Distributed Computing the Google way

An introduction to Apache Hadoop

3 million images
are uploaded to flickr everyday.

…enough images to fill a
375.000 page photo album.

Over **210 billion emails**
are sent out daily.

...which is **more than a year's worth**
of letter mail in the US.

Bloggers post **900.000** new articles every day.

2. Spiel
nicht belohnt
Werder Bremen ist im Acht
Republik Kosovo jetzt zum
für eine eigene Hymne aufg
Superstar Veltroni
Roms ehemaliger Bürg

Enough posts to fill the New York Times for 19 years!

43.339 TB are sent across all mobile phones globally everyday.

That is enough to fill...

1.7 million

Blu-rays

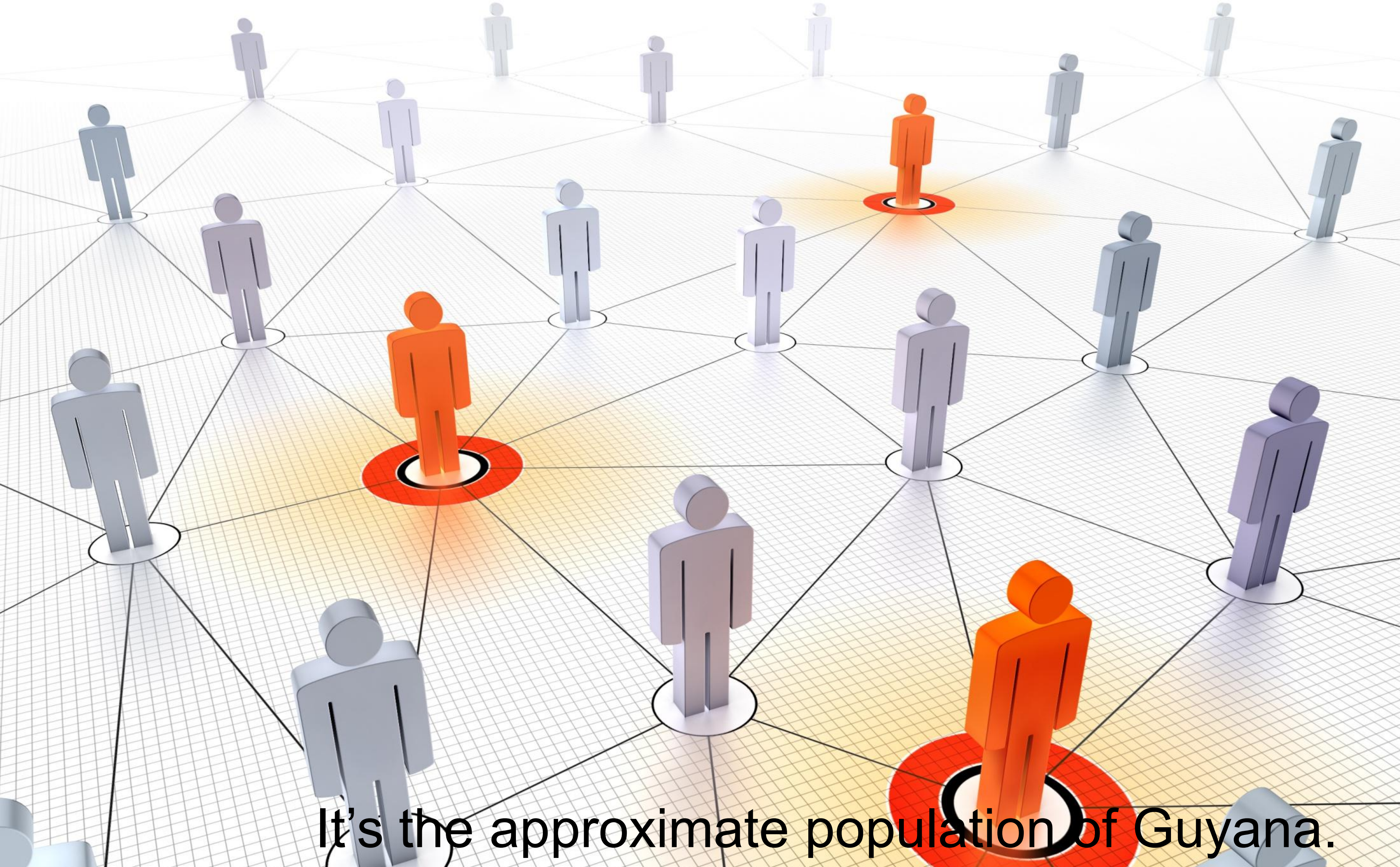
9.2 million

DVDs

63.9 trillion

3.5” diskettes

700.000 new members are signing up on Facebook everyday.



It’s the approximate population of Guyana.

Welcome!

Agenda

1 Introduction.

2 MapReduce.

3 Apache Hadoop.

4 RDBMS & MapReduce.

5 Questions & Discussion.

Eduard Hildebrandt

Solution Architect

+49 160 6307253

mail@eduard-hildebrandt.de

http://www.eduard-hildebrandt.de

Heinrich Freiherr von Schwerin

Consultant

+49 160 90608304

http://www.logica.com/de

Heinrich.freiherr.von.schwerin@logica.com

Did you know that Logica serves the 8 biggest business sectors, including Pharmaceuticals and Chemicals?

logica

be brilliant together

Did you know that Logica is a perfect partner when talking about outsourcing?

Why should I care?

It’s not just Google!

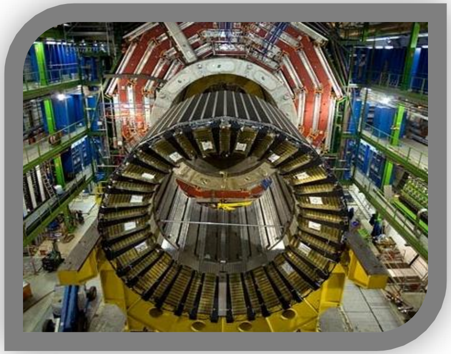
New York
Stock Exchange

1 TB trade data per day

Internet Archive
www.archive.org

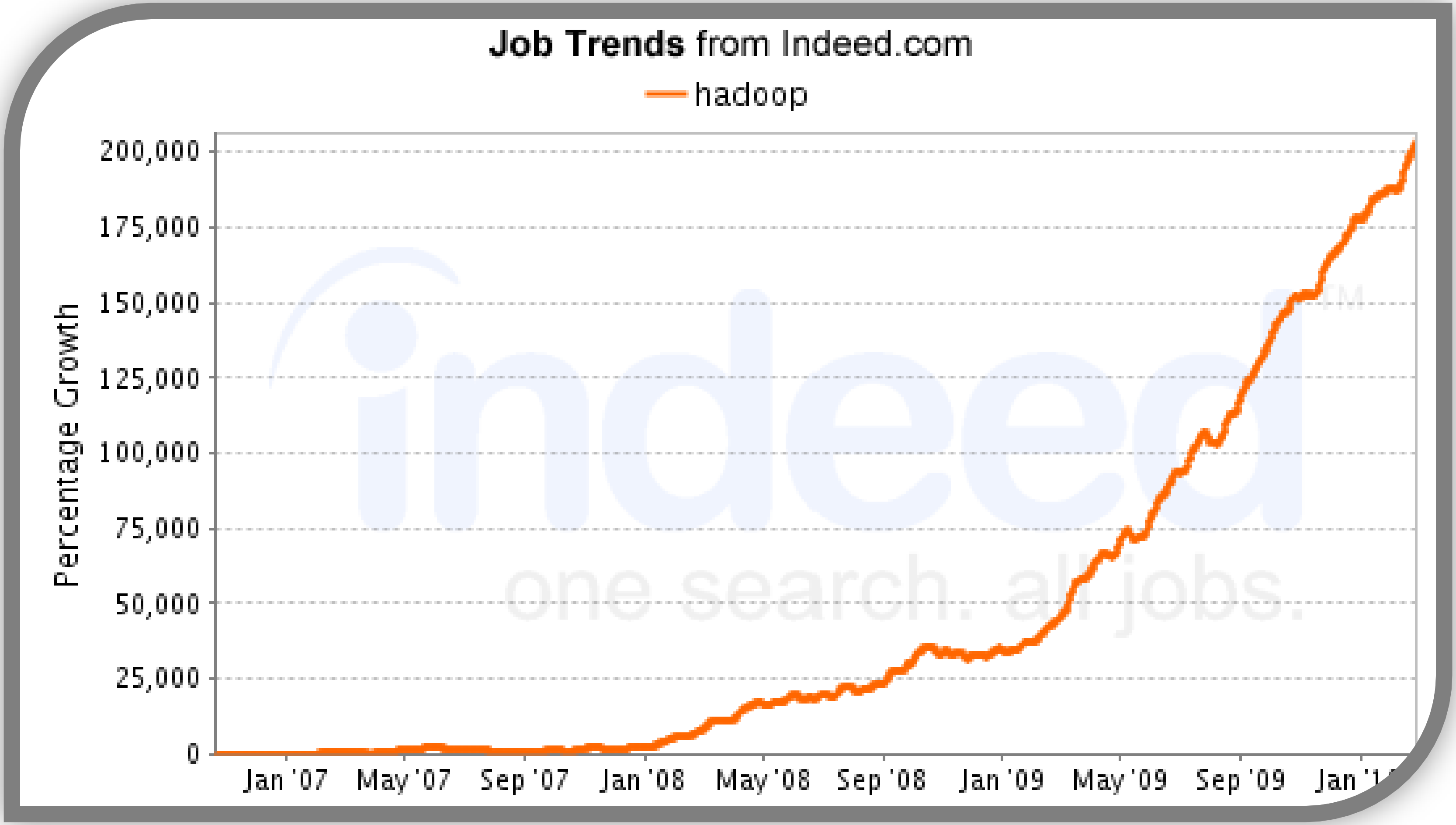
growing by 20 TB per month

Hadron Collider
Switzerland



producing 15 PB per year

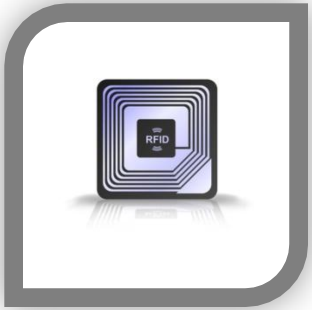
It’s a growing job market!



