Bevezetés a Spark használatába felhő környezetben

Emődi Márk

emodi.mark@sztaki.hu

Prezentáció célja

- Gyakorlat közeli ismeretek bővítése
- Gyakorlati példák
  - RDD alapvető funkciói
  - DF alapvető funkciók
  - Fájl beolvasás
  - Adatfolyam előkészületek és használata

RDD

A Spark a rugalmasan elosztott adathalmaz (RDD) absztrakcióján alapul, ahol:

- Az RDD adatok gyűjteményét reprezentálja
- Minden elem azonos szerkezetű (keret, rekord, sor)
- Csak olvasható (megváltoztathatatlan)
- Az elemeket a Spark elosztva tárolja
- Az együtt elhelyezett elemek a partícióból (szeletet vagy leosztást) alkotnak
- Minden partíció a kívánt helyen kerül feldolgozásra
- A Spark új RDD-ket épít a partíciók egyidejű párhuzamos felépítésével
- Nagy adat esetén egyetlen állomáson sem érhető el a teljes RDD

RDD műveletek

- Transzformáció: Spark-ban transzformációnak nevezzük azon műveleteket, amelyek egy új adathalmazt (RDD-t) eredményeznek (lévén, hogy az RDD csak olvasható (immutable))
  - Jellemző a transzformációkra, hogy utólagosan kerülnek kiértékelésre (lazy evaluation), azaz egészen addig nem, amíg konkrét végrehajtó parancsig (action) nem ér el a program:
  - Példa: beolvasunk szavakat egy szöveges fájlból, áttranszformáljuk azokat nagybetűssé, majd a tartalmát kiíratjuk. Egészen addig nem kerül beolvasásra a fájl, és a transzformáció sem megy végbe, amíg a kiíratás parancsig nem ér el a program.
- Akció: konkrét parancsok, melyek elindítják az előzetes transzformációk elvégzését, kezdeményezik az adatokat kiértékelését/aggregálást, majd visszaadják a végeredményt. Példa: collect(), take(), count()

RDD műveletek

- Transzformáció:

  map(): Visszaad egy új adathalmazt. Az adathalmaz a forrás halmaz minden elemét kiértékeli az adott metóduson

  filter(): Visszaad egy új adathalmazt. Az adathalmaz a forrás halmazon értelmezett logikai kiértékelés

  union(RDD): Visszaad egy új adathalmazt, ami az uniója a két adathalmaznak

- Akció:

  count(): Visszaadja az elemek számát az RDD-ben

  collect(): Visszaadja az összes elemet az RDD-ben

  first(): Lekéri az első elemet az adathalmazon (take(1) hasonló funkcionalitással bír)

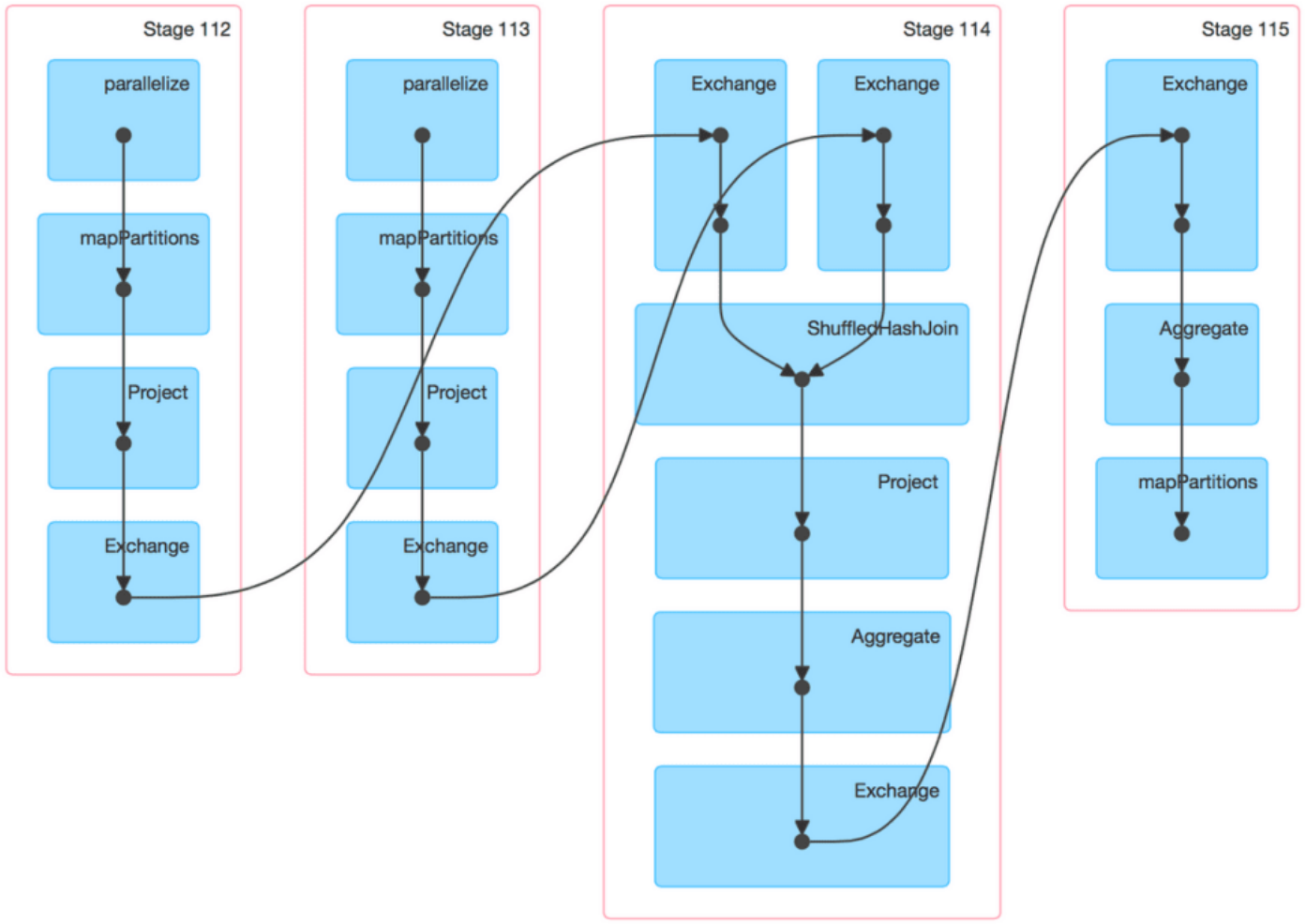
  take(n): Visszaadja az adathalmaz n elemét

  saveAsTextFile(útvonal): Leírja az adathalmazt a kívánt útvonalra

  foreach(): Lefuttatja a metódust az adathalmaz minden elemén

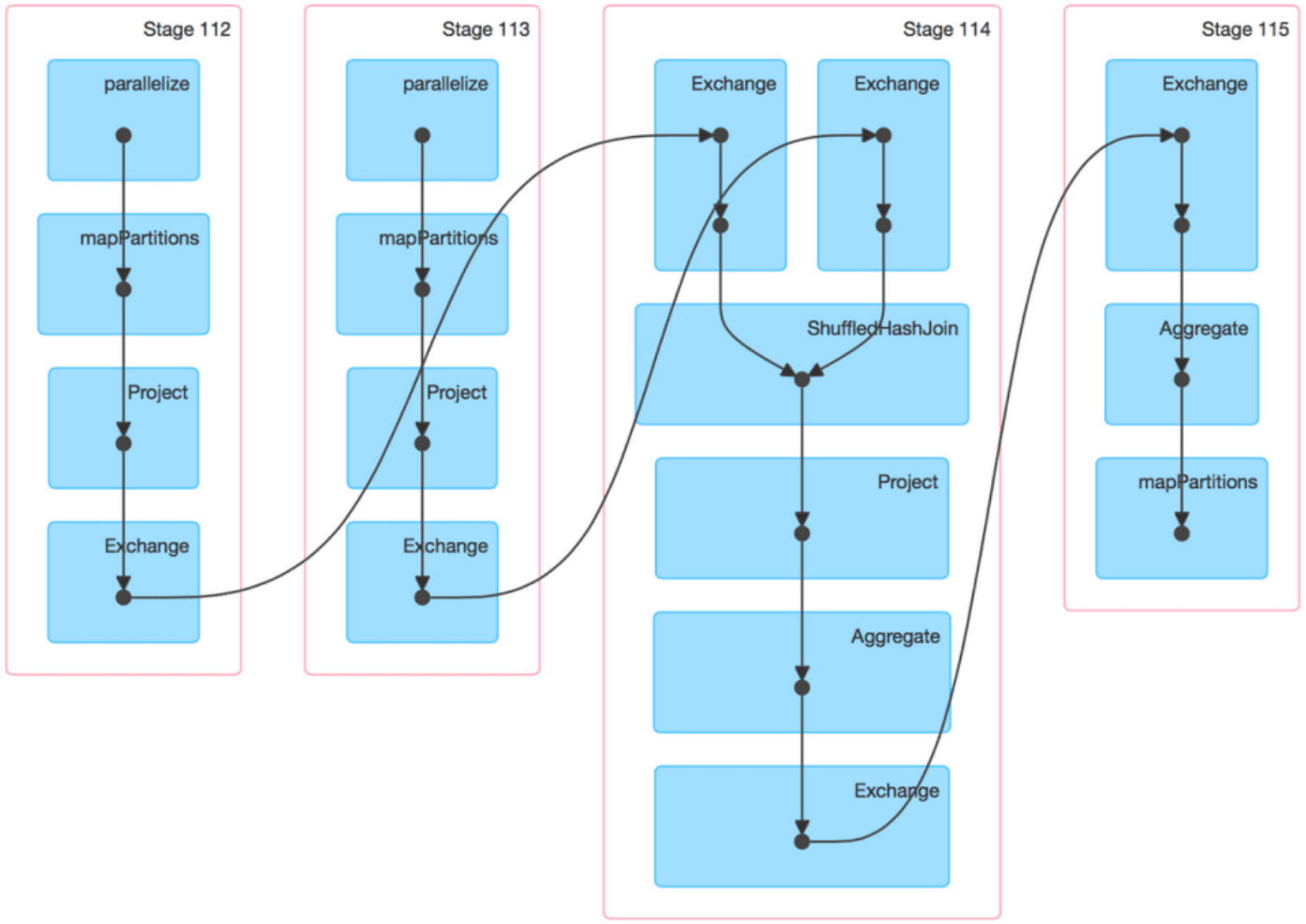
Spark kiértékelés

- A Spark lusta kiértékelést használ:
  - nem történik adatfeldolgozás, amíg a program végrehajtása el nem éri az akció (action) műveletet azaz az akciók váltják ki az RDD létrehozást
- Mivel egy RDD származhat (egy vagy több) RDD-ből, a Sparknak képesnek kell lennie a hierarchia (származás) megállapítására.



Spark kiértékelés

- A Spark egy irányított körmentes gráf (DAG) felépítésével valósítja meg az RDD létrehozást/átalakítást/műveleti parancsokat, ahol:
  - Minden csomópont egy RDD-nek feleltethető meg
  - Minden él egy transzformációnak feleltethető meg (függőség)
  - Párhuzamosítás és optimalizálás során nyújt hatalmas előnyt, mely az ütemező munkáját segíti elő



Spark gyorsítótár

- Az RDD partíciókat az ütemező a lehető legtovább tárolja a memóriában
  - Amikor egy RDD partíciót újra kell olvasni (egy másik művelettel/transzformációval), akkor nincs szükség I/O műveletre, ha megtalálható a gyorsítótárban
- Az adatokat lemezre írja, ha a gyorsítótár tele van
- A gyorsítótár irányelvei konfigurálhatóak
  - Pl. memóriában/lemezen tartás, ütemező algoritmus (régebbi adat törlésének ütemezése), méret, tömörítés, stb.

