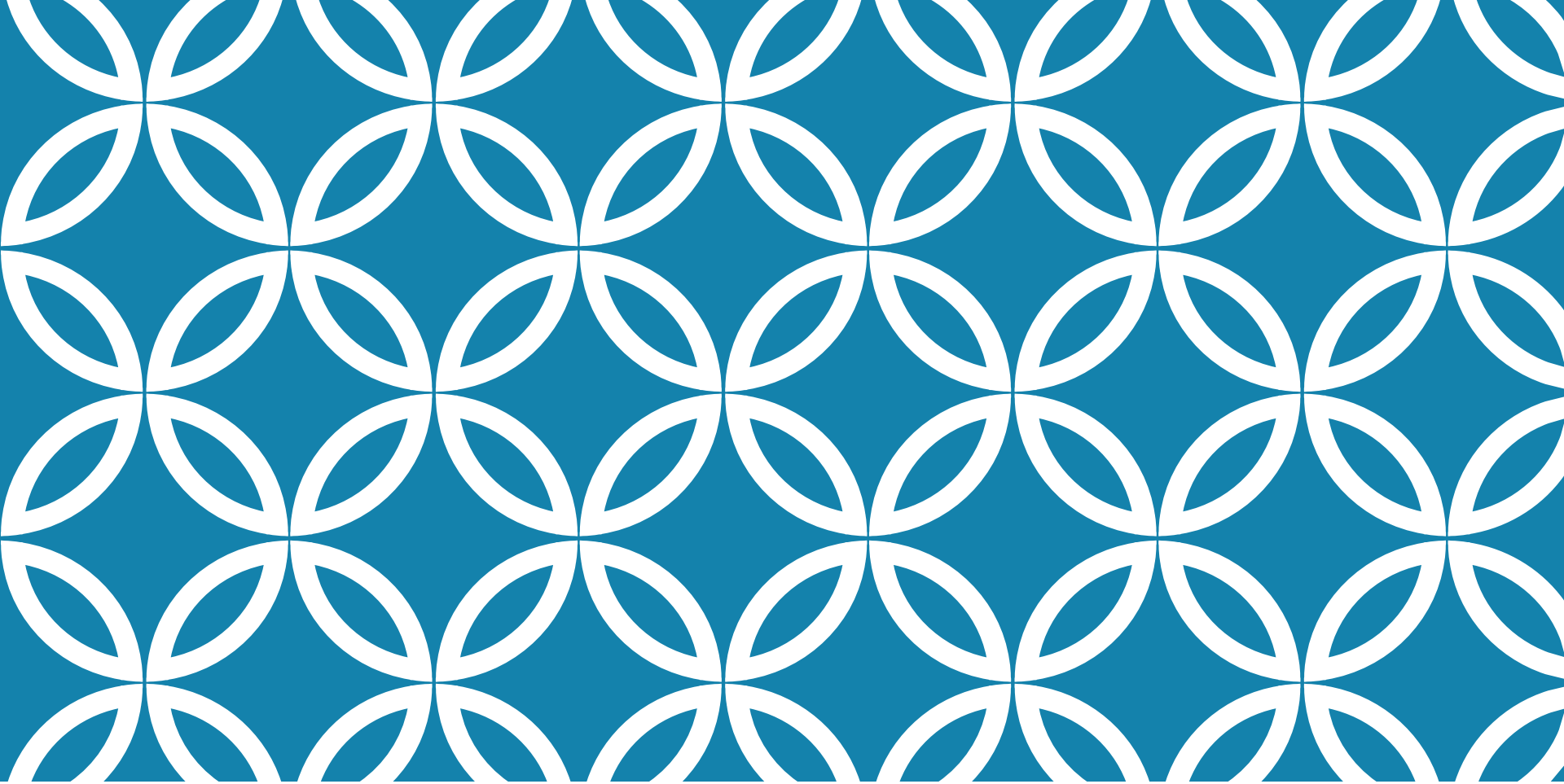


KORSZERŰ BIG DATA FELDOLGOZÓ KERETRENDSZEREK

2014.02.03.

Hermann Gábor
MTA-SZTAKI

MI AZ A BIG DATA?

MI AZ A BIG DATA?

Sok adat!

ENNYI?

BIG DATA

4V

- Volume
- Velocity
- Variety
- Veracity

+3V (7V)

- Variability
- Visualisation
- Value

BIG DATA?

Google keresője által számon tartott oldalak

- ~4.5 milliárd

Facebook gráf

- ~890 millió naponta aktív felhasználnó

YouTube-ra feltöltött videó

- Percenként 20 órányi

Twitter

- Napi 50 millió tweet

BIG DATA?

Honnan jön?

- Alkalmazottak szedték össze
- Emberek generálnak
- Gépek generálnak

Kihívás: feldolgozni

- Régebben
  - Külön helyen tárolás és feldolgozás
- Most
  - Vigyük a processzort az adathoz!

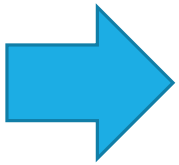
PROBLÉMA: WORDCOUNT

Mekkora a szavak előfordulása egy szövegben?

Motiváció: keresőmotor

Például:

```
This is John.
John likes watching TV.
John is always happy.
This is what he does.
```



```
(This, 1)
(is, 3)
(John, 3)
(likes, 1)
(watching, 1)
(TV, 1)
(what, 1)
(he, 1)
(does, 1)
```

TWEETEK

Szeretnénk megtudni, hogy mik a népszerű szavak!

Napi ~50 millió tweet

Egy tweet 140 karakter = 140 x 4 byte = 560 byte

Napi 50 x 560 MB = 28 GB

Kb. 34% angol

- Az is 9 GB naponta!

MEGOLDÁSOK

Egy gép

- „Klasszikus” programozás
- ~4 GB memória
- Nem fér el memóriában, kisebb adatra jó
- Pl. spanyol tweetek napi elemzése: ~3,3 GB

MEGOLDÁSOK

Egy gép

- „Klasszikus” programozás
- ~4 GB memória
- Nem fér el memóriában, kisebb adatra jó
- Pl. spanyol tweetek napi elemzése: ~3,3 GB

Nagyobb gép/szuperszámítógép

- Elfér a memóriában
- Probléma: párhuzamos algoritmus kell

Több kisebb gép

- Elfér a memóriában, de szét kell osztani
- Probléma: nem csak párhuzamos, de osztott algoritmus kell

MEGÉRI?

64 GB memória
32 mag

4 GB memória
2 mag
250$
16 db

MEGÉRI?

64 GB memória
32 mag
7000$

4 GB memória
2 mag
250$
16 db
4000$

MEGÉRI!

DE...

Adatot darabolni kell

Hálózaton kell küldözgetni

Az egészet számon kell tartani!

Sok hibalehetőség

Hardveres meghibásodás?

SEGÍTSÉG: OSZTOTT KERETRENDSZER

Adatot darabolni kell
- megoldja

Hálózaton kell küldözgetni
- megoldja

Az egészet számon kell tartani!
- megoldja

Sok hibalehetőség
- kevesebb!

Harveres meghibásodás
- hibatűrés

Csak az osztott algoritmus megalkotásával kell foglalkozni

„FEJLŐDÉS”

1. Egyszerű szekvenciális algoritmus
2. Párhuzamosítás
3. Elosztás „kézzel”
4. Apache Hadoop keretrendszer
   - Robosztus, „régi”
5. Apache Spark, Apache Flink keretrendszerek
   - Viszonylag újak, gyorsabbak

EGY GÉP – SZEKVENCIÁLISAN

„Klasszikus” programozás

Hogyan számoljunk szavakat?

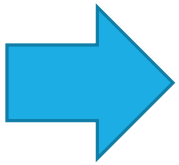
PROBLÉMA: WORDCOUNT

Mekkora a szavak előfordulása egy szövegben?

Motiváció: keresőmotor

Például:

```
This is John.
John likes watching TV.
John is always happy.
This is what he does.
```



```
(This, 1)
(is, 3)
(John, 3)
(likes, 1)
(watching, 1)
(TV, 1)
(what, 1)
(he, 1)
(does, 1)
```

EGY GÉP – SZEKVENCIÁLISAN

„Klasszikus” programozás

Hogyan számoljunk szavakat?

1. Szedjük szét szavakra!
2. Tároljunk kulcs-érték párokban (szó, előfordulás száma)!
3. Végeredményt irassuk ki!

EGY GÉP – SZEKVENCIÁLISAN

„Klasszikus” programozás

Hogyan számoljunk szavakat?

