# Mining useful data

## out of the Internet

Ing. Jan Lukavský – Development Team Leader

jan.lukavsky@firma.seznam.cz

SEZNAM.CZ

# About this presentation

- What is "Big Data"?
- How to handle it?
- Why bother?
- How to get "valuable" Big Data from the Internet?

# What is Big Data?

- @DEVOPS_BORAT
  - *"Big Data is any thing which is crash Excel."*
- http://wikibon.org/blog/big-data-statistics/
  - 2.7 ZiB (~ $10^{21}$ bytes, $10^{9}$ terabytes) by 2012 existed in digital universe
- http://www.sciencedaily.com/releases/2013/05/130522085217.htm
  - 90% of world's data was generated over last two years

# Bbbut, what it REALLY is?

- **more or less everything**
- web pages, web graph
- photos, videos
- tweets, likes, social media
- emails
- user data
  - click streams
  - location data
  - page views

# Where does it come from?

- historically, most data was stored in **structured format**
- "structured" means everything has to be converted to the storage format, *before* being stored
  - this concept is called **schema on write**
  - if data doesn't have a format that fits this schema, it cannot be stored, or the schema has to change (which is pretty costly)
- Big Data is commonly referred to as "unstructured data"
- this is just a different name for concept called **schema on read**
  - the schema is applied to the data during reading the data from disk
  - if the data change schema, all is needed is to change the way we interpret it on read, which is very cheap

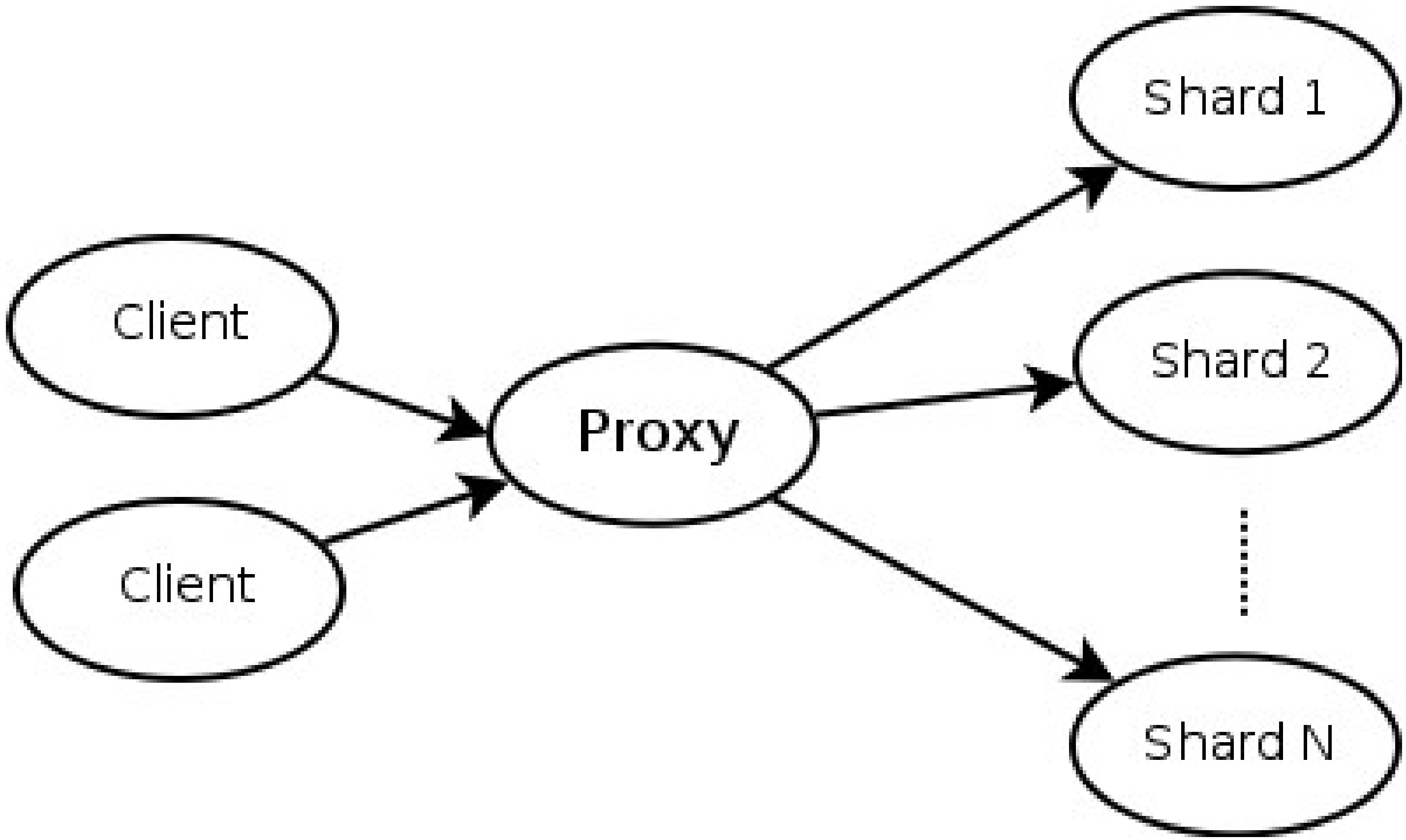
# Where does it come from?

