

**EKONOMICKÁ UNIVERZITA V BRATISLAVE  
FAKULTA HOSPODÁRSKEJ INFORMATIKY**

**MapReduce príklady**

**Zadanie z predmetu Big Data**

**2023**

**Bc. Ondrej Šima**

**Bc. Alena Stracenská**

# Obsah

Úvod	4
1 Spracovanie a analýza logov v Mapreduce	5
1.1 Použité dáta . . . . .	5
1.2 Tvorba <code>mapper.py</code> a <code>reducer.py</code> skriptov . . . . .	5
1.3 Spustenie oboch skriptov na Hadoop clusteri . . . . .	7
Záver	11
Zoznam použitej literatúry	12

## Zoznam obrázkov a tabuliek

Obrázok 1	Ukážka súboru s logmi, zdroj: [vlastné spracovanie] . . . . .	5
Obrázok 2	Vytvorenie adresára <code>halena</code> a nahranie súboru, zdroj: [vlastné spracovanie] . . . . .	7
Obrázok 3	Overenie umiestnenia súboru na <code>localhost:9870</code> , zdroj: [vlastné spracovanie] . . . . .	8
Obrázok 4	Príkazy na spustenie <code>mapper.py</code> a <code>reducer.py</code> skriptov, zdroj: [vlastné spracovanie] . . . . .	9
Obrázok 5	Vykonávanie jobov, zdroj: [vlastné spracovanie] . . . . .	9
Obrázok 6	Príkazy na zobrazenie output súboru, zdroj: [vlastné spracovanie] .	10
Obrázok 7	Vizualizácia výsledného súboru spracovaného MapReduce. Môžeme vidieť, že výsledok výskytu/aktivity najpočetnejšej IP adresy sa nám zhoduje aj v ostatných zadaniach, zdroj: [vlastné spracovanie]	10

# Úvod

MapReduce je programovací model pre paralelné spracovanie dát, ktorý bol pôvodne vyvinutý pre distribuované výpočty na veľkých datasetoch v spoločnosti Google. Hlavným cieľom MapReduce je zjednodušiť proces spracovania dát tým, že rozdeľuje dáta na menšie časti, ktoré sú spracované paralelne, a výsledky z týchto menších častí sú následne kombinované do výsledného výstupu.

V tomto zadaní si napíšeme vlastné `mapper.py` a `reducer.py` skripty, čo znamená, že budeme mať úplnú kontrolu nad tým, ako sú naše dáta spracované. To nám umožní prispôbiť MapReduce presne našim potrebám a zabezpečiť, že výsledky budú vyhovovať našim požiadavkám. Tento prístup nám dáva veľkú flexibilitu a umožňuje nám využiť plný potenciál MapReduce pre spracovanie dát.

Veríme, že tento dokument bude prínosný pre čitateľa nie len po teoretickej stránke, ale aj po praktickej, kde sa naučí ako možno spracovať dáta pomocou už spomínaného programovacieho modelu.

# 1 Spracovanie a analýza logov v Mapreduce

## 1.1 Použité dáta

Dáta, s ktorými sme pracovali, nie len teraz, ale aj v predošlých zadaniach, boli vo forme apačovských webových logov. Môžeme si ich stiahnuť na kaggle.com. Sú to pološtruktúrované dáta, ktorých veľkosť obmedzíme na 100 000 riadkov.