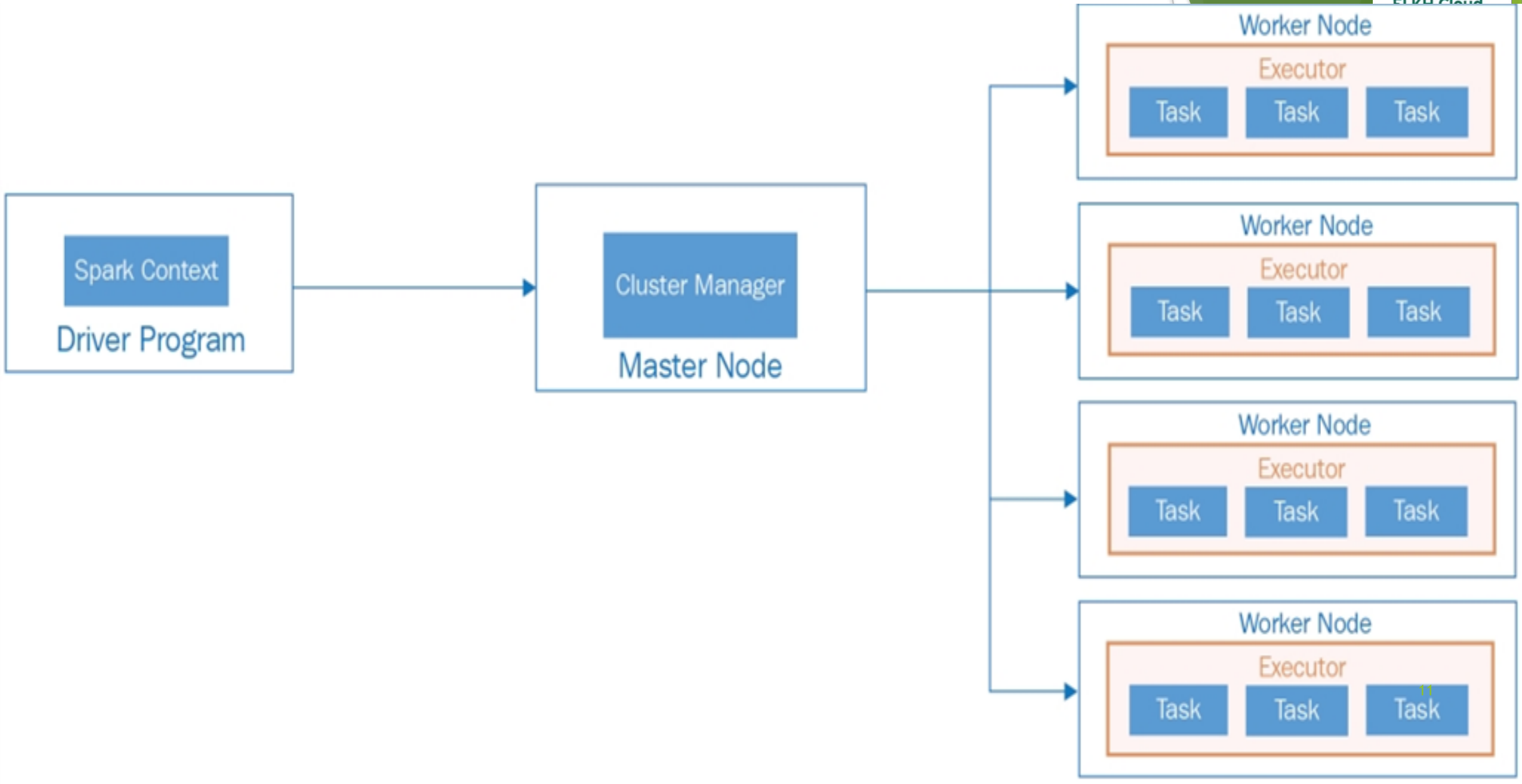
Tárolási opciók megadás

- ***MEMORY_ONLY***: minden objektum a Java Virtual Machine memóriájában tárolódik. Amennyiben a teljes RDD nem fér el a memóriában, egyes partíciók csak akkor kerülnek beolvasásra, amikor hely és szükség van rájuk
- ***MEMORY_AND_DISK***: amíg elfér a teljes RDD a memóriában addig ott tárol, különben a többi partíciót már a lemezen (alapértelmezett beállítás)
- ***DISK_ONLY***: az RDD-t teljes egészében a lemezen tárolja
- ***MEMORY_AND_SER***: ugyanaz mint a MEMORY_ONLY, csak byte array-ként tárol (szerializálva, és nem deszerializálva), ezáltal helytakarékosabb, de CPU igényesebb dolgozni vele
- ***MEMORY_AND_DISK_SER***: ugyanaz mint a MEMORY_AND_DISK, csak szintén byte array-ként tárol
- ***MEMORY_ONLY_2***, ***DISK_ONLY_2***: ugyanúgy csak memóriában/lemezen, annyi különbséggel hogy replika is készül az RDD-ből egy másik node-on

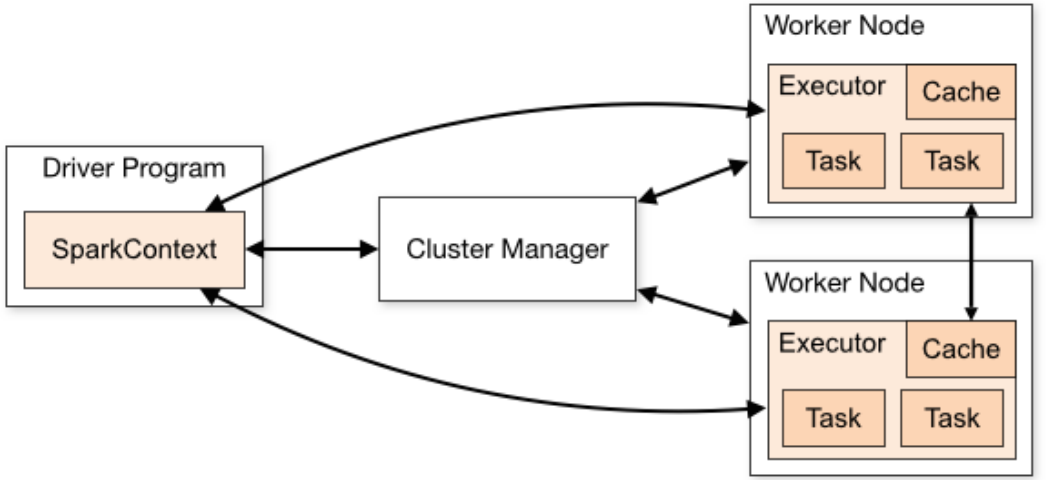
Rugalmas adattárolás

- Abban az esetben, ha valamelyik dolgozó állomás hibára fut, csak a nélkülözhetetlen RDD-ket szükséges újraépíteni futási időben a hierarchia információk alapján

Architektúra



Architektúra



- Alkalmazás: A programkódunk kliens oldalon fut
  - Létrehozza a SparkContext-et, RDD-ket, meghatároz transzformációkat / műveleteket, elmenti az eredményeket (HDFS-be), nincs végrehajtási logika
- Driver: az alkalmazásunkban indul, amikor a SparkContext létrejött, kliens oldalon fut
  - Csatlakozik a Spark Masterhez és az ütemező végrehajtókat foglal le
  - Futási gráfot épít fel az RDD-khez
  - Ütemezi a „fizikai” végrehajtási tervet
  - Kezeli az adat és kód megosztásokat az alkalmazás és a végrehajtók között

Architektúra

- Master: erőforrás kezelő, aki ismeri a dolgozó állomásokat és azok hardverspecifikus konfigurációját (CPU / RAM)
- Worker (worker/slave): egy démon fut minden dolgozó állomáson. Kezeli a rajta futó aktív feladatokat
- Végrehajtó (Executor): egy egyéni folyamat (JVM), ami a dolgozó állomásokon indul el az alkalmazás számára. Az alkalmazástól (context/driver) várja a végrehajtani kívánt feladatokat. A végrehajtási helyeket ő tartja számon, ettől függ, hogy hány párhuzamos feladat indítható el a dolgozó állomáson

Big Data - Adat

- Big Data feldolgozás során az adatok sokasága kritikus
- Elérhető adatok:
  - Különböző API-k használata:

    Közösségi API-k (Twitter, Twitch, Facebook, …) - Regisztrációhoz és engedélyhez kötött
  - Ingyenesen elérhető adathalmazok

    https://github.com/awesomedata/awesome-public-datasets

    https://github.com/datasets

Awesome Public Datasets

Table of Contents

Neuroscience

- Allen Institute Datasets
- Brain Catalogue
- Brainomics
- CodeNeuro Datasets [fixme]
- Collaborative Research in Computational Neuroscie
- FCP-INDI
- Human Connectome Project
- NDAR
- NIMH Data Archive
- NeuroData
- NeuroMorpho - NeuroMorpho.Org is a centrally cu
- Neuroelectro
- OASIS
- OpenNEURO
- OpenfMRI
- Study Forrest

Physics

- CERN Open Data Portal
- Crystallography Open Database
- IceCube - South Pole Neutrino Observatory
- Ligo Open Science Center (LOSC) - Gravitational wa
- NASA Exoplanet Archive
- NSSDC (NASA) data of 550 space spacecraft
- Sloan Digital Sky Survey (SDSS) - Mapping the Univ

Awesome Public Datasets

Biology

- 1000 Genomes - The 1000 Genomes Project ran between 2008 and 2015, [...] [fixme]
- American Gut (Microbiome Project) - The American Gut project is the [...]
- Broad Bioimage Benchmark Collection (BBBC) - The Broad Bioimage Benchmark [...]
- Broad Cancer Cell Line Encyclopedia (CCLE)
- Cell Image Library - This library is a public and easily accessible [...]
- Complete Genomics Public Data - A diverse data set of whole human genomes [...]
- EBI ArrayExpress - ArrayExpress Archive of Functional Genomics Data [...]
- EBI Protein Data Bank in Europe - The Electron Microscopy Data Bank [...]
- ENCODE project - The Encyclopedia of DNA Elements (ENCODE) Consortium is [...]
- Electron Microscopy Pilot Image Archive (EMPIAR) - EMPIAR, the Electron [...]
- Ensembl Genomes [fixme]
- Gene Expression Omnibus (GEO) - GEO is a public functional genomics data [...]
- Gene Ontology (GO) - GO annotation files
- Global Biotic Interactions (GloBI)
- Harvard Medical School (HMS) LINCS Project - The Harvard Medical School [...]
- Human Genome Diversity Project - A group of scientists at Stanford [...]
- Human Microbiome Project (HMP) - The HMP sequenced over 2000 reference [...]
- ICOS PSP Benchmark - The ICOS PSP benchmarks repository contains an [...]
- International HapMap Project
- Journal of Cell Biology DataViewer [fixme]
- KEGG - KEGG is a database resource for understanding high-level functions [...]
- MIT Cancer Genomics Data
- NCBI Proteins
- NCBI Taxonomy - The NCBI Taxonomy database is a curated set of names and [...]
- NCI Genomic Data Commons - The GDC Data Portal is a robust data-driven [...]
- NIH Microarray data
- OpenSNP genotypes data - openSNP allows customers of direct-to-customer [...]
- Palmer Penguins - The goal of palmerpenguins is to provide a great [...]
- Pathguid - Protein-Protein Interactions Catalog

Climate+Weather

- Actuaries Climate Index
- Australian Weather
- Aviation Weather Center - Consistent, timely and accurate w
- Brazilian Weather - Historical data (In Portuguese) - Data rel
- Canadian Meteorological Centre
- Climate Data from UEA (updated monthly)
- Dutch Weather - The KNMI Data Center (KDC) portal provid
- European Climate Assessment & Dataset
- Global Climate Data Since 1929
- Charting The Global Climate Change News Narrative 2009-2
- NASA Global Imagery Browse Services
- NOAA Bering Sea Climate
- NOAA Climate Datasets
- NOAA Realtime Weather Models
- NOAA SURFRAD Meteorology and Radiation Datasets
- The World Bank Open Data Resources for Climate Change

Finance

- BIS Statistics - BIS statistics, compiled
- Blockmodo Coin Registry - A registry
- CBOE Futures Exchange
- Complete FAANG Stock data - This da
- Google Finance
- Google Trends
- NASDAQ [fixme]
- NYSE Market Data
- OANDA
- OSU Financial data [fixme]
- Quandl
- St Louis Federal
- Yahoo Finance

Awesome Public Datasets

ComputerNetworks

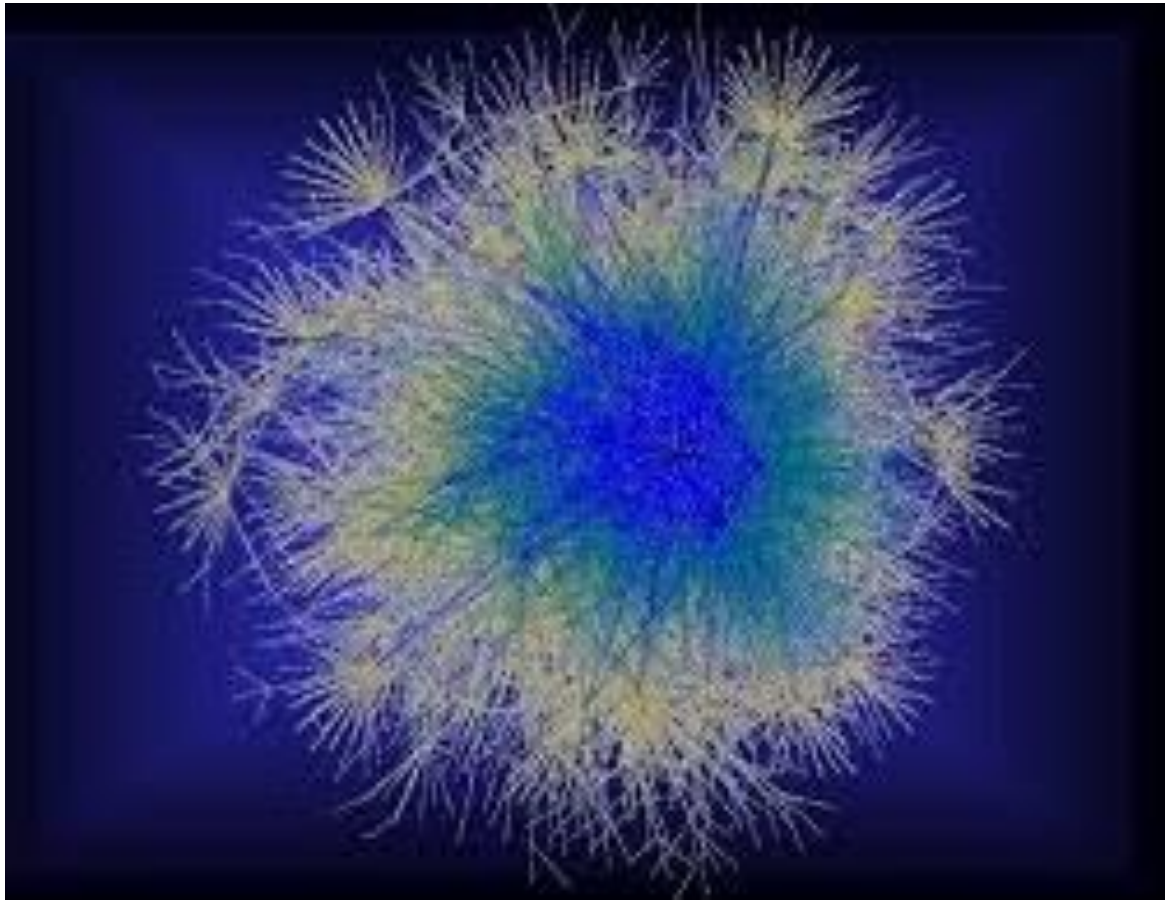
- 3.5B Web Pages from CommonCrawl 2012
- 53.5B Web clicks of 100K users in Indiana Univ.
- CAIDA Internet Datasets
- CRAWDAD Wireless datasets from Dartmouth Univ. [fixme]
- ClueWeb09 - 1B web pages
- ClueWeb12 - 733M web pages
- CommonCrawl Web Data over 7 years
- Criteo click-through data
- Internet-Wide Scan Data Repository [fixme]
- MIRAGE-2019 - MIRAGE-2019 is a human-generated dataset for mobile traffic [...] [fixme]
- OONI: Open Observatory of Network Interference - Internet censorship data
- Open Mobile Data by MobiPerf
- The Peer-to-Peer Trace Archive - Real-world measurements play a key role [...]
- Rapid7 Sonar Internet Scans
- UCSD Network Telescope, IPv4 /8 net

CyberSecurity

- CCCS-CIC-AndMal-2020 - The dataset includes 200K benign and 200K malware [...]
- Traffic and Log Data Captured During a Cyber Defense Exercise - This [...]

