Data Models & Processing

Lecture BigData Analytics

Julian M. Kunkel

julian.kunkel@googlemail.com

University of Hamburg / German Climate Computing Center (DKRZ)

2016-10-28

Disclaimer: Big Data software is constantly updated, code samples may be outdated.

Outline

Basic Considerations About Storing Big Data

Analysis requires efficient (real-time) processing of data

- New data is constantly coming (Velocity of Big Data)
  - How do we technically ingest the data?
    - In respect to performance and data quality
  - How can we update our derived data (and conclusions)?
    - Incremental updates vs. (partly) re-computation algorithms
- Storage and data management techniques are needed
  - How do we map the logical data to physical hardware and organize it?
  - How can we diagnose causes for problems with data (e.g., inaccuracies)?

Management of data

- Idea: Store facts (truth) and never change them (data lake idea)
  - Data value may degrade over time, garbage clean old data
- Raw data is usually considered to be **immutable**
  - Implies that an update of (raw) data is not necessary
- Create ad-hoc models for representing the data

Terminology

Data [1, 10]

- **Raw data**: collected information that is not derived from other data
- **Derived data**: data produced with some computation/functions
- **View**: presents derived data to answer specific questions
  - Convenient for users (only see what you need) + faster than re-computation
  - Convenient for administration (e.g., manage permissions)
  - Data access can be optimized

Dealing with unstructured data

- We need to extract information from raw unstructured data
  - e.g., perform text-processing using techniques from computer linguistics
- **Semantic normalization** is the process of reshaping free-form information into a structured form of data [11]
- Store raw data when your processing algorithm improves over time

Terminology for Managing Data [1, 10]

- **Data life cycle**: creation, distribution, use, maintenance & disposition
- **Information lifecycle management** (ILM): business term; practices, tools and policies to manage the data life cycle in a cost-effective way
- **Data governance**: "*control that ensures that the data entry ... meets precise standards such as business rule, a data definition and data integrity constraints in the data model*" [10]
- **Data provenance**: the documentation of input, transformations of data and involved systems to support analysis, tracing and reproducibility
- **Data-lineage** (Datenherkunft): forensics; allows to identify the source data used to generate data products (part of data provenance)
- **Service level agreements** (SLAs): contract defining quality, e.g., performance/reliability & responsibilities between service user/provider

Data-Cleaning and Ingestion

- Importing of raw data into a big data system is an important process
  - Wrong data results in wrong conclusions: Garbage in – Garbage out
- **Data wrangling**: process & procedures to clean/convert data from one format to another [1]
  - **Data extraction**: identify relevant data sets and extract raw data
  - **Data munging**: cleaning raw data, converting it to a format for consumption
- **ETL process** (Extract, Transform, Load): data warehouse term for importing data (from databases) into a data warehouse

Necessary steps

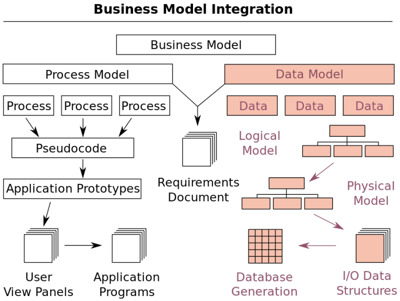
- Define and document data governance policies to ensure data quality
  - Identifying and dealing with duplicates, time(stamp) synchronization
  - Handling of missing values (NULL or replace them with default values)
- Document the conducted transformations (for data provenance)
  - Data sources
  - Conversions of data types, complex transformations
  - Extraction of information from unstructured data (semantic normalization)
- Implementation of the procedures for bulk loading and cleaning of data

Datawarehousing ETL Process

- Extract: read data from source (transactional) databases
- Transform: alter data on the fly
  - Perform quality control
  - Treat errors and uncertainty to improve quality
  - Change the layout to fit the schema of the data warehouse
- Load: integrate the data into the data warehouse
  - Restructure data to fit needs of business users
  - Rely on batch integration of large quantities of data

Data Models[1] and their Instances [12]

- A data model describes how information is organized in a system
  - It is a tool to specify, access and process information
  - A model provide operations for accessing and manipulating data that follow certain semantics
  - Typical information is some kind of entity (virtual object) (e.g., car)
- **Logical model**: abstraction expressing objects and operations
- **Physical model**: maps logical structures onto hardware resources (e.g., files, bytes)