1. Szedjük szét szavakra!
2. Tároljunk kulcs-érték párokban (szó, előfordulás száma)!
3. Végeredményt irassuk ki!

EGY GÉP – SZEKVENCIÁLISAN

1. This is a text. A text for you
2. this, is, a, text, a, text, for, you
3. (this, 1), (is, 1), (a, 1), (text, 1), (a, 1), (text, 1), (for, 1), (you, 1)
4. (this, 1)
   (is, 1)
   (a, 2)
   (text, 2)
   (for, 1)
   (you, 1)

WORDCOUNT: JAVA KÓD

```
Map<String, Integer> map =
                new HashMap<String, Integer>();

for (String line : lines) {
        String[] words = line.split(" ");
        for (String word : words) {
                int count = 1;

                if (map.containsKey(word)) {
                        count += map.get(word);
                }

                map.put(word, count);
        }
}
```

VAN ÉRTELME?

Ha „kisebb” adat, ez a legjobb

Fölösleges bonyolítani

Ha nagyobb vagy lassú...?

PÁRHUZAMOSÍTSUNK

GPU programozás (OpenCL)

Mi van, ha több adatom lesz? (Több felhasználó?)

Mi van, ha nem fér el?

PÁRHUZAMOSÍTSUNK

GPU programozás (OpenCL)

Mi van, ha több adatom lesz? (Több felhasználó?)

Mi van, ha nem fér el?

Veszek nagyobb gépet!

PÁRHUZAMOSÍTSUNK

GPU programozás (OpenCL)

Mi van, ha több adatom lesz? (Több felhasználó?)

Mi van, ha nem fér el?

Veszek nagyobb gépet!

Drága

OSSZUK MEG!

Adatot darabolni kell

Hálózaton kell küldözgetni

Az egészet számon kell tartani!

Van, hogy megéri (triviálisan elosztható feladat)

De általában fölösleges minddel foglalkozni

Mit csinál egy osztott keretrendszer?

APACHE HADOOP

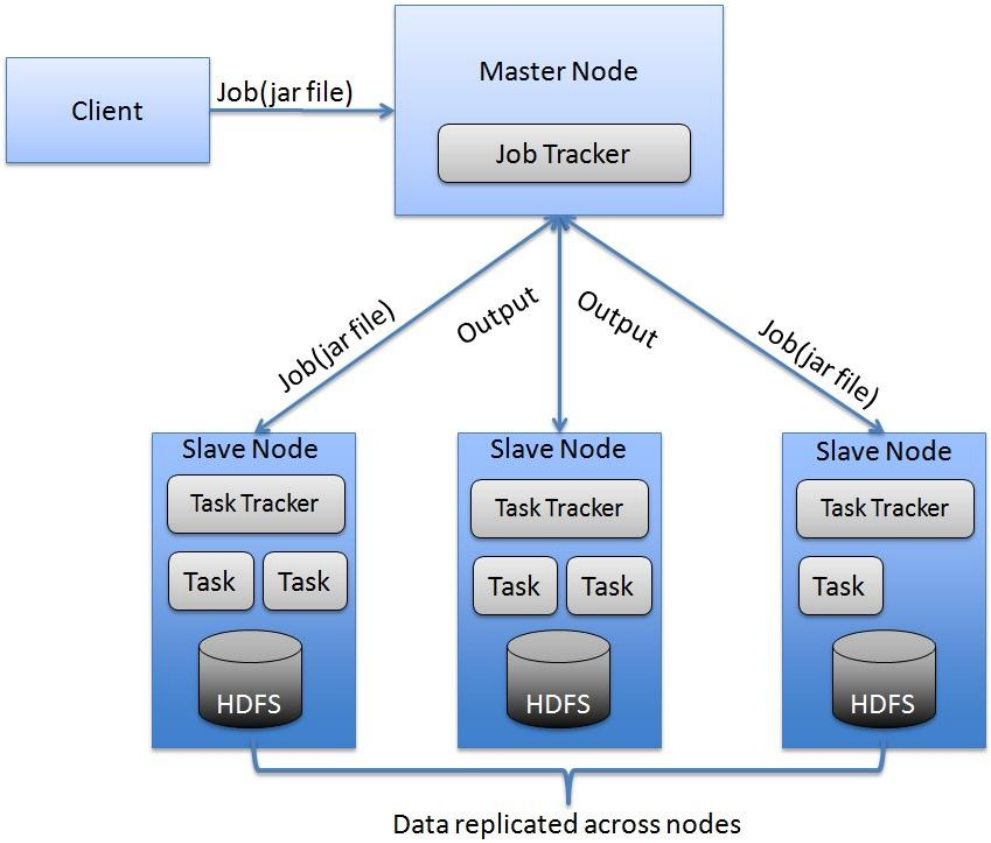
Osztott keretrendszer

Google MapReduce cikk (2004)

Nyílt forráskódú

Tíz éves projekt

HADOOP ARCHITEKTÚRA



MAPREDUCE MODELL

„Map” fázis

- Minden worker kap egy adathalmazt és átalakítja egy másik adathalmazzá
- Kulcs-érték párok

„Shuffle” fázis

- Varázslat!
- Az azonos kulcsúak egy helyre kerülnek

„Reduce” fázis

- Az azonos kulcsúakat egyesíti

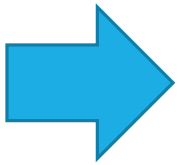
EMLÉKEZTETŐ: WORDCOUNT

Mekkora a szavak előfordulása egy szövegben?

Motiváció: keresőmotor

Például:

```
This is John.
John likes watching TV.
John is always happy.
This is what he does.
```

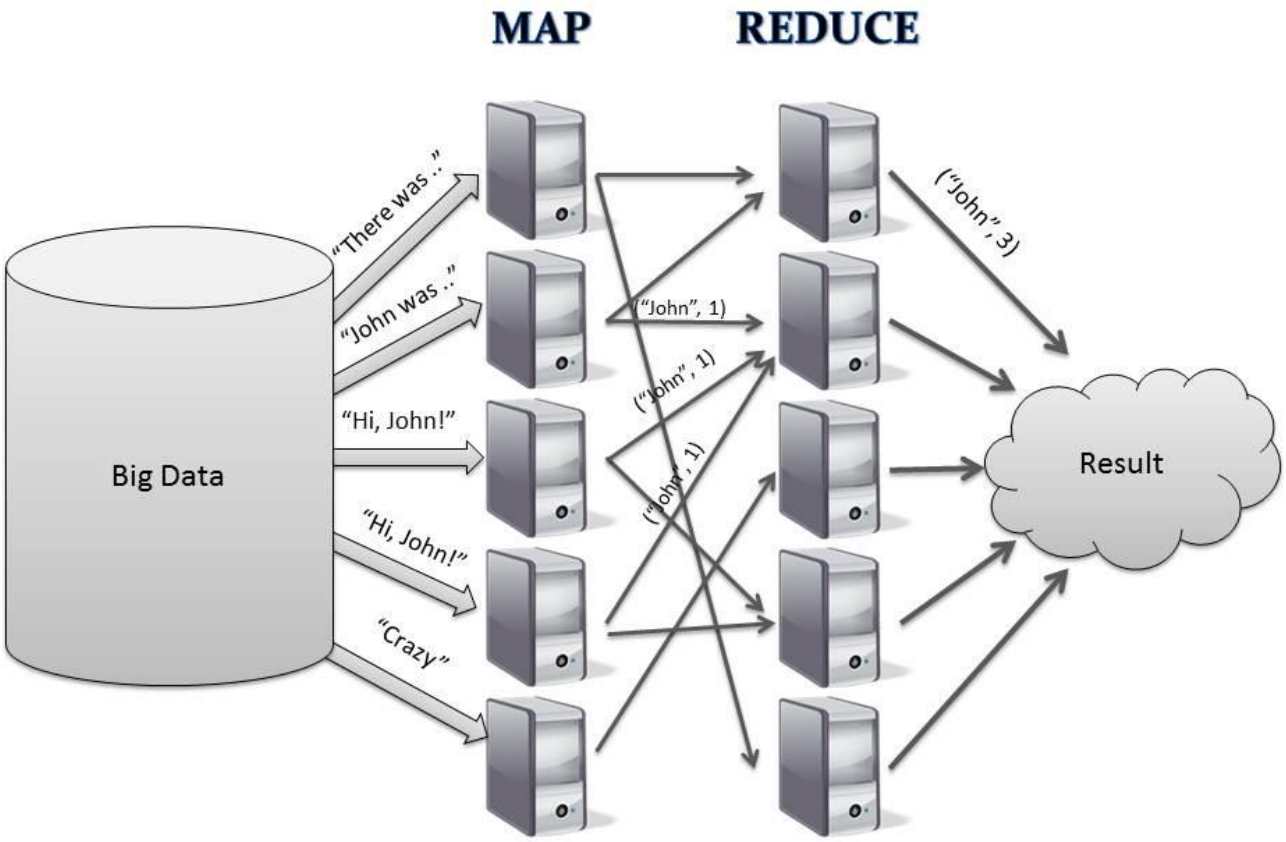


```
(This, 1)
(is, 3)
(John, 3)
(likes, 1)
(watching, 1)
(TV, 1)
(what, 1)
(he, 1)
(does, 1)
```

EMLÉKEZTETŐ: WORDCOUNT

1. This is a text. A text for you
2. this, is, a, text, a, text, for, you
3. (this, 1), (is, 1), (a, 1), (text, 1), (a, 1), (text, 1), (for, 1), (you, 1)
4. (this, 1)
   (is, 1)
   (a, 2)
   (text, 2)
   (for, 1)
   (you, 1)

MAPREDUCE


MAP
REDUCE
Big Data
"There was .."
"John was .."
"Hi, John!"
"Hi, John!"
"Crazy"
("John", 1)
("John", 1)
("John", 1)
("John", 3)
Result

HOGYAN ÍRJUNK HADOOP JOBOT?

