# 1.0 Importovanie potrebných knižníc

FindSpark je knižnica v jazyku Python, ktorá uľahčuje prácu s Apache Sparkom v Pythone. Hlavným cieľom FindSparku je zabezpečiť, aby Apache Spark bol k dispozícii v prostredí Python a aby jeho používanie bolo čo najjednoduchšie. FindSpark teda poskytuje rozhranie na pripojenie a konfiguráciu Apache Sparku v Pythone, vrátane nastavenia cesty k jeho inštalácii a konfigurácie systémových premenných. Tým umožňuje Python programátorom využívať všetky výhody a funkcie Apache Sparku bez nutnosti zvládnuť zložitú konfiguráciu a pripojenie.

In [1]:
```
# Inicializácia FindSpark
import findspark
findspark.init()
```

V tejto časti sú načítané knižnice PySpark a vytvorí sa relačné prostredie pre prácu s veľkými dátovými sadami pomocou Sparku. Konkrétne, kód importuje PySpark knižnicu a z nej moduly SparkContext, SQLContext a SparkSession. Tieto moduly slúžia na inicializáciu a konfiguráciu relačného prostredia.

- SparkContext vytvorí hlavný kontext Sparku a umožní interakciu s Hadoop Distributed File System (HDFS) alebo inými úložiskami dát.
- SQLContext poskytuje rozhranie na prácu s dátami v tabuľkách a DataFrame objektoch.
- SparkSession poskytuje užívateľské rozhranie pre interakciu s dátami v Sparku, vrátane podpory pre dávkové a interaktívne spracovanie dát.

Kód tiež vytvára inštancie týchto objektov a priradí ich premenným sc, sqlContext a spark, čím sa inicializuje relačné prostredie pre prácu s dátami v Sparku.

In [2]:
```
# Načítaj Spark a spusti jeho premenné relačného prostredia
import pyspark
from pyspark.context import SparkContext
from pyspark.sql.context import SQLContext
from pyspark.sql.session import SparkSession

sc = SparkContext()
sqlContext = SQLContext(sc)
spark = SparkSession(sc)
```

```
C:\spark\spark-3.3.2-bin-hadoop3\python\pyspark\sql\context.py:112: FutureWarning:
Deprecated in 3.0.0. Use SparkSession.builder.getOrCreate() instead.
  warnings.warn(
```

táto časť načíta dve závislosti, re a pandas, ktoré sa často používajú v jazyku Python na prácu s regulárnymi výrazmi a manipuláciu s dátami.

In [3]:
```
# Load up other dependencies
import re
import pandas as pd
```

# 2.0 Načítanie Datasetu

V tejto časti sú získané zoznamu názvov súborov s konkrétnou príponou (gz) v aktuálnom adresári.

In [4]:
```
import glob

raw_data_files = glob.glob('*.gz')
raw_data_files
```

Out[4]:
```
['NASA_access_log_Jul95.gz']
```

Táto časť kódu používa Apache Spark pre načítanie dát z niekoľkých súborov do DataFrame. Nakoniec, výraz (base_df.count(), len(base_df.columns)) vypisuje počet riadkov a počet stĺpcov v DataFrame.

In [5]:
```
base_df = spark.read.text(raw_data_files)
base_df.printSchema()

print((base_df.count(), len(base_df.columns)))
```

```
root
 |-- value: string (nullable = true)

(1891715, 1)
```

In [6]:
```
type(base_df)
```

Out[6]:
```
pyspark.sql.dataframe.DataFrame
```

# 3.0 Ukážka Datasetu

Táto časť kódu konvertuje dátový rámec base_df do distribuovaného dátového rámca (RDD) a potom vypisuje jeho typ pomocou funkcie type().

RDD (Resilient Distributed Dataset) je hlavnou abstrakciou v Apache Sparku, ktorá umožňuje distribuované spracovanie dát. RDD je dátová štruktúra, ktorá obsahuje nezmeniteľné objekty a je distribuovaná cez viacero uzlov v klastri. Táto dátová štruktúra poskytuje mnoho výhod pri spracovaní veľkých dátových súborov, pretože umožňuje rozdeľovať spracovanie dát do rôznych uzlov, ktoré sú spracovávané súbežne.

Pretože base_df_rdd je distribuovaný dátový rámec (RDD), môže byť použitý na paralelné spracovanie dát pomocou Apache Sparku. Použitie RDD miesto klasického dátového rámca umožňuje lepšie využitie paralelizmu a efektívne spracovanie veľkých objemov dát.