Source: [12]

- DM theory: Formal methods for describing data models with tool support
- Applying theory creates a **data model instance** for a specific application

[1]The term is often used ambivalently for a data (meta) model concept/theory or an instance

Process Model [13]

- A model describing processes
- **Process** [15]:
  - *"A series of events to produce a result, especially as contrasted to product."*
- Qualities of descriptions
  - Descriptive: Describe the events that occur during the process
  - Prescriptive
    - Define the intended process and how it is executed
    - Rules and guideliness steering the process
  - Explanatory
    - Provide rationales for the process
    - Describe requirements
    - Establish links between processes

Programming Paradigms [14]

Programming paradigms are process models for computation

- Fundamental style and abstraction level for computer programming
  - **Imperative** (e.g., Procedural)
  - **Declarative** (e.g., Functional, **Dataflow**, Logic)
  - **Data-driven** programming (describe patterns and transformations)
  - **Multi-paradigm** support several at the same time (e.g., **SQL**)
- Goals: productivity of the users and performance upon execution
  - Tool support for development, deployment and testing
  - Performance depends on single core efficiency but importantly parallelism
- **Parallelism** is an important aspect for processing of large data
  - In HPC, there are language extensions, libraries to specify parallelism
    - PGAS, Message Passing, OpenMP, data flow e.g., OmpSs, ...
  - In BigData Analytics, libraries and domain-specific languages
    - MapReduce, SQL, data-flow, streaming and data-driven

Domain-specific Language (DSL)

- DSLs are a contrast to general-purpose languages (GPL)
- Specialized programming language to an application domain
  - Mathematics, e.g., statistics, modeling
  - Description of graphs, e.g., graphviz (dot)
  - Processing of big data (Apache Pig)
- Standalone vs. embedded DSLs
  - Embedding into a GPL (e.g., regex, SQL) with library support
  - Standalone requires to provide its own toolchain (e.g., compiler)
  - Source-to-source compilation (DSL to GPL) an alternative
- Abstraction level
  - High-level: only covers (scientific) domain
  - Low-level: includes technical details (e.g., about hardware)

Selection of Theory (concepts) for Data Models

- I/O Middelware: NetCDF, HDF5, ADIOS
  - Self-describing formats containing N-dim variables with metadata
- Relational model (tuples and tables)
  - Can be physically stored in, e.g., a CSV file or database
- Relational model + raster data
  - Operations for N-dimensional data (e.g., pictures, scientific data)
- NoSQL data models: Not only SQL[2], lacks features of databases
  - Example DB models: columnar, document, key-value, named graph
- Fact-based: built on top of atomic facts, well-suited for BI [11]

Data modeling [10]

The process in software-engineering of creating a data model instance for an information system

[2] Sometimes people also call it No SQL

Semantics

Semantics describe I/O operations and their behavior

- Application programming interface (API)
- **Concurrency**: Behavior of simultaneously executed operations
  - Atomicity: Are partial modifications visible to other clients
  - Visibility: When are changes visible to other clients
  - Isolation: Are operations influencing other ongoing operations
- **Availability**: Readiness to serve operations
  - Robustness of the system for typical (hardware and software) errors
  - Scalability: availability and performance behavior depending on the number of clients, concurrent requests, request size, ...
  - Partition tolerance: Continue to operate even if network breaks partially
- **Durability**: Modifications should be stored on persistent storage
  - Consistency: Any operation leaves a consistent (correct) system state

Consensus [17]

- **Consensus**: several processes agree (decide) for a single data value
  - Processes may propose a value (any time)
- Consensus and consistency of distributed processes are related
- Consensus protocols such as Paxos ensure cluster-wide consistency
  - They tolerate typical errors in distributed systems
  - Hardware faults and concurrency/race conditions
  - Byzantine protocols additionally deal with forged (lying) information
- Properties of consensus
  - Termination: Every correct process decides upon a value
  - Validity: If all process propose the same value v, then all correct processes decide v
  - Integrity: All correct process decide upon at most one value v. If one decides v, then v has been proposed by some process
  - Agreement: Every correct process must agree on the same value

Assumptions for Paxos

Requirements and *fault-tolerance assumptions* [16]