- if it is cheap to change schema, you can give bussiness value to data **after** you store it
- you might not actually have a use for the data **now,** but the data might contain valuable information for the future
- predictive analysis / modelling
  - predict behavior of clients
  - build recommender systems
  - better target advertisment
- clustering
  - build systems for a (sub)-group of population with specific interests
- deep-learning
  - learn about our world from the data itself

# How to handle it?

- storing hundreds of terabytes of data is tough
- storing it in a way suitable for analysis is even tougher
- one can either build systems that **scale up** (or scale **vertically)**
  - *"if you need to store more data, buy a bigger super computer"*
  - scaling up is nowadays generally expensive
  - if your data doesn't fit the computer, you will probably need a new one
- or build systems that **scale out** (or scale **horizontally**)
  - and face administrative and programming issues due to more complex systems
  - sharding
  - **cloud computing**

# Sharding



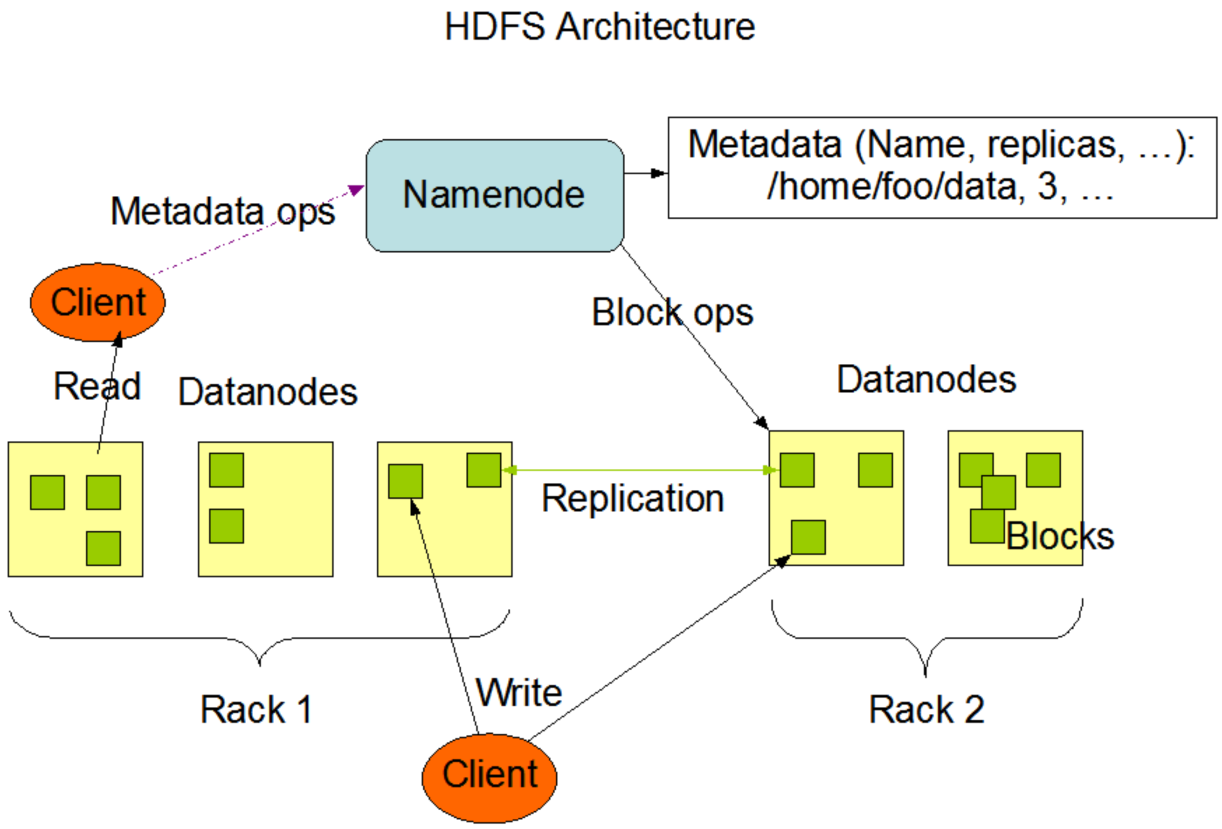
# Sharding

- break down the single system into several disjoint pieces (shards)
- clients communicate through a "proxy"
- pros
  - quite easy to implement (in most cases)
  - scales quite well (depending on data and sharding algorithm)
- cons
  - adding new shards can be tricky (repartitioning of all data might be necessary)
  - global analytics is really tough, because all the data is scattered around the cluster

