evaluation na optimalizáciu výpočtového plánu.

Poznámka: DataFrame API poskytuje optimalizácie pomocou Catalyst optimizéra, ktorý dokáže automaticky reorganizať operácie. Shuffle operácie sú náročné na sieťovú komunikáciu a často spôsobujú zníženie výkonu. Lazy evaluation umožňuje Sparku vytvoriť optimálny plán vykonávania pred samotným spustením.

Správa zdrojov

Správne dimenzujte pamäť a CPU pre exekútry. Využívajte dynamickú alokáciu zdrojov. Monitorujte využitie zdrojov a identifikujte úzke miesta.

Poznámka: Príliš veľké exekútry spôsobujú neefektívne využitie pamäte a časté garbage collection pauzy. Príliš malé exekútry zvyšujú réžiu. Dynamická alokácia automaticky prispôsobuje počet exekútorov aktuálnym potrebám. Spark History Server poskytuje detailné informácie o využití zdrojov po dokončení úlohy.

Správa dát

Používajte efektívne formáty ako Parquet alebo ORC. Partitioning a bucketing pre veľké datasety. Pravidelná údržba (compaction) pre optimálny výkon.

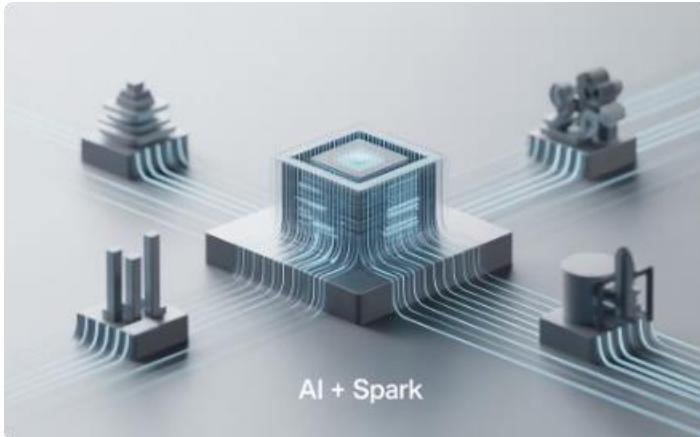
Poznámka: Parquet a ORC podporujú stĺpcové uloženie, kompresiu a predikát pushdown. Partitioning podľa často používaných filtrovacích stĺpcov redukuje množstvo čítaných dát. Bucketing zlepšuje výkon join operácií. Pravidelná compaction malých súborov je dôležitá pre odstránenie degradácie výkonu spôsobenej príliš veľkým množstvom malých súborov.

Debugging a testovanie

Testujte na malých datasetoch pred nasadením na veľké dátá. Používajte Spark UI na identifikáciu problémov. Implementujte robustné spracovanie chýb.

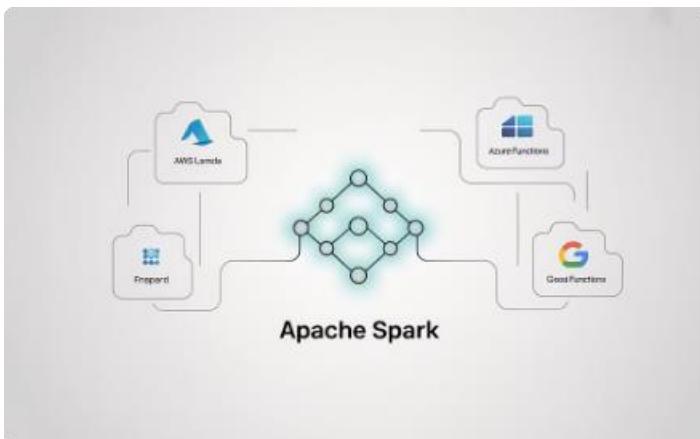
Poznámka: Testovanie na reprezentatívnej vzorke dát odhalí problémy skôr, než sa minú zdroje na veľkých datasetoch. Spark UI zobrazuje DAG grafické znázornenie úloh, ktoré pomáha identifikovať neefektívne operácie. Ošetrenie chýb by malo zahŕňať stratégii pre spracovanie zlých záznamov (napr. pomocou Dataset.exceptAll) a implementáciu retry mechanizmov pre dočasné zlyhania.

Budúcnosť Apache Spark



Integrácia s AI

Spark sa čoraz viac integruje s modernými AI frameworkami ako TensorFlow a PyTorch. Projekt Spark AI prináša natívnu podporu pre distribuované trénovanie deep learning modelov a inferenčné pipeline.



Cloud-native architektúra

Vývoj smeruje k plne cloud-native implementácii s podporou pre serverless computing, kde používatelia platia len za skutočne využité výpočtové zdroje bez potreby správy infraštruktúry.



Real-time spracovanie

Continuous processing mode v Structured Streaming znižuje latenciu na milisekundy, čo umožňuje skutočne real-time analýzy a reakcie na udalosti prakticky okamžite po ich vzniku.

Apache Spark Learning Path



Zdroje pre ďalšie štúdium



Knihy

Learning Spark (Holden Karau, et al.),
Spark: The Definitive Guide (Bill Chambers,
Matei Zaharia), High Performance Spark
(Holden Karau, Rachel Warren)

Poznámka: Knihy poskytujú hĺbkové pochopenie architektúry a implementácie. Odporúčame začať s "The Definitive Guide" pre komplexný prehľad a pokračovať s "High Performance Spark" pre optimalizáciu výkonu.



Online kurzy

VITA Academy, Databricks Academy,
Coursera: Big Data Analysis with Spark,



Komunita

Stack Overflow, Apache Spark mailing list,
Meetup skupiny, konferencie: Spark
Summit, Strata Data Conference

Poznámka: Aktívna účasť v komunite urýchľuje učenie. Stack Overflow je výborný pre riešenie konkrétnych problémov, zatiaľ čo konferencie ako Spark Summit predstavujú najnovšie trendy a umožňujú networking s expertmi.



Dokumentácia

Oficiálna dokumentácia Apache Spark,
Databricks dokumentácia a blogy, GitHub
repozitáre s príkladmi

Poznámka: Oficiálna dokumentácia je nevyhnutná pre referencie API, zatiaľ čo Databricks blogy často vysvetľujú najnovšie funkcie a best practices. GitHub repozitáre poskytujú praktické príklady implementácií v rôznych scenároch.