Database Systems

Session 1 – Sub-Topic 2
Data Management Roadmap

Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Agenda

1 **Introduction**

2 **Unstructured Data and Big Data Tools**

3 **Relational Databases and NoSQL**

4 **Data Warehouse, Data Lakes, and Data Swamp**

5 **Distrib./Parallel Processing: Hadoop/Spark/MPP**

6 **Cloud Services**

7 **Case Study: Recommendation App DS Infra.**

10 **Summary and Conclusion**

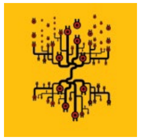
Icons / Metaphors

Information

Common Realization

Knowledge/Competency Pattern

Governance

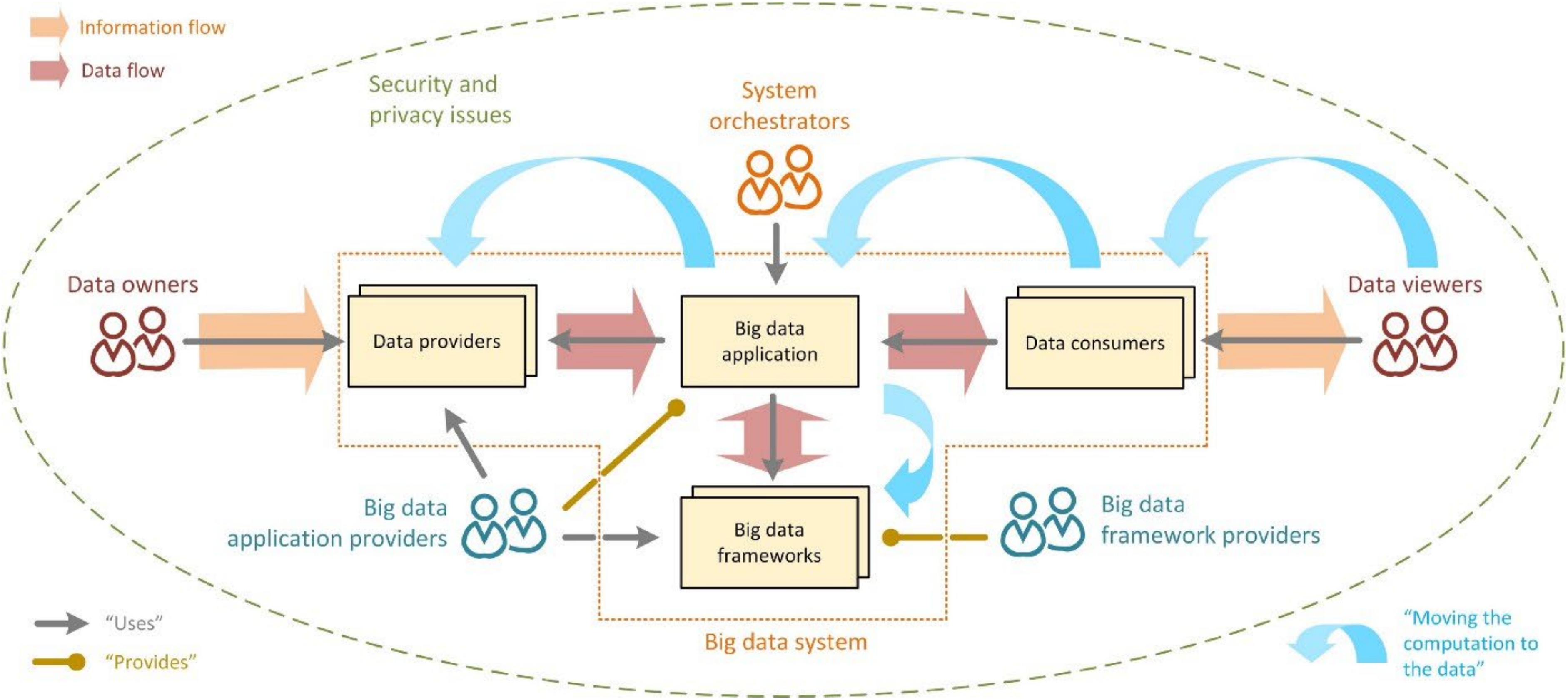

Alignment

Solution Approach

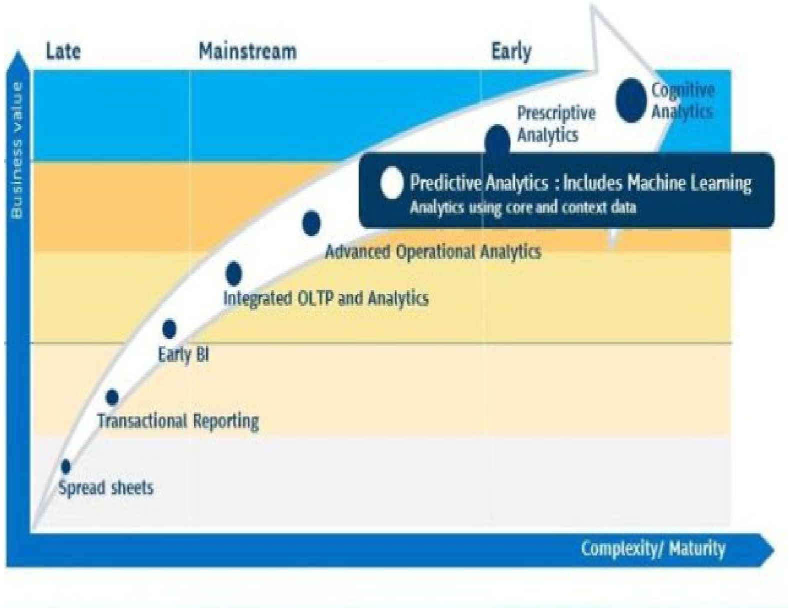
Data Management for Computer/Data Science

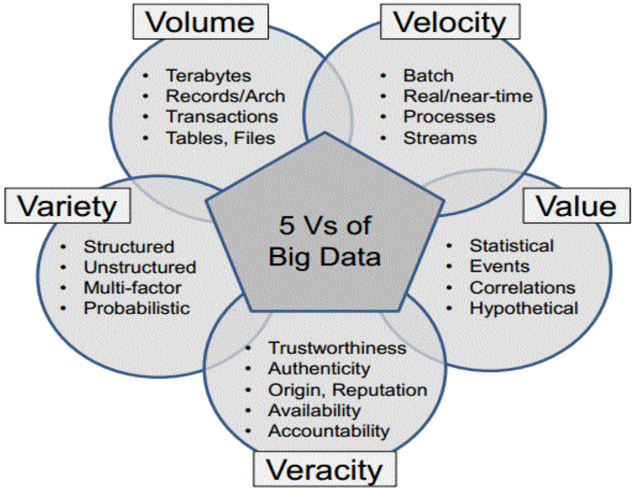
- Storing data to produce a product such as:
  - Database application (computer science)
  - Predictive model (data science)
- However Data science focuses on ML/DL
  - Data management treated as afterthought
  - Focus is on modeling processed/cleaned data in text files stored locally
- In the industry, getting raw data ready to model from various sources is about 80% of the work

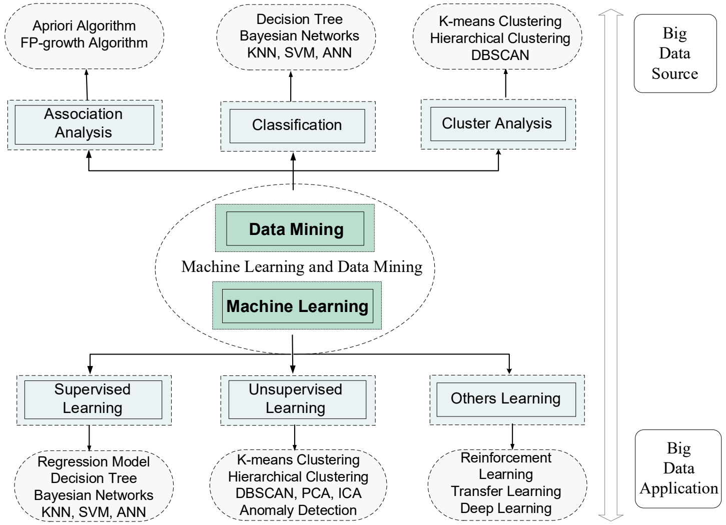
NIST Big Data Taxonomy (Source: WikiCommons)