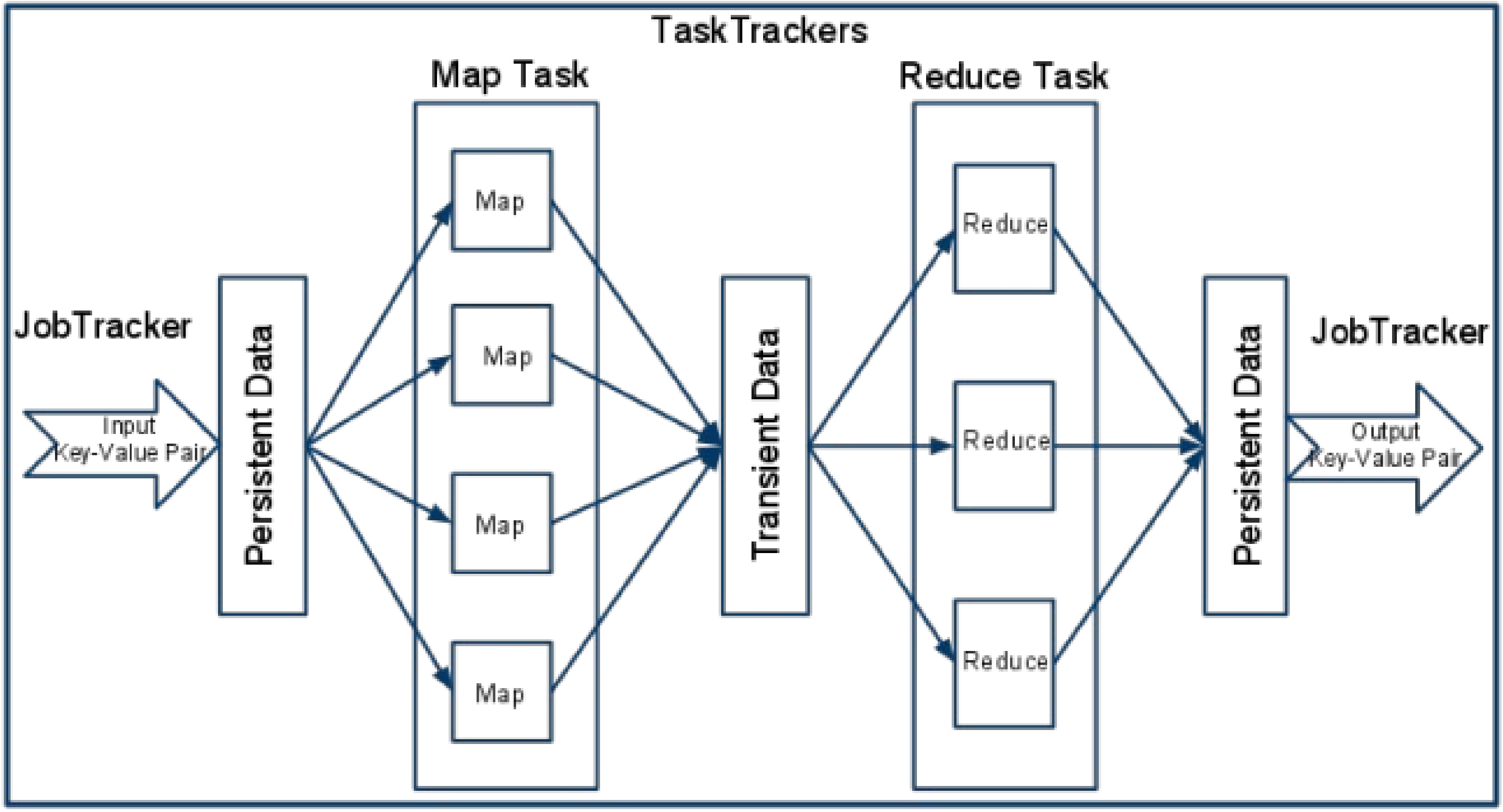
# Distributed computing

- generally very tricky and prone to hard-to-debug errors
- luckily, there exist frameworks that help to develop distributed systems
- basic tasks for a Big Data platform are
  - **storage** (HDFS, HBase, Cassandra, Kafka, ...)
  - **data processing** (MapReduce, Spark, Tez, Hive, Pig, Samza, ...)
- these systems share some common properties
  - fault tolerance
  - scaling out by adding computational (and storage) nodes
- but differ in others
  - in-memory vs. on disk processing
  - batch vs. stream processing

# Hadoop HDFS


HDFS Architecture
Metadata ops
Namenode
Metadata (Name, replicas, …):
/home/foo/data, 3, …
Client
Block ops
Read
Datanodes
Datanodes
Replication
Blocks
Rack 1
Write
Rack 2
Client

# Hadoop MapReduce



# Modern data processing

- MapReduce is quite hard to code and debug
  - instead of two operators – Map and Reduce – introduce complex operators
  - Join, Union, Sort, Split, Map, Group, ...
- memory is cheaper and larger => move to computation from disk to memory
  - if data fits into memory these systems can easily beat mapreduce by a factor of 100
- Apache Spark, Cascading, Hive, Pig, and many more
  - difficult to keep in touch with all the new systems

# Why bother with Big Data?

- managing Big Data is hard and expensive