It may be the future of distributed computing!

Think about…

GPS tracker



RFID

genom analysis

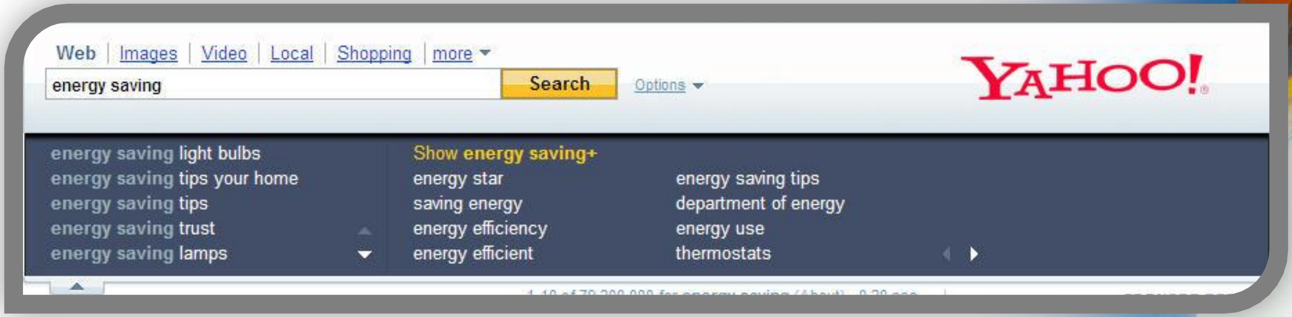
medical monitors

The amount of data we produce will rise from year to year!

It's about performance!

BEFORE

Development: 2-3 Weeks
Runtime: 26 days



energy saving

energy saving light bulbs
energy saving tips your home
energy saving tips
energy saving trust
energy saving lamps

Show energy saving+
energy star
saving energy
energy efficiency
energy efficient

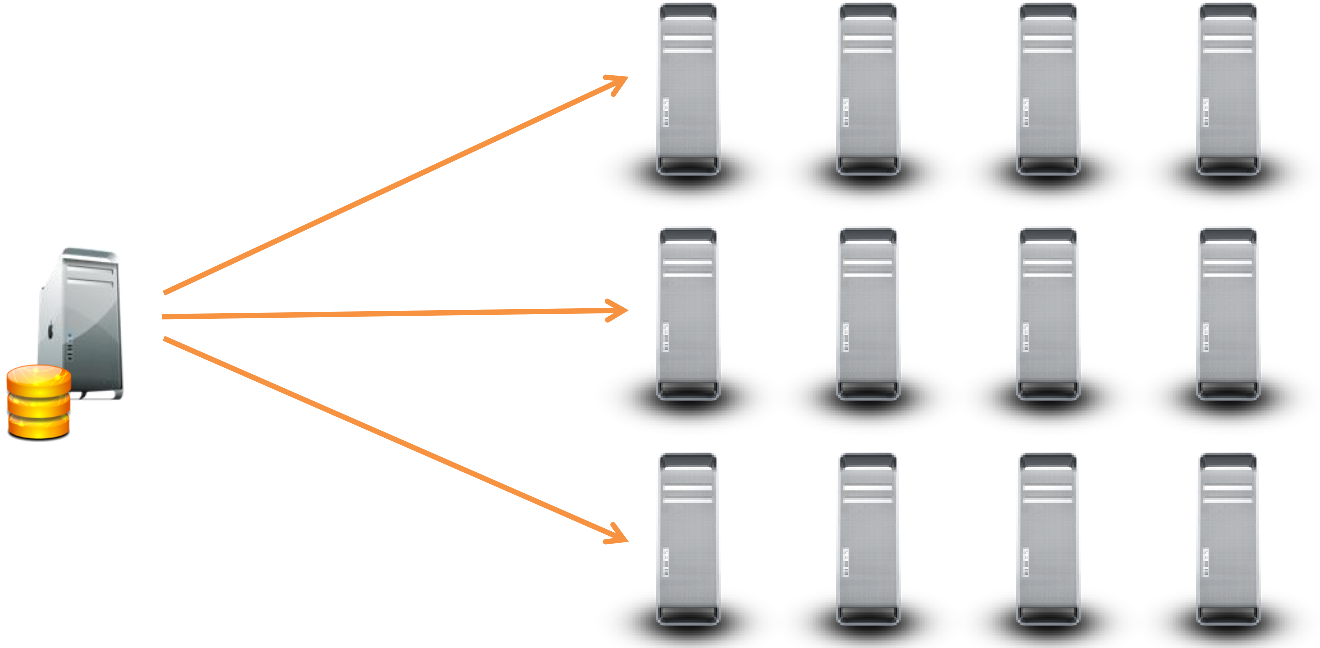
energy saving tips
department of energy
energy use
thermostats

AFTER

Development: 2-3 Days
Runtime: 20 minutes

Grid computing

focus on: distributing **workload**



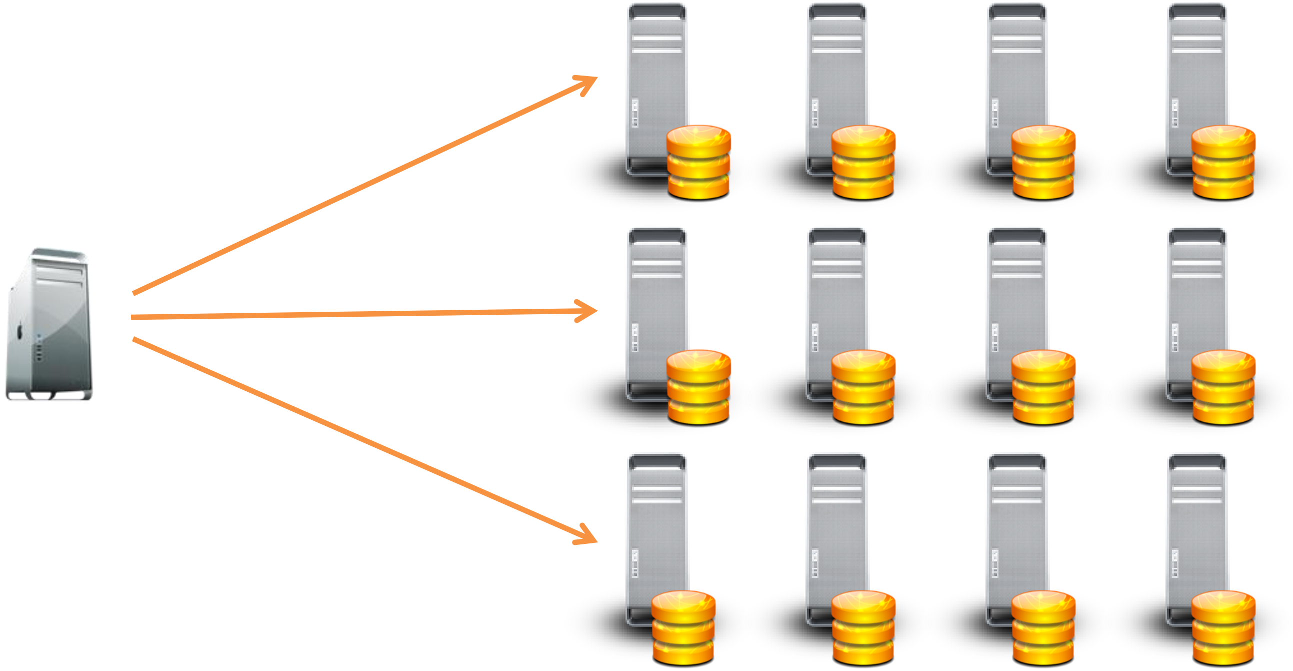
- one SAN drive, many compute nodes
- works well for small data sets and long processing time
- examples: SETI@home, Folding@home

Problem: Sharing data is slow!

Google processed 400 PB per month in 2007 with an average job size of 180 GB. It takes ~ 45 minutes to read a 180 GB file sequentially.