- Processors
  - **do not collude, lie, or otherwise attempt to subvert the protocol**
  - *operate at arbitrary speed*
  - *may experience failures*
  - *may re-join the protocol after failures (when they keep data durable)*
- Messages
  - **can be send from one processor to any other processor**
  - **are delivered without corruption**
  - *are sent asynchronously and may take arbitrarily long to deliver*
  - *may be lost, reordered, or duplicated*

Fault tolerance

- With 2F+1 processors, F faults can be tolerated
- With dynamic reconfiguration more, but $\leq$ F can fail simultaneously

Consistency Limitations in Distributed Systems

CAP-Theorem

- It initially discusses implications if the network is partitioned[3]
  - Consistency (here: visibility of changes among all clients)
  - Availability (we'll receive a response for every request)
  - Any technology can only achieve either consistency or availability

$\Rightarrow$ It is impossible to meet the attributes together in a distributed system:
  - Consistency
  - Availability
  - Partition tolerance (system operates despite network failures)

[3] This means that network failures split the network peers into multiple clusters that cannot communicate.

Example Semantics

POSIX I/O

- Atomicity and isolation for individual operations, locking possible

ACID

- Strict semantics for database systems to prevent data loss
- Atomicity, consistency, isolation and durability for **transactions**

BASE

- BASE is a typical semantics for Big Data due to the CAP theorem
- Basically Available replicated Soft state with Eventual consistency [26]
  - Availability: Always serve but may return a failure, retry may be needed
  - Soft state: State of the system may change over time without requests due to eventual consistency
  - Consistency: If no updates are made any more, the last state becomes visible to all clients after some time (eventually)
- Big data solutions usually exploit the immutability of data

Columnar Model

- Data is stored in rows and “columns” (evtl. tables)
- A column is a tuple (name, value and timestamp)
- Each row can contain different columns
  - Columns can store complex objects, e.g., collections
- Wide columnar model: very sparse table of 100k+ columns
- Example technology: HBase, Cassandra, Accumulo

Row/Column:	student name	matrikel	lectures	lecture name
1	"Max Mustermann"	4711	[3]	-
2	"Nina Musterfrau"	4712	[3,4]	-
3	-	-	-	"Big Data Analytics"
4	-	-	-	"Hochleistungsrechnen"

Example columnar model for the students, each value has its own timestamp (not shown).
Note that lectures and students should be modeled with two tables

Key-Value Store

- Data is stored as value and addressed by a key
- The value can be complex objects, e.g., JSON or collections
- Keys can be forged to simplify lookup (evtl. tables with names)
- Example technology: CouchDB, BerkeleyDB, Memcached, BigTable

Key	Value
stud/4711	<name>Max Mustermann</name><attended><id>1</id></attended>
stud/4712	<name>Nina Musterfrau</name><attended><id>1</id><id>2</id></attended>
lec/1	<name>Big Data Analytics</name>
lec/2	<name>Hochleistungsrechnen</name>

Example key-value model for the students with embedded XML

Document Model

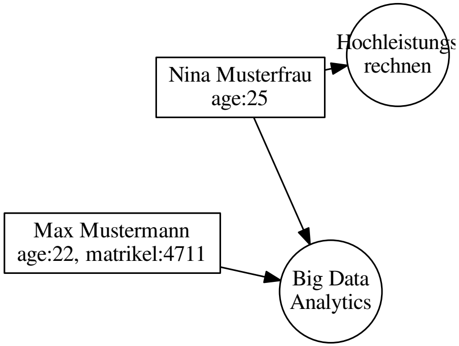
- Documents contain semi-structured data (JSON, XML)
- Each document can contain data with other structures
- Addressing to lookup documents are implementation specific
  - e.g., bucket/document key, (sub) collections, hierarchical namespace
- References between documents are possible
- Example technology: MongoDB, Couchbase, DocumentDB

```
<students>
  <student><name>Max Mustermann</name><matrikel>4711</matrikel>
    <lecturesAttended><id>1</id></lecturesAttended>
  </student>
  <student><name>Nina Musterfrau</name><matrikel>4712</matrikel>
    <lecturesAttended><id>1</id><id>2</id></lecturesAttended>
  </student>
</students>
```

Example XML document storing students. Using a bucket/key namespace, the document could be addressed with key: “uni/stud” in the bucket “app1”