- requires a lot of skills and a lot of resources
- storing vast amount of data requires a lot of servers, racks, electricity
- why should one be interested in this buzzword?
- *"Unreasonable effectiveness of data"*
  - small data → need much work and really smart algorithms
  - big data → sometimes only basic statistics can be sufficient to analyze the data

# Famous applications

- recommender systems
  - related items in an online store
  - playlist generation (Spotify's radio)
- semantic analysis
  - analysing text and group semantically related topics (LDA, LSA, LSI, deep learning, semantic hashing)
  - basically these algorithm learn the meaning from data
- machine translation
- feature prediction from web-graph
  - language, porn, ...
- *...* and many more!

# Log-likelihood ratio

- a very simple metric for finding structure in data
- can be used to filter "random coincidence" from pattern

| | B | not B |
|---|---|---|
| A | 1000 | 0 |
| not A | 1 | 1 |

| | B | not B |
|---|---|---|
| A | 1000 | 200 |
| not A | 1 | 1 |

| | B | not B |
|---|---|---|
| A | 1000 | 0 |
| not A | 1 | 500 |

| | B | not B |
|---|---|---|
| A | 1000 | 0 |
| not A | 200 | 500 |

# Log-likelihood ratio

- a very simple metric for finding structure in data
- can be used to filter "random coincidence" from pattern

