Statistical Machine Learning

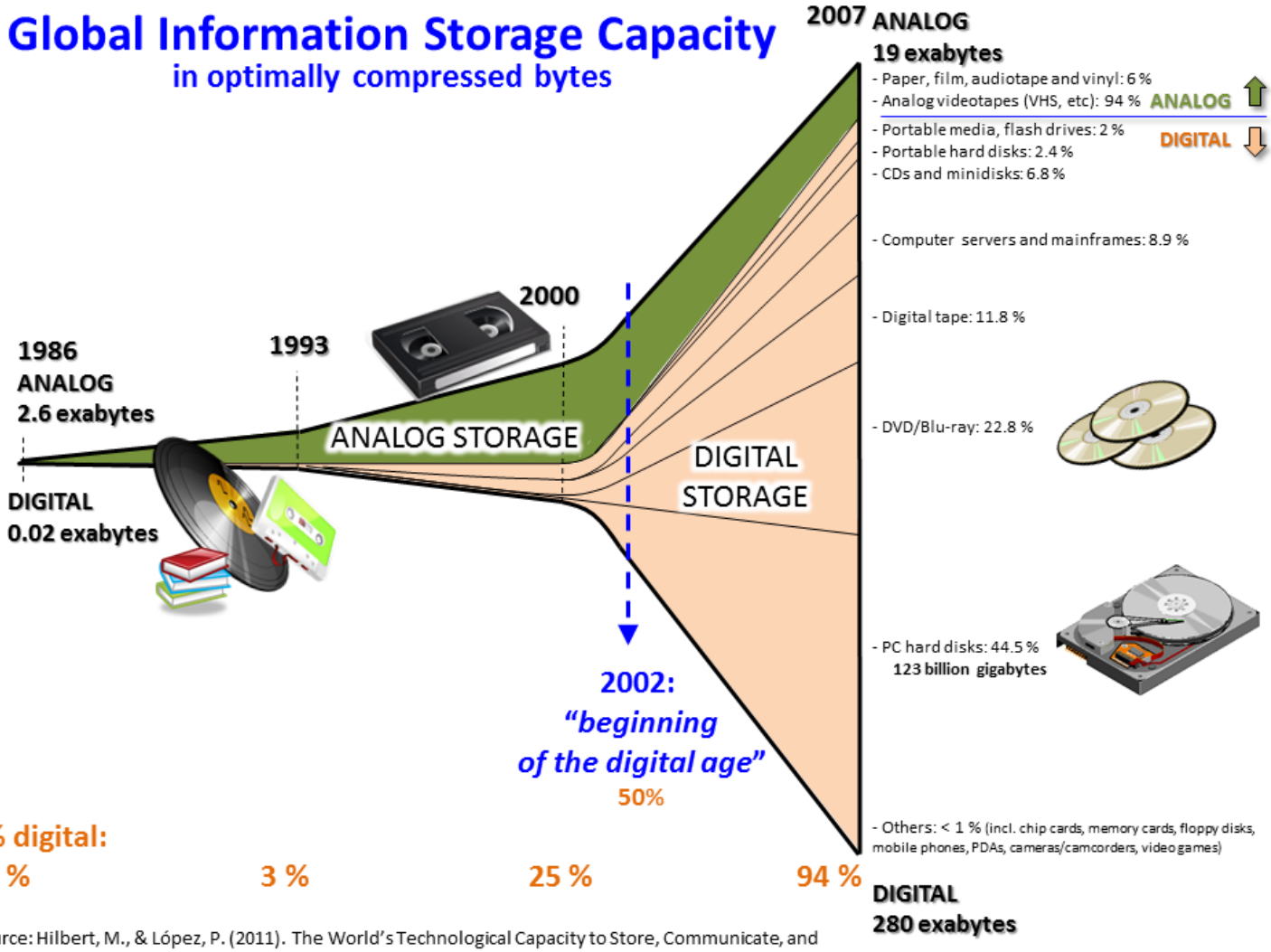
Big Data

Spring 2020

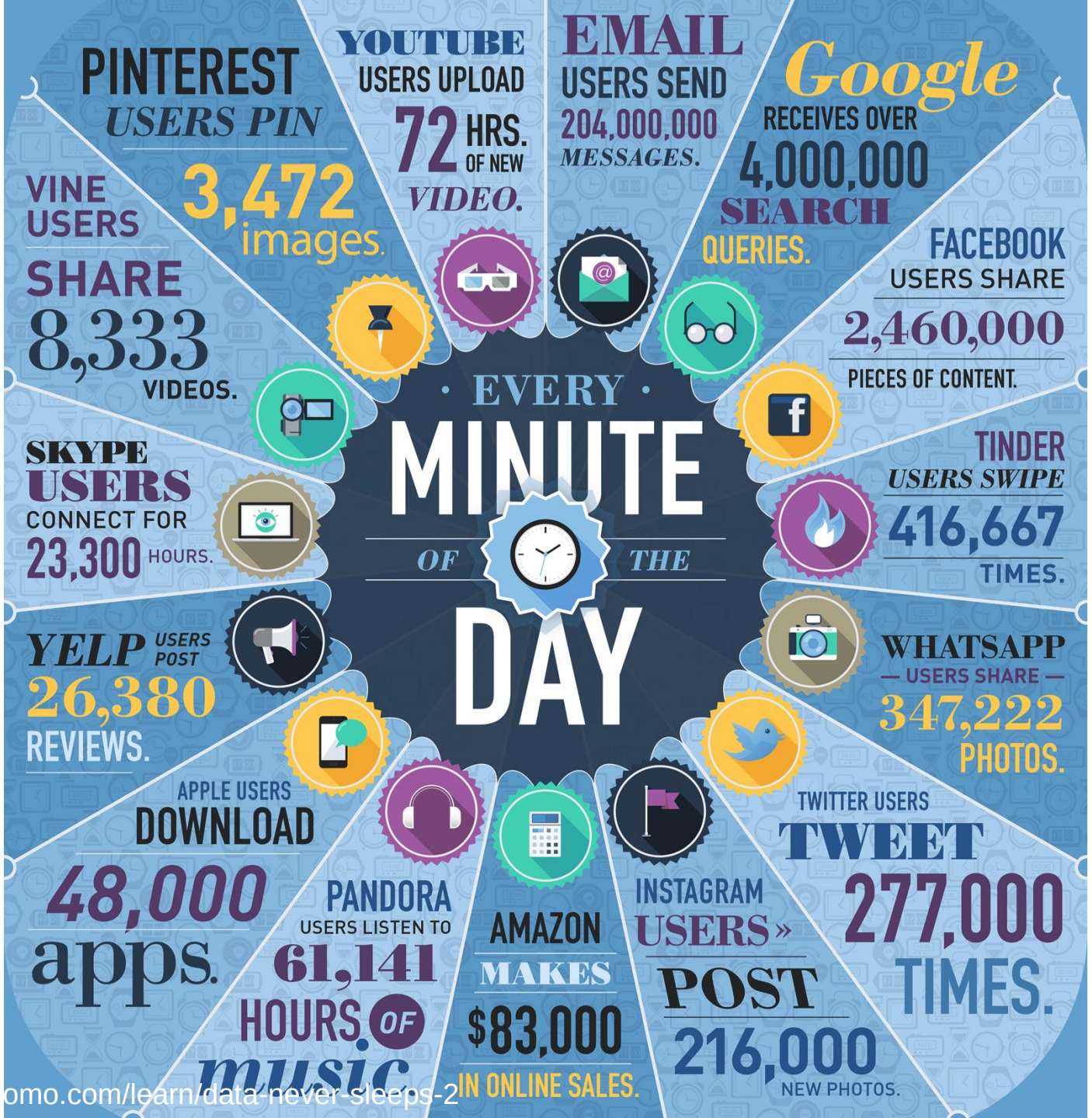
Topics

- Scope: Big Data & Analytics
- Topics:
  - Foundation of Data Analytics and Data Mining
  - Hadoop/Map-Reduce
  - Spark
  - Statistical Machine Learning & Big Data

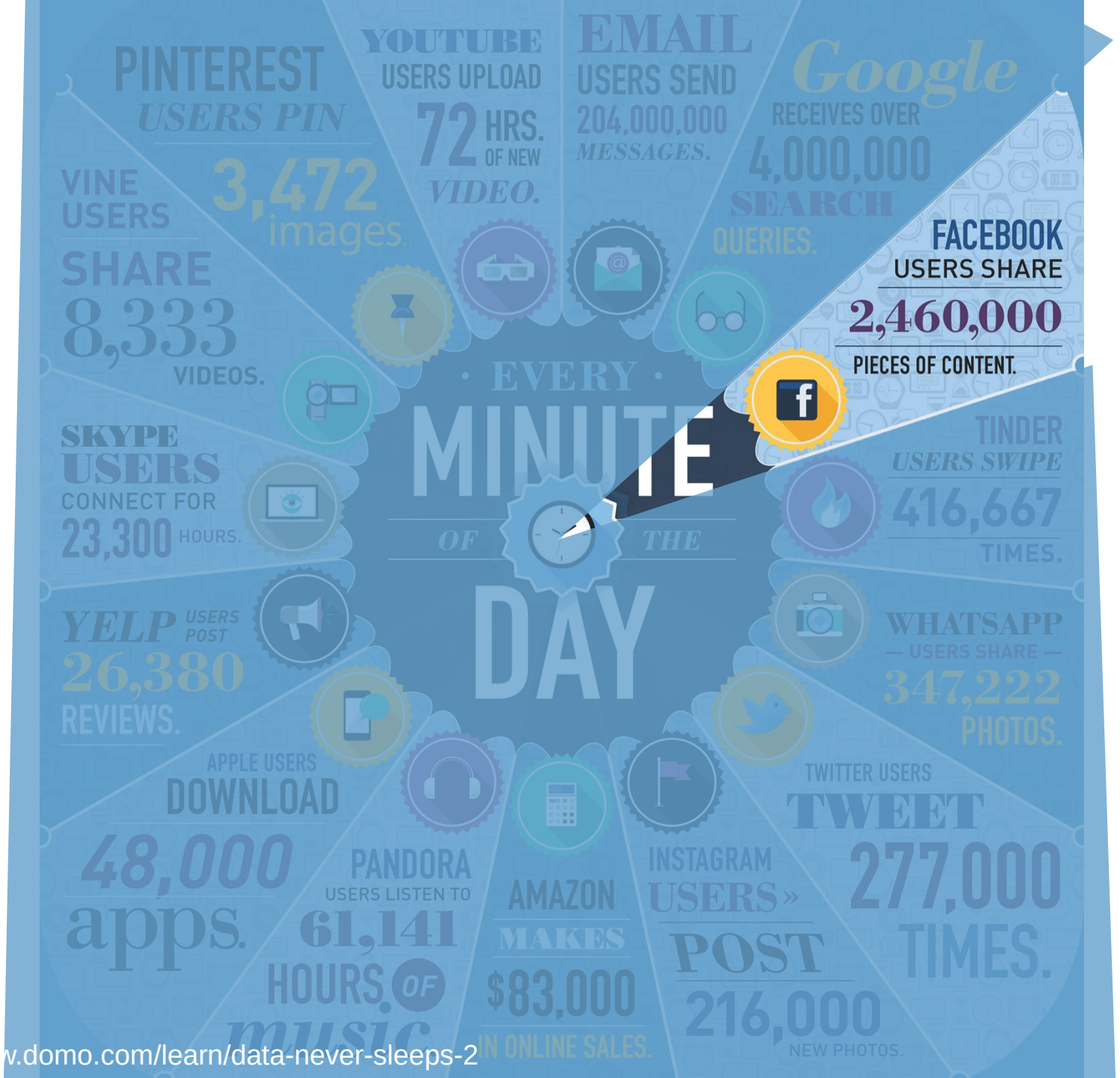
What's Big Data?