1. Írjuk meg a MapReduce programot
2. Fordítsuk le
3. Teszteljük lokális környezetben
4. Deploy-oljuk clusterre
5. Futtassuk klaszteren

MIT NYERTÜNK?

Megbízhatóság szoftversen

- Szoftveres redundancia
- Olcsóbb
- Lehet használni régi hardvert is!

MIT NYERTÜNK?

Megbízhatóság szoftversen

- Szoftveres redundancia
- Olcsóbb
- Lehet használni régi hardvert is!

Elosztott

- Párhuzamos, így gyorsabb
- Skálázható, szinte lineárisan

MIT NYERTÜNK?

Megbízhatóság szoftversen

- Szoftveres redundancia
- Olcsóbb
- Lehet használni régi hardvert is!

Elosztott

- Párhuzamos, így gyorsabb
- Skálázható, szinte lineárisan

Erőteljes

- Sok adatot, gyorsan (párhuzamosan)

Könnyebben programozható

DE…

Map-fázis, Reduce-fázis egymás után jöhet csak

MapReduce fázisok között lemezre írás/olvasás

- Pl. gráf-algoritmus esetén a gráf szerkezetét is, ami nem változik!
- Iteráció lassú!

Konfiguráció?

Könnyen programozható?

KÖNNYEN PROGRAMOZHATÓ? (WORDCOUNT)

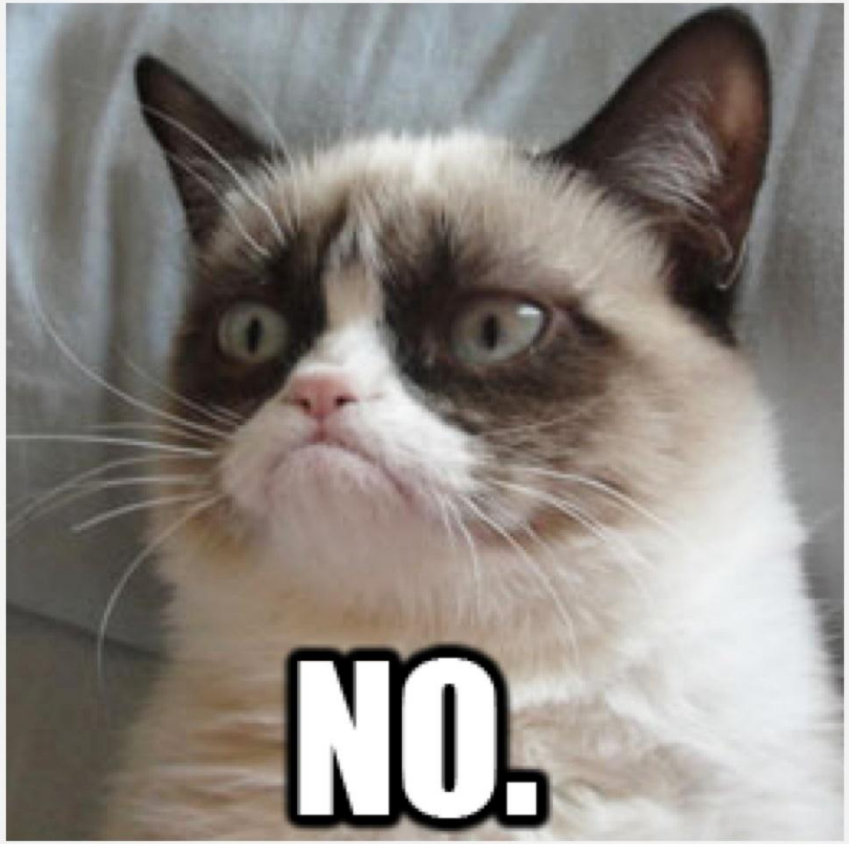
```
public static class TokenizerMapper
     extends Mapper<Object, Text, Text, IntWritable>{

  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map(Object key, Text value, Context context
                  ) throws IOException, InterruptedException {
    StringTokenizer itr = new StringTokenizer(value.toString());
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      context.write(word, one);
    }
  }
}
```

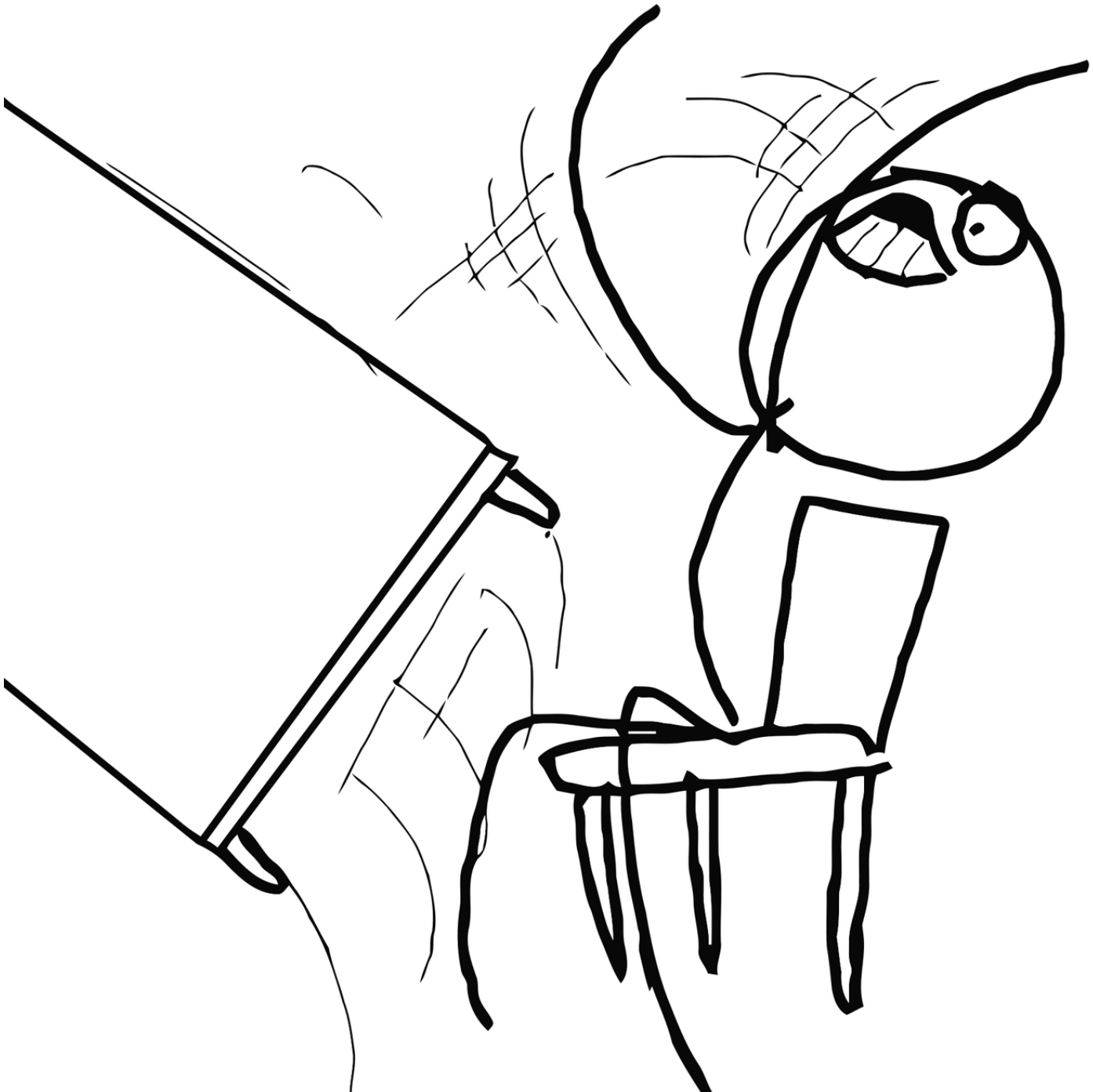
KÖNNYEN PROGRAMOZHATÓ? (WORDCOUNT)

```
public static class IntSumReducer
     extends Reducer<Text,IntWritable,Text,IntWritable> {
  private IntWritable result = new IntWritable();

  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
  }
}
```

KÖNNYEN PROGRAMOZHATÓ? (WORDCOUNT)

```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```


NO.



SZEBBEN NÉZNE KI...