Graph	#Nodes	#Arcs
Page Graph	3,563 million	128,736 million
Subdomain Graph	101 million	2,043 million
1st Level Subdomain Graph	95 million	1,937 million
PLD Graph	43 million	623 million

Data Set	Index File	Arc File
Page Graph	see below (45 GB)	see below (331 GB)
Subdomain Graph	download (832 MB)	download (9.2 GB)
1st Subdomain Graph	download (757 MB)	download (8.7 GB)
PLD Graph	download (297 MB)	download (2.8 GB)



Datasets

Data Packaged Core Datasets

Important, commonly-used datasets in high quality, easy-to-use & open form as data pac

The Internet http://datahub.io/docs/core-data

Repositories 137 Packages People 8 Projects

Pinned repositories

awesome-data

Awesome datasets curated. By topic and for core data https://datahub.io/docs/core-data

440 75

Find a repository... Type: All Language: All

covid-19

Novel Coronavirus 2019 time series data on cases

dataset data-package datapackage coronavirus covid-19 covid covid19-data

Python 574 1,048 30 0 Updated 3 hours ago

emojis

Unicode Emoji as UTS #51 specification

Python 3 6 0 0 Updated 8 hours ago

s-and-p-500-companies

List of companies in the S&P 500 together with associated financials

Python 284 293 4 0 Updated 8 hours ago

s-and-p-500-companies-financials

List of companies in the S&P 500 (Standard and Poor's 500).

HTML 63 34 2 1 Updated 8 days ago

dac-and-crs-code-lists

Machine readable DAC CRS codelists

Python 6 6 2 0 Updated 16 days ago

natural-gas

Natural Gas Prices including Henry Hub

Python 12 10 1 0 Updated 21 days ago

un-locode

United Nations Codes for Trade and Transport Locations (UN/LOCODE) and Country Codes

Shell 45 84 0 0 Updated on Dec 21 2020

country-codes

Comprehensive country code information, including ISO 3166 codes, ITU dialing codes, ISO 4217 currency codes, and many others

Python 395 636 14 4 Updated on Nov 29 2020

currency-codes

ISO 4217 List of Currencies and Currency Codes

Shell 79 92 3 1 Updated on Oct 12 2020

glacier-mass-balance

Average cumulative mass balance of "reference" Glaciers worldwide

Python Unlicense 9 6 0 1 Updated on Sep 26 2020

Datasets

unece-units-of-measure
Standardised codes from Recommendation 20, mantained by UNECE.
6 4 1 1 Updated on Apr 9 2020

world-religion-projections
Word Religion Projections (2010-2050)
5 8 1 0 Updated on Mar 28 2020

airport-codes
List of Airport codes, locations and other information around the world
Python 74 225 15 0 Updated on Mar 16 2020

ISO-Container-Codes
Coded list of ISO 6346 shipping containers, used in international trade and electronic shipping messages.
14 17 0 0 Updated on Feb 25 2020

harmonized-system
HS Code as a datapackage
10 18 0 0 Updated on Feb 17 2020

core-datasets
DataHub.io awesome datasets - curated collections of high quality dataset organized by topic
JavaScript 14 49 0 (2 issues need help) 0 Updated on Dec 15 2019

geo-countries
Country polygons as GeoJSON in a datapackage
Dockerfile 82 313 6 0 Updated on Aug 2 2019

owid-datasets
Forked from owid/owid-datasets
OWID Dataset Collection
100 3 0 0 Updated on Jul 8 2019

oil-prices
Brent crude and WTI oil prices from US EIA
Python Unlicense 35 43 0 0 Updated on Aug 31 2020

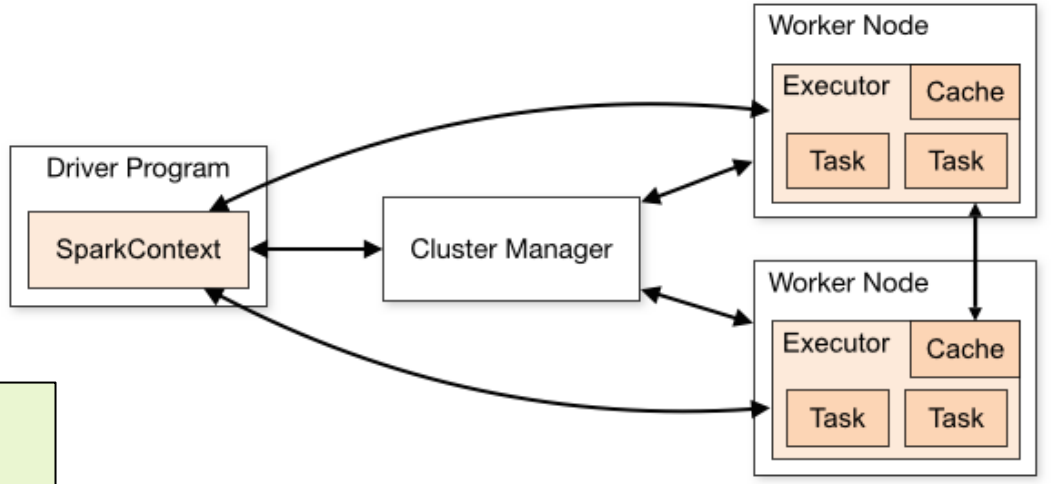
awesome-data
Awesome datasets curated. By topic and for core data
https://datahub.io/docs/core-data
75 440 209 0 Updated on Jun 20 2020

fips-10-4
List of FIPS (Federal Information Processing Standards) region codes
Python 6 6 1 1 Updated on May 10 2020

population-reference-bureau
Collect datasets from Population Reference Bureau about demographic and health
Python 2 1 0 0 Updated on May 4 2020

population
Population figures for countries, regions (e.g. Asia) and the world.
dataset population data-package datapackage population-figures
Python 111 80 0 0 Updated on Apr 14 2020

Spark – Sor számlálás



- Python programnyelven egyszerű példa (sorok számlálása):

```
from pyspark import SparkContext, SparkConf

SparkMasterIP="192.168.10.85"
SparkContext.setSystemProperty('spark.executor.memory', '2g')
sc = SparkContext(appName="test", master="spark://"+SparkMasterIP+":7077")
text = sc.textFile("hdfs://"+SparkMasterIP+":9000/text.txt")
nonempty_lines = text.filter(lambda x: len(x) > 0)
print("Sorok száma: {}".format(text.count()))
print("Nem üres sorok száma: {}".format(nonempty_lines.count()))
sc.stop()

---

> Sorok száma: 91087
> Nem üres sorok száma: 44116
```

```
text.txt:
--------------
Sor: 91087 sor
Méret: 9,58 MB
Szó: 1 791 669
```

- Első lépésben importáljuk a programunkhoz szükséges csomagokat.
- Második lépésben létrehozzuk a SparkContext-et (megadjuk a menedzser (master) elérhetőségét és a végrehajtók memória méret igényét)
- Harmadik lépésben sc.textFile metódussal beolvassuk a szövegfájlunkat HDFS-ről RDD struktúrába
- Negyedik lépésben megszámoljuk a szöveget tartalmazó sorok számát
- Utolsó lépésben kiíratjuk a sorok (RDD-ben található rekordok) és a nemüres sorok számát (rdd.count())

Spark – Szavak megszámlálása

- Python programnyelven egyszerű példa (szavak számlálása):

```
from pyspark import SparkContext, SparkConf

SparkMasterIP="192.168.10.85"
SparkContext.setSystemProperty('spark.executor.memory', '2g')
sc = SparkContext(appName="test", master="spark://"+SparkMasterIP+":7077")
text = sc.textFile("hdfs://"+SparkMasterIP+":9000/text.txt")
nonempty_lines = text.filter(lambda x: len(x) > 0)
counts = nonempty_lines.flatMap(lambda line: line.split(" ")) \
        .map(lambda word: (word, 1)) \
        .reduceByKey(lambda a, b: a + b)
print(counts.take(10))
sc.stop()

[('“We', 813), ('start', 94), ('back,”', 22), ('Gared', 26), ('urged', 90),
 ('as', 13998), ('dark', 704), ('around', 1210), ('them.', 1355), ('wildlings', 239)]
```

```
text.txt:
--------------
Sor: 91087 sor
Méret: 9,58 MB
Szó: 1 791 669
```

- `nonempty_lines.flatMap(lambda line: line.split(" "))`
- `.map(lambda word: (word, 1))`
- `.reduceByKey(lambda a, b: a + b)`

Átlagos futási idő 4-6 másodperc (inicializálás nélkül)

Spark CSV feldolgozás

- Adatok begyűjtése
- Adatok elhelyezése a HDFS rendszeren
- Vizualizációhoz segédprogramok letöltése

```
!wget https://raw.githubusercontent.com/datasets/covid-19/main/data/countries-aggregated.csv
!/home/sparkuser/hadoop/bin/hdfs dfs -put ./countries-aggregated.csv /
!pip3 install pandas
```

Spark CSV feldolgozás - RDD

- Hasonló módon, mint a szövegfeldolgozásnál

```
from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession

SparkMasterIP="192.168.10.85"
SparkContext.setSystemProperty('spark.executor.memory', '2g')
sc = SparkContext(appName="test", master="spark://"+SparkMasterIP+":7077")
text = sc.textFile("hdfs://"+SparkMasterIP+":9000/countries-aggregated.csv")


['Date,Country,Confirmed,Recovered,Deaths',
 '2020-01-22,Afghanistan,0,0,0',
 '2020-01-23,Afghanistan,0,0,0',
 '2020-01-24,Afghanistan,0,0,0',
 '2020-01-25,Afghanistan,0,0,0']
```

Spark CSV feldolgozás - RDD

```
rdd = text.map(lambda line: line.split(","))

[['Date', 'Country', 'Confirmed', 'Recovered', 'Deaths'],
 ['2020-01-22', 'Afghanistan', '0', '0', '0'],
 ['2020-01-23', 'Afghanistan', '0', '0', '0'],
 ['2020-01-24', 'Afghanistan', '0', '0', '0'],
 ['2020-01-25', 'Afghanistan', '0', '0', '0']]

for line in rdd.take(10):
    print("Dátum: {}, Ország: {}, Megerősített: {}, Gyógyultak: {}, Elhunytak: {}".format(line[0]
,line[1],line[2],line[3],line[4]))

>>
Dátum: Date, Ország: Country, Megerősített: Confirmed, Gyógyultak: Recovered, Elhunytak: Deaths
Dátum: 2020-01-22, Ország: Afghanistan, Megerősített: 0, Gyógyultak: 0, Elhunytak: 0
Dátum: 2020-01-23, Ország: Afghanistan, Megerősített: 0, Gyógyultak: 0, Elhunytak: 0
```

Spark CSV feldolgozás - DataFramek

```
spark = SparkSession \
    .builder \
    .master("spark://"+SparkMasterIP+":7077") \
    .appName(„Covid")\
    .config("spark.executor.memory", "2g") \
    .getOrCreate()

dataframe = spark.read.format("csv") \
    .option("inferSchema", "true") \
    .option("header", "true") \
    .option("sep", ",")\
    .load("hdfs://"+SparkMasterIP+":9000/countries-aggregated.csv")

dataframe.printSchema()

>>
root
 |-- Date: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- Confirmed: integer (nullable = true)
 |-- Recovered: integer (nullable = true)
 |-- Deaths: integer (nullable = true)
```

Spark CSV feldolgozás - DataFramek

- DataFrame utolsó 20 elemének megjelenítése

```
dataframe.show()

>>
+----------+-----------+---------+---------+------+
|      Date|    Country|Confirmed|Recovered|Deaths|
+----------+-----------+---------+---------+------+
|2020-01-22|Afghanistan|        0|        0|     0|
|2020-01-23|Afghanistan|        0|        0|     0|
...
```

- DataFramen használhatóak az RDD során megismert metódusok nagyrésze

Spark CSV feldolgozás - DataFramek

ELKH Cloud

```
from pyspark.sql.functions import desc
dataframe_hun = dataframe.select("Date", "Confirmed", "Deaths")\
         .filter("Country = 'Hungary' and Confirmed <> 0")\
           .sort(desc("Date"))

dataframe_hun.show()
print("Napok száma, amikor a fertőzöttek száma > 0 Magyarországon: {} nap".format(dataframe_hun.count()))

>>
+----------+---------+------+
|      Date|Confirmed|Deaths|
+----------+---------+------+
|2021-02-19|   397116| 14145|
|2021-02-18|   394023| 14035|
...

Napok száma, amikor a fertőzöttek száma > 0 Magyarországon: 353 nap
```

Spark CSV feldolgozás - DataFramek

```
spark = SparkSession \
    .builder \
    .master("spark://"+SparkMasterIP+":7077") \
    .appName("PythonPi")\
    .config("spark.executor.memory", "2g") \
    .getOrCreate()

dataframe = spark.read.format("csv") \
    .option("inferSchema", "true") \
    .option("header", "true") \
    .option("sep", ",")\
    .load("hdfs://"+SparkMasterIP+":9000/countries-aggregated.csv")

dataframe.registerTempTable('covid19')
```

- Adattábla regisztrálás után a táblán SQL lekérdezések értelmezhetőek

Spark CSV feldolgozás - DataFramek

```
last_date = dataframe.select("Date").rdd.max()[0]
dataframe_last = spark.sql("select Country, Confirmed, Deaths \
                    from covid19 \
                    where Date = '" + str(last_date) + "' \
                    order by Confirmed desc")
dataframe_last.show()

>>
+--------------+---------+------+
|       Country|Confirmed|Deaths|
+--------------+---------+------+
|            US| 28004315|495693|
|         India| 10977387|156212|
|        Brazil| 10084208|244737|
|United Kingdom|  4107286|120147|
|        Russia|  4092649| 81048|
|        France|  3596167| 83543|
|         Spain|  3133122| 67101|
|         Italy|  2780882| 95235|
|        Turkey|  2624019| 27903|
|       Germany|  2381259| 67741|
|      Colombia|  2217001| 58511|
|     Argentina|  2054681| 51000|
|        Mexico|  2030491|178965|
|        Poland|  1623218| 41823|
|          Iran|  1558159| 59341|
|  South Africa|  1500677| 48859|
|       Ukraine|  1340054| 26320|
|     Indonesia|  1263299| 34152|
|          Peru|  1261804| 44489|
|       Czechia|  1134957| 18913|
+--------------+---------+------+
```

