DĚLAT
DOBRÝ SOFTWARE
NÁS BAVÍ

PROFINIT

Spark SQL, Spark Streaming

Jan Hučín

20. listopadu 2019

Osnova

1. Spark SQL
2. Další rozšíření Sparku
   - Spark streaming
   - GraphX
   - Spark ML

PROFINIT

Spark SQL

Spark SQL – proč?

- Tradiční RDD přístup na strukturovaná data je nešikovný
- Příklad – výpočet státu s nejvyšší prům. teplotou v létě:

```
AQW00061705,7,30,4,804,P,-14.3306,-170.7136,3.7,AS,PAGO PAGO WSO AP
(stat, (teplota, 1))
(stat, (soucet=sum(teplota), pocet=sum(1)))
(stat, (prumer=soucet/pocet))
seřadit, vypsat první
```

PROFINIT

Spark SQL a DataFrames (DataSets)

Datová struktura **DataFrame** = RDD se sloupci

- obdoba databázové relační tabulky
- obsahuje i schéma
- nad rámec RDD – práce se sloupci
- možnost použití syntaxe podobné SQL nebo přímo SQL

1;Andrea;35;64.3;Praha
2;Martin;43;87.1;Ostrava
3;Simona;18;57.8;Brno

id	jmeno	vek	hmotnost	mesto
1	Andrea	35	64.3	Praha
2	Martin	42	87.1	Ostrava
3	Simona	18	57.8	Brno

PROFINIT

Spark SQL – výhody a nároky

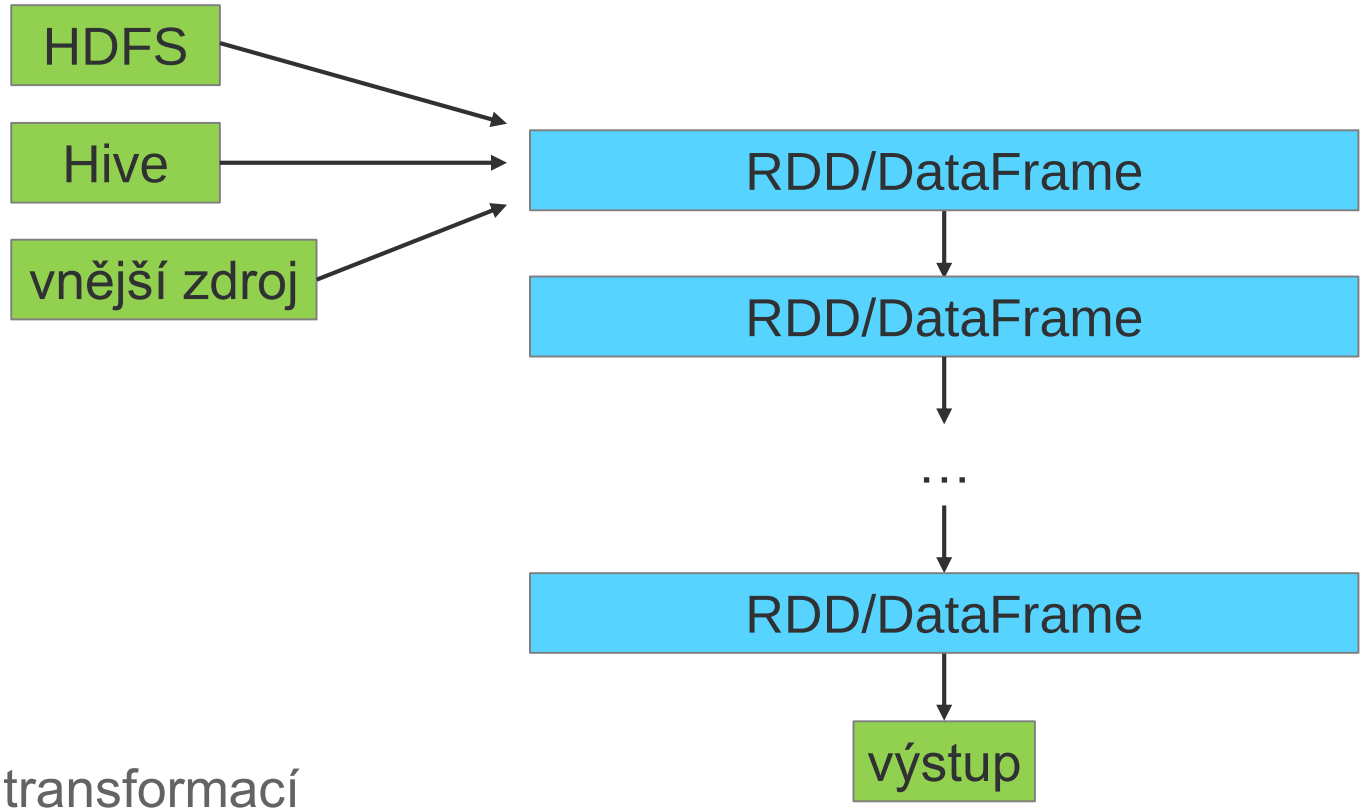
- Výhody oproti tradičnímu Sparku (RDD):
  - stručnější a jednodušší kód
  - využití Hive
  - snazší optimalizace
  - ⇨ rychlejší běh