Graph

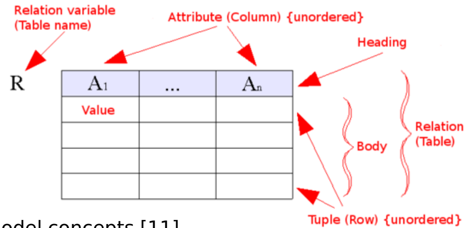
- Entities are stored as nodes and relations as edges in the graph
- Properties/Attributes provide additional information as key/value
- Example technology: Neo4J, InfiniteGraph



Graph representing the students (attributes are not shown)

Relational Model [10]

- Database model based on first-order predicate logic
  - Theoretic foundations: relational algebra and relational calculus
- Data is represented as tuples
  - In its original style, it does not support collections
- Relation/Table: groups tuples with similar semantics
  - Table consists of rows and named columns (attributes)
  - No duplicates of complete rows allowed
- Schema: specify structure of tables
  - Datatypes (domain of attributes)
  - Consistency via constraints
  - Organization and optimizations



Source: Relational model concepts [11]

Example Relational Model for Students Data

Matrikel	Name	Birthday
242	Hans	22.04.1955
245	Fritz	24.05.1995

Student table

ID	Name
1	Big Data Analytics
2	Hochleistungsrechnen

Lecture table

Matrikel	LectureID
242	1
242	2
245	2

Attends table representing a relation

Fact-Based Model [11][5]

- Store raw data as timestamped atomic facts
- Never delete true facts: Immutable data
- Make individual facts unique to prevent duplicates

Example: social web page

- Record all changes to user profiles as facts
- Benefits
  - Allows reconstruction of the profile state at any time
  - Can be queried at any time[4]

Example: purchases

- Record each item purchase as facts together with location, time, ...

[4] If the profile is changed recently, the query may return an old state.

[5] Note that the definitions in the data warehousing (OLAP) and big data [11] domains are slightly different

Wishlist for Big Data Technology [11]

- High-availability, fault-tolerance
- Linear scalability in respect to data volume
  - i.e., 2n servers handle 2n the data volume + same processing time
- Real-time data processing capabilities (interactive)
  - Up-to-date data
- Extensible, i.e., easy to introduce new features and data
- Simple programming models to increase user productivity
- Debuggability
  - In respect to coding errors and performance issues
- Cheap & ready for the cloud

  $\Rightarrow$ Technology works with TCP/IP

Components for Big Data Analytics

Required components for a big data system

- Servers, storage, processing capabilities
- User interfaces

Storage

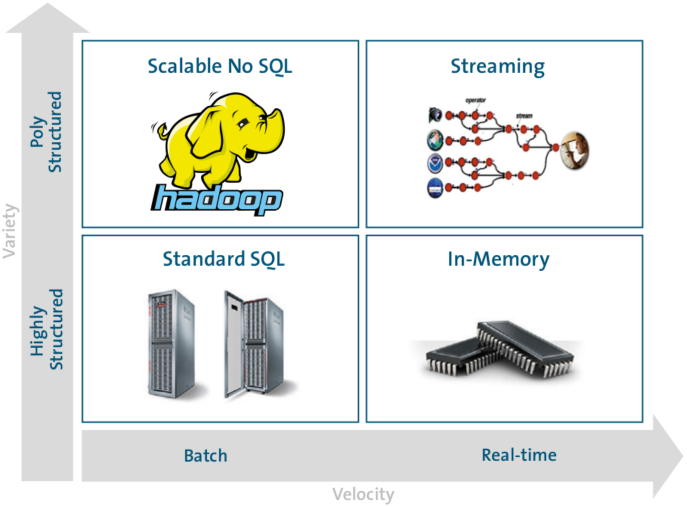
- NoSQL databases are non-relational, distributed and scale-out
  - Hadoop Distributed File System (HDFS)
  - Cassandra, CouchDB, BigTable, MongoDB [6]
- Data Warehouses are useful for well known and repeated analysis

Processing capabilities

- Interactive processing is difficult
- Available technology offers
  - Batch processing (hours to a day processing time)
  - “Real-time” processing (seconds to minutes turnaround)