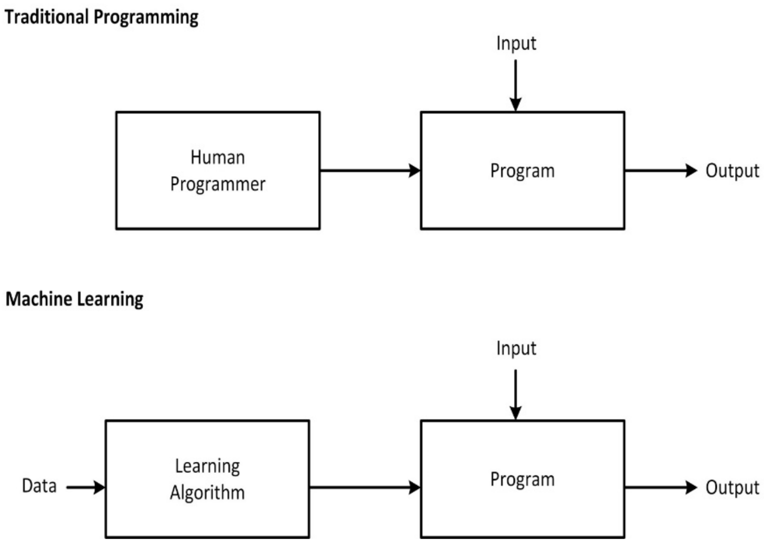


Evolution of Analytics Methods – Traditional Programming vs. ML

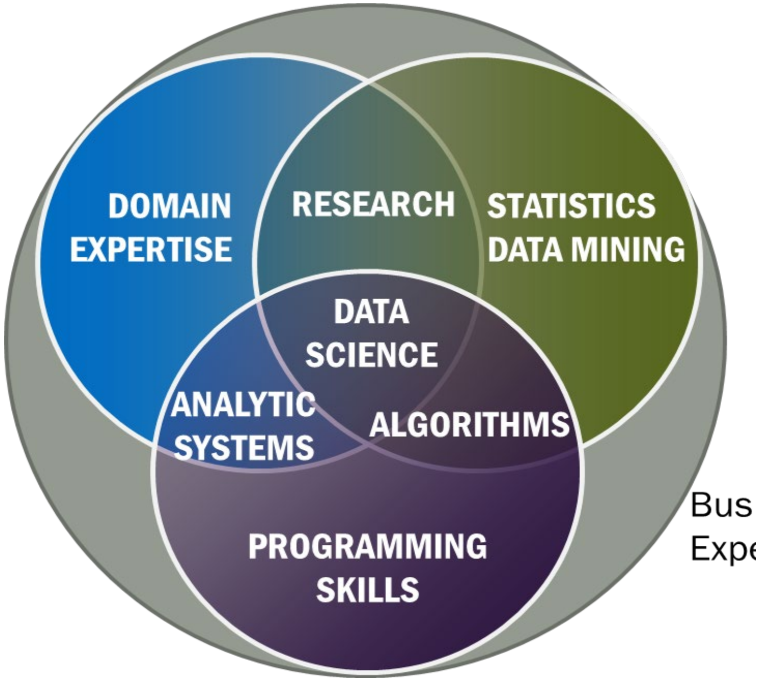


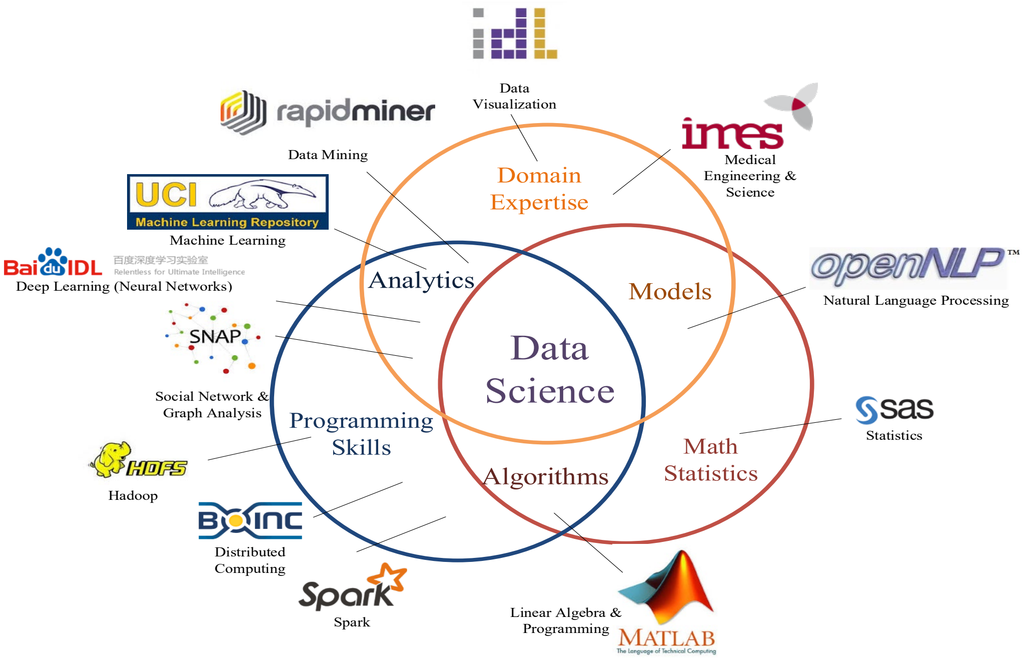






Computer/Data Science and Computer/Data Scientists

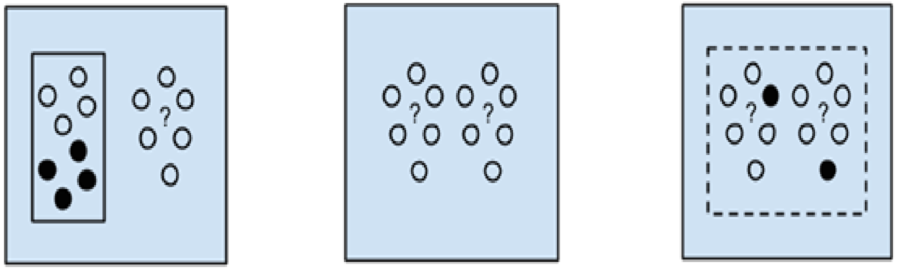



idl
Data Visualization
rapidminer
Data Mining
imes
Medical Engineering & Science
UCI Machine Learning Repository
Machine Learning
Bai du IDL 百度深度学习实验室 Relentless for Ultimate Intelligence
Deep Learning (Neural Networks)
openNLP™
Natural Language Processing
SNAP
Social Network & Graph Analysis
HDFS
Hadoop
BOINC
Distributed Computing
Spark
Spark
Linear Algebra & Programming
MATLAB The Language of Technical Computing
sas
Statistics
Domain Expertise
Analytics
Models
Data Science
Programming Skills
Algorithms
Math Statistics

- Data science: extraction of actionable information(/knowledge?) from data through a process of discovery, and hypothesis analysis
- Data scientist: practitioner with (some) expertise in the following overlapping areas: business needs, domain knowledge, analytics, and programming
- Computer science: study of algorithms used to perform computations, including their formal properties, hardware/linguistic realizations, and applications
- Computer scientist: practitioner with (some) expertise in building end-to-end solutions to problems found in business or in various domains of science

Machine Learning / AI Models and Applications vs. Data Management?

Machine Learning Based on Learning Styles

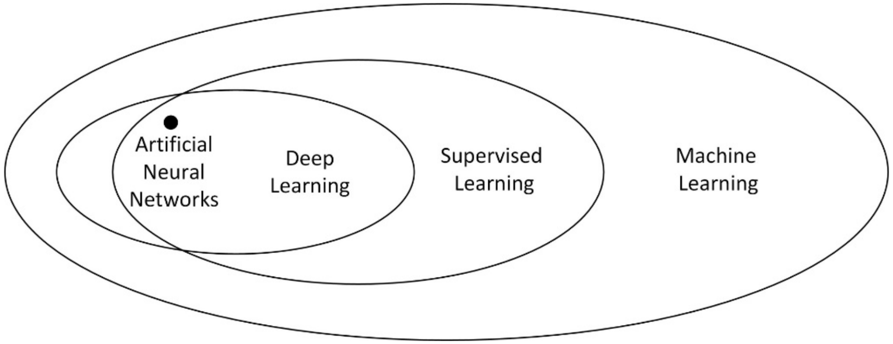


Supervised Learning Algorithms
(a)

Unsupervised Learning Algorithms
(b)

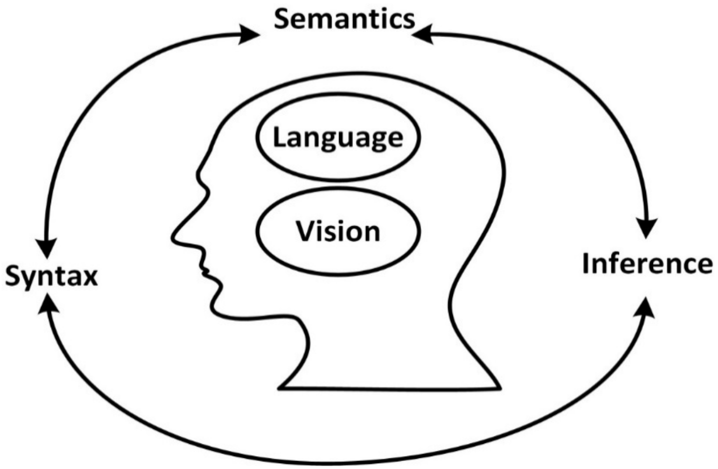
Semi-supervised Learning Algorithms
(c)

Figure 4.1 Machine learning algorithms grouped by different learning styles



What are we learning and where is it stored?

AI and Cognitive Computing Applications



Three components must interact to achieve AI:

(1) Syntax (structure)
(2) Semantics (meaning)
(3) Inference (reasoning/planning)

Human intelligence also requires these three components that rely on some form of data management

Data Management in the Industry

- Enterprise projects usually involve a massive amount of data
  - » Data cannot be stored locally
  - » Modeling process takes place in the cloud
  - » Applications and databases are hosted on servers in data centers
- Data management often handled by separate data engineering team
- Many computer/data scientists know little about data storage and infrastructure,
  - » Cannot make the right decisions

Data Management Roadmap

- What should a computer/data scientist know about data management?
  - Types of databases?
  - Where data is stored?
  - Where data is processed?
  - etc.
- What are the current commercial options?
- Read further to get answers to these questions

