



TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies ■

TECHNOLOGIE PRO BIG DATA

CVIČENÍ V.

APACHE SPARK

Lukáš Matějů

2.11.2023 | TPB



PŘÍPRAVA CVIČENÍ

- Apache Spark
 - cvičení je založené na Docker [image](#) od Bitnami
 - pro simulaci clusteru je používán Docker Compose
 - jak na to?
 1. stáhněte si z elearningu Bitnami Spark docker-compose.yml
 2. soubor umístěte do libovolné pracovní složky
 3. soubor rozšiřte, k services spark, spark-worker-1 a spark-worker-2 přiřadte volumes

```
version: '2'

services:
  spark:
    image: docker.io/bitnami/spark:3
    environment:
      - SPARK_MODE=master
      - SPARK_RPC_AUTHENTICATION_ENABLED=no
      - SPARK_RPC_ENCRYPTION_ENABLED=no
      - SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
      - SPARK_SSL_ENABLED=no
    ports:
      - '8080:8080'
    volumes:
      - "./files:rw"
```

```
spark-worker-1:
  image: docker.io/bitnami/spark:3
  environment:
    - SPARK_MODE=worker
    - SPARK_MASTER_URL=spark://spark:7077
    - SPARK_WORKER_MEMORY=1G
    - SPARK_WORKER_CORES=1
    - SPARK_RPC_AUTHENTICATION_ENABLED=no
    - SPARK_RPC_ENCRYPTION_ENABLED=no
    - SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
    - SPARK_SSL_ENABLED=no
  volumes:
    - "./files:rw"
```

PŘÍPRAVA CVIČENÍ

- Apache Spark
 - jak na to?
 3. soubor rozšiřte, k services spark, spark-worker-1 a spark-worker-2 přiřadte volumes
volumes:
 - "./files:rw"
 - soubory budou v Dockeru přístupné pod /files
 - simuluje distribuovaný souborový systém
 4. přesuňte se v konzoli do vybrané pracovní složky a spusťte Docker Compose
docker-compose up -d
 - 5. přepněte se do běžícího kontejneru mastera
docker exec -i -t spark_spark_1 /bin/bash

```
D:\Prezentace\ostatní\TPB\Spark>docker-compose up -d
Recreating spark_spark-worker-2_1 ... done
Recreating spark_spark_1 ... done
Recreating spark_spark-worker-1_1 ... done
```

```
D:\>docker exec -i -t spark_spark_1 /bin/bash
I have no name!@fa367db42f31:/opt/bitnami/spark$
```

PŘÍPRAVA CVIČENÍ

- Apache Spark
 - funguje vše jak má?
 1. v kontejneru se přesuňte do /files a vypište si obsah adresáře
 - cd /files
 - ls
 - pokud nevidíte soubor docker-compose.yml, něco je špatně
 - 2. v prohlížeči přejděte na adresu <http://localhost:8080/> a zkopírujte si adresu mastera

```
I have no name!@fa367db42f31:/opt/bitnami/spark$ cd /files/  
I have no name!@fa367db42f31:/files$ ls  
docker-compose.yml  lines.txt  ml-100k  ratings-counter.py
```



Spark Master at spark://fa367db42f31:7077

URL: spark://fa367db42f31:7077

Alive Workers: 2

Cores in use: 2 Total, 0 Used

Memory in use: 2.0 GiB Total, 0.0 B Used

Resources in use:

Applications: 0 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

PŘÍPRAVA CVIČENÍ

- Apache Spark
 - funguje vše jak má?
 3. v kontejneru se přesuňte do /bin a spusťte spark shell na masterovi

```
cd /bin
spark-shell spark://fa367db42f31:7077
```

 - pokud se nedostanete do Scala Spark konzole, něco je špatně

```
:quit
pyspark --master spark://fa367db42f31:7077
```

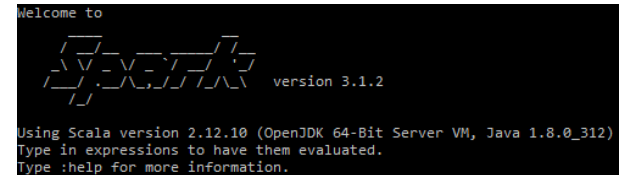
 - pokud se nedostanete do Python Spark konzole, něco je špatně
 4. zadejte cvičný program počítající počet řádků souboru (PySpark)

```
rdd = sc.textFile("files/docker-compose.yml")
rdd.count()
quit()
```

 - pokud nedostanete výsledek, něco je špatně
 5. z elearningu si stáhněte balík souborů ke cvičení a rozbalte je do vybrané pracovní složky

