TÉCNICO LISBOA

Sistemas de Informação e Bases de Dados

Class 22: Big Data and NoSQL Databases

Prof. Paulo Carreira

Class outline

- Big Data
- Structured vs Unstructured Data
- No SQL Systems
- No SQL Technologies
  - Map Reduce
  - Key Value Stores
  - Document Stores
  - Graph Databases

Big Data

Big Data

Big data can be defined based on ***large volumes*** *of* ***extensively varied data*** *that are* ***generated***, ***captured***, *and* ***processed*** *at* ***high velocity***

As such, these data are difficult to process using existing technologies

How big is Big Data?

- 1 million records with client names and addresses ~200MB
- 1 billion records with with client names and addresses ~200GB
- 1 billion client data records <u>fits on a pen drive</u>

Even billions of records is not necessarily big data yet!

Laney’s 3 Vs of Big Data

- **Volume**. Refers to the magnitude of data (TB, PB, EB, ZB)
- **Variety**. Refers to the structural heterogeneity in a dataset (Structured, Semi-structured, Unstructured)
- **Velocity**. Refers to the rate at which data are generated and the speed at which it should be analyzed and acted upon (Batch, Near Real-Time, Real-Time, Streaming)

Laney, D. (2001, February 6). 3-D data management: Controlling data volume, velocity and variety. Application Delivery Strategies by META Group Inc. Retrieved from http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data- Management-Controlling-Data-Volume-Velocity-and-Variety.pdf

The 5 Vs of Big Data

- **Veracity** (IBM). Represents the reliability of some sources of data (dimensions of Quality, Accuracy and Trustworthiness of the data)

- **Variability** (SAS). Variability refers to the variation in the data flow rates anda data sources/

The 6th V… 🙀

- **Value** (Oracle): The value for the business of mission of the organisation created by the data

Structured Data *vs* Unstructured Data

Types of Data

Structured Data

All records follow a schema known upfront

- Databases
- Structured record files

Semi-Structured Data

Each record carries its own schema

- XML Files
- JSON Files

Unstructured Data

Data records without schema

- Text files
- Photos/Video files/ Audio files
- Call center transcripts, online reviews of products, chatbot conversations Webpages and blog posts
- Social media comments

Structured *vs.* Unstructured

- **Structured data** has semantics, searchable and queryable
- **Unstructured data** has loose semantics, not easily searchable (audio, video, and social media) and not easily queryable

Unstructured data must be converted into structured data to be queryable

Unstructured Data

Data without a formal schema

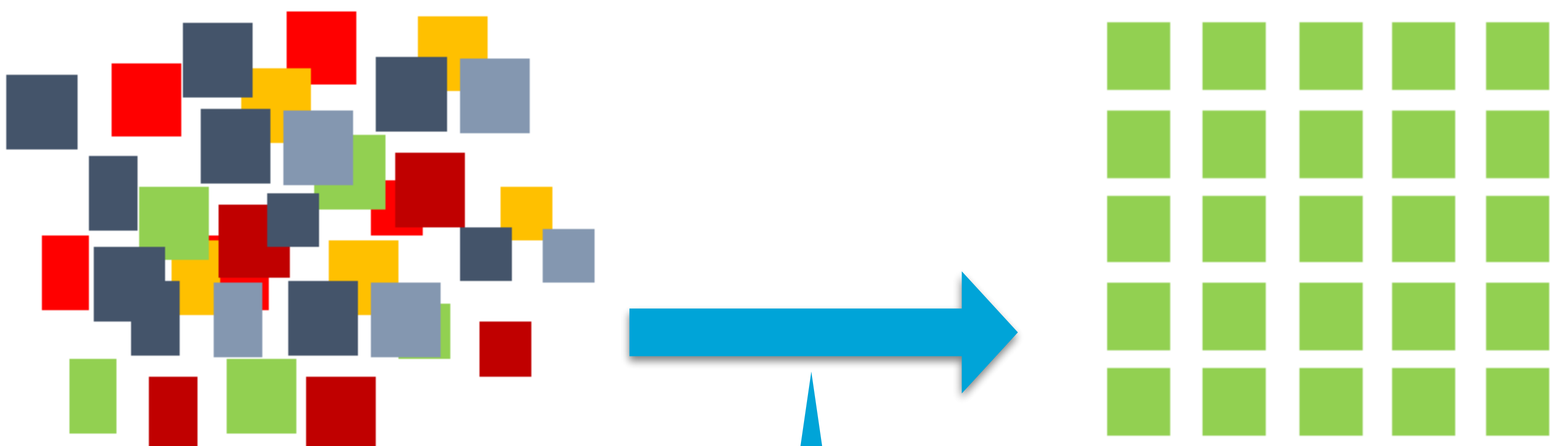
- Text files
- Photos/Video files/Audio files
- Call center transcripts, online reviews of products, chatbot conversations Webpages and blog posts
- Social media mentions