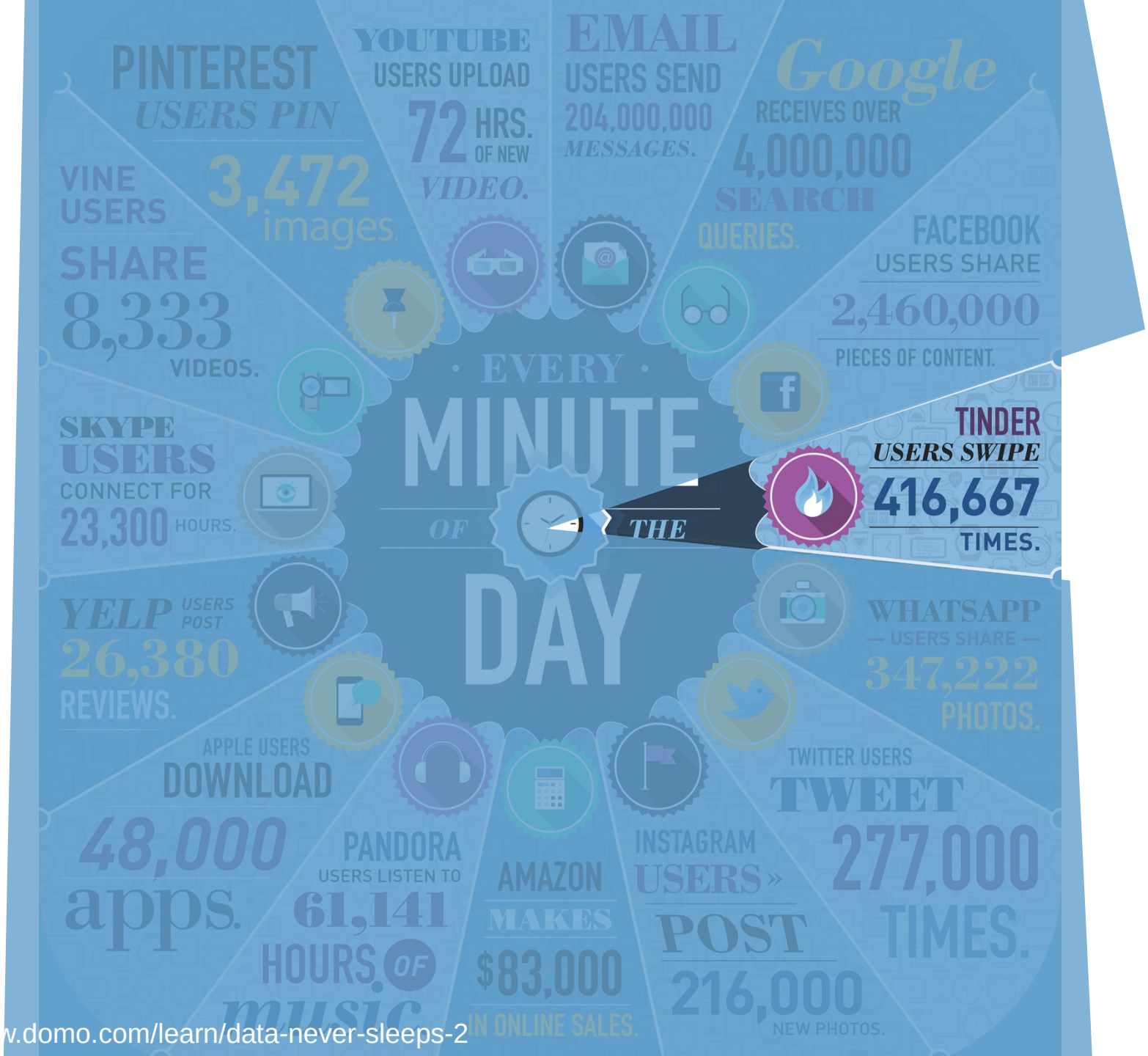
Source: Hilbert, M., & López, P. (2011). The World's Technological Capacity to Store, Communicate, and Compute Information. *Science*, 332(6025), 60 –65. http://www.martinhilbert.net/WorldInfoCapacity.html



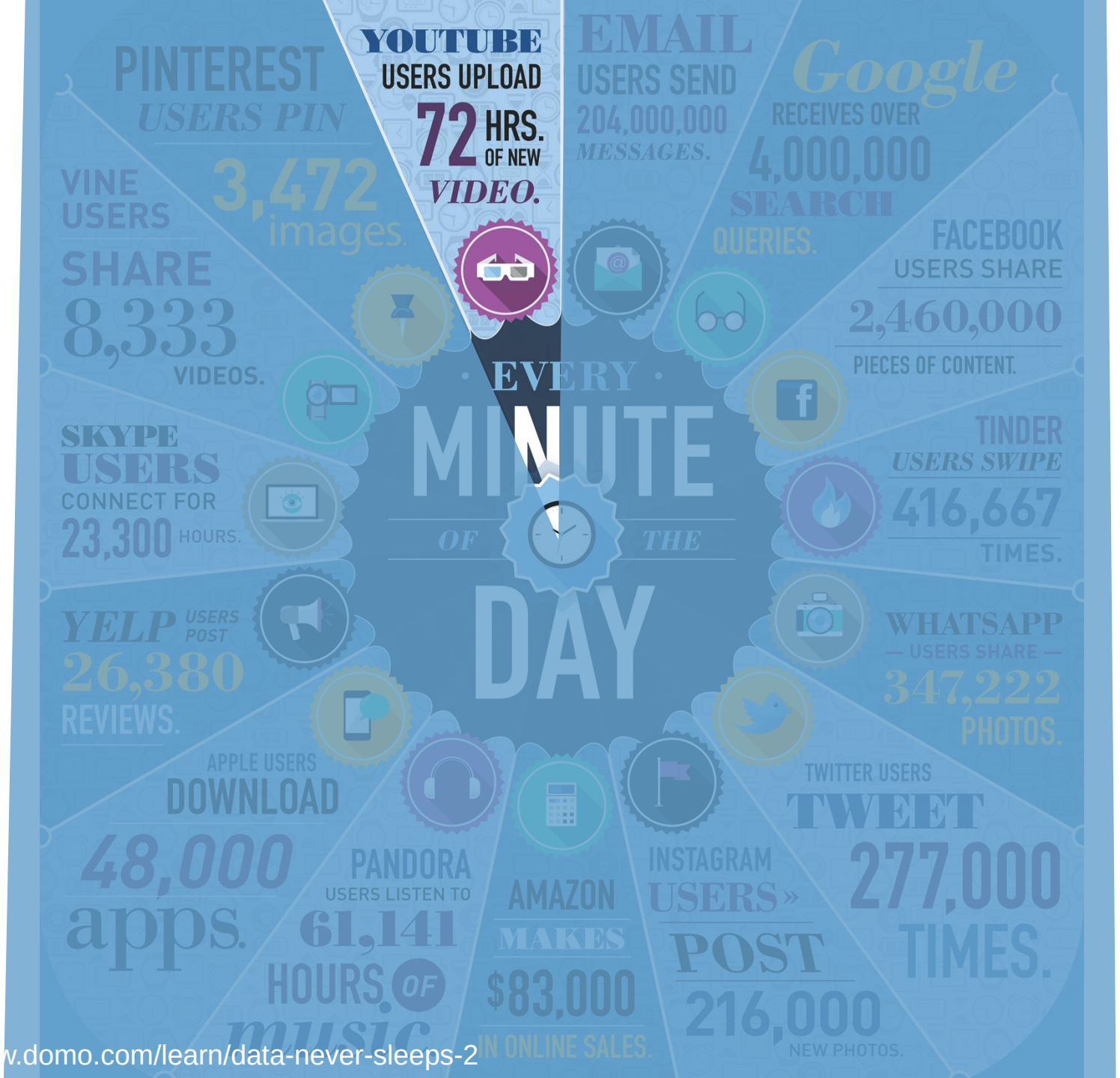
omo.com/learn/data-never-sleeps-2



w.domo.com/learn/data-never-sleeps-2



w.domo.com/learn/data-never-sleeps-2

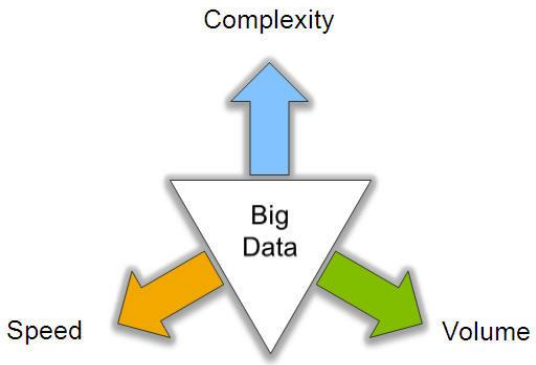


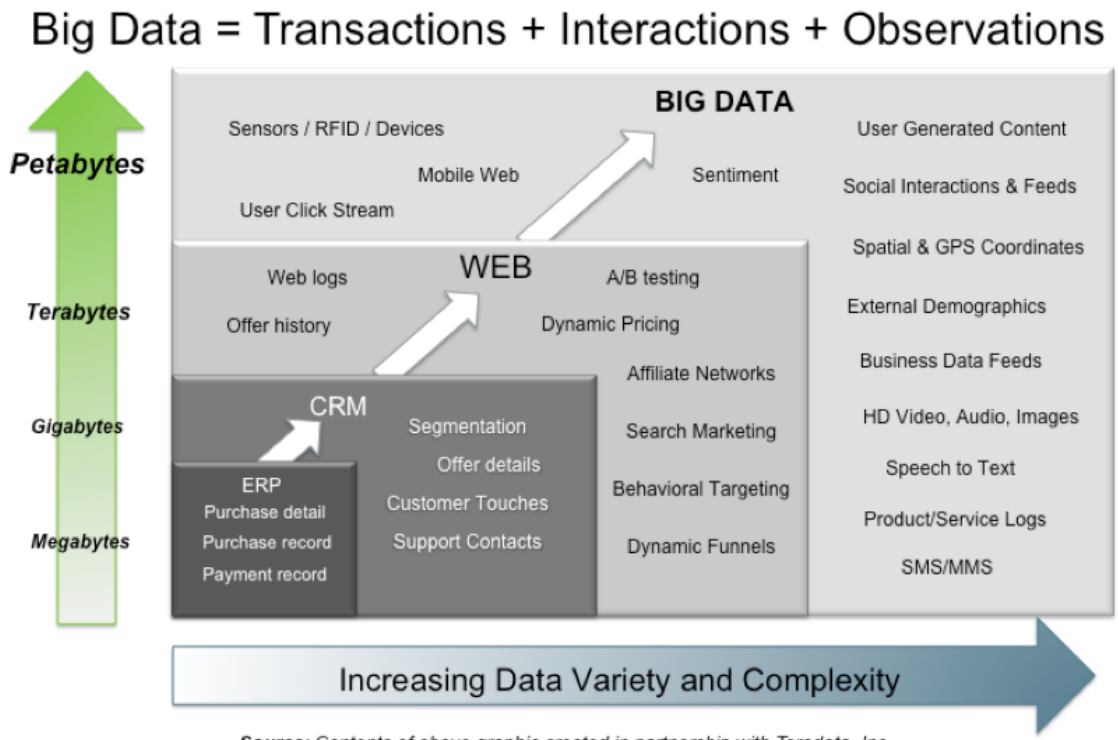
v.domo.com/learn/data-never-sleeps-2

What's Big Data?

- No single definition!
- Wikipedia: Big data is a field that treats ways to analyze, systematically extract information from, or otherwise deal with data sets that are too large or complex to be dealt with by traditional data-processing application software.
- Big Data Challenges: capture, curation, storage, search, sharing, transfer, analysis, and visualization.

Big Data: 3V's

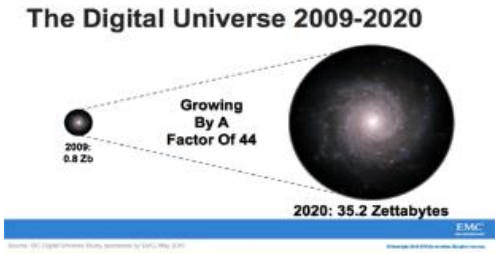


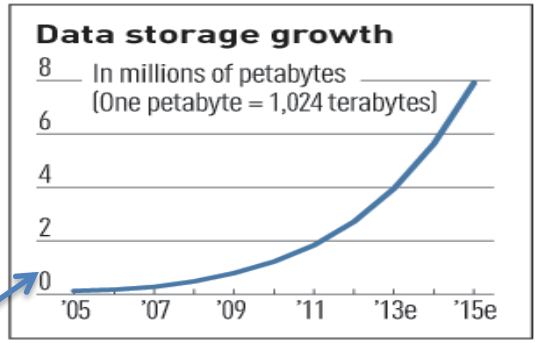


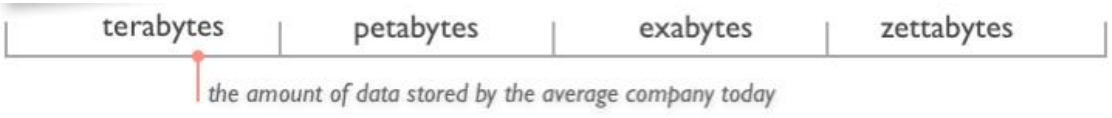
Source: Contents of above graphic created in partnership with Teradata, Inc.