| | B | not B |
|---|---|---|
| A | 1000 | [illegible]0 |
| not A | 1 | 1 |

6.52

| | B | not B |
|---|---|---|
| A | 1000 | 200 |
| not A | 1 | 1 |

0.57

| | B | not B |
|---|---|---|
| A | 1000 | [illegible] |
| not A | 1 | 500 |

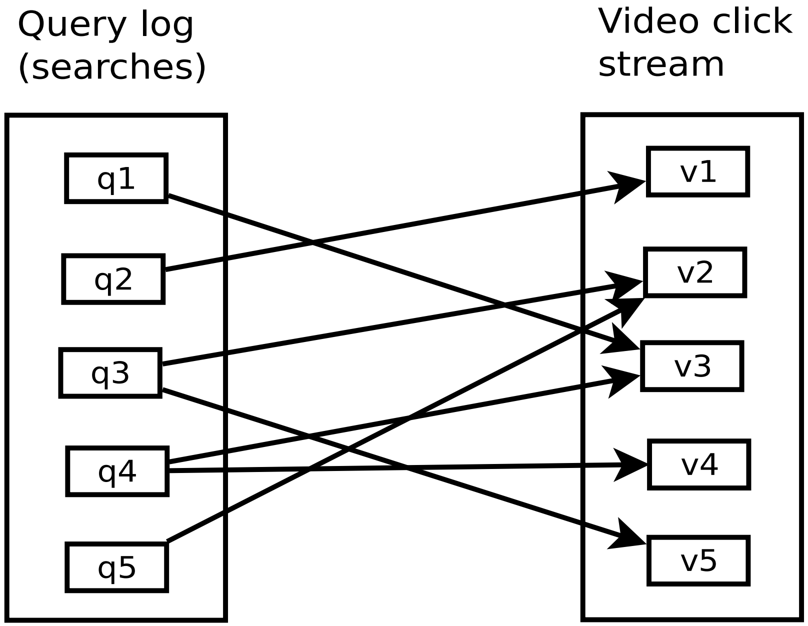
947.96

| | B | not B |
|---|---|---|
| A | 1000 | [illegible] |
| not A | 200 | 500 |

611.06

# LLR - applications

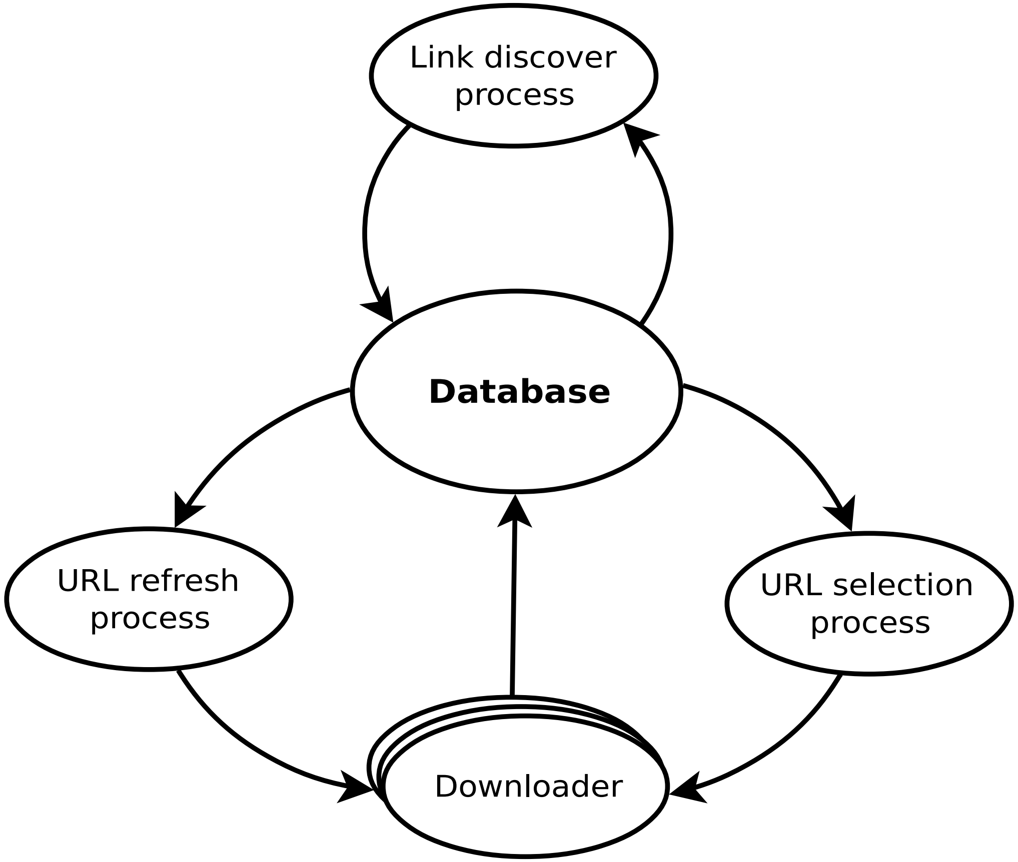
- recommendations
  - obvious, if A and B co-occur, we can recommend B to user who is interested in A and vice versa
- **search ?!?!?**