Semi-Structured Data

Data that carries its own schema
(typically hierarchical data)

- **eXtensible Markup language (XML)** is a semi-structured document language. XML is a set of document encoding rules that defines a human- and machine-readable format.
- **JavaScript Object Notation (JSON)** is another semi-structured data interchange format.

Processing Unstructured Data



AI Algorithms

Computer Vision
Speech-to-text Conversion
Natural Language Processing
Sentiment Analysis
Pattern Recognition

NoSQL

NoSQL: The Name

- SQL = Traditional relational DBMS
  - Recognition over past decades
  - However: Not every data management/analysis <u>problem is best solved exclusively using a traditional relational DBMS</u>
- NoSQL = No SQL = Not using traditional relational DBMS
- No SQL ≠ Don't use SQL language
- NoSQL = **Not Only** SQL

What is wrong with RDBMSs?

Many modern web apps have different needs (than the apps that RDBMS were designed for)

Characteristics of many modern web apps

1. Very simple data model but at a very large scale
2. Typically simple queries that retrieve elements by key
3. No need for Transactions nor Strong Consistency (virtually no concurrent access to the same objects)
4. Storage of documents instead of records
5. Correctness and quality of data is <u>not a big issue</u>

What is wrong with RDBMSs?

- Lack of support for flexible schemas / semi-structured data
- Lack of support for approximate and rank querying
- Lack of scalability & elasticity (at low cost)

Very expensive to deploy with high availability and with geographic distribution (multiple data centers)

NoSQL Systems

- Alternative to traditional relational DBMS
- Flexible schema
- Massive scalability

But...

- **No declarative query language**
  - No SQL!
  - You have to program the query in code
- **Relaxed consistency**
  - Higher performance & availability
  - Fewer guarantees

SQL is still relevant

- **Relational model** is ideal to combine data in many different ways
- **SQL transactions** offer all guarantees
- There is a large variety of **available tools** with which programmers are familiar with

Fundamental Ideas That NoSQL seems to forget

- The idea of defining the schema, the navigational structure, and the constraints explicitly (a breakthrough idea!)
- The idea of storing data and deriving information
- The ideas of Domain Specific Language; Program Transformation and Program Derivation

Why are NoSQL Systems Faster?

- No declarative query language: no time is spent optimising the query
- No constraints: No time spent checking for constraints (foreign keys, unique, and other)
- Relaxed consistency: No transaction control and no logging(!)
- No user/security enforcement: No time spend checking security (!)

NoSQL Systems Myths

1. Faster than RDBMSs
2. Easier to use to use than RDBMSs
3. Technologically more advanced than RDBMSs

The (New==Better) Fallacy

New and good ideas

"(...) You book has many new and good ideas. However, the new ones are not good and the good ones are not new (...)"

Newer does **not necessarily** equals better.

You have the <u>right</u> and the <u>responsibility</u> of judging ideias (and technologies) by their own (technical) merits, and not by recency!

Example Applications for NoSQL

Example 1: Web server log analysis

Record structure:

`(userid, URL, timestamp, additional-info)`

- **Task**: Load into database system
- **Task**: Find all records for UserID/URL/timestamp
- **Task**: Find average age of users accessing given URL from additional-info

Example 2: Social-network

Record structure:

```
(user_id1, mention_text, user_id2)
```

- **Task**: Find friends that *"think like"* friends of friends of … friends of given user

Example 3: Wikipedia pages

Record structure:

`(page_id, page_title, Text)`

- **Task**: Retrieve introductory paragraph of all pages about U.S. presidents before 1900