In [7]:
```
base_df_rdd = base_df.rdd
type(base_df_rdd)
```

Out[7]:
```
pyspark.rdd.RDD
```

In [8]:
```
base_df.show(10, truncate=False)
```

```
+------------------------------------------------------------------------------
|value
+------------------------------------------------------------------------------
|199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/ HTTP/1.0" 200 
|unicomp6.unicomp.net - - [01/Jul/1995:00:00:06 -0400] "GET /shuttle/countdown/ HTT
|199.120.110.21 - - [01/Jul/1995:00:00:09 -0400] "GET /shuttle/missions/sts-73/miss
|burger.letters.com - - [01/Jul/1995:00:00:11 -0400] "GET /shuttle/countdown/liftof
|199.120.110.21 - - [01/Jul/1995:00:00:11 -0400] "GET /shuttle/missions/sts-73/sts-
|burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET /images/NASA-logosmall.gi
|burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET /shuttle/countdown/video/
|205.212.115.106 - - [01/Jul/1995:00:00:12 -0400] "GET /shuttle/countdown/countdown
|d104.aa.net - - [01/Jul/1995:00:00:13 -0400] "GET /shuttle/countdown/ HTTP/1.0" 20
|129.94.144.152 - - [01/Jul/1995:00:00:13 -0400] "GET / HTTP/1.0" 200 7074
+------------------------------------------------------------------------------
only showing top 10 rows
```

# 4.0 Vytvorenie štrukturovaných údajov pomocou regulérnych výrazov

Táto časť kódu zobrazuje prvých 10 riadkov dátového rámca (DataFrame) "base_df". Parameter "truncate=False" znamená, že ak by hodnota v niektorom stĺpci prekročila maximálnu šírku stĺpca, hodnota by nebola zkrátená a zobrazila by sa celá.

V tejto časti kódu sa definujú regulárne výrazy, ktoré budú použité na vyhľadávanie konkrétnych vzorov v textovom reťazci. Regulárne výrazy sú zápisom, ktorý opisuje formát alebo vzor, ktorý má hľadaný text spĺňať.

Konkrétne sa tu vytvárajú niekoľko regulárnych výrazov:

- host_pattern - tento regulárny výraz slúži na hľadanie názvu hostiteľa v textovom reťazci. Výraz začína s počiatkom riadku (^), potom obsahuje neprázdny znak (\S+) a k nemu môže byť pridaný ľubovoľný počet ďalších neprázdnych znakov nasledovaných bodkou ([\S+.]+), pričom výraz musí končiť neprázdny znak (\S+). Celý tento výraz sa nachádza v zátvorkách a bude slúžiť na identifikáciu názvu hostiteľa v texte.

- ts_pattern - tento regulárny výraz slúži na hľadanie časovej značky v textovom reťazci. Výraz začína s lomenými zátvorkami, nasleduje dátum vo formáte DD/MMM /YYYY:HH:MM:SS a nakoniec sa výraz končí pomlčkou a štvorciferným číslom (predstavujúcim posun o časové pásmo vzhľadom k UTC). Tento výraz sa nachádza v zátvorkách a bude slúžiť na identifikáciu časovej značky v texte.

- method_uri_protocol_pattern - tento regulárny výraz slúži na hľadanie informácií o HTTP metóde, URI a protokole v textovom reťazci. Výraz začína a končí úvodzovkami a obsahuje tri bloky - prvý blok (\S+) obsahuje HTTP metódu, druhý blok (\S+) obsahuje URI a tretí blok (\S*) obsahuje protokol (ak je uvedený). Tento výraz sa nachádza v zátvorkách a bude slúžiť na identifikáciu týchto informácií v texte.

- status_pattern - tento regulárny výraz slúži na hľadanie HTTP statusového kódu v textovom reťazci. Výraz obsahuje medzeru, nasleduje trojmiestne číslo a nakoniec sa výraz končí medzerou. Tento výraz sa nachádza v zátvorkách a bude slúžiť na identifikáciu HTTP statusového kódu v texte.

- content_size_pattern - tento regulárny výraz slúži na hľadanie veľkosti obsahu v textovom reťazci. Výraz obsahuje medzeru a potom ľubovoľný počet čísel. Tento výraz sa nachádza v zátvorkách a bude slúžiť na identifikáciu veľkosti obsahu v texte.

