Czech Technical University in Prague, Faculty of Information Technology
MIE-PDB: **Advanced Database Systems**
http://www.ksi.mff.cuni.cz/~svoboda/courses/171-MIE-PDB/

Lecture 4

# Introduction

**Martin Svoboda**
martin.svoboda@fit.cvut.cz

24. 10. 2017

Charles University, Faculty of Mathematics and Physics
NDBI040: **Big Data Management and NoSQL Databases**

# Lecture Outline

**Big Data**

- Characteristics
- Current trends

**NoSQL databases**

- Motivation
- Features

Overview of NoSQL database types

- **Key-value**, **wide column**, **document**, **graph**, …

# What is Big Data?

**Buzzword? Bubble? Gold rush? Revolution?**


*Dan Ariely*:
**Big Data** is like teenage sex: **everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it**.

# What is Big Data?

No standard definition

- Gartner (*research and advisory company*):
  **High Performance Computing**

> **Big Data** is **high volume**, **high velocity**, and/or **high variety** information assets that require **new forms of processing** to enable enhanced decision making, insight discovery and process optimization.

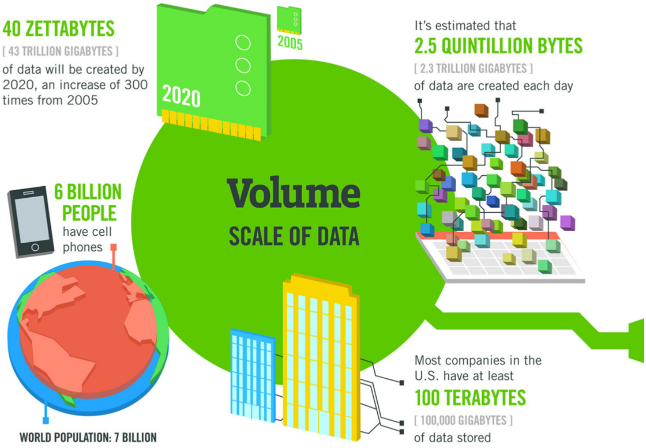
# Where is Big Data?

Sources of Big Data

- **Social media and networks**
  - …all of us are generating data
- **Scientific instruments**
  - …collecting all sorts of data
- **Mobile devices**
  - …tracking all objects all the time
- **Sensor technology and networks**
  - …measuring all kinds of data

# Big Data Characteristics

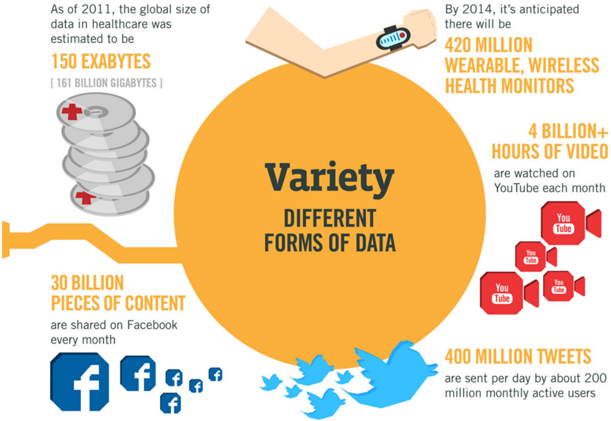
**Volume** (Scale)




Source: http://www.ibmbigdatahub.com/

# Big Data Characteristics

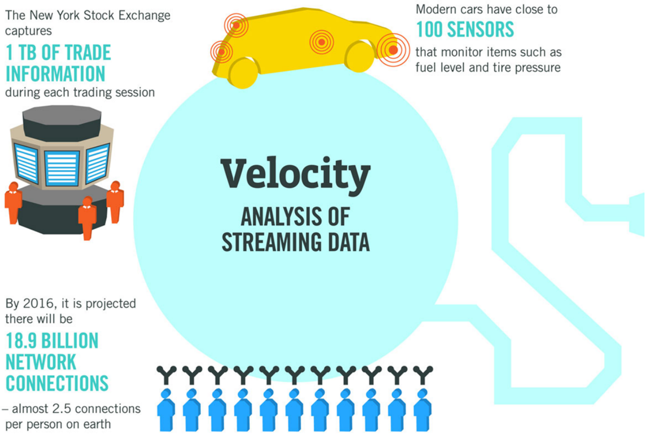
**Variety** (Complexity)