- Nároky navíc:
  - rozšířené API: objekt **`sqlContext`**, ev. další

- Kdy nelze použít?
  - úlohy nevhodné pro SQL ⇨ tradiční Spark
  - úlohy náročné na paměť ⇨ map-reduce, Hive

PROFINIT

Spark RDD a SQL



- série transformací zakončená akcí
- lze transformovat RDD na DataFrame a obráceně

PROFINIT

Spark RDD a SQL

- Transformace RDD → RDD
  - už známe: map, flatMap, filter, …
- DataFrame → DataFrame
  RDD → DataFrame
  DataFrame → RDD
  - naučíme se

PROFINIT

Jak vyrobit DataFrame?

- transformace z existujícího RDD
  - je-li převoditelné do sloupců
- přímé načtení souboru
  - s již definovanými sloupci (např. Parquet, ORC)
  - převoditelné do sloupců (např. CSV)
- výsledek dotazu do Hive
- výsledek dotazu do jiné DB (JDBC konektor)

Jak vyrobit DataFrame?

- transformace z existujícího RDD
  - `sqlContext.createDataFrame(RDD[, schema])`
- přímé načtení souboru
  - `sqlContext.read.format(formát).load(cesta)`
- výsledek dotazu do Hive
  - `sqlContext.sql(dotaz_sql)`

PROFINIT

Příklad

Data z meteostanic USA
(již jsme využili pro Hive a Spark RDD)

Struktura dat:

```
stanice,mesic,den,hodina,teplota,flag,latitude,longitude,vyska,stat,nazev
AQW00061705,1,1,1,804,P,-14.3306,-170.7136,3.7,AS,PAGO PAGO WSO AP
AQW00061705,1,2,1,804,P,-14.3306,-170.7136,3.7,AS,PAGO PAGO WSO AP
AQW00061705,1,3,1,803,P,-14.3306,-170.7136,3.7,AS,PAGO PAGO WSO AP
AQW00061705,1,4,1,802,P,-14.3306,-170.7136,3.7,AS,PAGO PAGO WSO AP
AQW00061705,1,5,1,802,P,-14.3306,-170.7136,3.7,AS,PAGO PAGO WSO AP
```

PROFINIT

Postup 1 (přímé načtení CSV → DataFrame)

```
tpDF1 = sqlContext.read \
        .format("com.databricks.spark.csv") \
        .option("header", "true") \
        .option("delimiter", ",") \
        .option("inferSchema", "true") \
        .load("/user/pascepet/data/teplota")
```

Postup 2 (Hive → DataFrame)

```
tpDF2 = sqlContext.sql('select * from fel_bigdata.teplota')
```

PROFINIT

Postup 3 (RDD → DataFrame)

- řádek → Row(pole1=hodnota, pole2=hodnota, ...)
- Row – něco jako *list*, umožňuje spojit data do sloupců
- CreateDataFrame si určí typy podle hodnot v Row
- můžeme ale při konverzi definovat i vlastní schéma

```
from pyspark.sql import Row
tp_prep = tp_raw.map(uprav_radek_df)
tpDF3 = sqlContext.createDataFrame(tp_prep)
```

PROFINIT

Jak pracovat s DataFrame?

1. registrovat jako dočasnou tabulku + dotazování SQL – výsledek je opět DataFrame
2. pseudo-SQL operace – výsledek je opět DataFrame
3. operace RDD – výsledek může být jen obyčejné RDD

Jak pracovat s DataFrame?