```
val env = ExecutionEnvironment.getExecutionEnvironment

val text = env.readTextFile(filePath)

val counts = text.flatMap { _.toLowerCase.split("\\W+") }
  .map { (_, 1) }
  .groupBy(0)
  .sum(1)

counts.print()

env.execute("WordCount Example")
```

ÚJABB RENDSZEREK?

ÚJABB RENDSZEREK

Apache Flink

APACHE SPARK VS. APACHE FLINK

Flink

- „Kutatósabb” (kutatási projektnek indult, TU Berlin)
- „Európai”

Spark

- Amerikai (UC Berkley)

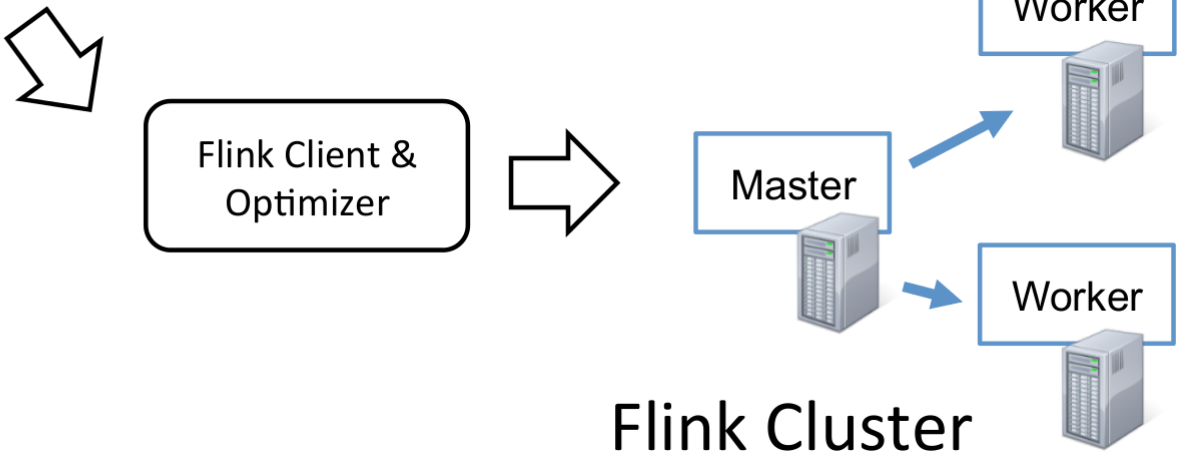
Mindkettő

- Nyílt forráskódú (Apache projekt)
- Hasonló programozási absztrakció
- Jelenleg mindkettő nagy ipari támogatásra hajt
- 5 év múlva nem lesz különbség (vagy kihal az egyik)

APACHE FLINK HASZNÁLATA

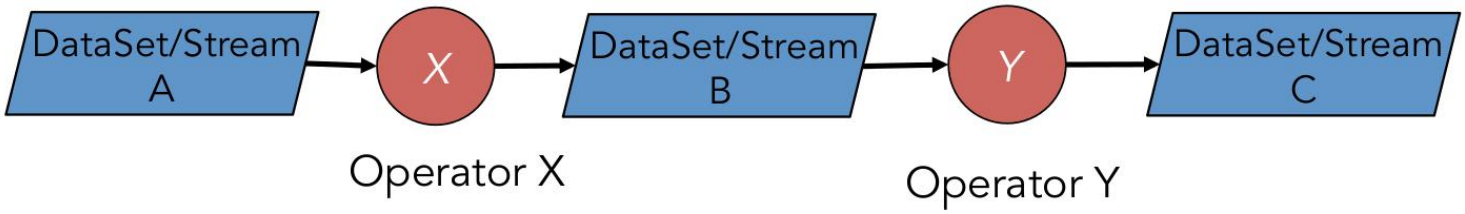
Analytical Program

```
DataSet<String> text = env.readTextFile(input);

DataSet<Tuple2<String, Integer>> result = text
         .flatMap((str, out) -> {
              for (String token : value.split("\\W")) {
                   out.collect(new Tuple2<>(token, 1));
              })
         .groupBy(0)
         .aggregate(SUM, 1);
```

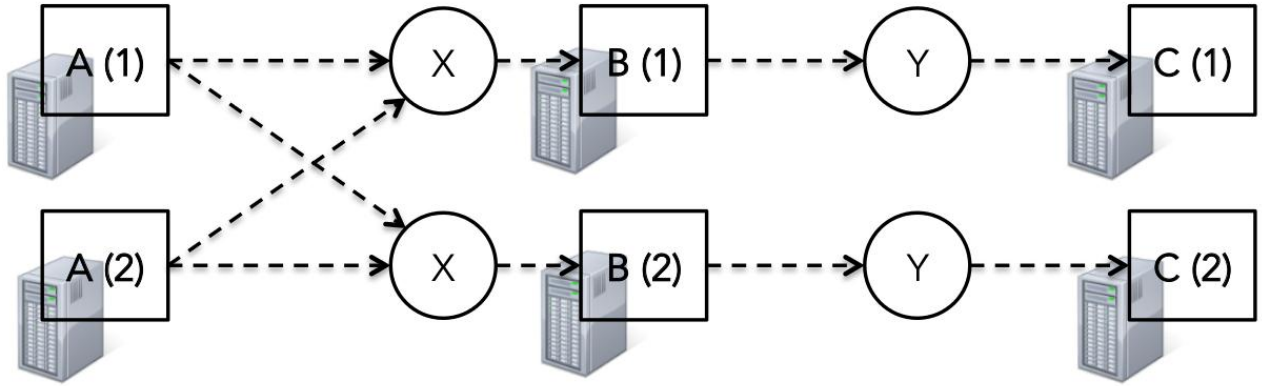


DATASET ABSZTRAKCIÓ

Program

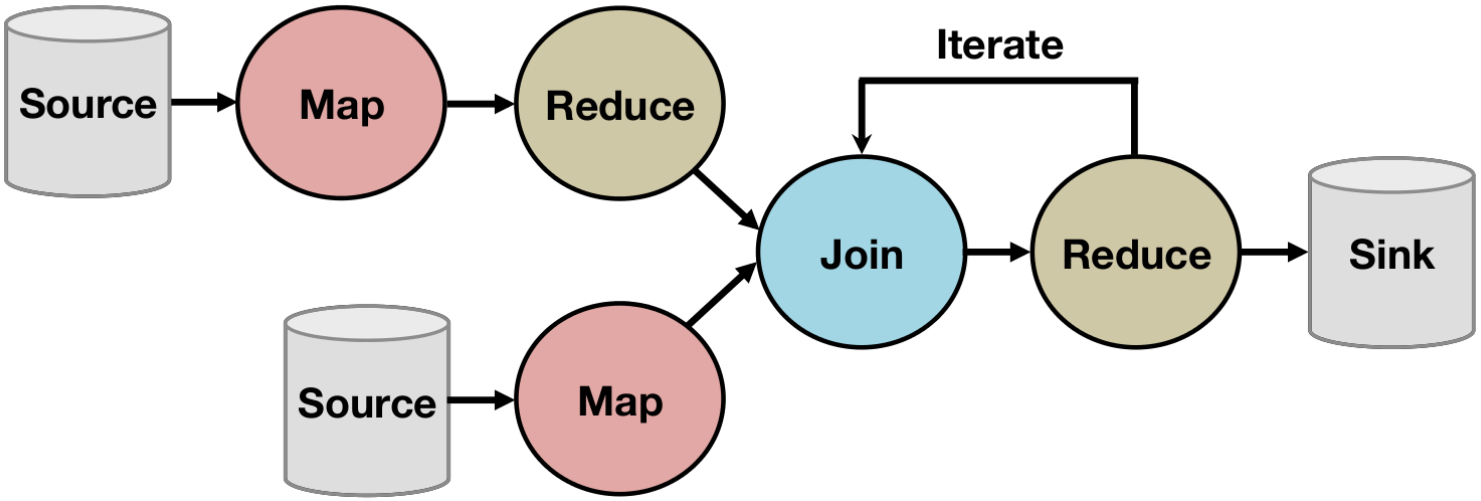


Parallel Execution



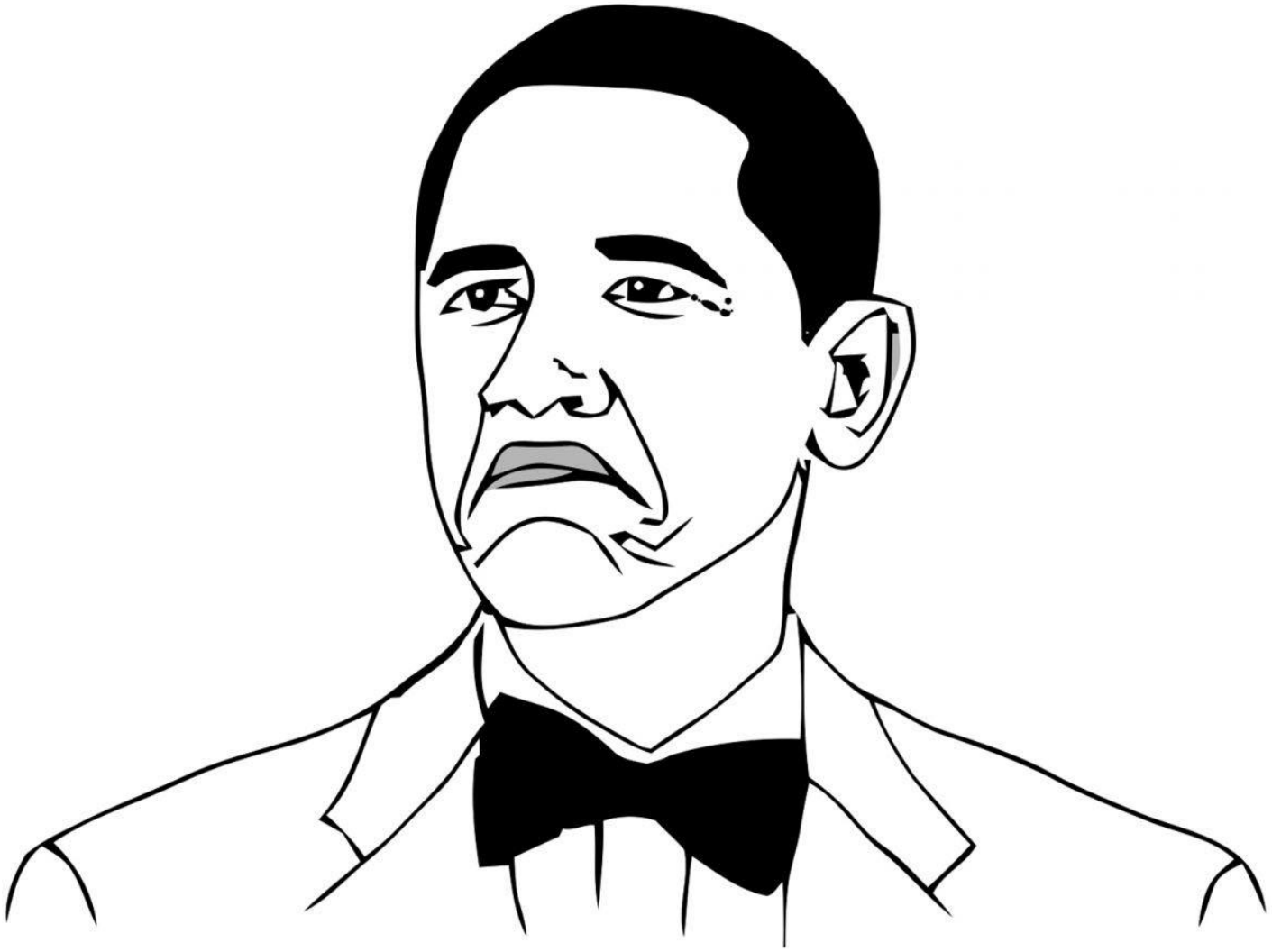
OPERÁTOROK

Map, FlatMap, MapPartition, Filter, Project, Reduce, ReduceGroup, Aggregate, Distinct, Join, CoGroup, Cross, Iterate, Iterate Delta, Iterate-Vertex-Centric, Windowing



WORDCOUNT