Source: http://www.ibmbigdatahub.com/

# Big Data Characteristics

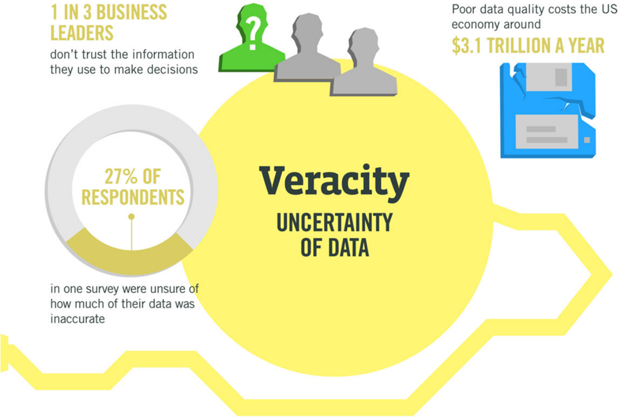
**Velocity** (Speed)




Source: http://www.ibmbigdatahub.com/

# Big Data Characteristics

**Veracity** (Uncertainty)




Source: http://www.ibmbigdatahub.com/

# Big Data Characteristics

## Basic 4V

- **Volume** (Scale)
  - Data volume is increasing exponentially, not linearly
  - Even large amounts of small data can result into Big Data
- **Variety** (Complexity)
  - Various formats, types, and structures
    (from semi-structured XML to unstructured multimedia)
- **Velocity** (Speed)
  - Data is being generated fast and needs to be processed fast
- **Veracity** (Uncertainty)
  - Uncertainty due to inconsistency, incompleteness, latency, ambiguities, or approximations

# Big Data Characteristics

## Additional V

- **V**alue
  - Business value of the data (needs to be revealed)
- **V**alidity
  - Data correctness and accuracy with respect to the intended use
- **V**olatility
  - Period of time the data is valid and should be maintained

# Relational Databases

Data model

| Instance $\rightarrow$ **database** $\rightarrow$ **table** $\rightarrow$ **row** |
| --- |

Query languages

- Real-world: **SQL** (*Structured Query Language*)
- Formal: **Relational algebra**, relational calculi (domain, tuple)

Query patterns

- **Selection** based on complex conditions, **projection**, **joins**, **aggregation**, derivation of new values, recursive queries, …

Representatives

- Oracle Database, Microsoft SQL Server, IBM DB2
- MySQL, PostgreSQL

# Relational Databases

Representatives

# Relational Databases

## Features: Normal Forms

Model

- Functional dependencies
- 1NF, 2NF, 3NF, BCNF (Boyce-Codd normal form)

Objective

- **Normalization of database schema** to BCNF or 3NF
- Algorithms: decomposition or synthesis

Motivation

- Diminish **data redundancy**, prevent update anomalies
- However:
  - Data is scattered into small pieces (high granularity), and so
  - these pieces have to be joined back together when querying!

# Relational Databases

## Features: Transactions

Model

- **Transaction** = flat sequence of database operations (`READ`, `WRITE`, `COMMIT`, `ABORT`)

Objectives

- Enforcement of ACID properties
- **Efficient parallel / concurrent execution** (slow hard drives, …)

**ACID** properties

- Atomicity – partial execution is not allowed (all or nothing)
- Consistency – transactions turn one valid database state into another
- Isolation – uncommitted effects are concealed among transactions
- Durability – effects of committed transactions are permanent

# Current Trends

**Big Data**

- **Volume**: terabytes $\rightarrow$ zettabytes
- **Variety**: structured $\rightarrow$ structured and unstructured data
- **Velocity**: batch processing $\rightarrow$ streaming data
- …

**Big users**

- Population online, hours spent online, devices online, …
- Rapidly growing companies / web applications
  - Even millions of users within a few months