Agenda

1 Introduction

2 Unstructured Data and Big Data Tools

3 Relational Databases and NoSQL

4 Data Warehouse, Data Lakes, and Data Swamp

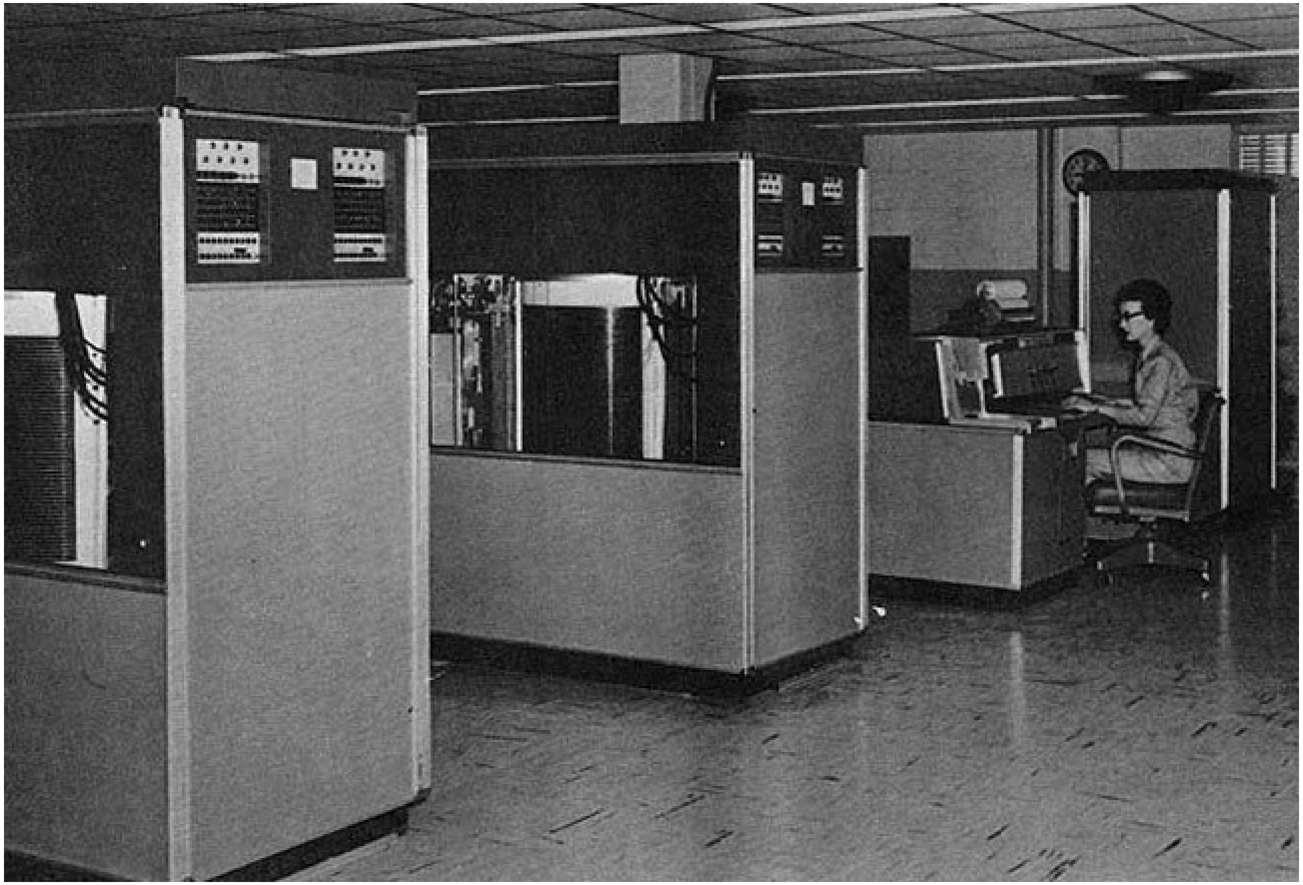
5 Distrib./Parallel Processing: Hadoop/Spark/MPP

6 Cloud Services

7 Case Study: Recommendation App DS Infra.

10 Summary and Conclusion

Rise of Unstructured Data and Big Data Tools



IBM 305 RAMAC (Source: WikiCommons)

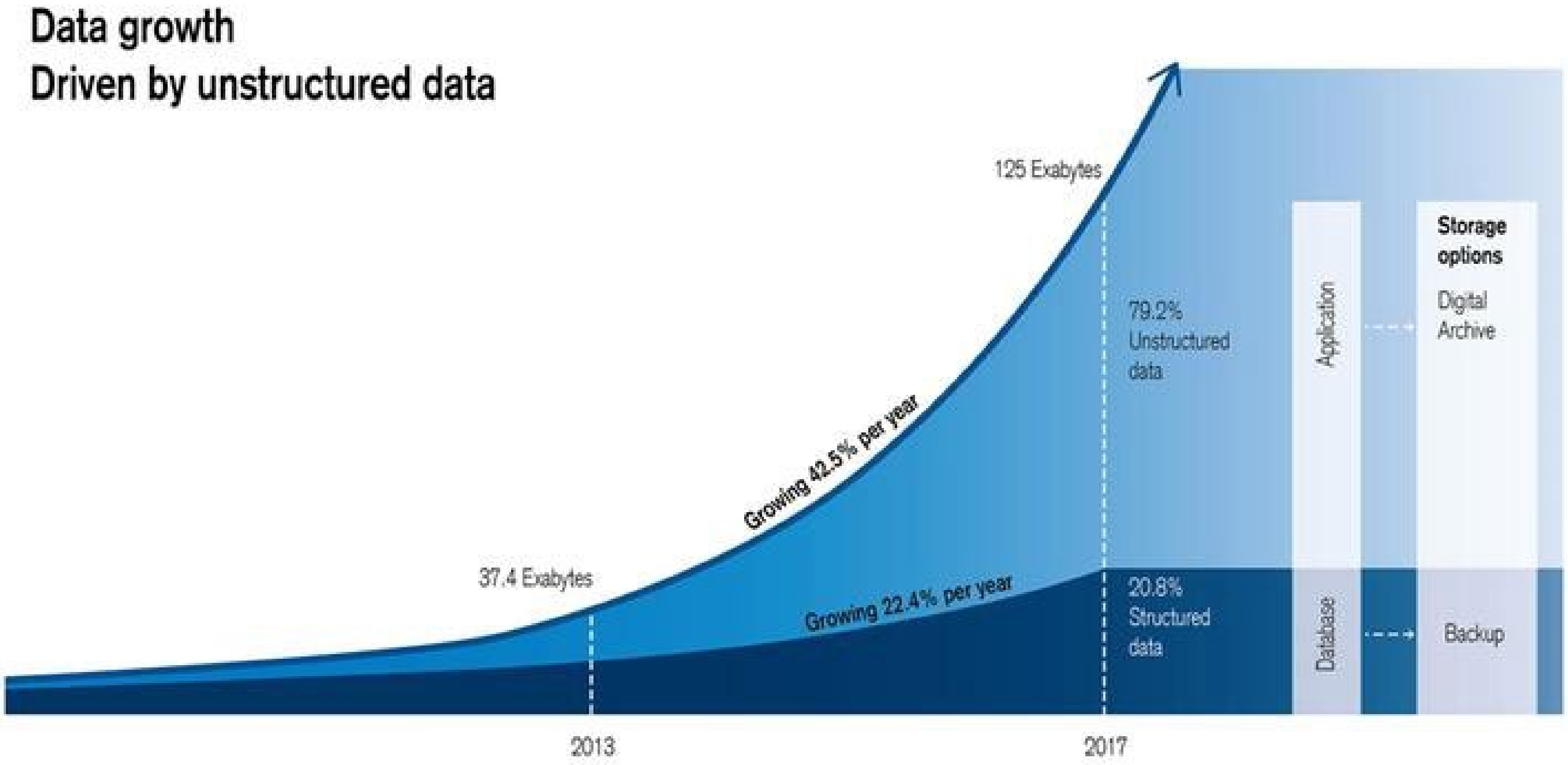
Story of Data Management is the Story of Data Storage

- Pre-digital age:
  - Data stored in our heads, clay tablets, or paper
  - Aggregating and analyzing data was extremely time-consuming
- 1956:
  - IBM introduced the first commercial computer with a magnetic hard drive (305 RAMAC)
    - 30 ft x 50 ft of physical space and over a ton
    - A disk platter stores a hundred bits per square inch
    - Companies could lease it for $3,200 a month to store up to 5 MB of data
- Next 60 years:
  - prices per gigabyte in DRAM dropped from $2.64 billions (1965) to $4.9 (2017)
  - data storage became much denser/smaller in size
  - A disk platter typically stores over a trillion bits per square inch

Big Data Analytics and Applications

- Reduced cost and size in data storage makes today’s big data analytics and applications possible
- Building data management infrastructures to collect and extract insights from huge amount of data is a profitable approach for businesses
- IoT devices can constantly generate and transmit users’ data so businesses are collecting data on an ever increasing number of activities, creating big data
  - e.g. emails, videos, audio, chat messages, social media posts
- Unstructured data accounts for almost 80% of total enterprise data today and is growing twice as fast as structured data in the past decade
- Need for end-to-end applications and underlying data management infrastructures to take full advantage of the current opportunities

Unstructured Data Growth



125 Exabytes of enterprise data was stored in 2017; 80% was unstructured data. (Source: Credit Suisse)

Technologies and Tools Required to Manage Big Data

- Massive data growth transformed the way data is stored and analyzed
  - Traditional tools and approaches were not equipped to handle big data requirements
  - New technologies had to be developed to handle the increasing volume and variety of data at a faster speed and lower cost
  - Newer tools allowing computer/data scientists to monetize big data by performing analytics and building new applications that were not possible before
- Data Management Innovations to know about:
  - Relational and NoSQL DBMSs
  - DataWarehouses, Data Lakes, and Data Swamps
  - Distributed & Parallel Processing (Hadoop, Spark, and Massively Parallel Processing DBMSs)
  - Cloud data management/processing services

Agenda

1 **Introduction**

2 **Unstructured Data and Big Data Tools**

3 **Relational Databases and NoSQL**

4 **Data Warehouse, Data Lakes, and Data Swamp**

5 **Distrib./Parallel Processing: Hadoop/Spark/MPP**

6 **Cloud Services**

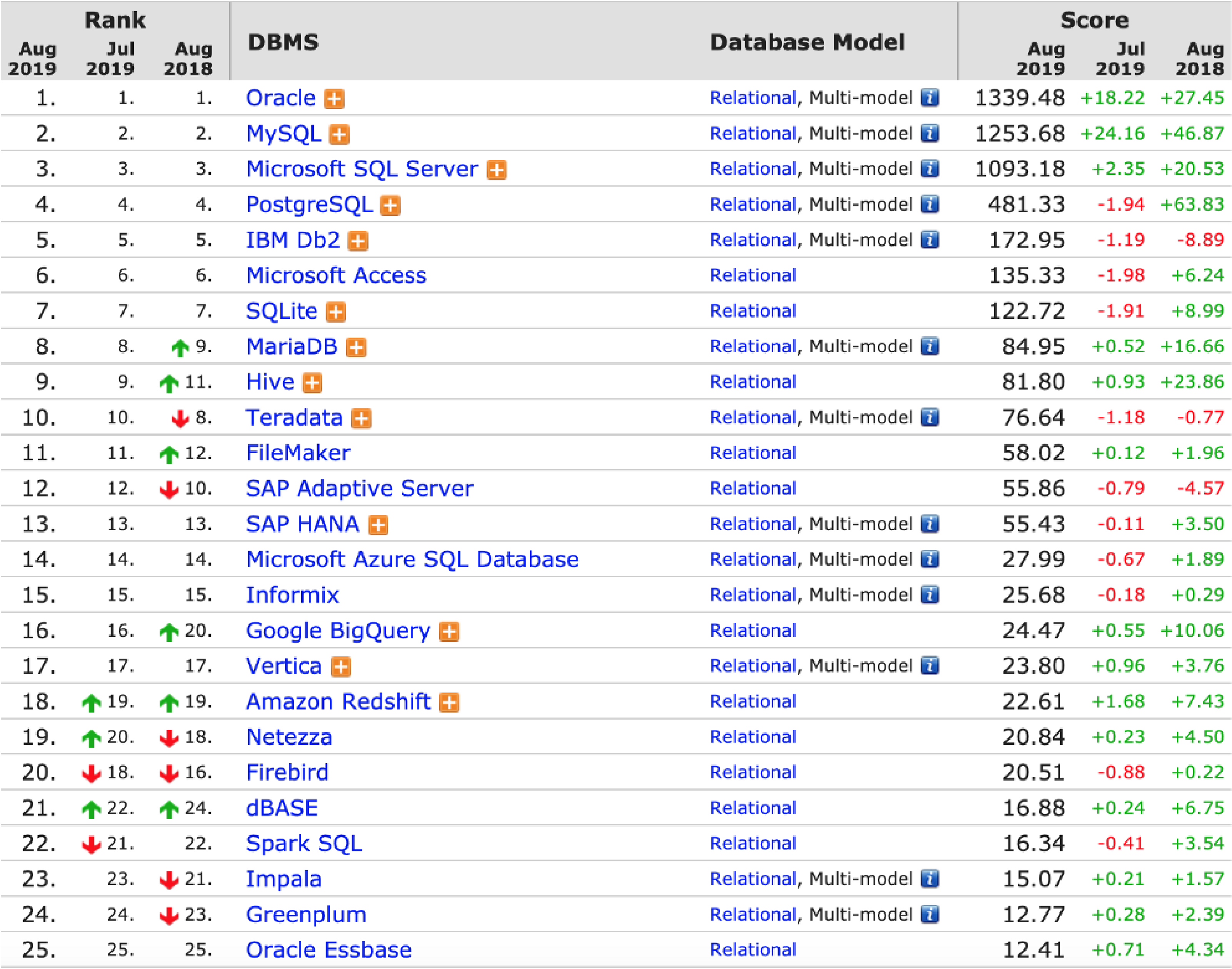
7 **Case Study: Recommendation App DS Infra.**