NoSQL Technologies

Major Features of NoSQL

Major Features of NoSQL

1. Ability to horizontally scale simple operation throughput over many servers
2. Ability to replicate and distribute (i.e., partition) data over many servers
3. A simple call level interface (typically REST in contrast to SQL)

Major Features of NoSQL

4. A weaker concurrency/consistency model than ACID transactions in RDBMSs
5. Efficient use of distributed indexes and RAM for storage
6. Ability to dynamically add new attributes to data records

NoSQL Systems

- **MapReduce**
  - A general framework for data processing

- **NoSQL Data Management Systems**
  - Key-value stores
  - Document stores
  - Graph database systems

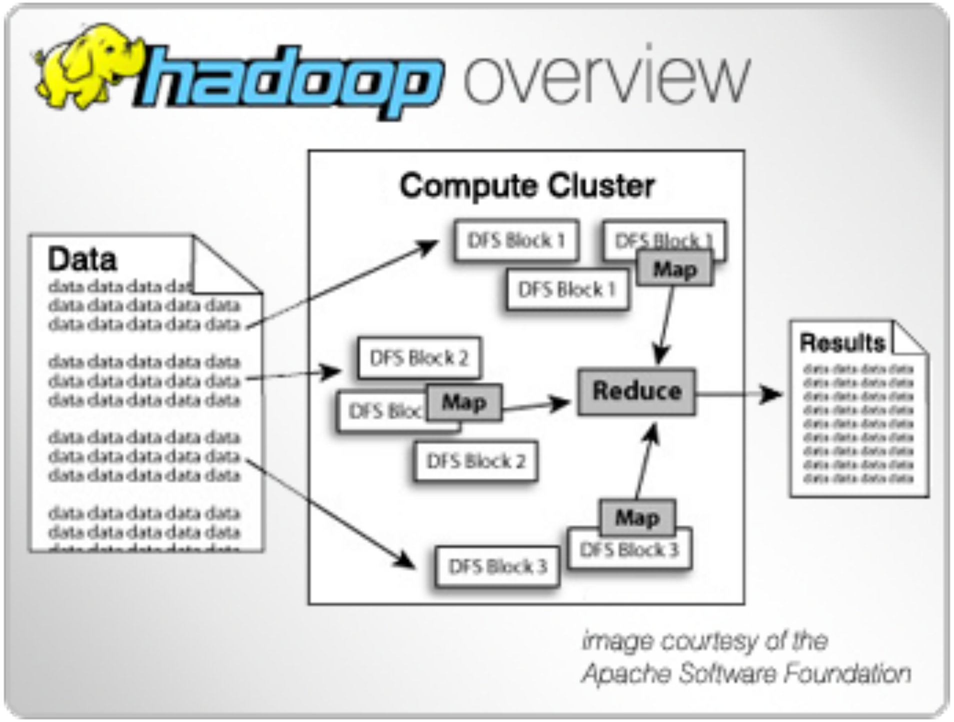
Map-Reduce

Map-Reduce Framework

- Originally from Google - open source Hadoop
  - **Distributed File System** (GFS or HDFS): No data model, data stored in files:
  - **Distributed Processing System:** User provides specific functions: map() and reduce()
- The framework provides data processing library, scalability, fault-tolerance

Map-Reduce is not a database system!

Hadoop/Map Reduce Framework



https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

Map and Reduce Meta-Functions

- **Map**: Pre-processes (divides) problem into sub-problems
  - map(item) → <key, value> pairs (0 or many)
- **Reduce**: Aggregates (combines) the results of sub-problems
  - reduce(key, *list-of-values*) → 0 or more records

MapReduce Framework: implementations

- Different implementations
  - Apache Hadoop, Spark, Disco, mrjob, …
- Schemas and declarative queries are missed, so:
  - **Pig** – more imperative, but with relational operators
  - **Hive** – schemas, SQL-like query language
  - Both compile to "workflow" of Hadoop (MapReduce) jobs

MapReduce Example: Web Server log analysis

- **Task**: Count number of accesses (based on additional-info) for each domain (from the URL)

  - Step 1: **map**(record) → <URL, count>
  - Step 2: **reduce**(URL, list-of-counts) → <URL, sum>