```
ExecutionEnvironment env =
ExecutionEnvironment.getExecutionEnvironment();

DataSet<String> text = env.readTextFile(input);

DataSet<Tuple2<String, Integer>> result = text
              .flatMap((str, out) -> {
                    for (String token : value.split("\\W")) {
                          out.collect(new Tuple2<>(token, 1));
                    })
              .groupBy(0)
              .sum(1);

counts.print();
env.execute("WordCount Example");
```



NOT BAD

WORDCOUNT

```
ExecutionEnvironment env =
ExecutionEnvironment.getExecutionEnvironment();

DataSet<String> text = env.readTextFile(input);

DataSet<Tuple2<String, Integer>> result = text
              .flatMap((str, out) -> {
                    for (String token : value.split("\\W")) {
                          out.collect(new Tuple2<>(token, 1));
                    })
              .groupBy(0)
              .sum(1);

counts.print();
env.execute("WordCount Example");
```

RUN EVERYWHERE!

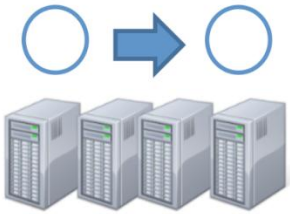
```
DataSet<String> text = env.readTextFile(input);

DataSet<Tuple2<String, Integer>> result = text
          .flatMap((str, out) -> {
               for (String token : value.split("\\W")) {
                    out.collect(new Tuple2<>(token, 1));
               }})
          .groupBy(0)
          .aggregate(SUM, 1);
```

Local
Debugging

As Java Collection
Programs

Java