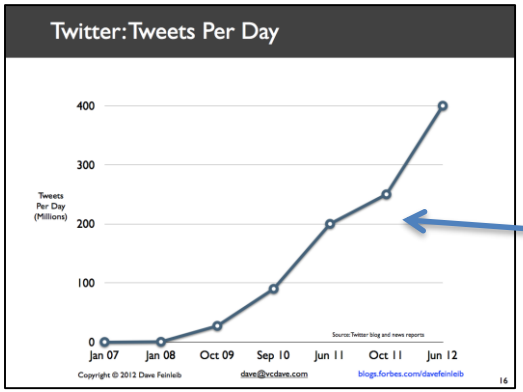
Volume (Scale)

- **Data Volume**
  - 44x increase from 2009 2020
  - From 0.8 zetta-bytes to 35zb
- Data volume is increasing exponentially









Exponential increase in collected/generated data

Volume (Scale)

12+ TBs
of tweet data every day

? TBs of data every day

25+ TBs **of log data every day**

30 billion RFID tags today (1.3B in 2005)

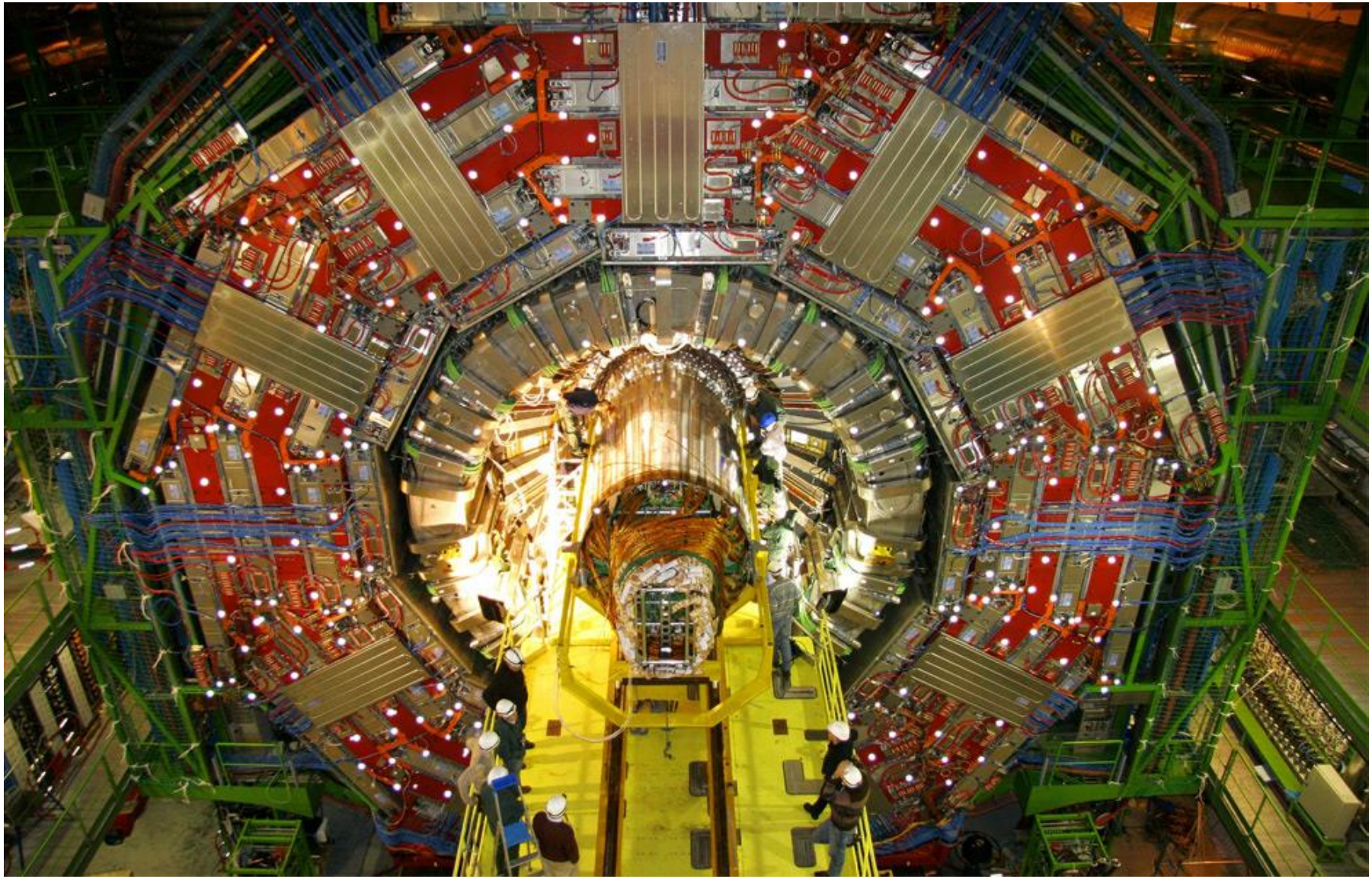
4.6 billion camera phones world wide

100s of millions of GPS enabled devices sold annually

2+ billion people on the Web by end 2011

76 million smart meters in 2009…
200M by 2014

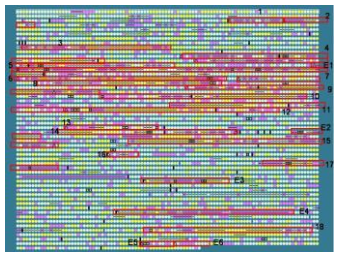
Volume (Scale)

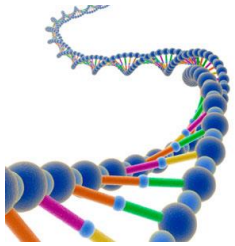


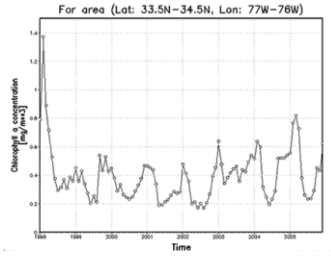
CERN’s Large Hydron Collider (LHC) generates 15 PB a year

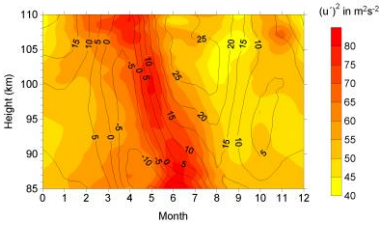
Variety (Complexity)

- Relational Data (Tables/Transaction/Legacy Data)
- Text Data (Web)
- Semi-structured Data (XML)
- Graph Data
  - Social Network, Semantic Web (RDF), …
- Streaming Data
  - You can only scan the data once
- A single application can be generating/collecting many types of data
- Big Public Data (online, weather, finance, etc)



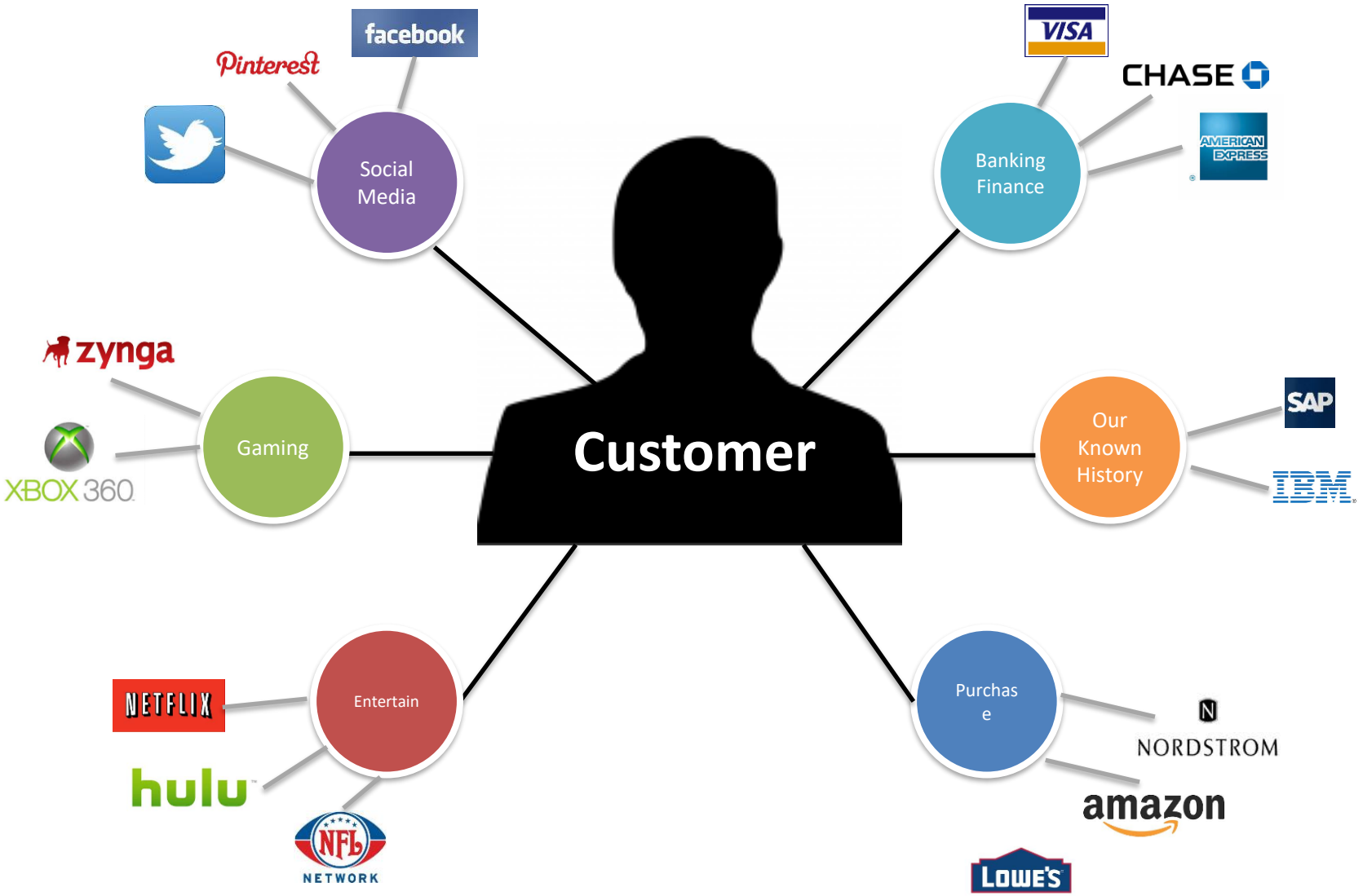






To extract knowledge➔ all these types of data need to linked together

A Single View to the Customer



Velocity (Speed)

- Data is begin generated fast and need to be processed fast
- Online Data Analytics
- Late decisions ➔ missing opportunities

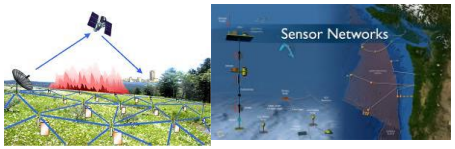
- **Examples**
  - **E-Promotions:** Based on your current location, your purchase history, what you like ➔ send promotions right now for store next to you
  - **Healthcare monitoring:** sensors monitoring your activities and body ➔ any abnormal measurements require immediate reaction

Real-time/Fast Data

Social media and networks
(all of us are generating data)

Scientific instruments
(collecting all sorts of data)

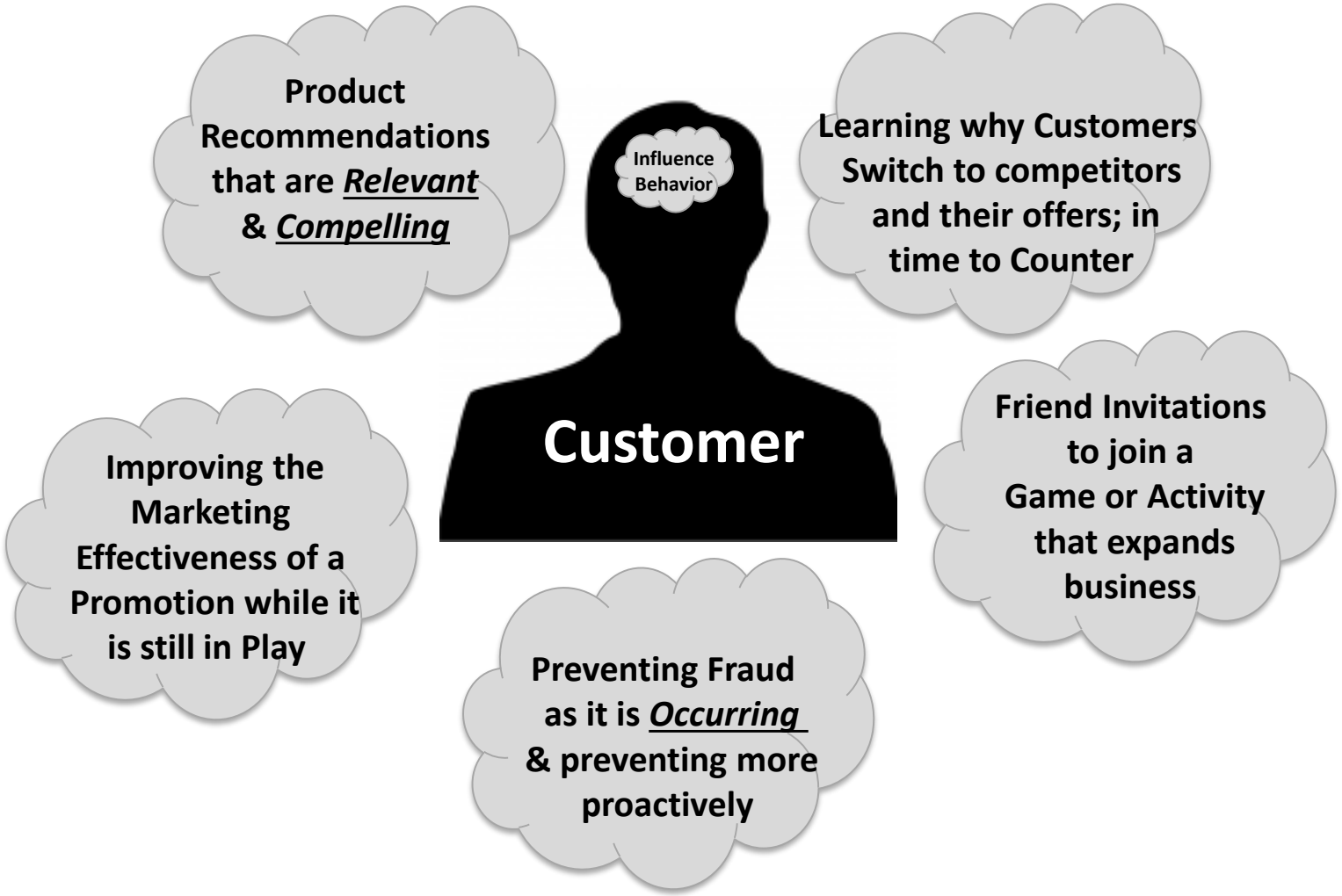
Mobile devices
(tracking all objects all the time)



