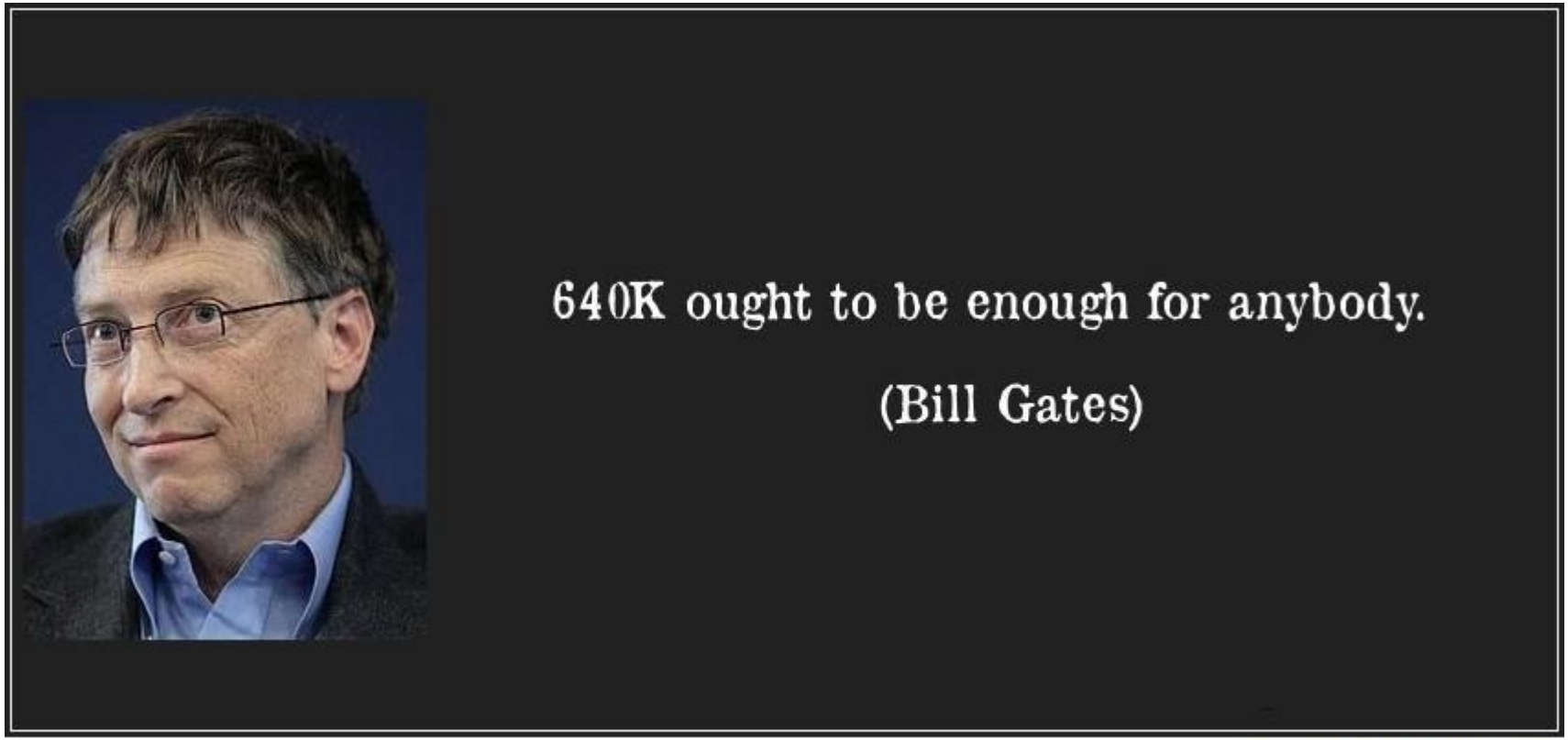
Big Data,
Distributed Storage & Computing

Gombos Gergő

640K Big Data?

1981



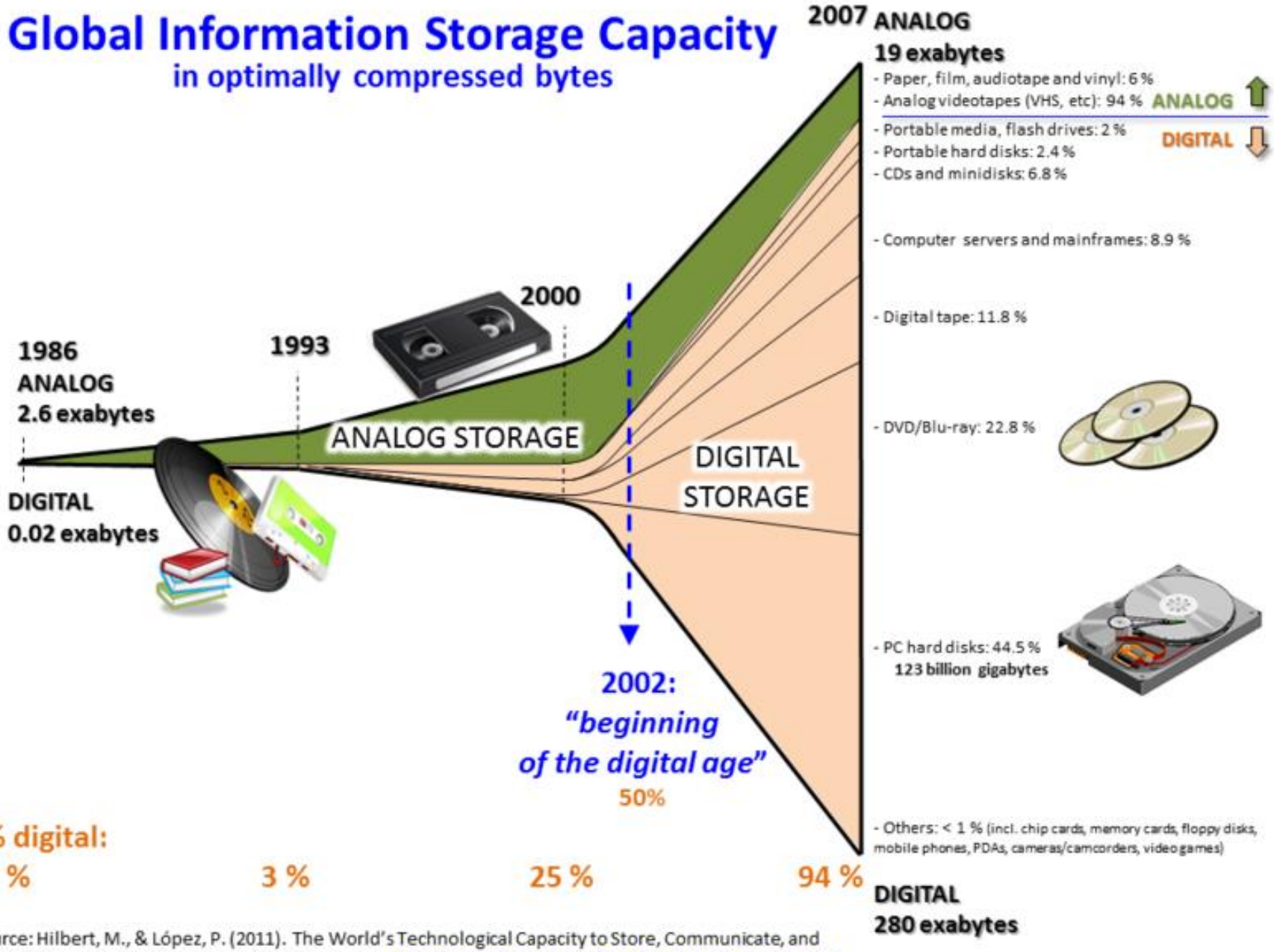
Big Data

- „Big data is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications.”

Wikipédia

Global Information Storage Capacity

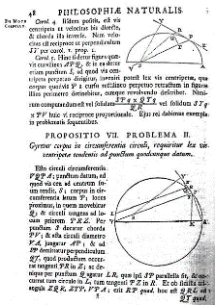
in optimally compressed bytes

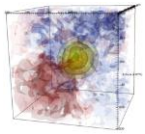


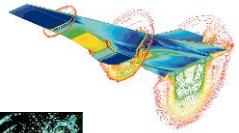
Source: Hilbert, M., & López, P. (2011). The World's Technological Capacity to Store, Communicate, and Compute Information. *Science*, 332(6025), 60 –65. http://www.martinhilbert.net/WorldInfoCapacity.html

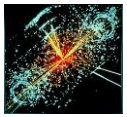
1000^n	10^n	Prefix	Symbol	Short scale	Long scale	Decimal equivalent in SI writing style
1000^8	10^{24}	**yotta**	Y	Septillion	Quadrillion	1 000 000 000 000 000 000 000 000
1000^7	10^{21}	zetta	Z	Sextillion	Trilliard (thousand trillion)	1 000 000 000 000 000 000 000
1000^6	10^{18}	exa	E	Quintillion	Trillion	1 000 000 000 000 000 000
1000^5	10^{15}	peta	P	Quadrillion	Billiard (thousand billion)	1 000 000 000 000 000
1000^4	10^{12}	tera	T	Trillion	Billion	1 000 000 000 000
1000^3	10^9	giga	G	Billion	Milliard (thousand million)	1 000 000 000
1000^2	10^6	mega	M	Million		1 000 000
1000^1	10^3	kilo	k	Thousand		1 000
$1000^{2/3}$	10^2	hecto	h	Hundred		100
$1000^{1/3}$	10^1	deca, deka	da	Ten		10
1000^0	10^0	*(none)*	*(none)*	One		1
$1000^{-1/3}$	10^{-1}	deci	d	Tenth		0.1
$1000^{-2/3}$	10^{-2}	centi	c	Hundredth		0.01
1000^{-1}	10^{-3}	milli	m	Thousandth		0.001
1000^{-2}	10^{-6}	micro	μ	Millionth		0.000 001
1000^{-3}	10^{-9}	nano	n	Billionth	Milliardth	0.000 000 001
1000^{-4}	10^{-12}	pico	p	Trillionth	Billionth	0.000 000 000 001
1000^{-5}	10^{-15}	femto	f	Quadrillionth	Billiardth	0.000 000 000 000 001
1000^{-6}	10^{-18}	atto	a	Quintillionth	Trillionth	0.000 000 000 000 000 001
1000^{-7}	10^{-21}	zepto	z	Sextillionth	Trilliardth	0.000 000 000 000 000 000 001
1000^{-8}	10^{-24}	yocto	y	Septillionth	Quadrillionth	0.000 000 000 000 000 000 000 001

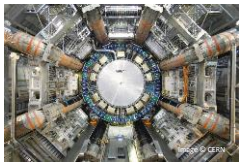
Kutatási módszerek

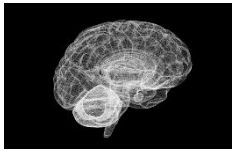












4000 years | 500 years | ~50 years | Today

1 – Empirical observations

2 - Generalization Theoretical models

3 - Simulations Computational sciences

4 - Data-driven science eScience

„Big Daták”

- Okostelefon GPS és Internetkapcsolattal
  - 4,6 milliárd mobil
  - 1-2 milliárd rendelkezik internetkapcsolattal
- Szenzorok
  - Nasa klímaadatok: 32 petabyte
  - Nagy hadron ütköztető (LHC):
    - 150 millió szenzor
    - 40 millió/s mérés
    - 0,001%-át használják

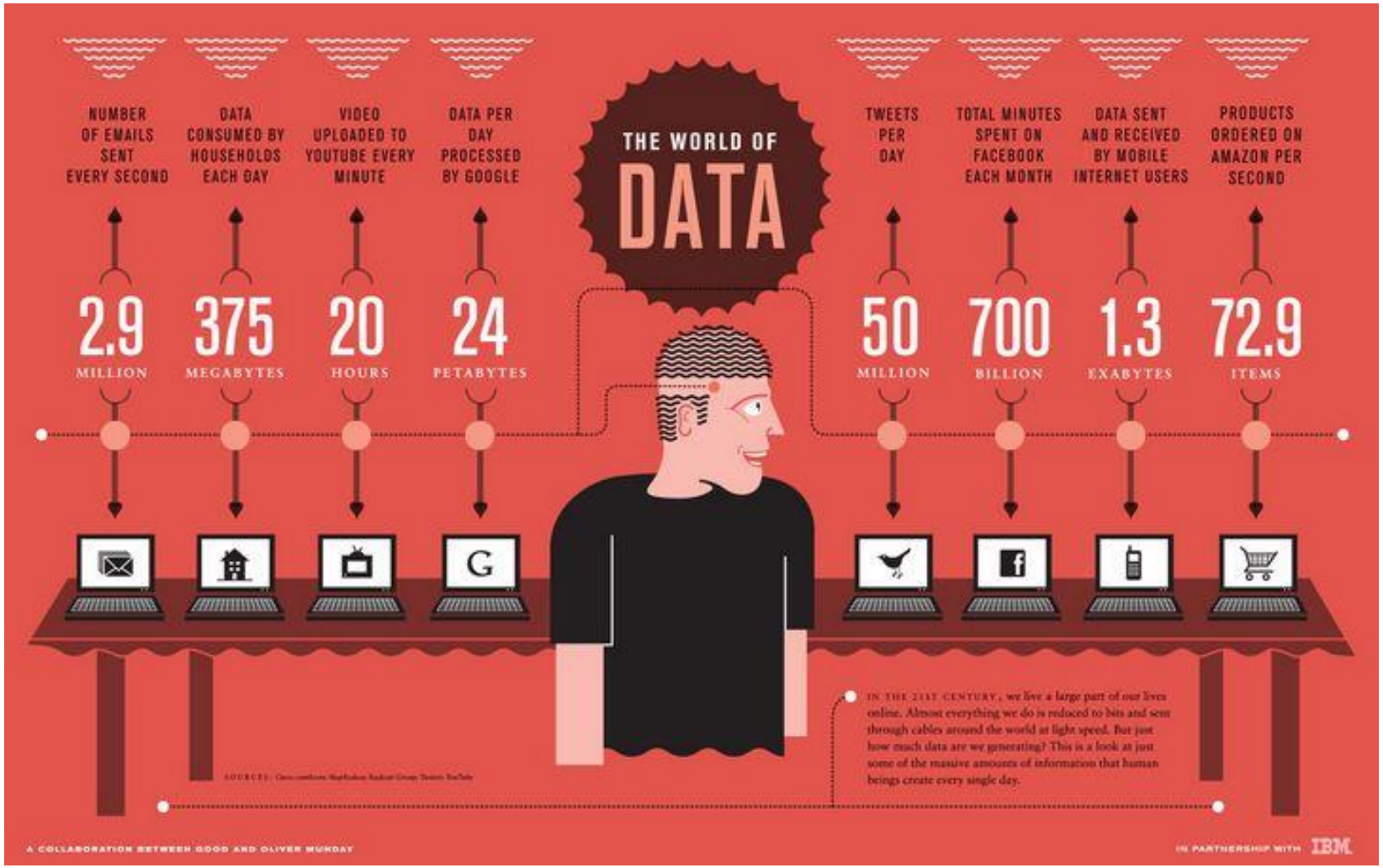
„Big Daták”