10 **Summary and Conclusion**

Relational Database Management Systems (RDBMs)

- Relational Database Management Systems (RDBMS):
  - » Emerged in the 1970’s and used to store data as tables with rows and columns, using Structured Query Language (SQL) statements to query and maintain the database
  - » A relational database is basically a collection of tables, each with a schema that rigidly defines the attributes and types of data that they store, as well as keys that identify specific columns or rows to facilitate access
  - » The RDBMS landscape was once ruled by Oracle and IBM, but today many open source options, like MySQL, SQLite, and PostgreSQL are just as popular

RDBMs Ranked by Popularity (Source: DB-Engines)



Rank			DBMS	Database Model	Score		
Aug 2019	Jul 2019	Aug 2018			Aug 2019	Jul 2019	Aug 2018
1.	1.	1.	Oracle	Relational, Multi-model	1339.48	+18.22	+27.45
2.	2.	2.	MySQL	Relational, Multi-model	1253.68	+24.16	+46.87
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1093.18	+2.35	+20.53
4.	4.	4.	PostgreSQL	Relational, Multi-model	481.33	-1.94	+63.83
5.	5.	5.	IBM Db2	Relational, Multi-model	172.95	-1.19	-8.89
6.	6.	6.	Microsoft Access	Relational	135.33	-1.98	+6.24
7.	7.	7.	SQLite	Relational	122.72	-1.91	+8.99
8.	8.	↑ 9.	MariaDB	Relational, Multi-model	84.95	+0.52	+16.66
9.	9.	↑ 11.	Hive	Relational	81.80	+0.93	+23.86
10.	10.	↓ 8.	Teradata	Relational, Multi-model	76.64	-1.18	-0.77
11.	11.	↑ 12.	FileMaker	Relational	58.02	+0.12	+1.96
12.	12.	↓ 10.	SAP Adaptive Server	Relational	55.86	-0.79	-4.57
13.	13.	13.	SAP HANA	Relational, Multi-model	55.43	-0.11	+3.50
14.	14.	14.	Microsoft Azure SQL Database	Relational, Multi-model	27.99	-0.67	+1.89
15.	15.	15.	Informix	Relational, Multi-model	25.68	-0.18	+0.29
16.	16.	↑ 20.	Google BigQuery	Relational	24.47	+0.55	+10.06
17.	17.	17.	Vertica	Relational, Multi-model	23.80	+0.96	+3.76
18.	↑ 19.	↑ 19.	Amazon Redshift	Relational	22.61	+1.68	+7.43
19.	↑ 20.	↓ 18.	Netezza	Relational	20.84	+0.23	+4.50
20.	↓ 18.	↓ 16.	Firebird	Relational	20.51	-0.88	+0.22
21.	↑ 22.	↑ 24.	dBASE	Relational	16.88	+0.24	+6.75
22.	↓ 21.	22.	Spark SQL	Relational	16.34	-0.41	+3.54
23.	23.	↓ 21.	Impala	Relational, Multi-model	15.07	+0.21	+1.57
24.	24.	↓ 23.	Greenplum	Relational, Multi-model	12.77	+0.28	+2.39
25.	25.	25.	Oracle Essbase	Relational	12.41	+0.71	+4.34

RDBMs Pros

- Relational databases found a home in the business world due to some very appealing properties:
  - » Data integrity is paramount in relational databases
  - » RDBMS satisfy the requirements of Atomicity, Consistency, Isolation, and Durability (ACID)
  - » ACID properties achieved imposing a number of constraints to ensure that the stored data is reliable and accurate
    - Makes RDMBs ideal for tracking and storing account numbers, orders, and payments

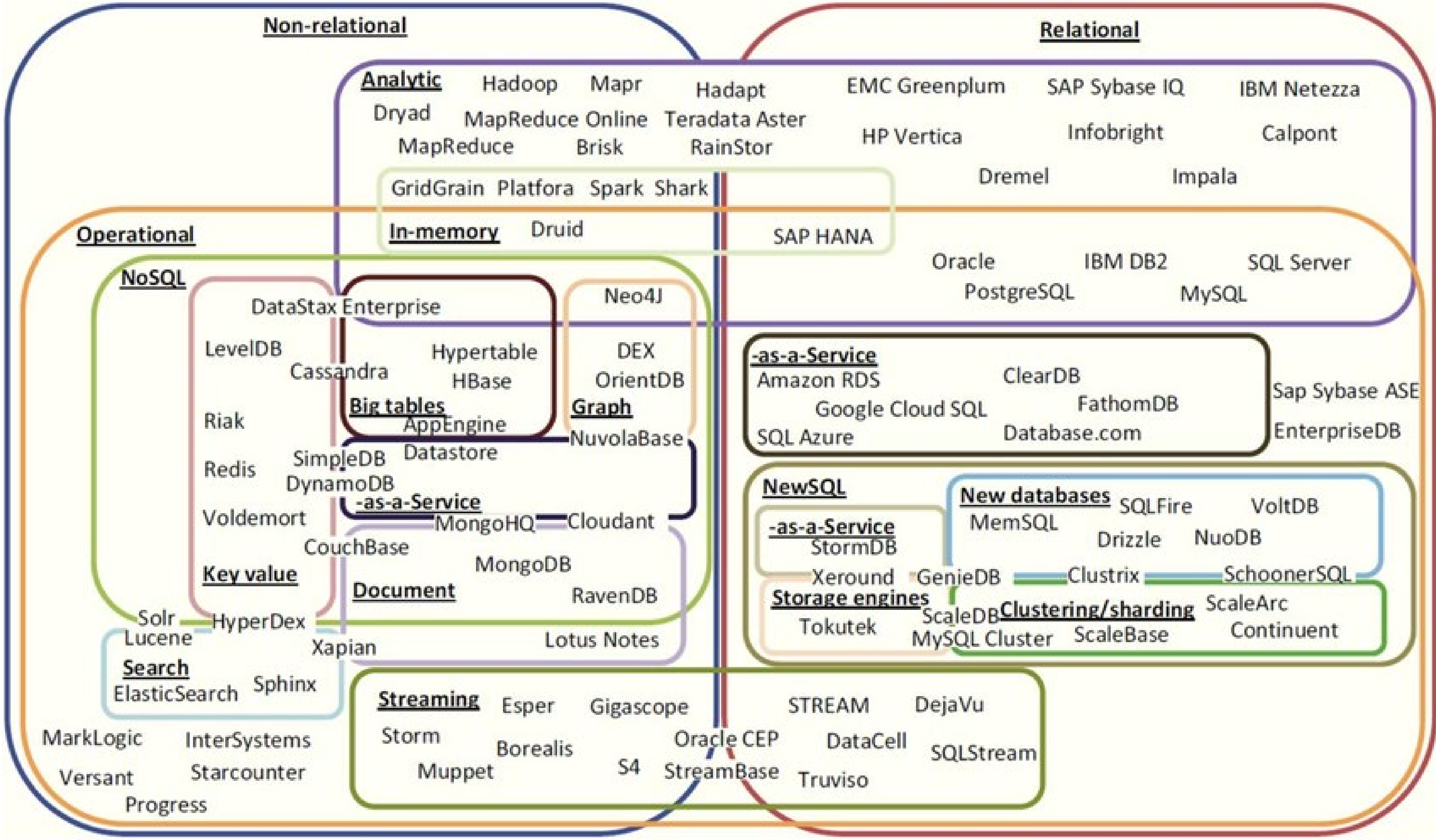
RDBMs Cons

- RDMBS constraints come with costly tradeoffs
  - Because of the schema and type constraints, RDBMS are not very good at storing unstructured or semi-structured data
  - The rigid schema also makes RDBMS more expensive to set up, maintain and grow
  - Setting up a RDBMS requires users to have specific use cases in advance; any changes to the schema are usually difficult and time-consuming
  - In addition, traditional RDBMS were designed to run on a single computer node, which means their speed is significantly slower when processing large volumes of data
  - Sharding RDBMS in order to scale horizontally while maintaining ACID compliance is challenging
- As a result, traditional RDBMS ill-equipped to handle modern big data

NoSQL DBMSs

- By the mid-2000’s, the existing RDBMS faced difficulties handling the changing needs and exponential growth of a few very successful online businesses
  - Many non-relational (or NoSQL) databases were developed as a result
  - See Facebook story on how the company dealt with the limitations of MySQL when their data volume started to grow
- Online businesses invented new approaches and tools to handle the massive amount of unstructured data they collected:
  - Google created GFS, MapReduce, and BigTable
  - Amazon created DynamoDB
  - Yahoo created Hadoop
  - Facebook created Cassandra and Hive
  - LinkedIn created Kafka
- Some businesses open sourced their work; some published research papers detailing their designs
  - This resulted in a proliferation of databases with the new technologies, and NoSQL databases emerged as a major player in the industry