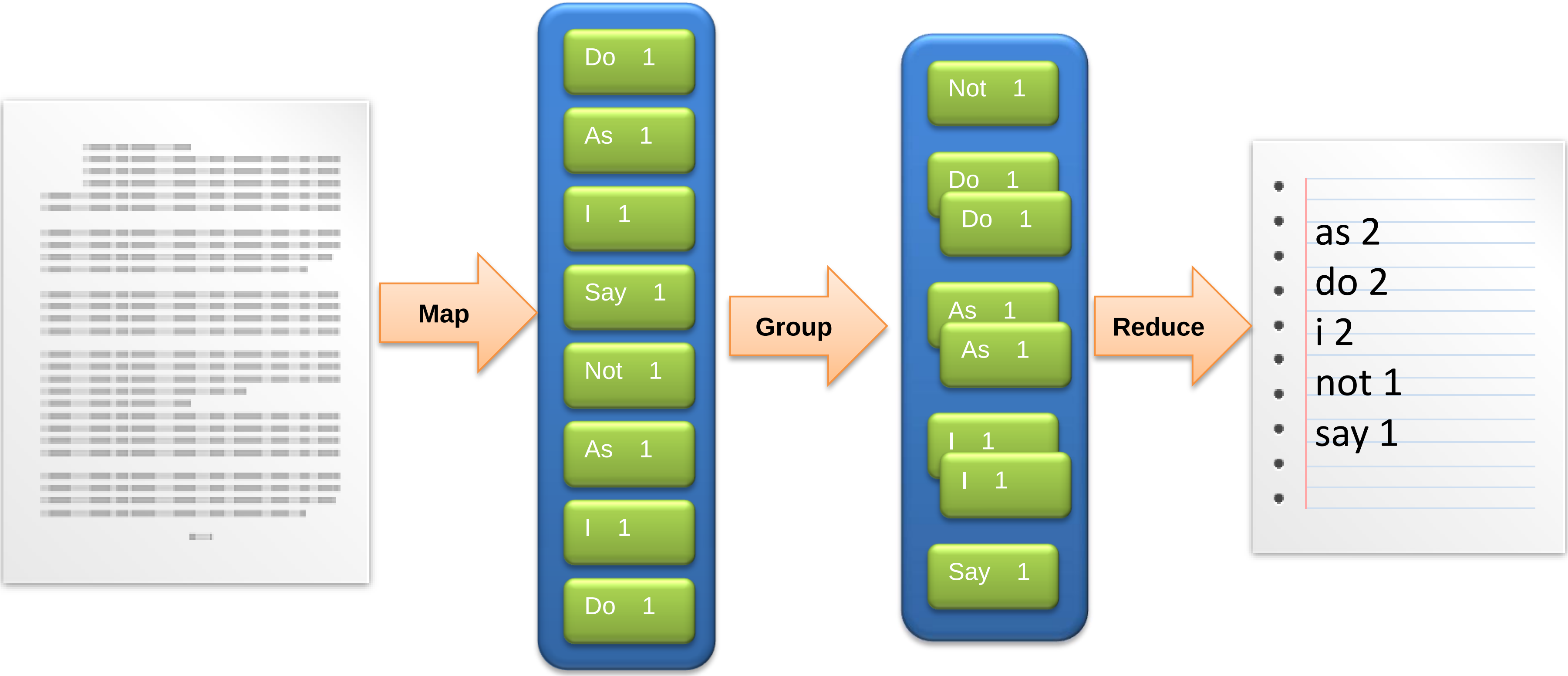
Modern approach

focus on: distributing the **data**



- stores data locally
- parallel read / write
  - 1 HDD → ~75 MB/sec
  - 1000 HDD → ~75000 MB/sec

The MAP and REDUCE algorithm



Map

Do 1

As 1

I 1

Say 1

Not 1

As 1

I 1

Do 1

Group

Not 1

Do 1

Do 1

As 1

As 1

I 1

I 1

Say 1

Reduce

as 2
do 2
i 2
not 1
say 1

It‘s really map – group – reduce!

Implementation of the MAP algorithm

```
public static class MapClass extends MapReduceBase implements
        Mapper<LongWritable, Text, Text, IntWritable> {

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

    public void map(LongWritable key, Text value,
        OutputCollector<Text, IntWritable> output, Reporter reporter)
        throws IOException {

        String line = value.toString();
        StringTokenizer itr = new StringTokenizer(line);
        while (itr.hasMoreTokens()) {
            word.set(itr.nextToken());
            output.collect(word, one);
        }
    }
}
```

Could it be even simpler?

Implementation of the REDUCE algorithm

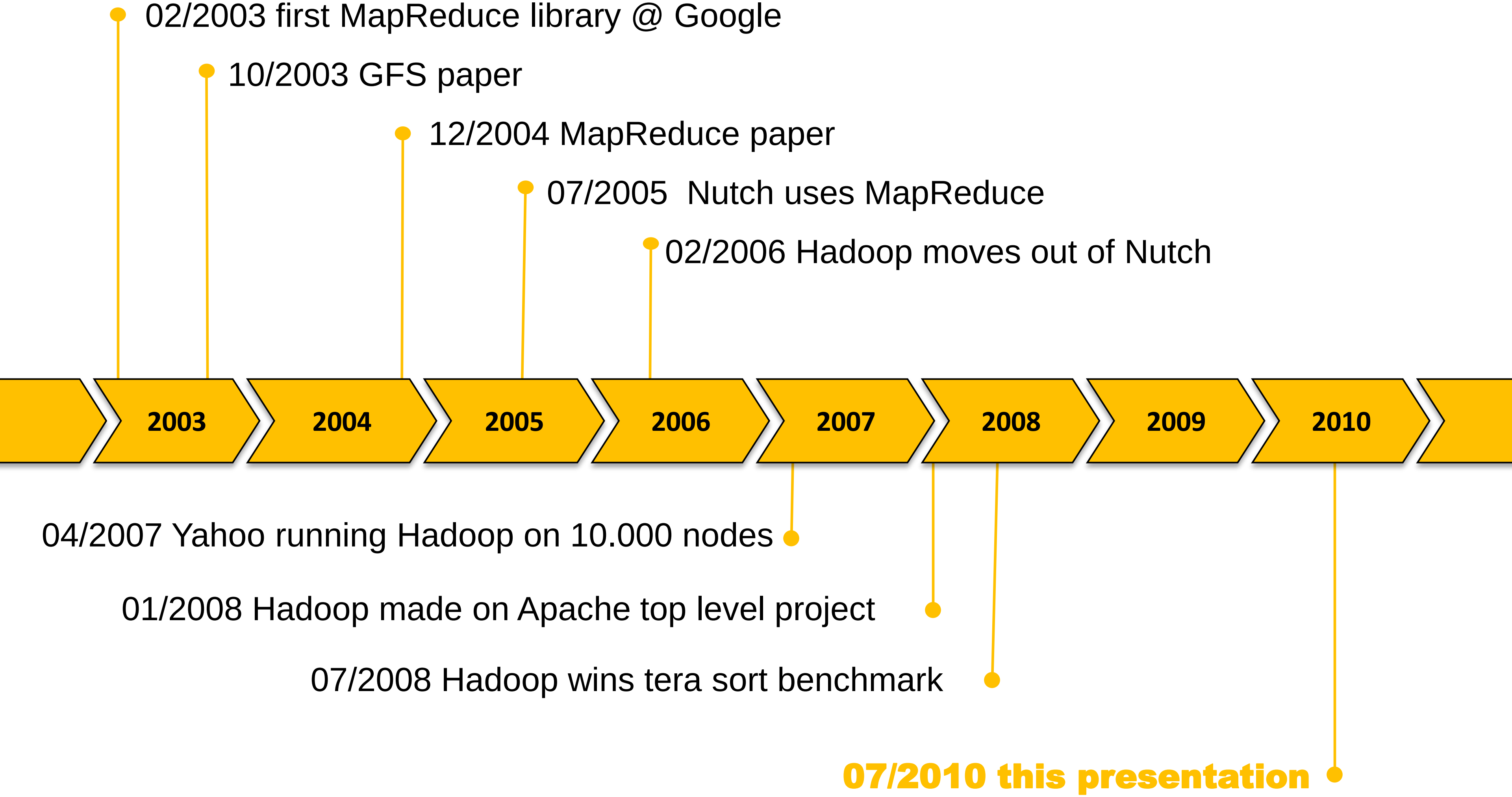
```
public static class Reduce extends MapReduceBase implements
        Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text key, Iterator<IntWritable>      values,
        OutputCollector<Text, IntWritable> output, Reporter_reporter)
        throws IOException {

        int sum = 0;
        while (values.hasNext()) {
            sum += values.next().get();
        }
        output.collect(key, new IntWritable(sum));
    }
}
```

Just REDUCE it!

Apache Hadoop

Hadoop is an open-source Java framework for parallel processing **large data** on clusters of **commodity hardware**.