- Sloan Digital Sky Survey
  - Csillagászati adatok
  - 200 GB / éjszakánként
  - 140 terrabyte adat
- Közösségi hálózatok
  - Facebook
    - 1,06 milliárd felhasználó naponta
    - 30 milliárd elem megosztása naponta
  - Twitter
    - 175 millió tweet naponta
    - 465 millió felhasználó

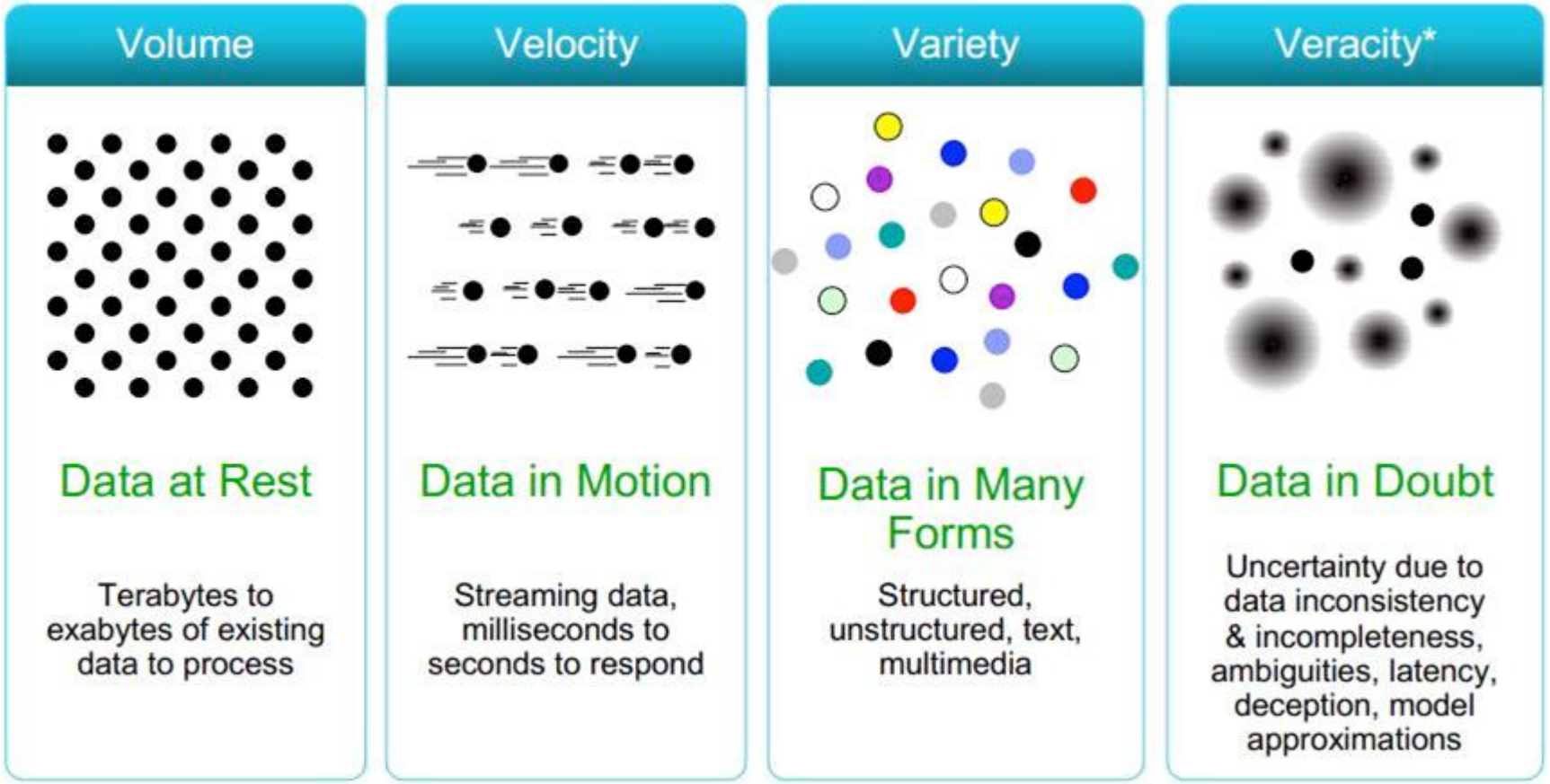
Milyen „big” a Big Data?

- 2.7 Zetabyte méretű
- 2020-ra 50x több adat mint ma
- 2012-ben az adatok 90%-a 2 év alatt „termelődött”
- 2 nap alatt több információt generálunk mint 2003 óta összesen

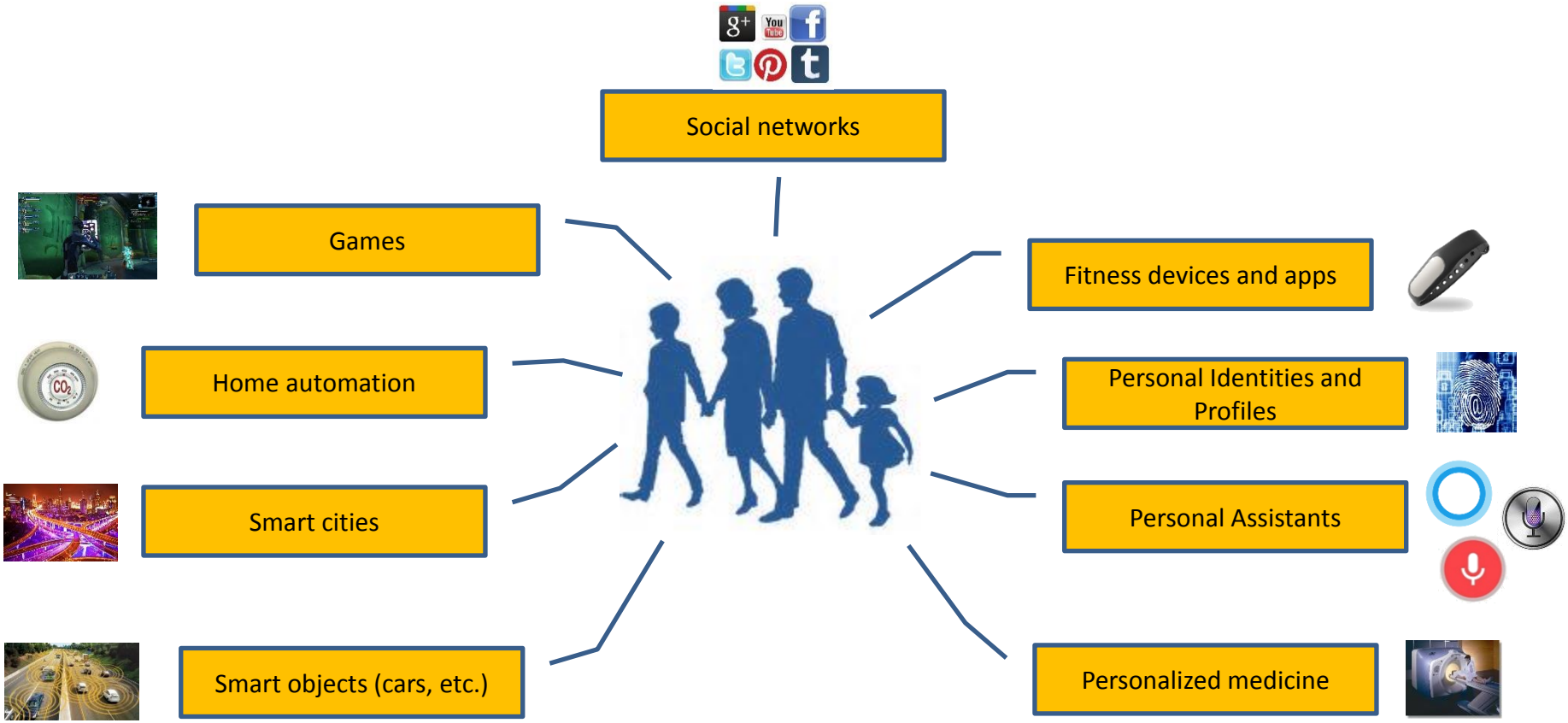
World of Big Data



Big Data 4V


Volume
Data at Rest
Terabytes to exabytes of existing data to process
Velocity
Data in Motion
Streaming data, milliseconds to seconds to respond
Variety
Data in Many Forms
Structured, unstructured, text, multimedia
Veracity*
Data in Doubt
Uncertainty due to data inconsistency & incompleteness, ambiguities, latency, deception, model approximations

Internet of Things (IoT)



Big Data célja

- „A vast quantity of UNSTRUCTURED data, which we now have the ability to process in **REAL-TIME**.”
  - http://www.youtube.com/watch?v=449twsMTrJI

    A day in big data

Big Data kezelés

- Adattárolási technikák (Lemezek, kazetták, memoria)
- Gyors adatelérés
- Dinamikus hálózatok

Big Data elemzés

- Adatelemzés, elemzési platformok
- Minta felismerés, machine learning, előrejelzés
- Adat vizualizálás
- Alkalmazás szintű feldatok
  - Mobil elérés, adatok megjelenítése

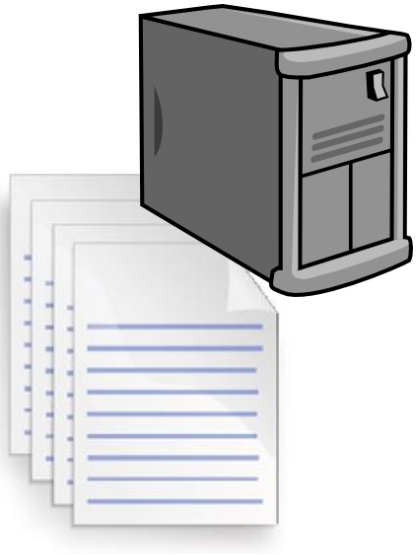
Big Data elemzés

- Adatelemzés, elemzési platformok
  - Distributed Computing
- Adattárolási technikák
  - Distributed Storage

Big Data tárolás

Feladat: tároljuk az adatokat hatékonyan.

Egyszerű eset: nagy server.

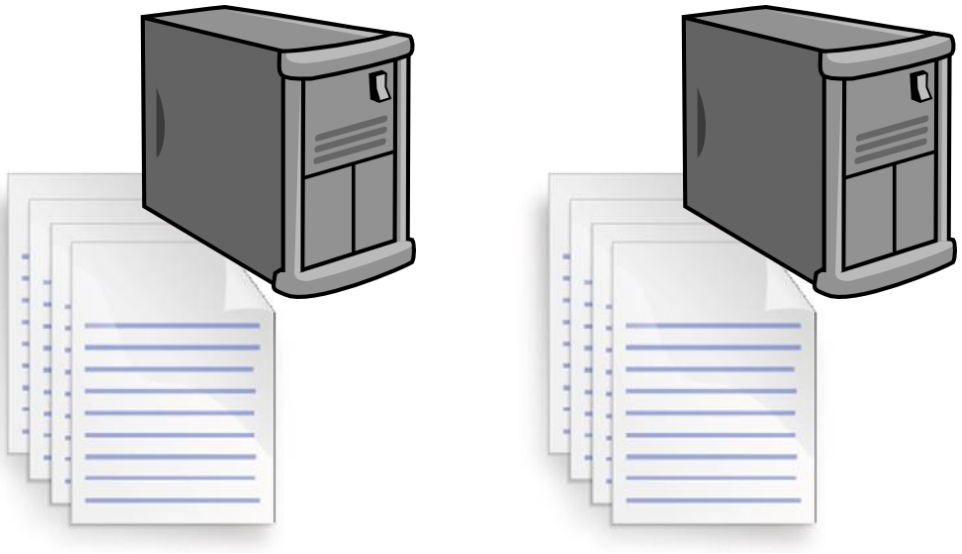


Big Data tárolás

Probléma: Elfogy a tárhely a serveren.

Megoldás: nagyobb server.

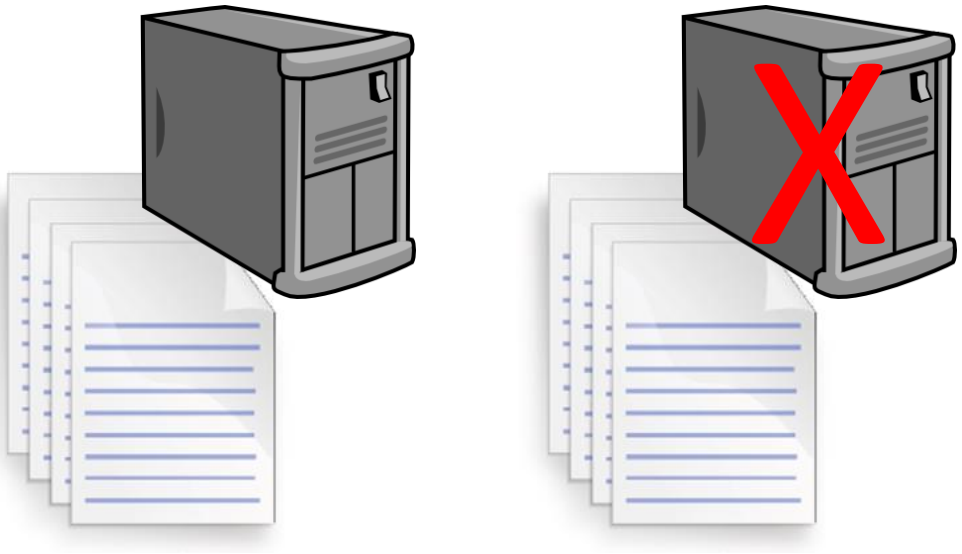
Jobb megoldás: Több server.