# Current Trends

Everything is in **cloud**

- SaaS: Software as a Service
- PaaS: Platform as a Service
- IaaS: Infrastructure as a Service

Processing paradigms

- OLTP: Online Transaction Processing
- OLAP: Online Analytical Processing
- *…but also…*
- RTAP: **Real-Time Analytical Processing**

# Current Trends

Data assumptions

- **Data format** is becoming unknown or inconsistent
- Linear growth $\rightarrow$ **unpredictable exponential growth**
- **Read requests** often prevail **write requests**
- Data updates are no longer frequent
- Data is expected to be replaced
- Strong **consistency** is no longer mission-critical

# Current Trends

⇒ **New approach is required**

- Relational databases simply do not follow the current trends

Key technologies

- Distributed **file systems**
- **MapReduce** and other programming models
- Grid computing, cloud computing
- **NoSQL databases**
- Data warehouses
- Large scale machine learning

# NoSQL Databases

What does **NoSQL** actually mean?

A bit of history …

- 1998
  - First used for a relational database that omitted usage of SQL
- 2009
  - First used during a conference to advocate non-relational databases

So?

- Not: *no to SQL*
- Not: *not only SQL*
- NoSQL is an **accidental term with no precise definition**

# NoSQL Databases

What does **NoSQL** actually mean?

**NoSQL movement** = The whole point of **seeking alternatives** is that you need to solve a problem that **relational databases are a bad fit for**

**NoSQL databases** = Next generation databases mostly addressing some of the points: being **non-relational**, **distributed**, **open-source** and **horizontally scalable**. The original intention has been modern web-scale databases. Often more characteristics apply as: **schema-free**, **easy replication support**, **simple API**, **eventually consistent**, a **huge data amount**, and more.

Source: http://nosql-database.org/

# Types of NoSQL Databases

Core types

- **Key-value** stores
- **Wide column** (column family, column oriented, …) stores
- **Document** stores
- **Graph** databases

Non-core types

- **Object** databases
- Native **XML** databases
- **RDF** stores
- …

# Key-Value Stores

Data model

- The most simple NoSQL database type
  - Works as a simple hash table (mapping)
- **Key-value pairs**
  - Key (id, identifier, primary key)
  - Value: binary object, black box for the database system

Query patterns

- Create, update or remove value for a given key
- **Get value** for a given key

Characteristics

- Simple model ⇒ **great performance, easily scaled, …**
- Simple model ⇒ **not for complex queries nor complex data**

# Key-Value Stores

**Suitable use cases**

- Session data, user profiles, user preferences, shopping carts, …
  - I.e. **when values are only accessed via keys**

When not to use

- **Relationships among entities**
- Queries requiring **access to the content of the value part**
- **Set operations** involving multiple key-value pairs

Representatives

- **Redis**, **MemcachedDB**, **Riak KV**, Hazelcast, Ehcache, Amazon SimpleDB, Berkeley DB, Oracle NoSQL, Infinispan, LevelDB, Ignite, Project Voldemort
- *Multi-model*: OrientDB, ArangoDB

# Key-Value Stores

Representatives

redis

MEMCACHED

riakKV

hazelcast

EHCACHE

AEROSPIKE

SimpleDB

ORACLE
BERKELEY DB

ArangoDB

# Document Stores

Data model

- **Documents**
  - Self-describing
  - **Hierarchical tree structures** (JSON, XML, …)
    - Scalar values, maps, lists, sets, nested documents, …
  - Identified by a **unique identifier** (key, …)
- Documents are **organized into collections**

Query patterns

- Create, update or remove a document
- **Retrieve documents according to complex query conditions**

Observation

- Extended key-value stores where the value part is examinable!