An Explosion of DBMs Since the 2000's - Source: Korflatis et. al (2016)



NoSQL DBMSs Pros

- NoSQL databases are schema agnostic and provide the flexibility needed to store and manipulate large volumes of unstructured and semi-structured data
- Users don’t need to know what types of data will be stored during set-up, and the system can accommodate changes in data types and schema
- Designed to distribute data across different nodes, NoSQL databases are generally more horizontally scalable and fault-tolerant

NoSQL DBMSs Cons

- NoSQL DBMSs' performance benefits also come with a cost: they are not ACID compliant and data consistency is not guaranteed
- NoSQL DBMSs provide "eventual consistency" instead: when old data is getting overwritten, they return results that are a little wrong temporarily
  - Google's search engine index can't overwrite its data while people are simultaneously searching a given term, so it doesn't give us the most up-to-date results when we search, but it gives us the latest, best answer it can
- While this setup won't work in situations where data consistency is absolutely necessary (such as financial transactions); it's just fine for tasks that require speed rather than pin-point accuracy

Various Categories of NoSQL DBMSs Serving Specific Purposes

- Key-Value Stores:
  - e.g., Redis, DynamoDB, and Cosmos DB
  - Store only key-value pairs and provide basic functionality for retrieving the value associated with a known key
  - Work best with a simple database schema and when speed is important
- Wide Column Stores:
  - e.g., Cassandra, Scylla, and Hbase
  - Store data in column families or tables
  - Built to manage petabytes of data across a massive, distributed system
- Document Stores:
  - e.g., MongoDB and Couchbase
  - Store data in XML or JSON format, with the document name as key and the contents of the document as value
  - Documents can contain many different value types, and can be nested, making them particularly well-suited to manage semi-structured data across distributed systems
- Graph Databases:
  - e.g., Neo4J and Amazon Neptune
  - Represent data as a network of related nodes or objects in order to facilitate data visualizations and graph analytics
  - Particularly useful for analyzing the relationships between heterogeneous data points, such as in fraud prevention or Facebook’s friends graph

MongoDB

- The most popular NoSQL database, and has delivered substantial values for some businesses that have been struggling to handle their unstructured data with the traditional RDBMS approach
- Here are two industry examples:
  - After MetLife spent *years* trying to build a centralized customer database on a RDBMS that could handle all its insurance products, someone at an internal hackathon built one with MongoDB within hours, which went to production in 90 days
  - YouGov, a market research firm that collects 5 gigabits of data an hour, saved 70 percent of the storage capacity it formerly used by migrating from RDBMS to MongoDB

Agenda

1 Introduction

2 Unstructured Data and Big Data Tools

3 Relational Databases and NoSQL

e **4 Data Warehouse, Data Lakes, and Data Swamp**

5 Distrib./Parallel Processing: Hadoop/Spark/MPP

6 Cloud Services

7 Case Study: Recommendation App DS Infra.

10 Summary and Conclusion

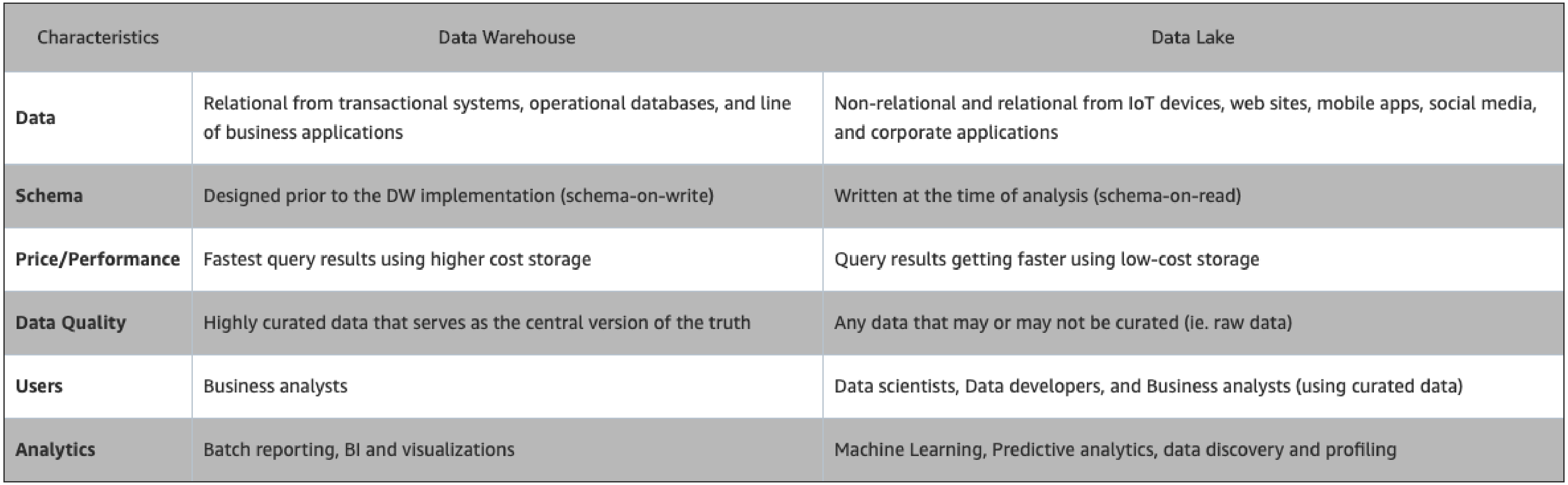
Data Warehouse, Data Lake, & Data Swamp

- As data sources continue to grow, performing data analytics with multiple databases became inefficient and costly
- One solution called Data Warehousing emerged in the 1980's
  - Centralizes an enterprise's data from all of its databases
  - Supports the flow of data from operational systems to analytics/decision systems by creating a single repository of data from various sources (both internal and external)
  - Is (in most cases) a relational database that stores processed data that is optimized for gathering business insights
  - Collects data with predetermined structure and schema coming from transactional systems and business applications, and the data is typically used for operational reporting and analysis

From Data Warehouses to Data Lakes

- Data going into data warehouses needs to be processed before it gets stored
- With today's massive amount of unstructured data, that could take significant time and resources
- In response, businesses started maintaining Data Lakes in the 2010's
  - Store all of an enterprise's structured and unstructured data at any scale
  - Store raw data, and can be set up without having to first define the data structure and schema
  - Allow users to run analytics without having to move the data to a separate analytics system, enabling businesses to gain insights from new sources of data that was not available for analysis before, for instance by building machine learning models using data from log files, click-streams, social media, and IoT devices
  - By making all of the enterprise data readily available for analysis, computer/data scientists can answer a new set of business questions, or tackle old questions with new data

Data Warehouse and Data Lake Comparisons (Source: : AWS)



Characteristics	Data Warehouse	Data Lake
Data	Relational from transactional systems, operational databases, and line of business applications	Non-relational and relational from IoT devices, web sites, mobile apps, social media, and corporate applications
Schema	Designed prior to the DW implementation (schema-on-write)	Written at the time of analysis (schema-on-read)
Price/Performance	Fastest query results using higher cost storage	Query results getting faster using low-cost storage
Data Quality	Highly curated data that serves as the central version of the truth	Any data that may or may not be curated (ie. raw data)
Users	Business analysts	Data scientists, Data developers, and Business analysts (using curated data)
Analytics	Batch reporting, BI and visualizations	Machine Learning, Predictive analytics, data discovery and profiling

From Data Lakes to Data Swamps

- A common challenge with the Data Lake architecture:
  - » Without the appropriate data quality and governance framework in place, when terabytes of structured and unstructured data flow into the Data Lakes, it often becomes extremely difficult to sort through their content
- Data Lakes can turn into Data Swamps as the stored data become too messy to be usable
- Many organizations are now calling for more data governance and metadata management practices to prevent Data Swamps from forming

Agenda

1 **Introduction**

2 **Unstructured Data and Big Data Tools**

3 **Relational Databases and NoSQL**

4 **Data Warehouse, Data Lakes, and Data Swamp**

5 **Distrib./Parallel Processing: Hadoop/Spark/MPP**

6 **Cloud Services**

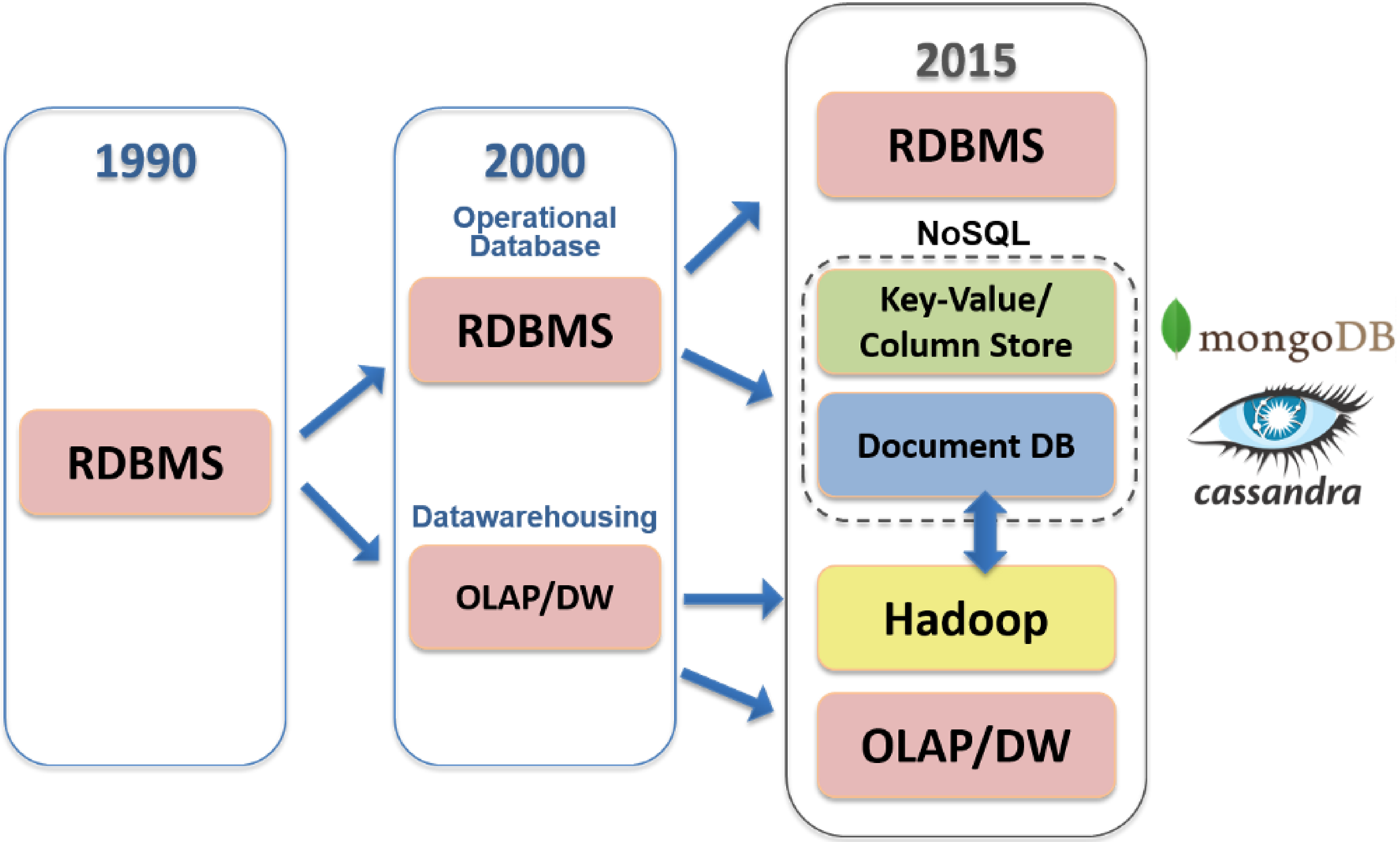
7 **Case Study: Recommendation App DS Infra.**