Big Data tárolás

Probléma: Elromlik az egyik serverünk

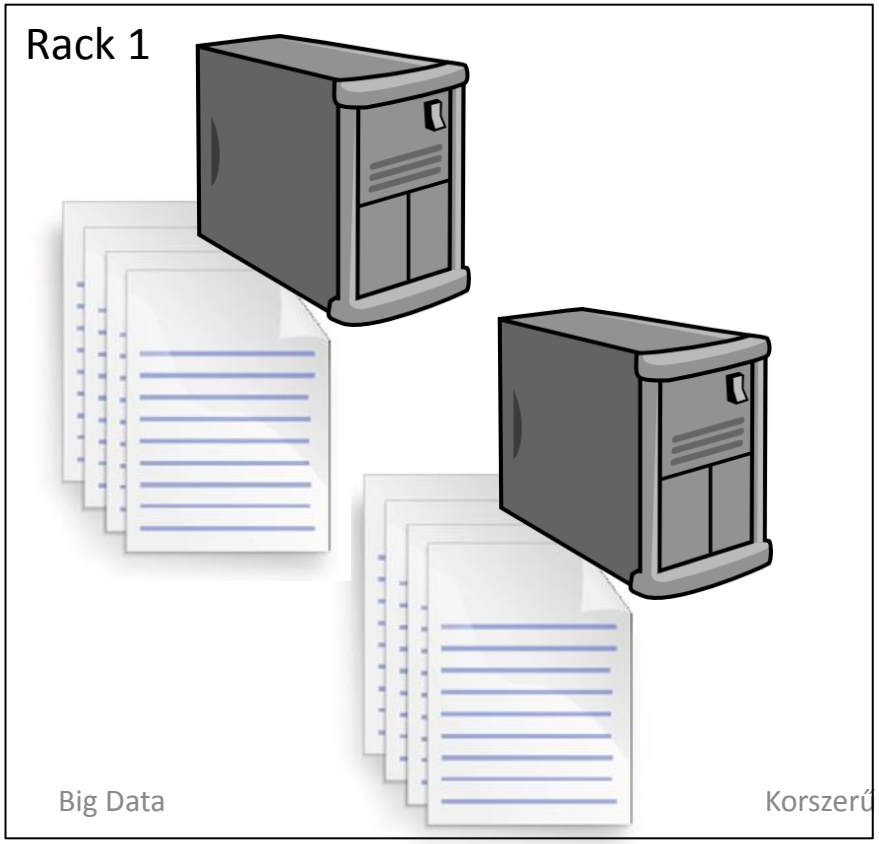
Megoldás: Replica. Ugyanaz az adat több serveren is tárolódjon.

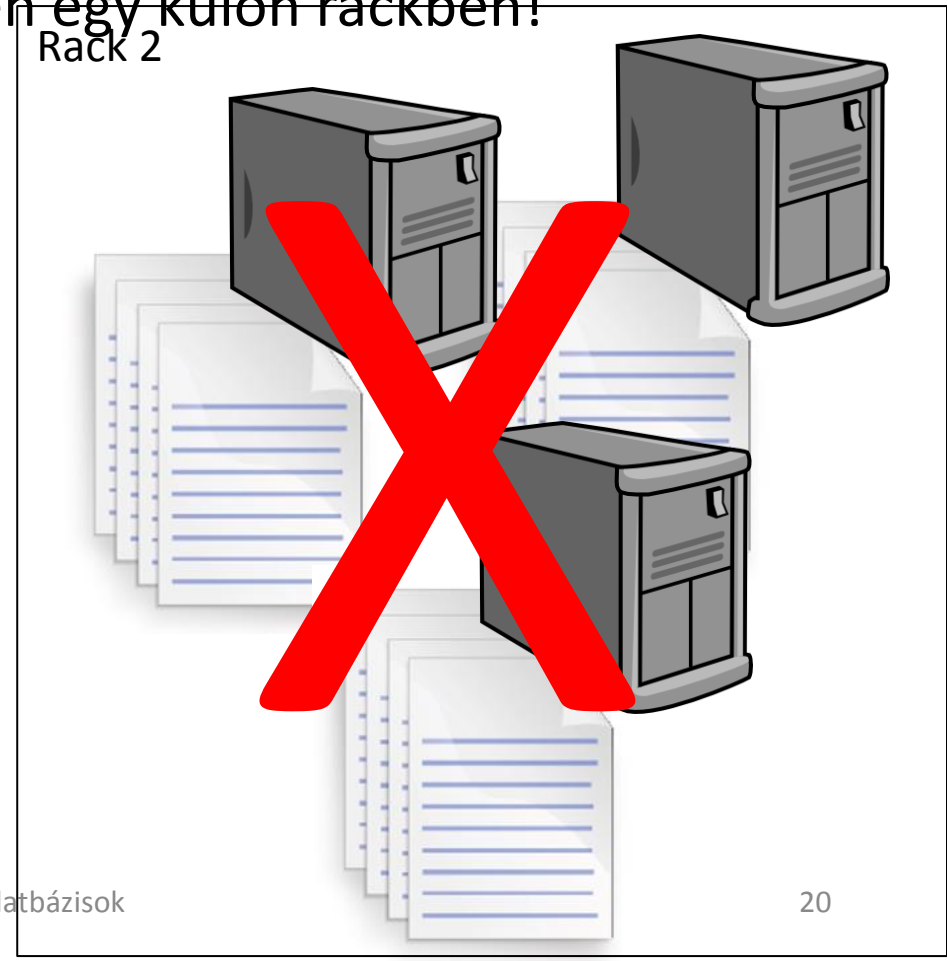


Big Data tárolás

Probléma: Elromlik a rack és a replica-k egy rackben tárolódnak.

Megoldás: 1 Replica mindig legyen egy külön rackben!

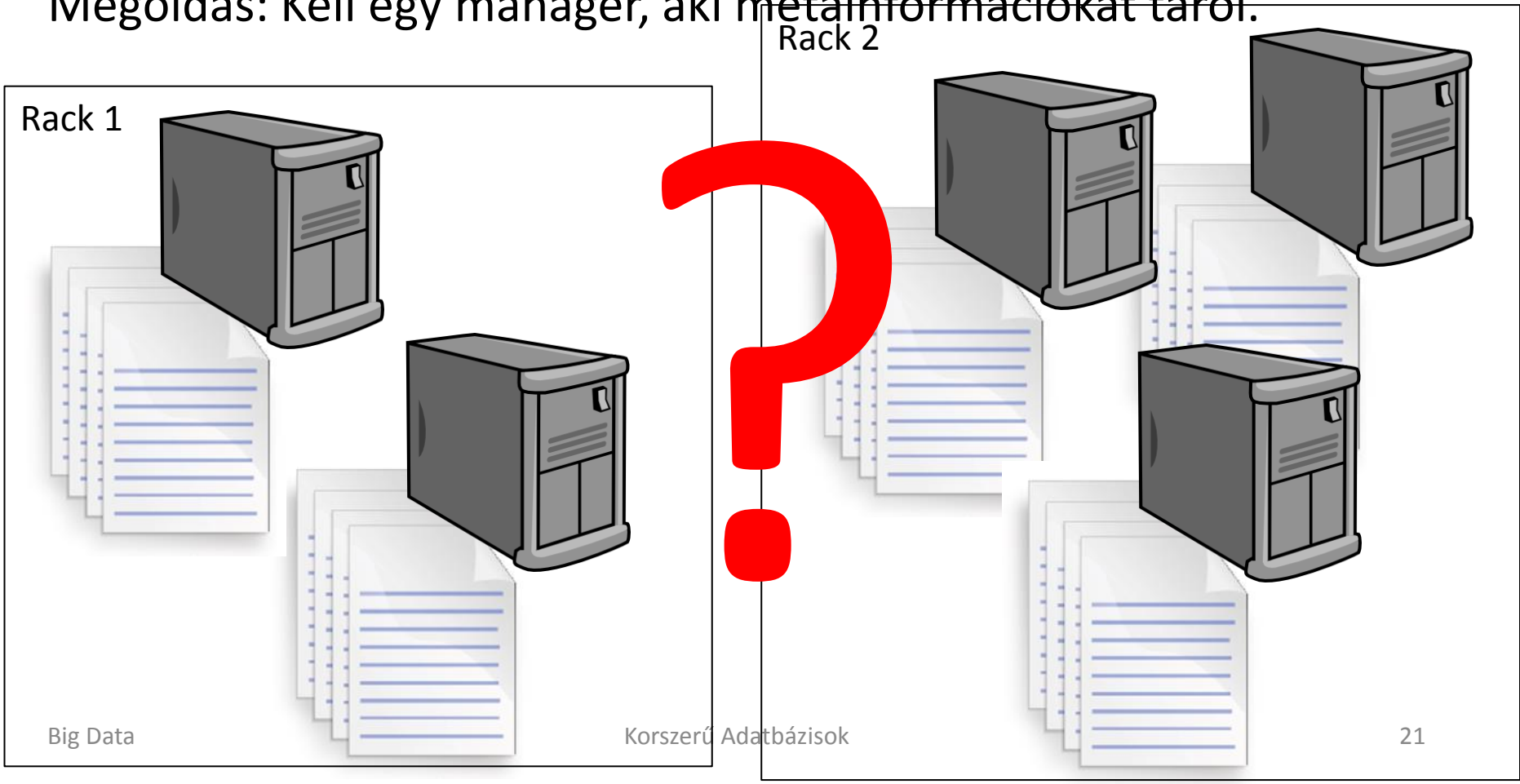




Big Data tárolás

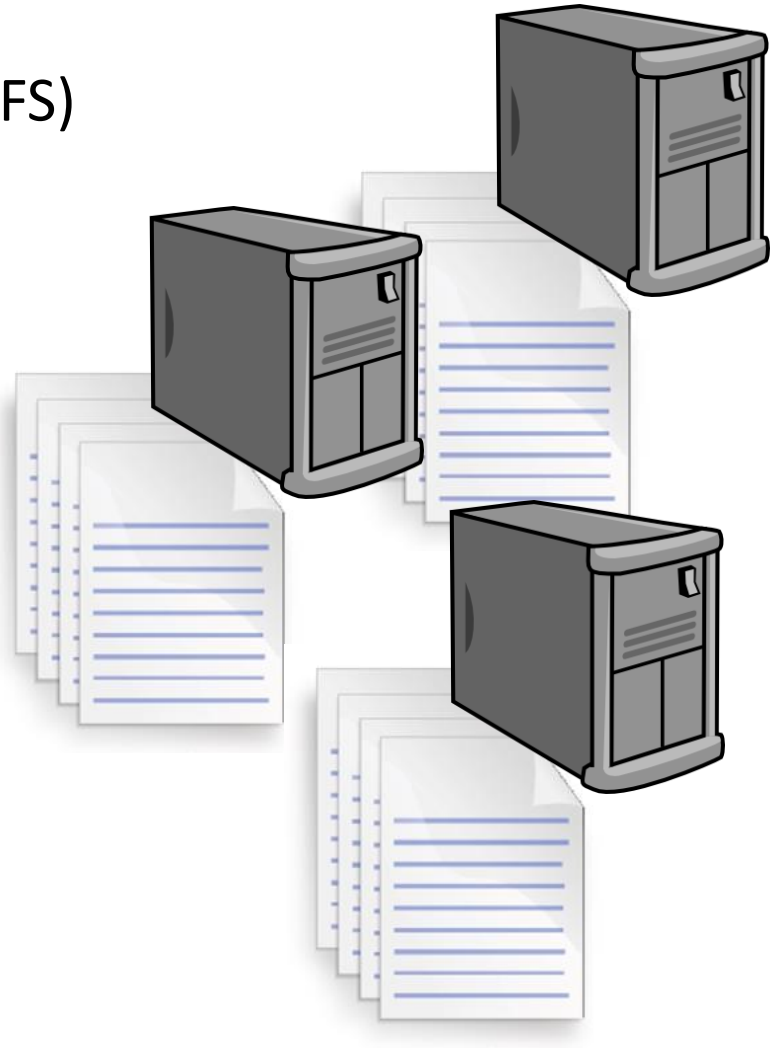
Probléma: Honnan tudjuk melyik adat hol van?

Megoldás: Kell egy manager, aki ~~metainformációkat tárol~~.



HDFS

- Hadoop Distributed Filesystem (HDFS)
- Default replica szám: 3
- Replica elhelyezés:
  - 1 mindig egy másik rackben
- 2 kompnens:
  - Namenode: metaadat manager
  - Datanode: tárolja az adatot
- Skálázható, új gépek hozzávételével
- Unix-szerű-en tudjuk használni



HDFS példa

hadoop@dbpc62:~$ hdfs dfs -ls /user/ggombos/giraphInput

-rw-r--r-- 3 ggombos ggombos 112 2015-01-29 15:55 /user/ggombos/giraphInput/tiny_giraph.txt

hadoop@dbpc62:~$ hdfs dfs -cat /user/ggombos/giraphInput/tiny_giraph.txt

[0,0,[[1,1],[3,3]]]
[1,0,[[0,1],[2,2],[3,1]]]
[2,0,[[1,2],[4,4]]]
[3,0,[[0,3],[1,1],[4,4]]]
[4,0,[[3,4],[2,4]]]

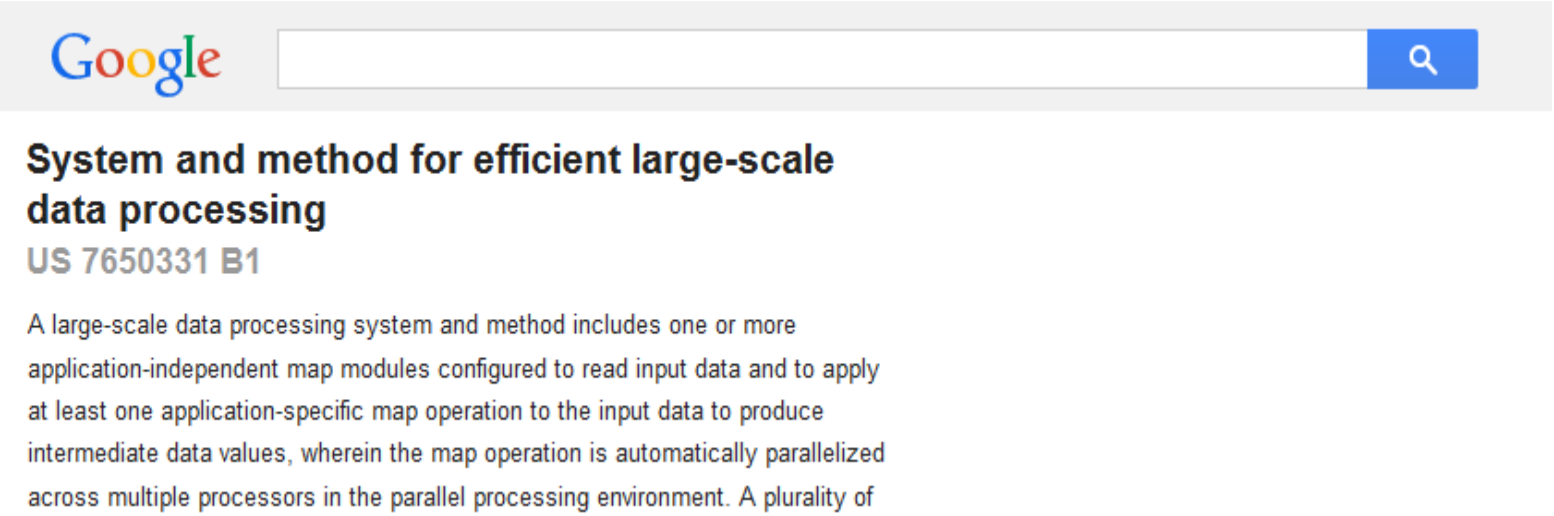
hadoop@dbpc62:~$ hdfs dfs -mkdir KorszeruDemo

hadoop@dbpc62:~$ hdfs dfs -rmdir KorszeruDemo

hadoop@dbpc62:~$ hdfs dfs -put valami.txt KorszeruDemo

hadoop@dbpc62:~$ hdfs dfs -get KorszeruDemo/valami.txt .

