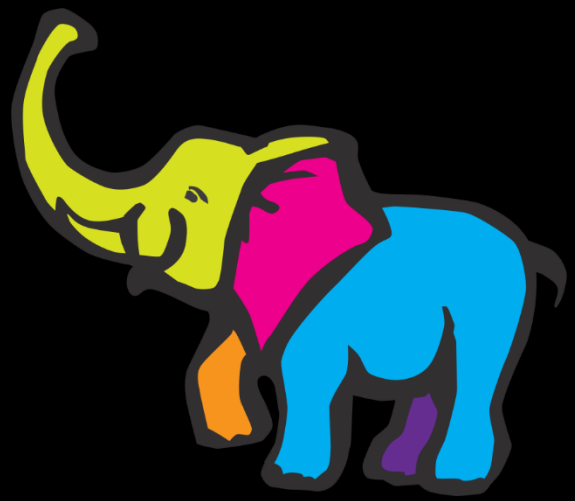
EECS4415: Big Data Systems



Course Review

Data contains value and knowledge

What is the purpose of big data systems?

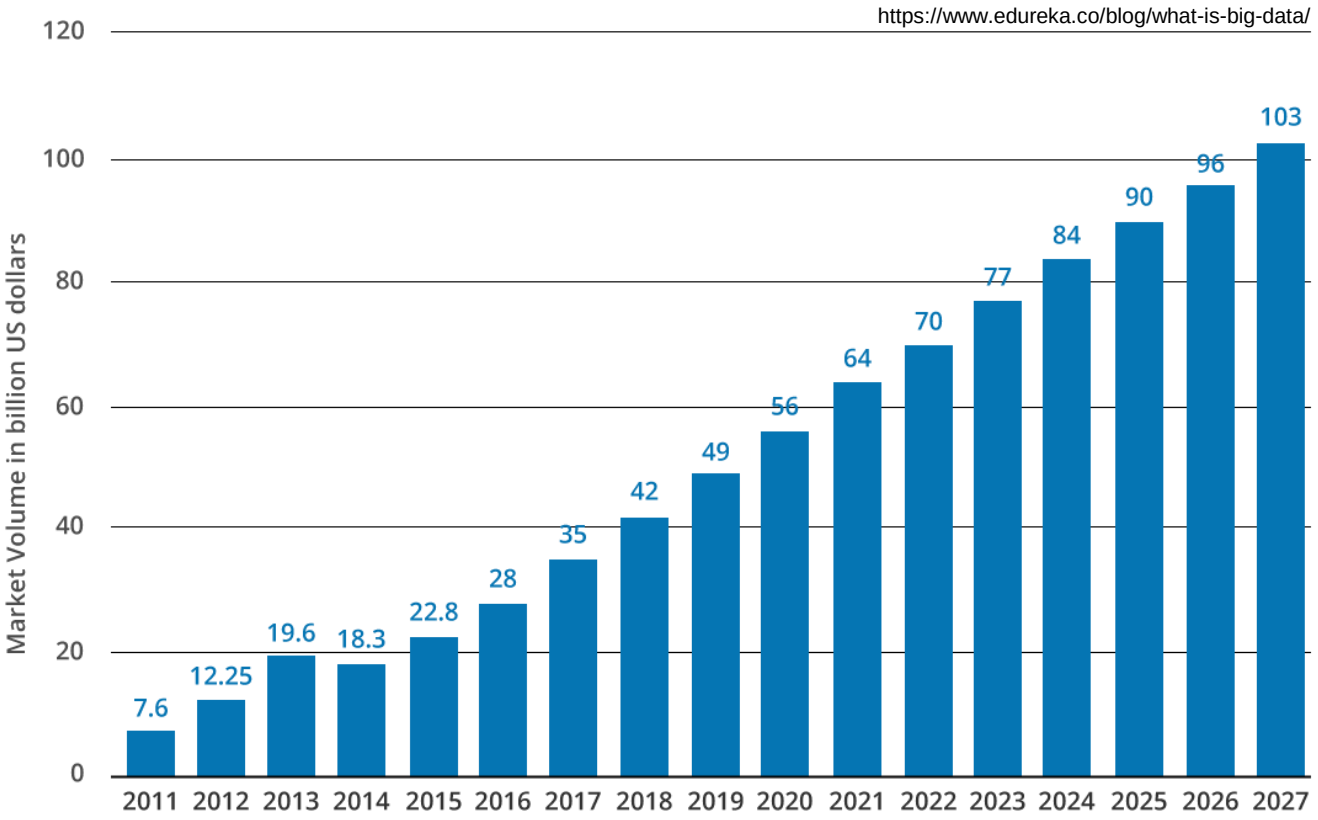
To support analysis and knowledge discovery from very large amounts of data

Big Data Analytics

- But to extract the knowledge data needs to be
  - Stored ← emphasis on this class
  - Managed ← emphasis on this class
  - Analyzed ← emphasis on this class
  - Visualized

Data Analytics ≈ Data Mining ≈ Big Data ≈ Predictive Analytics ≈ Data Science

Demand for Big Data Skills

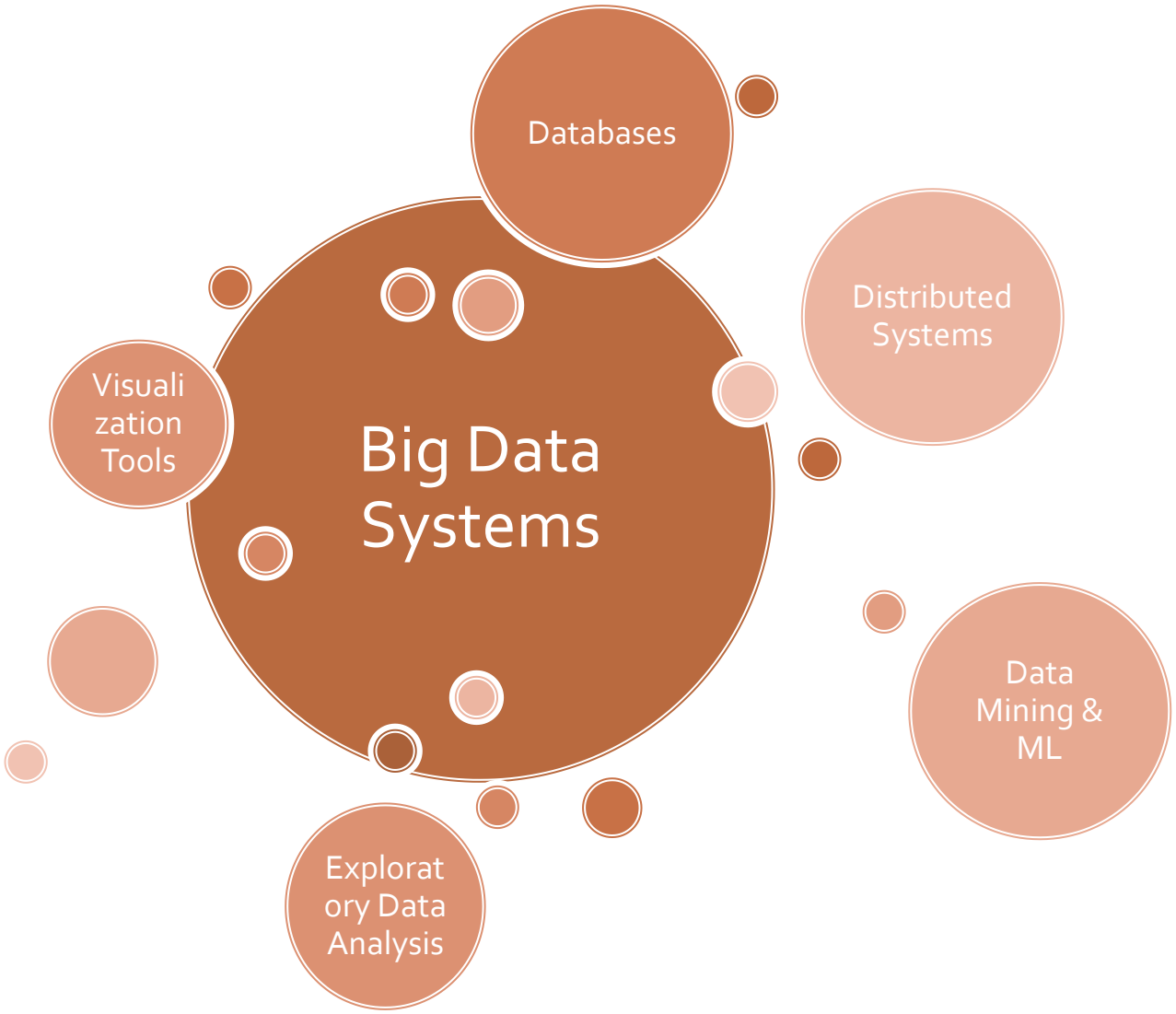


Growing market revenue of Big Data in billion U.S. dollars from the year 2011 to 2027

This Class: EECS4415

- This class stressed more on
  - Big Data Analytics Architectures
  - Storage Systems
  - Distributed Computing Platforms
  - Algorithms, Scalability Issues

Course Intellectual Content



What have you learned?

- **How to process different types of data:**
  - Small/Large size data
  - Structured/Semi-structured/No-structure data
  - Batch/streaming data
- **How to use different models of computation:**
  - Single machine in-memory
  - Distributed (MapReduce)
  - Streams and online algorithms

What have you learned?

- Hands-on experience working with systems and **tools for storing and processing big data**:
  - MapReduce/Hadoop
  - Hive/BigQuery
  - Apache Spark
  - OpenRefine
  - …
- … and more
  - python
  - tf.idf, skip-grams, sentiment analysis, …

Elements of a BD Analytics project

Need for data collection
Need for data storage
Need for data analysis
Need for data visualization (optionally)

…but, more of an iterative process than a sequence

Data Driven Organizations (DDOs)

How non-DDOs make decisions?

- Intuition
- Ad-hoc or based on few customers feedback
- Look at competition
- Try to be different
- Based on assumptions, that may be wrong
- Without knowing how to validate if it was the right decision

What do DDO's do?

- Make decisions based on data not intuition
- More precise on what they want to achieve
- Measure and validate with data

How DDOs do it?

DDO's

- collect data
- make decisions based on data, not intuition
- use data to drive applications

To be a DDO, you need an efficient way of storing and retrieving data

Challenge

- A variety of solutions/technologies available
- There is no one solution/technology that solves all possible data analytics problems
- Most solutions solve a range of problems, but are outstanding on a specific type

How to map problems to DDO solutions?