10 **Summary and Conclusion**

Distributed & Parallel Processing: Hadoop, Spark, & MPP

- While storage and computing needs grew by leaps and bounds in the last several decades, traditional hardware has not advanced enough to keep up
- Enterprise data no longer fits neatly in standard storage, and the computation power required to handle most big data analytics tasks may take weeks, months, or simply not possible to complete on a standard computer
- To overcome this deficiency, many new technologies have evolved to include multiple computers working together, distributing the database to thousands of commodity servers
  - When a network of computers are connected and work together to accomplish the same task, the computers form a cluster
  - A cluster can be thought of as a single computer, but can dramatically improve the performance, availability, and scalability over a single, more powerful machine, and at a lower cost by using commodity hardware
  - Apache Hadoop is an example of distributed data infrastructures that leverage clusters to store and process massive amounts of data, and what enables the Data Lake architecture

Evolution of DBMS Technologies (Source: Business Analytic 3.0)


1990

RDBMS

2000

Operational Database

RDBMS

Datawarehousing

OLAP/DW

2015

RDBMS

NoSQL

Key-Value/ Column Store

Document DB

Hadoop

OLAP/DW

MapReduce Innovation

- MapReduce is a two step computational approach for processing large (multi-terabyte or greater) data sets distributed across large clusters of commodity hardware in a reliable, fault-tolerant way
  - First step: distribute your data across multiple computers (Map), with each performing a computation on its slice of the data in parallel
  - Next step: combine those results in a pair-wise manner (Reduce)
    - Google published a paper on MapReduce in 2004, which got picked up by Yahoo programmers who implemented it in the open source Apache environment in 2006, providing every business the capability to store an unprecedented volume of data using commodity hardware
- Even though there are many open source implementations of the idea, the Google brand name MapReduce has stuck around

Hadoop's Architecture

- Hadoop leverages "distribution" and consists of three main components:
  - Hadoop Distributed File System (HDFS), a way to store and keep track of your data across multiple (distributed) physical hard drives
  - MapReduce, a framework for processing data across distributed processors
  - Yet Another Resource Negotiator (YARN), a cluster management framework that orchestrates the distribution of things such as CPU usage, memory, and network bandwidth allocation across distributed computers

Hadoop’s Computational Approach

- Hadoop is built for iterative computations, scanning massive amounts of data in a single operation from disk, distributing the processing across multiple nodes, and storing the results back on disk
- Querying zettabytes of indexed data that would take 4 hours to run in a traditional data warehouse environment could be completed in 10–12 seconds with Hadoop and Hbase
- Hadoop is typically used to generate complex analytics models or high volume data storage applications
  - e.g., retrospective and predictive analytics, machine learning and pattern matching, customer segmentation and churn analysis, and active archives

From Hadoop to Spark (1/2)

- MapReduce processes data in batches and is therefore not suitable for processing real-time data
- Apache Spark was built in 2012 to fill that gap
  - Parallel data processing tool that is optimized for speed and efficiency by processing data in-memory
  - Operates under the same MapReduce principle, but runs much faster by completing most of the computation in memory and only writing to disk when memory is full or the computation is complete

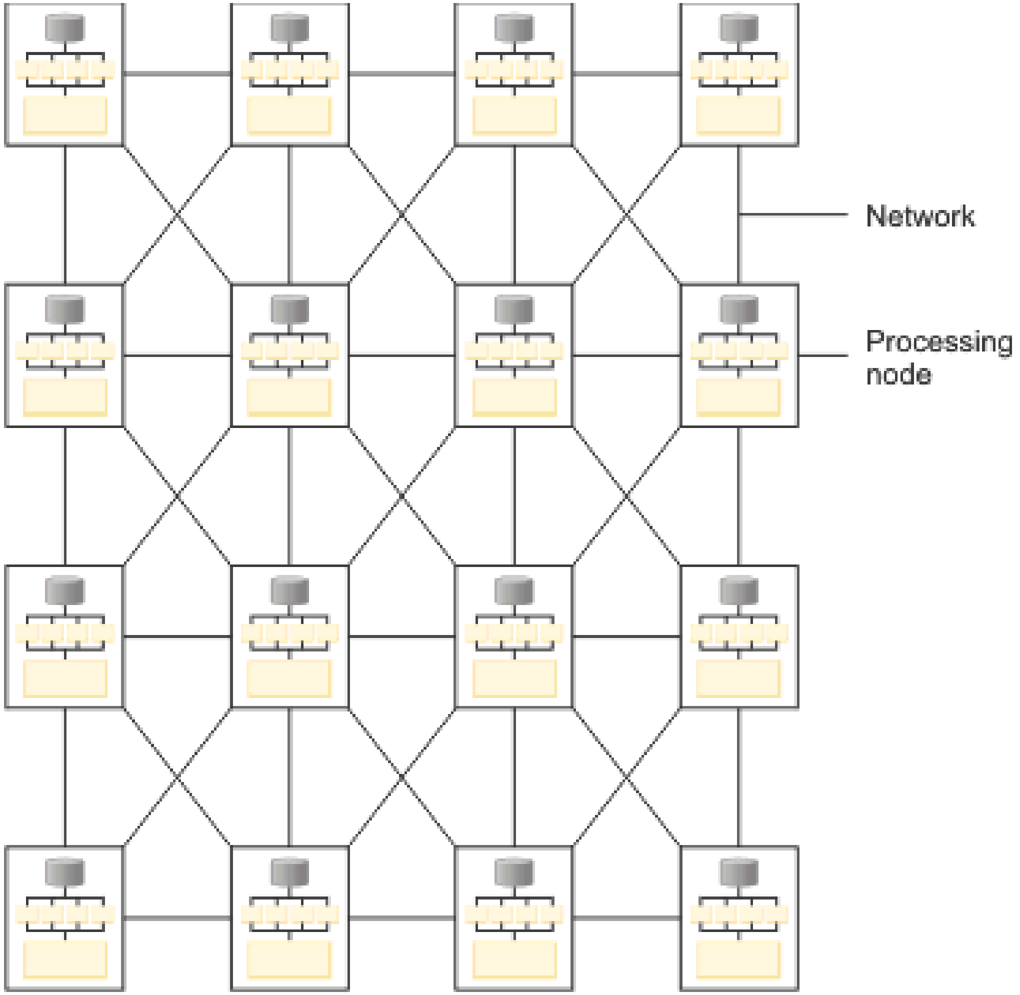
From Hadoop to Spark

- In-memory computation allows Spark to “run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk”
- However, when the data set is so large that insufficient RAM becomes an issue (usually hundreds of gigabytes or more), Hadoop MapReduce might outperform Spark
- Spark also has an extensive set of data analytics libraries covering a wide range of functions:
  - Spark SQL for SQL and structured data
  - MLib for machine learning
  - Spark Streaming for stream processing
  - GraphX for graph analytics
- Since Spark’s focus is on computation, it does not come with its own storage system and instead runs on a variety of storage systems such as Amazon S3, Azure Storage, and Hadoop’s HDFS

Massively Parallel Processing – MPP (1/2)

- Hadoop and Spark are not the only technologies that leverage clusters to process large volumes of data
- Another popular computational approach to distributed query processing is called Massively Parallel Processing (MPP)
  - Similar to MapReduce, MPP distributes data processing across multiple nodes, and the nodes process the data in parallel for faster speed
  - Unlike Hadoop, MPP is used in RDBMS and utilizes a “share-nothing” architecture
    - Each node processes its own slice of the data with multi-core processors, making them many times faster than traditional RDBMS
    - Some MPP databases, like Pivotal Greenplum, have mature machine learning libraries that allow for in-database analytics

MPP Systems



In an MPP system, all the nodes are interconnected and data could be exchanged across the network (Source: IBM)

Massively Parallel Processing – MPP (2/2)

- However, as with traditional RDBMS, most MPP databases do not support unstructured data, and even structured data will require some processing to fit the MPP infrastructure
  - » Therefore it takes additional time and resources to set up the data pipeline for an MPP database
  - » Since MPP databases are ACID-compliant and deliver much faster speed than traditional RDBMS, they are usually employed in high-end enterprise data warehousing solutions
    - e.g., Amazon Redshift, Pivotal Greenplum, and Snowflake
- Case study:
  - » The New York Stock Exchange receives four to five terabytes of data daily and conducts complex analytics, market surveillance, capacity planning and monitoring
  - » The company had been using a traditional database that couldn’t handle the workload, which took hours to load and had poor query speed
  - » Moving to an MPP database reduced their daily analysis run time by eight hours