Obrázok 1: Ukážka súboru s logmi, zdroj: [vlastné spracovanie]

## 1.2 Tvorba mapper.py a reducer.py skriptov

Najskôr sme si vytvorili skript s názvom `mapper.py`, v ktorom sme si najprv nainštalovali modul `sys` a `datetime`, potom sme si rozdelili riadok, načítaný zo štandardného vstupu na jednotlivé polia a extrahovali z logu IP adresu a dátum. Následne sme si extrahovali dátum z časovej pečiatky a konvertovali ho na objekt `datetime`. IP adresu a správne naformátovaný čas v tvare DD-MM-YYYY sme oddelený tabulátorom vypísali do štandardného výstupu.

```
#!/usr/bin/python
```

```
#import modulov
```

```

import sys
from datetime import datetime

for line in sys.stdin:
    # rozdelenie riadku na jednotlivé polia
    fields = line.strip().split()

    # vyber IP adresy
    ip_address = fields[0]
    # extrahovanie datumu z casovej peciatky
    date_str = fields[3][1:12]
    # zmena objektu date z string na objekt datetime
    date = datetime.strptime(date_str, '%d/%b/%Y')
    # formatovanie datumu na DD-MM-YYY
    date_formatted = date.strftime('%d-%m-%Y')

    # vystup IP adresa a datum
    print('{0}\t{1}'.format(ip_address + '\t' + date_formatted, 1))

```

Po vytvorení skriptu `mapper.py` sme si vytvorili skript `reducer.py`. V `reducer.py` sme si museli taktiež importovať modul `sys`, potom sme si inicializovali premenné `current_key` a `current_count` na hodnotu 0. Potom sme každý riadok zo štandardného vstupu rozdělili cez tabulátor na a následne podmienke kontrolujeme, či sa líši aktuálne čítaná IP adresa od IP adresy z predchádzajúceho behu cyklu. V prípade ak sa líšia (a zároveň sa nejedná o prvú IP adresu zo zoznamu), vypíšeme celkový počet nájdených IP adries a nastavíme hodnotu aktuálneho kľúča a vynulujeme počítadlo. V opačnom prípade len inkrementujeme počítadlo.

```

#!/usr/bin/python

import sys

#inicializacia
current_key = None
current_count = 0

# pre kazdy riadok
for line in sys.stdin:
    # rozdel podla tabu
    key_value = line.strip().split('\t')

    # ak mame iny kluc ako predchadzajuci cyklus
    if key_value[0] != current_key:

```

```

# osetrenie prveho kluca
if current_key is not None:
    print('{0}\t{1}'.format(current_key, current_count))

# nastav novy kluc, reset pocitadla
current_key = key_value[0]
current_count = 0

# inkrementuj pocitadlo
current_count += 1

# osetrenie posledneho kluca
if current_key is not None:
    print('{0}\t{1}'.format(current_key, current_count))

```

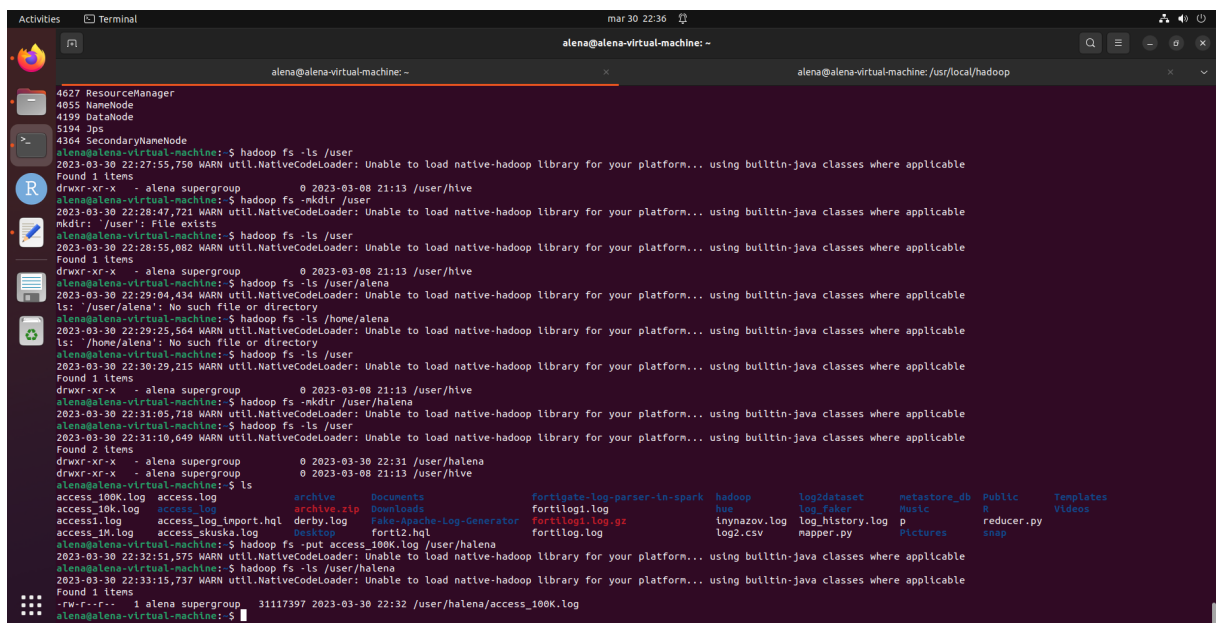
### 1.3 Spustenie oboch skriptov na Hadoop clusteri