Hadoop History



02/2003 first MapReduce library @ Google

10/2003 GFS paper

12/2004 MapReduce paper

07/2005 Nutch uses MapReduce

02/2006 Hadoop moves out of Nutch

04/2007 Yahoo running Hadoop on 10.000 nodes

01/2008 Hadoop made on Apache top level project

07/2008 Hadoop wins tera sort benchmark

07/2010 this presentation

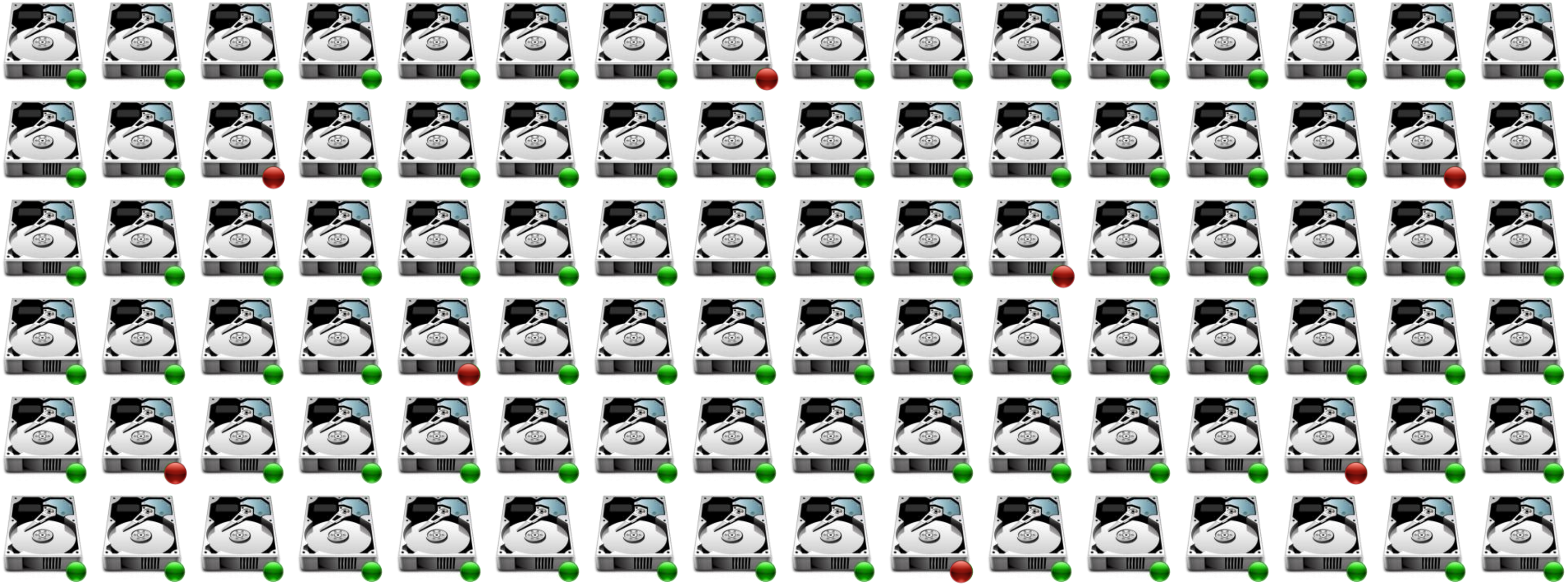
Who is using Apache Hadoop?

New York Times

“Failure is the defining difference between distributed and local programming.”

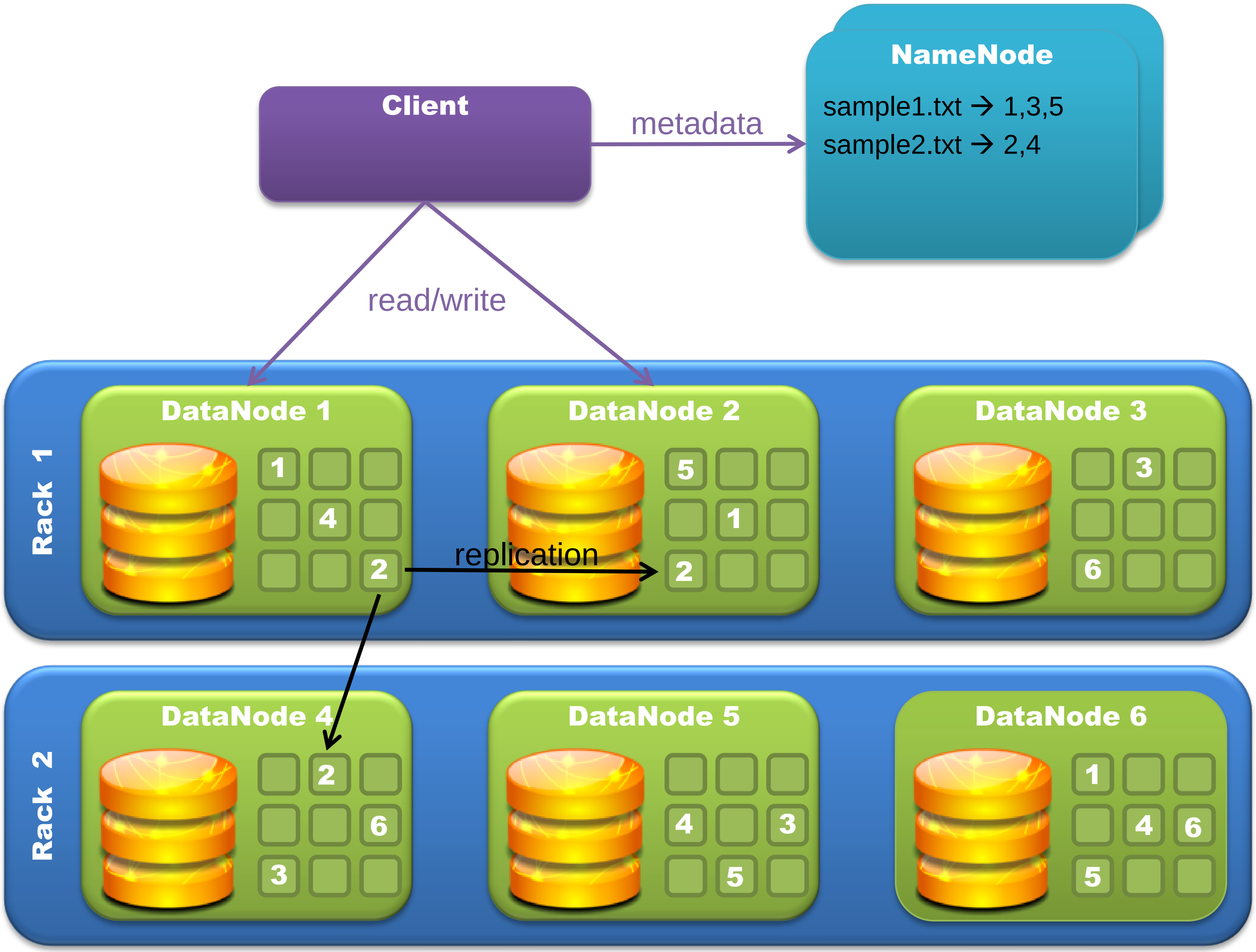
-- Ken Arnold, CORBA designer

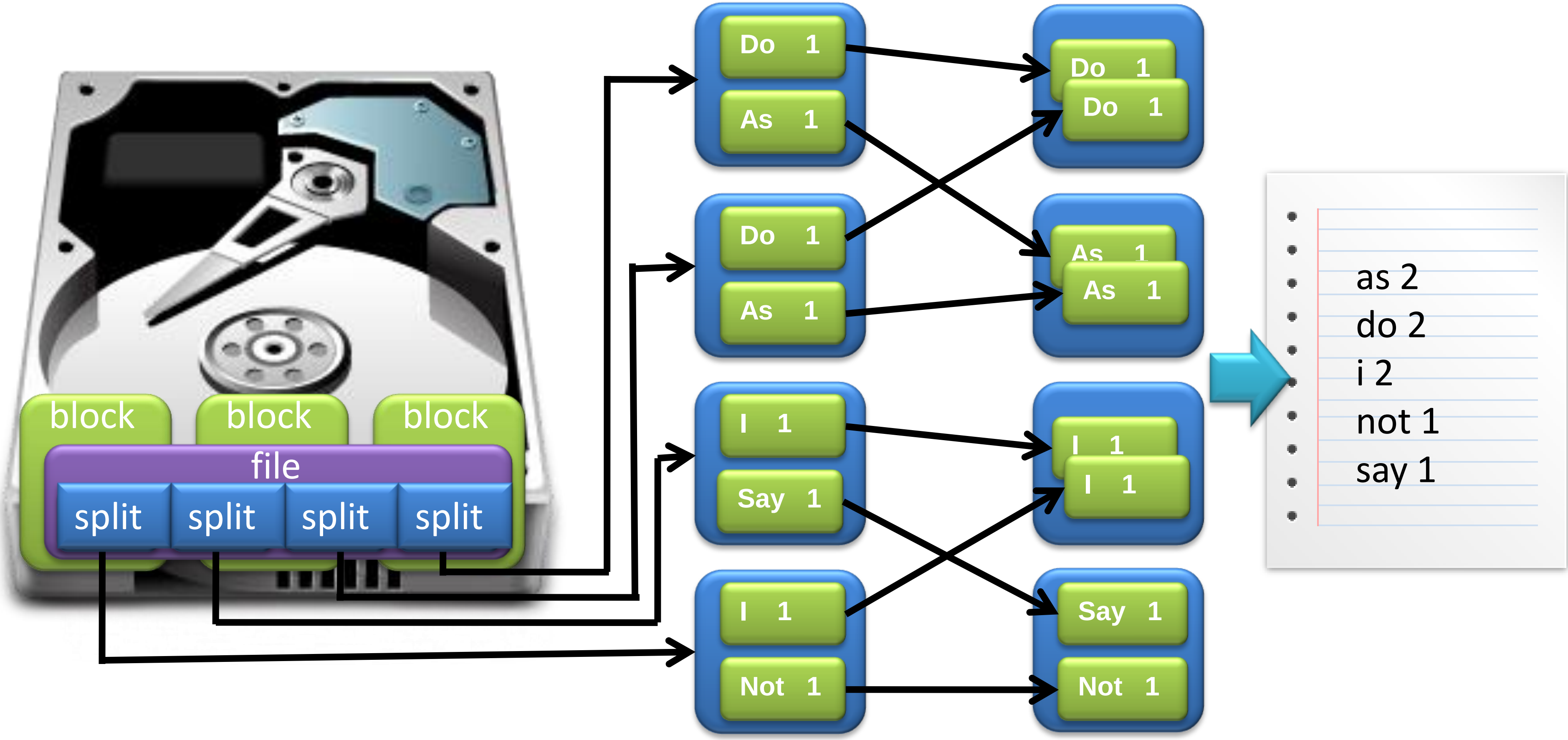
mean time between failures of a HDD: 1.200.000 hours



If your cluster has 10.000 hard drives,
then you have **a hard drive crash every 5 days** on average.