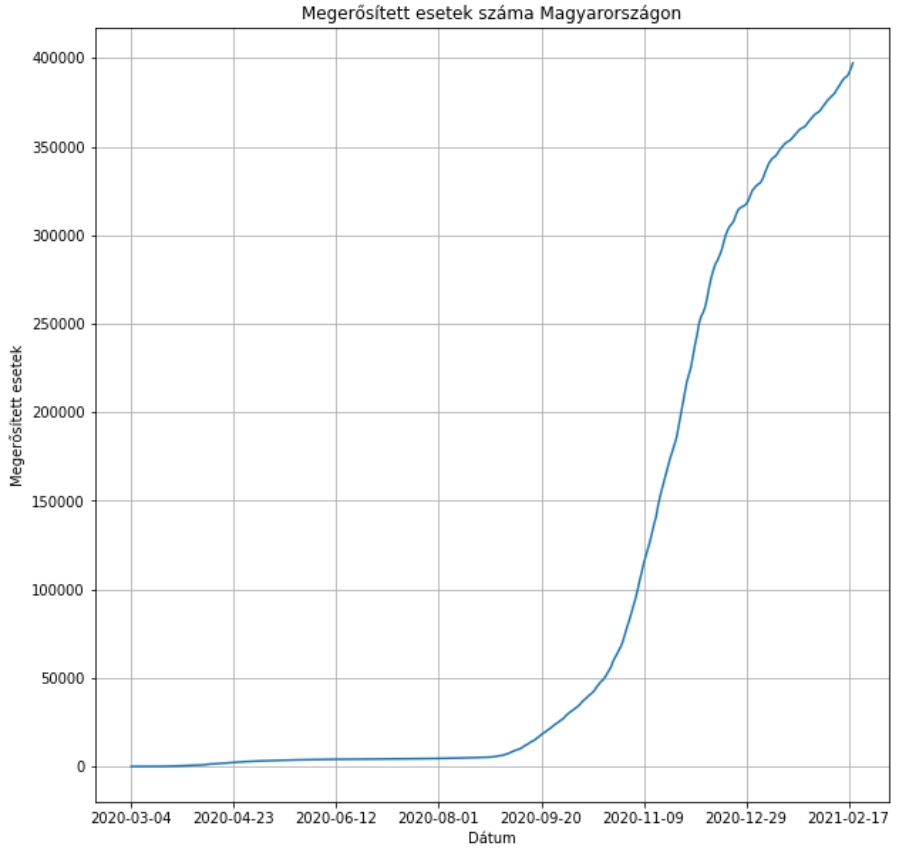
Spark CSV feldolgozás - DataFramek

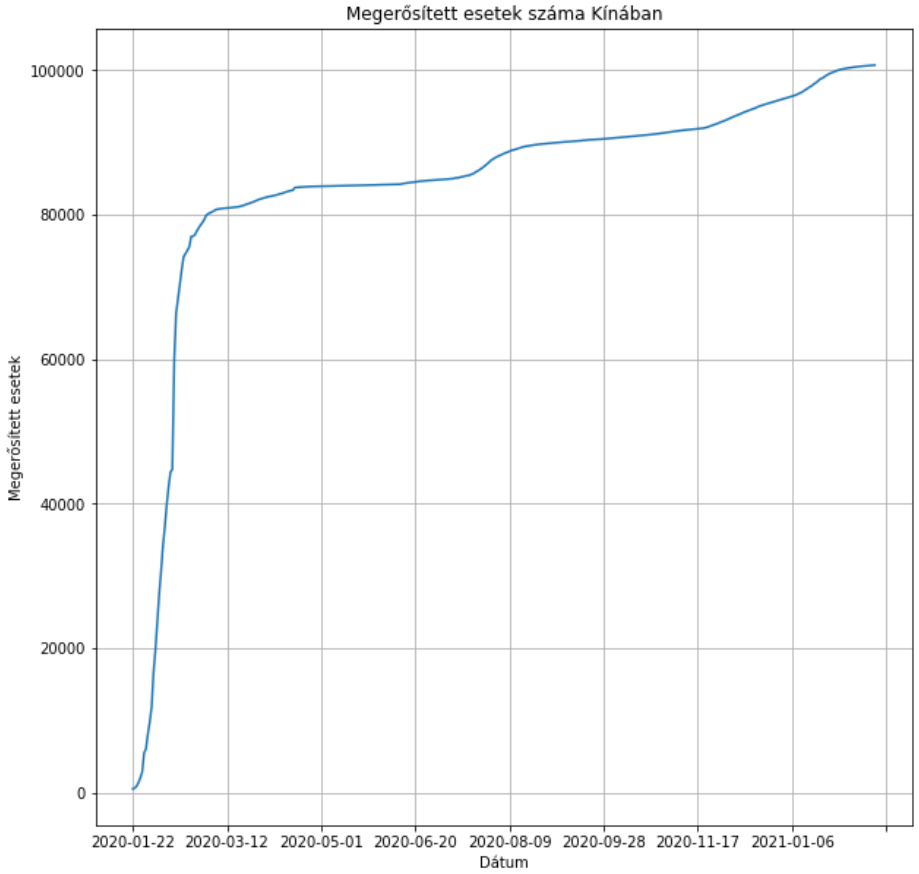
```
dataframe_hun = spark.sql("select Date, Confirmed \
                  from covid19 \
                  where Confirmed > 0 and Country = 'Hungary' \
                  order by Date")
dataframe_hun.show()

>>
+----------+---------+
|      Date|Confirmed|
+----------+---------+
|2020-03-04|        2|
|2020-03-05|        2|
```

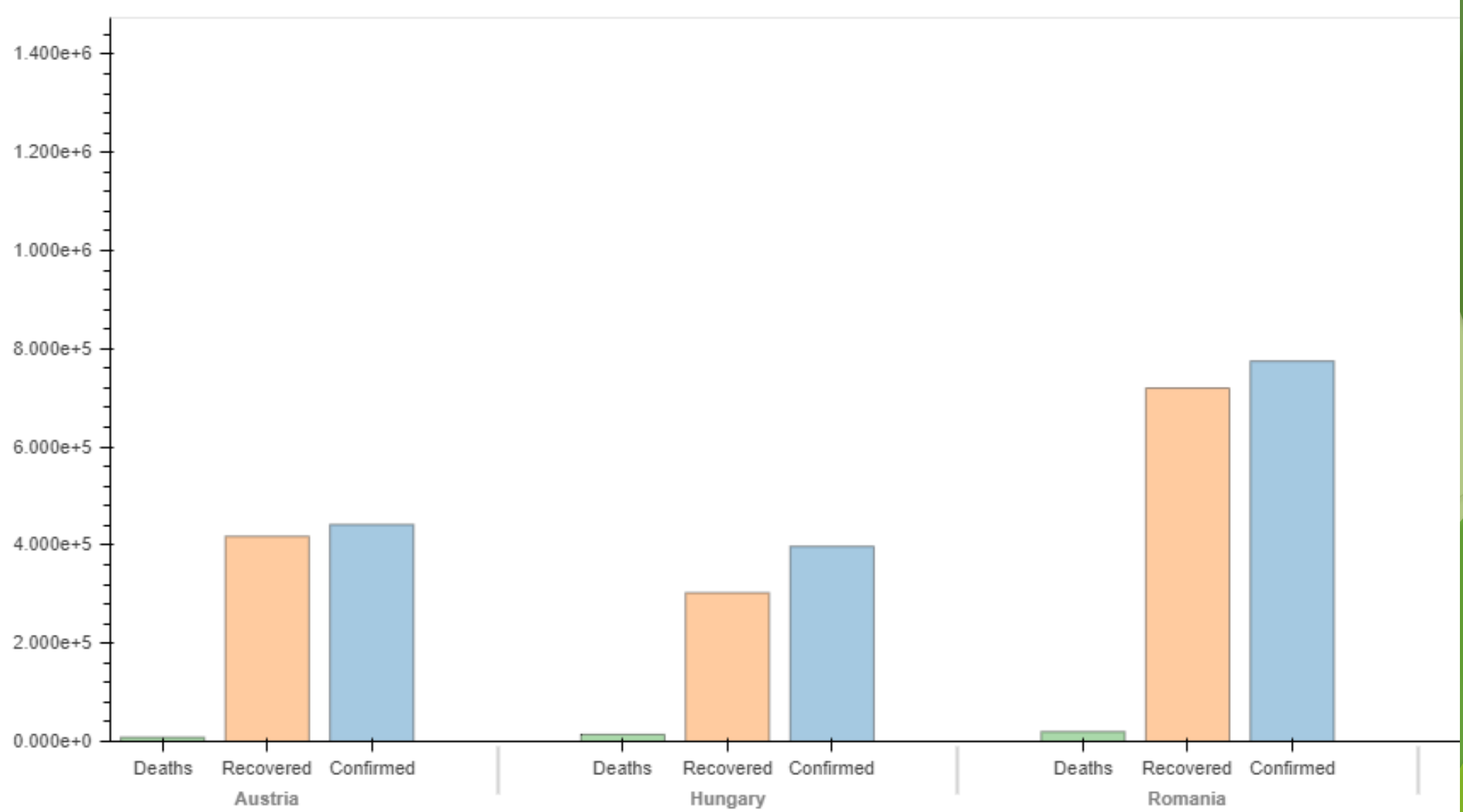
Spark CSV feldolgozás – DataFramek vizualizálása

```
dataframe_hun.toPandas().plot(figsize=(10,10), grid=True, ylabel='Megerősített esetek', xlabel="Dátum", x='Date’,
title="Megerősített esetek száma Magyarországon", legend=False)
```