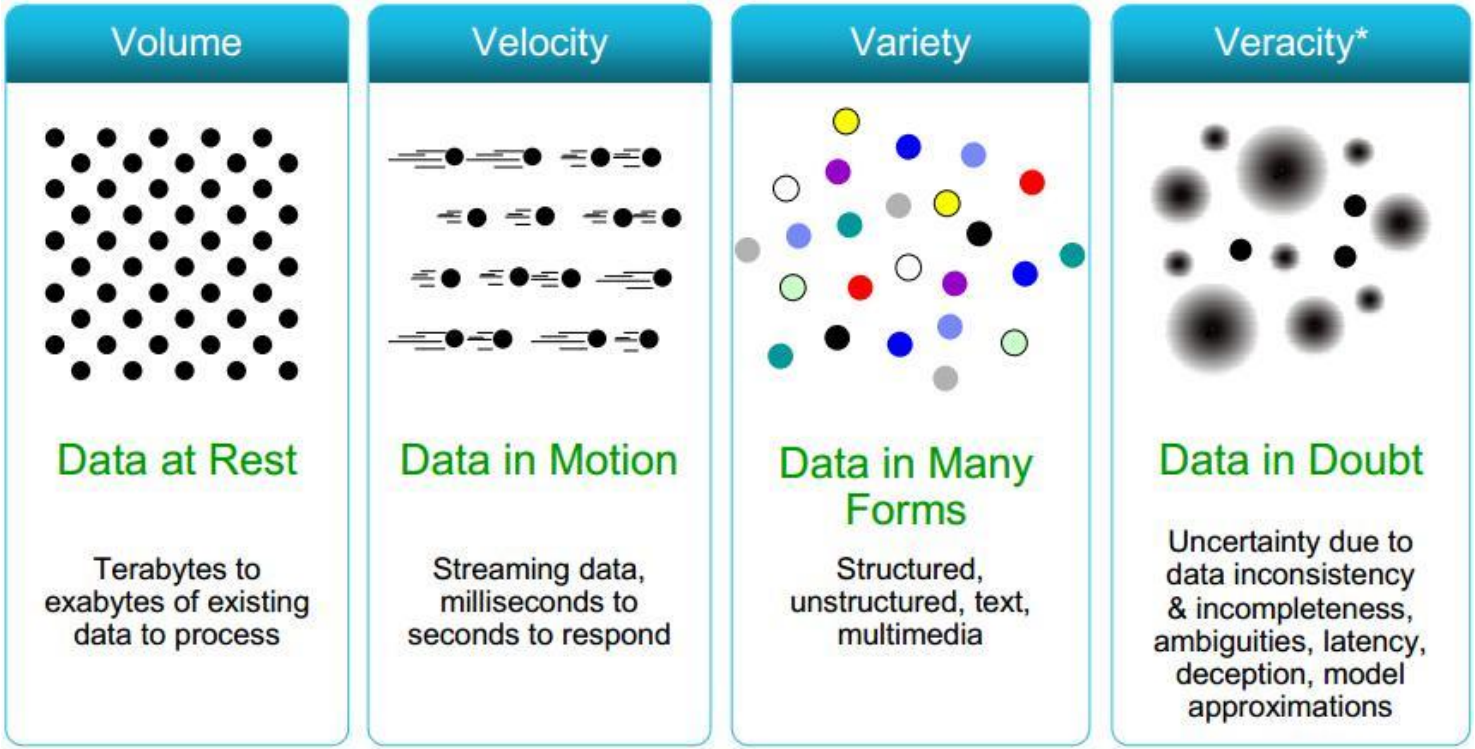
Sensor technology and networks
(measuring all kinds of data)

- The progress and innovation is no longer hindered by the ability to collect data
- But, by the ability to manage, analyze, summarize, visualize, and discover knowledge from the collected data in a timely manner and in a scalable fashion

Real-Time Analytics/Decision Requirement

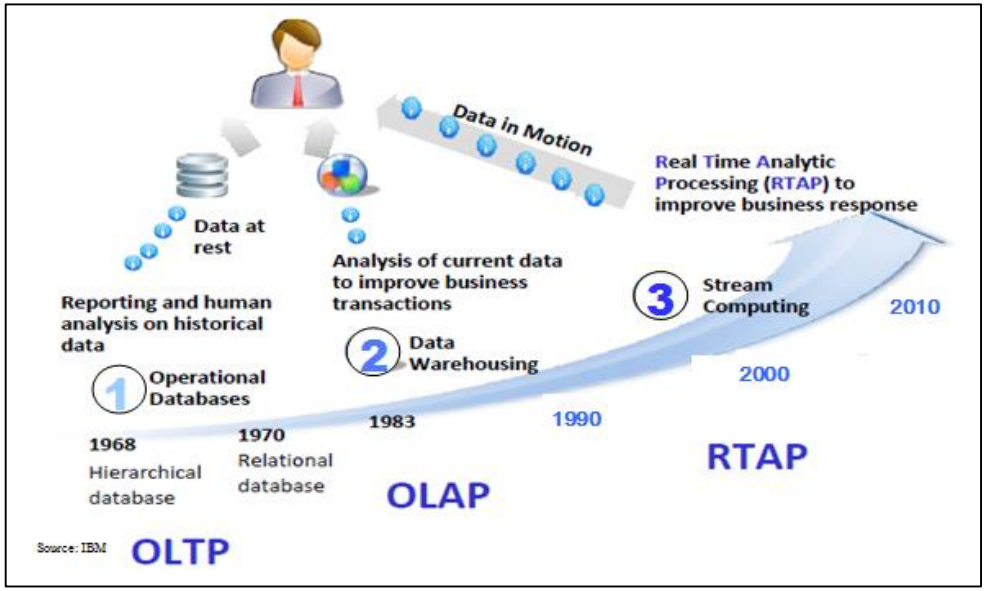


Some Make it 4V's



And Even Five: Value

Harnessing Big Data



- **OLTP:** Online Transaction Processing (DBMSs)
- **OLAP:** Online Analytical Processing (Data Warehousing)
- **RTAP:** Real-Time Analytics Processing (Big Data Architecture & technology)

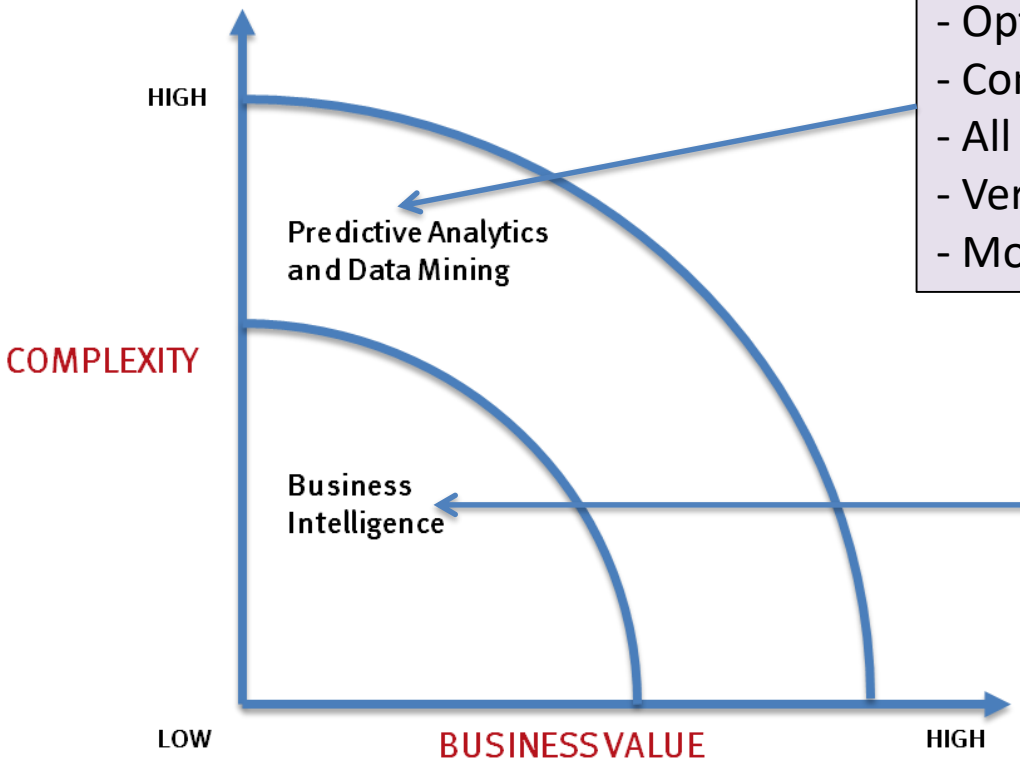
The Model Has Changed…

- **The Model of Generating/Consuming Data has Changed**

Old Model: Few companies are generating data, all others are consuming data

New Model: all of us are generating data, and all of us are consuming data

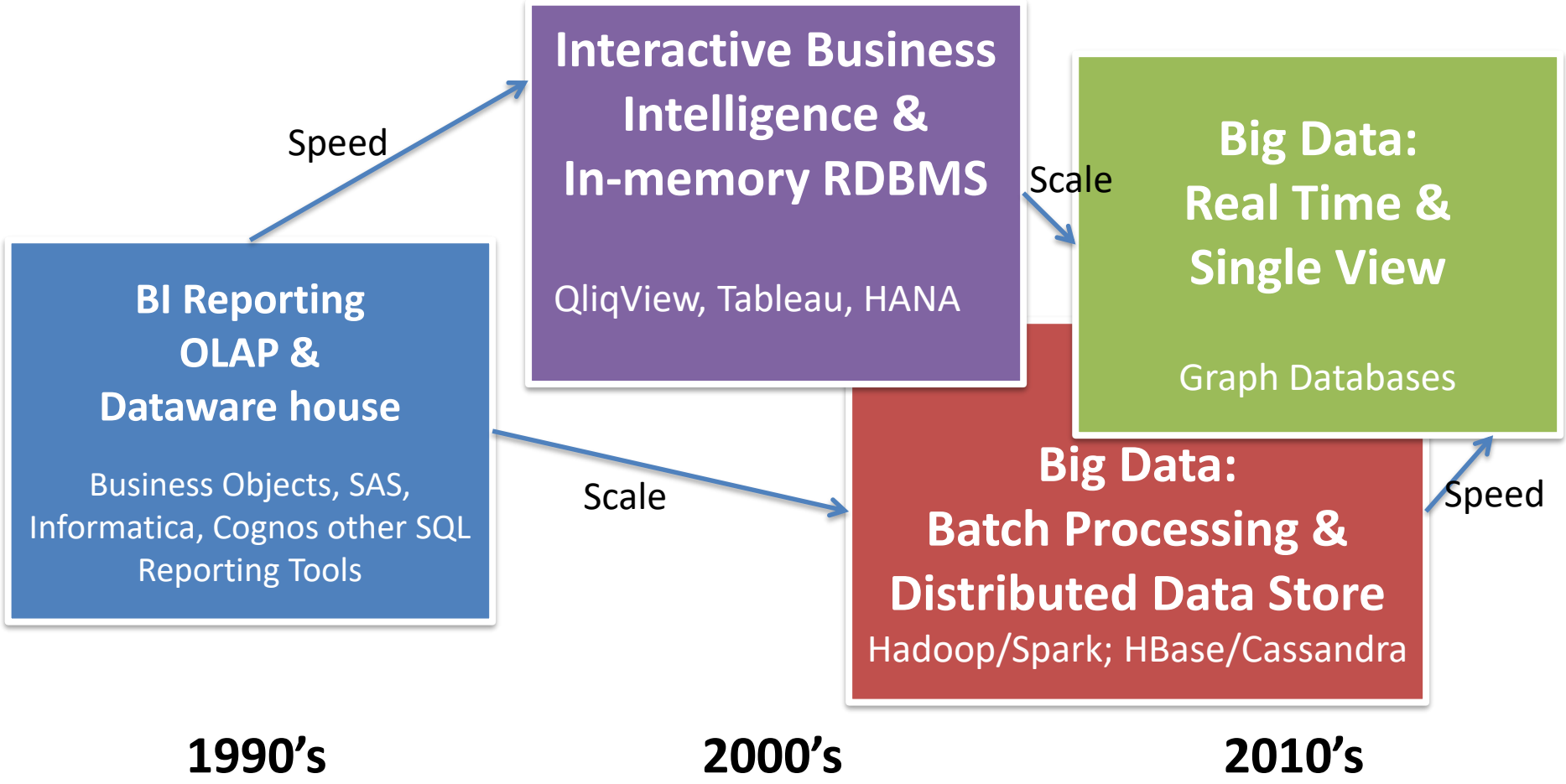
What's driving Big Data



- Optimizations and predictive analytics
- Complex statistical analysis
- All types of data, and many sources
- Very large datasets
- More of a real-time