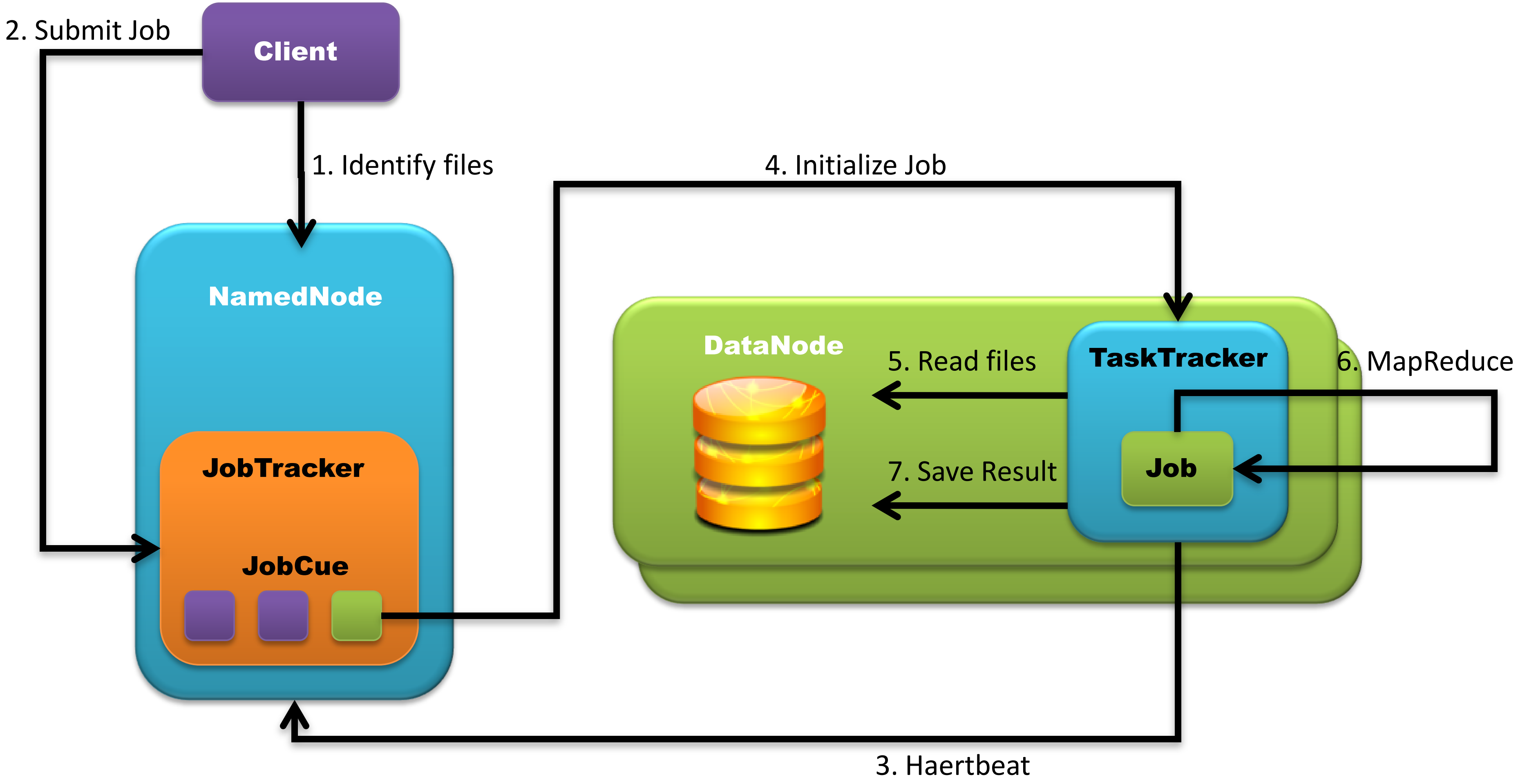
HDFS


NameNode
sample1.txt → 1,3,5
sample2.txt → 2,4
Client
metadata
read/write
Rack 1
DataNode 1
1
4
2
DataNode 2
5
1
2
replication
DataNode 3
3
6
Rack 2
DataNode 4
2
6
3
DataNode 5
4
3
5
DataNode 6
1
4
6
5

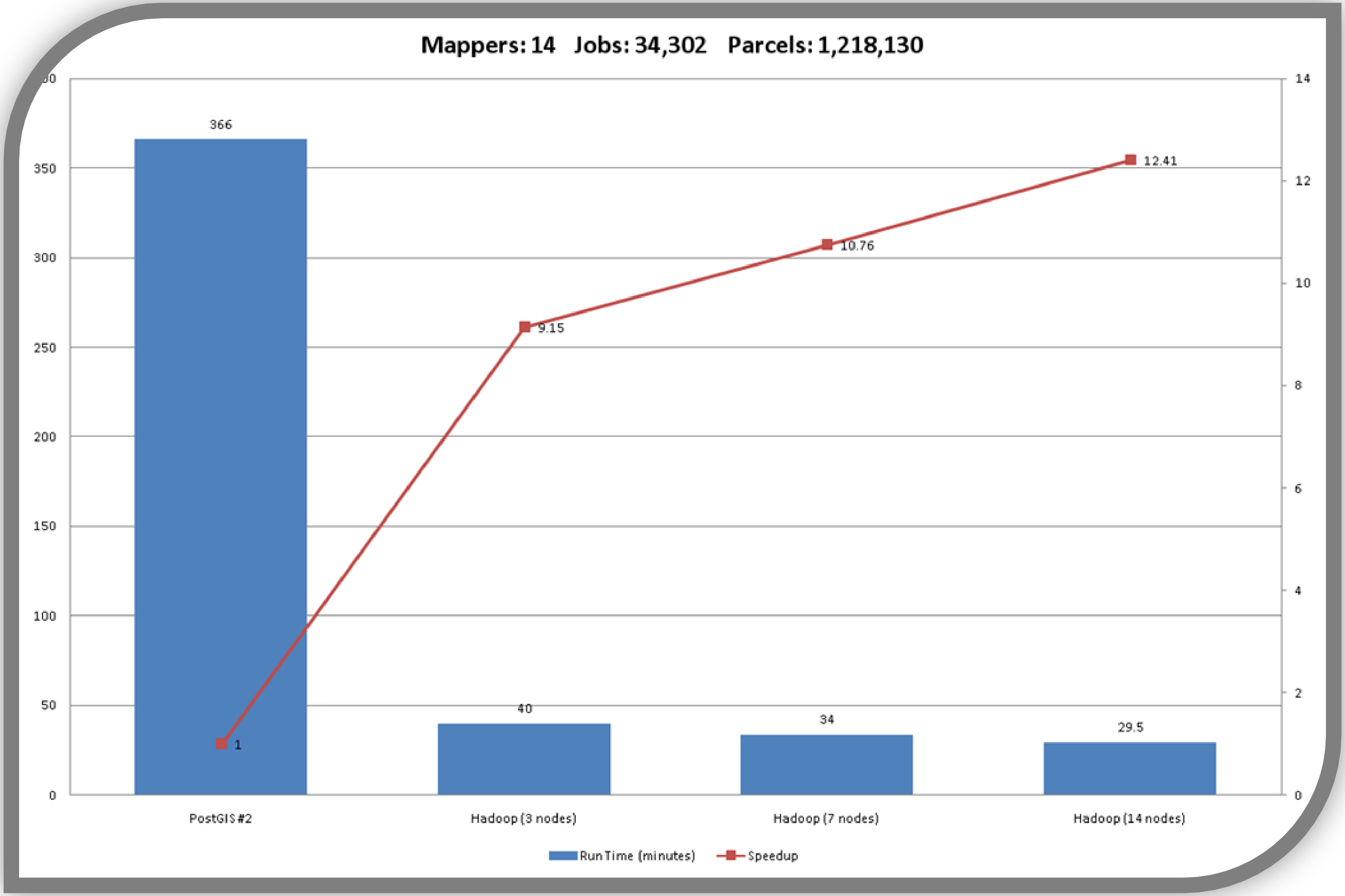


How does it fit together?