# Document Stores

**Suitable use cases**

- Event logging, content management systems, blogs, web analytics, e-commerce applications, …
  - I.e. **for structured documents with similar schema**

When not to use

- **Set operations** involving multiple documents
- Design of document structure is constantly changing
  - I.e. when the required level of granularity would outbalance the advantages of aggregates

# Document Stores

Representatives

- **MongoDB**, **Couchbase**, Amazon **DynamoDB**, **CouchDB**, RethinkDB, RavenDB, Terrastore
- *Multi-model*: **MarkLogic**, **OrientDB**, OpenLink Virtuoso, ArangoDB

# Document Stores

Representatives

mongoDB®

Couchbase

amazon DynamoDB

CouchDB relax

RethinkDB

terrastore

RAVENDB Safe by Default, Optimized for Efficiency

MarkLogic™

OrientDB®

# Wide Column Stores

Data model

- Column family (table)
  - Table is a collection of **similar rows** (not necessarily identical)
- Row
  - Row is a collection of **columns**
    - Should encompass a group of data that is accessed together
  - Associated with a unique **row key**
- Column
  - Column consists of a **column name** and **column value** (and possibly other metadata records)
  - Scalar values, but also **flat sets, lists or maps** may be allowed

# Wide Column Stores

Query patterns

- Create, update or remove a row within a given column family
- **Select rows according to a row key or simple conditions**

Warning

- Wide column stores are not just a special kind of RDBMSs with a variable set of columns!

# Wide Column Stores

**Suitable use cases**

- Event logging, content management systems, blogs, …
  - I.e. **for structured flat data with similar schema**

When not to use

- **ACID transactions** are required
- **Complex queries**: aggregation (SUM, AVG, …), joining, …
- Early prototypes: i.e. when **database design may change**

Representatives

- Apache **Cassandra**, Apache **HBase**, Apache Accumulo, Hypertable, **Google Bigtable**

# Wide Column Stores

Representatives

# Graph Databases

Data model

- Property graphs
  - **Directed / undirected graphs**, i.e. collections of …
    - nodes (vertices) for real-world entities, and
    - relationships (edges) between these nodes
  - Both the nodes and relationships can be associated with additional properties

Types of databases

- **Non-transactional** = small number of very large graphs
- **Transactional** = large number of small graphs

# Graph Databases

Query patterns

- Create, update or remove a node / relationship in a graph
- **Graph algorithms** (shortest paths, spanning trees, …)
- General **graph traversals**
- **Sub-graph** queries or **super-graph** queries
- Similarity based queries (approximate matching)

Representatives

- **Neo4j**, **Titan**, Apache Giraph, InfiniteGraph, FlockDB
- *Multi-model*: **OrientDB**, OpenLink **Virtuoso**, **ArangoDB**

# Graph Databases

**Suitable use cases**

- Social networks, routing, dispatch, and location-based services, recommendation engines, chemical compounds, biological pathways, linguistic trees, …
  - I.e. simply **for graph structures**

When not to use

- **Extensive batch operations** are required
  - Multiple nodes / relationships are to be affected
- **Only too large graphs** to be stored
  - Graph distribution is difficult or impossible at all

# Graph Databases

## Representatives

- neo4j
- OrientDB®
- TITAN
- ArangoDB
- APACHE GIRAPH
- INFINITEGRAPH®

# Native XML Databases

Data model

- **XML documents**
  - Tree structure with nested elements, attributes, and text values (beside other less important constructs)
  - Documents are organized into collections

Query languages

- **XPath**: *XML Path Language* (navigation)
- **XQuery**: *XML Query Language* (querying)
- **XSLT**: *XSL Transformations* (transformation)

Representatives

- **Sedna**, **Tamino**, BaseX, eXist-db
- *Multi-model*: **MarkLogic**, OpenLink **Virtuoso**

# Native XML Databases

Representatives

MarkLogic™

OpenLink Virtuoso Universal Server

sedna
Native XML Database System

Tamino XML Server

existdb

# RDF Stores

Data model

- **RDF triples**
  - Components: **subject**, **predicate**, and **object**
  - Each triple represents a statement about a real-world entity
- Triples can be viewed as **graphs**
  - **Vertices** for subjects and objects
  - **Edges** directly correspond to individual statements

Query language

- **SPARQL**: *SPARQL Protocol and RDF Query Language*

Representatives

- Apache **Jena**, **rdf4j** (Sesame), Algebraix
- *Multi-model*: **MarkLogic**, OpenLink **Virtuoso**