Key-Value Stores

Key-Value Stores

- Extremely simple interface
  - Data model: (**key**, value) pairs
  - Operations: **Insert**(key, value), **Fetch**(key), **Update**(key, value), **Delete**(key)

- Some allow complex values (not Relational Tables)
- Some allow fetch on range of keys

Key-Value Stores

Efficiency, scalability, fault-tolerance

1. **Typical implementation**: distributed hash tables
2. **Distribution**: Records often distributed to nodes based on key
3. **Replication**: Across multiple nodes (fault tolerance)
4. **Transactions**: Single-record transactions, per-site transactions, and/or eventual consistency

Key-Value Stores

- Example systems
  - Google's **BigTable**
  - Amazon Dynamo, Apache Cassandra, LinkedIn's Voldemort, Apache HBase, …
- And also
  - Oracle's BerkleyDB, MapDB, …

Document Stores

Document Stores

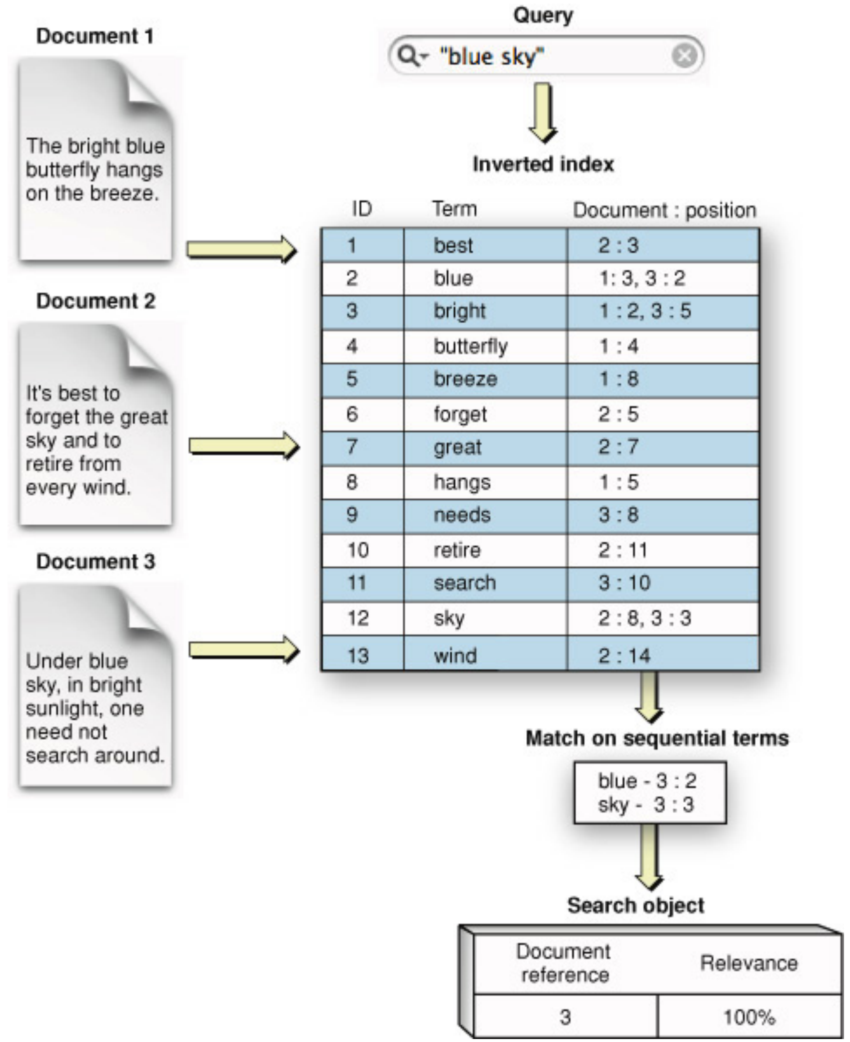
- Like Key-Value stores, except value is document
  - Data model: (key, document) pairs
  - Document: JSON, XML, other semi-structured formats
- Basic operations: Insert(key, document), Fetch(key), Update(key), Delete(key)
- Also fetch based on document contents
- Secondary indexes (and inverted indexes for text)

Document Stores

- CouchDB,
- MongoDB,
- SimpleDB,
- XBase,
- Zorba,
- TerraStore,
- Apache SOLR (i.e., Lucene)