In [9]:
```
# Vytváranie množín regulárnych výrazov pre vyhľadávanie
host_pattern = r'(^\S+\.[\S+\.]+\S+)\s'
ts_pattern = r'\[(\d{2}/\w{3}/\d{4}:\d{2}:\d{2}:\d{2} -\d{4})]'
method_uri_protocol_pattern = r'\"(\S+)\s(\S+)\s*(\S*)\"'
status_pattern = r'\s(\d{3})\s'
content_size_pattern = r'\s(\d+)$'
```

Táto časť načíta a analyzuje logovacie súbory a vytvára DataFrame. Používa funkciu regexp_extract z knižnice pyspark.sql.functions na extrakciu informácií z logovacieho súboru pomocou regexov. Každá extrahovaná informácia je priradená k stĺpcu vytvorenému v DataFrame. Následne DataFrame zobrazí prvých 10 riadkov a počet riadkov a stĺpcov v DataFrame.

In [10]:
```
from pyspark.sql.functions import regexp_extract

logs_df = base_df.select(regexp_extract('value', host_pattern, 1).alias('host'),
                         regexp_extract('value', ts_pattern, 1).alias('timestamp'),
                         regexp_extract('value', method_uri_protocol_pattern, 1).al
                         regexp_extract('value', method_uri_protocol_pattern, 2).al
                         regexp_extract('value', method_uri_protocol_pattern, 3).al
                         regexp_extract('value', status_pattern, 1).cast('integer')
                         regexp_extract('value', content_size_pattern, 1).cast('int
logs_df.show(10, truncate=True)
print((logs_df.count(), len(logs_df.columns)))
```

```
+--------------------+--------------------+------+--------------------+--------+---
|                host|           timestamp|method|            endpoint|protocol|sta
+--------------------+--------------------+------+--------------------+--------+---
|        199.72.81.55|01/Jul/1995:00:00...|   GET|    /history/apollo/|HTTP/1.0|
|unicomp6.unicomp.net|01/Jul/1995:00:00...|   GET| /shuttle/countdown/|HTTP/1.0|
|      199.120.110.21|01/Jul/1995:00:00...|   GET|/shuttle/missions...|HTTP/1.0|
|  burger.letters.com|01/Jul/1995:00:00...|   GET|/shuttle/countdow...|HTTP/1.0|
|      199.120.110.21|01/Jul/1995:00:00...|   GET|/shuttle/missions...|HTTP/1.0|
|  burger.letters.com|01/Jul/1995:00:00...|   GET|/images/NASA-logo...|HTTP/1.0|
|  burger.letters.com|01/Jul/1995:00:00...|   GET|/shuttle/countdow...|HTTP/1.0|
|     205.212.115.106|01/Jul/1995:00:00...|   GET|/shuttle/countdow...|HTTP/1.0|
|         d104.aa.net|01/Jul/1995:00:00...|   GET| /shuttle/countdown/|HTTP/1.0|
|      129.94.144.152|01/Jul/1995:00:00...|   GET|                   /|HTTP/1.0|
+--------------------+--------------------+------+--------------------+--------+---
only showing top 10 rows

(1891715, 7)
```

# Úprava zlých hodnôt

Najprv overme, či sa v pôvodnom dátovom rámci nenachádzajú žiadne nulové riadky.

In [11]:
```
(base_df
    .filter(base_df['value']
                .isNull())
    .count())
```

Out[11]: 0

V našom logovom súbore bolo NULA nulových riadkov, čo znamená, že v každom riadku máme aspoň niečo zachytené. V našom dátovom rámci však stále môžeme mať niektoré nulové stĺpce, pretože všetky roly nemusia mať všetky hodnoty stĺpcov.

In [12]:
```
logs_df.filter(logs_df['endpoint'].isNull()).count()
```

Out[12]: 0

In [13]:
```
bad_rows_df = logs_df.filter(logs_df['host'].isNull()|
                             logs_df['timestamp'].isNull() |
                             logs_df['method'].isNull() |
                             logs_df['endpoint'].isNull() |
                             logs_df['status'].isNull() |
                             logs_df['content_size'].isNull()|
                             logs_df['protocol'].isNull())
bad_rows_df.count()
```

Out[13]: 19727

Tu skontrolujeme celkový počet chýbajúcich hodnôt v každom stĺpci.