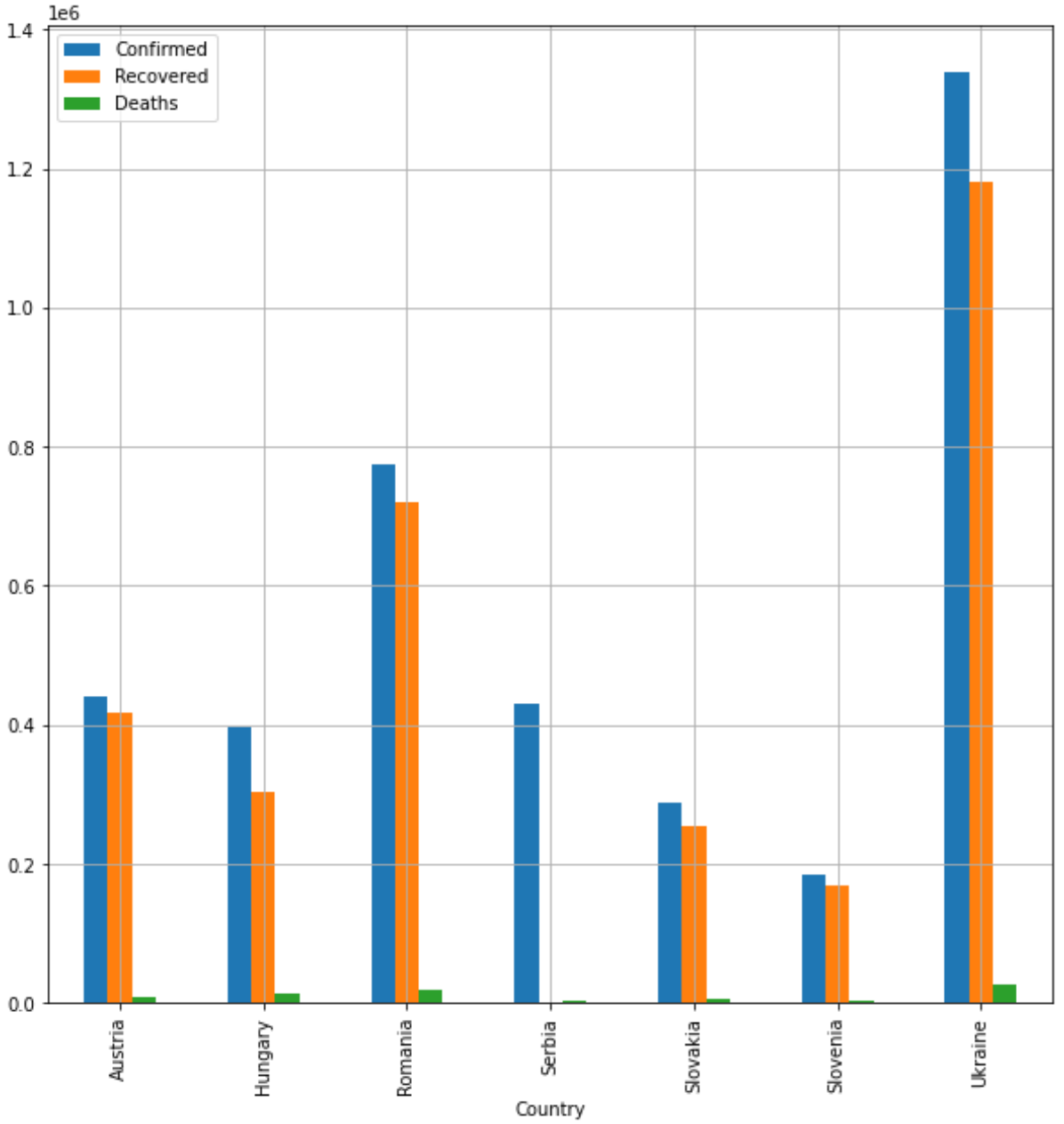




Spark CSV feldolgozás – DataFramek vizualizálása

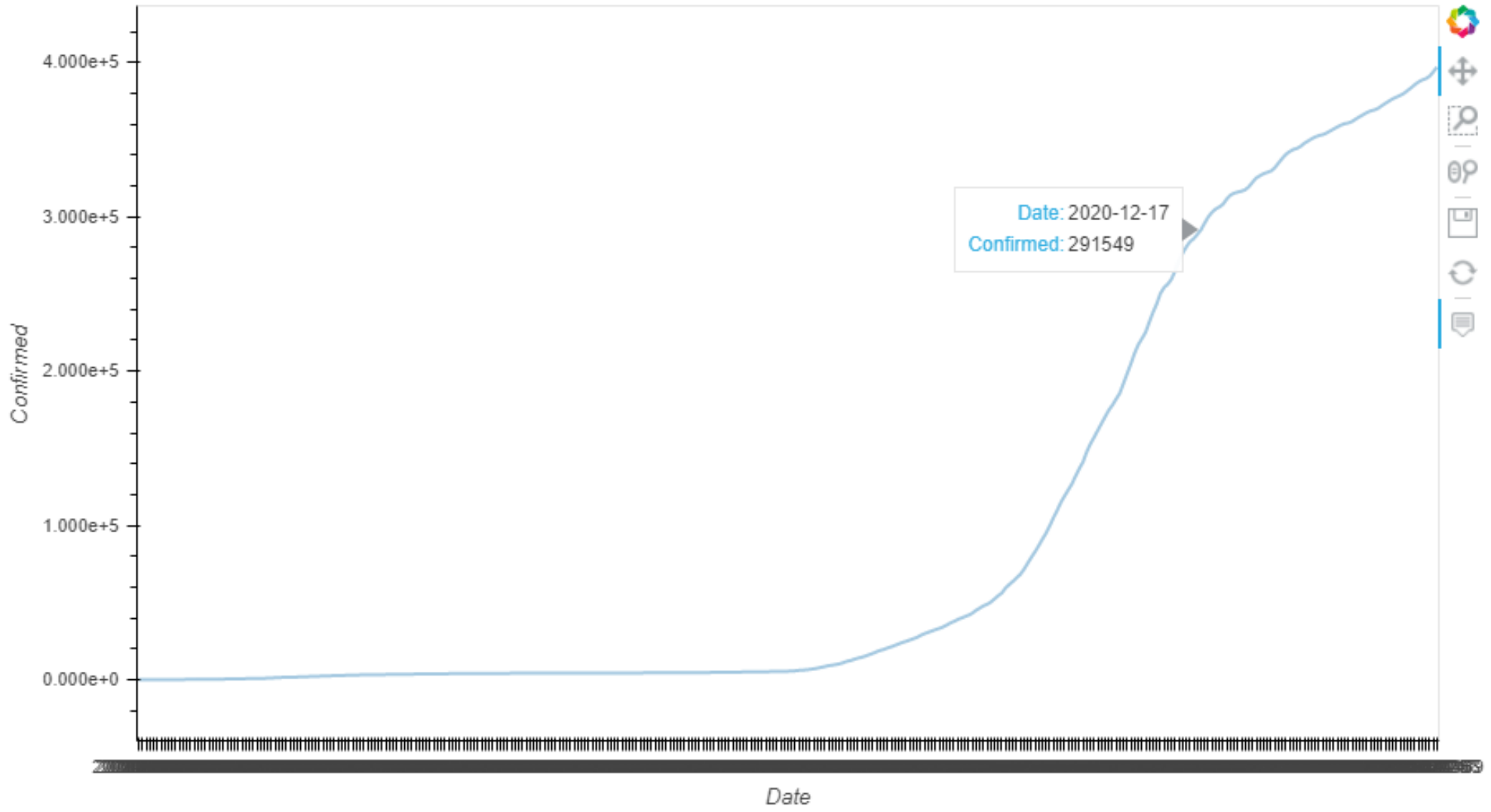


Spark CSV feldolgozás – DataFramek vizualizálása



Spark CSV feldolgozás - DataFramek vizualizálása

- Python adat vizualizálási eszközök:
  - Matplotlib, Bokeh, hvplot, Altair, seaborn

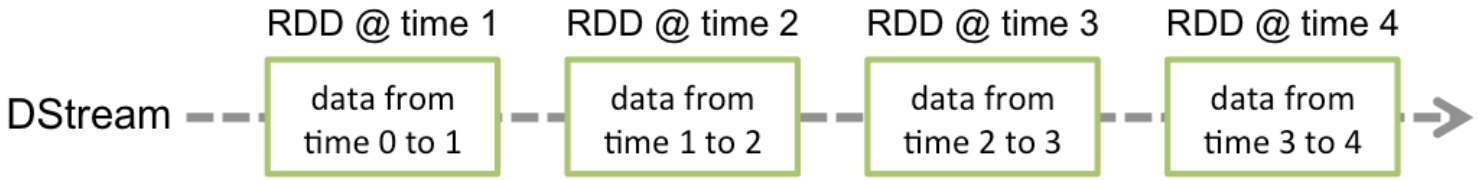


Spark adatfolyam feldolgozás

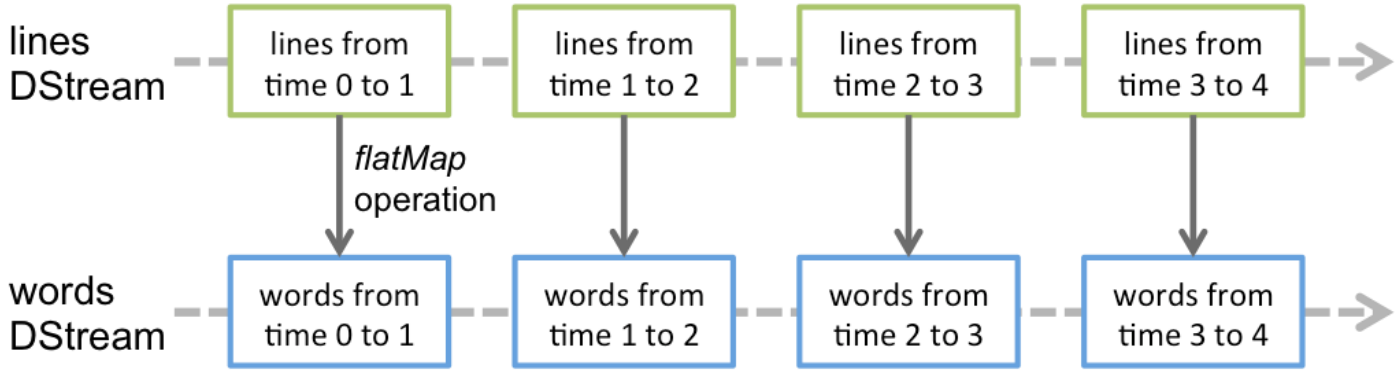
- A Spark Streaming magas szintű absztrakciót biztosít
  - DStream adatstuktúrát nyújt
  - Folyamatos adatfolyam dinamikus lekezlése
- A DStreams létrehozható különböző adatforrásból
  - Socket Stream
  - Kafka
  - Kinesis

Spark adatfolyam feldolgozás

- Belsőleg a DStream RDD-k sorozataként jelenik meg
- A DStream minden RDD-je egy bizonyos intervallum adatait tartalmazza



- A DStream-en alkalmazott bármely művelet az alapul szolgáló RDD-k műveleteinek felel meg



Spark adatfolyam feldolgozás

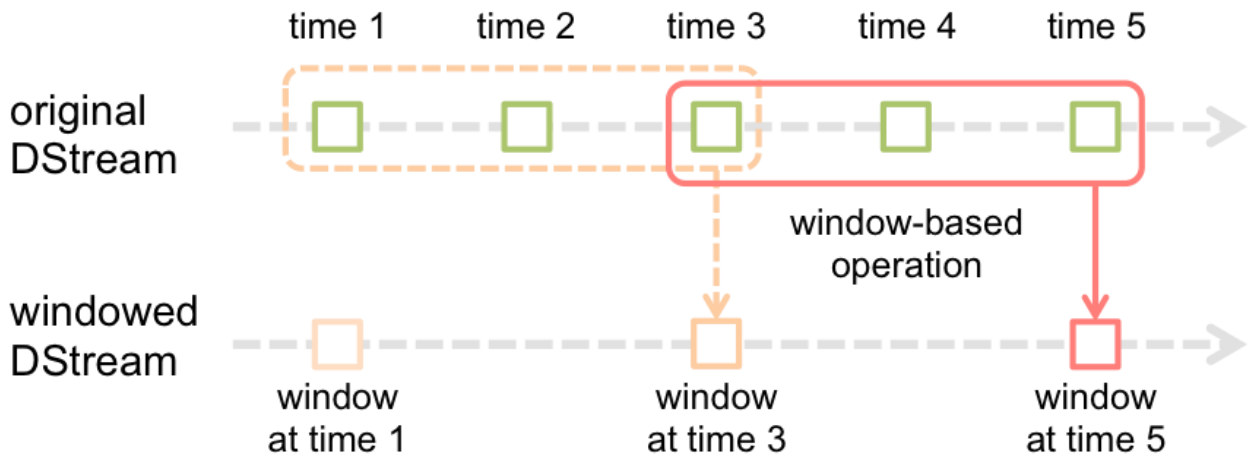
- A Spark Streaming két kategóriába sorolható a beépített streaming forrásokat:
  - Alapforrások: közvetlenül használhatóak StreamingContext API keresztül (pl. fájlrendszerek, socket kapcsolatok)
  - Haladó források: olyan források, melyek külön segédprogramokon (Spark plugin) keresztül érhetők el (pl. Kafka, Kinesis)

Spark adatfolyam feldolgozás – Műveletek

- Az RDD-khez hasonlóan a transzformációk lehetővé teszik a bemenő DStream adatainak módosítását
- A DStreamek támogatják a normál Spark RDD-ken elérhető transzformációkat (count, map, filter, ...)
- További streamekkel kapcsolatos műveletek:
- transform() – Lehetőséget ad a DStreameken tetszőleges számú RDD művelet végrehajtására. Bármely RDD művelet alkalmazására használható, amelyet a DStream API nem támogat
- updateStateByKey() - Visszaad egy új állapotú DStream-t, ahol az egyes kulcsok állapota frissül úgy, hogy az adott függvényt a kulcs előző állapotára és a kulcs új értékeire alkalmazza. Ez felhasználható tetszőleges állapotadatok fenntartására az egyes kulcsokhoz.

Spark adatfolyam feldolgozás – Csúszóablak

- A Spark Streaming csúszóablakos számításokat is biztosít, amelyek lehetővé teszik a transzformációk alkalmazását egy csúszó adatablakon

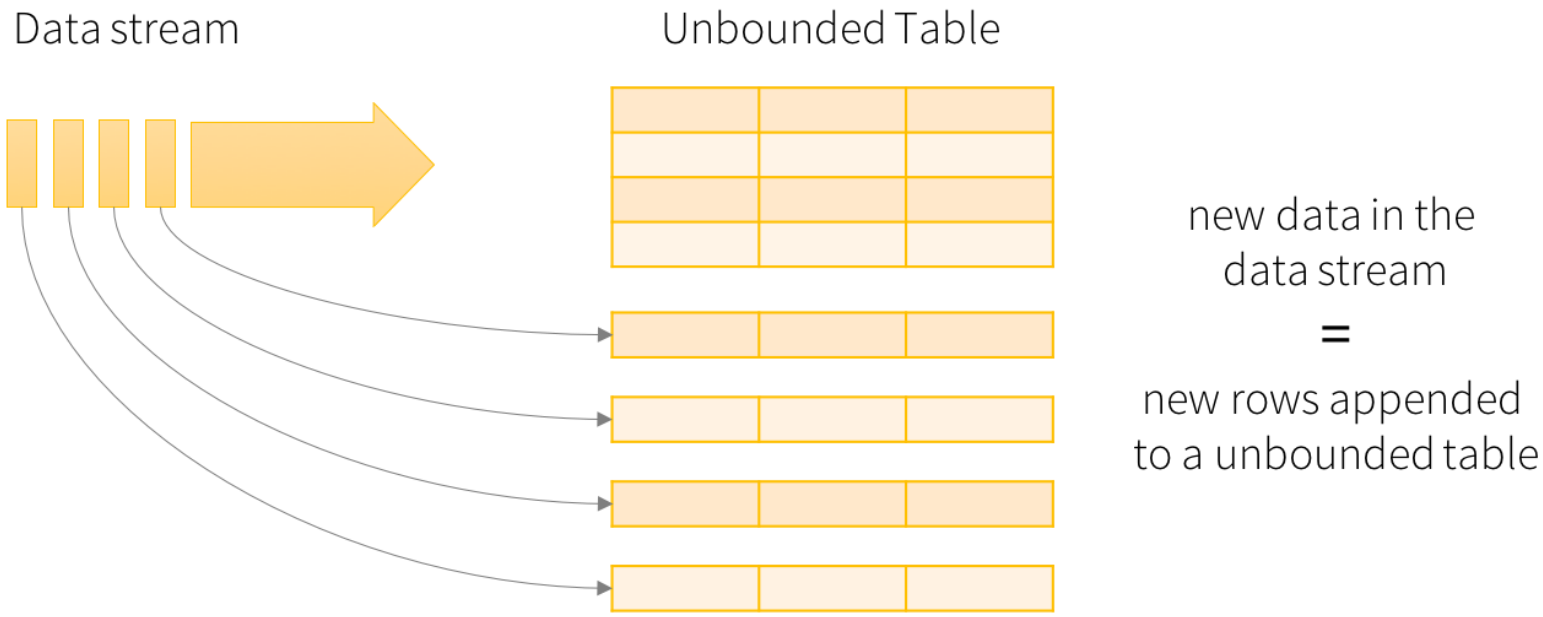


- A csúszóablak során két paramétert fontos megadni:
  - A csúszóablak időtartamát
  - Csúszási időtartam, mely során leképezésre kerül a RDD adathalmaz
- Minden alkalommal, amikor az ablak áthúzódik egy forrás DStream felett, a forrás RDD-k, amik egy ablakba esnek összevonásra kerülne, és egy RDD készül az ablakban értelmezett DStreamből. Ábrán látható példában műveletet az adatok utolsó 3 időegységén alkalmazzuk, és 2 időegységgel csúsztatjuk az ablakot

Spark adatfolyam feldolgozás - DStream kimeneti műveletek

- A kimeneti műveletek lehetővé teszik a DStream adatainak exportálását külső rendszerekbe (pl. adatbázis, fájlrendszer)
- A kimeneti műveletek a Dstreamek esetében a transzformációkat kiértékeli (hasonlóan RDD-ben használt action)
  - (p)print() – Kiírja az adatfolyamok első tíz elemét egy DStream-ben a streaming alkalmazást futtató állomáson.
  - saveAsTextFiles(előtag, [utótag]) – Elmeneti a DStream tartalmát szöveges formátumban. Az előtag paraméter fogja a fájl elnevezést tartalmazni, az utótag, pedig a kiterjesztésért felelős.
  - foreachRDD(func) – A legáltalánosabb kimeneti operátor, amely a func függvényt alkalmazza a streamből generált RDD-kre (összesre). Ennek a funkciónak az egyes RDD-kben lévő adatokat egy külső rendszerbe szükséges kiküldenie, például menteni kell az RDD-t fájlokba, vagy a hálózaton keresztül írni egy adatbázisba.