1. registrovat jako dočasnou tabulku + dotazování SQL
   - `DF.registerTempTable("tabulka")`
   - `sqlContext.sql("select * from tabulka")`
2. pseudo-SQL operace
   - `DF.operace`, např. select, filter, join, groupBy, sort...
3. operace RDD – výsledek může být jen obyčejné RDD
   - např. map, flatMap...
   - řádek v DataFrame je typu **Row** – práce jako s typem **list**

PROFINIT

Registrace dočasné tabulky + SQL

```
tpDF.registerTempTable("teploty")

tp_stDF = sqlContext.sql("""select stat, avg(teplota) as
tepl_prum from teploty
where mesic in (6,7,8)
group by stat order by tepl_prum desc""")

tp_stDF.show(1)
```

PROFINIT

Pseudo-SQL a další operace

- **select** (omezení na uvedené sloupce)
- **filter** (omezení řádků podle podmínky)
- **join** (připojení jiného DataFrame)
- **groupBy** (seskupení)
- **agg, avg, count** (agregační funkce)
- **toDF** (přejmenování sloupců)
- **withColumn** (transformace sloupců)
- **show** (hezčí výpis obsahu DataFrame)

PROFINIT

Totéž pomocí pseudo-SQL

```
tpDF.registerTempTable("teploty")

tp_stDF = sqlContext.sql("""select stat, avg(tepl) as
tepl_prum from teploty
where mesic in (6,7,8)
group by stat order by tepl_prum desc""")

tp_stDF.show(1)
```

```
tpDF2 = tpDF2.filter((tpDF2['mesic']>=6) & (tpDF2['mesic']<=8)) \
    .select('stat','teplota').dropna()
tpDF2 = tpDF2.groupBy('stat').avg() \
    .toDF('stat','tepl_prum')
tpDF2.orderBy(tpDF2['tepl_prum'].desc()).show(1)
```

PROFINIT

Spark Streaming

Co to je a jak to využít

- dávkové zpracování přicházejících dat
- příchozí data neklepou na dveře, sedí v čekárně
- near real-time, pevné nastavení časového okna

Možné využití:

- filtrování logů, zpráv
- monitorování, reakce na událost
- vyhledávání v nakešovaných datech

PROFINIT

Princip zpracování

- streamovací modul dávkuje příchozí data – posloupnost RDD
- klasický Spark postupně odbavuje RDD ve frontě
- API pro Javu, Scalu, s malým omezením i Python

Příklad

Úkol: pro každou dávku ze socketu spočítat četnosti slov

```
sc = SparkContext(appName="Priklad")
ssc = StreamingContext(sc, 10)

lines = ssc.socketTextStream("localhost", 9999)
counts = lines.flatMap(lambda line: line.split(" ")) \
                  .map(lambda word: (word, 1)) \
                  .reduceByKey(lambda a, b: a+b)

counts.pprint()

ssc.start()
ssc.awaitTermination()
```

PROFINIT

GraphX
Spark ML

GraphX

- rozšíření pro algoritmy prohledávající grafy
- ve stadiu vývoje
- připravené algoritmy:
  - PageRank
  - rozklad na podgrafy
  - počet trojúhelníků
  - label propagation
  - a další...
- API pro iterativní procházení grafů (Pregel)

PROFINIT

Spark ML (machine learning)

- klasické algoritmy machine learning, např.:
  - regrese, lineární modely
  - rozhodovací stromy
  - naivní Bayesův klasifikátor
  - shluková analýza
- algoritmy pro velká data, např.:
  - doporučovací systém
  - asociační pravidla, časté podmnožiny
- statistické metody, např.:
  - popisná statistika
  - testování hypotéz
- mnohorozměrné metody, např.:
  - hlavní komponenty
  - faktorová analýza

PROFINIT

Spark ML (machine learning)

- praktický smysl mají jen algoritmy pro velká data
- u ostatních metod:
  - z velkých dat se vybere vzorek
  - na vzorku se tradičními nástroji modeluje
  - navržený model se naprogramuje ve Sparku (bez Spark ML)

Díky za pozornost

PROFINIT

Profinit, s.r.o.
Tychonova 2, 160 00 Praha 6

Telefon
+ 420 224 316 016

Web
www.profinit.eu

LinkedIn
linkedin.com/company/profinit

Twitter
twitter.com/Profinit_EU