How to compare alternative DDO solutions?

Need for a Reference Model

Purpose of the Reference Model

- Provides **a framework** for
  - **understanding** your needs
  - **comparing** solutions
- Not complete, but gives an approach to understanding data analytics systems

Dimensions

Data

What characteristics should be considered with respect to **data**?

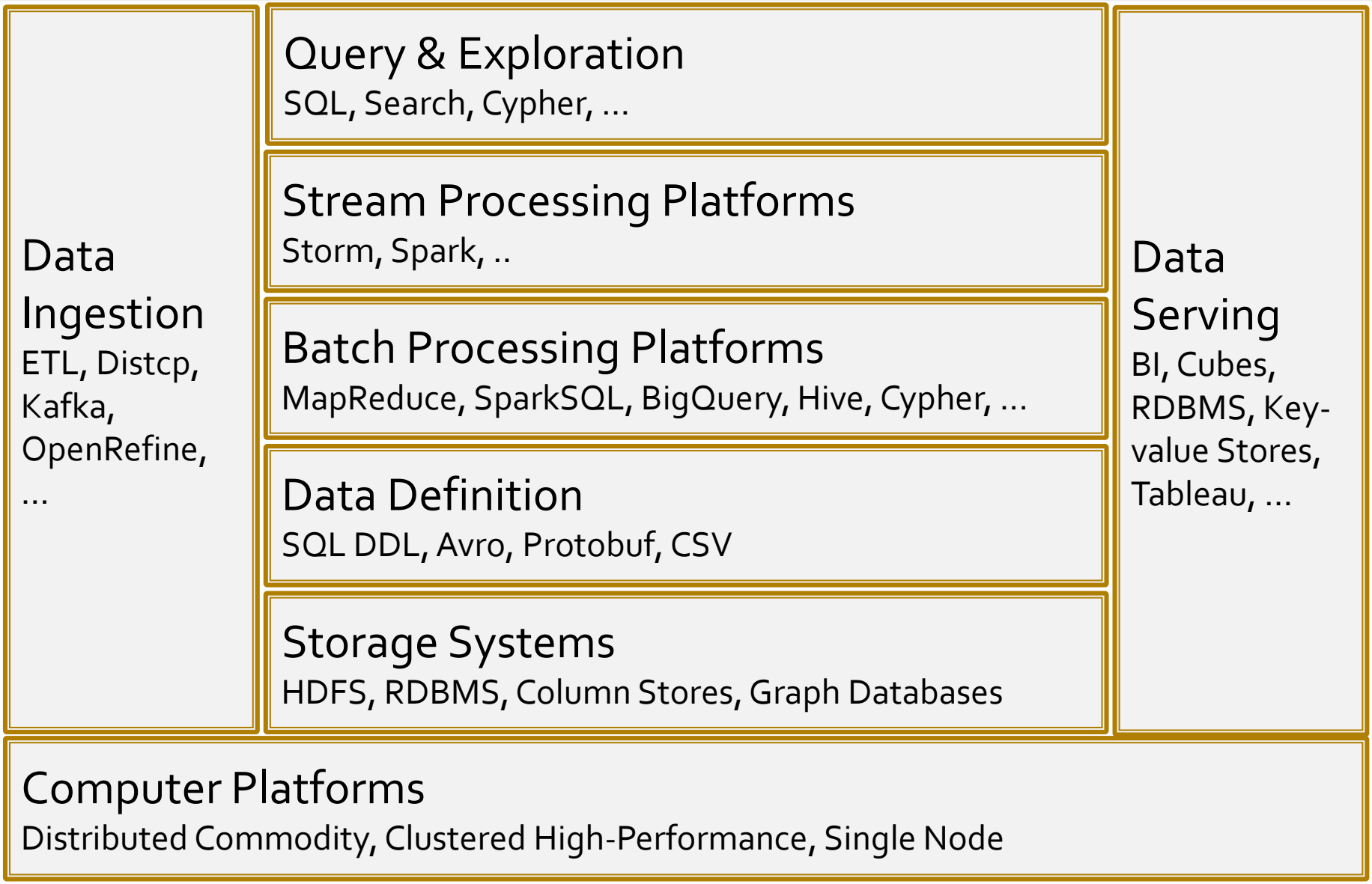
Processing

What characteristics should be considered with respect to **processing**?

Other dimensions (not covered):

cost, implementation complexity

Big Data Technology & Analytics



Data Ingestion and Data Quality

Big Data Technology & Analytics

Data Ingestion
ETL, Distcp, Kafka, OpenRefine, …

Query & Exploration
SQL, Search, Cypher, …

Stream Processing Platforms
Storm, Spark, ..

Batch Processing Platforms
MapReduce, SparkSQL, BigQuery, Hive, Cypher, …

Data Definition
SQL DDL, Avro, Protobuf, CSV

Storage Systems
HDFS, RDBMS, Column Stores, Graph Databases

Data Serving
BI, Cubes, RDBMS, Key-value Stores, Tableau, …

Computer Platforms
Distributed Commodity, Clustered High-Performance, Single Node

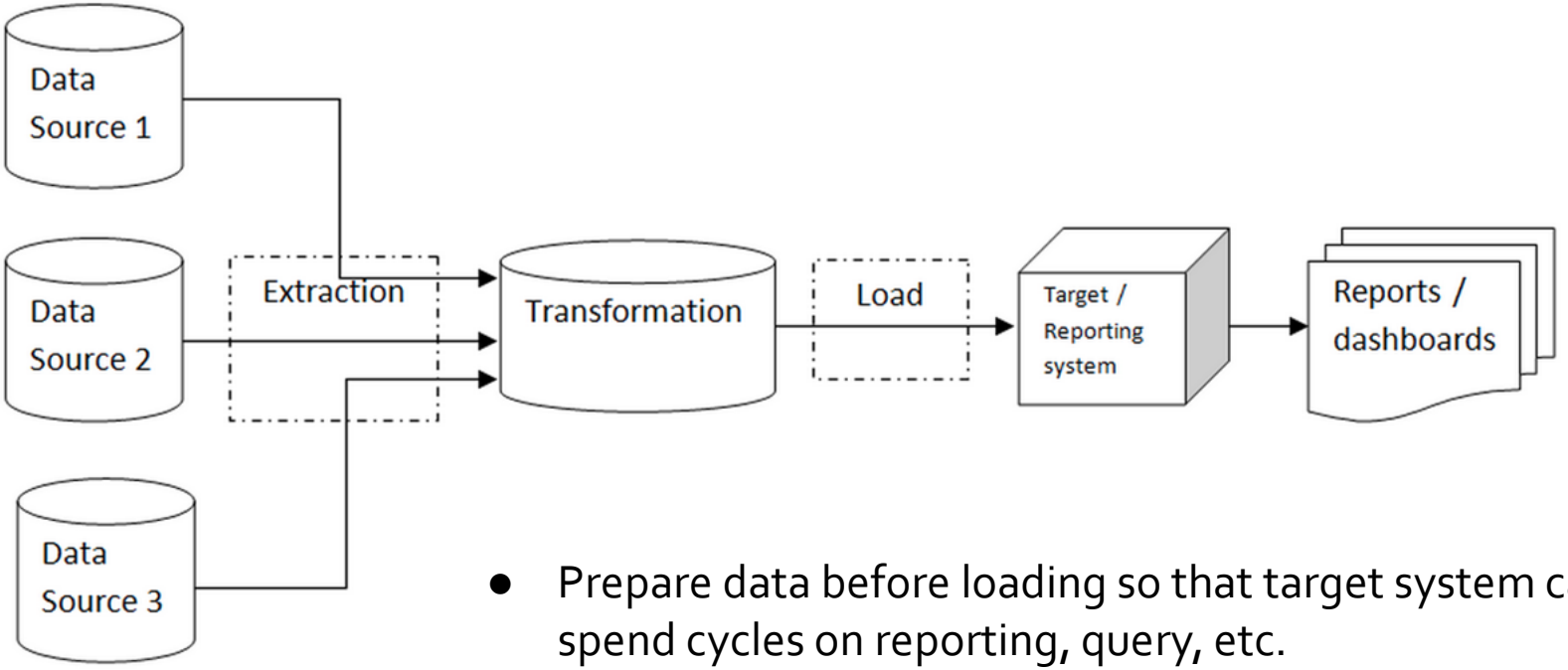
Bottom Line

Analytics solutions start with **data ingestion**

Data integration challenges:

- **volume** (many similar integrations)
- **variety** (many different integrations)
- **velocity** (batch v.s real-time)
- (or all of the above)