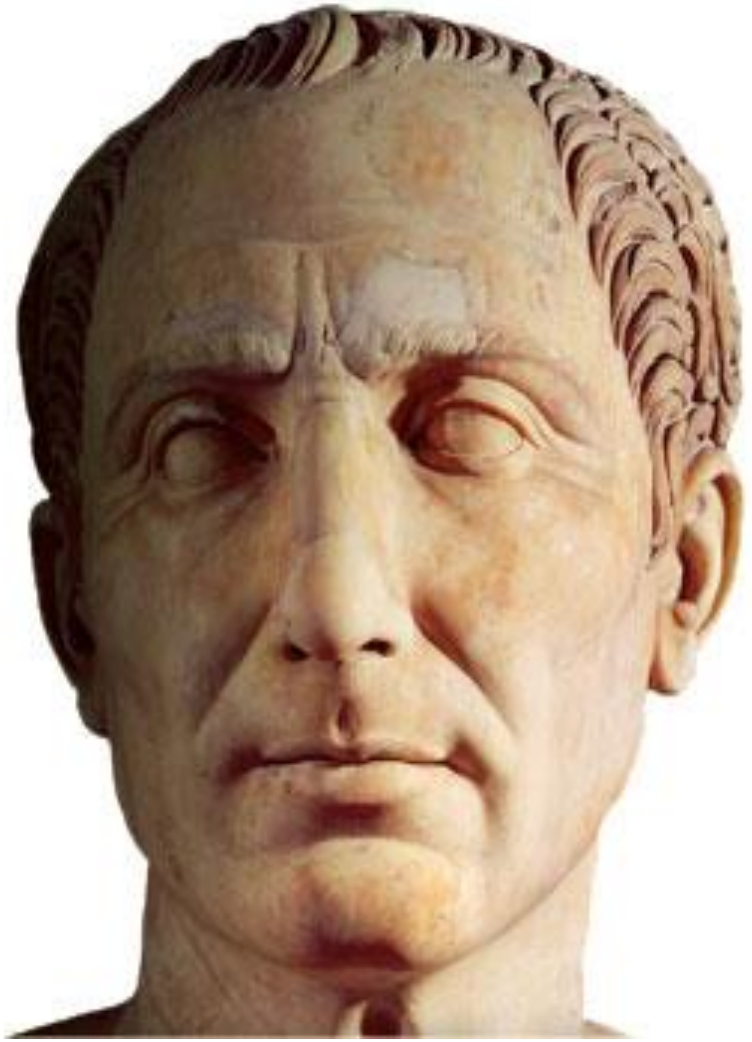
Distributed Computing



Legismertebb algoritmus, a MapReduce algoritmus, amit a Google talált ki.

DE, Google előtt is feltalálták: Julius Caesar

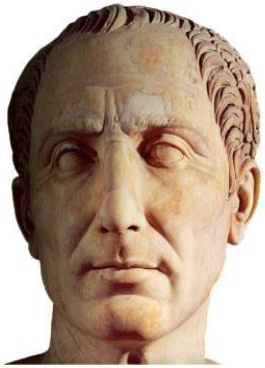
Divide et impera*



* Divide and conquer

Példa

Hány oldal íródott latin nyelven az ókori Alexandria könyvtárában?



GREEK REF7 P20 | LATIN REF4 P73 | LATIN REF1 P45

LATIN ✓
Oldal 45

LATIN REF5 P34 | GREEK REF2 P128

Vár
Olvas..

Reducer

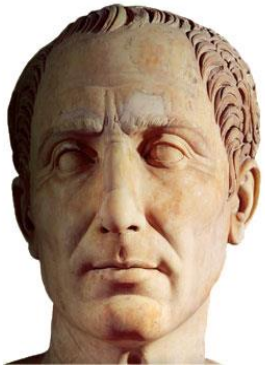
GREEK REF8 P230 | EGYPT REF6 P10 | EGYPT REF3 P12

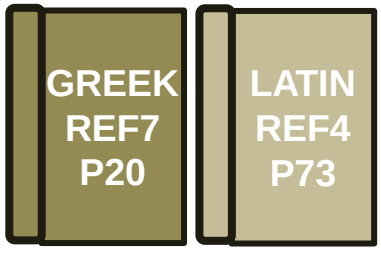
EGYPTIAN ✗

Mappers

Példa

Hány oldal íródott latin nyelven az ókori Alexandria könyvtárában?





vár
olvas…

GREEK ✗

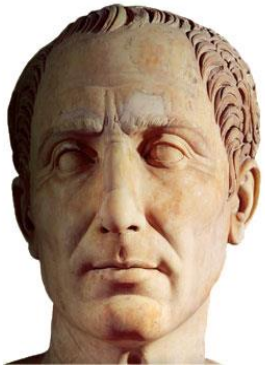
Reducer

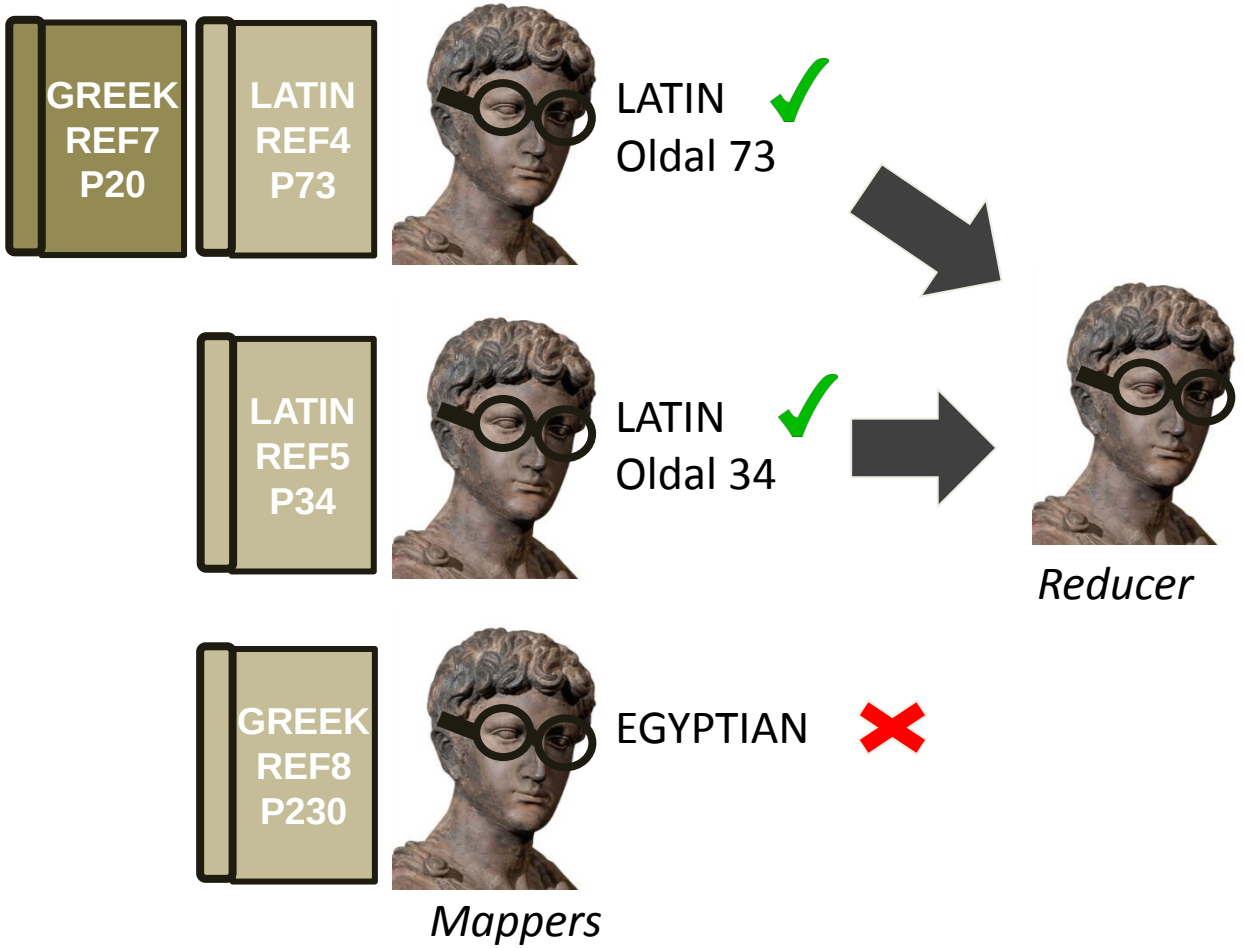
EGYPTIAN ✗

Mappers

Példa

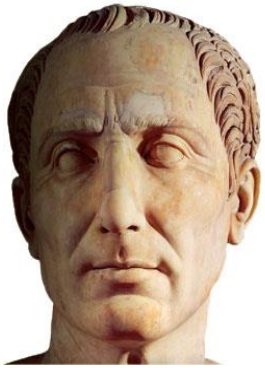
Hány oldal íródott latin nyelven az ókori Alexandria könyvtárában?





Példa

Hány oldal íródott latin nyelven az ókori Alexandria könyvtárában?

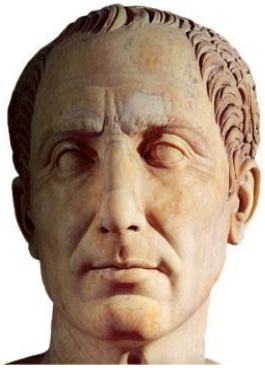


Mappers

Reducer

Példa

Hány oldal íródott latin nyelven az ókori Alexandria könyvtárában?



vár…

vár…

vár…

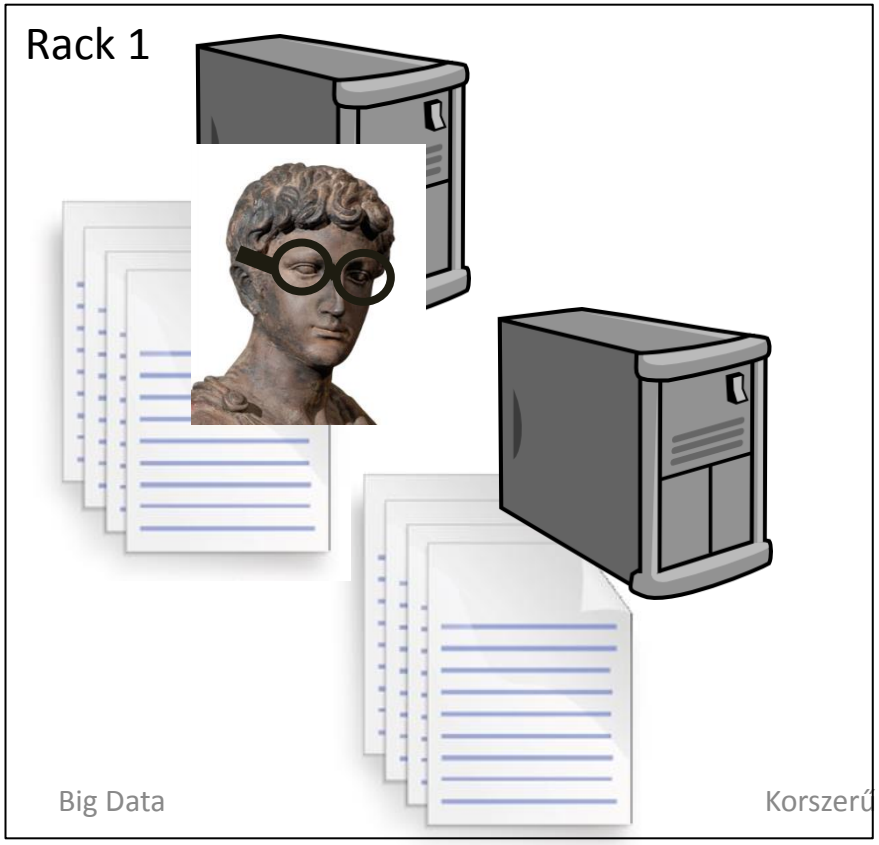
Mappers

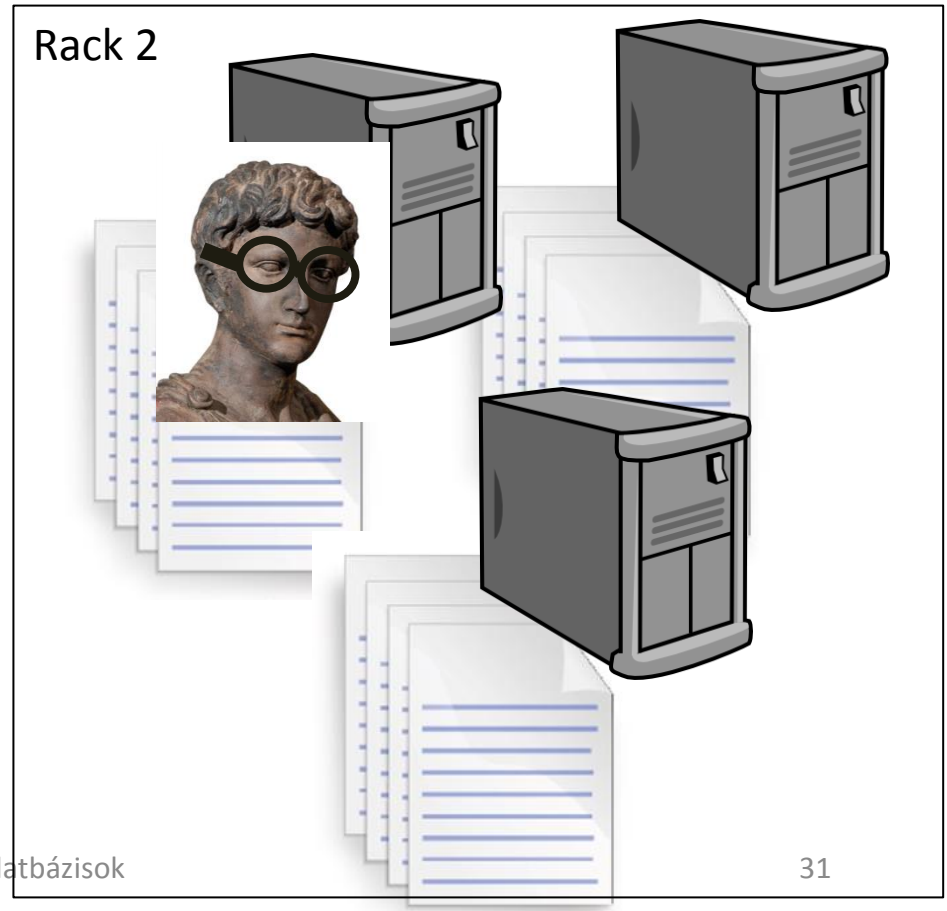
Reducer

Distributed Computing

Probléma: Nagy méretű adatok mozgatása a hálózaton!