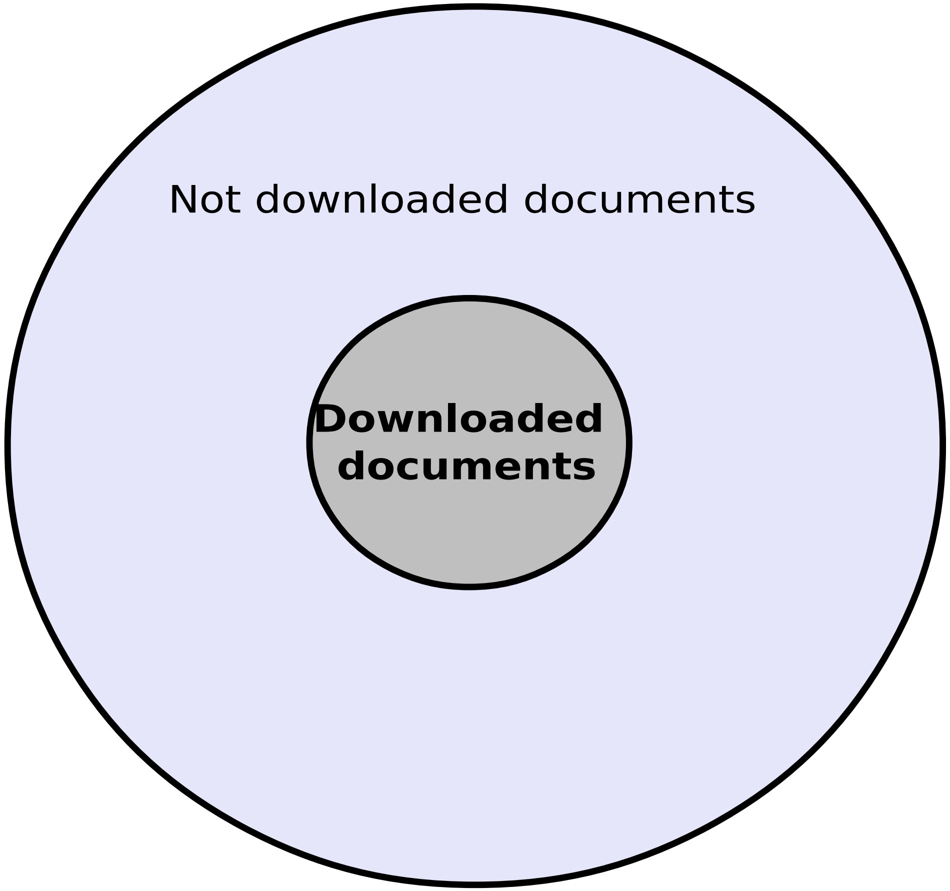
# How to crawl valuable data?

- crawling is a complex task
  - the computing capacity is limited at both sides (the crawler and the web servers)
  - although the Internet is not mathematically infinite, it is impossible to crawl each page
- the most challenging parts of designing a web scale crawler are
  - how to choose the best pages to crawl (selection policy)
  - be polite to web servers (politeness policy)
  - cope with duplicites and *canonical groups*
  - keep focused! (crawl only relevant content to your application)
  - **crawl the Web quickly**