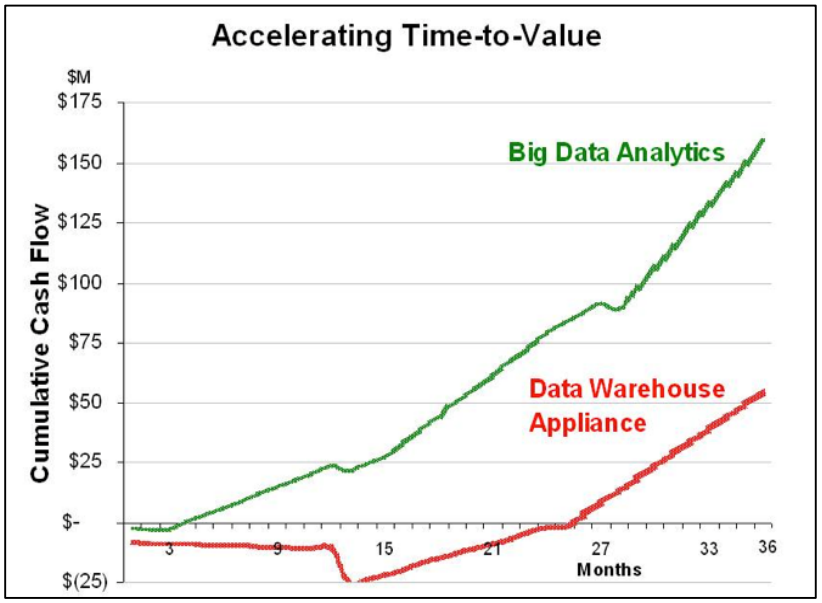
- Ad-hoc querying and reporting
- Data mining techniques
- Structured data, typical sources
- Small to mid-size datasets

THE EVOLUTION OF BUSINESS INTELLIGENCE

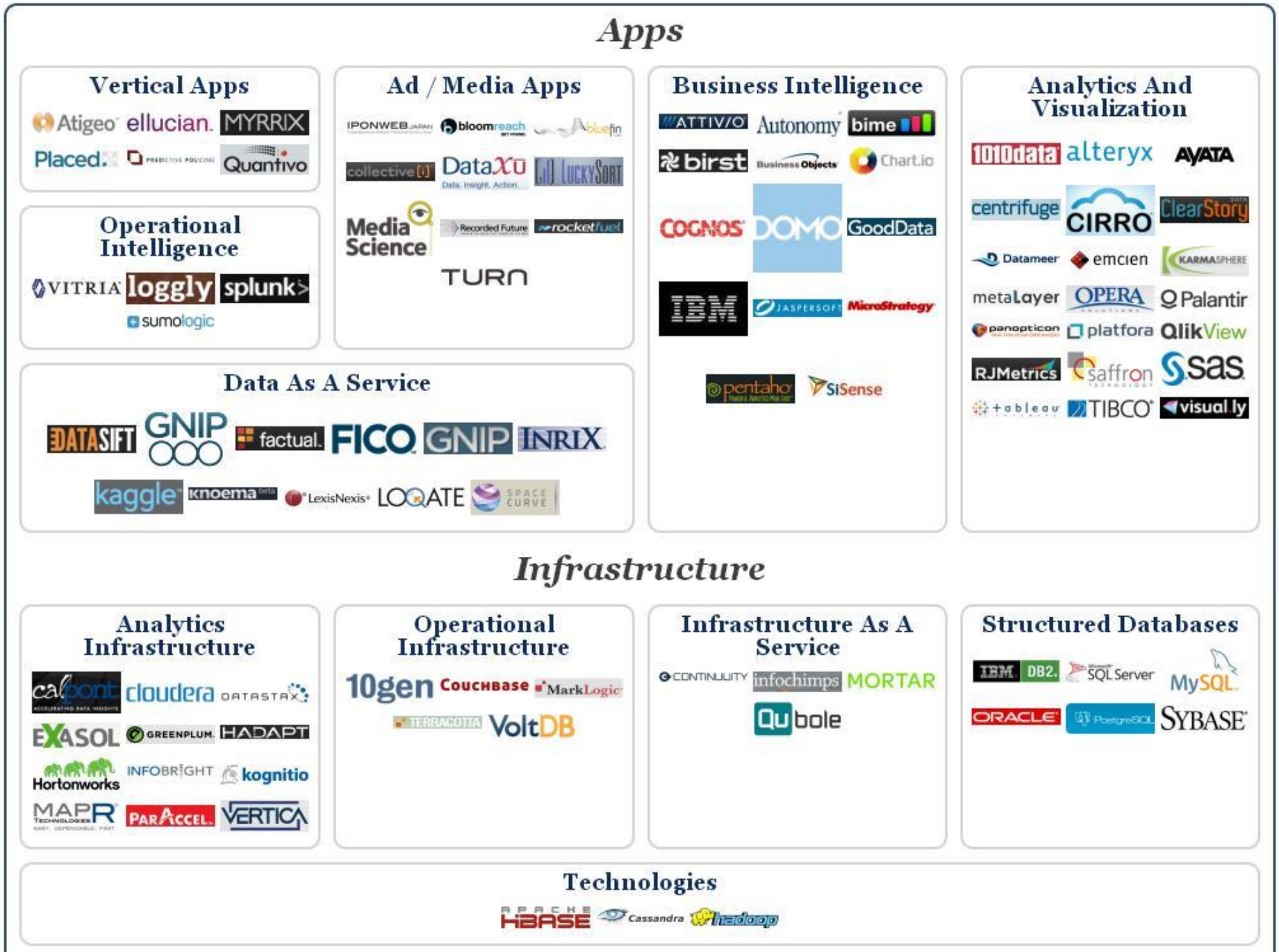


Big Data Analytics

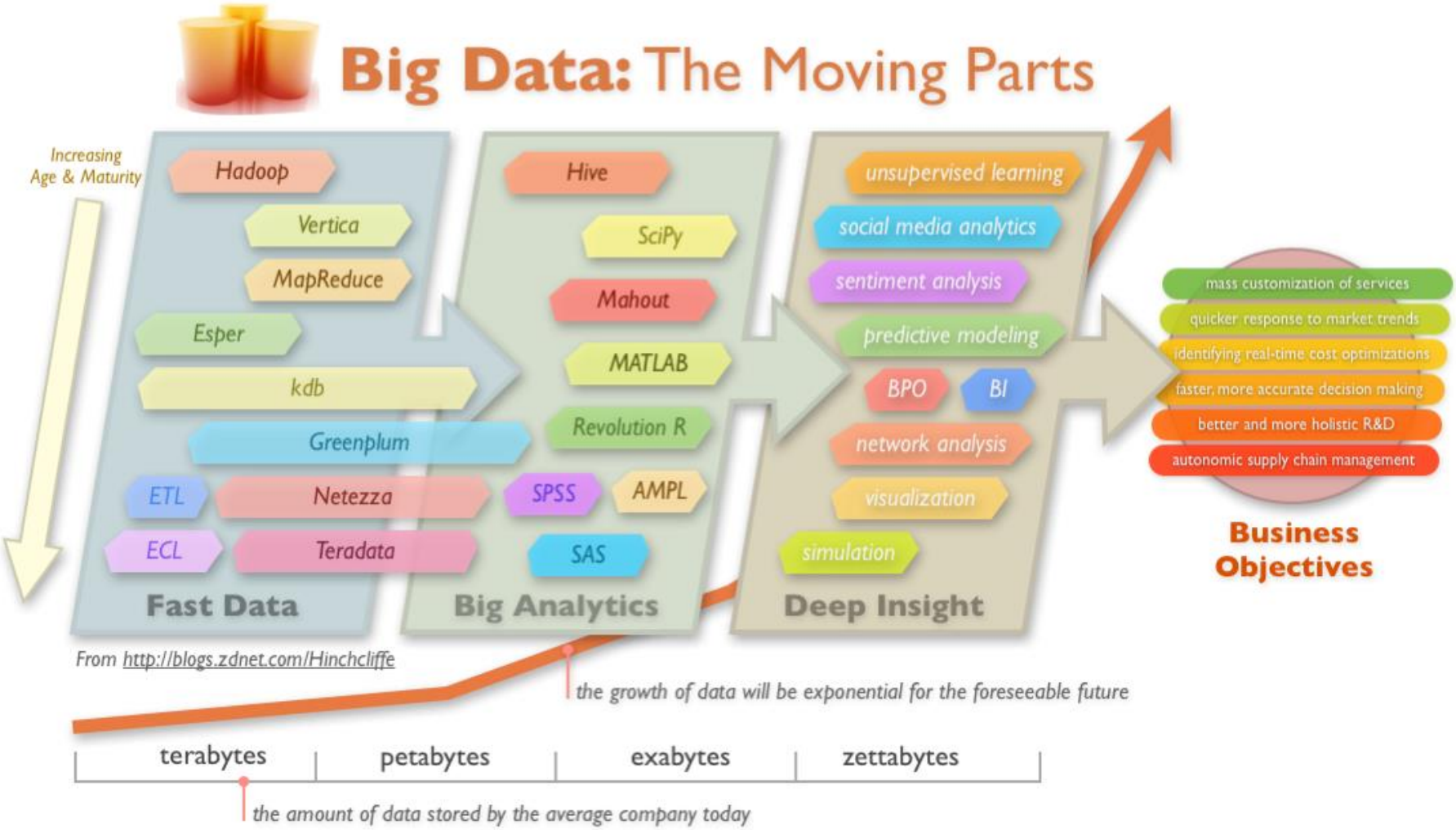
- Big data is more real-time in nature than traditional DW applications
- Traditional DW architectures (e.g. Exadata, Teradata) are not well-suited for big data apps
- Massively parallel processing, scale out architectures are well-suited for big data apps



The Big Data Landscape

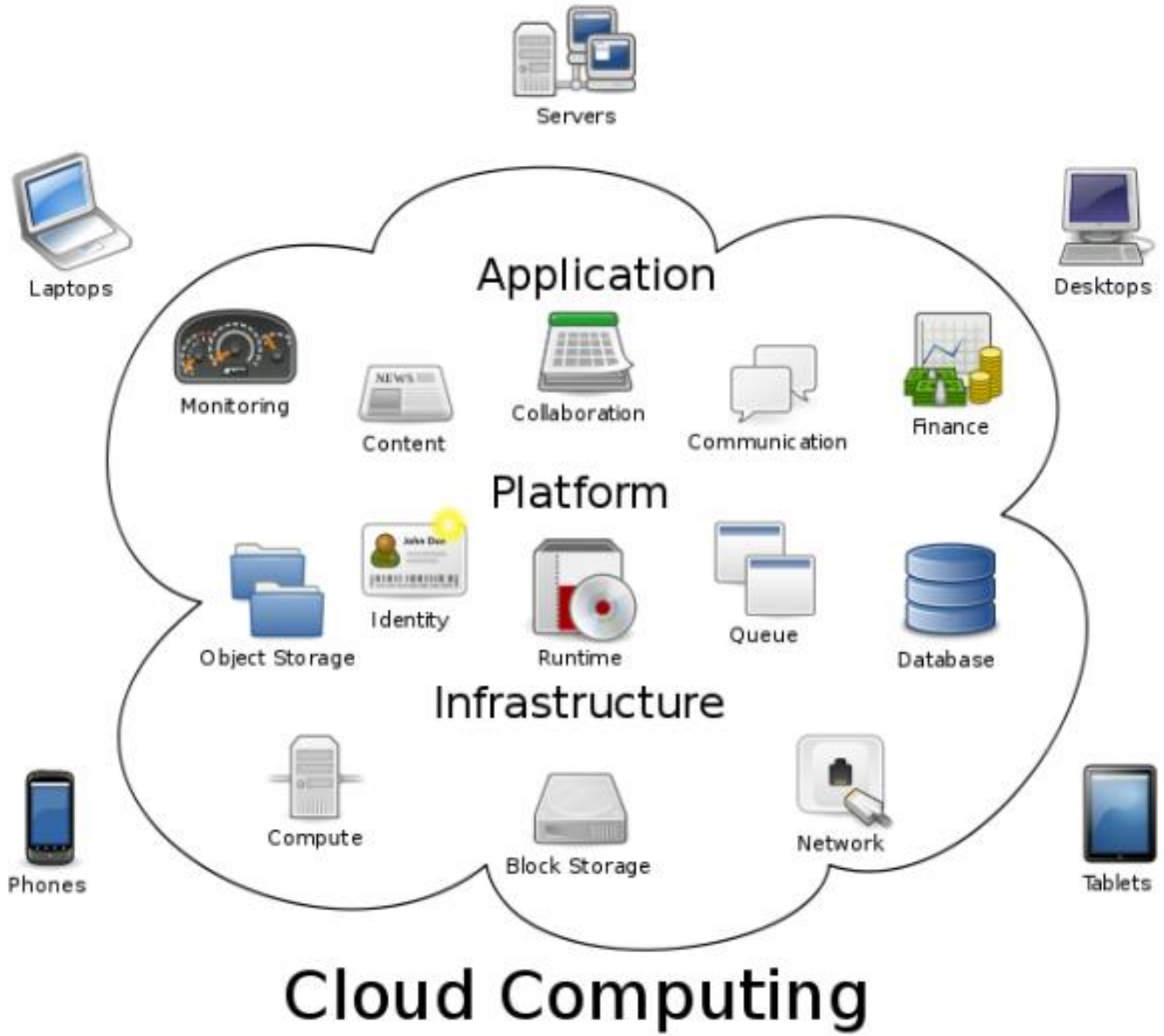


Big Data Technology



Cloud Computing

- IT resources provided as a service
  - Compute, storage, databases, queues
- Clouds leverage economies of scale of commodity hardware
  - Cheap storage, high bandwidth networks & multicore processors
  - Geographically distributed data centers
- Offerings from Microsoft, Amazon, Google, …