Prvým krokom bolo vytvoriť si adresár halena, kde sme si následne nahrali súbor access\_100K.log:

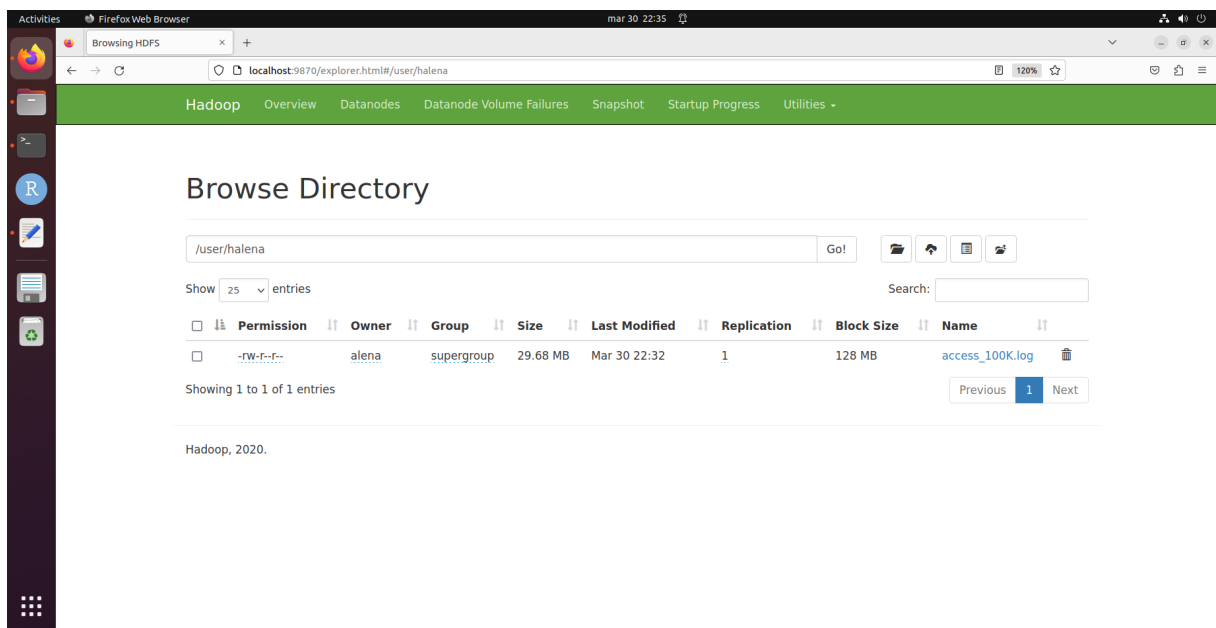
```

$hadop fs -mkdir /user/halena
$hadop fs -put access_100K.log /user/halena