In [14]:
```
from pyspark.sql.functions import col
from pyspark.sql.functions import sum as spark_sum
from pyspark.sql.functions import count, isnan, lit, when

logs_df.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in logs_d
```

```
+----+---------+------+--------+--------+------+------------+
|host|timestamp|method|endpoint|protocol|status|content_size|
+----+---------+------+--------+--------+------+------------+
|   0|        0|     0|       0|       0|     1|       19727|
+----+---------+------+--------+--------+------+------------+
```

Najčastejšie vyskytujúcich sa 15 hostov

In [15]:
```
from pyspark.sql.functions import desc

host_mostly_count= logs_df.groupBy("host").count().orderBy(desc("count")).limit(15)
```

```
+--------------------+-----+
|                host|count|
+--------------------+-----+
|piweba3y.prodigy.com|17572|
|piweba4y.prodigy.com|11591|
|piweba1y.prodigy.com| 9868|
|  alyssa.prodigy.com| 7852|
| siltb10.orl.mmc.com| 7573|
|piweba2y.prodigy.com| 5922|
|  edams.ksc.nasa.gov| 5434|
|        163.206.89.4| 4906|
|         news.ti.com| 4863|
|                    | 4797|
|disarray.demon.co.uk| 4353|
|www-a2.proxy.aol.com| 4179|
|www-d1.proxy.aol.com| 4158|
|  vagrant.vf.mmc.com| 4146|
|       198.133.29.18| 4136|
+--------------------+-----+
```

počty jednotlivých metód

In [16]:
```
logs_df.groupBy("method").agg(count("*").alias("count")).show()
```

```
+------+-------+
|method|  count|
+------+-------+
|  POST|    111|
|  HEAD|   3950|
|   GET|1886791|
|      |    863|
+------+-------+
```

počty jednotlivých status kódov

In [17]:
```
logs_df.groupBy("status").agg(count("*").alias("count")).show()
```

```
+------+-------+
|status|  count|
+------+-------+
|   501|     14|
|  null|      1|
|   500|     62|
|   400|      5|
|   403|     54|
|   404|  10843|
|   200|1701536|
|   304| 132627|
|   302|  46573|
+------+-------+
```

zistí z requestov od ktorých hostov vznikaju serverové alebo klientské chyby

In [18]:
```
from pyspark.sql.functions import count, when

logs_df.where(logs_df["status"].between(400, 599)) \
  .groupBy("host") \
  .agg(count(when(logs_df["status"].between(400, 599), True)).alias("count")) \
  .orderBy("count", ascending=False) \
  .show(15)
```

```
+--------------------+-----+
|                host|count|
+--------------------+-----+
|hoohoo.ncsa.uiuc.edu|  251|
|jbiagioni.npt.nuw...|  131|
|piweba3y.prodigy.com|  110|
|                    |   92|
|piweba1y.prodigy.com|   92|
|        163.205.1.45|   70|
|phaelon.ksc.nasa.gov|   64|
|www-d4.proxy.aol.com|   61|
|monarch.eng.buffa...|   56|
|piweba4y.prodigy.com|   56|
|  alyssa.prodigy.com|   54|
|www-a2.proxy.aol.com|   52|
|www-b4.proxy.aol.com|   48|
|www-b6.proxy.aol.com|   44|
|www-b3.proxy.aol.com|   43|
+--------------------+-----+
only showing top 15 rows
```

zistí na ktorých endpointoch dochádzalo najčastejšie k chybám

In [19]:
```
from pyspark.sql.functions import count, when

logs_df.where(logs_df["status"].between(400, 599)) \
  .groupBy("endpoint") \
  .agg(count(when(logs_df["status"].between(400, 599), True)).alias("count")) \
  .orderBy("count", ascending=False) \
  .show(15)
```

```
+--------------------+-----+
|            endpoint|count|
+--------------------+-----+
|/pub/winvn/readme...|  667|
|/pub/winvn/releas...|  547|
|/history/apollo/a...|  286|
|/shuttle/resource...|  232|
|/history/apollo/a...|  230|
|/://spacelink.msf...|  215|
|/history/apollo/p...|  215|
|/images/crawlerwa...|  214|
|/history/apollo/s...|  183|
|/shuttle/resource...|  180|
|/shuttle/missions...|  175|
|/shuttle/missions...|  168|
|/elv/DELTA/uncons...|  163|
|/history/apollo/p...|  140|
|/shuttle/missions...|  107|
+--------------------+-----+
only showing top 15 rows
```