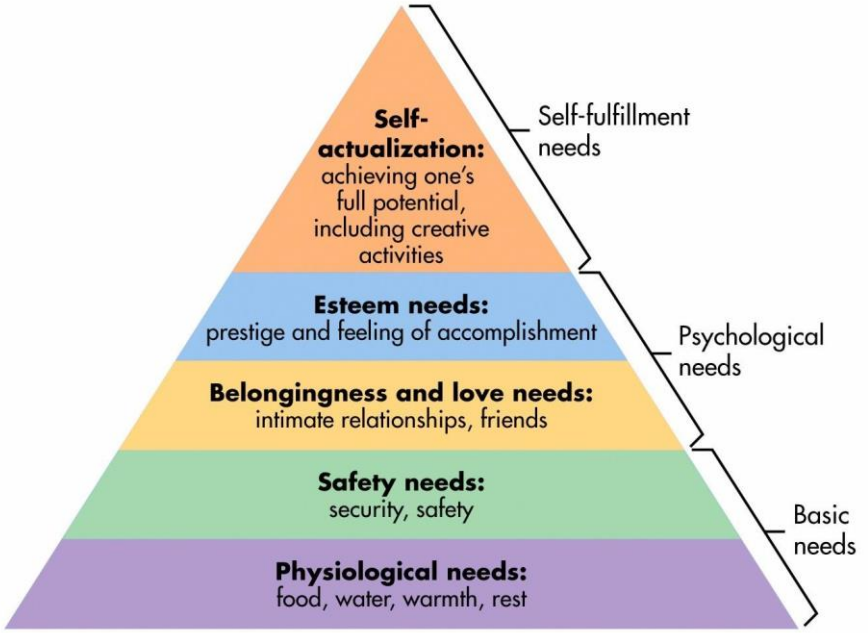
ETL: Extract, Transform, Load



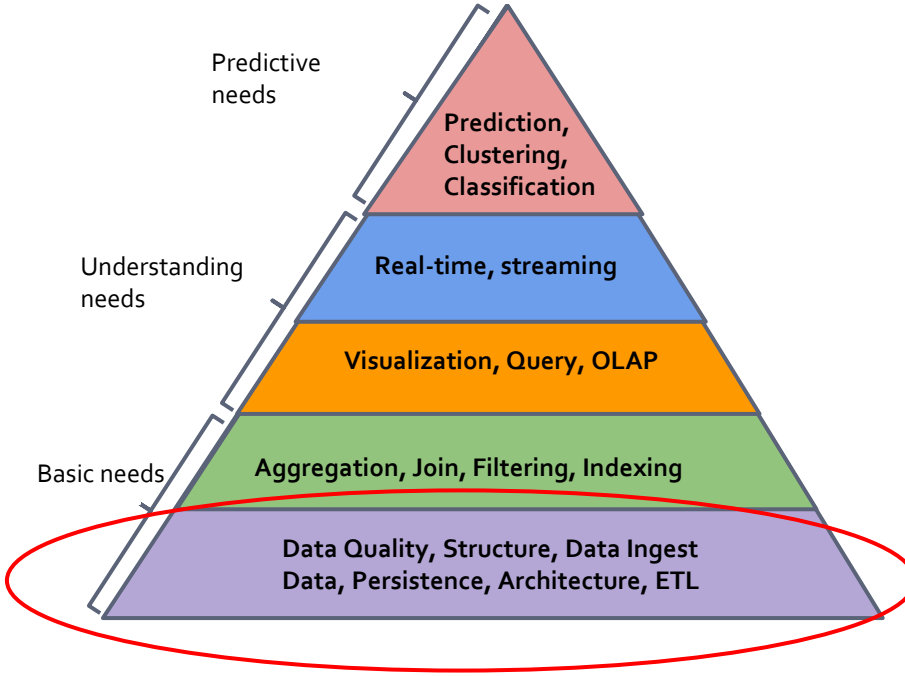
- Prepare data before loading so that target system can spend cycles on reporting, query, etc.
- Requires transforms to know what reporting, query to enable

Needs of Data Analytics

Maslow's hierarchy of needs*

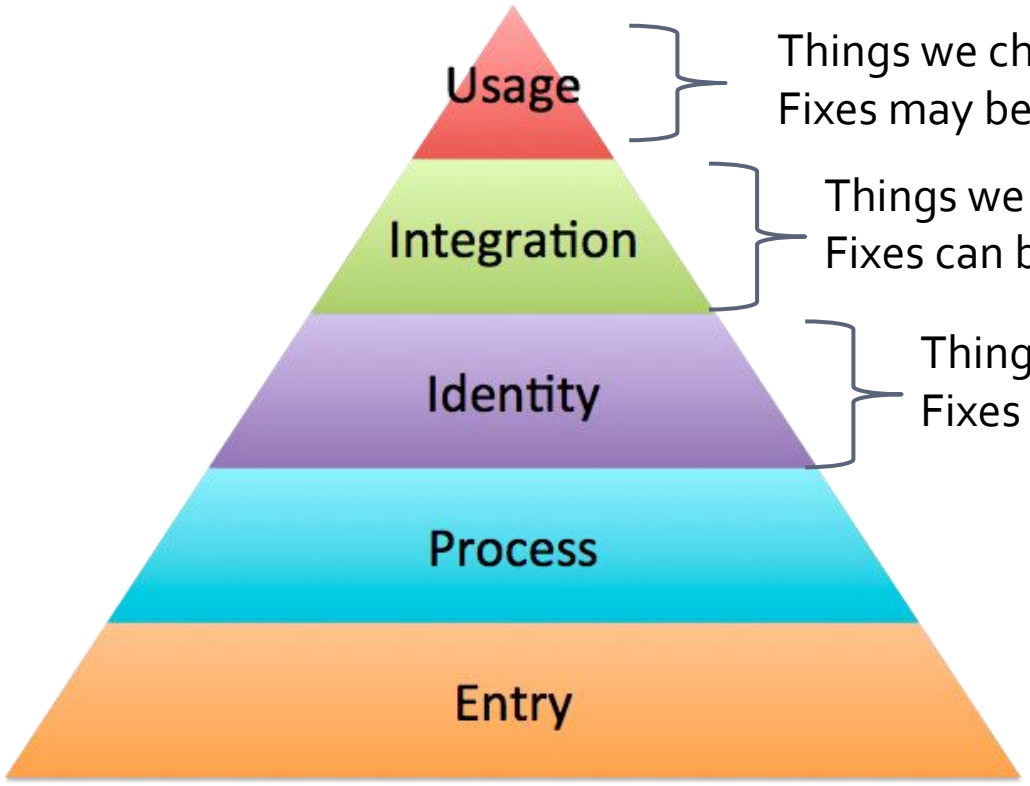


Hierarchy of effective analytics



* A theory in psychology proposed by Abraham **Maslow** in 1943. Needs lower down in the hierarchy must be satisfied before individuals can attend to needs higher up.

Data Quality as a Hierarchy



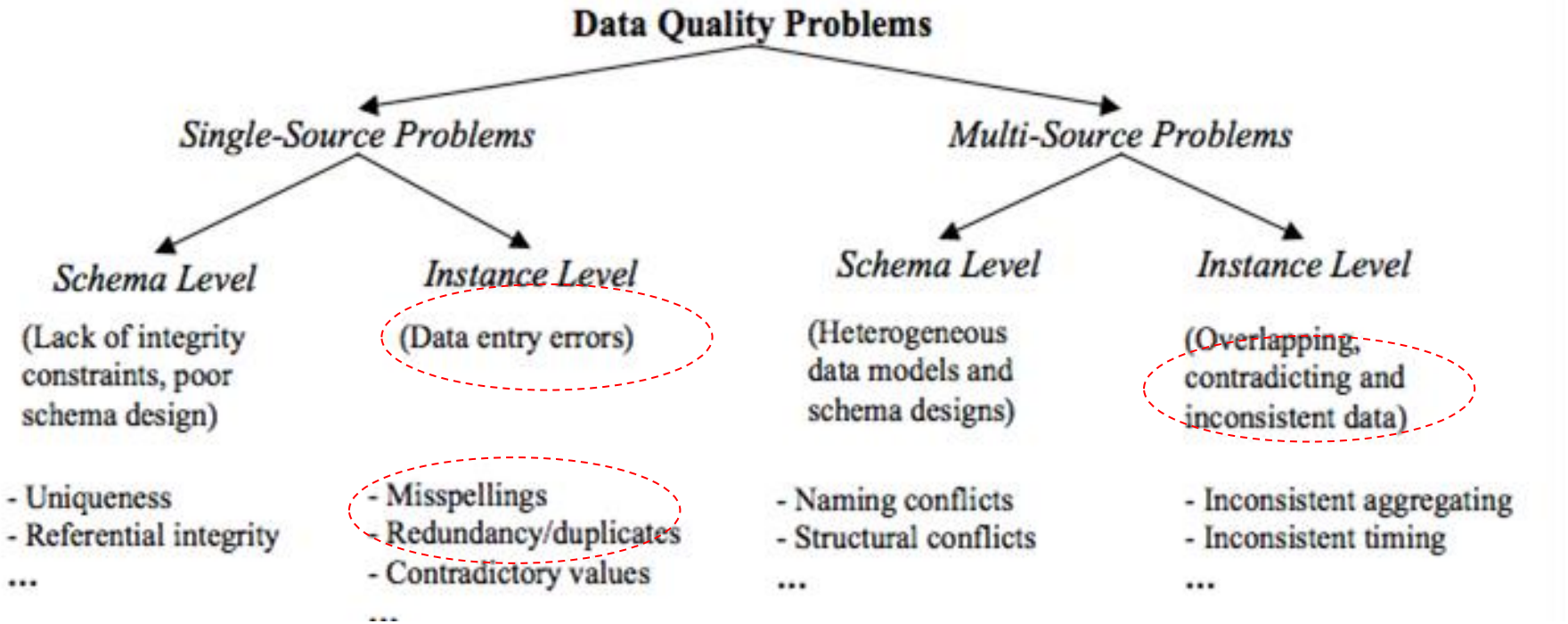
Things we check **in the organization**.
Fixes may be non-technical.

Things we check **in architecture**.
Fixes can be costly!

Things we check across **many data sets**.
Fixes may need extra intelligence.

Things we check **in single record sets** and **data streams**.
Fixes can be automatic and independent.

Types of Data Quality Problems



Quality Evaluation Framework

Observation

- It's too expensive to clean **all** the data **every** way
- How do we decide what to clean?

We need **a framework** that helps to:

- Determine what issues might occur in the data
- Weight the criticality of the issues
- Profile the data to score quality

The framework allows:

- to approach quality as an ever-increasing **standard**
- To **prioritize** data cleaning activities

Computing Platforms, Storage Systems, Data Definition

Big Data Technology & Analytics

Data Ingestion
ETL, Distcp, Kafka, OpenRefine, …

Query & Exploration
SQL, Search, Cypher, …

Stream Processing Platforms
Storm, Spark, ..