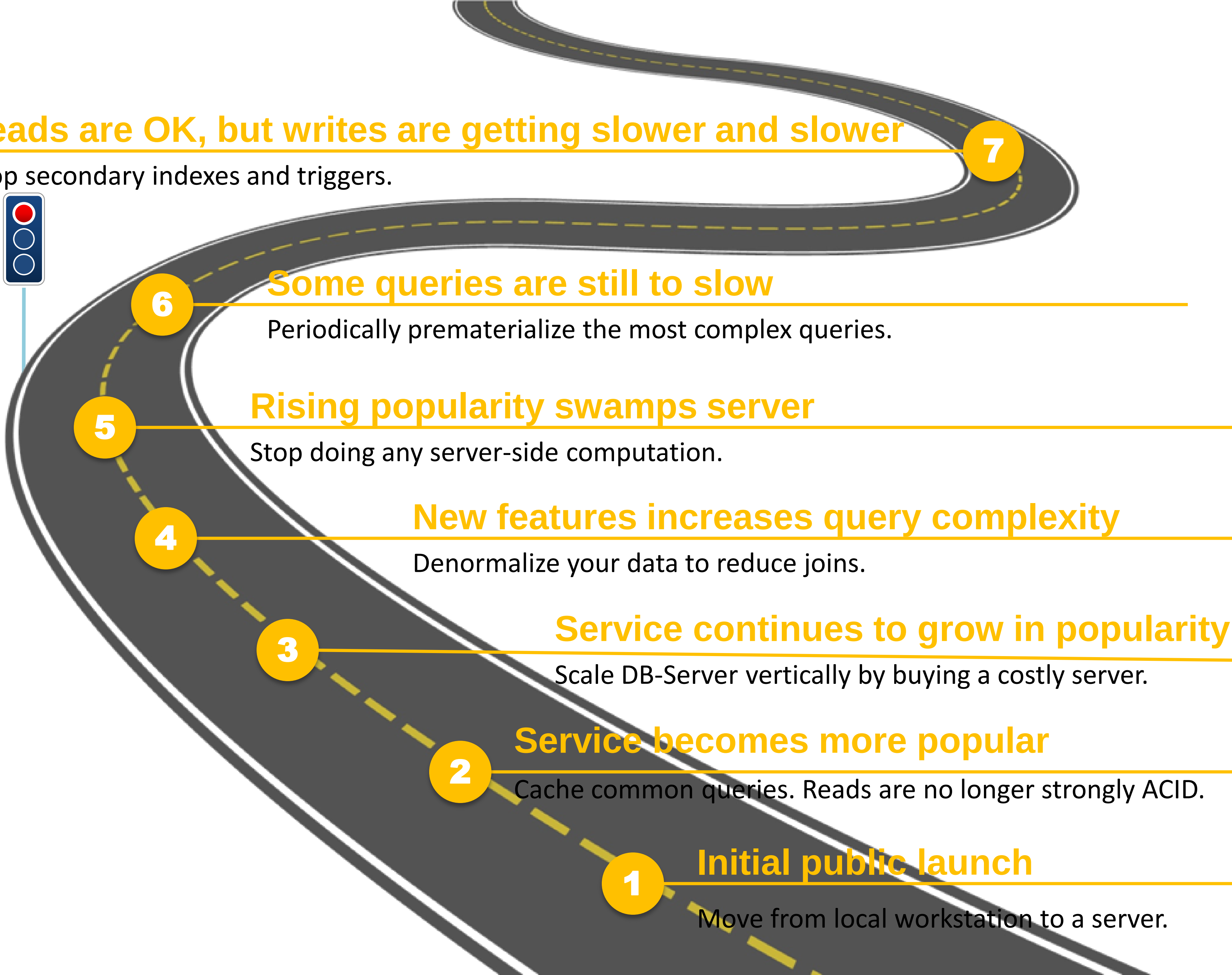
Hadoop architecture



Reduce it to the max!



Performance **improvement** when **scaling** with your hadoop system



Reads are OK, but writes are getting slower and slower

Drop secondary indexes and triggers.

Some queries are still to slow

Periodically prematerialize the most complex queries.

Rising popularity swamps server

Stop doing any server-side computation.

New features increases query complexity

Denormalize your data to reduce joins.

Service continues to grow in popularity

Scale DB-Server vertically by buying a costly server.

Service becomes more popular

Cache common queries. Reads are no longer strongly ACID.

Initial public launch

Move from local workstation to a server.

How can we solve this scaling problem?

Join

page_view

pageid	userid	time
1	111	10:18:21
2	111	10:19:53
1	222	11:05:12

X

user

userid	age	gender
111	22	female
222	33	male

=

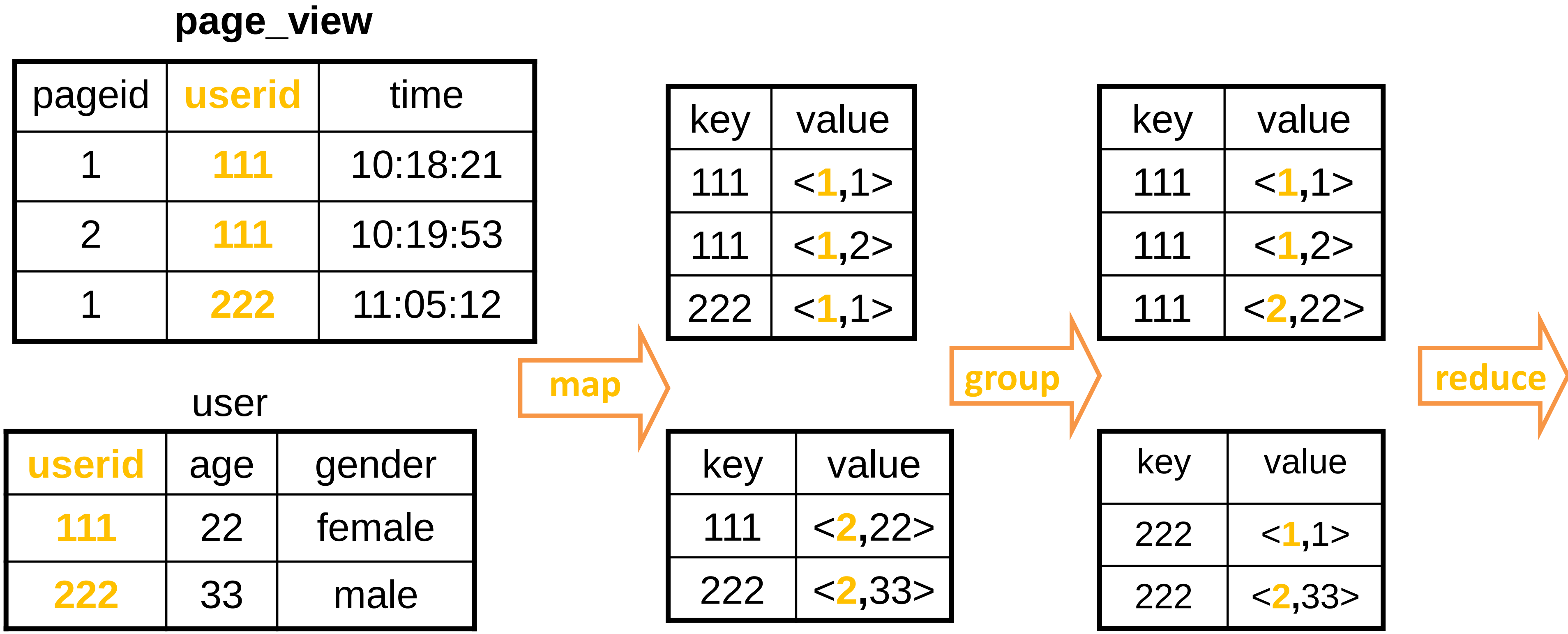
pv_users

pageid	age
1	22
2	22
1	33

SQL:

```
    INSERT INTO TABLE pv_users
    SELECT pv.pageid, u.age
    FROM page_view pv JOIN user u ON (pv.userid = u.userid);
```

Join with MapReduce



page_view

pageid	userid	time
1	111	10:18:21
2	111	10:19:53
1	222	11:05:12

user

userid	age	gender
111	22	female
222	33	male

key	value
111	<1,1>
111	<1,2>
222	<1,1>

key	value
111	<2,22>
222	<2,33>

key	value
111	<1,1>
111	<1,2>
111	<2,22>

key	value
222	<1,1>
222	<2,33>

HBase

HBase is an open-source, **distributed**, versioned, column-oriented **store** modeled after Google' Bigtable.

- **No real indexes**
- **Automatic partitioning**
- **Scale linearly and automatically with new nodes**
- **Commodity hardware**
- **Fault tolerant**
- **Batch processing**

RDBMS vs. MapReduce

	RDBMS	MapReduce
Data size	gigabytes	petabytes
Access	interactive and batch	batch
Updates	read and write many times	write once read many times
Structure	static schema	dynamic schema
Integrity	high	low
Scaling	nonlinear	linear

Use the right tool!

MapReduce is a screwdriver.

good for:

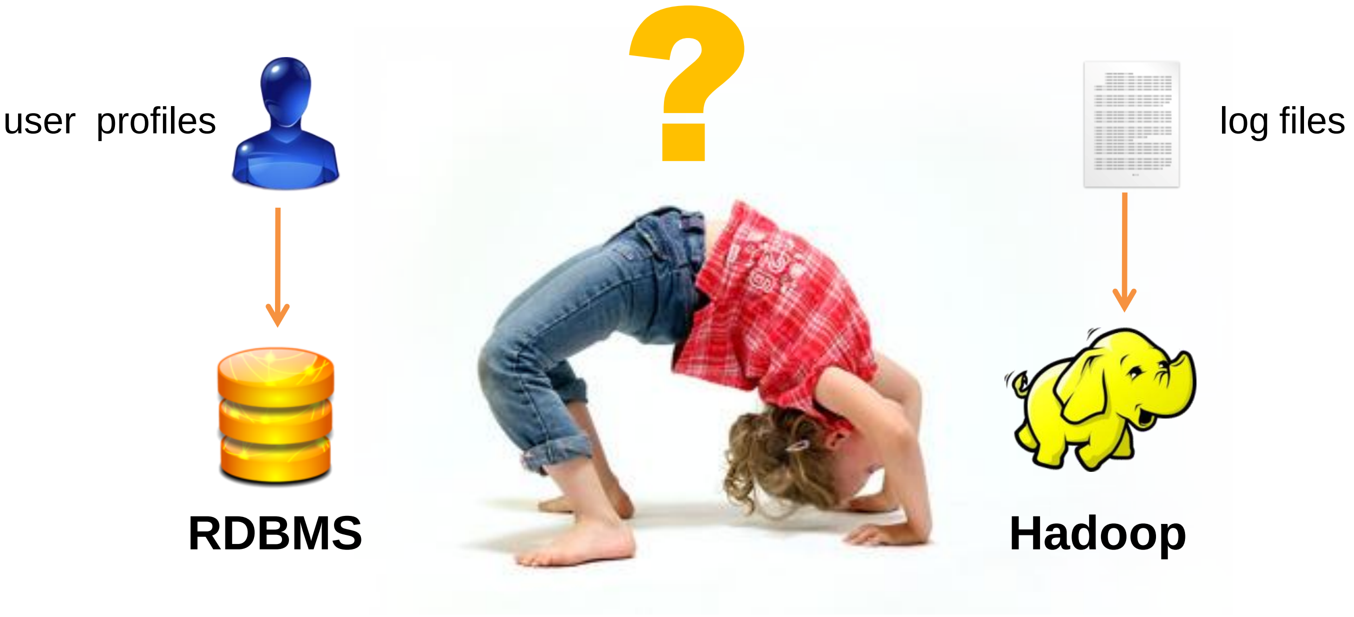
- unstructured data
- data intensive computation
- batch operations
- scale horizontal

good for:

- structured data
- transactions
- interactive requests
- scale vertically

Databases are hammers.