# RDF Stores

Representatives

# Features of NoSQL Databases

**Data model**

- Traditional approach: relational model
- (New) possibilities:
  - **Key-value**, **document**, **wide column**, **graph**
  - Object, XML, RDF, …
- Goal
  - Respect the real-world nature of data
    (i.e. data structure and mutual relationships)

# Features of NoSQL Databases

**Aggregate structure**

- Aggregate definition
  - Data unit with a complex structure
  - **Collection of related data pieces we wish to treat as a unit** (with respect to data manipulation and data consistency)
- Examples
  - **Value** part of key-value pairs in key-value stores
  - **Document** in document stores
  - **Row** of a **column family** in wide column stores

# Features of NoSQL Databases

**Aggregate structure**

- Types of systems
  - **Aggregate-ignorant**: relational, graph
    - It is not a bad thing, it is a feature
  - **Aggregate-oriented**: key-value, document, wide column
- Design notes
  - No universal strategy how to draw **aggregate boundaries**
  - **Atomicity** of database operations:
    just a single aggregate at a time

# Features of NoSQL Databases

**Elastic scaling**

- Traditional approach: scaling-up
  - Buying bigger servers as database load increases
- New approach: scaling-out
  - Distributing database data across multiple hosts
    - Graph databases (unfortunately): difficult or impossible at all

**Data distribution**

- Sharding
  - Particular ways how database data is split into separate groups
- Replication
  - Maintaining several data copies (performance, recovery)

# Features of NoSQL Databases

**Automated processes**

- Traditional approach
  - Expensive and highly trained database administrators
- New approach: **automatic recovery, distribution, tuning, …**

**Relaxed consistency**

- Traditional approach
  - **Strong consistency** (ACID properties and transactions)
- New approach
  - **Eventual consistency** only (BASE properties)
  - I.e. we have to make trade-offs because of the data distribution

# Features of NoSQL Databases

**Schemalessness**

- Relational databases
  - Database schema present and **strictly enforced**
- NoSQL databases
  - **Relaxed schema** or **completely missing**
  - Consequences: **higher flexibility**
    - Dealing with **non-uniform data**
    - **Structural changes** cause no overhead
  - However: there is (usually) an **implicit schema**
    - We must know the data structure at the application level anyway

# Features of NoSQL Databases

**Open source**

- Often community and enterprise versions (with extended features or extent of support)

**Simple APIs**

- Often state-less application interfaces (HTTP)

# Features of NoSQL Databases

## Current State: Five advantages

- **Scaling**
  - Horizontal distribution of data among hosts
- **Volume**
  - High volumes of data that cannot be handled by RDBMS
- **Administrators**
  - No longer needed because of the automated maintenance
- **Economics**
  - Usage of cheap commodity servers, lower overall costs
- **Flexibility**
  - Relaxed or missing data schema, easier design changes

# Features of NoSQL Databases

## Current State: Five challenges

- **Maturity**
  - Often still in pre-production phase with key features missing
- **Support**
  - Mostly open source, limited sources of credibility
- **Administration**
  - Sometimes relatively difficult to install and maintain
- **Analytics**
  - Missing support for business intelligence and ad-hoc querying
- **Expertise**
  - Still low number of NoSQL experts available in the market

# Conclusion

**The end of relational databases?**

- Certainly no
  - They are still suitable for most projects
  - Familiarity, stability, feature set, available support, …
- However, we should also consider different database models and systems
  - Polyglot persistence = **usage of different data stores in different circumstances**

# Course Overview

## Outline and Objectives

Principles

- **Scaling**, **distribution**, **consistency**
- Transactions, visualization, …

Technologies

- **MapReduce** programming model
  - Apache Hadoop
- **Data formats**
  - XML, JSON, RDF, …
- **NoSQL databases**
  - Core: **RiakKV**, **Redis**, **MongoDB**, **Cassandra**, **Neo4j**
  - Non-core: XML, RDF
  - *Data models, query languages, …*

# Lecture Conclusion

**Big Data**

- 4V characteristics: volume, variety, velocity, veracity

**NoSQL databases**

- (New) **logical models**
  - Core: key-value, wide column, document, graph
  - Non-core: XML, RDF, …
- (New) **principles and features**
  - Horizontal scaling, data sharding and replication, eventual consistency, …