Batch Processing Platforms
MapReduce, SparkSQL, BigQuery, Hive, Cypher, …

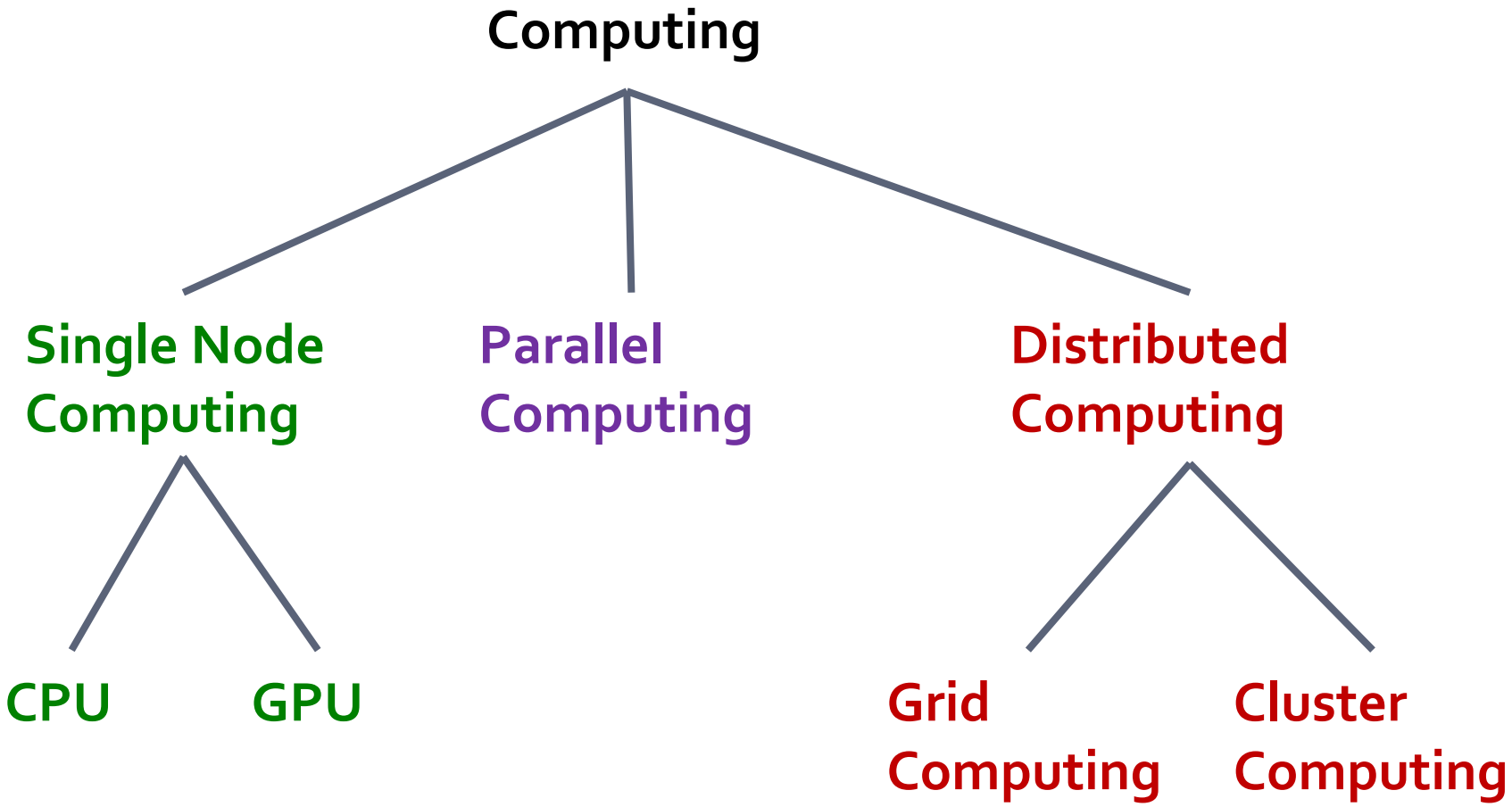
Data Definition
SQL DDL, Avro, Protobuf, CSV

Storage Systems
HDFS, RDBMS, Column Stores, Graph Databases

Data Serving
BI, Cubes, RDBMS, Key-value Stores, Tableau, …

Computing Platforms
Distributed Commodity, Clustered High-Performance, Single Node

Computing Platforms



Data Storage:
Data Warehouse vs Data Lakes

Data Warehouse vs Data Lake

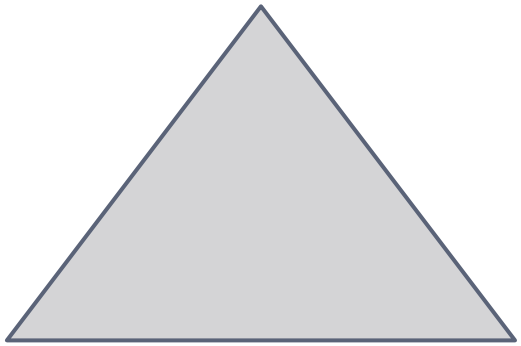
Data Warehouse

- Data Transformed to defined schema
- Loaded when usage identified
- Allows for quick response of defined queries

Data Lake

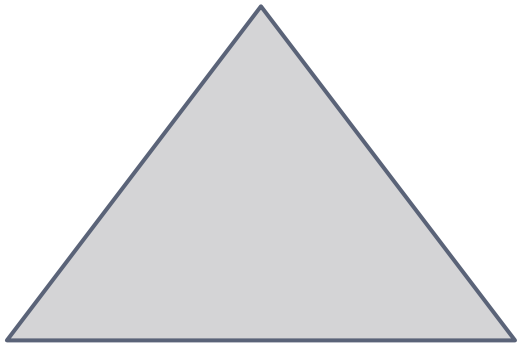
- Many data sources
- Retain all data
- Allows for exploration
- Apply transform as needed
- Apply schema as needed

Data Warehouse Model



Big Data Model

Master Data
(fact based, Immutable, Dimensions)



Transaction Data
(Log items)

Analytics Data
(Aggregates, Roll-ups)

Storage Systems

DBMS: Relational and Columnar

Two kinds of ***database management systems***

- **Relational Databases**
  - Presents via Declarative Query Languages
  - Organize underlying storage **row-wise**
    - Sometimes column-wise
- **Columnar Databases**
  - Presents via API and Declarative Query Languages
  - Organize underlying storage **column-wise**

NoSQL and HDFS

Two approaches for ***distributed data storage***

- **HDFS (Hadoop Distributed File System)**
  - Presents like a local filesystem
  - Distribution mechanics handled automatically
- **NoSQL Databases (Key/Value Stores)**
  - Typically store records as "**key-value pairs**"
  - Distribution mechanics tied to record keys

OS and SDS Storage

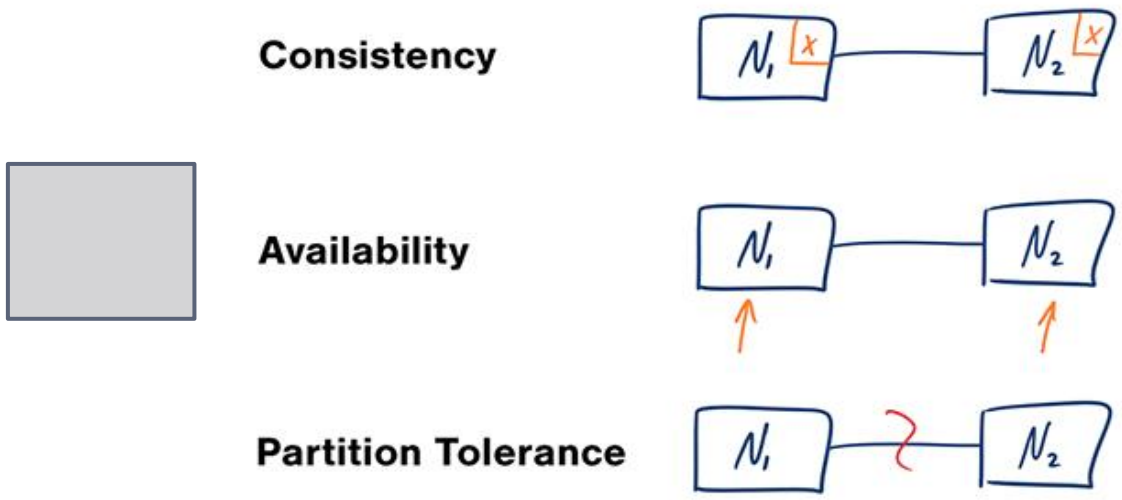
Two more concepts

- **Object Storage (OS)**: as a new abstraction for storing data
- **Software Defined Storage (SDS)**: An architecture that enables cost effective, scalable, highly available (HA) storage systems

Combining OS and SDS provides an efficient solution for certain data applications

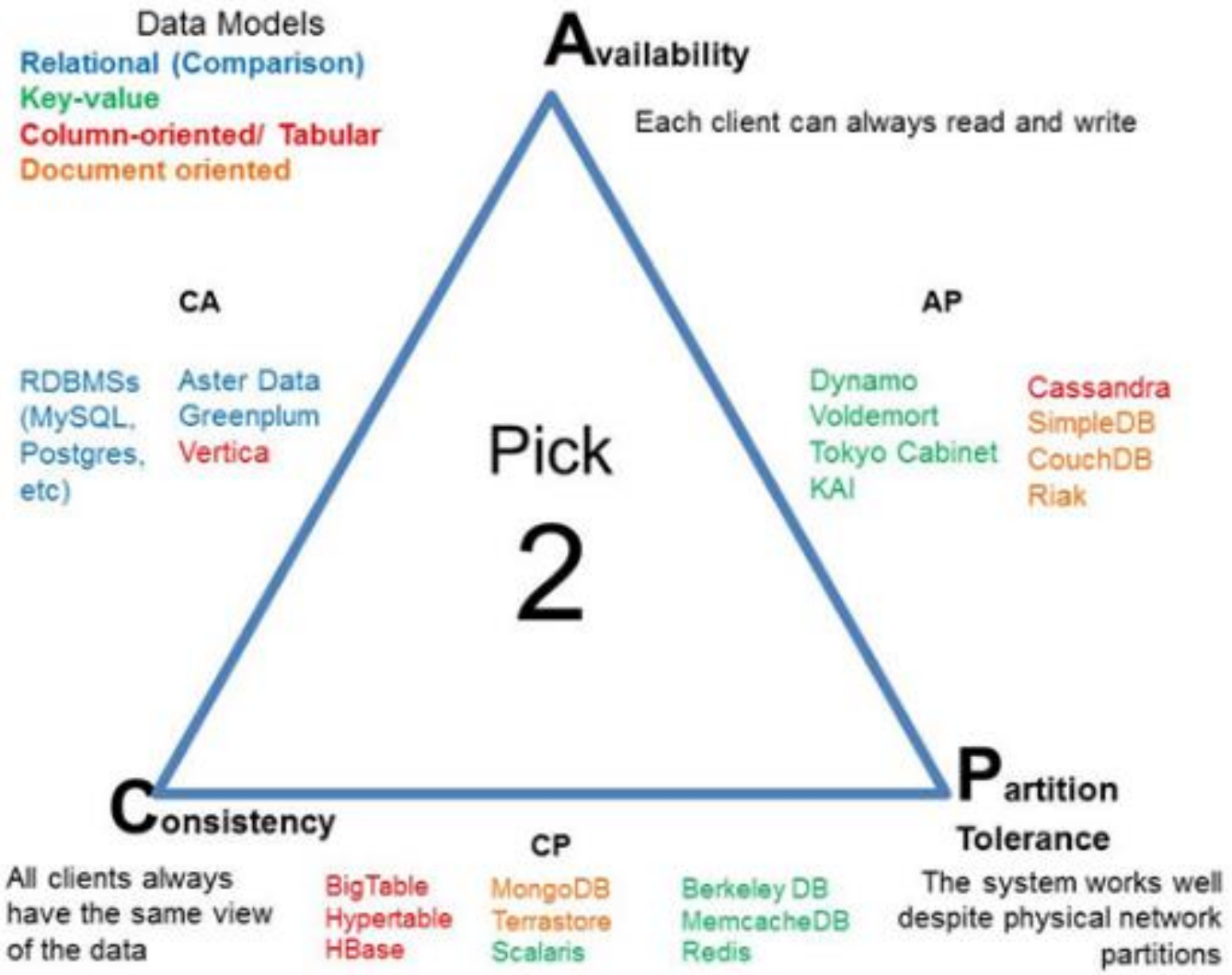
CAP Theorem

It is impossible for a distributed data store to simultaneously provide more than two out of the following three guarantees



Consistency: Every read receives the most recent write or an error
Availability: Every request receives a (non-error) response – without guarantee that it contains the most recent write
Partition tolerance: The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes

CAP Theorem (Simplified)



Moving Large Data Considerations

Moving Large Data

- If data is distributed how can you leverage parallelism?
- What kind of source and sink is involved?
- How do you use network bandwidth efficiently?
- How to handle different formats and structures?
- Large files take a long-time, how are failures handled?