Agenda

1 Introduction

2 Unstructured Data and Big Data Tools

3 Relational Databases and NoSQL

4 Data Warehouse, Data Lakes, and Data Swamp

5 Distrib./Parallel Processing: Hadoop/Spark/MPP

6 Cloud Services

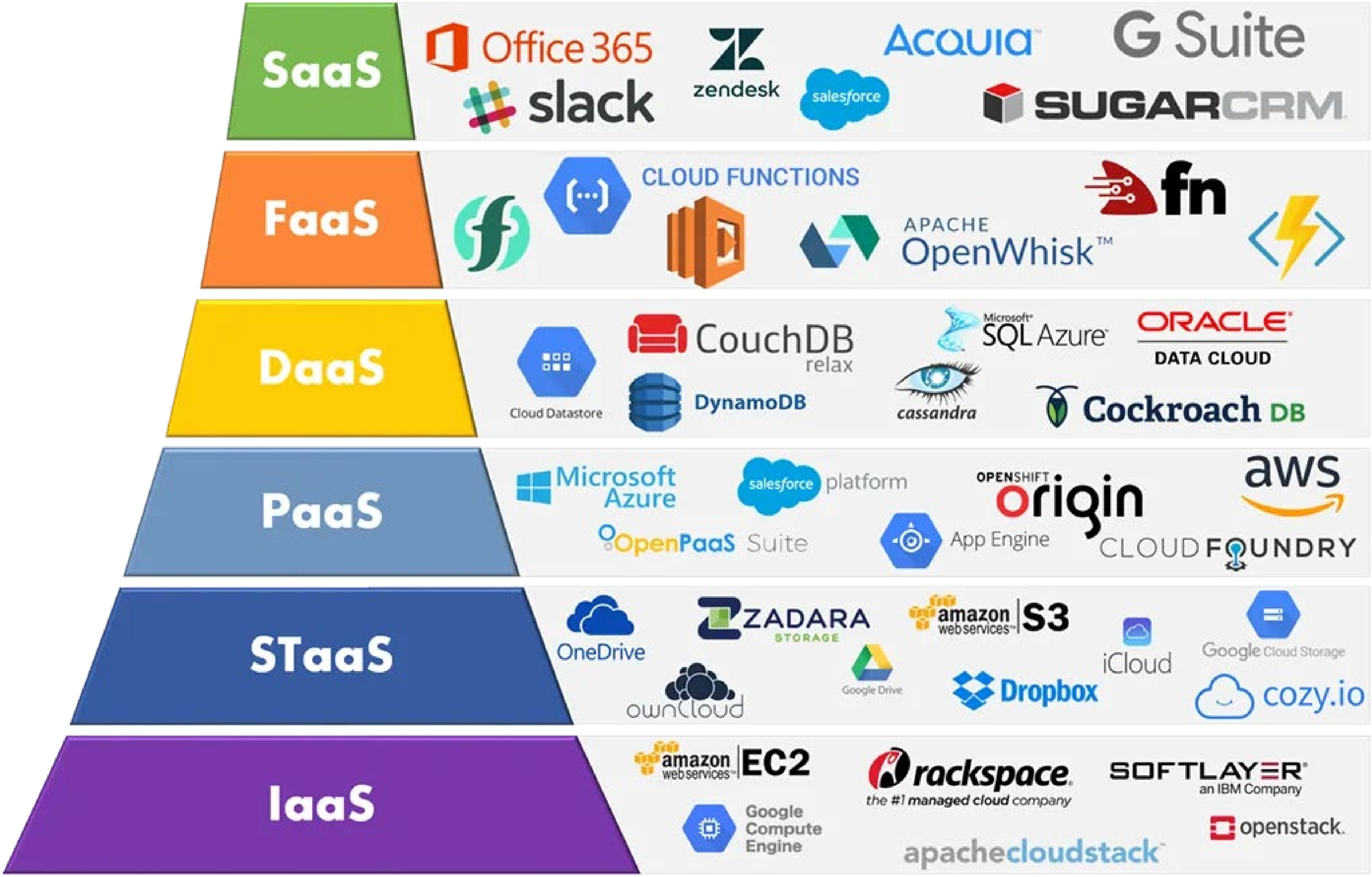
7 Case Study: Recommendation App DS Infra.

10 Summary and Conclusion

Cloud Services

- Another innovation that completely transformed enterprise big data analytics capabilities is the rise of cloud services
- Before cloud services were available, businesses had to:
  - Buy on-premises data storage and analytics solutions from software and hardware vendors
  - Pay upfront perpetual software license fees and annual hardware maintenance and service fees
    - On top of those fees were the costs of power, cooling, security, disaster protection, IT staff, etc., for building and maintaining the on-premises infrastructure
    - Even when it was technically possible to store and process big data, most businesses found it cost prohibitive to do so at scale
  - Scaling with on-premises infrastructure also require an extensive design and procurement process, which takes a long time to implement and requires substantial upfront capital
- Many potentially valuable data collection and analytics possibilities were ignored then as a result

“As a Service” Providers



Infrastructure as a Service (IaaS) and Storage as a Service (STaaS) (Source: IMELGRAT.ME)

On-Premises vs. Cloud Models (1/2)

- The on-premises model began to lose market share quickly when cloud services were introduced in the late 2000's
- The global cloud services market has been growing 15% annually in the past decade
- Cloud service platforms provide subscriptions to a variety of services (from virtual computing to storage infrastructure to databases)
  - Services are delivered over the internet on a pay-as-you-go basis, offering customers rapid access to flexible and low-cost storage and virtual computing resources
  - Cloud service providers are responsible for all of their hardware and software purchases and maintenance, and usually have a vast network of servers and support staff to provide reliable services

On-Premises vs. Cloud Models (2/2)

- Cloud for business:
  - Businesses can significantly reduce costs and improve operational efficiencies with cloud services
  - Businesses are able to develop and productionize their products more quickly with the out-of-the-box cloud resources and their built-in scalability
- Cloud services remove the upfront costs and time commitment to build on-premises infrastructure
- Cloud services also lower the barriers to adopt big data tools, which has the effect of democratizing big data analytics for small and med-size businesses

Cloud Services Models (1/2)

- Public cloud (most common)
  - All hardware, software, and other supporting infrastructure are owned and managed by the cloud service provider
  - Customers share the cloud infrastructure with other “cloud tenants” and access their services through a web browser
- Private cloud
  - Often used by organizations with special security needs such as government agencies and financial institutions
  - Services and infrastructure are dedicated solely to one organization and are maintained on a private network
  - Private cloud can be on-premises, or hosted by a third-party service provider elsewhere

Cloud Services Models (2/2)

- Hybrid clouds
  - Combine private clouds with public clouds, allowing organizations to reap the advantages of both
  - Data and applications can move between private and public clouds for greater flexibility
    - e.g. the public cloud could be used for high-volume, lower-security data, and the private cloud for sensitive, business-critical data like financial reporting
- Multi-cloud model
  - Involves multiple cloud platforms, each delivers a specific application service
  - A multi-cloud can be a combination of public, private, and hybrid clouds to achieve the organization’s goals
  - Organizations often choose multi-cloud to suit their particular business, locations, and timing needs, and to avoid vendor lock-in

Agenda

1 Introduction

2 Unstructured Data and Big Data Tools

3 Relational Databases and NoSQL

4 Data Warehouse, Data Lakes, and Data Swamp

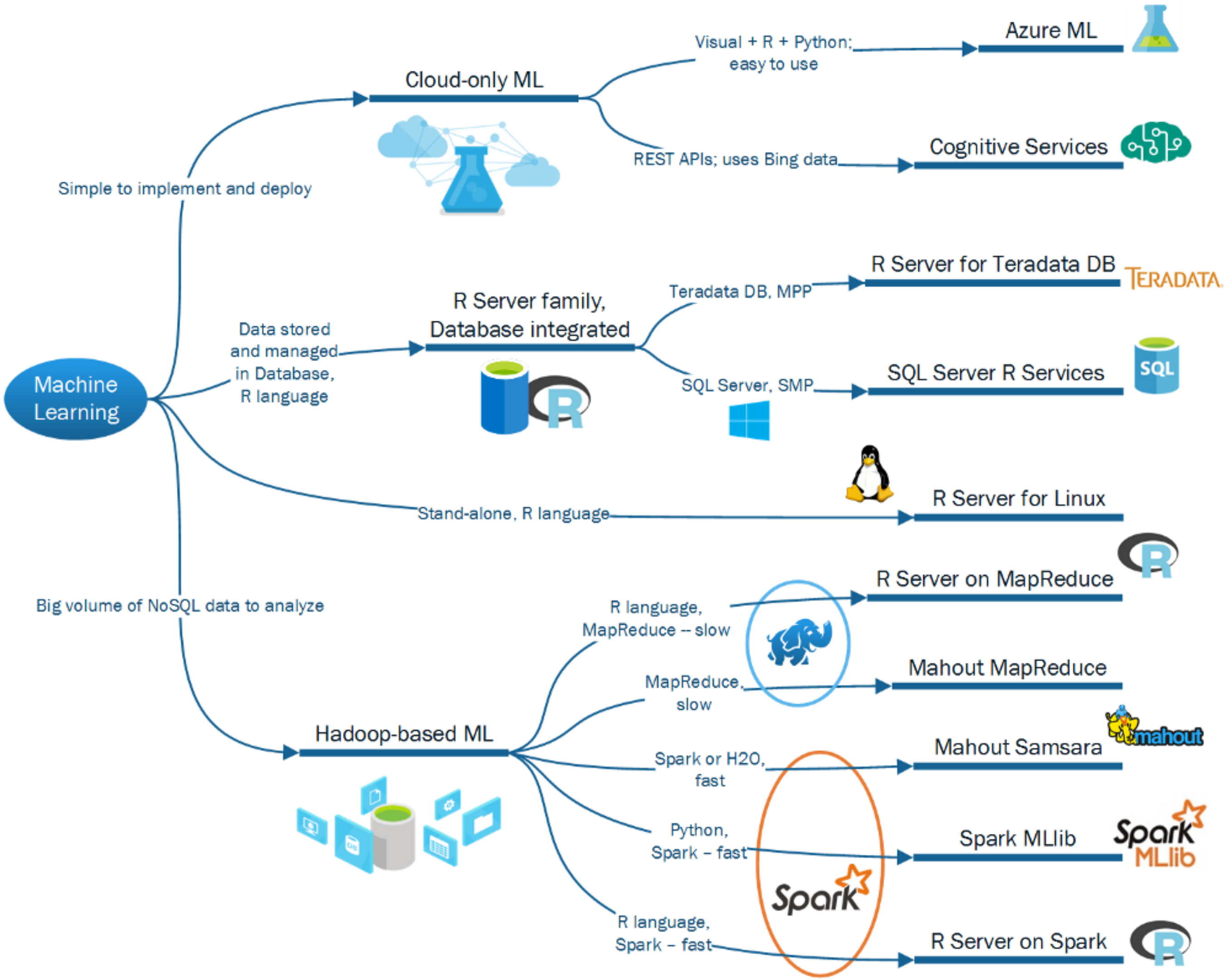
5 Distrib./Parallel Processing: Hadoop/Spark/MPP