Spark strukturált adatfolyam feldolgozás

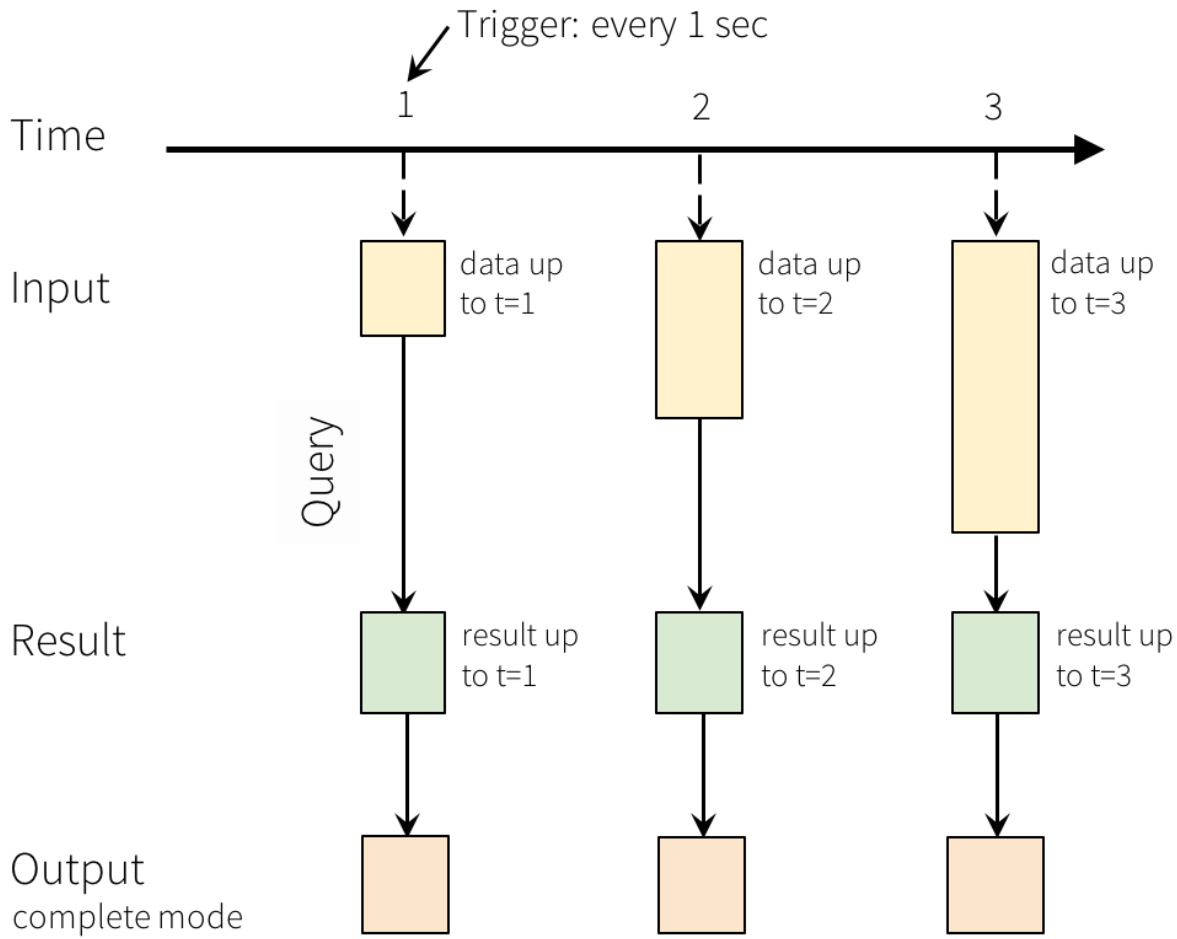


Data stream as an unbounded table

- A strukturált Streaming egy skálázható és hibatűrő adatfeldolgozó motor, amely a Spark SQL motorra épül
- A műveletek ugyanúgy fejezhetőek ki, mint statikus adatok esetében
- A bemeneti adatok tekinthetőek egy bemeneti táblának
  - Minden adat, amely az adatfolyamra érkezik, úgy viselkedik, mintha egy új sort csatolnának a beviteli táblához
  - A bemeneten található lekérdezés létrehozza az eredmény táblát

Spark strukturált adatfolyam feldolgozás

- Minden kiváltó esemény (trigger) intervallum során (pl. 1 mp), új sorok kerülnek a bemeneti táblához, amely végül frissíti az eredménytáblát



Programming Model for Structured Streaming

Spark Streaming - Tárolás

- A kimenet perzisztálása többféle módon határozható meg:
  - Teljes mód - A teljes frissített eredménytábla a külső tárhelyre kerül. A tárolást biztosító eszköztől függ, hogy hogyan kezeli le a teljes tábla írását
  - Hozzáfűzési mód - Csak az eredménytáblához csatolt új sorok íródnak a külső tárhelyre. Ez csak azokra a lekérdezésekre vonatkozik, ahol az eredménytáblázat meglévő sorai várhatóan nem változnak
  - Frissítési mód - Csak azokat a sorokat írjuk a külső tárhelyre, amelyeket az eredménytáblázatban frissítettek az utolsó kiváltó esemény (trigger) óta. Ez a mód csak azokat a sorokat írja ki, amelyek az utolsó trigger óta megváltoztak. Ha a lekérdezés nem tartalmaz összesítéseket, akkor egyenértékű lesz a hozzáfűzési móddal

Spark adatfolyam feldolgozás Twitter # analízis

- Python használata adatfolyam létrehozáshoz

```
import socket

s = socket.socket()
host = "192.168.10.85"
port = 5555
s.bind((host, port))
print('socket is ready')
s.listen(4)
print('socket is listening')
c_socket, addr = s.accept()
print("Received request from: " + str(addr))
sendData(c_socket, keyword = [bigdata','job','ML', 'programming'])
```

Spark adatfolyam feldolgozás Twitter # analízis

- Tweepy könyvtár használata tweetek lekérdezésére

```
consumer_key=
consumer_secret=
access_token=
access_token_secret=
```

```
class TweetsListener(StreamListener):
  def __init__(self, csocket):
    self.client_socket = csocket
  def on_data(self, data):
    try:
      msg = json.loads( data )
      print("new message")
      # if tweet is longer than 140 characters
      if "extended_tweet" in msg:
        # add at the end of each tweet "t_end"
        self.client_socket \
            .send(str(msg['extended_tweet']['full_text']+"t_end") \
            .encode('utf-8'))
        print(msg['extended_tweet']['full_text'])
      else:
        # add at the end of each tweet "t_end"
        self.client_socket\
            .send(str(msg['text']+"t_end")\
            .encode('utf-8'))
        print(msg['text'])
      return True
    except BaseException as e:
        print("Error on_data: %s" % str(e))
    return True
  def on_error(self, status):
    print(status)
    return True
```

Spark adatfolyam feldolgozás Twitter # analízis

- PySpark programunk

```
from pyspark import SparkConf,SparkContext
from pyspark.streaming import StreamingContext
from pyspark.sql import Row,SQLContext
import sys
import requests

conf = SparkConf()
conf.setAppName("TwitterStreamApp")
sc = SparkContext(conf=conf)
sc.setLogLevel("ERROR")
# creat the Streaming Context with window size 2 seconds
ssc = StreamingContext(sc, 2)
# setting a checkpoint to allow RDD recovery
ssc.checkpoint("checkpoint_TwitterApp")
dataStream = ssc.socketTextStream("192.168.10.85",5555)
```

Spark adatfolyam feldolgozás Twitter # analízis

ELKH Cloud

- PySpark programunk #2

```
def get_sql_context_instance(spark_context):
    if ('sqlContextSingletonInstance' not in globals()):
        globals()['sqlContextSingletonInstance'] = SQLContext(spark_context)
    return globals()['sqlContextSingletonInstance']
def process_rdd(time, rdd):
    print("----------- %s -----------" % str(time))
    try:
        # Get spark sql singleton context from the current context
        sql_context = get_sql_context_instance(rdd.context)
        # convert the RDD to Row RDD
        row_rdd = rdd.map(lambda w: Row(hashtag=w[0], hashtag_count=w[1]))
        # create a DF from the Row RDD
        hashtags_df = sql_context.createDataFrame(row_rdd)
        # Register the dataframe as table
        hashtags_df.registerTempTable("hashtags")
        # get the top 10 hashtags from the table using SQL and print them
        hashtag_counts_df = sql_context.sql("select hashtag, hashtag_count from hashtags order by has
htag_count desc limit 10")
        hashtag_counts_df.show()
        # call this method to prepare top 10 hashtags DF and send them
        send_df_to_dashboard(hashtag_counts_df)
    except:
        e = sys.exc_info()[0]
        print("Error: %s" % e)
```

Spark adatfolyam feldolgozás Twitter # analízis

- PySpark programunk

```
def aggregate_tags_count(new_values, total_sum):
    return sum(new_values) + (total_sum or 0)

words = dataStream.flatMap(lambda line: line.split(" "))
# filter the words to get only hashtags, then map each hashtag to be a
pair of (hashtag,1)
#hashtags = words.map(lambda x: (x, 1))
hashtags = words.filter(lambda w: '#' in w).map(lambda x: (x, 1))
# adding the count of each hashtag to its last count
tags_totals = hashtags.updateStateByKey(aggregate_tags_count)
# do processing for each RDD generated in each interval
tags_totals.foreachRDD(process_rdd)

# start the streaming computation
ssc.start()
# wait for the streaming to finish
ssc.awaitTermination()
```

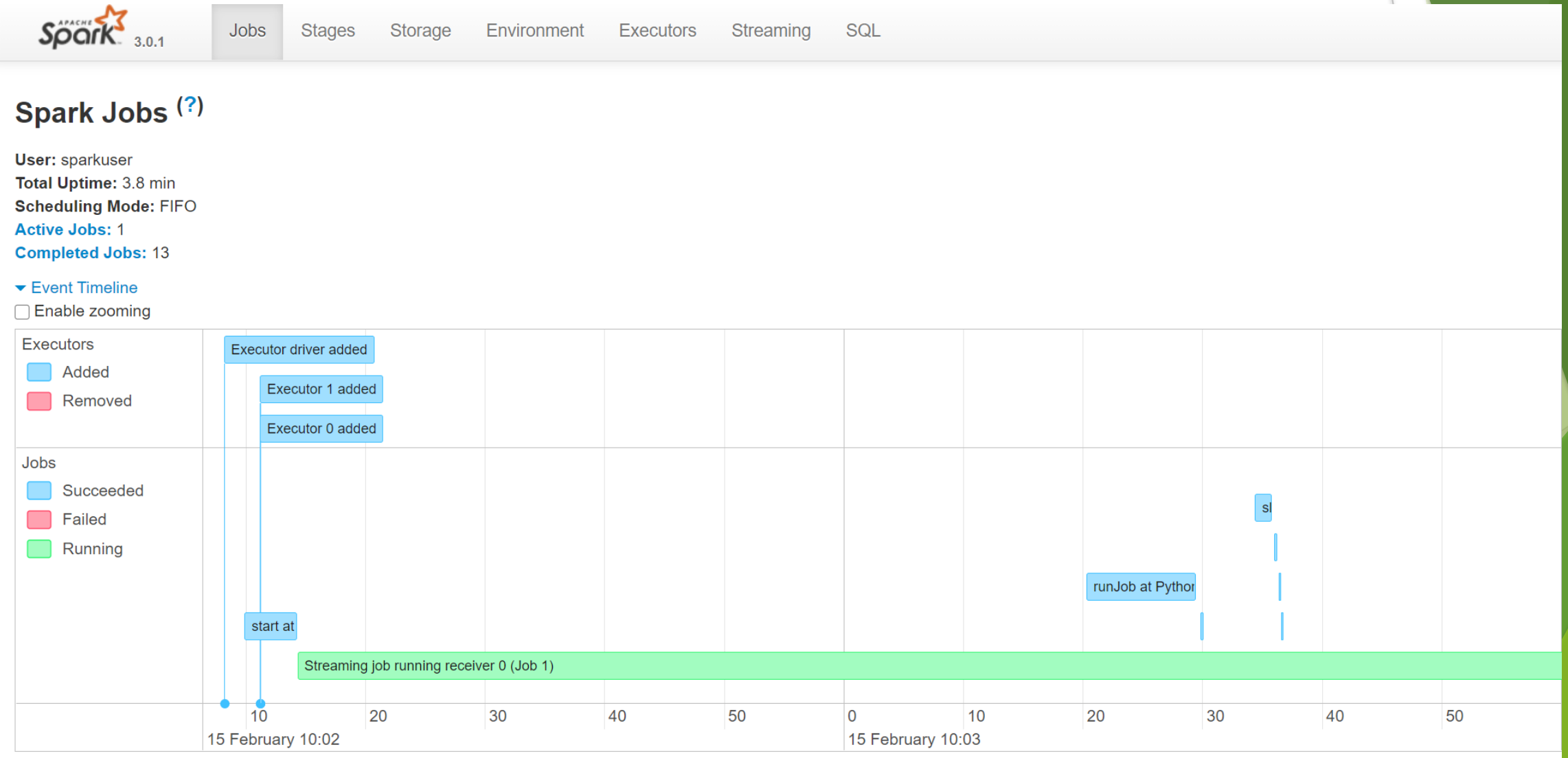
Spark adatfolyam feldolgozás Twitter # analízis

- Eredmény:

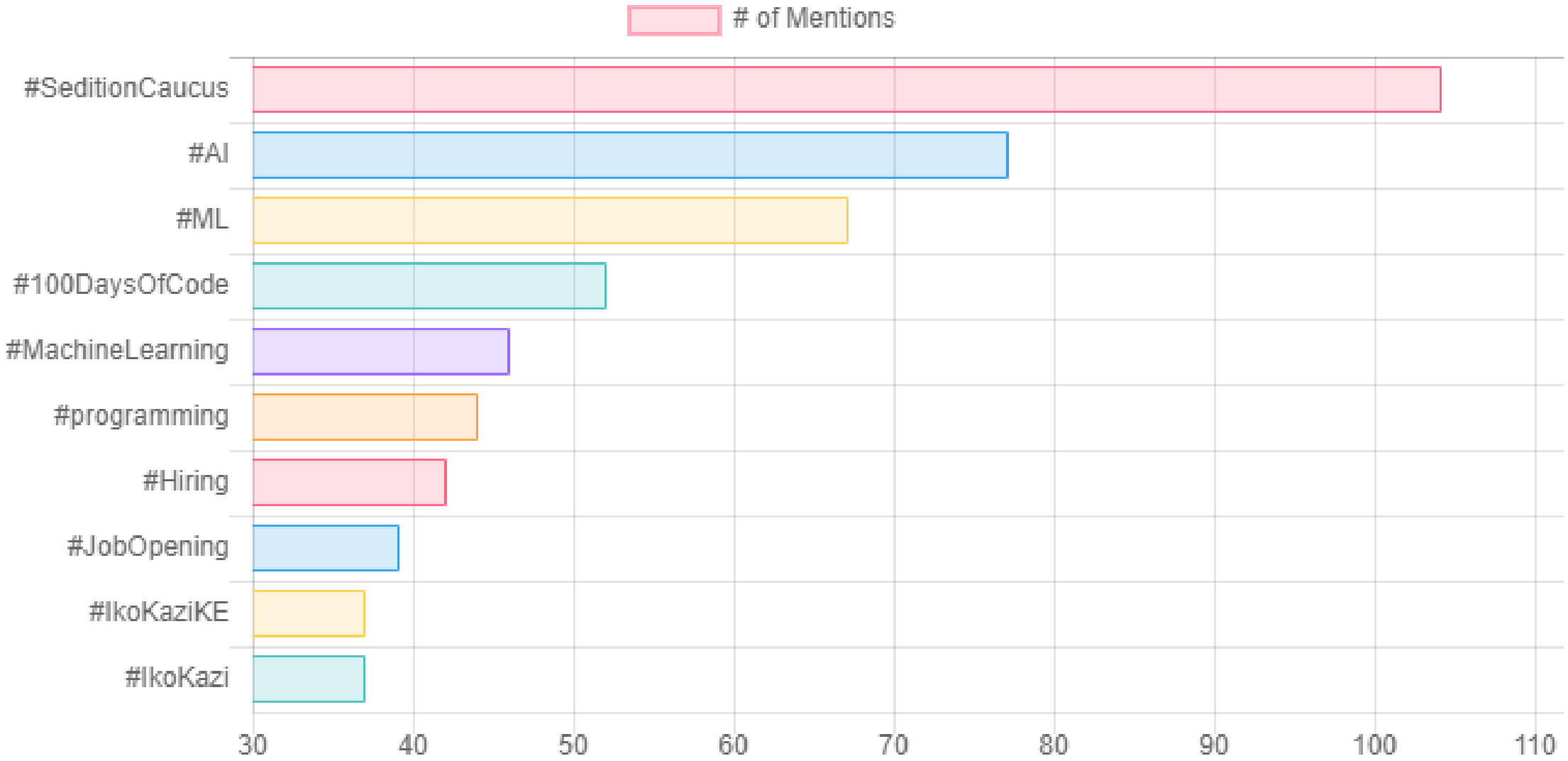
hashtag	hashtag_count
#SeditionCaucus	96
#AI	56
#ML	52
#100DaysOfCode	43
#Hiring	42
#JobOpening	39
#programming	39
#MachineLearning	38
#IkoKazi	37

Spark adatfolyam feldolgozás Twitter # analízis

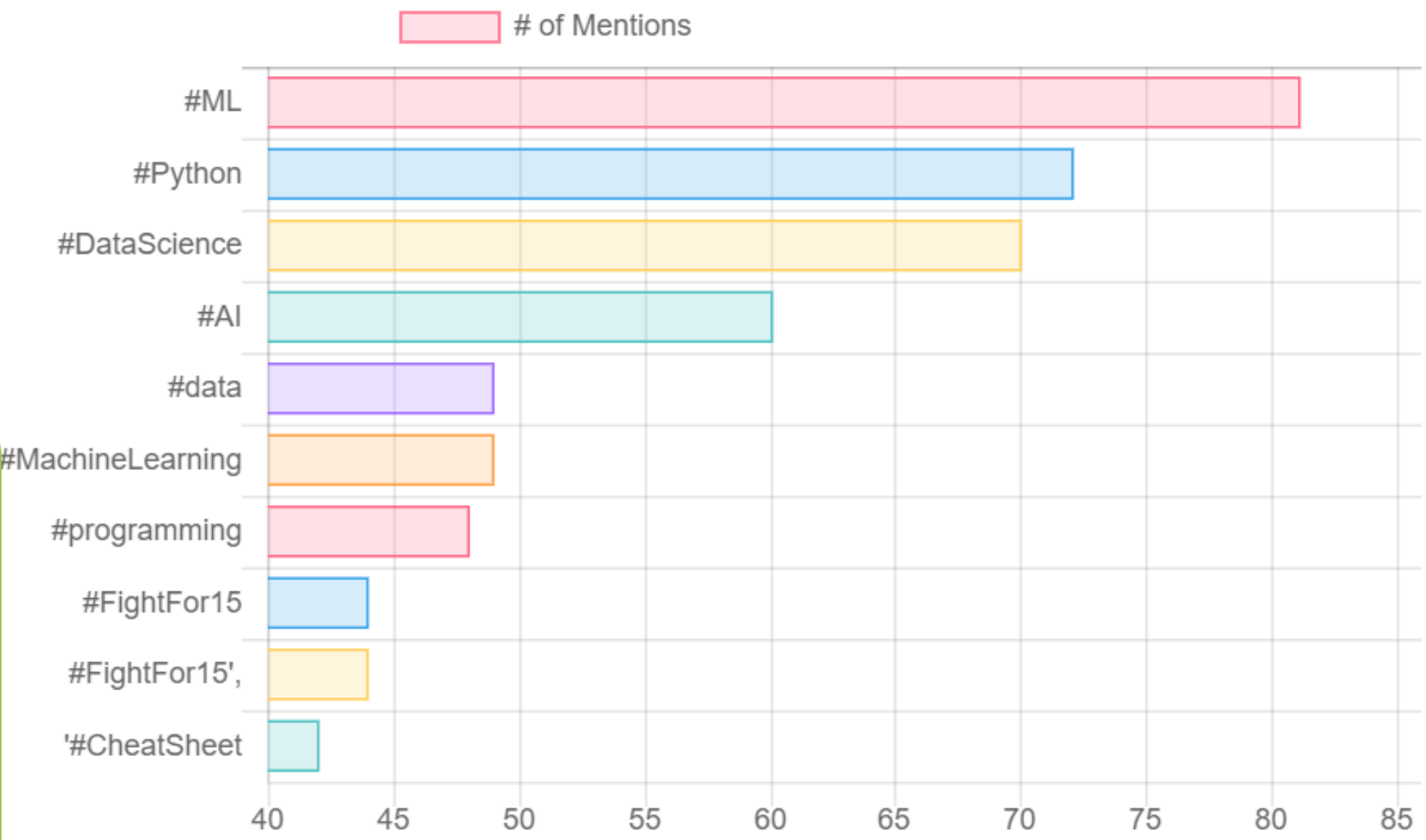
ELKH Cloud



Top Trending Twitter Hashtags



Top Trending Twitter Hashtags

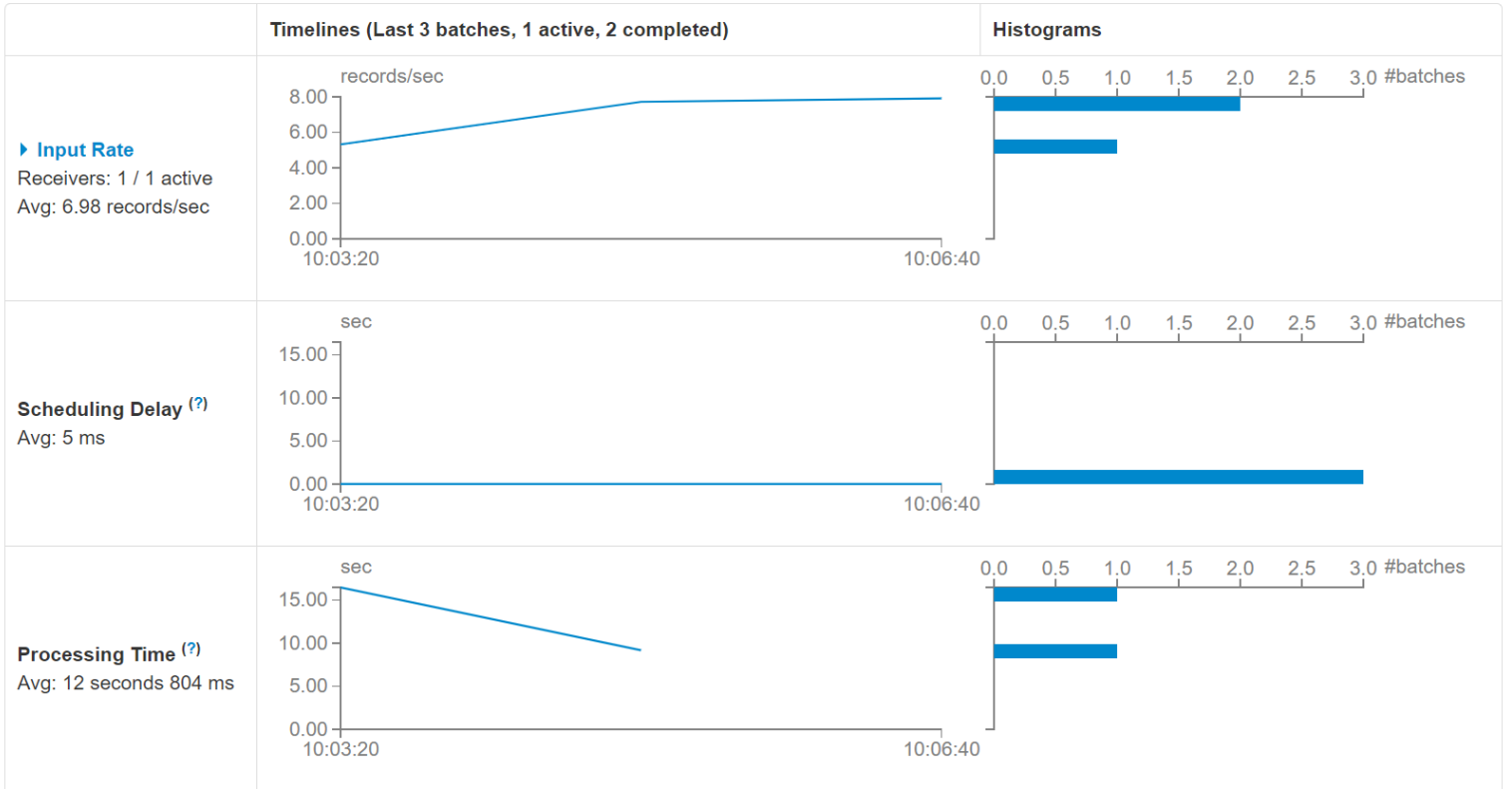


ELKH Cloud

Spark adatfolyam feldolgozás Twitter # analízis

Streaming Statistics

Running batches of **1 minute 40 seconds** for **4 minutes 30 seconds** since **2021/02/15 10:02:14** (**2** completed batches, **2095** records)



Active Batches (1)

Batch Time	Records	Scheduling Delay (?)	Processing Time (?)
2021/02/15 10:06:40	791 records	2 ms	-

Completed Batches (last 2 out of 2)

Batch Time	Records	Scheduling Delay (?)	Processing Time (?)
2021/02/15 10:05:00	772 records	1 ms	9 s
2021/02/15 10:03:20	532 records	14 ms	16 s

Spark adatfolyam feldolgozás Twitter hangulatelemzés

- Alkalmazásunk bővítése
  - Hangulatelemzés
  - TextBlob (python könyvtár)
  - Térség analízise