Megoldás: Ott fusson, ahol az adat található.

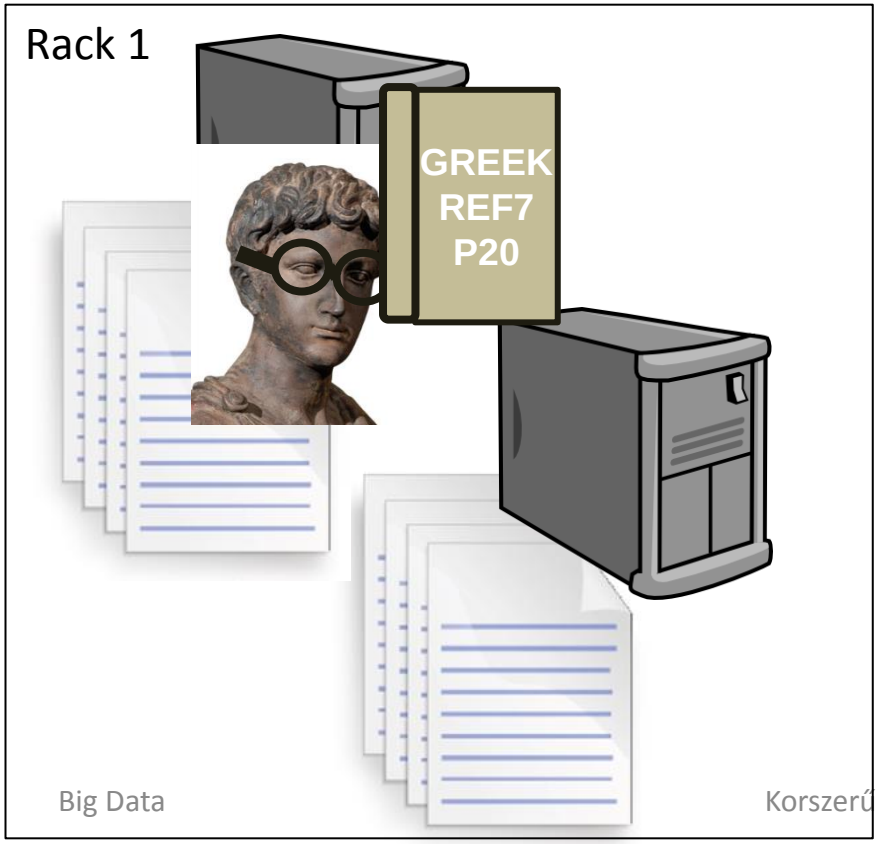


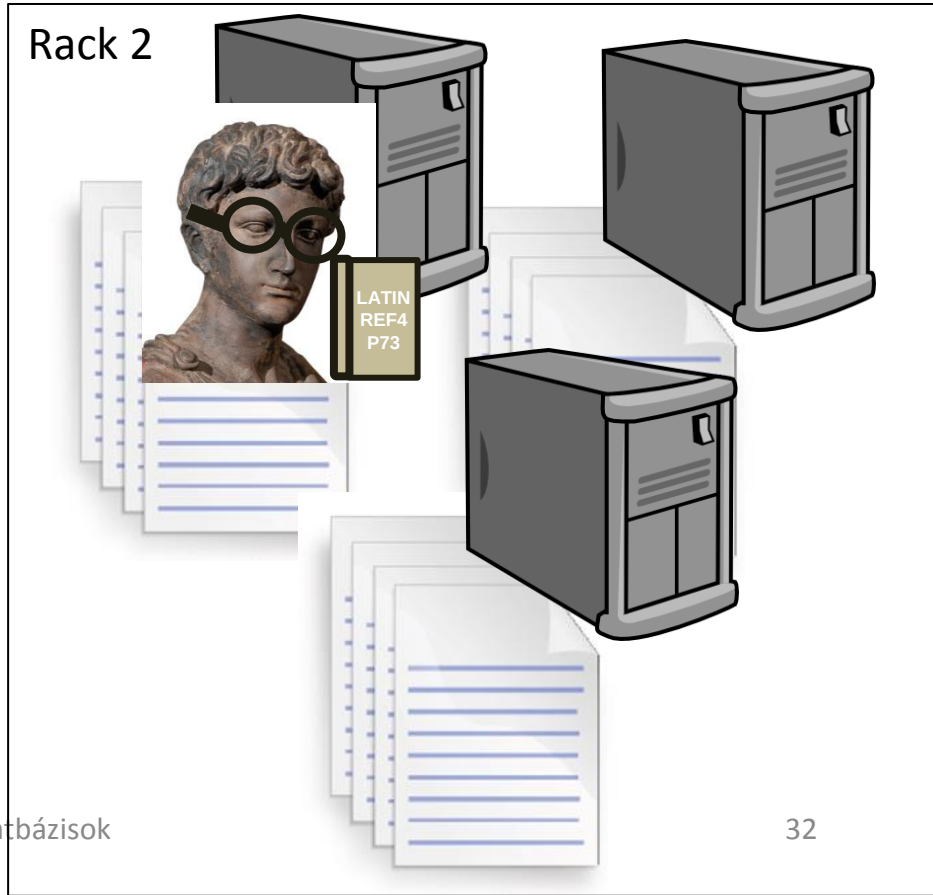


Distributed Computing

Probléma: Több méretű fájlok. (1 MB vs 1 TB)

Megoldás: Egység méretűek (block) legyenek a fájlok. (Split)

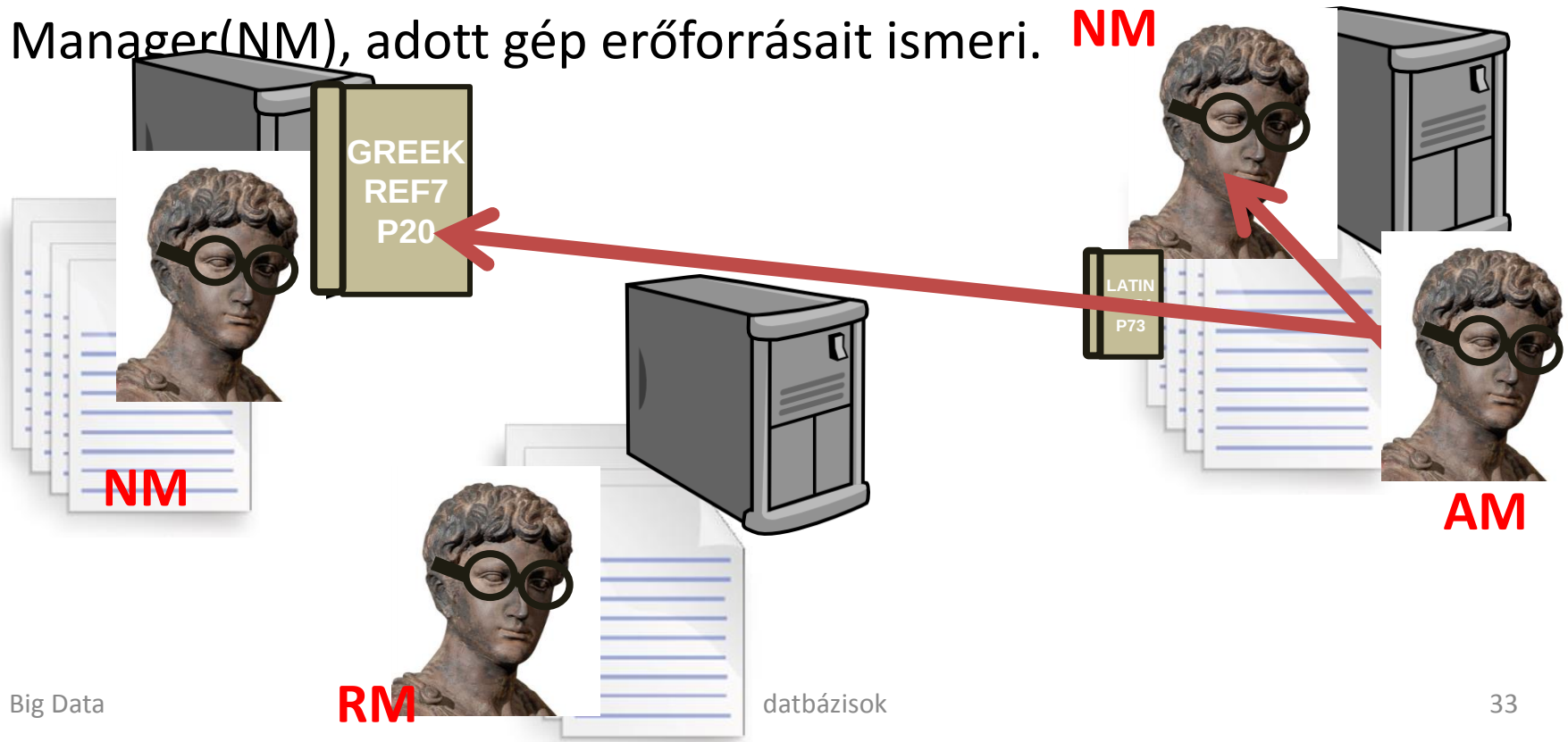




Distributed Computing

Probléma: Vezényelni kell a futást.

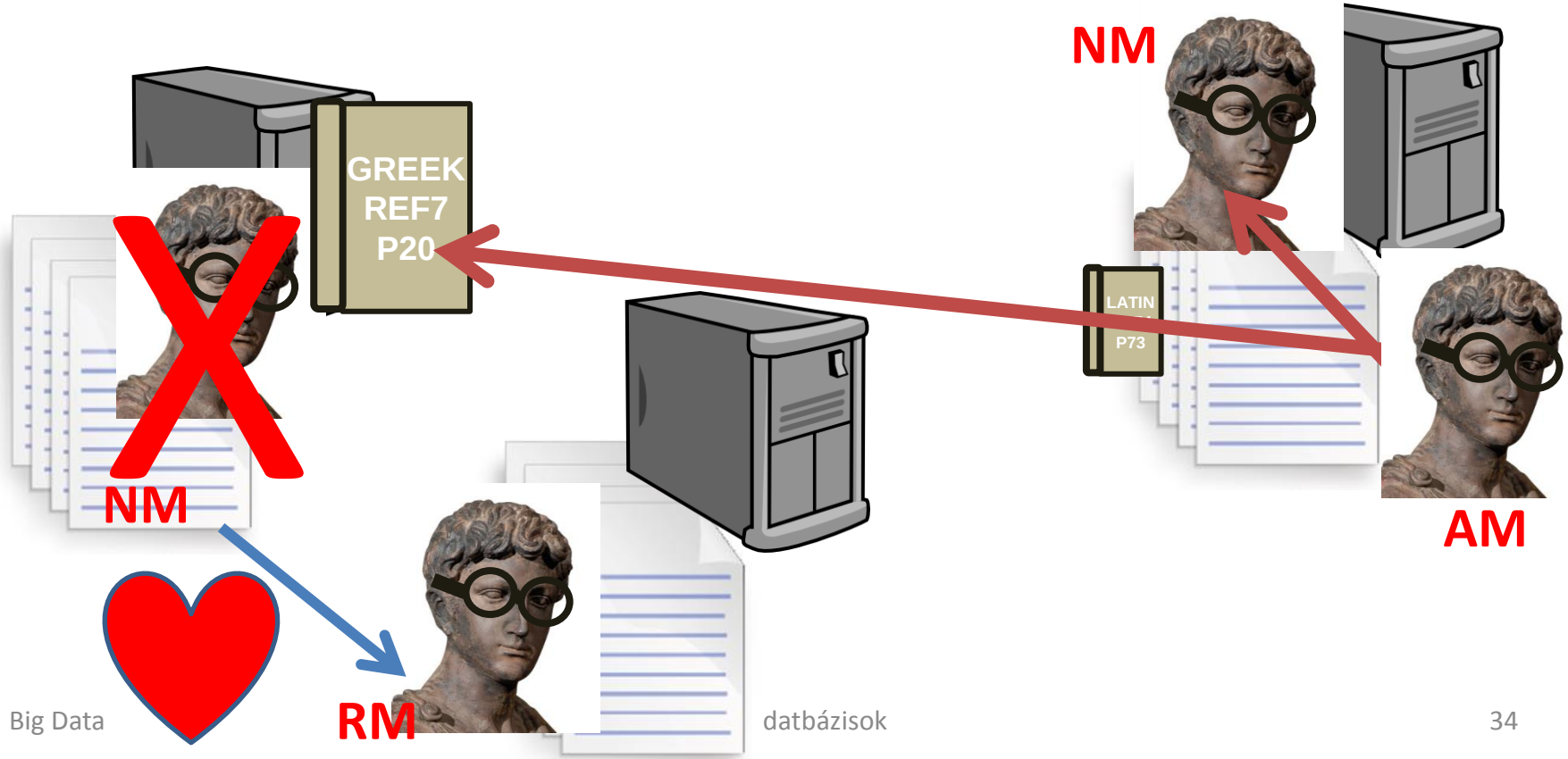
Megoldás: Application Master (AM), aki vezényli a munkát. Resource Manager (RM), erőforrás kiosztást ismeri. Node Manager(NM), adott gép erőforrásait ismeri.



Distributed Computing

Probléma: Kiesik egy NM. Hogyan vesszük észre?