As a data scientist you need to understand how to think about data transfer and movement

Example Tools

Tool	What
Sqoop	RDBMS, BDW to Hadoop
distcp2	HDFS to HDFS copy
Rsync	FS to FS copy, FS to FS synchronization

SQL DDL, Avro, Protobuf, CSV

Data Definition

When is Schema Applied?

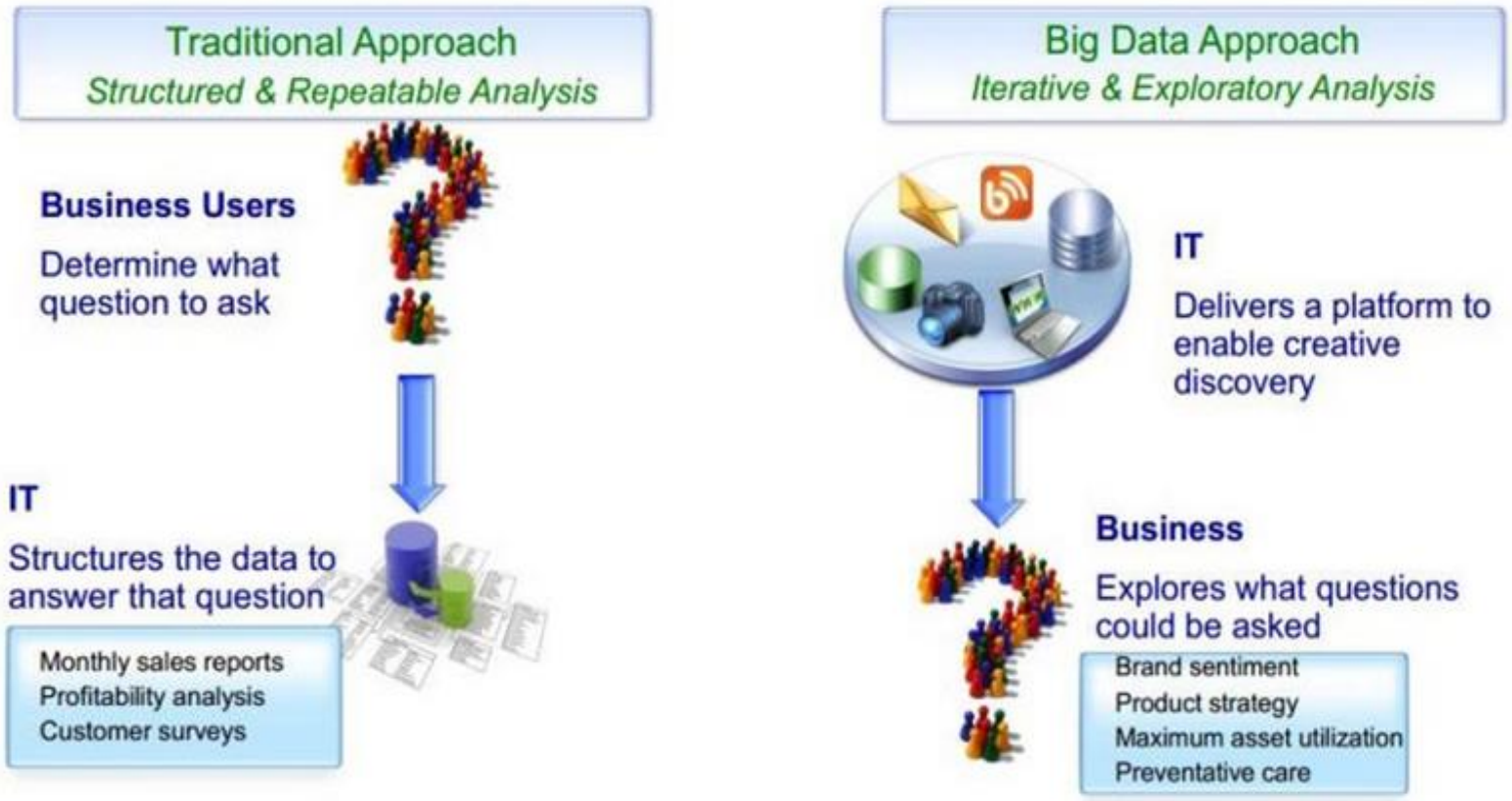
- Schemas represent the logical view of data
- We can apply them
  - When data is **written (schema-on-write)**
  - When data is **read (schema-on-read)**
- The application of schema comes with trade-offs

BigTable: a Hybrid Approach

- Column-Family Database
  - Organize data into a hierarchy
    - Columns → record details
    - Column families → groups of columns
  - Column families are **schema-on-write**
  - Columns are **schema-on-read**
    - Can add columns, interpret bytes variably
- Examples:
  - Apache HBase
  - Apache Cassandra

Big Data Architectures

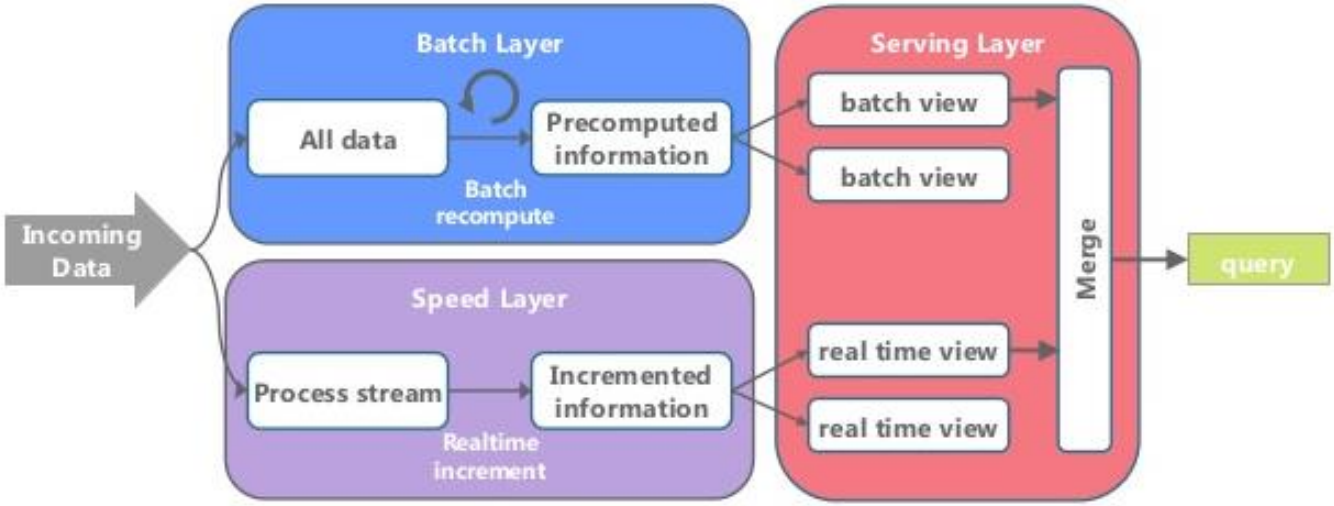
Difference in Approach



Notice the difference!

Big Data Analytics Architecture

Example:
Lambda Architecture



Other examples:
Kappa Architecture
Netflix Architecture

Processing Platforms

Big Data Technology & Analytics

Data Ingestion
ETL, Distcp, Kafka, OpenRefine, …

Query & Exploration
SQL, Search, Cypher, …

Stream Processing Platforms
Storm, Spark, ..