# Typical crawler architecture



# Crawl frontier

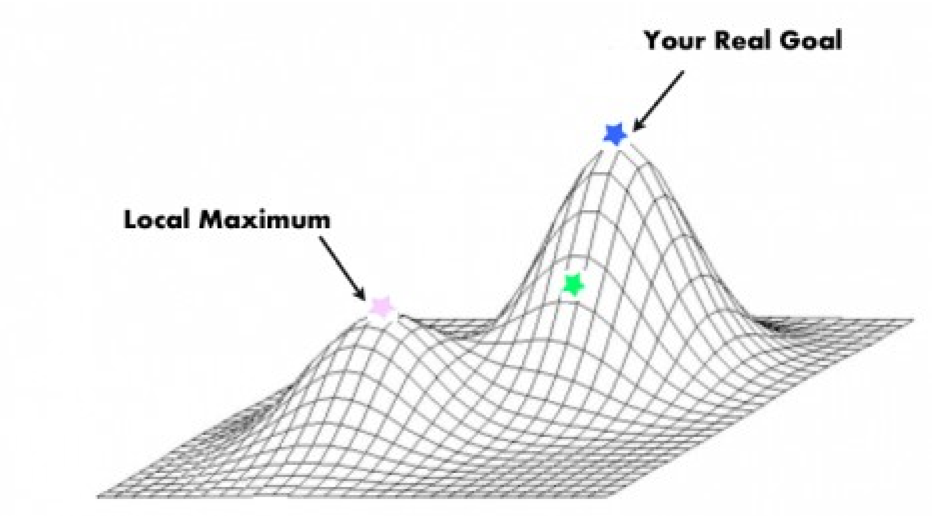


# Crawl frontier

- about 70 - 90% of documents in crawler's database are not downloaded
  - these documents **might** contain valuable information
  - or **might** link to other valuable documents
  - crawler must use predictions and heuristics to sort these pages
- but downloading one new page will statistically create new 7 – 9 previously unknown documents :-((
  - need a measure on a page quality with
  - PageRank as a measure of is not focused
  - need to modify PageRank to incorporate preferences of the crawler

# Focused crawling

- crawl only pages relevant to your bussiness
  - Seznam.cz don't need to crawl chinese pages, since our users will not search them
  - focused crawlers need to be able to get from local extremes by some sort of "uniform" sampling of the Internet, or random walk behavior



# Selection policy

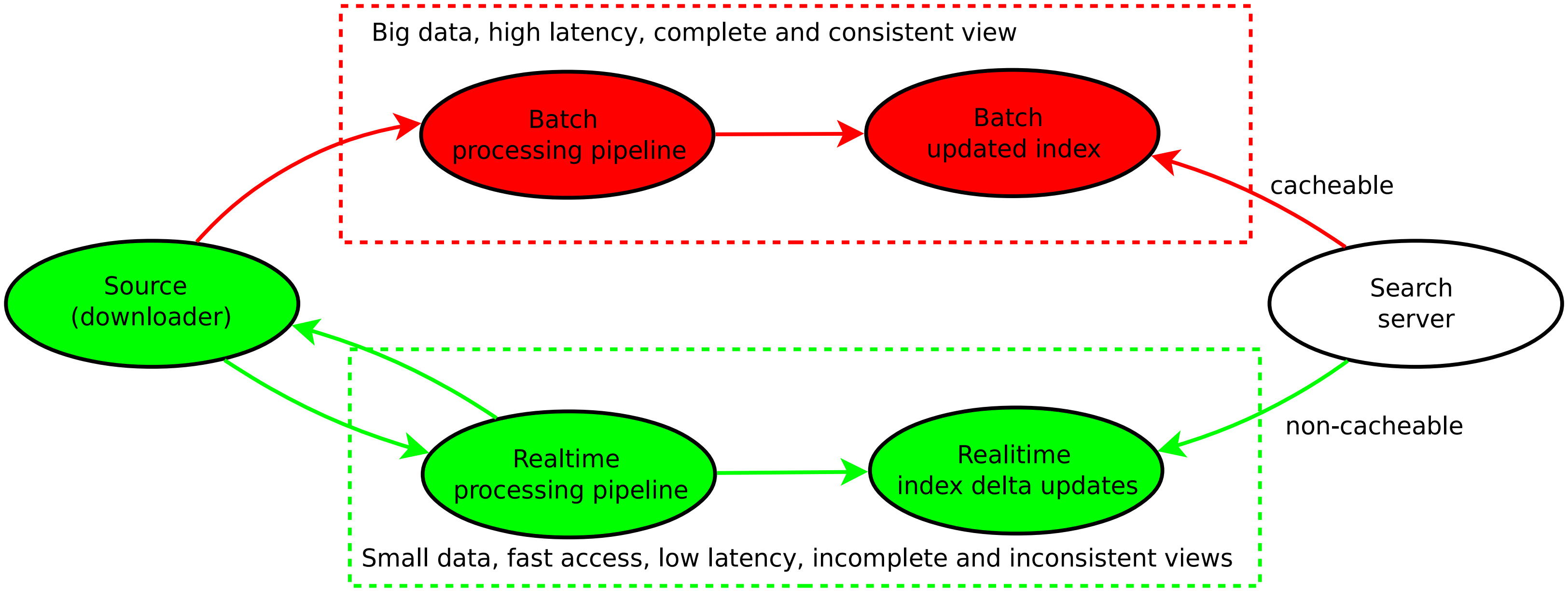
- pick best documents based on some metric
  - static metrics are often PageRank-based
- for focused crawlers we need focused PageRank
  - $$Pr(A) = w_D(A) + d \cdot \sum_{B \in Bw(A)} Pr(B) \cdot w_L(B \rightarrow A)$$
  - focused PageRank can place more weight on (apriori) relevant documents
- still need to care about "spider traps" and SEO spam
  - some pages try to mislead crawlers by black SEO techniques
  - PageRank tends to create "hot spots"
  - more complex machine learned metrics and predictions are needed
  - metric needs to be randomized to be able to escape from local optima