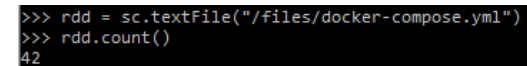
```
cd /files
ls
```

 - pokud soubory nevidíte, něco je špatně

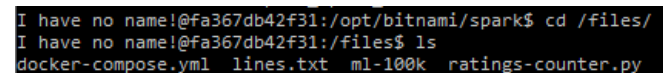


```
Welcome to
      _ _ _ _ _
     / _ _ _ \   version 3.1.2
    / _ _ _ \
   / _ _ _ \
  / _ _ _ \
 / _ _ _ \
/_ _ _ _ \

Using Scala version 2.12.10 (OpenJDK 64-Bit Server VM, Java 1.8.0_312)
Type in expressions to have them evaluated.
Type :help for more information.
```



```
>>> rdd = sc.textFile("/files/docker-compose.yml")
>>> rdd.count()
42
```



```
I have no name!@fa367db42f31:/opt/bitnami/spark$ cd /files/
I have no name!@fa367db42f31:/files$ ls
docker-compose.yml  lines.txt  ml-100k  ratings-counter.py
```

PŘÍPRAVA CVIČENÍ

- Apache Spark
 - funguje vše jak má?
 - 6. otevřete soubor *ratings.counter.py* a nahradte master svým masterem, zkontrolujte cestu

```
conf = SparkConf().setMaster("spark://8dfb04f99b96:7077").setAppName("RatingsHistogram")
sc = SparkContext(conf = conf)

lines = sc.textFile("/files/ml-100k/u.data")
```

- 7. spusťte skript
spark-submit ratings-counter.py
 - skript využívá [data](#) hodnocení filmů uživateli ([MovieLens](#)) a počítá počet výskytů jednotlivých typů hodnocení ([dataset](#))

```
21/10/25 08:20:22 INFO DAGScheduler: Job 0 finished: countByValue at /files/ratings-counter.py:9, took 4.611650 s
1 6110
2 11370
3 27145
4 34174
5 21201
```

- v případě, že nedošlo k žádné chybě a obdrželi jste výsledky, Apache Spark je připravený k použití

PŘÍPRAVA CVIČENÍ

- Apache Spark
 - jak udělat Spark méně ukecaný?
 - přesuňte se do adresáře /opt/bitnami/spark/conf
`cd /opt/bitnami/spark/conf`
 - otevřete soubor log4j2.properties.template
 - např. přesunem do /files a úpravou ve vybraném textovém editoru na vašem PC
 - pozměňte řádek rootLogger.level
 - z info na warn
 - ```
rootLogger.level = warn
```
  - soubor uložte pod názvem log4j2.properties
  - umístěte jej zpátky do /opt/bitnami/spark/conf

```
Running BFS iteration# 1
Processing 8330 values.
Running BFS iteration# 2
Processing 220615 values.
Hit the target character! From 1 different direction(s).
```

# DNEŠNÍ CVIČENÍ

1. zjistěte teplotní maxima pro každou meteostanici v roce 1800
  - k dispozici máte soubory *1800.csv* (data) a *min-temperatures.py* (kód)
  - vhodně upravte kód z přednášky
    - výsledky vraťte ve stupních Celsia

```
ITE00100554,18000101,TMAX,-75,,,E,
ITE00100554,18000101,TMIN,-148,,,E,
GM000010962,18000101,PRCP,0,,,E,
EZE00100082,18000101,TMAX,-86,,,E,
EZE00100082,18000101,TMIN,-135,,,E,
```



# DNEŠNÍ CVIČENÍ

## 2. zjistěte počet výskytů jednotlivých slov v textovém souboru

- k dispozici máte soubory *book.txt* (data) a *word-count.py* (kód)
- vhodně rozšiřte kód z přednášky
  - jednotlivá slova knihy normalizujte za pomoci regulárních výrazů
    - převod na malá písmena a odstranění interpunkce
    - v praxi by bylo možné využít např. toolkit NLTK
  - výsledky vraťte seřazené od slov s nejvyšší počtem výskytů a vypište 20 nejčastějších slov

|       |      |       |     |      |     |           |     |
|-------|------|-------|-----|------|-----|-----------|-----|
| you:  | 1878 | of:   | 970 | is:  | 560 | s:        | 391 |
| to:   | 1828 | and:  | 934 | for: | 537 | i:        | 387 |
| your: | 1420 | that: | 747 | on:  | 428 | business: | 383 |
| the:  | 1292 | it:   | 649 | are: | 424 | can:      | 376 |
| a:    | 1191 | in:   | 616 | if:  | 411 | be:       | 369 |

- **BONUS:** skript aplikujte na texty stažené z portálu iDNES.cz
  - skript rozšiřte a vypište 20 nejčastějších slov v článcích o délce alespoň 6 znaků

# DNEŠNÍ CVIČENÍ

## 3. zjistěte celkovou výši objednávek pro každého zákazníka

- k dispozici máte soubor *customer-orders.csv* (data)
  - id zákazníka, id předmětu, zaplacená cena
- vytvořte skript vracející pro každého zákazníka celkovou utracenou částku
  - skript pojmenujte *total-spent-by-customer.py*
- **BONUS:** seznam vraťte seříděný podle celkové utracené částky
  - pro seřídění využijte RDD



<https://www.udemy.com/course/taming-big-data-with-apache-spark-hands-on/>