Embedded
(e.g., Web Container)



Cluster (Batch)

Fink Runtime or Apache Tez

Cluster (Streaming)

ADATOK SZERIALIZÁLÁSA

Apache Hadoop

- Mindig szerializál
- Robosztus, nem hagy cserben
- Lassú

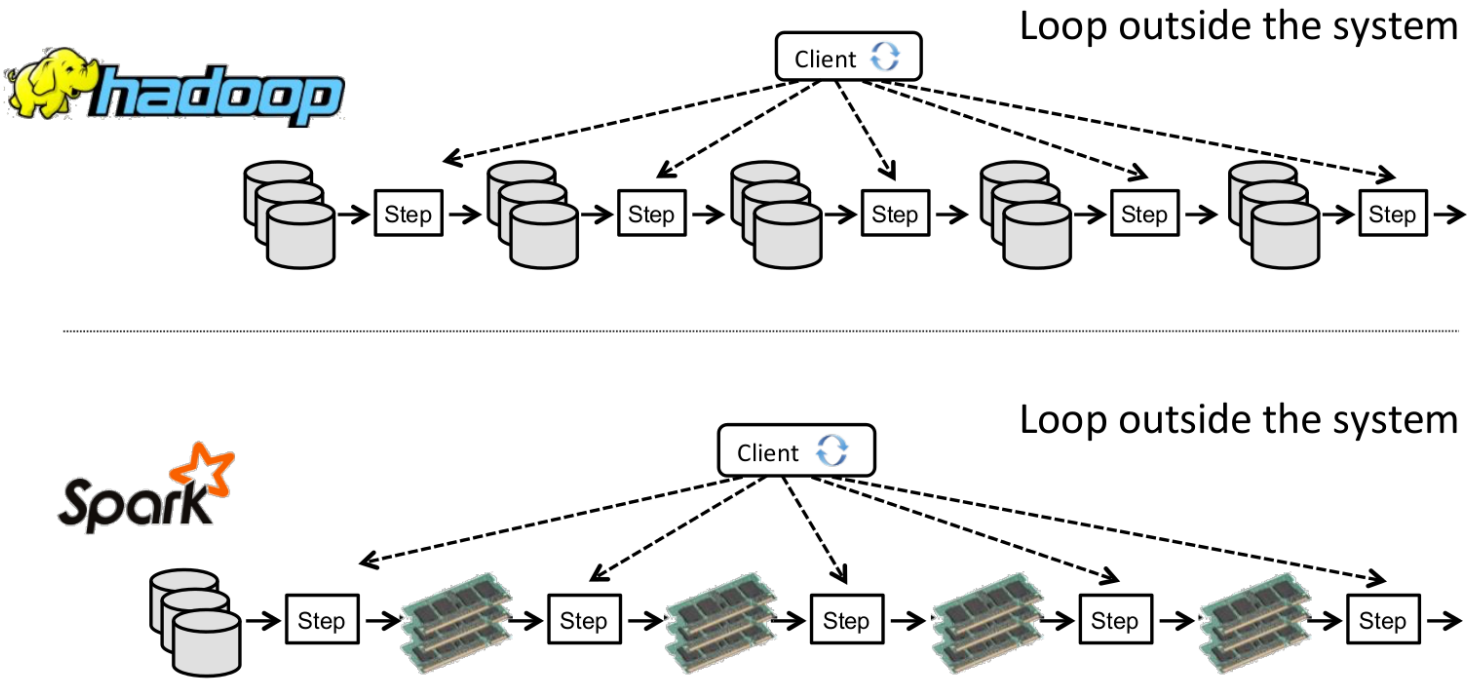
Apache Spark

- Objektumokon dolgozik
- Szerializáció van, de lassúnak tartott, ezért csak akkor ha szükséges

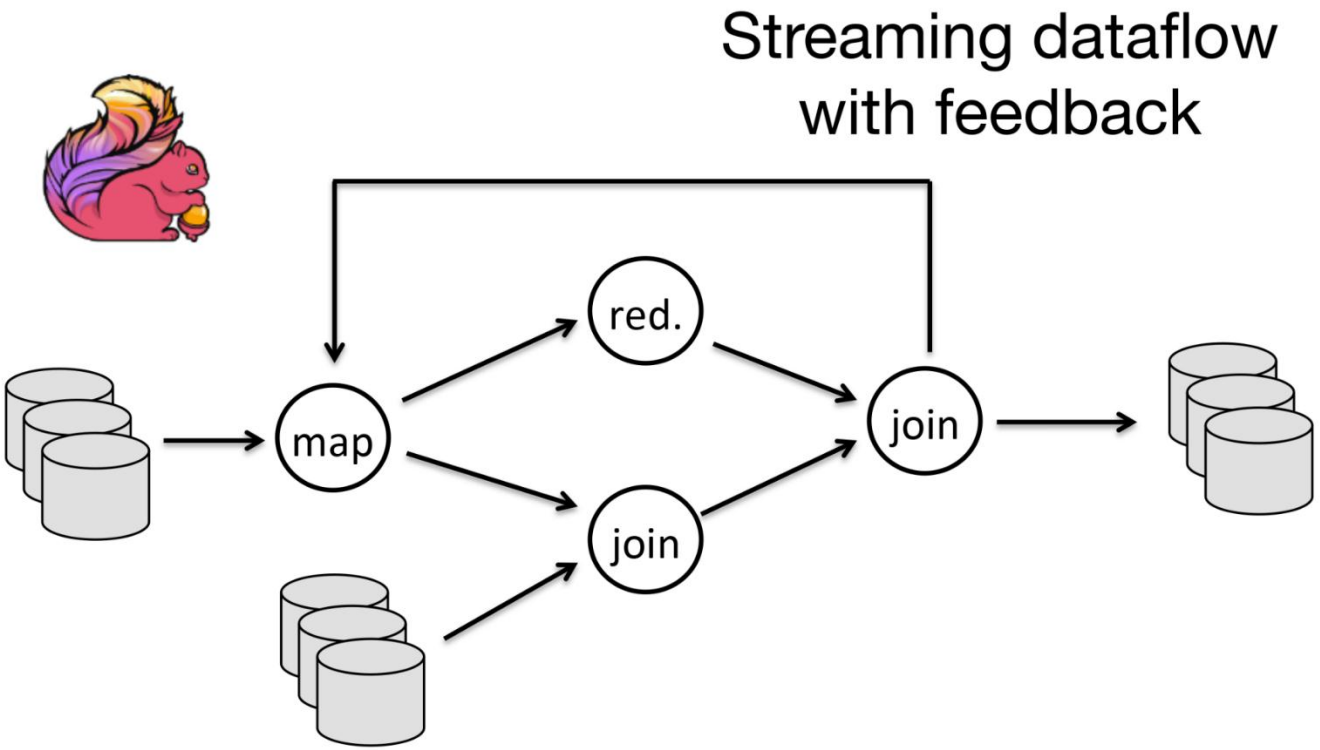
Apache Flink

- Olcsóvá teszi a szerializációt
- Részben szerializált adattal dolgozik
- Hatékony és robosztus!

ITERÁCIÓK


hadoop
Loop outside the system
Client
Step
Step
Step
Step
Step
Spark
Loop outside the system
Client
Step
Step
Step
Step
Step

APACHE FLINK ITERÁCIÓ

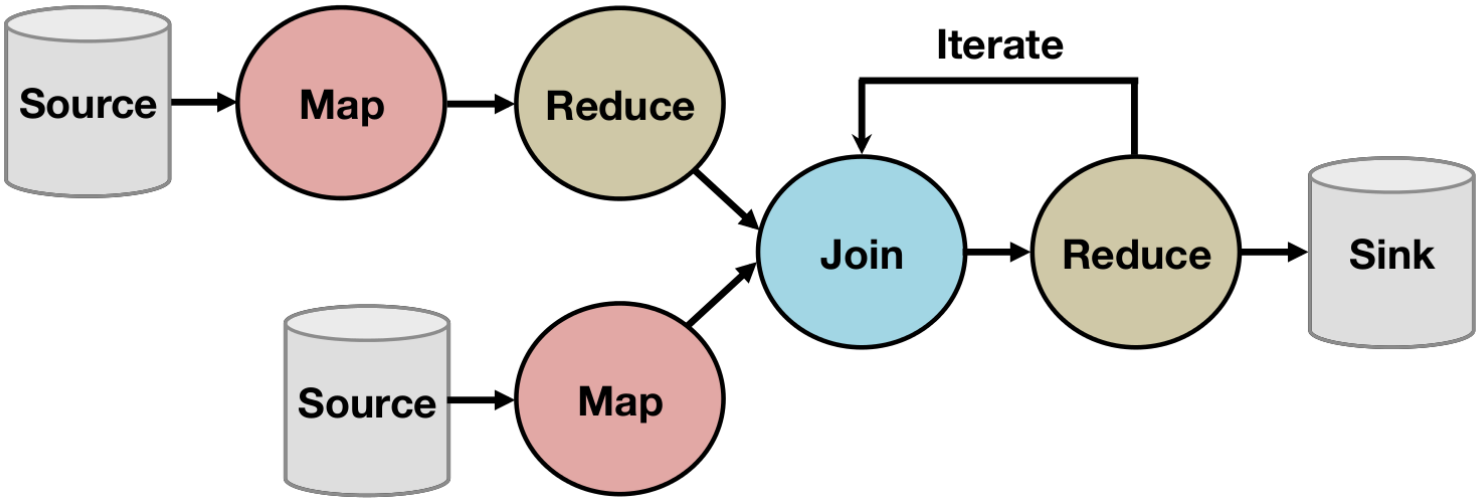


System is iteration-aware, performs automatic optimization

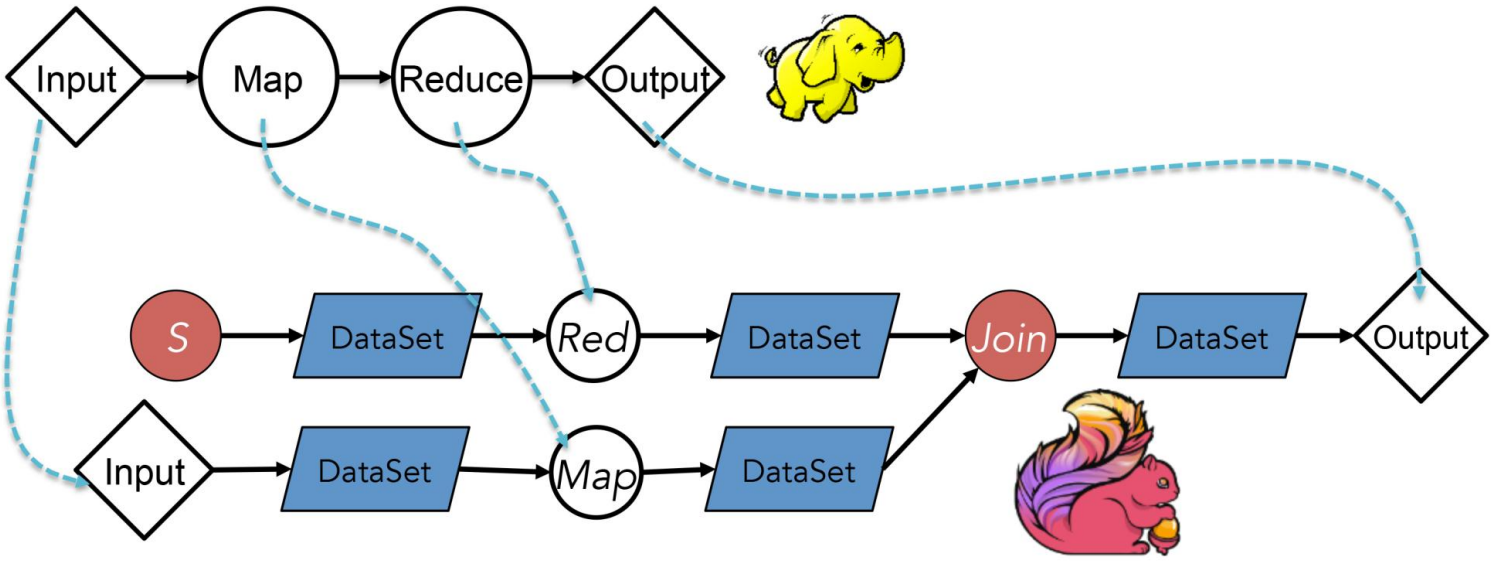
HADOOP KÓD ÁTÜLTETÉSE?

OPERÁTOROK (EMLÉKEZTETŐ)

Map, FlatMap, MapPartition, Filter, Project, Reduce, ReduceGroup, Aggregate, Distinct, Join, CoGroup, Cross, Iterate, Iterate Delta, Iterate-Vertex-Centric, Windowing



HADOOP KÓD ÁTÜLTETÉSE



OPTIMIZER

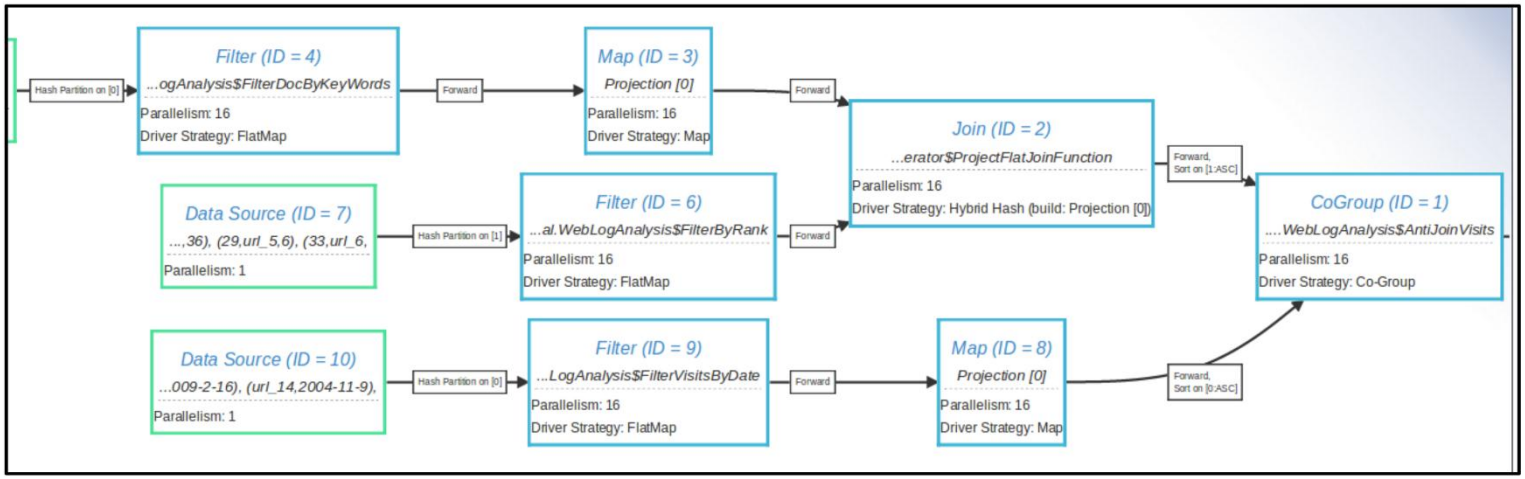
Varázslat!

De nem kell félni ☺

Adatbázislekérdezések optimalizálva

API programok hasonlóan

PLAN VISUALIZER



STREAM-FELDOLGOZÁS

Adatfolyam

Valós idejű

- Latency kritikus feladatok

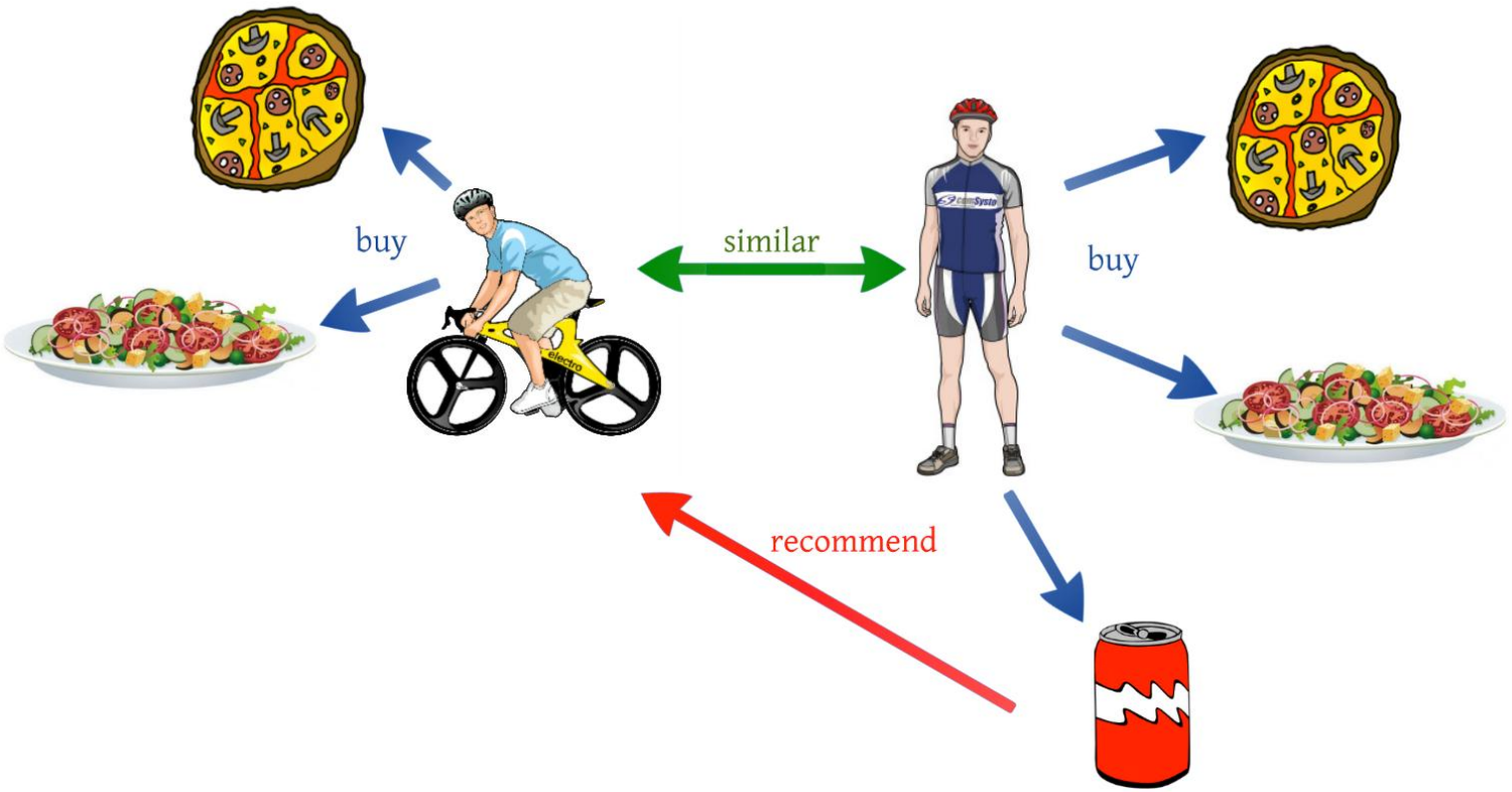
Sok adat

- Magas throughput

Latency – throughput trade-off