Where is the bridge?



user profiles

RDBMS

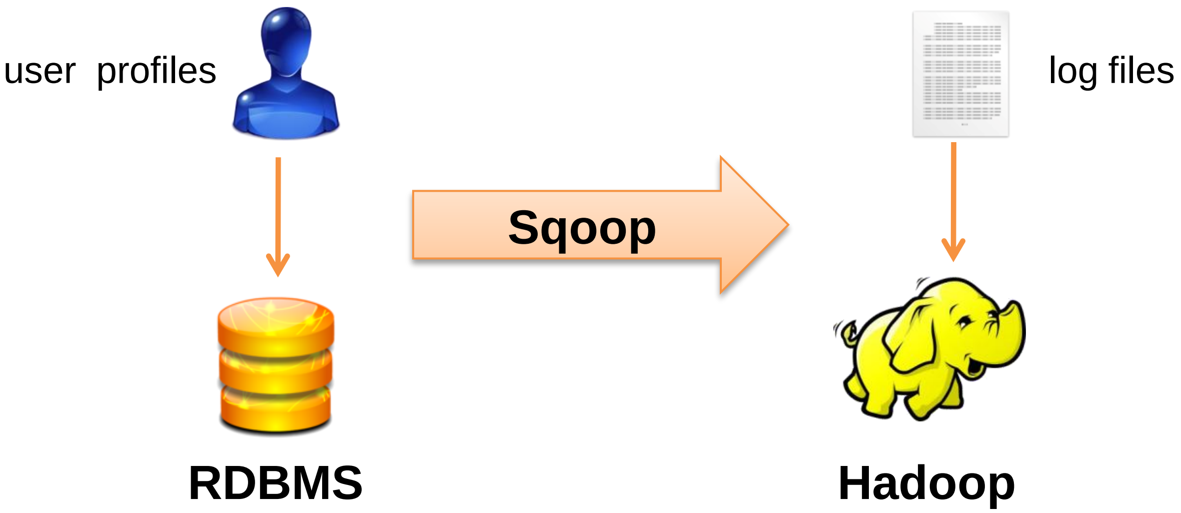
?

log files

Hadoop

Sqoop

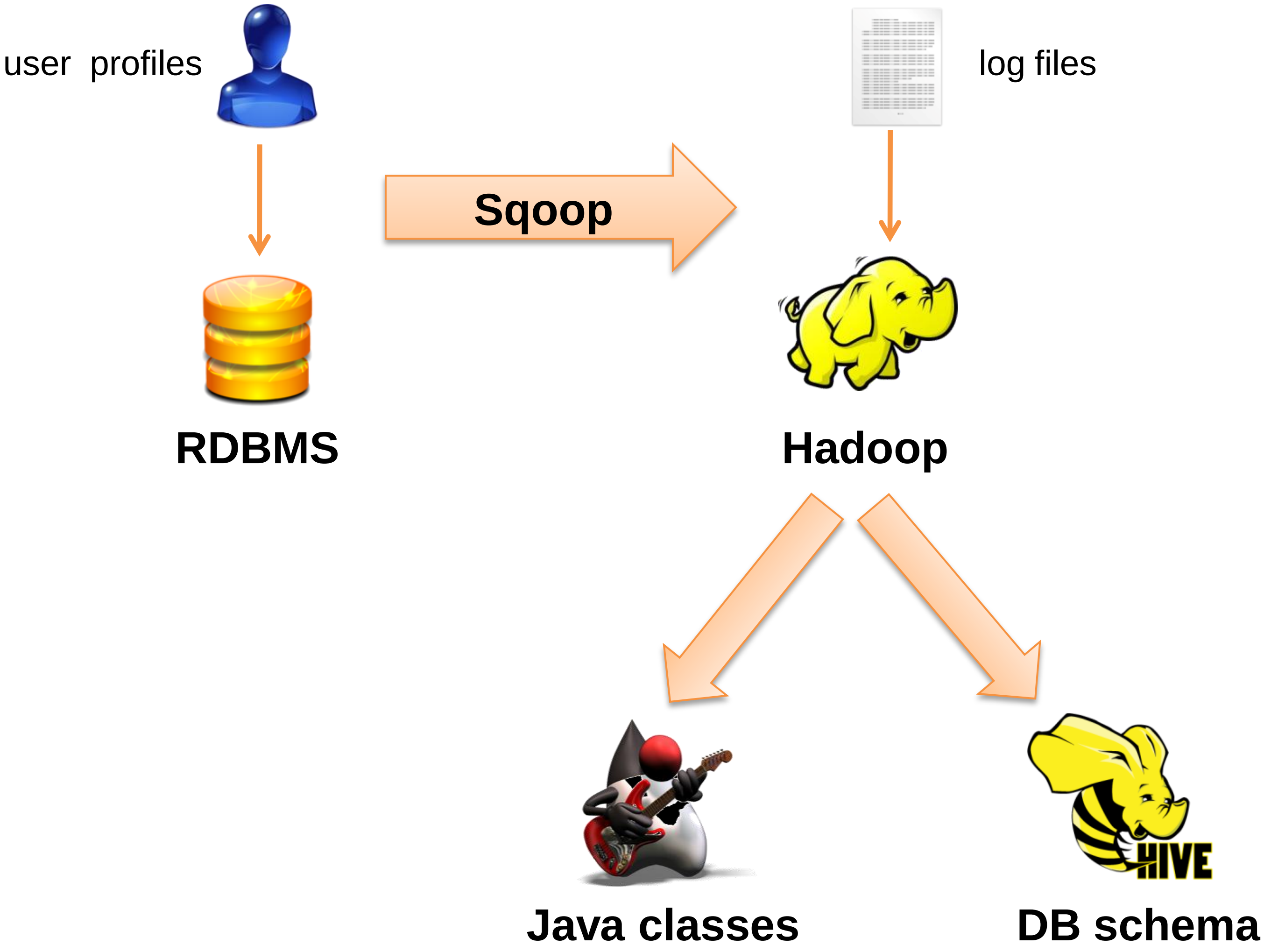
SQL-to-Hadoop **database import** tool



```
$ sqoop –connect jdbc:mysql://database.example.com/users \
  –username aaron –password 12345 –all-tables \
  –warehouse-dir /common/warehouse
```

Sqoop

SQL-to-Hadoop **database import** tool



user profiles

log files

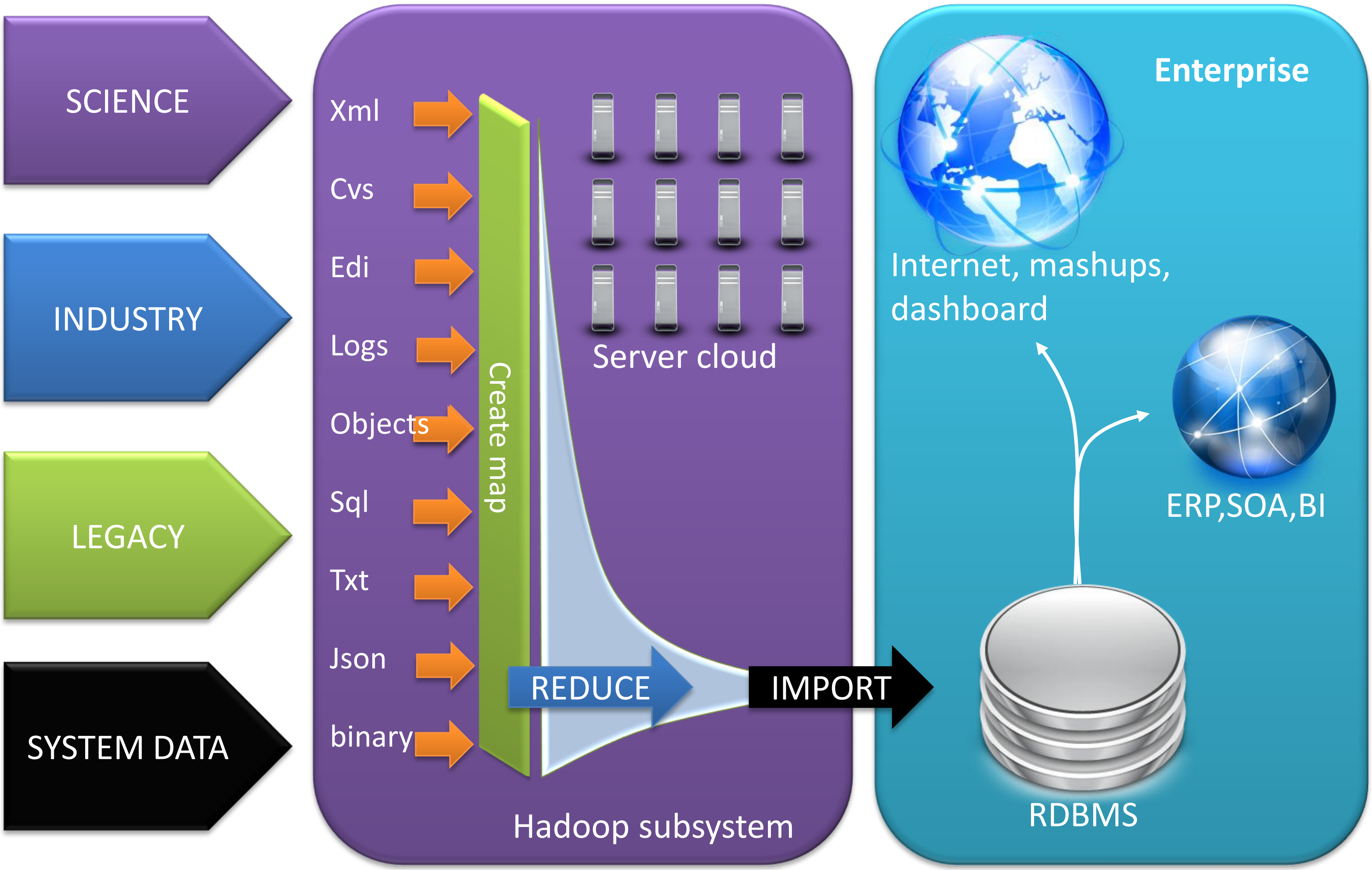
Sqoop

RDBMS

Hadoop

Java classes

DB schema



SCIENCE

INDUSTRY

LEGACY

SYSTEM DATA

Xml

Cvs

Edi

Logs

Objects

Sql

Txt

Json

binary

Create map

Server cloud

REDUCE

IMPORT

Hadoop subsystem

Enterprise

Internet, mashups, dashboard

ERP,SOA,BI

RDBMS

1 High Volume Data

2 MapReduce algorithm

3 Consume results

“Appetite comes with eating.”