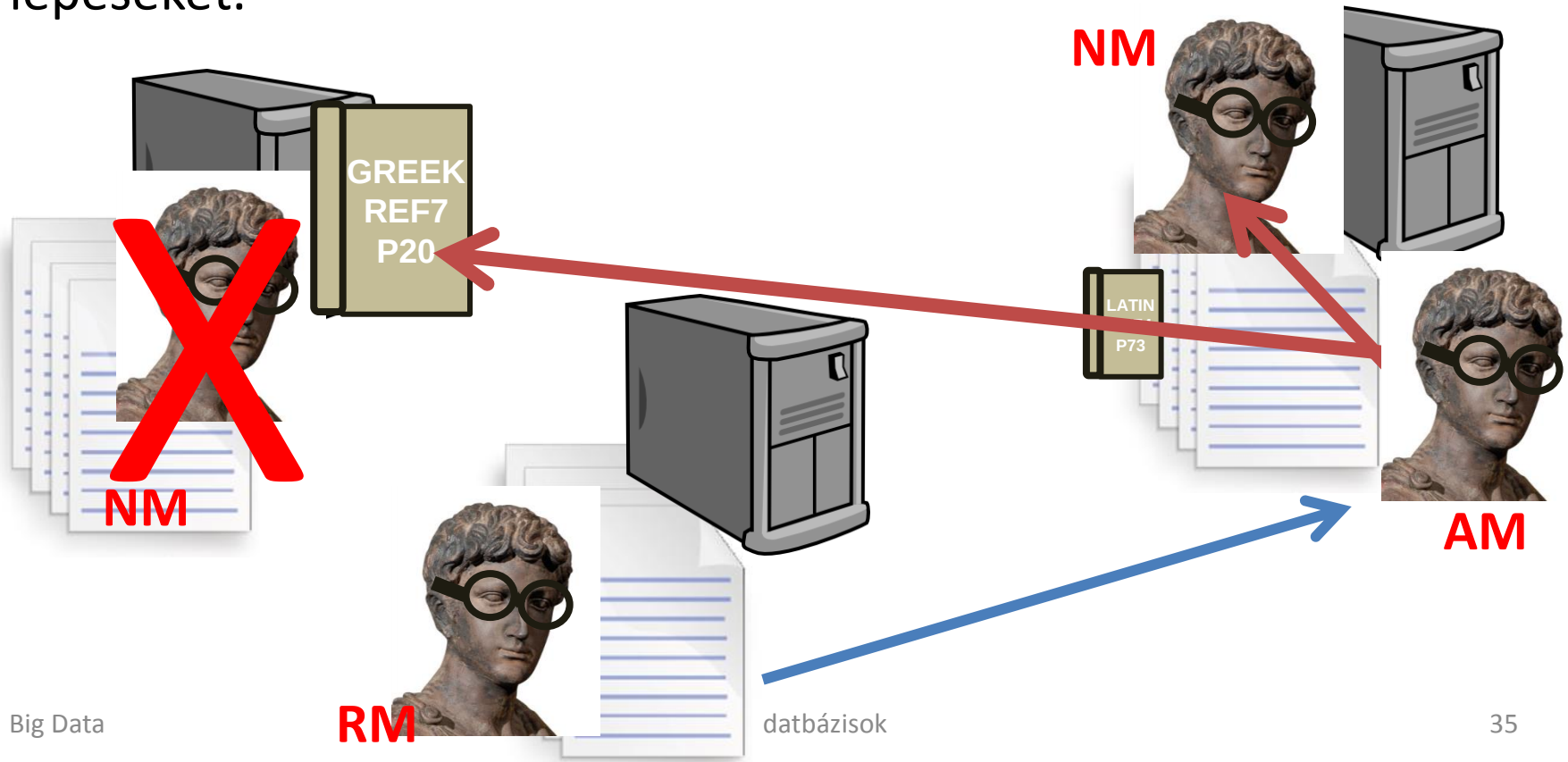
Megoldás: Heartbeat. Időközönként jelzünk a RM-nak.



Distributed Computing

Probléma: Kiesik egy NM. Hogyan orvosoljuk?

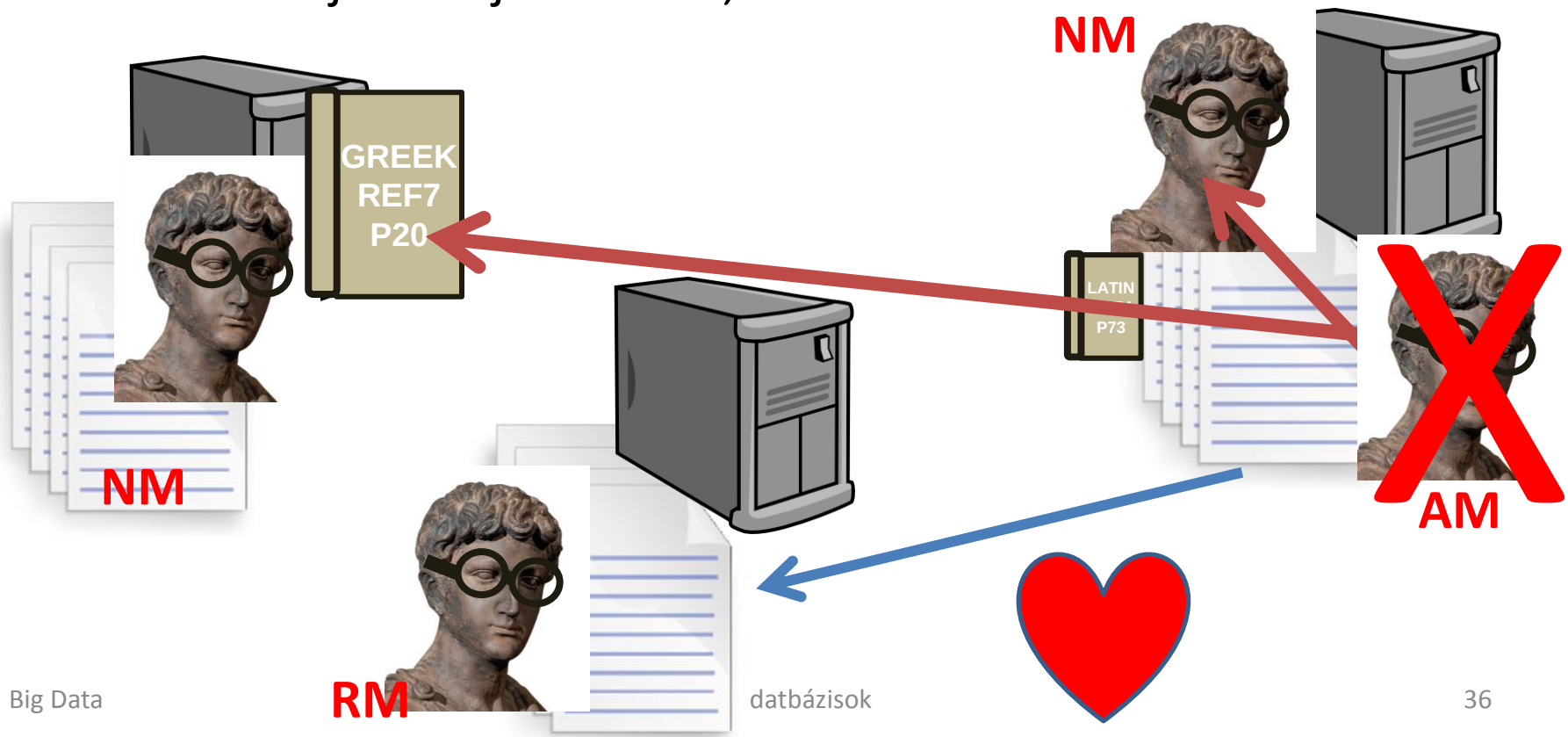
Megoldás: RM szól a megfelelő AM-nek, aki megteszi a megfelelő lépéseket.



Distributed Computing

Probléma: Kiesik az AM. Hogyan orvosoljuk?

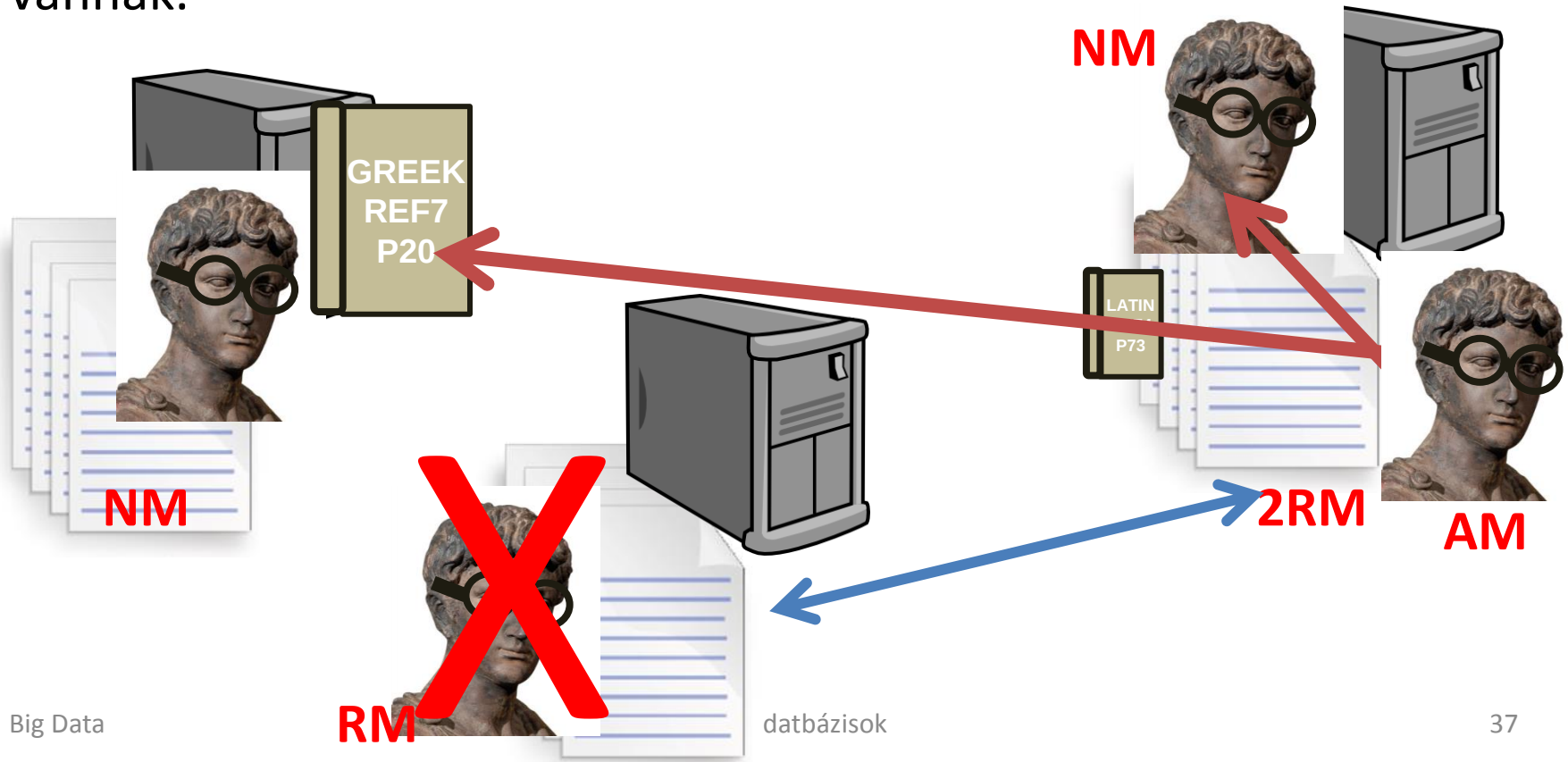
Megoldás: AM szintén heartbeat-eket küld az RM-nek. Ha nincs akkor az RM újraindítja az AM-t, aki szinkronizál a NM-ekkel.



Distributed Computing

Probléma: Kiesik a RM. Hogyan orvosoljuk?

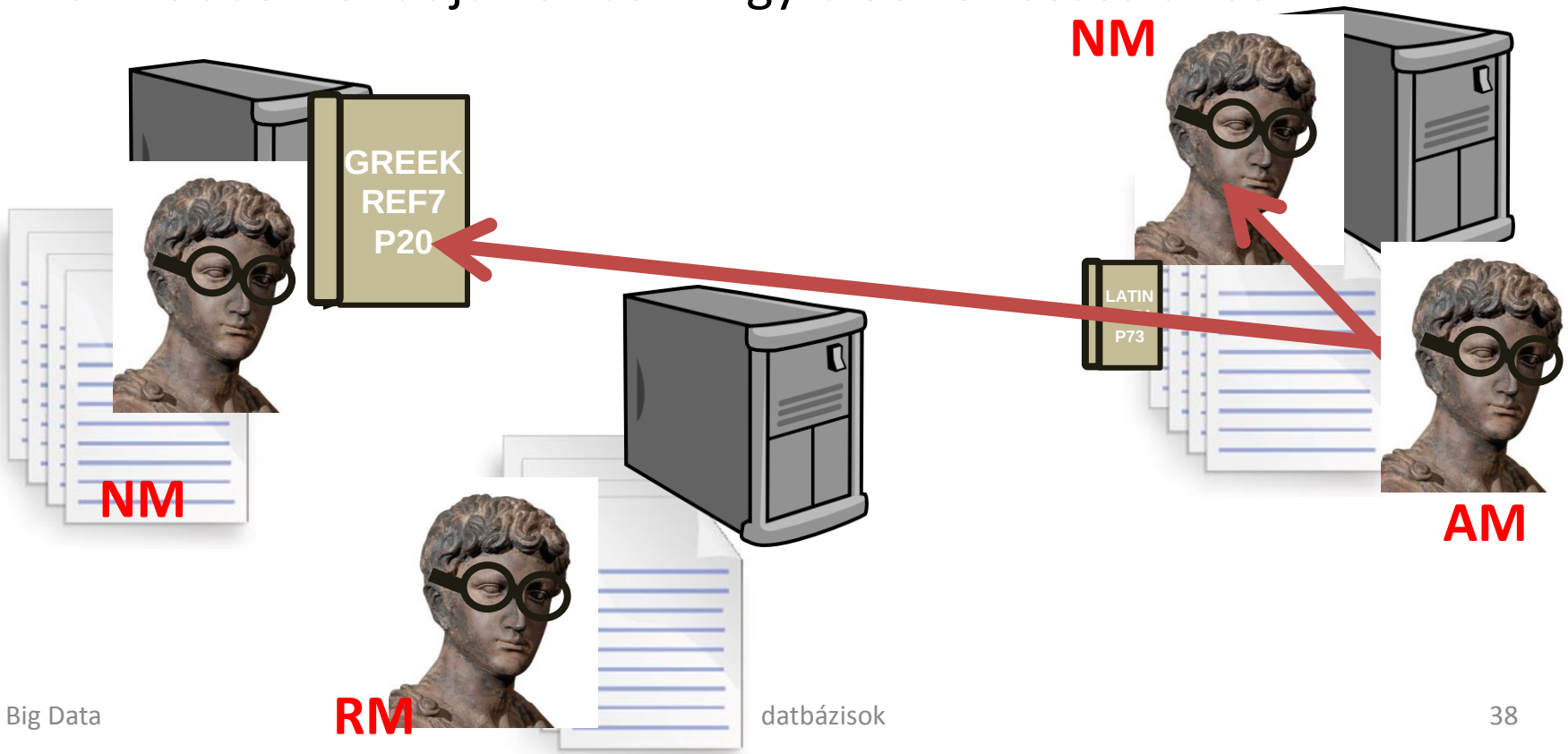
Megoldás: Secondary RM, aki átveszi szerepét. Szinkronban vannak.



Distributed Computing

Probléma: Kihasználatlan node-ok.