PÉLDA: AJÁNLÓRENDSZER

AJÁNLÓRENDSZER



AJÁNLÓRENDSZER

Nem fognak várni perceket

- Alacsony latency

Sok felhasználót kell egyszerre kiszolgálni

- Magas throughput

Sok adatot kell kombinálni az eredményhez

- Magas throughput

STREAMING ELVÁRÁSOK

Tipikusan „végtelen ideig” fusson

Hibatűrés (fault tolerance)

Batch feladat logikájának átültetése

Natív ablakozás

Könnyű csatlakozás

RENDSZEREK

samza

Spark

STORM

APACHE STORM

Recordonkénti feldolgozás

- Alacsony latency

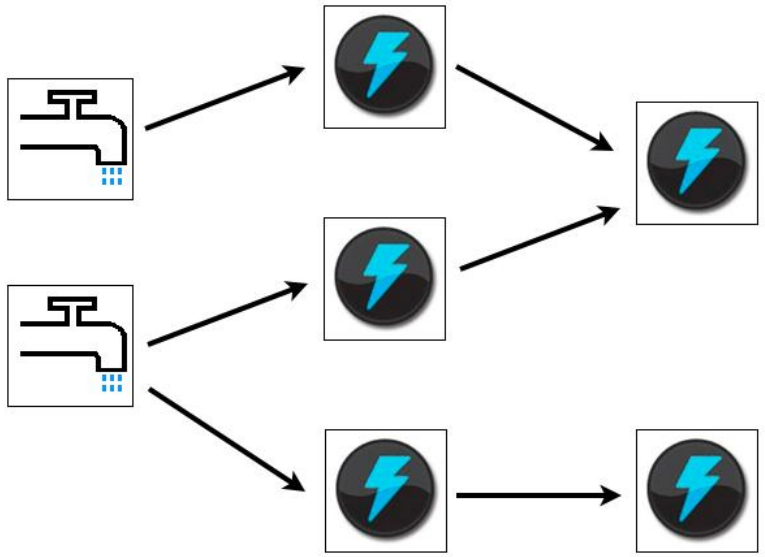
Hibatűrés

- Replikáció a forrásban
- Probabilisztikus ellenőrzés
- At least once hibatűrés
- Lassabb visszajátszás
- Állapot replikáció nincs

Alacsonyabb szintű API

Trident

- Storm fölötti absztrakció



APACHE SAMZA

samza

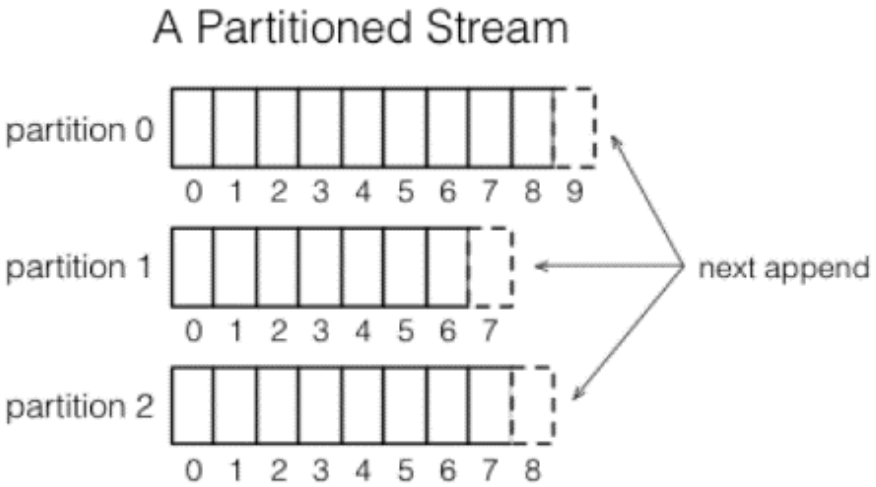
LinkedIn-ben projekt

Kafka (message queue) fölé épül

Hibatűrés könnyűvé téve

Sorrendiség

- Visszajátszhatóság



APACHE SPARK STREAMING

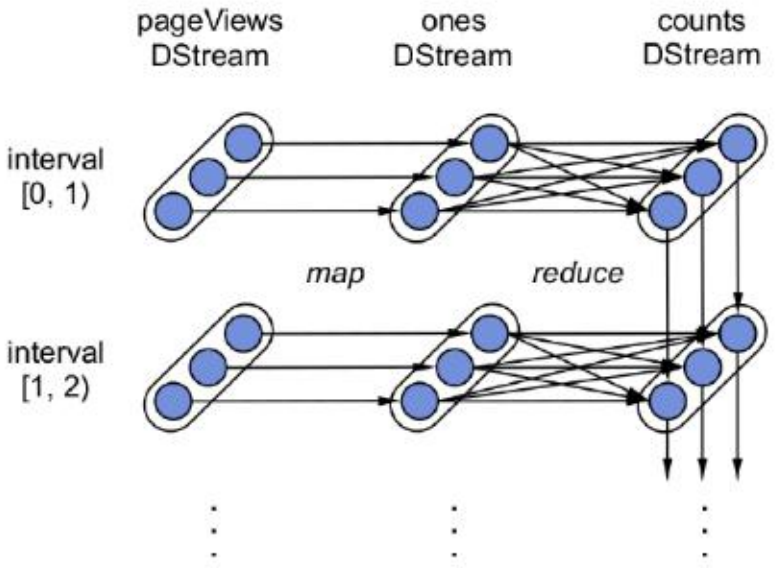
Mini-batch

- Magas throughput, viszonylag alacsony latency
- (Csalás! :)

Nem módosítható állapot

- Immutable state
- Biztonságosabb

Exactly-once hibatűrés



APACHE FLINK STREAMING

Állítható pufferméret

- Trade-off szabályozása

Flexibilis ablakozás

Könnyű kódátültetés

Könnyű összeköttetés

- Tetszőleges input/output
- Előre megírt: objektumlista, fájl, HDFS, Kafka, RabbitMQ

FLINK WORDCOUNT