Batch Processing Platforms
MapReduce, SparkSQL, BigQuery, Hive, Cypher, …

Data Definition
SQL DDL, Avro, Protobuf, CSV

Storage Systems
HDFS, RDBMS, Column Stores, Graph Databases

Data Serving
BI, Cubes, RDBMS, Key-value Stores, Tableau, …

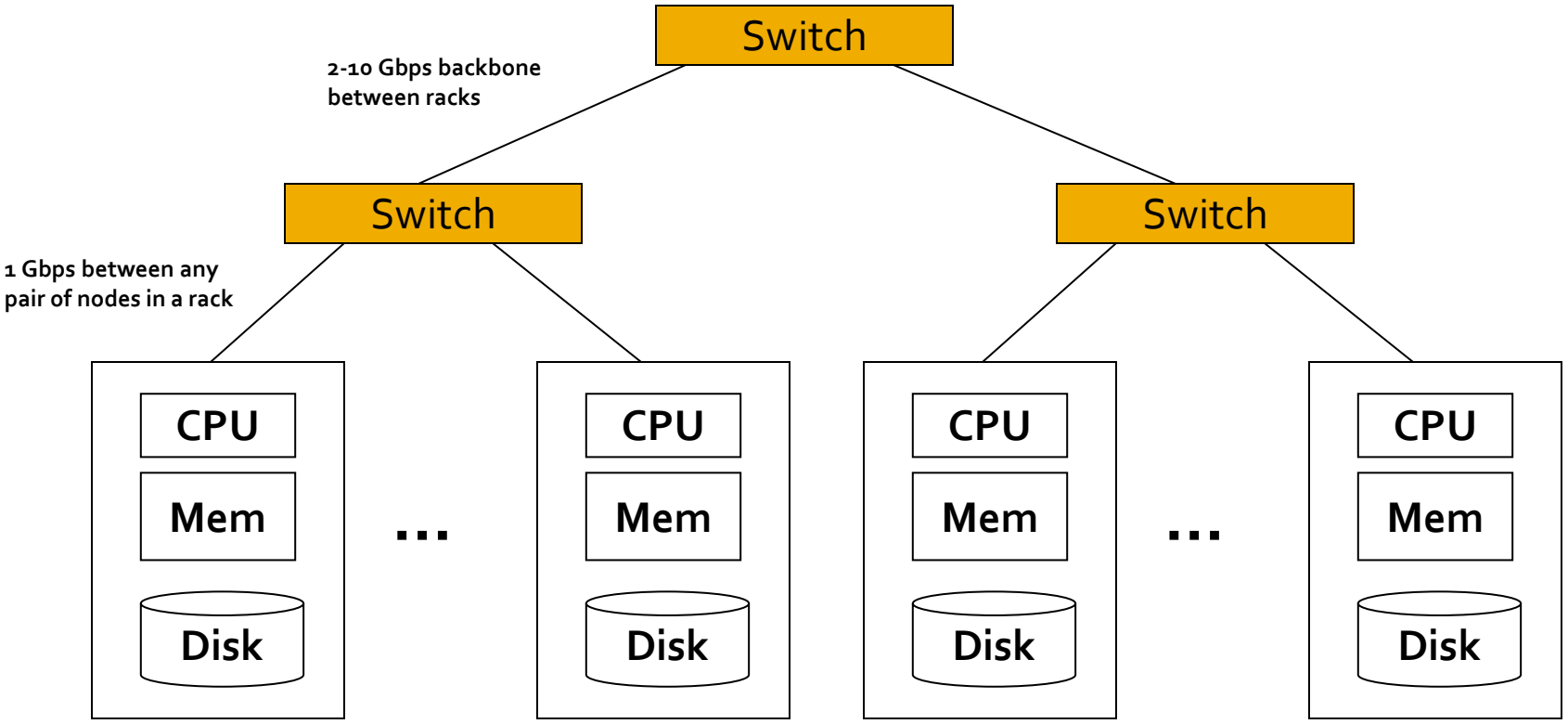
Computing Platforms
Distributed Commodity, Clustered High-Performance, Single Node

Processing Platforms

- Batch Processing
  - Google GFS/MapReduce (2003)
  - Apache Hadoop HDFS/MapReduce (2004)
- SQL
  - BigQuery (based on Google Dremel, 2010)
  - Apache Hive (HiveQL) (2012)
- Streaming Data
  - Apache Storm (2011) / Twitter Huron (2015)
- Unified Engine (Streaming, SQL, Batch, ML)
  - Apache Spark (2012)

Map-Reduce and the New Software Stack

Cluster Architecture



Each rack contains 16-64 nodes

In 2011 it was guestimated that Google had 1M machines, http://bit.ly/Shh0RO

Large-scale Computing

- **Large-scale computing** for **data analytics** problems on **commodity hardware**
- **Challenges:**
  - **How can we store large data?**
  - **How can we distribute computation?**
  - **How can we make it easy to write distributed programs?**
  - **How can we manage machine failures?**

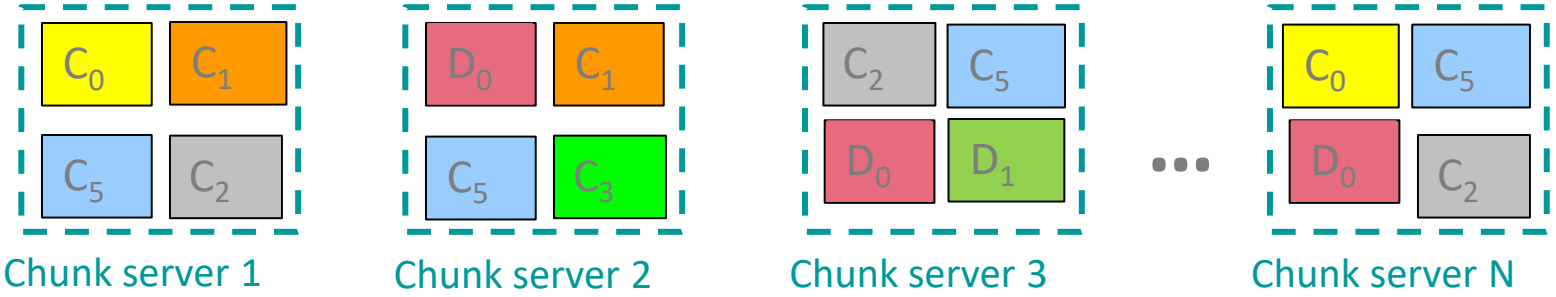
Idea and Solution

- **Key Ideas:**
  - Store files multiple times for reliability
  - Bring computation close to the data

- **Storage Infrastructure:** **Distributed File system**
  - Google: GFS. Hadoop: HDFS
- **Programming Model:** **Map-Reduce**
  - Google's computational/data manipulation model
  - Elegant way to work with big data

Distributed File System

- **Reliable distributed file system**
- Data kept in “chunks” spread across machines
- Each chunk replicated on different machines
  - Seamless recovery from disk or machine failure



Bring computation directly to the data!

Chunk servers also serve as compute servers

MapReduce: Overview

- **Sequentially read a lot of data**
- **Map: Extract something you care about**
- **Group by key: Sort and Shuffle**
- **Reduce: Aggregate, summarize, filter or transform**
- **Write the result**

Outline stays the same, **Map** and **Reduce** steps change to fit the problem

More Specifically

- **Input:** a set of key-value pairs
- Programmer specifies two methods:
  - **Map(k, v)** $\rightarrow$ <k', v'>*
    - Takes a key-value pair and outputs a set of key-value pairs
      - E.g., key is the filename, value is a single line in the file
    - There is one Map call for every *(k,v)* pair
  - **Reduce(k', <v'>*)** $\rightarrow$ <k', v''>*
    - **All values *v'* with same key *k'* are reduced together and processed in *v'* order**
    - There is one Reduce function call per unique key *k'*

Map-Reduce: Environment

Map-Reduce environment takes care of:

- **Partitioning** the input data
- **Scheduling** the program's execution across a set of machines
- Performing the **group by key** step
- Handling machine **failures**
- Managing required inter-machine **communication**

Processing Platforms

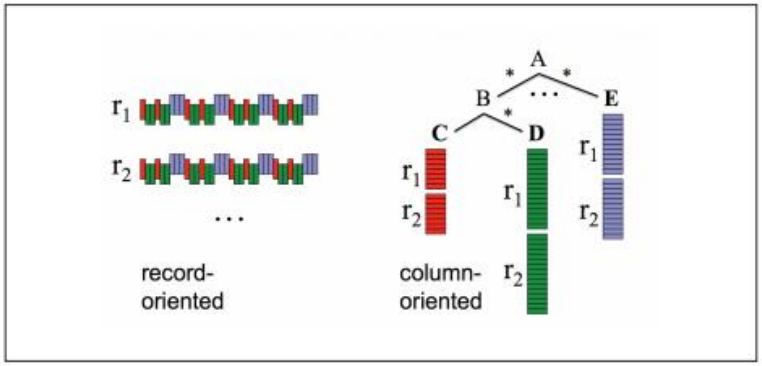
- Batch Processing
  - Google GFS/MapReduce (2003)
  - Apache Hadoop HDFS/MapReduce (2004)
- SQL
  - BigQuery (based on Google Dremel, 2010)
  - Apache Hive (HiveQL) (2012)
- Streaming Data
  - Apache Storm (2011) / Twitter Huron (2015)
- Unified Engine (Streaming, SQL, Batch, ML)
  - Apache Spark (2012)

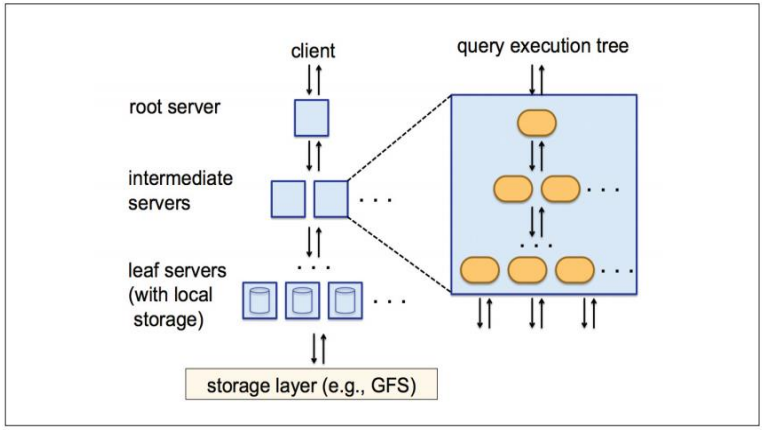
Processing Platforms: Structured Data

Processing Structured Data

Distributed computation of **interactive queries over structured data**

- **Dremel/BigQuery**
  - Nested Columnar Storage
  - Hierarchical Query Processing
  - Dealing with disk failures and slow/straggling jobs





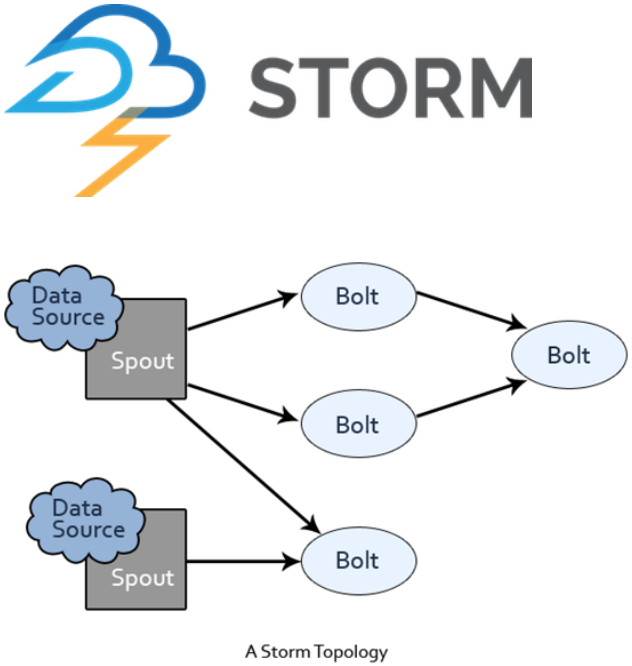
Tree architecture of Dremel

Processing Platforms: Streaming Data

Processing Streaming Data

Scalable analytics over **streaming data**

- **Apache/Twitter Storm**
  - **Topology**: Acyclic Graph
  - **Spouts**: Sources of Data
  - **Bolts**: Transformations
- **Twitter Heron**
  - Next generation of Storm
  - Faster
  - Backwards compatibility