```



Obrázok 2: Vytvorenie adresára halena a nahranie súboru, zdroj: [vlastné spracovanie]



**Obrázok 3:** Overenie umiestnenia súboru na localhost:9870,  
zdroj: [vlastné spracovanie]

Následne sme už len sledovali, ako Hadoop cez MapReduce vykonáva skripty. Python skripty pre mapper a reducer sú načítavané lokálne, vstup a výstup z HDFS.

```
mapred streaming -files mapper.py,reducer.py \  
-input /user/halena/access_100K.log \  
-output /user/halena/output \  
-mapper "python3 mapper.py" \  
-reducer "python3 reducer.py"
```



```
alena@alena-virtual-machine: ~
Deleted /user/halena/output
alena@alena-virtual-machine:~$ mapred streaming -files mapper.py, reducer.py \
-input /user/halena/access_100k.log \
-output /user/halena/output \
-mapper "python3 mapper.py" \
-reducer "python3 reducer.py"
2023-03-30 23:26:42,073 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
packageJobJar: [1] [/usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar] /tmp/streamjob4078869676104109296.jar tmpDir=null
2023-03-30 23:26:42,731 INFO client.DefaultHARMFalloverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2023-03-30 23:26:42,946 INFO client.DefaultHARMFalloverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2023-03-30 23:26:43,348 INFO mapreduce.JobResourceUploader: Disabling Erasure coding for path: /tmp/hadoop-yarn/staging/alena/.staging/job_1680207992063_0011
2023-03-30 23:26:44,216 INFO mapred.FileInputFormat: Total input files to process : 1
2023-03-30 23:26:44,690 INFO mapreduce.JobSubmitter: number of splits:2
2023-03-30 23:26:44,803 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1680207992063_0011
2023-03-30 23:26:44,803 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-03-30 23:26:44,949 INFO conf.Configuration: resource-types.xml not found
2023-03-30 23:26:44,949 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2023-03-30 23:26:45,021 INFO lnpl.VarnClientImpl: Submitted application application_1680207992063_0011
2023-03-30 23:26:45,064 INFO mapreduce.Job: The url to track the job: http://alena-virtual-machine:8080/proxy/application_1680207992063_0011/
2023-03-30 23:26:45,064 INFO mapreduce.Job: Running job: job_1680207992063_0011
2023-03-30 23:26:51,450 INFO mapreduce.Job: Job job_1680207992063_0011 running in uber mode : false
2023-03-30 23:26:51,452 INFO mapreduce.Job: map 0% reduce 0%
2023-03-30 23:26:57,669 INFO mapreduce.Job: map 100% reduce 0%
2023-03-30 23:27:01,702 INFO mapreduce.Job: map 100% reduce 100%
2023-03-30 23:27:02,720 INFO mapreduce.Job: Job job_1680207992063_0011 completed successfully
2023-03-30 23:27:02,806 INFO mapreduce.Job: Counters: 54
File System Counters
  FILE: Number of bytes read=2856312
  FILE: Number of bytes written=6516539
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=31121695
  HDFS: Number of bytes written=63377
  HDFS: Number of read operations=11
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=2
  HDFS: Number of bytes read erasure-coded=0
Job Counters
  Launched map tasks=2
  Launched reduce tasks=1
  Data-local map tasks=2
  Total time spent by all maps in occupied slots (ms)=7091
  Total time spent by all reduces in occupied slots (ms)=2202
  Total time spent by all map tasks (ms)=7091
  Total time spent by all reduce tasks (ms)=2202
```