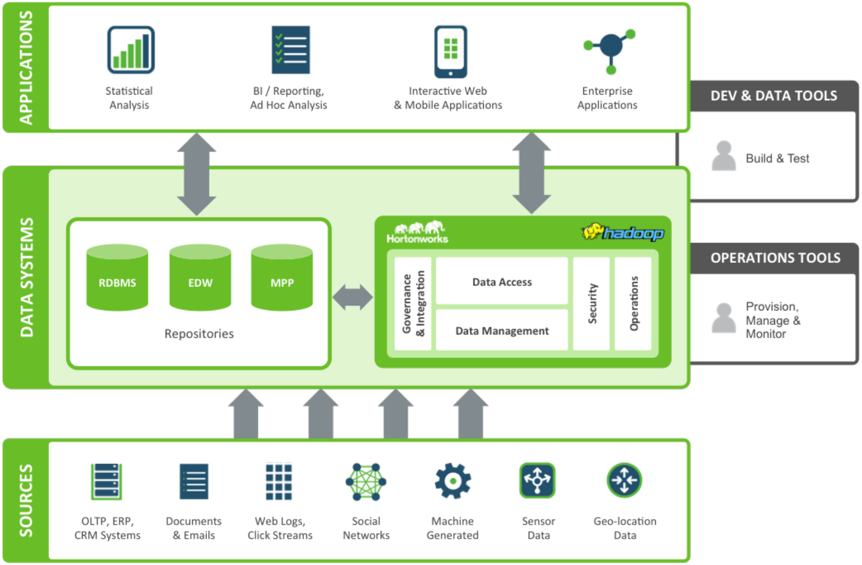
[6] See http://nosql-database.org/ for a big list

Alternative Processing Technology



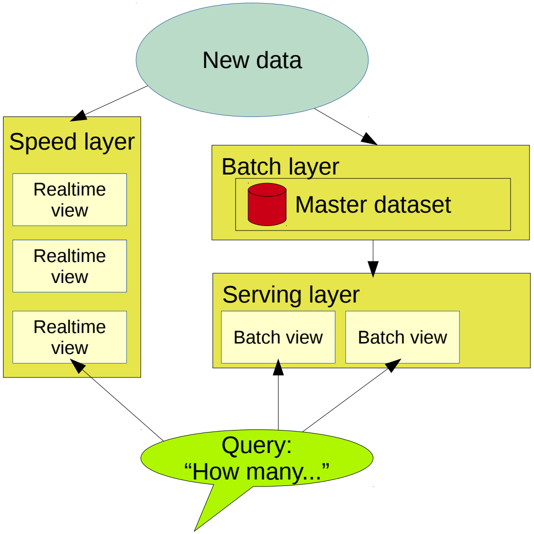
Source: Forrester Webinar. Big Data: Gold Rush Or Illusion? [4]

The Hadoop Ecosystem (of the Hortonworks Distribution)



Source: [20]

The Lambda Architecture [11]



Redrawn figure. Source: [11], Fig. 2.1

- Goal: Interactive Processing
- Batch layer pre-processes data
  - Master dataset is immutable/never changed
  - Operations are periodically performed
- Serving layer offers performance optimized views
- Speed layer serves deltas of batch and recent activities, may approximate results
- Robust: Errors/inaccuracies of realtime views are corrected in batch view

Summary

- Data-cleaning and ingestion is a key to successful modeling
- Big data can be considered to be immutable (remember: data lake)
- Data models describe how information is organized
  - Various I/O middleware, relational model
  - NoSQL: Column, document, key-value, graphs
- Semantics describe operations and behavior, e.g., POSIX, ACID, BASE
- Process models and programming paradigms describe how to transform and analyze data
- Hadoop ecosystem offers means for batch and real-time processing
- Lambda architecture is a concept for enabling real-time processing

Bibliography

4 Forrester Big Data Webinar. Holger Kisker, Martha Bennet. Big Data: Gold Rush Or Illusion?

10 Wikipedia

11 Book: N. Marz, J. Warren. Big Data – Principles and best practices of scalable real-time data systems.

12 https://en.wikipedia.org/wiki/Data_model

13 https://en.wikipedia.org/wiki/Process_modeling

14 https://en.wikipedia.org/wiki/Programming_paradigm

15 https://en.wiktionary.org/wiki/process

16 https://en.wikipedia.org/wiki/Paxos_(computer_science)

17 https://en.wikipedia.org/wiki/Consensus_(computer_science)

20 http://hortonworks.com/blog/enterprise-hadoop-journey-data-lake/

26 Overcoming CAP with Consistent Soft-State Replication https://www.cs.cornell.edu/Projects/mrc/IEEE-CAP.16.pdf