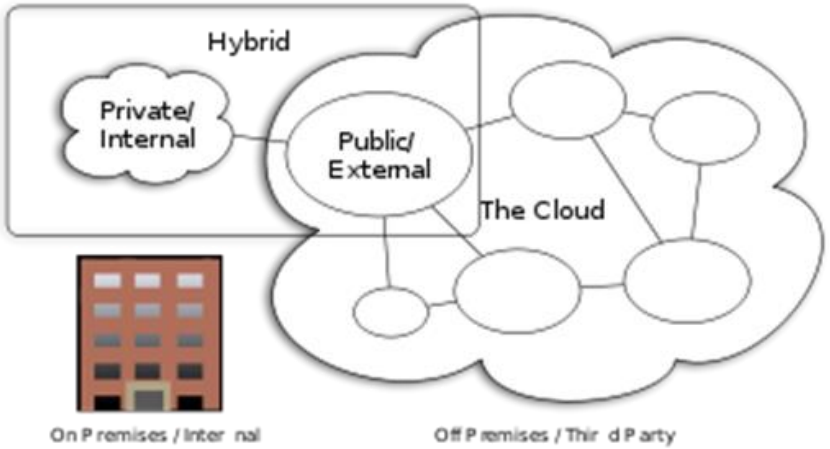
wikipedia:Cloud Computing

Benefits

- Cost & management
  - Economies of scale, “out-sourced” resource management
- Reduced Time to deployment
  - Ease of assembly, works “out of the box”
- Scaling
  - On demand provisioning, co-locate data and compute
- Reliability
  - Massive, redundant, shared resources
- Sustainability
  - Hardware not owned

Types of Cloud Computing

- **Public Cloud**: Computing infrastructure is hosted at the vendor's premises.
- **Private Cloud**: Computing architecture is dedicated to the customer and is not shared with other organizations.
- **Hybrid Cloud**: Organizations host some critical, secure applications in private clouds. The not so critical applications are hosted in the public cloud
  - **Cloud bursting**: the organization uses its own infrastructure for normal usage, but cloud is used for peak loads.
- **Community Cloud**



Classification of Cloud Computing based on Service Provided

- Infrastructure as a service (IaaS)
  - Offering hardware related services using the principles of cloud computing. These could include storage services (database or disk storage) or virtual servers.
  - Amazon EC2, Amazon S3, Rackspace Cloud Servers and Flexiscale.
- Platform as a Service (PaaS)
  - Offering a development platform on the cloud.
  - Google's Application Engine, Microsofts Azure, Salesforce.com's force.com .
- Software as a service (SaaS)
  - Including a complete software offering on the cloud. Users can access a software application hosted by the cloud vendor on pay-per-use basis. This is a well-established sector.
  - Salesforce.coms' offering in the online Customer Relationship Management (CRM) space, Googles gmail and Microsofts hotmail, Google docs.

More Refined Categorization

- Storage-as-a-service
- Database-as-a-service
- Information-as-a-service
- Process-as-a-service
- Application-as-a-service
- Platform-as-a-service
- Integration-as-a-service
- Security-as-a-service
- Management/ Governance-as-a-service
- Testing-as-a-service
- Infrastructure-as-a-service

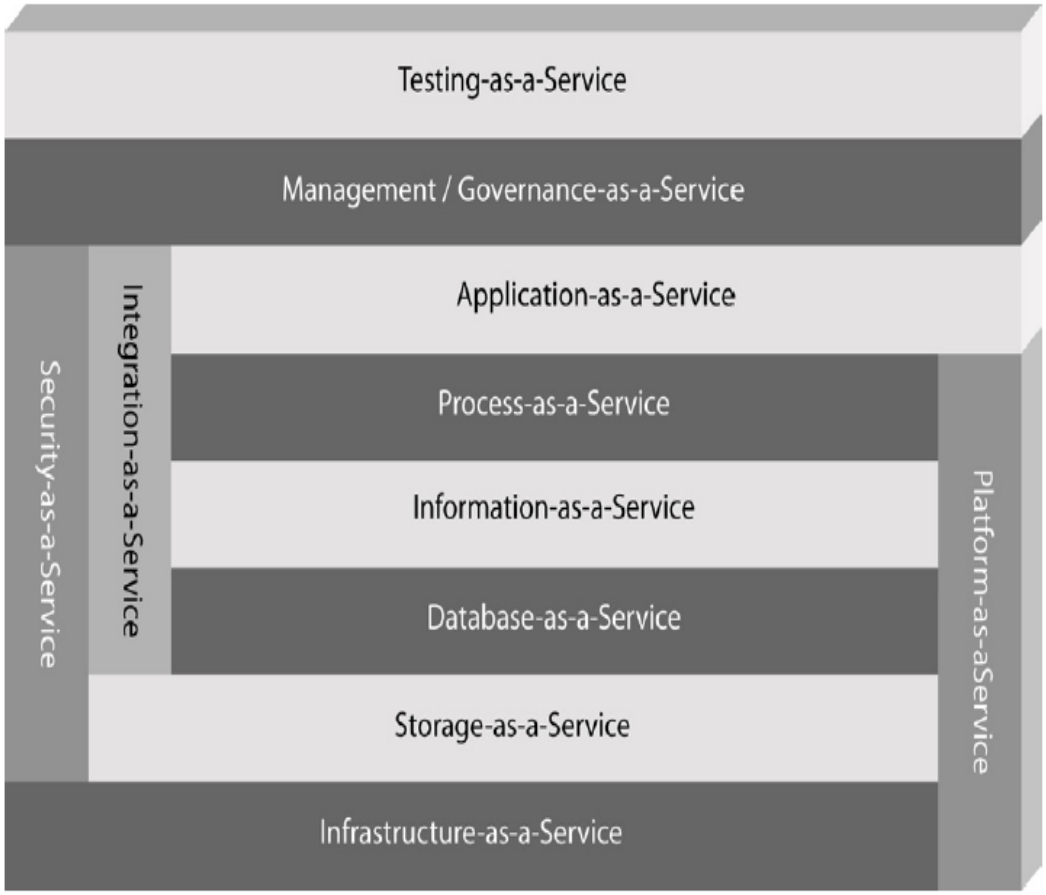


Figure 1: The patterns or categories of cloud computing providers allow you to use a discrete set of services within your architecture.

InfoWorld Cloud Computing Deep Dive

Summary: Key Ingredients in Cloud Computing

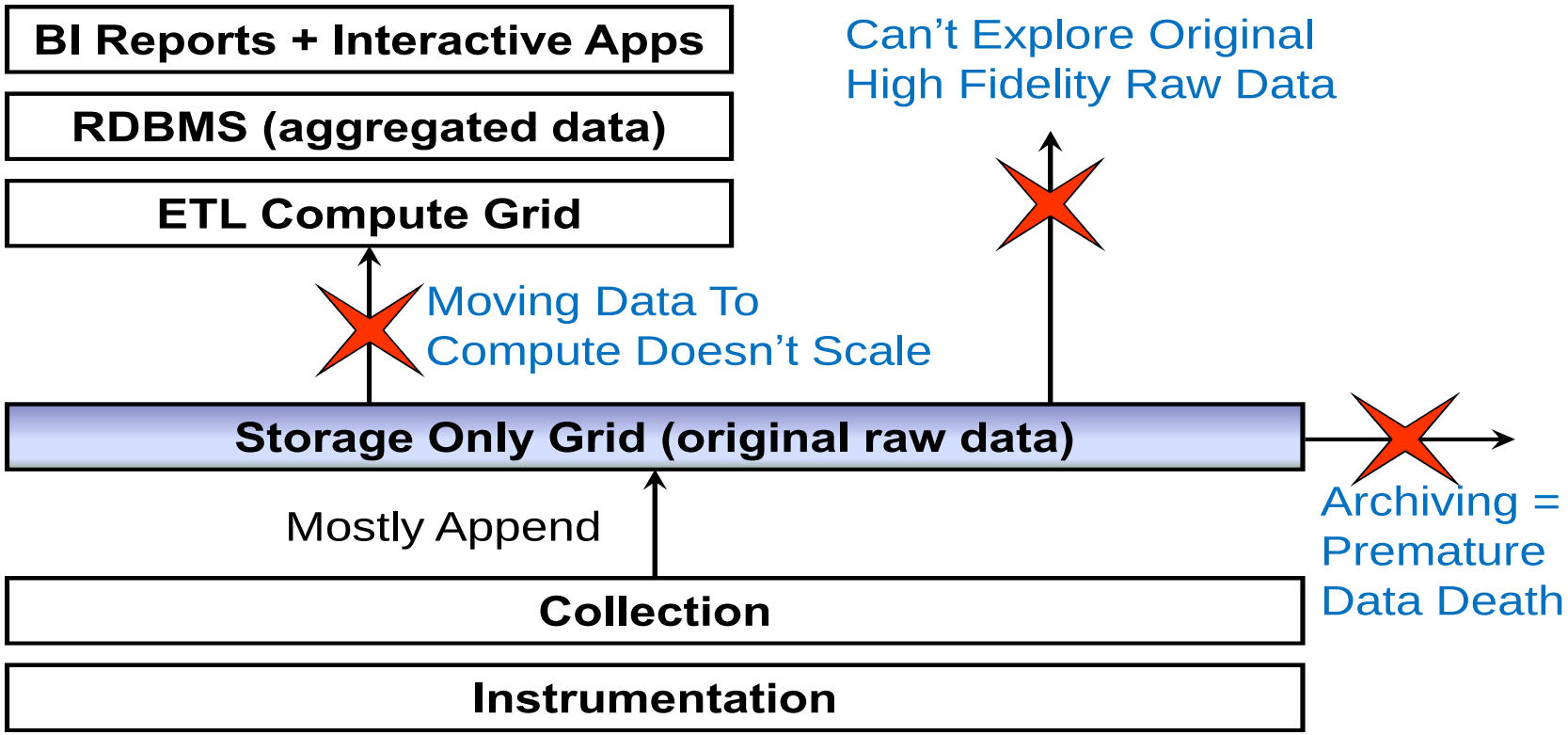
- Service-Oriented Architecture (SOA)
- Utility Computing (on demand)
- Virtualization (P2P Network)
- SAAS (Software As A Service)
- PAAS (Platform AS A Service)
- IAAS (Infrastructure AS A Servie)
- Web Services in Cloud

Big Data Processing & Statistical Analysis

MapReduce

is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster.

Limitations of Existing Data Analytics Architecture



Slides from **Dr. Amr Awadallah's Hadoop talk at Stanford, CTO & VPE from Cloudera**

Typical Large-Data Problem

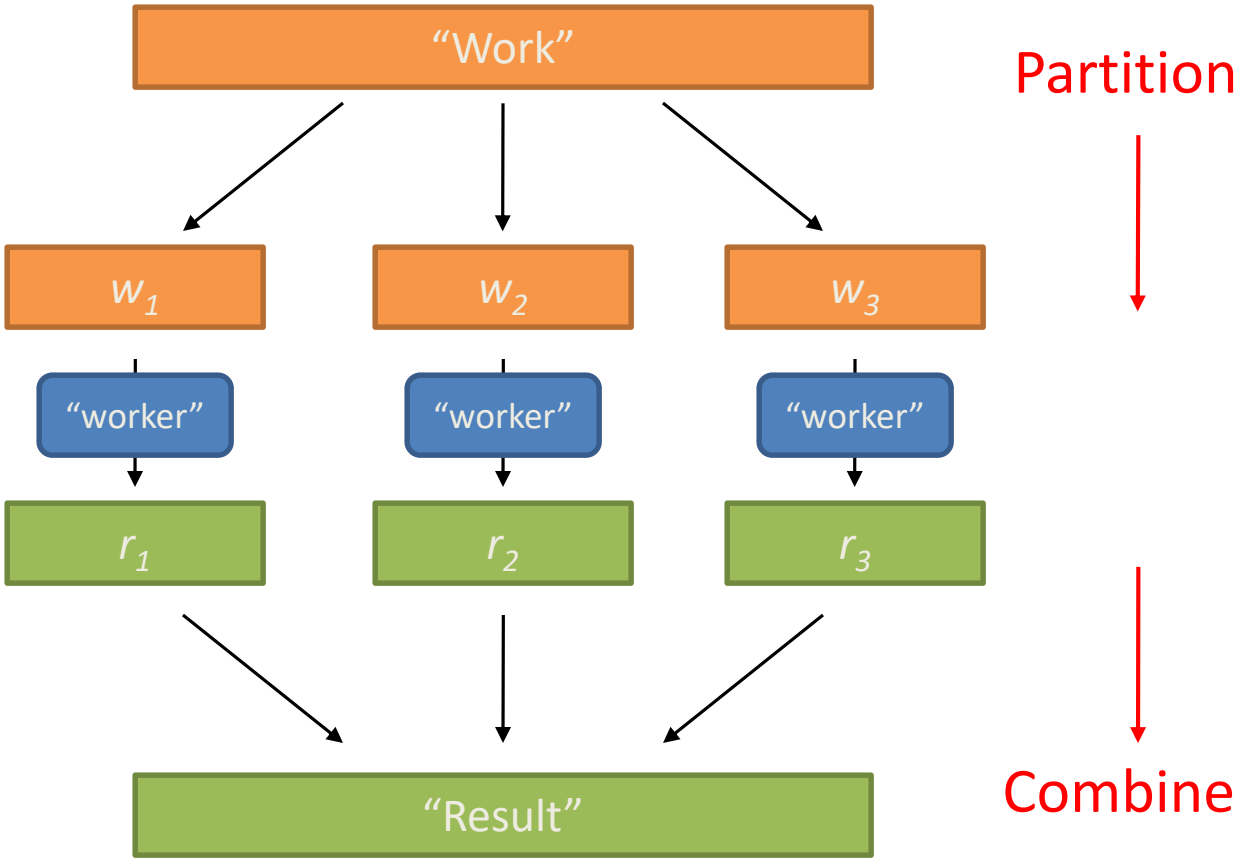
- Iterate over a large number of records
- Extract something of interest from each
- Shuffle and sort intermediate results
- Aggregate intermediate results
- Generate final output

- The problem:
  - Diverse input format (data diversity & heterogeneity)
  - Large Scale: Terabytes, Petabytes
  - Parallelization

How to leverage a number of cheap off-the-shelf computers?