```
ExecutionEnvironment env =
ExecutionEnvironment.getExecutionEnvironment();

DataSet<String> text = env.readTextFile(input);

DataSet<Tuple2<String, Integer>> result = text
            .flatMap((str, out) -> {
                  for (String token : value.split("\\W")) {
                        out.collect(new Tuple2<>(token, 1));
                  })
            .groupBy(0)
            .sum(1);

counts.print();
env.execute("WordCount Example");
```

FLINK STREAMING WORDCOUNT

```
StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

DataStream<String> text = env.readTextFile(input);

DataStream<Tuple2<String, Integer>> result = text
            .flatMap((str, out) -> {
                  for (String token : value.split("\\W")) {
                        out.collect(new Tuple2<>(token, 1));
                  })
            .groupBy(0)
            .sum(1);

counts.print();
env.execute("WordCount Example");
```

APACHE FLINK STREAMING

Állítható pufferméret

- Trade-off szabályozása

Flexibilis ablakozás

Könnyű kódátültetés 

Könnyű összeköttetés

- Tetszőleges input/output
- Előre megírt: objektumlista, fájl, HDFS, Kafka, RabbitMQ

FLINK STREAMING WORDCOUNT WINDOW

```
StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

DataStream<String> text = env.readTextFile(input);
DataStream<Tuple2<String, Integer>> result = text
            .flatMap((str, out) -> {
                  for (String token : value.split("\\W")) {
                        out.collect(new Tuple2<>(token, 1));
                  })
            .window(Time.of(60, TimeUnit.SECONDS))
            .every(Time.of(5, TimeUnit.SECONDS))
            .groupBy(0)
            .sum(1);
counts.print();
env.execute("WordCount Example");
```

APACHE FLINK STREAMING

Állítható pufferméret

- Trade-off szabályozása

Flexibilis ablakozás 

Könnyű kódátültetés 

Könnyű összeköttetés

- Tetszőleges input/output
- Előre megírt: objektumlista, fájl, HDFS, Kafka, RabbitMQ

FLINK STREAMING WORDCOUNT

```
StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

DataStream<String> text = env.readTextFile(input);

DataStream<Tuple2<String, Integer>> result = text
            .flatMap((str, out) -> {
                  for (String token : value.split("\\W")) {
                        out.collect(new Tuple2<>(token, 1));
                  })
            .setBufferTimeout(100)
            .groupBy(0)
            .sum(1);

counts.print();
env.execute("WordCount Example");
```

APACHE FLINK STREAMING

Állítható pufferméret 

- Trade-off szabályozása

Flexibilis ablakozás 

Könnyű kódátültetés 

Könnyű összeköttetés

- Tetszőleges input/output
- Előre megírt: objektumlista, fájl, HDFS, Kafka, RabbitMQ

FLINK STREAMING WORDCOUNT

```
StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

DataStream<String> text = env.fromElements("This is a line.",
            "This is another row.");

DataStream<Tuple2<String, Integer>> result = text
            .flatMap((str, out) -> {
                  for (String token : value.split("\\W")) {
                        out.collect(new Tuple2<>(token, 1));
                  })
            .groupBy(0)
            .sum(1);

counts.print();
env.execute("WordCount Example");
```

FLINK STREAMING WORDCOUNT

```
StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

DataStream<String> text = env.socketTextStream("ilab.sztaki.hu",
                                                1294);

DataStream<Tuple2<String, Integer>> result = text
            .flatMap((str, out) -> {
                  for (String token : value.split("\\W")) {
                        out.collect(new Tuple2<>(token, 1));
                  })
            .groupBy(0)
            .sum(1);

counts.print();
env.execute("WordCount Example");
```

APACHE FLINK STREAMING

Állítható pufferméret 

- Trade-off szabályozása

Flexibilis ablakozás 

Könnyű kódátültetés 

Könnyű összeköttetés 

- Tetszőleges input/output
- Előre megírt: objektumlista, fájl, HDFS, Kafka, RabbitMQ

FLINK STREAMING HIBATŰRÉS

Megvalósítás alatt

Replikálás

- Osztottan memóriában
- Perzisztens tárolón (lemez)

Forrásból újraküldés

- At least once, de
- Bloom-filter: „látottak” szűrése (Google MillWheel)

Állapot (state) rögzítés

HOVA TART A FLINK?

Hibatűrés

Python API

SQL-szerű nyelv

Gépi tanulás Flink fölé (pl. Mahout DSL)

Gráf DSL

On-the-fly skálázás

... és még sok más

MIT HOZHAT A JÖVŐ ÁLTALÁBAN?

Elterjednek a korszerűbb rendszerek

Új alkalmazások jönnek (gépi tanulás)

Kompatibilisebb rendszerek (minden fusson mindenen!)

Javuló performancia (optimalizáció)

Biztonság!

... és

GAME ON!

Spark

samza

STORM

HOL ÉRTEK UTOL?

http://flink.apache.org/

http://hadoop.apache.org/

https://spark.apache.org/

Hermann Gábor
MTA-SZTAKI
ghermann@ilab.sztaki.hu