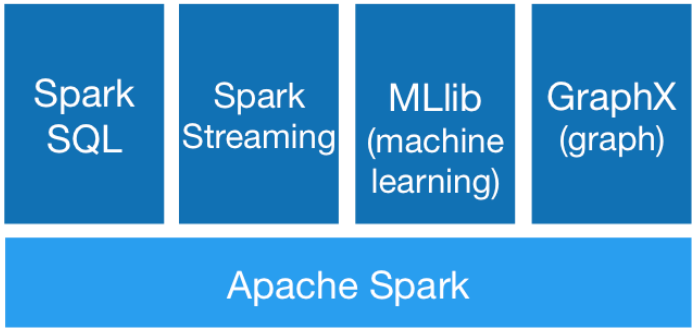


A Storm Topology

Processing Platforms: A Unified Engine

Processing Different Types of Data

- **Apache Spark: A Unified Engine**
  - Efficient Data Sharing
  - Spark Programming Model: RDDs
  - Resilient Distributed Datasets (RDDs)
    - Collections of objects stored in RAM or disk across cluster
    - Built via parallel transformations (map, filter, …)
    - Automatically rebuilt on failure
- Distributed Computation of
  - complex, multi-pass algorithms
  - interactive ad-hoc queries
  - real-time stream processing
  - ML models



Data Serving

Big Data Technology & Analytics

Data Ingestion
ETL, Distcp, Kafka, OpenRefine, …

Query & Exploration
SQL, Search, Cypher, …

Stream Processing Platforms
Storm, Spark, ..

Batch Processing Platforms
MapReduce, SparkSQL, BigQuery, Hive, Cypher, …

Data Definition
SQL DDL, Avro, Protobuf, CSV

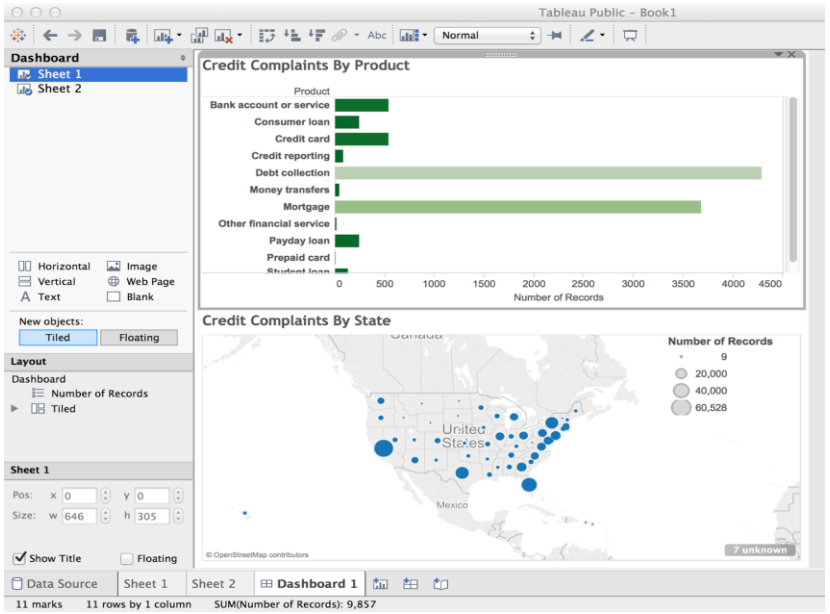
Storage Systems
HDFS, RDBMS, Column Stores, Graph Databases

Data Serving
BI, Cubes, RDBMS, Key-value Stores, Tableau, …

Computer Platforms
Distributed Commodity, Clustered High-Performance, Single Node

Data Serving

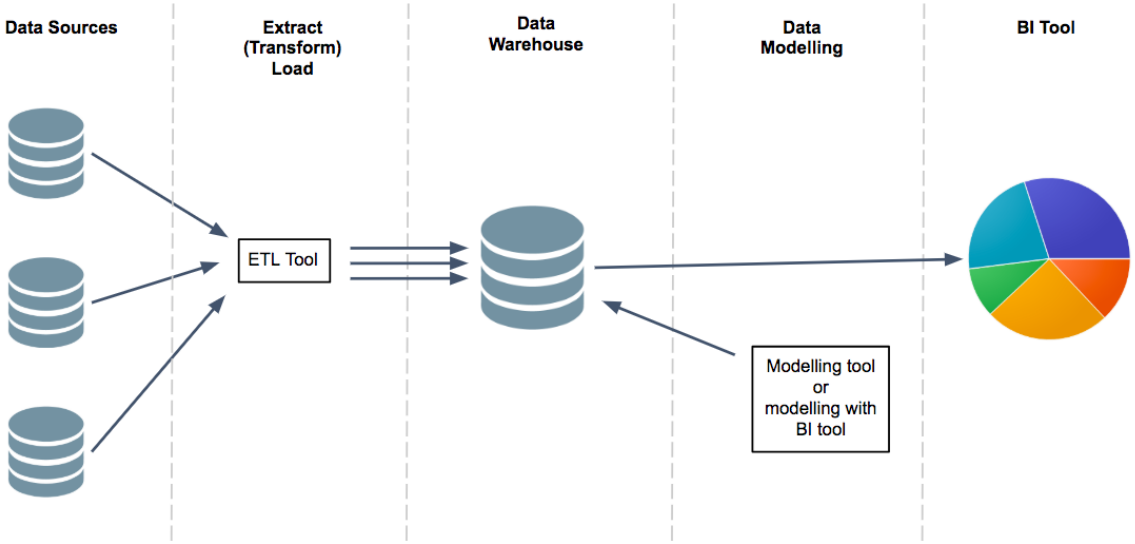
- Reporting is accomplished by **Business Intelligence (BI) tools**



- Real-time analytics are accomplished by **In-application Analytics**

Business Intelligence (BI) Tools

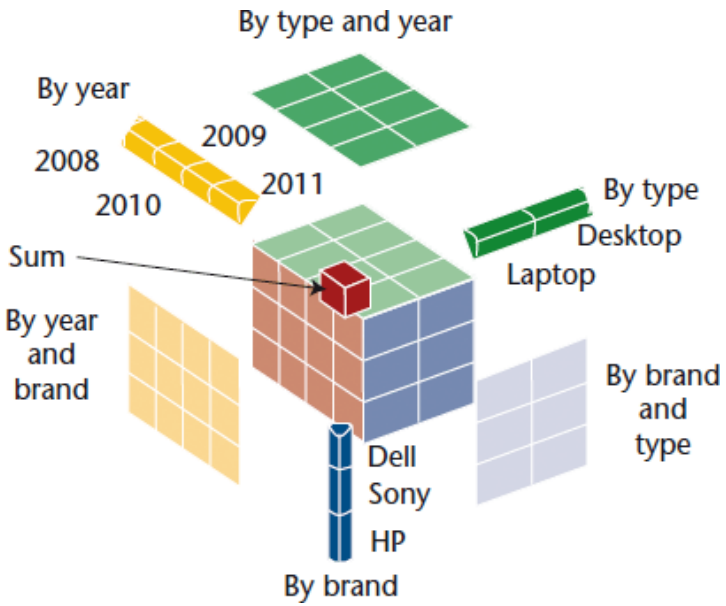
How Does a BI Tool Work?



- Popular Tools
  - MicroStrategy
  - Tableau
  - Pentaho
  - Cognos
  - Spotfire
- Do-It-Yourself
  - HTML5
  - d3 and friends
  - API to get to data

Data Cubes

- An efficient solution for **OLAP** (online analytical processing)
- Operations
  - Slicing
  - Dicing
  - Drill down / Roll Up
  - Pivoting
- Computation and storage intensive
  - different implementations and optimizations



ROLAP / MOLAP

ROLAP

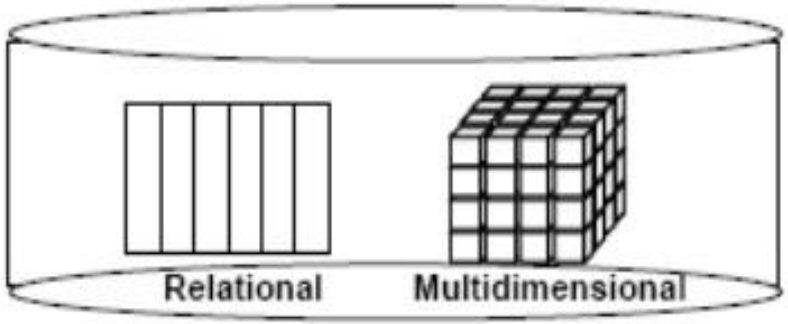
Data stored in relational database

- Performance depends on underlying query
- Generally slower than MOLAP
- Can be partially materialized and partially based on dynamic computation

MOLAP

Data stored in multidimensional array

- Good performance
- Pre-computed
- Proprietary query language and structures



Data cubes and cuboids

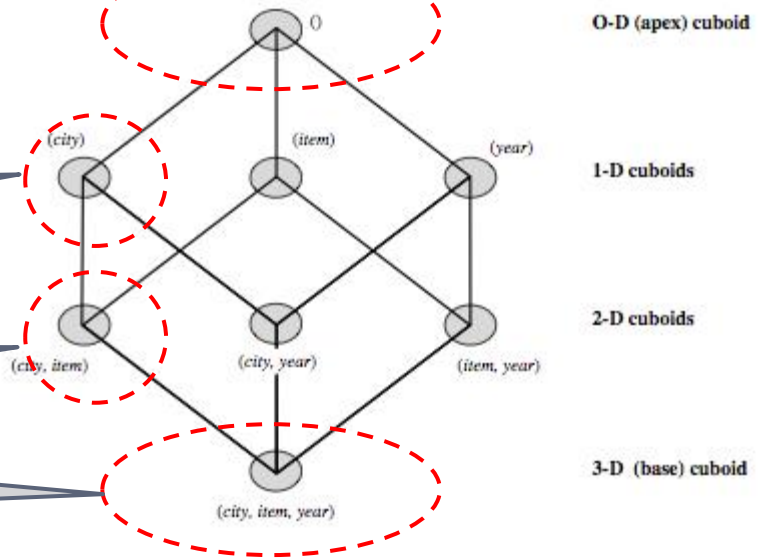
A data cube can be viewed as a lattice of **cuboids**