Image from http://wiki.apache.org/hadoop-data/attachments/HadoopPresentations/attachments/aw-apachecon-eu-2009.pdf

MapReduce Philosophy: Divide and Conquer



Parallelization Challenges

- How do we assign work units to workers?
- What if we have more work units than workers?
- What if workers need to share partial results?
- How do we aggregate partial results?
- How do we know all the workers have finished?
- What if workers die?

Common Theme?

- Parallelization problems arise from:
  - Communication between workers (e.g., to exchange state)
  - Access to shared resources (e.g., data)
- Thus, we need a synchronization mechanism

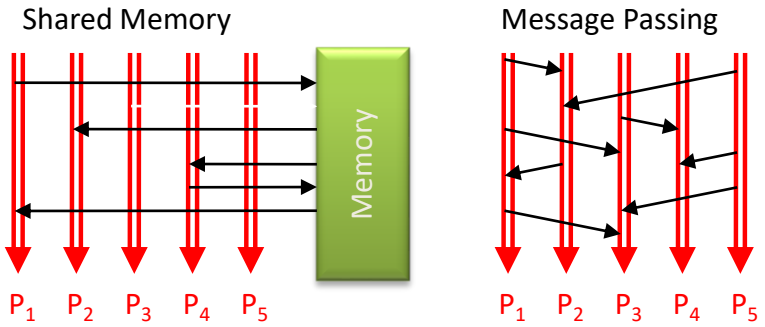
Source: Ricardo Guimarães Herrmann

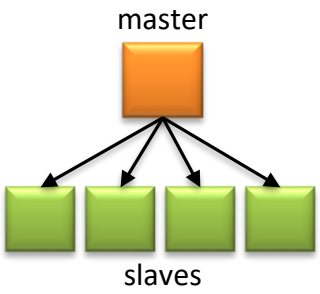
Managing Multiple Workers

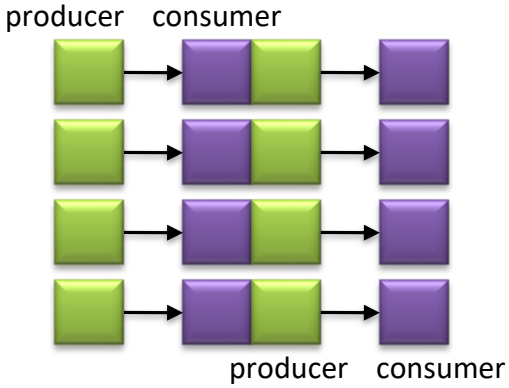
- Difficult because
  - We don't know the order in which workers run
  - We don't know when workers interrupt each other
  - We don't know the order in which workers access shared data
- Thus, we need:
  - Semaphores (lock, unlock)
  - Conditional variables (wait, notify, broadcast)
  - …
- Still, lots of problems:
  - Deadlock, livelock, race conditions...
  - …
- Moral of the story: be careful!

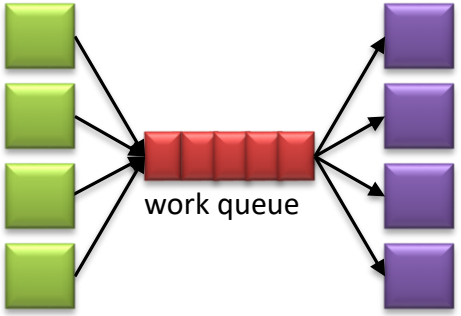
Current Tools

- Programming models
  - Shared memory (p threads)
  - Message passing (MPI)
- Design Patterns
  - Master-slaves
  - Producer-consumer flows
  - Shared work queues









Hadoop
MapReduce: Big Data Processing Abstraction

Apache Hadoop

- Scalable fault-tolerant distributed system for Big Data:
  - Data Storage
  - Data Processing
  - A virtual Big Data machine
  - Borrowed concepts/Ideas from Google; Open source under the Apache license

- Core Hadoop has two main systems:
  - **Hadoop/MapReduce**: distributed big data processing infrastructure (abstract/paradigm, fault-tolerant, schedule, execution)
  - **HDFS (Hadoop Distributed File System)**: fault-tolerant, high-bandwidth, high availability distributed storage

Typical Large-Data Problem

Map

- Iterate over a large number of records
- Extract something of interest from each
- Shuffle and sort intermediate results
- Aggregate intermediate results
- Generate final output

Reduce

Key idea: provide a functional abstraction for these two operations!

MapReduce

- Programmers specify two functions:

  map (k, v) → [(k', v')]

  reduce (k', [v']) → [(k', v')]

  – All values with the same key are sent to the same reducer

- The execution framework handles everything else…

MapReduce “Runtime”

- Handles scheduling
  - Assigns workers to map and reduce tasks
- Handles “data distribution”
  - Moves processes to data
- Handles synchronization
  - Gathers, sorts, and shuffles intermediate data
- Handles errors and faults
  - Detects worker failures and restarts
- Everything happens on top of a distributed FS (later)

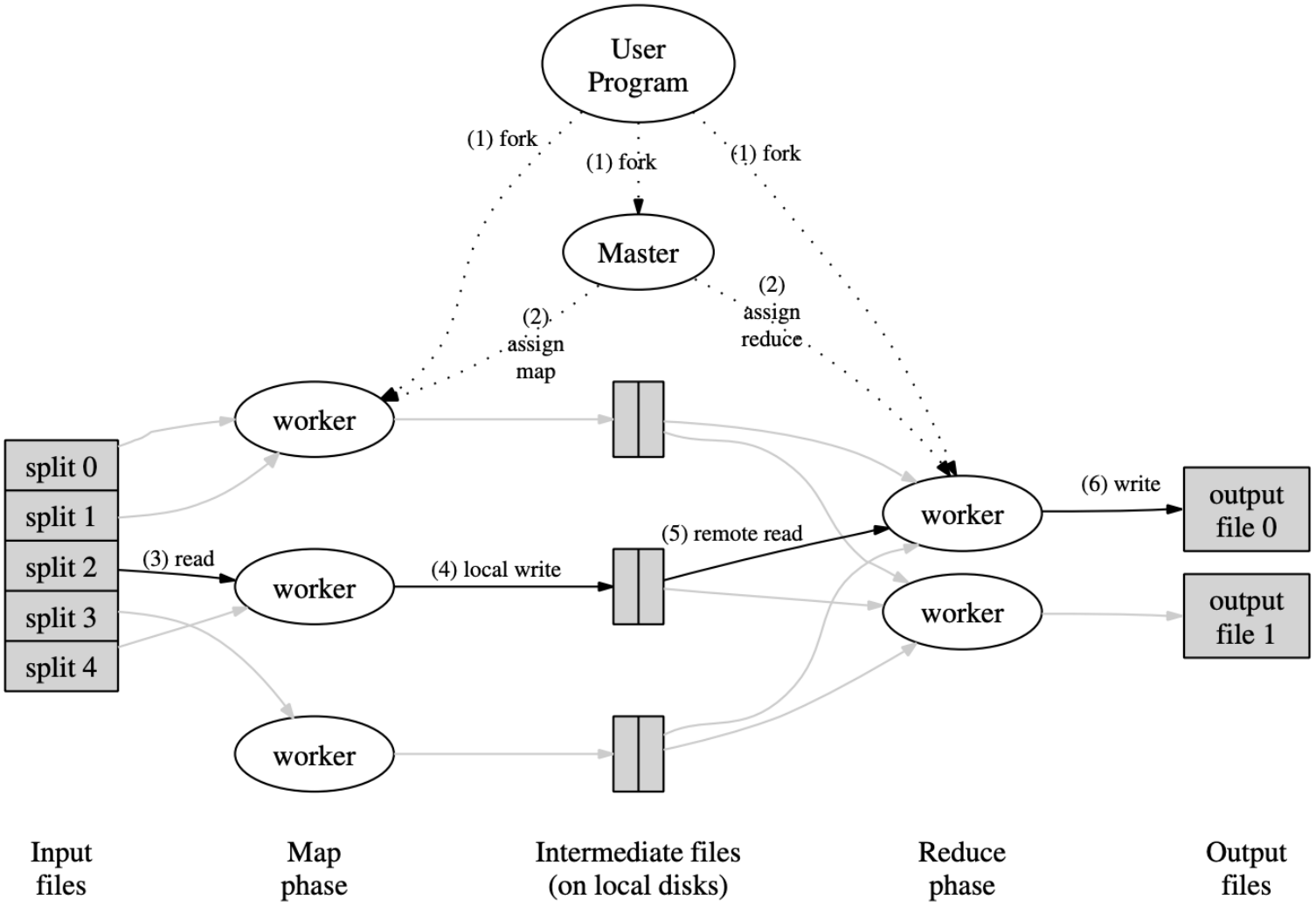
MapReduce

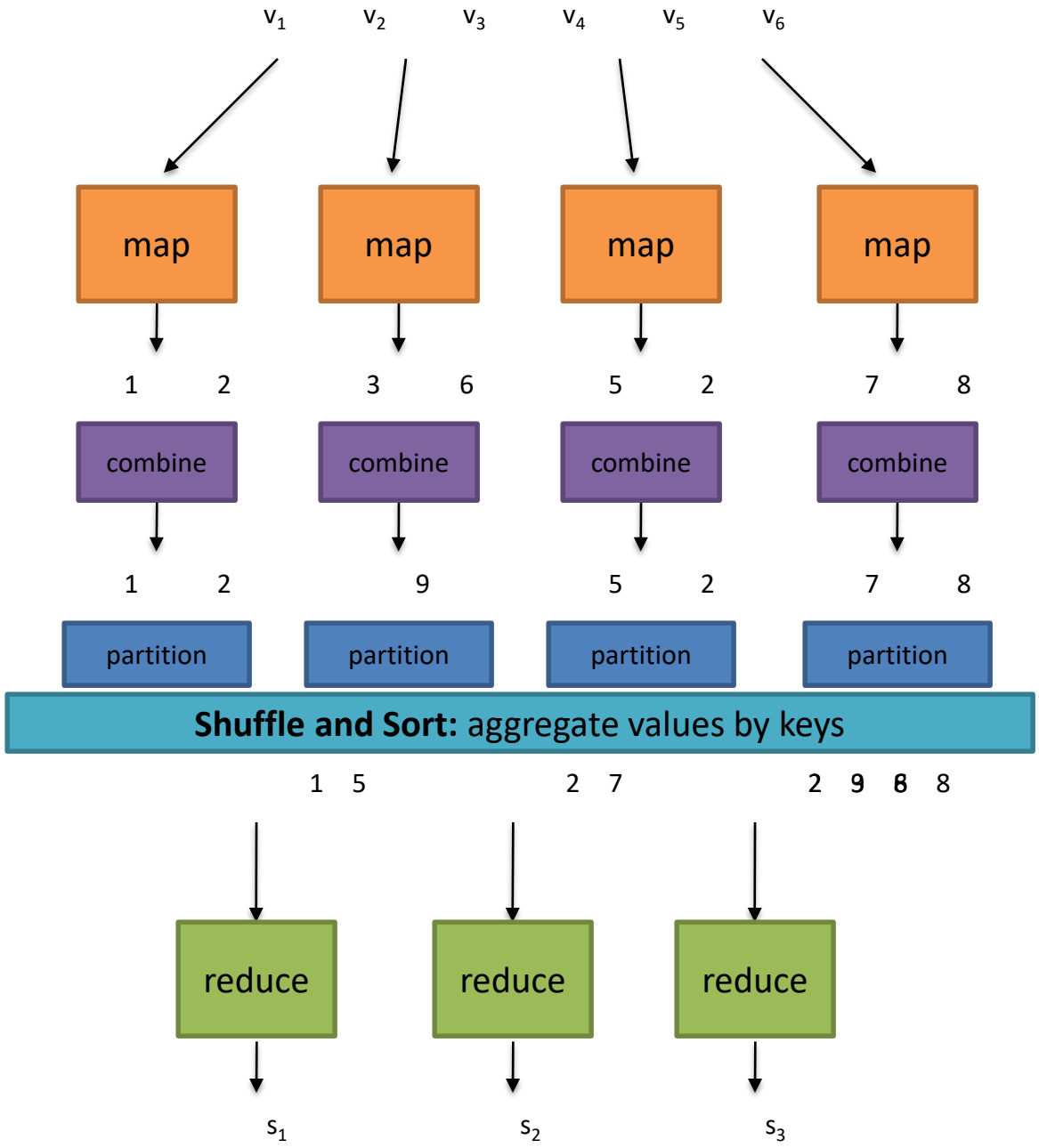
- Programmers specify two functions:
  map (k, v) → [(k', v')]
  reduce (k', [v']) → [(k', v')]
  – All values with the same key are reduced together
- The execution framework handles everything else…
- Not quite…usually, programmers also specify:
  partition (k', number of partitions) → partition for k'
  – Often a simple hash of the key, e.g., hash(k') mod n
  – Divides up key space for parallel reduce operations
  combine (k', [v']) → [(k', v'')]
  – Mini-reducers that run in memory after the map phase
  – Used as an optimization to reduce network traffic