-- François Rabelais

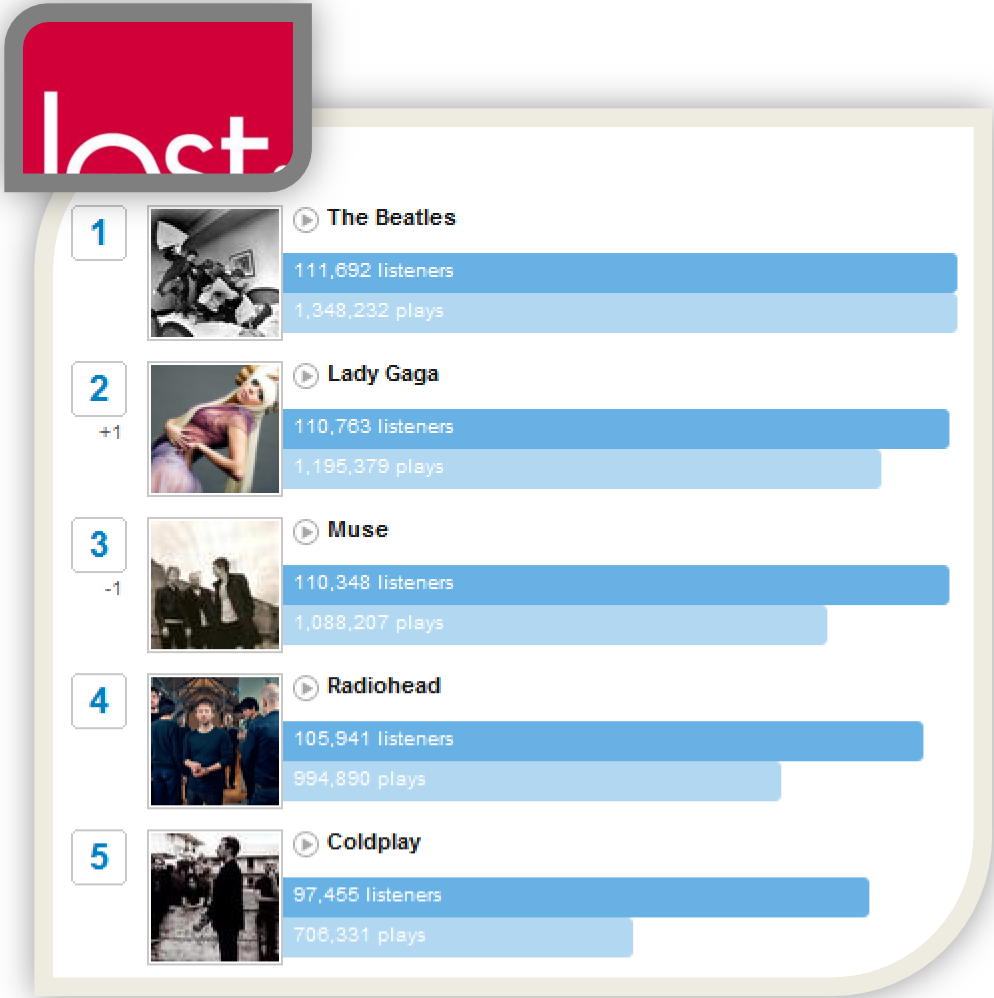
Case Study 1

Listening data:

user id	track id	scrobble	radio	skip
123	456	0	1	1
451	789	1	0	1
241	234	0	1	1

Hadoop jobs for:

- number of unique listeners
- number of times the track was:
    - scrobbled
    - listened to on the radio
    - listened to in total
    - skipped on the radio



1 The Beatles 111,692 listeners 1,348,232 plays

2 +1 Lady Gaga 110,763 listeners 1,195,379 plays

3 -1 Muse 110,348 listeners 1,088,207 plays

4 Radiohead 105,941 listeners 994,890 plays

5 Coldplay 97,455 listeners 706,331 plays

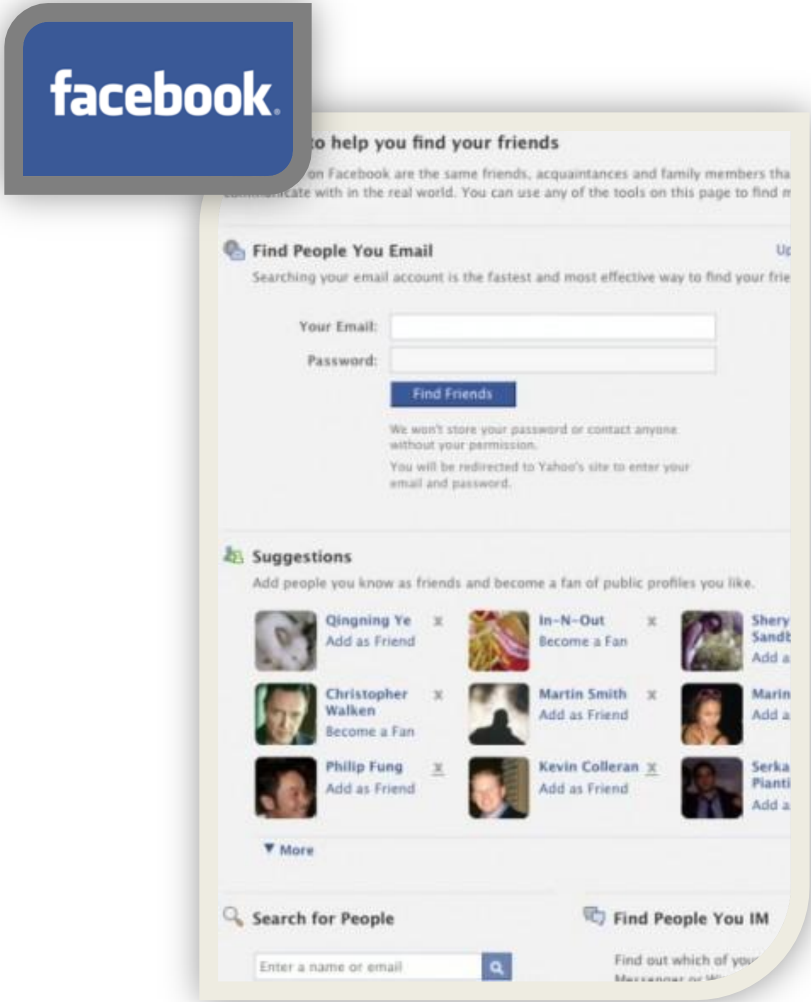
Case Study 2

User data:

- 12 TB of compressed data added per day
- 800 TB of compressed data scanned per day
- 25,000 map-reduce jobs per day
- 65 millions files in HDFS
- 30,000 simultaneous clients to the HDFS NameNode

Hadoop jobs for:

- friend recommendations
- Insights for the Facebook Advertisers



o help you find your friends

on Facebook are the same friends, acquaintances and family members tha
cate with in the real world. You can use any of the tools on this page to find m

Find People You Email

Searching your email account is the fastest and most effective way to find your frie

Your Email:

Password:

We won't store your password or contact anyone without your permission.

You will be redirected to Yahoo's site to enter your email and password.

Suggestions

Add people you know as friends and become a fan of public profiles you like.

Search for People

Find People You IM

Find out which of you

That was just the tip of the iceberg!

Core

HDFS

HBase

Hive

Thrift

Pig

Avro

Nutch

ZooKeeper

Chukwa

Jaql

Cassandra

Dumbo

Solr

KosmosFS

Cascading

Mahout

Scribe

Ganglia

Katta

Hypertable

Hadoop is a good choice for:

- analyzing log files
- sort a large amount of data
- search engines
- contextual adds
- image analysis
- protein folding
- classification

Hadoop is a **poor choice** for:

- figuring PI to 1.000.000 digits
- calculating Fibonacci sequences
- a general RDBMS replacement

Final thoughts

1. Data intensive computation is a fundamentally different challenge than doing CPU intensive computation over small dataset.
2. New ways of thinking about problems are needed.
3. Failure is acceptable and inevitable.
   Go cheap! Go distributed!
4. RDBMS is not dead!
   It just got new friends and helpers.
5. Give Hadoop a chance!

Time for questions!

Let’s get connected!

XING LinkedIn

http://www.soa-at-work.com

Thank you

Eduard Hildebrandt, Heinrich Freiherr von Schwerin

logica

be brilliant together