ELKH Cloud

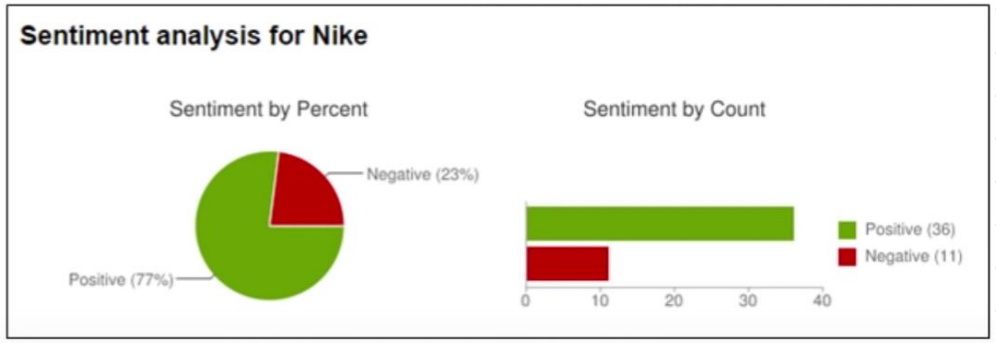


Figure: *Twitter Sentiment Analysis For Nike*

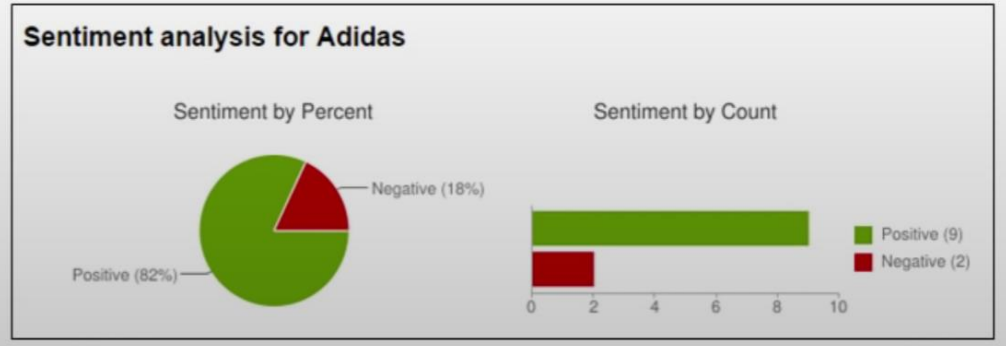
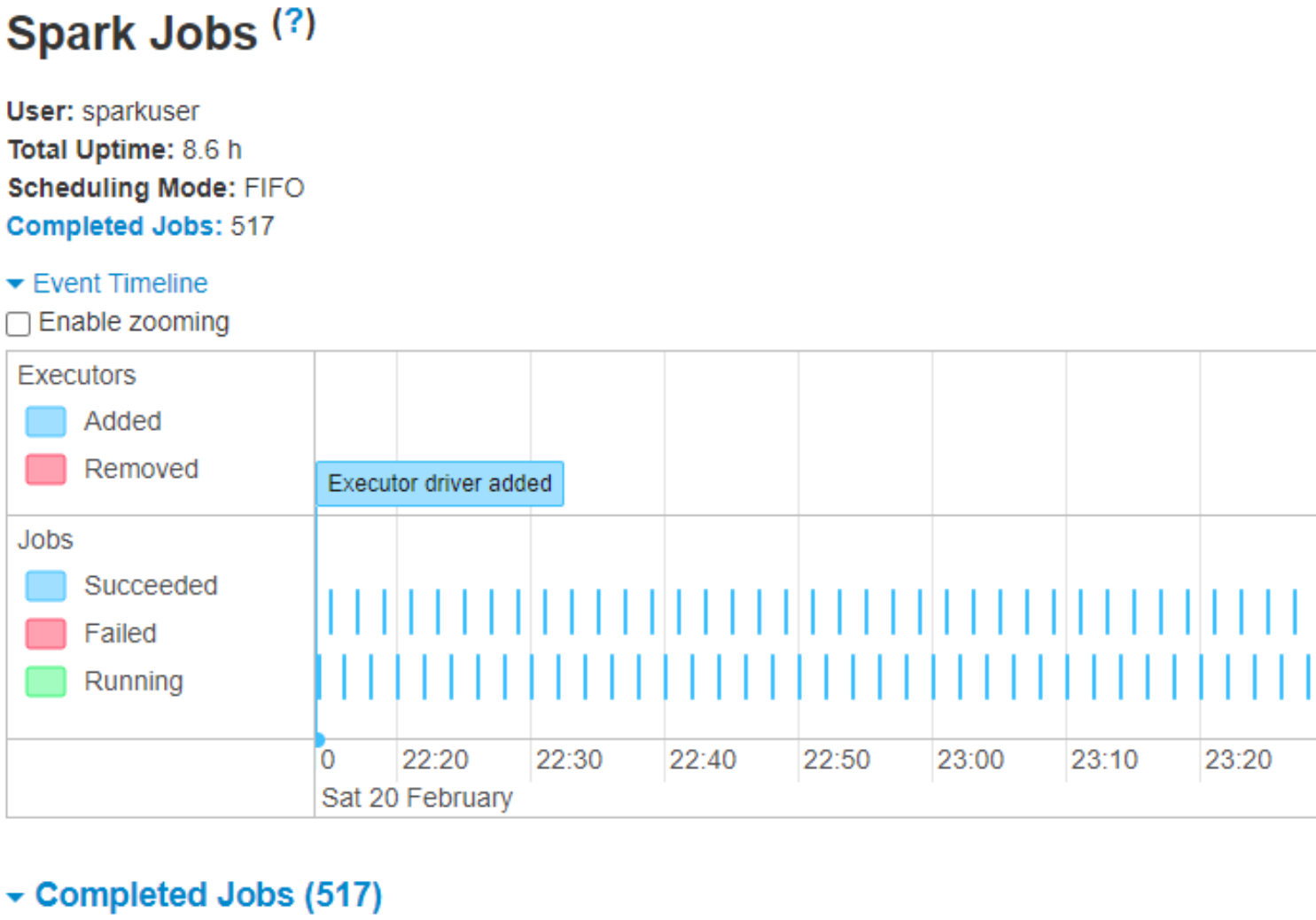


Figure: *Twitter Sentiment Analysis For Adidas*

word	polarity	subjectivity
Here's how you know Joe Biden is doing an awful job...	-1.0	1.0
Great Job Ladies!!	1.0	0.75
I swear TSSF is the best band to blast in the car	1.0	0.3
Great job!!! ♡♡♡♡	1.0	0.75
Amazing job, friend!!!	1.0	0.9
Congratulations to Milfordsoar junior Eli Vondra as he finishes 6th at the Class C 132 lb State ...	1.0	0.75
Awesome job, congrats!	1.0	1.0
in the beginning every year they do a great job setting up the collegehoops superfans to Bel...	1.0	0.75
Great job today Janice!	1.0	0.75
Great job today girls! So proud of you all♡♡♡ WeAreBethpage	0.9	0.875
Just a reminder to all you wonderful mummy's out there, you are doing a great job and don't ...	0.9	0.875
The police officers face says it all. Brilliant	0.9	1.0
When we come together, we make a difference! GOOD JOB EVERYONE!	0.875	0.600000000000...
What's a wolf shifter to do when he finds a beautiful vampire unconscious in a crashed car?	0.85	1.0
And I'm horrible dentist. I'm bad at my job which is cleaning up teeth	-0.84999999999...	0.833333333333...
Please tell your daughter good luck and welcome aboard! This job is a great ride.	0.833333333333...	0.75
Hate to see this for , car will not start, seems to not have any power whatsoever.	-0.8	0.9
I hate those " it's their job , so they'll do it " mfs	-0.8	0.9
Great photo and great job by the pilots UA328	0.8	0.75
the fact that I'm at a wedding where I know people and I'm still hiding in my car because I do...	0.8	0.7
Ever since he was appointed we've had media pundits and experts telling us he was doing ...	0.8	0.75
Gonna fly through the field.	0.8	0.9
Thank you, great job.	0.8	0.75
Can we get some doggy loving celebs to retweet this ? Our boy PADDY living his best life ♡...	0.8	0.625
Great job	0.8	0.75
Great job	0.8	0.75
Happy to out w/ & others	0.8	1.0
It's with great sorrow we learn that Sgt. Jeffrey Reidy of the Department has passed away. ...	0.8	0.75
nope, it's draining slower today, i didn't, i didn't do a great job	0.8	0.75
You've done a great job again ♡	0.8	0.75
Great Job Officers,Thank You.	0.8	0.75
...Understanding everything you had to go through to do that remarkable job today... I think y...	0.75	0.75

Spark adatfolyam feldolgozás Twitter # analízis

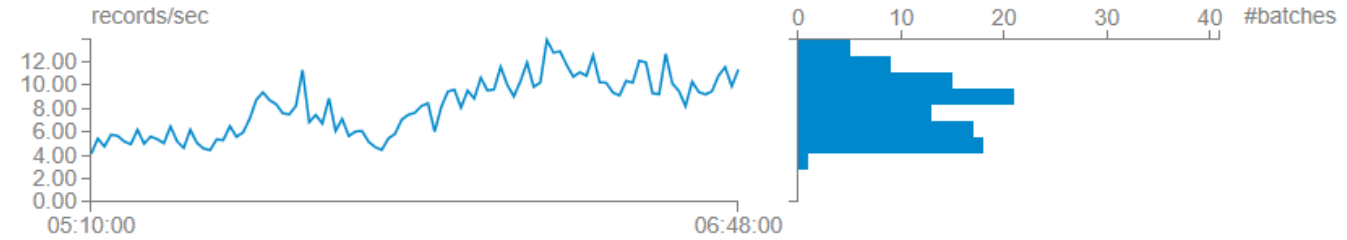
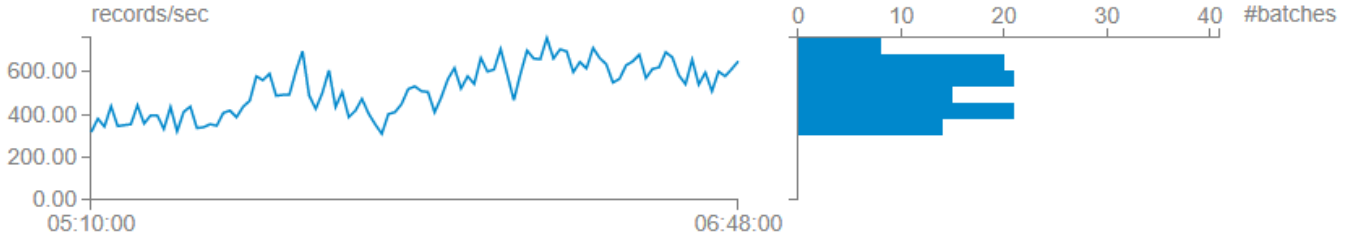
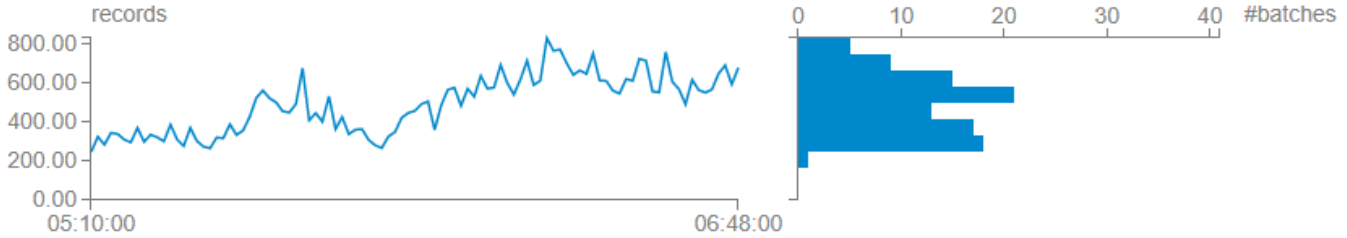
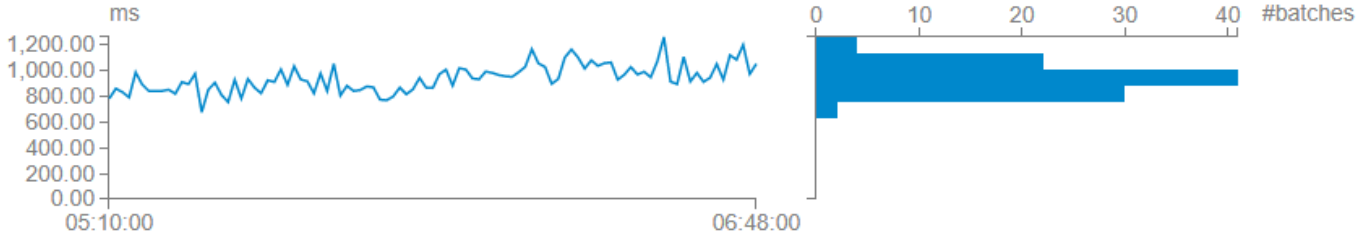
- Adatgyűjtés, későbbi elemzésekre
  - 438375 bejegyzés
  - Méret: ~ 27 MB



Streaming Query Statistics

Running batches for **8 hours 34 minutes 55 seconds** since **2021/02/20 22:14:04** (**515** completed batches)

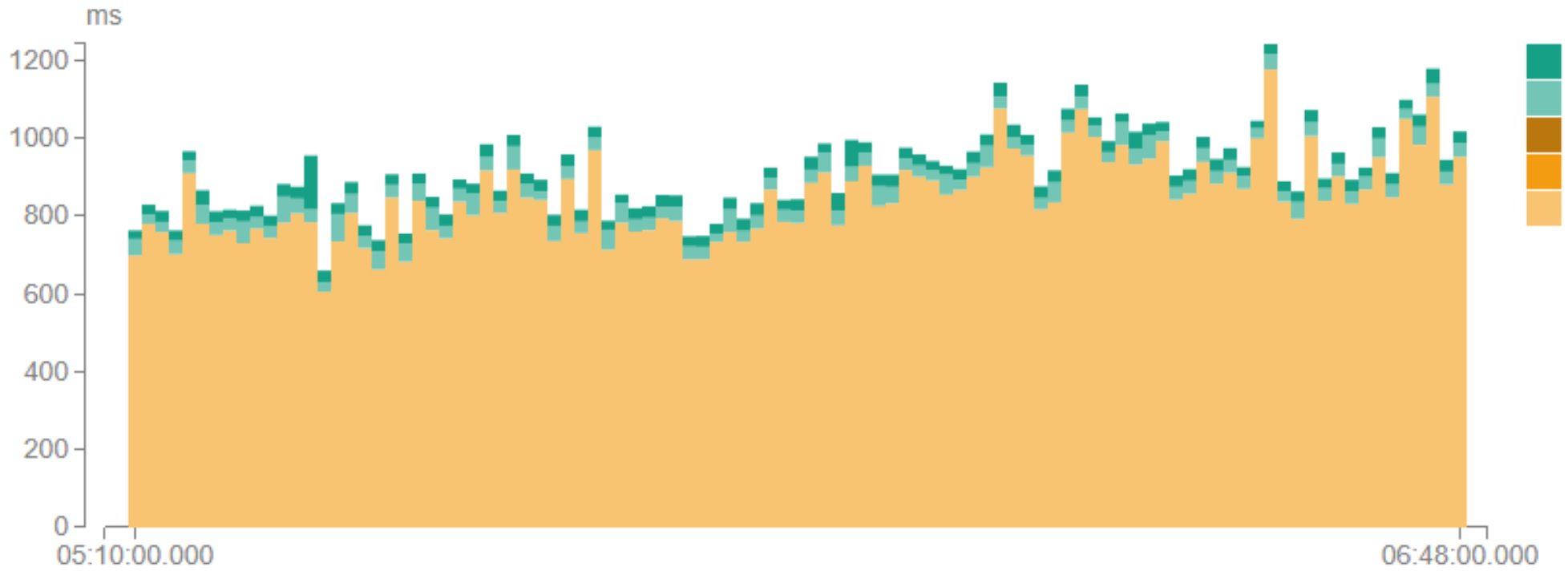
Name: all_tweets
Id: 3c2e5d75-fa14-47d4-b852-559d5196b5c4
RunId: bc0c7dd7-da65-4852-b78c-bc6acab15d1b

	Timelines	Histograms
Input Rate (?)		
Process Rate (?)		
Input Rows (?)		
Batch Duration (?)		

ELKH Cloud

ELKH Cloud

Operation Duration (?)



Összefoglaló

- A Spark egy elosztott futásidejű környezet (keretrendszer)
- Az adatfeldolgozó kódokat párhuzamosan és adatoptimalizált módon hajta végre
- Ha a kód Spark metódusokat használ (azaz az RDD-kbe tölti be az adatokat, RDD tranformációkat és akciókat használ), a Spark elrejti az elosztott feldolgozás technikai részleteit
- A Spark automatikusan optimalizálja a párhuzamos kódfuttatást (felosztja a kódrészleteket a különböző végrehajtók között, küldi / fogadja az adatokat, gyorsítótárat, stb.)
- Hiba álló működést nyújt
- A nem Spark struktúrákat használó alkalmazásokat nem mindig triviális átalakítani

Összefoglaló

- A Spark számos adatforrást, különféle adatforrástípusokat (fájl, HDFS, S3, adatbázisok, stb.) És formátumokat (bináris, szöveges, csv, json,…) támogat.
- Számos programozási paradigma érhető el (map-reduce, pipeline, kulcsérték tárolás)
- És magas szintű függvények (groupByKey, aggregate, sql lekérdezések, …)

Köszönöm a figyelmet!