6 Cloud Services

7 Case Study: Recommendation App DS Infra.

10 Summary and Conclusion

Case Study: End-to-End Data Mgmt. Infrastruct. for Recommender App



Machine learning packages for different types of data environment (Source: Kosyakov (2016))

Viable Data Management Infrastructures

- A viable data management infrastructure involves much more than just building a machine learning model with scikit-learn, pickling it, and loading it on a server
- Requires an understanding of how all the parts of the enterprise's ecosystem work together
  - Where/how the data flows into the data team
  - Environment where the data is processed/transformed
  - Enterprise's conventions for visualizing/presenting data
  - How the model output will be converted as input for some other enterprise applications
- The main goal should be to build a process that will be easy to maintain and where
  - Models can be iterated on and the performance is reproducible
  - A model's output can be easily understood and visualized for other stakeholders so that they may make better informed business decisions
- Achieving those goals require selecting the right tools, as well as an understanding of what others in the industry are doing and the best practices

Example Scenario: Building a Recommender App for a Startup

- App is expected to collect hundreds of gigabytes of both structured (customer profiles, temperatures, prices, and transaction records) and unstructured (customers' posts/comments and image files) data from users daily
- Predictive models will need to be retrained with new data weekly and make recommendations instantaneously on demand
- Data collection, storage, and analytics capacity would have to be extremely scalable
- Questions at hand:
  - How to design a scalable data science process and productionize the models?
  - What are the tools needed to get the job done?

Step 1: Set up A Data Pipeline

- Data pipeline should take in the raw data from data sources, processes the data, and feeds the processed data to databases
- Data pipeline should demonstrate:
  - » Low event latency
    - Ability to query data as soon as it’s been collected
  - » High scalability
    - Ability to handle massive amount of data as your product scales
  - » Interactive querying capabilities
    - Support for both batch queries and smaller interactive queries that allow data scientists to explore the tables and schemas
  - » Versioning capabilities
    - Ability to make changes to the pipeline without bringing down the pipeline and losing data
  - » Monitoring capabilities
    - The pipeline should generate alerts when data stops coming in
  - » Testing capabilities
    - Ability to test the pipeline without interruptions
- Most importantly, the data pipeline should not interfere with daily business operations
  - » New model being tested should not cause the operational database to grind to a halt
- Building and maintaining the data pipeline is usually the responsibility of a data engineer but a data scientist should at least be familiar with the process, its limitations, and the tools needed to access the processed data for analysis

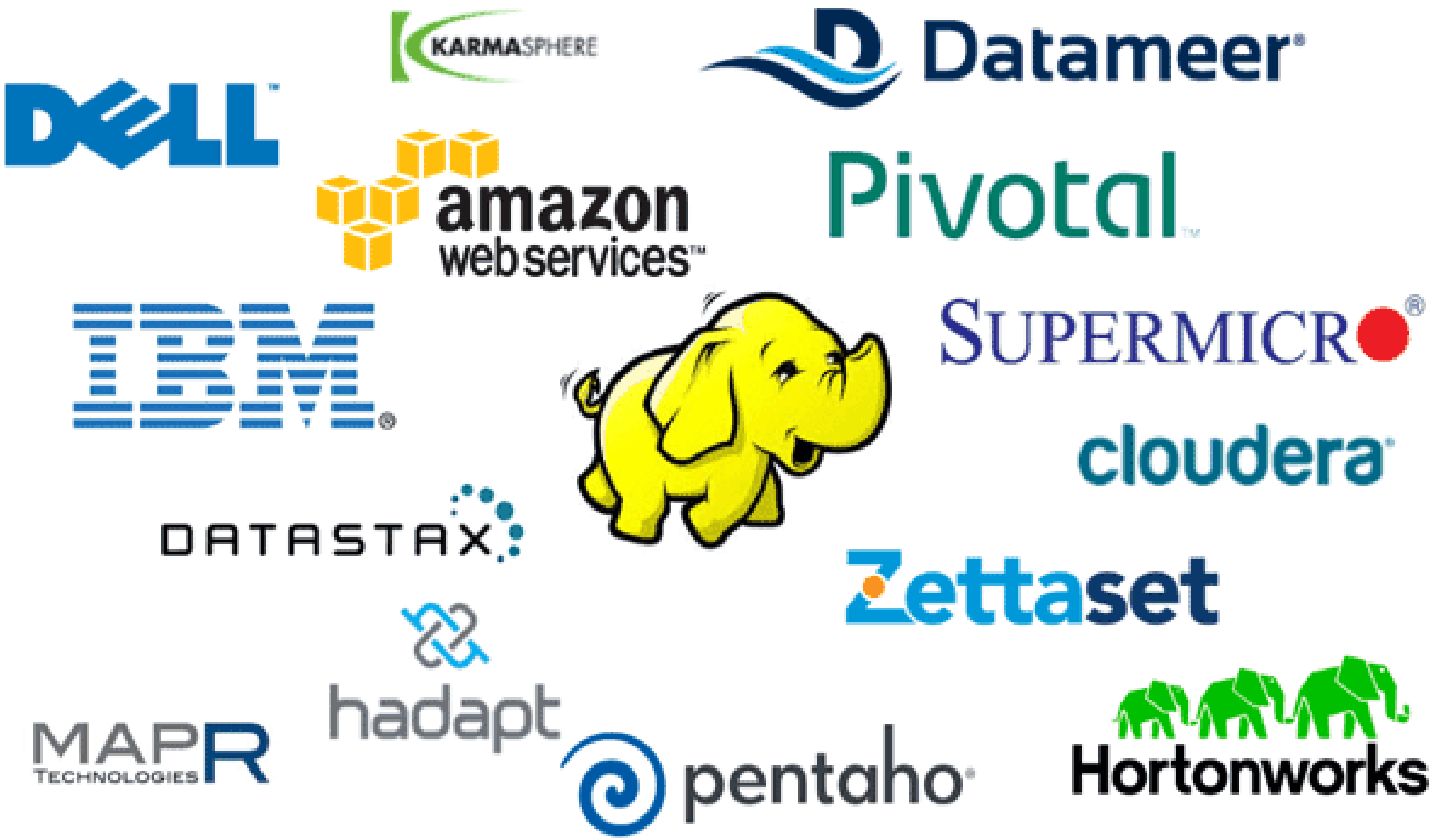
Step 2: Decide On-Premise Infrastructure vs. Cloud Services?

- Top priority in this scenario is to scale data collection without scaling operational resources
- On-premise infrastructure requires huge upfront and maintenance costs, so cloud services tend to be a better option for startups
- Cloud services allow scaling to match demand and require minimal maintenance efforts, so that a small team of staff could focus on the product and analytics instead of infrastructure management

Step 3: Choose a Cloud Service Provider If Applicable (1/4)

- Establish the data that you would need for analytics and figure out which databases and analytics infrastructure are most suitable for those data types
  - Since there would be both structured and unstructured data in your analytics pipeline, you might want to set up both a Data Warehouse and a Data Lake
- Then Consider whether the storage layer supports the big data tools that are needed to build the models, and if the database provides effective in-database analytics
  - For example, some ML libraries such as Spark’s MLlib cannot be used effectively with databases as the main interface for data
    - The data would have to be unloaded from the database before it can be operated on, which could be extremely time-consuming as data volume grows and might become a bottleneck when you’ve to retrain your models regularly

Step 3: Example of Vendors Providing Hadoop-Based Solutions (2/4)



Source: WikiCommons

Step 3: Investigate Cloud Service Provider’s ML/DL Capabilities (3/4)

- Investigate the state of the market when it comes to machine learning capabilities
- Most cloud providers are working hard to develop their native machine learning capabilities
  - Allows data scientists to build and deploy machine learning models easily with data stored in their own platform
    - Amazon’s SageMaker
    - Google’s BigQuery ML
    - Microsoft’s Azure Machine Learning
- The toolsets are still developing and often incomplete
  - e.g., BigQuery ML currently only support linear regression, binary and multiclass logistic regression, K-means clustering, and TensorFlow model importing
- If you decide to use these tools, you need to test their capabilities thoroughly to make sure they do what you need them to do

Step 3: Avoid Cloud Vendor-Lock in (4/4)

- If you choose a proprietary cloud database solution, you most likely won’t be able to access the software or the data in your local environment, and switching vendor would require migrating to a different database, which could be costly
- One way to address this problem is to choose vendors that support open source technologies
  - Another advantage of using open source technologies is that they tend to attract a larger community of users, meaning it will be easier for you to hire someone who has the experience and skills to work within your infrastructure
- Another way to address the problem is to choose third-party vendors (e.g., Pivotal Greenplum and Snowflake) that provide cloud database solutions using other major cloud providers as storage backend, which also allows you to store your data in multiple clouds if that fits your company’s needs

Step 4: Develop Robust Cloud Management Practices

- Finally, if you expect the company to grow, you need to put in place a robust cloud management practice to secure your cloud and prevent data loss and leakages
    - i.e., manage data access and secure interfaces and APIs, implement data governance best practices to maintain data quality and ensure your Data Lake will not turn into a Data Swamp

Agenda

1 Introduction

2 Unstructured Data and Big Data Tools

3 Relational Databases and NoSQL

4 Data Warehouse, Data Lakes, and Data Swamp

5 Distrib./Parallel Processing: Hadoop/Spark/MPP

6 Cloud Services

7 Case Study: Recommendation App DS Infra.

10 Summary and Conclusion

Lessons Learned So Far …

- There is clearly much more to an Enterprise computer/data science project than tuning the hyperparameters in machine learning models
- The goal of this high-level overview was to get you acquainted with the DMBSs aspects we will learn in the Database Systems' course

Any Questions?