Inverted Indexes

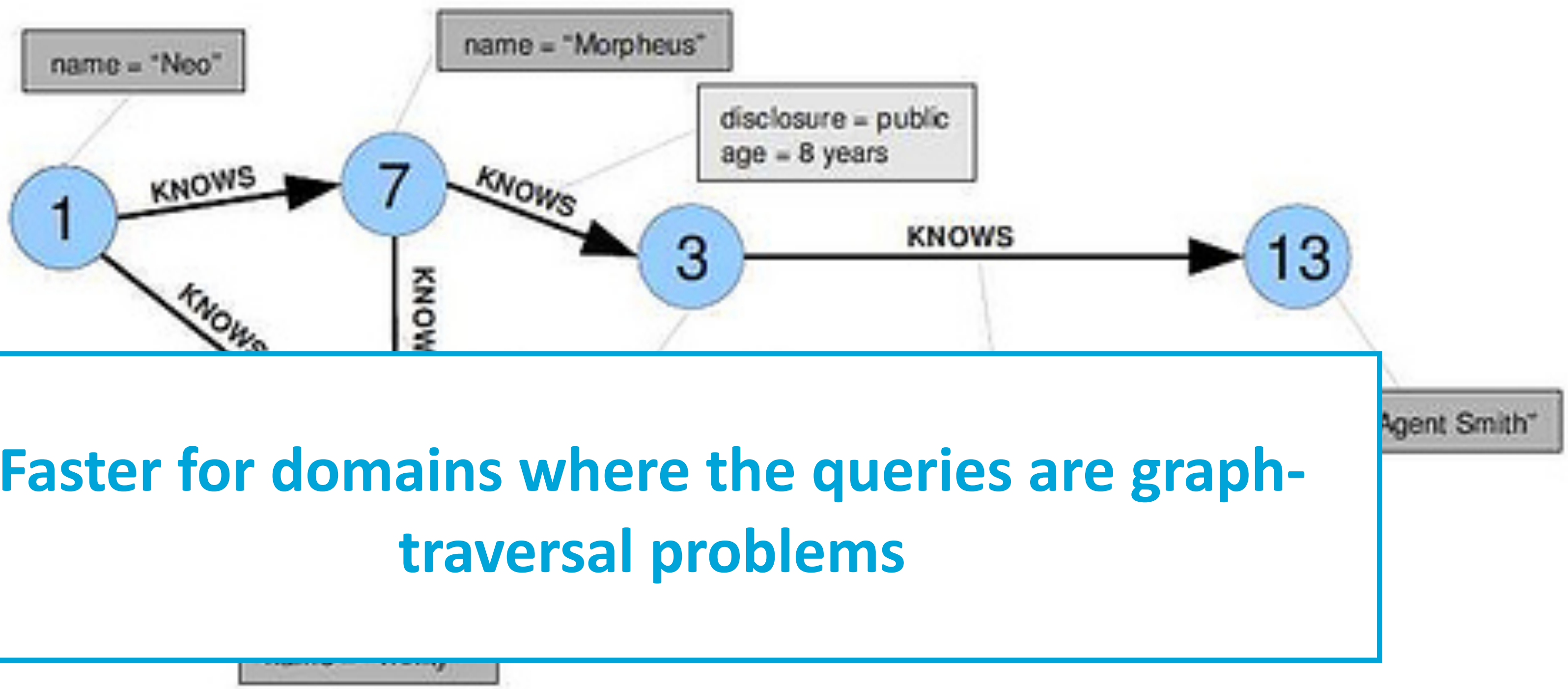
Often built through distributed data processing methods (e.g., map-reduce)



Graph Databases

Graph Database Systems

- Data model: nodes and edges
- Nodes may have properties (including ID)
- Edges may have labels and/or roles



Faster for domains where the queries are graph-traversal problems

Graph Database Systems

- Interfaces and query languages vary (no standard)
- Neo4j implements several : Cypher versus Gremlin
- Single-step versus path expressions versus full recursion (i.e., traverse the graph)
- Example systems: Neo4j, Twitter's FlockDB, Pregel, …
- RDF "triple stores" can map to graph databases Ex: AllegroGraph, Virtuoso, Apache Jena, …