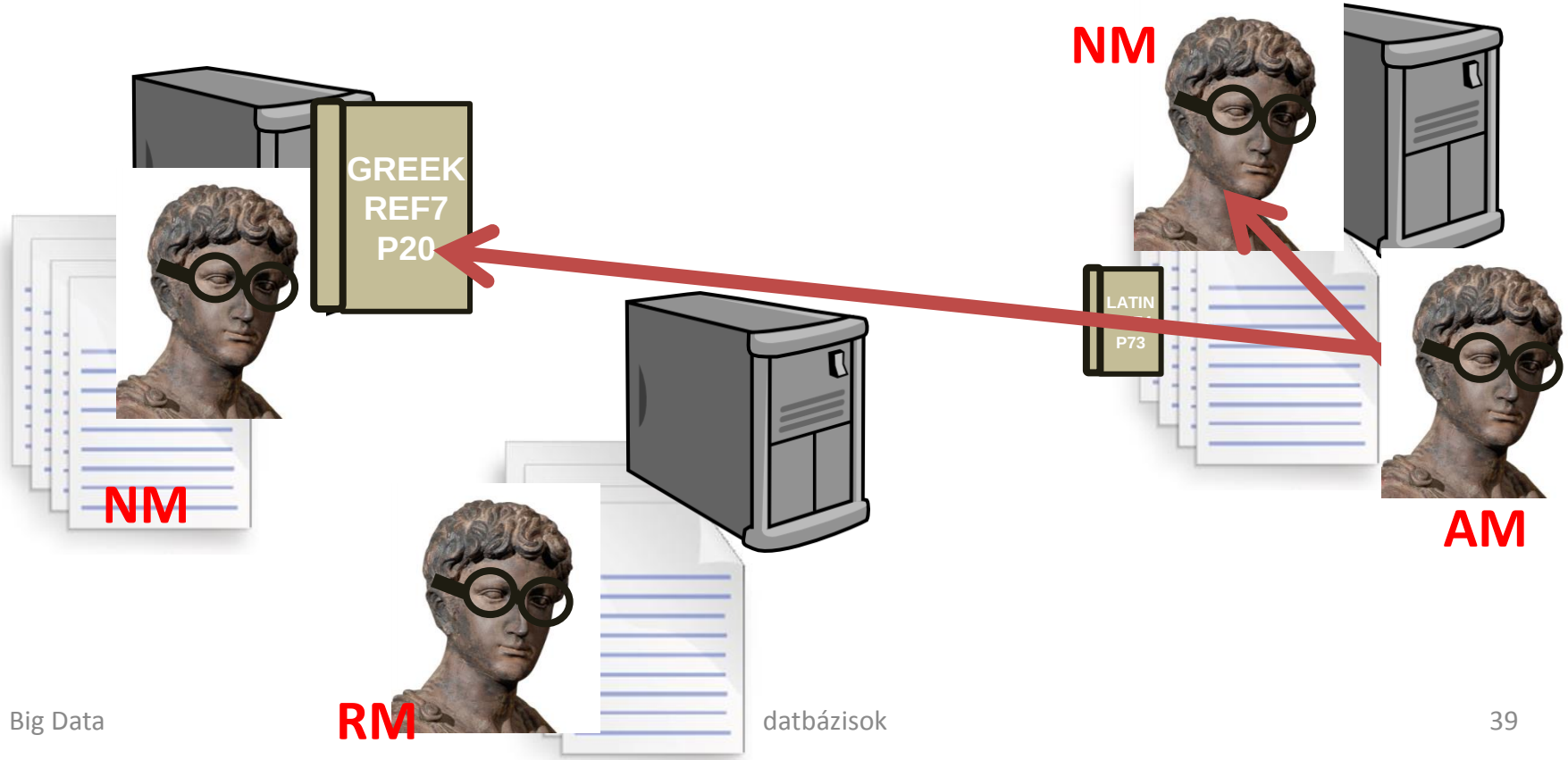
Megoldás: Minden node olvassa a nála lévő block-ot. A replica miatt többen is tudják olvasni. Egy block olvasása a Task.



Distributed Computing

Probléma: Leáll egy Task futása.

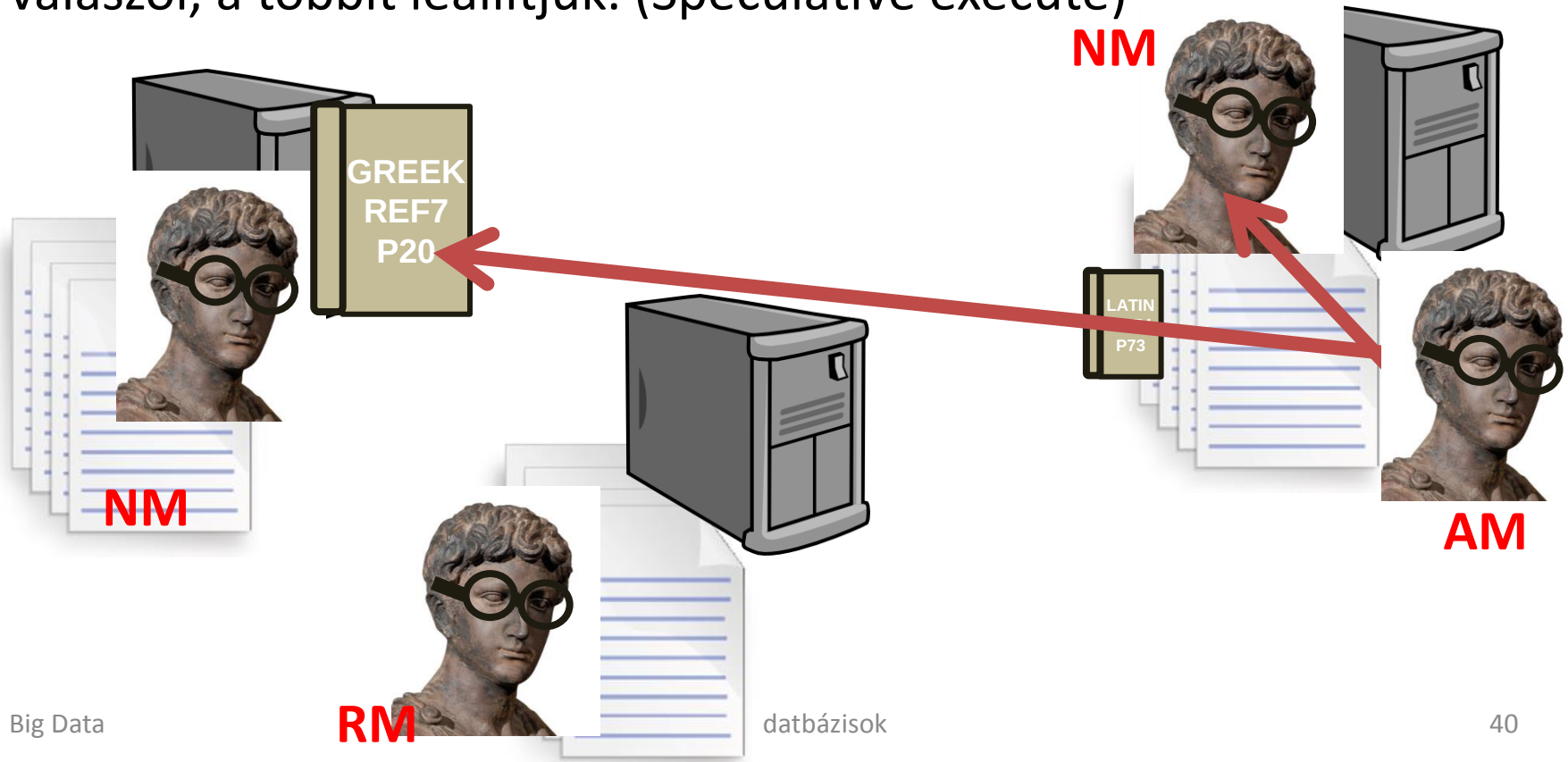
Megoldás: Újraindítjuk.



Distributed Computing

Probléma: Egyes node-ok lassúak.

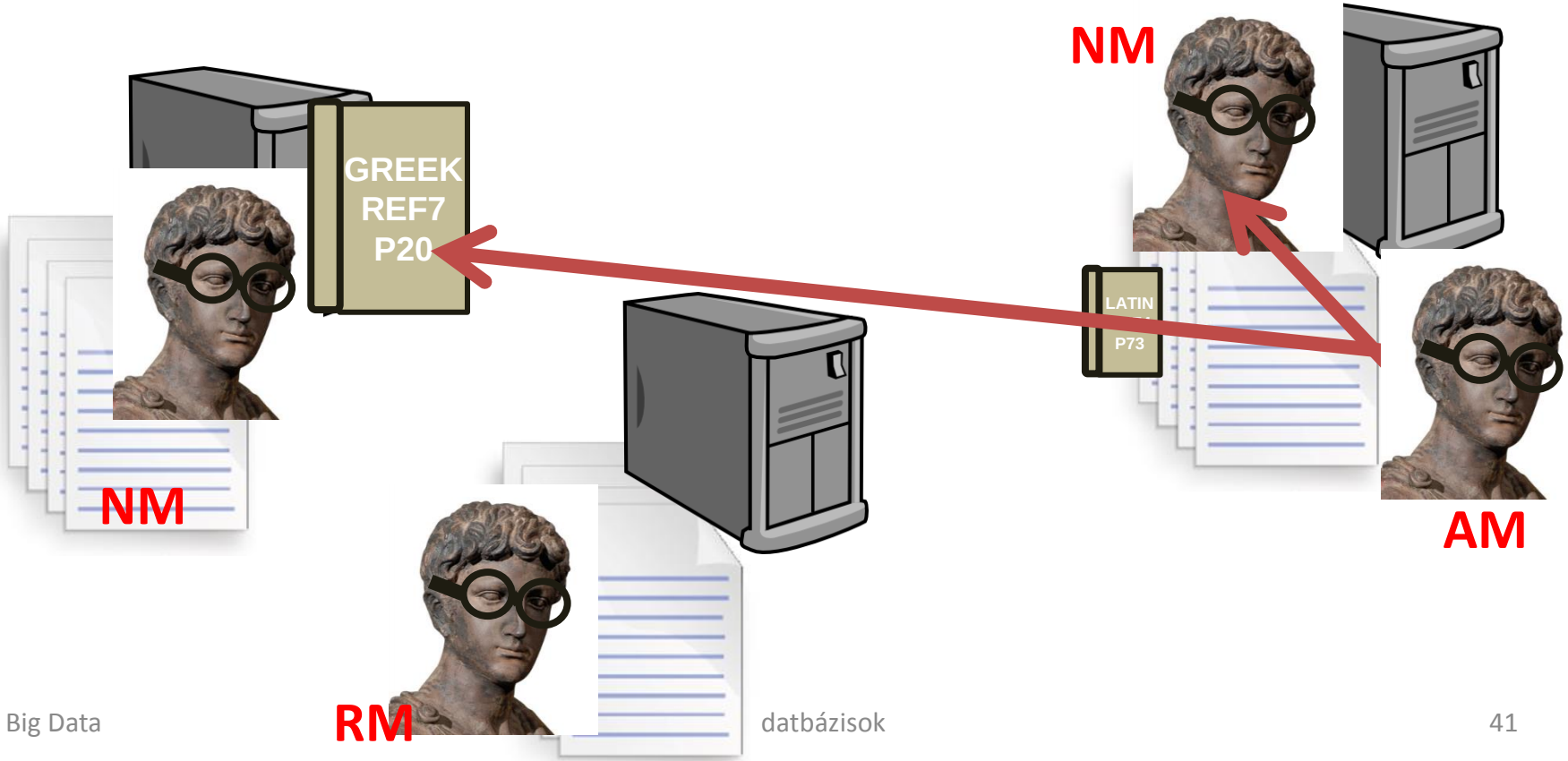
Megoldás: Többen olvassák ugyanazt a block-ot, és a legyorsabb válaszol, a többit leállítjuk. (Speculative execute)



Distributed Computing

Probléma: Több alkalmazás is futhat a rendszerben.

Megoldás: Ütemezők alkalmazása. (Scheduler)



Hadoop

Kiknek jó a hadoop?

"... to create building blocks for programmers who just happen to have ***lots of data to store, lots of data to analyze, or lots of machines to coordinate***, and who don't have the time, the skill, or the inclination to become distributed systems experts to build the infrastructure to handle it."

Tom White

Hadoop: The Definitive Guide

Kik haszálják?

Hadoop történelem

- 2004 – Google publikálja a MapReduce technikát
- 2006 – Apache projekt lett Yahoo! támogatással
- További támogatók:

Hadoop Sorts a Petabyte in 16.25 Hours and a Terabyte in 62 Seconds

By aanand – Mon, May 11, 2009 11:00 AM EDT

Recommend 0 Tweet 3

We used Apache Hadoop to compete in Jim Gray's Sort benchmark. Jim's Gray's sort benchmark consists of a set of many related benchmarks, each with their own rules. All of the sort benchmarks measure the time to sort different numbers of 100 byte records. The first 10 bytes of each record is the key and the rest is the value. The **minute sort** must finish end to end in less than a minute. The **Gray sort** must sort more than 100 terabytes and must run for at least an hour. The best times we observed were:

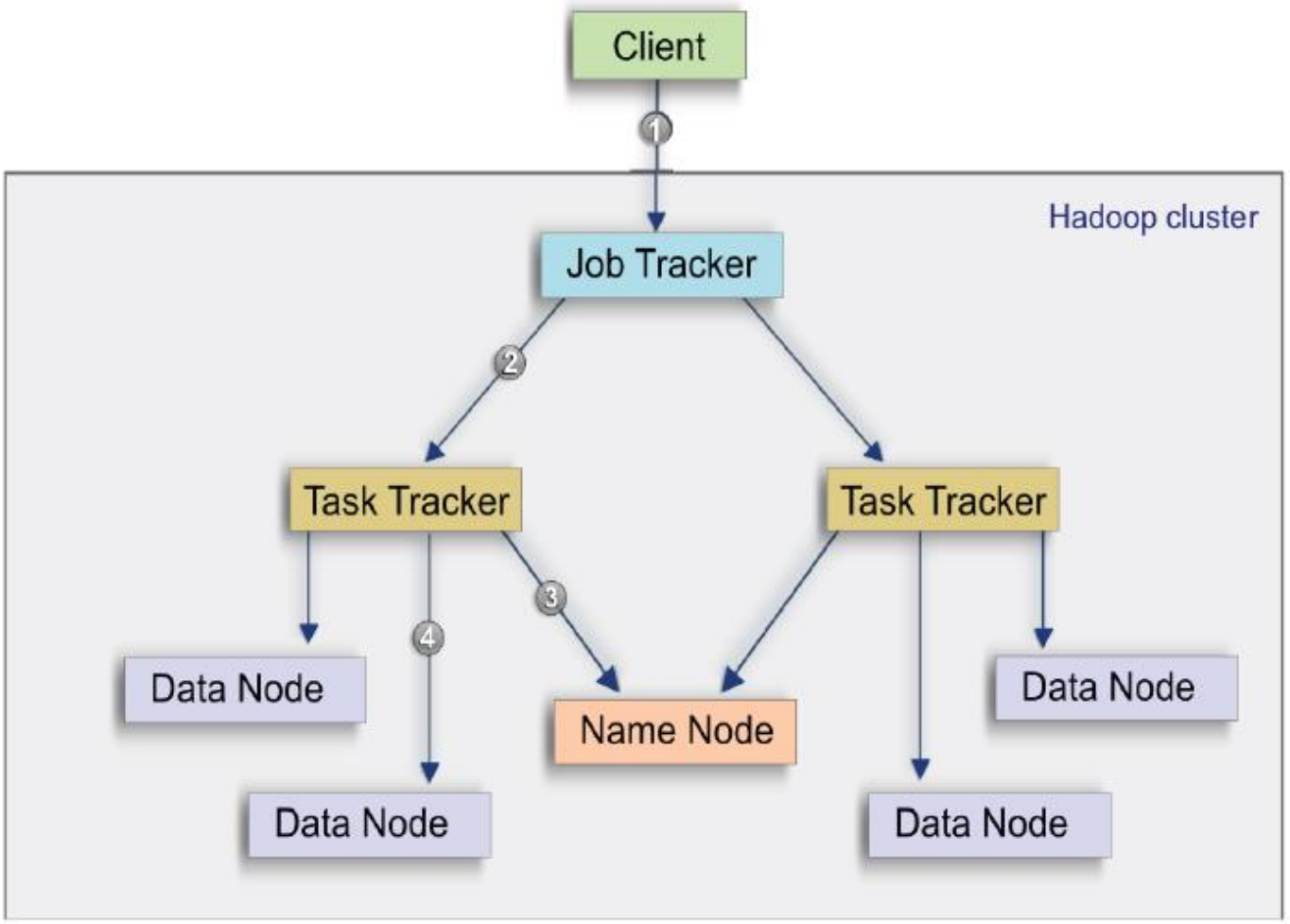
Bytes	Nodes	Maps	Reduces	Replication	Time
500,000,000,000	1406	8000	2600	1	59 seconds
1,000,000,000,000	1460	8000	2700	1	62 seconds
100,000,000,000,000	3452	190,000	10,000	2	173 minutes
1,000,000,000,000,000	3658	80,000	20,000	2	975 minutes

approximately 3800 nodes

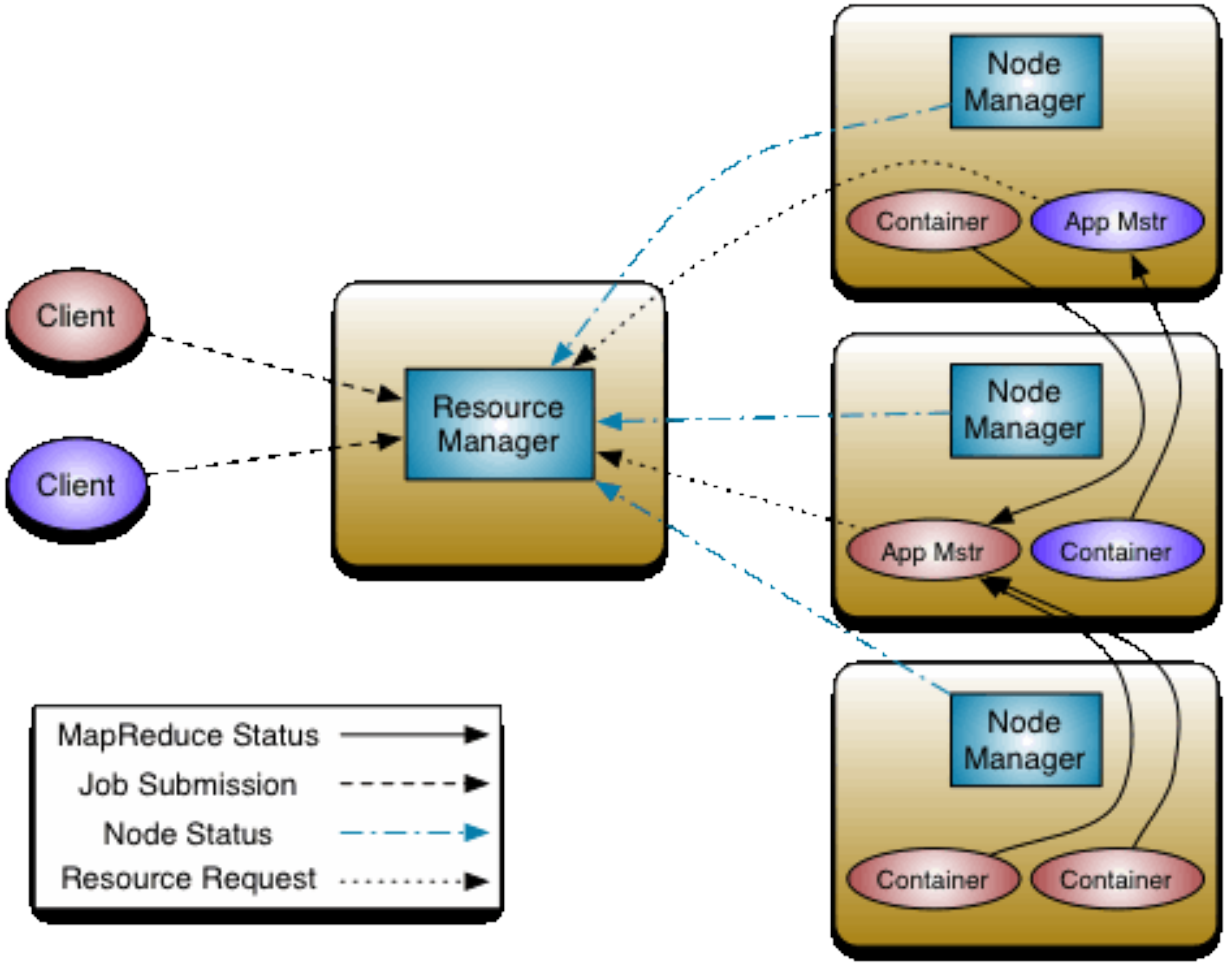
Verziók

- 18 November, 2014: Release 2.6.0 available
- 19 November, 2014: Release 2.5.2 available
- 12 September, 2014: Relase 2.5.1 available
- 11 August, 2014: Release 2.5.0 available
- 30 June, 2014: Release 2.4.1 available
- 27 June, 2014: Release 0.23.11 available
- 07 April, 2014: Release 2.4.0 available
- 20 February, 2014: Release 2.3.0 available
- 11 December, 2013: Release 0.23.10 available
- **15 October, 2013: Release 2.2.0 available**
- 23 September, 2013: Release 2.1.1-beta available
- 25 August, 2013: Release 2.1.0-beta available
- 23 August, 2013: Release 2.0.6-alpha available
- **1 Aug, 2013: Release 1.2.1 (stable) available**
- 8 July, 2013: Release 0.23.9 available
- 6 June, 2013: Release 2.0.5-alpha available
- 5 June, 2013: Release 0.23.8 available
- 13 May, 2013: Release 1.2.0 available
- 25 April, 2013: Release 2.0.4-alpha available
- 18 April, 2013: Release 0.23.7 available
- 15 February, 2013: Release 1.1.2 available
- 14 February, 2013: Release 2.0.3-alpha available

Architektúra (Hadoop 1.x)



Architektúra (Hadoop 2.x)



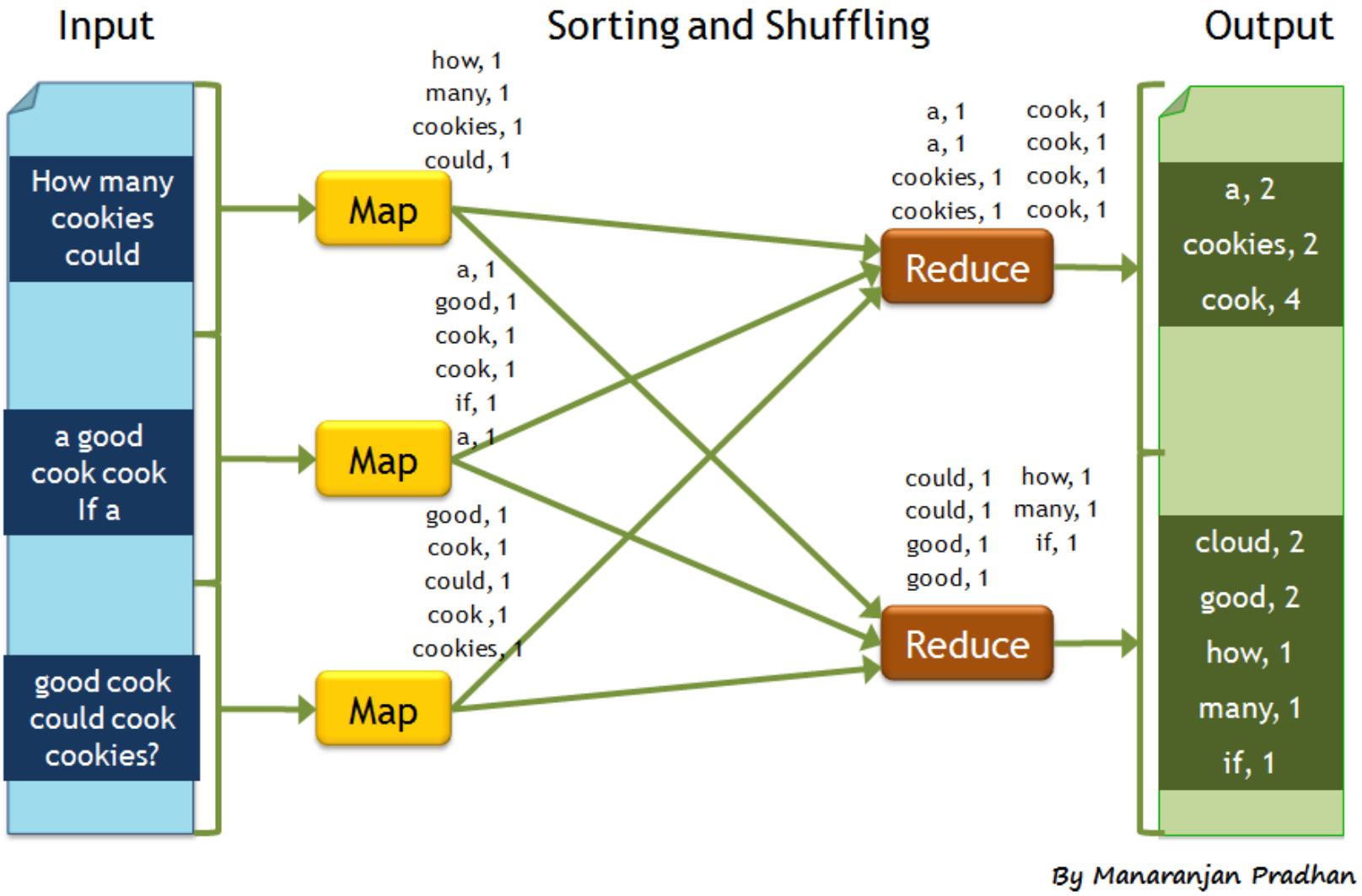
Hadoop

- Hadoop 1.0
  - MapReduce futtatás
- Hadoop 2.0
  - Yarn (Yet Another Resource Negotiator)
  - Többféle alkalmazás futtatása
- Lokalitási szabály a task futására:
  1. Ott fut ahol az adat van
  2. Abban a rackben fut ahol az adat van
  3. Bárhol

Hadoop alprojektek

- Ambari (Hadoop menedzser)
- Avro (adat formátumok támogatása)
- Cassandra (adatbázis)
- Chukwa (monitorozó)
- Hama (analizáló eszköz)
- Hbase (adatbázis)
- Hive (SQL szerű nyelv)
- Mahout (gépi tanulás csomag)
- Pig (lekérdező nyelv)
- Spark (gyors, általánosított motor elemzésekhez)
- ZooKeeper (koordinátor)

Példa


Input
Sorting and Shuffling
Output
How many cookies could
a good cook cook If a
good cook could cook cookies?
Map
Map
Map
how, 1 many, 1 cookies, 1 could, 1
a, 1 good, 1 cook, 1 cook, 1 if, 1 a, 1
good, 1 cook, 1 could, 1 cook ,1 cookies, 1
Reduce
Reduce
a, 1 a, 1 cookies, 1 cookies, 1
cook, 1 cook, 1 cook, 1 cook, 1
could, 1 could, 1 good, 1 good, 1
how, 1 many, 1 if, 1
a, 2 cookies, 2 cook, 4
cloud, 2 good, 2 how, 1 many, 1 if, 1
By Manaranjan Pradhan

További lehetőségek

- Partitioners
  - Map kulcsok hashelésének felüldefiniálása
- Combiners
  - Map kimenetének redukálása
- Compression
  - Output tömörítése
- Counters
  - Folyamatjelzők
- Zero Reduces
  - Csak a map fut, nincs sort és shuffle

MapReduce alkalmazás

- Tipikusan Java-ba íródnak.
  - Hadoop Streaming-nél lehet más nyelv.
- Futtatás:
  - Yarn jar <jar file> <main-class> <input> <output>
- Alkalmazás felépítése
  - Driver – input, output típusának definiálása, Mapper, Reducer osztály megadása
  - Mapper – Inputot dolgozza fel.
  - Reducer – A Mapper eredményét dolgozza fel.

Példa: Mapper.java

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

   private final Text globalKey = new Text(”num_pages");
   private final IntWritable bookPages = new IntWritable();

   @Override
   public void map(Object key, Text value, Context context)
   throws Exception {
      String[] fields = value.toString().split(“,”);

      if (fields[1].equals(“latin”)) {
         bookPages.set(fields[2]);
         context.write(globalKey, bookPages);
      }
   }
}
```

Példa: Reducer.java

```
public static class DemoReducer extends
Reducer<Text, IntWritable, Text, IntWritable> {

   private final IntWritable totalPages= new IntWritable();

   @Override
   public void reduce(Text globalKey, Iterable<IntWritable>
   bookPages, Context context) throws Exception {
      int sum = 0;

      for (IntWritable val: bookPages) {
         sum += val.get();
      } // for

      totalPages.set(sum);
      context.write(globalKey, totalPages);
   } // reduce

}
```

Péda: DemoDriver.class

```
public class DemoDriver extends Configured implements Tool {
    public static void main(String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new
                  DemoDriver(), args);
        System.exit(res);
    }
    @Override
    public int run(String[] args) throws Exception {
            Configuration conf = this.getConf();
            Job job = Job.getInstance(conf, "julius caesar");
            job.setJarByClass(DemoDriver.class);
            job.setMapperClass(DemoMapper.class);
            job.setCombinerClass(DemoReducer.class);
            job.setReducerClass(DemoReducer.class);
            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(IntWritable.class);
            FileInputFormat.addInputPath(job, new Path(args[0]));
            FileOutputFormat.setOutputPath(job, new Path(args[1]));
            return job.waitForCompletion(true) ? 0 : 1;
    }
}
```

Köszönöm a figyelmet!