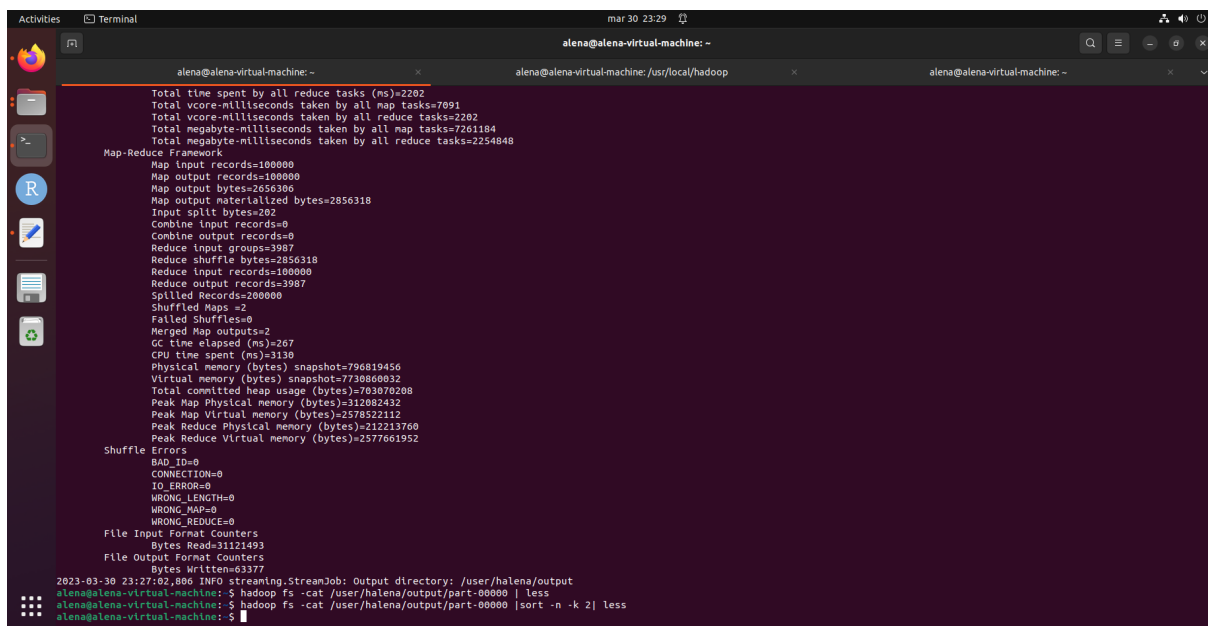
Obrázok 4: Príkazy na spustenie mapper.py a reducer.py skriptov,  
zdroj: [vlastné spracovanie]

```
alena@alena-virtual-machine: ~
HDFS: Number of large read operations=0
HDFS: Number of write operations=2
HDFS: Number of bytes read erasure-coded=0
Job Counters
  Launched map tasks=2
  Launched reduce tasks=1
  Data-local map tasks=2
  Total time spent by all maps in occupied slots (ms)=7091
  Total time spent by all reduces in occupied slots (ms)=2202
  Total time spent by all map tasks (ms)=7091
  Total time spent by all reduce tasks (ms)=2202
  Total vcore-millisecods taken by all map tasks=7091
  Total vcore-millisecods taken by all reduce tasks=2202
  Total megabyte-millisecods taken by all map tasks=7261184
  Total megabyte-millisecods taken by all reduce tasks=2254848
Map-Reduce Framework
  Map input records=100000
  Map output records=100000
  Map output bytes=2656306
  Map output materialized bytes=2856318
  Input split bytes=202
  Combine input records=0
  Combine output records=0
  Reduce input groups=3987
  Reduce shuffle bytes=2856318
  Reduce input records=100000
  Reduce output records=3987
  Spilled Records=260000
  Shuffled Maps =2
  Failed Shuffles=0
  Merged Map outputs=2
  GC time elapsed (ms)=267
  CPU time spent (ms)=3130
  Physical memory (bytes) snapshot=796819456
  Virtual memory (bytes) snapshot=773066032
  Total committed heap usage (bytes)=703070208
  Peak Map Physical memory (bytes)=312082432
  Peak Map Virtual memory (bytes)=2578522112
  Peak Reduce Physical memory (bytes)=212213760
  Peak Reduce Virtual memory (bytes)=2577661952
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
```