Most generalized, 1 value with complete aggregate (all cities, all items, all years)

Per city, all items and all years

Per city, per item items, all years

least generalized, each base value: (Chicago, Peppers, 2015)



The Technical Problem

Full cube computation of ***n*-dimensional cube** requires **2^n cuboids** (exponential to the number of dimensions) and is thus very expensive

Questions:

- How can we reduce the cost of computing a cube?
  - iceberg cuboids
  - cuboid shells
  - shell fragments
- What are the trade-offs?
  - Identify the right cuboids
  - Some queries cannot be answered
  - Costly updates

In-Application Analytics

In-application Analytics

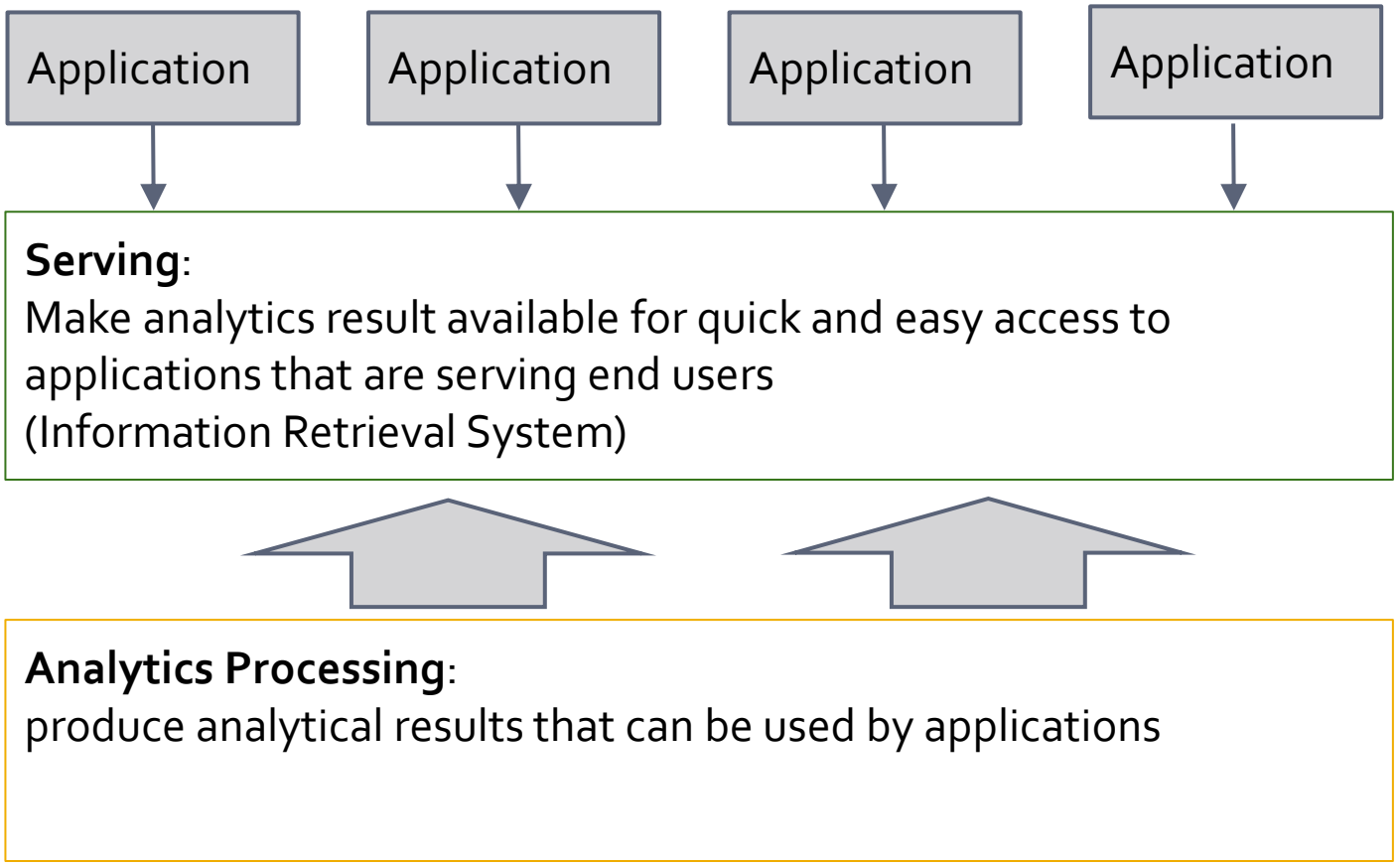
Face detection (FB tag friends)

User Engagement (retweets, likes...)

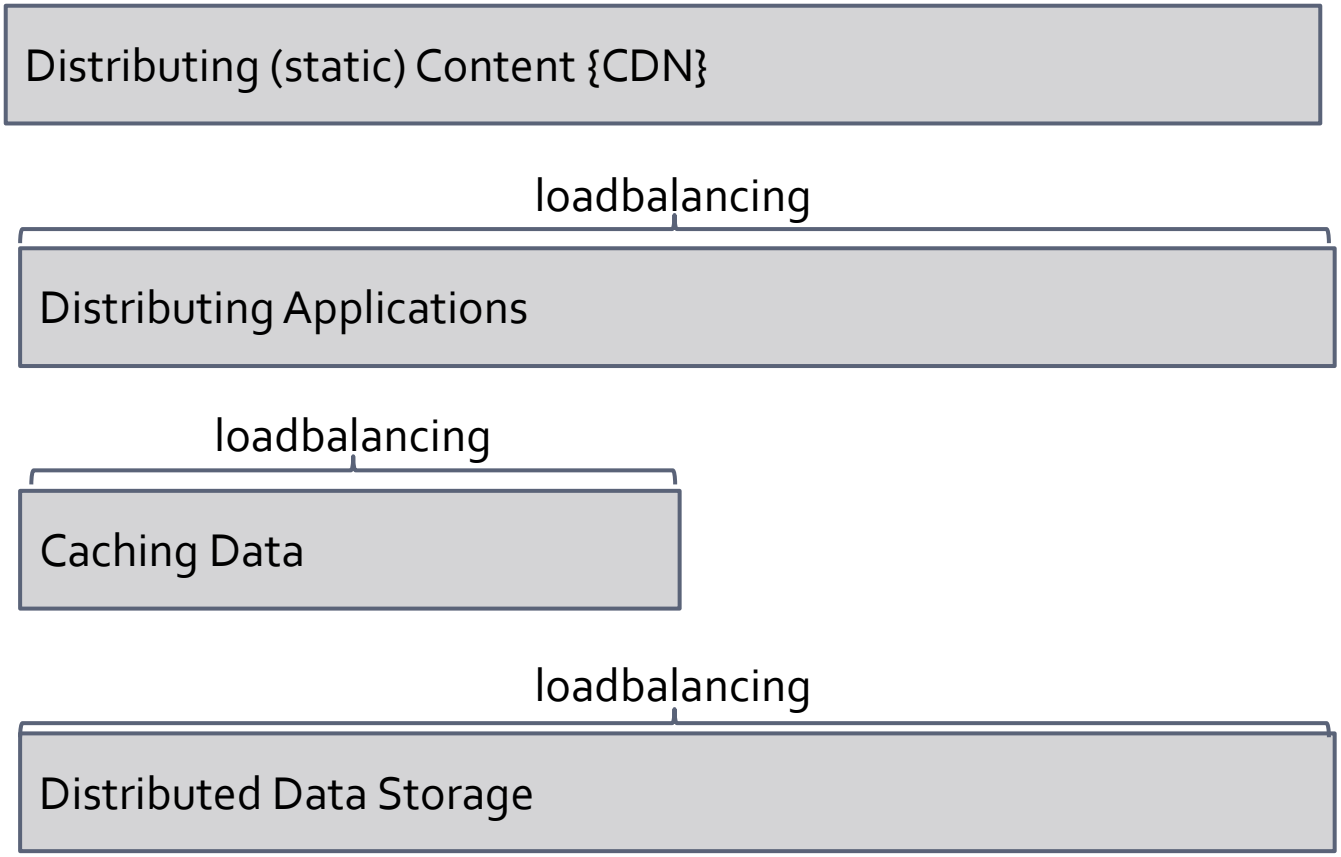
Recommendations (books, friends, …)

Serving at-Scale

Layers


Application
Application
Application
Application
Serving:
Make analytics result available for quick and easy access to applications that are serving end users
(Information Retrieval System)
Analytics Processing:
produce analytical results that can be used by applications

Scaling Principles



What's Next?

What's Next?

- **Final exam**
  - **Fri, Jun 25, 2021 @ 9:00am**
    - Closed book
    - Short answers, multiple choice, open answers
    - Online (Zoom)
  - Material
    - Slides
    - Required readings
    - Tutorials
    - Assignments

What you have achieved?

Interest in Data Science

Demonstrated interest in the general area of data science

Interest in Big Data Technologies

Demonstrated interest in big data systems & engineering

Interest in Big Data Analytics

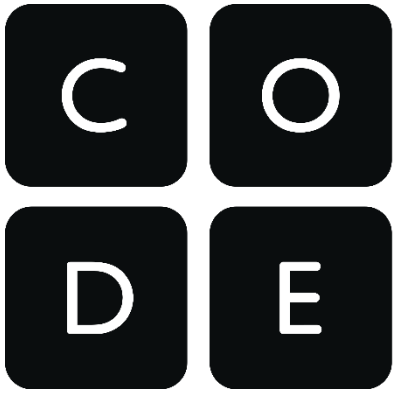
Demonstrated interest in finding interesting patterns and insights in large amounts of data

What is Next?

- **EECS4414: Information Networks**
  - graph mining, network model, network analysis
  - Probably offered next year (TBD)
- **Data Mining Lab (http://dminer.eecs.yorku.ca)**
  - data mining
  - graph mining
  - big data analytics
  - machine learning
  - natural language processing (NLP)
  - city science/IoT

Working with Us?

(solid)
Math & Stat



(solid)
Programming

(interest in)
Data Mining & ML

In Closing…

You have worked a lot…

…and (hopefully) learned a lot!

Course Evaluations

Don't forget to submit your …

…course evaluation!

until Tue, Jun 22!

Happy Holiday

Thanks!

Contact:
Manos Papagelis, LAS 3050
papaggel@eecs.yorku.ca
www.eecs.yorku.ca/~papaggel/