# Crawling – biggest challenges

- mining canonization rules
  - need to **predict** that certain page will be duplicate of already downloaded one
- soft-404 and other useless pages detection
  - soft-404 – page returning 200, but without useful content
- scaling, scaling, scaling
  - using *scalable* framework doesn't ensure *scalability*
- crawling Javascript and AJAX
- crawling deep web
  - "invisible" or "inaccessible" web

# Crawling in realtime

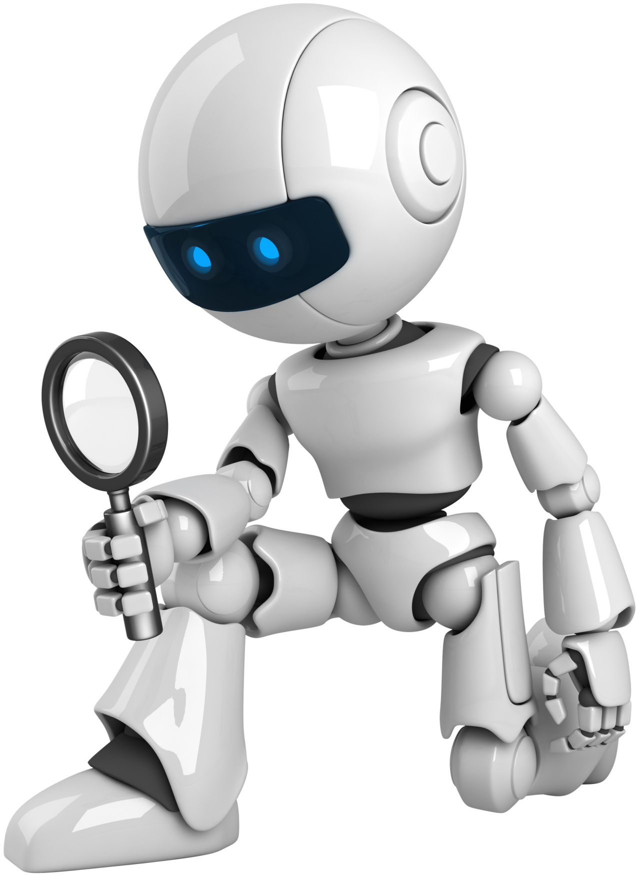
- application of **Lambda Architecture**
  - two data paths – batch and realtime
  - need estimates for static signals (PageRank cannot be calculated realtime)



# Crawling at Seznam.cz

- ~ 500 servers in two Hadoop clusters at two distinct localities
- 100 TiB of compressed downloaded data
- more than 1 PiB of all data and metadata
- 1.5 billion documents and 1 billion images
- up to 5000 URLs per second downloaded in peaks
- hundreds of millions downloaded and processed documents per day

# Questions?



Seznam.cz

...najdu tam, co neznám!

Ing. Jan Lukavský

jan.lukavsky@firma.seznam.cz