Obrázok 5: Vykonávanie jobov, zdroj: [vlastné spracovanie]

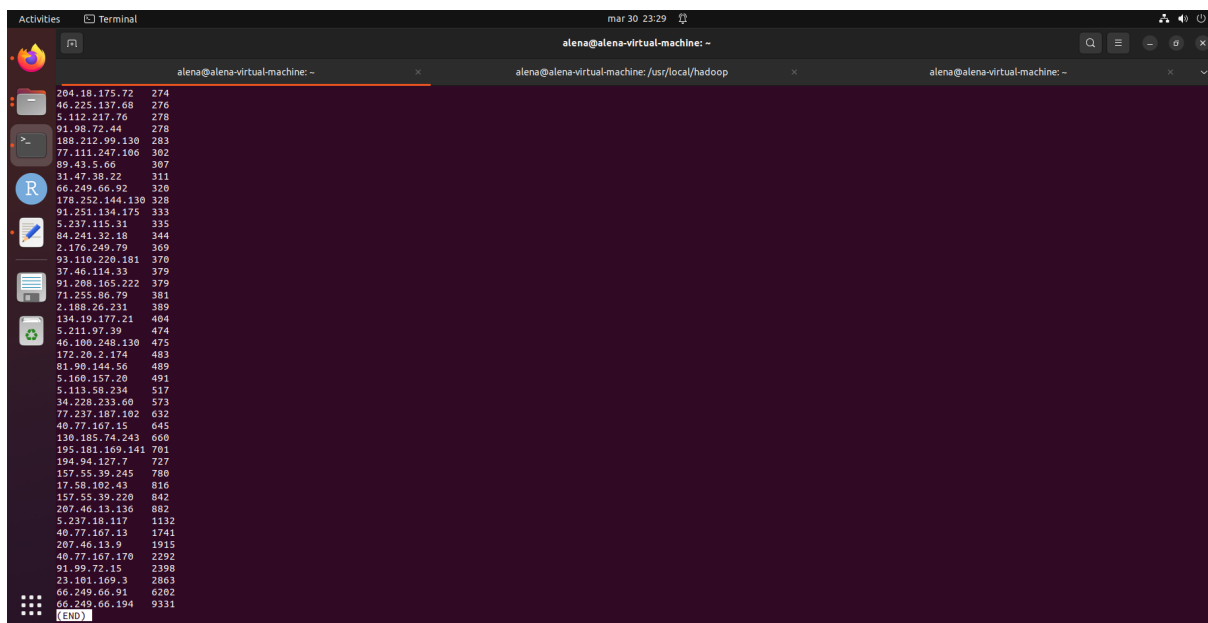
Výsledný output súbor sme si potom pozreli zosortovaný podľa hodnoty. [1]

```
hadoop fs -cat /user/halena/output/part-00000 |sort -n -k 2| less
```



```
alena@alena-virtual-machine: ~  
Total time spent by all reduce tasks (ms)=2202  
Total vcore-milliseconds taken by all map tasks=7091  
Total vcore-milliseconds taken by all reduce tasks=2202  
Total megabyte-milliseconds taken by all map tasks=7261184  
Total megabyte-milliseconds taken by all reduce tasks=2254848  
Map-Reduce Framework  
Map input records=100000  
Map output records=100000  
Map output bytes=2656306  
Map output materialized bytes=2856318  
Input split bytes=202  
Combine input records=0  
Combine output records=0  
Reduce input groups=3987  
Reduce shuffle bytes=2856318  
Reduce input records=100000  
Reduce output records=3987  
Spilled Records=200000  
Shuffled Maps=2  
Failed Shuffles=0  
Merged Map outputs=2  
GC time elapsed (ms)=267  
CPU time spent (ms)=3130  
Physical memory (bytes) snapshot=796819456  
Virtual memory (bytes) snapshot=7730860032  
Total committed heap usage (bytes)=703070208  
Peak Map Physical memory (bytes)=312082432  
Peak Map Virtual memory (bytes)=2578522112  
Peak Reduce Physical memory (bytes)=212213760  
Peak Reduce Virtual memory (bytes)=2577661952  
Shuffle Errors  
BAD_ID=0  
CONNECTION=0  
ID_ERROR=0  
WRONG_LENGTH=0  
WRONG_MAP=0  
WRONG_REDUCE=0  
File Input Format Counters  
Bytes Read=31121493  
File Output Format Counters  
Bytes Written=63377  
2023-03-30 23:27:02,006 INFO streaming.StreamJob: Output directory: /user/halena/output  
alena@alena-virtual-machine: $ hadoop fs -cat /user/halena/output/part-00000 | less  
alena@alena-virtual-machine: $
```

Obrázok 6: Príkazy na zobrazenie output súboru, zdroj: [vlastné spracovanie]



```
alena@alena-virtual-machine: ~  
204.18.175.72 274  
46.225.137.68 276  
5.112.217.76 278  
91.98.72.44 278  
188.212.99.130 283  
77.111.247.106 302  
89.43.5.66 307  
31.47.38.22 311  
66.249.66.92 320  
178.252.144.130 328  
91.251.134.175 333  
5.237.115.31 335  
84.241.32.18 344  
2.176.249.79 369  
93.110.220.181 370  
37.46.114.33 379  
91.208.165.222 379  
71.255.86.79 381  
2.188.26.231 389  
134.19.177.21 404  
5.211.97.39 474  
46.100.248.130 475  
172.20.2.174 483  
81.90.144.56 489  
5.168.157.20 491  
5.113.50.234 517  
34.228.233.60 573  
77.237.187.102 632  
48.77.167.15 645  
130.185.74.243 660  
195.181.169.141 701  
194.94.127.7 727  
157.35.39.245 780  
17.50.102.43 816  
157.55.39.220 842  
207.46.13.136 882  
5.237.18.117 1132  
48.77.167.13 1741  
207.46.13.9 1915  
48.77.167.170 2292  
91.99.72.15 2398  
23.101.169.3 2863  
66.249.66.91 6202  
66.249.66.194 9331  
[END]
```

Obrázok 7: Vizualizácia výsledného súboru spracovaného MapReduce. Môžeme vidieť, že výsledok výskytu/aktivity najpočetnejšej IP adresy sa nám zhoduje aj v ostatných zadaniach, zdroj: [vlastné spracovanie]

## Záver

Cieľom tohto zadania bolo opísať tvorbu `mapper.py` a `reducer.py` skriptov a logiku ich vykonávania. Následne ich spustiť na Hadoop clusteri a vizualizovať výsledok. Cieľ tohto zadania sme splnili.

Problémy sa nám vyskytli najmä pri tvorbe skriptov, kde nám trvalo dlhší čas pochopiť, ako a čo nastaviť, aby sme docielili výsledok, aký chceme. Aj v porovnaní s ostatnými zadaniami môžeme vidieť, že výsledok nám vyšiel rovnaký ako napríklad keď sme rovnaký súbor spúšťali v Hortonworkse alebo pod Hive v Linuxe. Z toho usudzujeme, že skripty sme napísali správne a výsledok tomu zodpovedá.

Veríme, že toto zadanie prinesie čitateľovi nový pohľad na možnosť spracovávať veľké súbory a získa nové prípadne obohatí už doterajšie vedomosti a schopnosti o tvorbu takýchto skriptov.

## Zoznam použitej literatúry

- [1] APACHE SOFTWARE FOUNDATION. 2018. *MapReduce Tutorial* In `hadoop.apache.org` [online]. 2018. [citované dňa 31.03.2023]. Dostupné na internete: <https://hadoop.apache.org/docs/r3.1.1/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>.