MapReduce “Runtime”

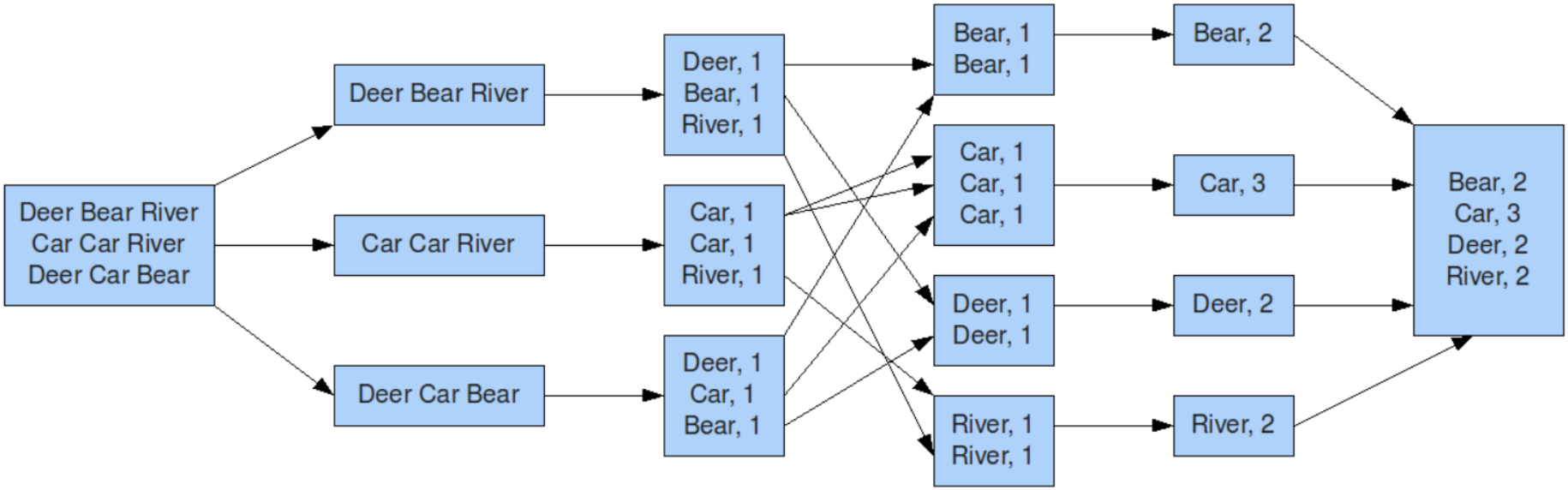
- Handles scheduling
  - Assigns workers to map and reduce tasks
- Handles “data distribution”
  - Moves processes to data
- Handles synchronization
  - Gathers, sorts, and shuffles intermediate data
- Handles errors and faults
  - Detects worker failures and restarts

MapReduce Original Paper 2004




v1 v2 v3 v4 v5 v6
map map map map
1 2 3 6 5 2 7 8
combine combine combine combine
1 2 9 5 2 7 8
partition partition partition partition
Shuffle and Sort: aggregate values by keys
1 5 2 7 2 9 8 8
reduce reduce reduce
s1 s2 s3

MapReduce Example
Word Count



MapReduce Implementations

- Google has a proprietary implementation in C++
  - Bindings in Java, Python
- Hadoop is an open-source implementation in Java
  - Development led by Yahoo, used in production
  - Now an Apache project
  - Rapidly expanding software ecosystem
- Lots of custom research implementations
  - For GPUs, cell processors, etc.

What is Hadoop?

- Hadoop provides a software framework for distributed storage and processing of big data using the MapReduce programming model.
- It is a collection of open-source software utilities that facilitate using a network of many computers to solve problems involving massive amounts of data and computation.
- All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework.

History of Hadoop

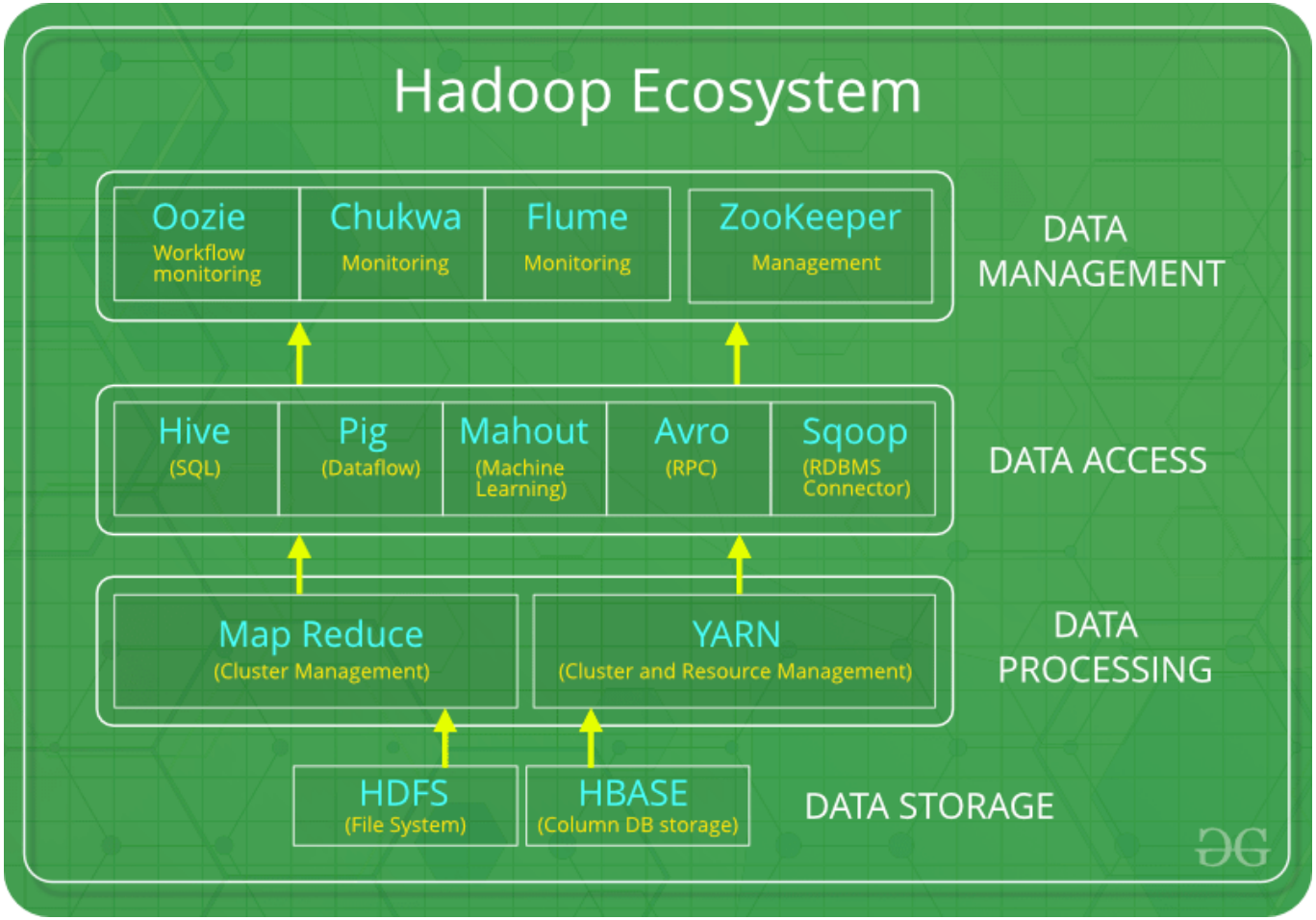
- **Dec 2004 –** Google GFS paper published
- **July 2005 –** Nutch uses MapReduce
- **Feb 2006 –** Becomes Lucene subproject
- **Apr 2007 –** Yahoo! on 1000-node cluster
- **Jan 2008 –** An Apache Top Level Project
- **Jul 2008 –** A 4000 node test cluster
- **Sept 2008 –** Hive becomes a Hadoop subproject
- **Feb 2009 –** The Yahoo! Search Webmap is a Hadoop application that runs on more than 10,000 core Linux cluster and produces data that is now used in every Yahoo! Web search query.
- **June 2009 –** On June 10, 2009, Yahoo! made available the source code to the version of Hadoop it runs in production.
- **In 2010** Facebook claimed that they have the largest Hadoop cluster in the world with 21 PB of storage. On July 27, 2011 they announced the data has grown to 30 PB.

Who uses Hadoop?

- Amazon
- Facebook
- Google
- IBM
- Joost
- Last.fm
- New York Times
- PowerSet
- Veoh
- Yahoo!

Components of Hadoop

There are four major elements of Hadoop i.e. HDFS, MapReduce, YARN, and Hadoop Common.



Components of Hadoop

- **HDFS:** Hadoop Distributed File System
- **YARN:** Yet Another Resource Negotiator
- **MapReduce:** Programming based Data Processing
- **Spark:** In-Memory data processing
- **PIG, HIVE:** Query based processing of data services
- **HBase:** NoSQL Database
- **Mahout, Spark MLLib:** Machine Learning algorithm libraries
- **Solar, Lucene:** Searching and Indexing
- **Zookeeper:** Managing cluster
- **Oozie:** Job Scheduling

Components of Hadoop

- **HDFS**
- HDFS is the primary or major component of Hadoop ecosystem and is responsible for storing large data sets of structured or unstructured data across various nodes and thereby maintaining the metadata in the form of log files.
- HDFS consists of two core components:
  - Name node
  - Data Node
- Name Node is the prime node which contains metadata (data about data) requiring comparatively fewer resources than the data nodes that stores the actual data. These data nodes are commodity hardware in the distributed environment. Undoubtedly, making Hadoop cost effective.
- HDFS maintains all the coordination between the clusters and hardware, thus working at the heart of the system.

Components of Hadoop

- **YARN**
- Yet Another Resource Negotiator, as the name implies, YARN is the one who helps to manage the resources across the clusters. In short, it performs scheduling and resource allocation for the Hadoop System.
- Consists of three major components i.e.
  - Resource Manager
  - Nodes Manager
  - Application Manager
- Resource manager has the privilege of allocating resources for the applications in a system whereas Node managers work on the allocation of resources such as CPU, memory, bandwidth per machine and later on acknowledges the resource manager. Application manager works as an interface between the resource manager and node manager and performs negotiations as per the requirement of the two.

Components of Hadoop

- **MapReduce**
- By making the use of distributed and parallel algorithms, MapReduce makes it possible to carry over the processing's logic and helps to write applications which transform big data sets into a manageable one.
- MapReduce makes the use of two functions i.e. Map() and Reduce() whose task is:
- Map() performs sorting and filtering of data and thereby organizing them in the form of group. Map generates a key-value pair based result which is later on processed by the Reduce() method.
- Reduce(), as the name suggests does the summarization by aggregating the mapped data. In simple, Reduce() takes the output generated by Map() as input and combines those tuples into smaller set of tuples

Components of Hadoop

- **PIG**
- Pig was basically developed by Yahoo which works on a pig Latin language, which is Query based language similar to SQL.
- It is a platform for structuring the data flow, processing and analyzing huge data sets.
- Pig does the work of executing commands and in the background, all the activities of MapReduce are taken care of. After the processing, pig stores the result in HDFS.
- Pig Latin language is specially designed for this framework which runs on Pig Runtime. Just the way Java runs on the JVM.
- Pig helps to achieve ease of programming and optimization and hence is a major segment of the Hadoop Ecosystem.

Components of Hadoop

- **HIVE**
- With the help of SQL methodology and interface, HIVE performs reading and writing of large data sets. However, its query language is called as HQL (Hive Query Language).
- It is highly scalable as it allows real-time processing and batch processing both. Also, all the SQL datatypes are supported by Hive thus, making the query processing easier.
- Similar to the Query Processing frameworks, HIVE too comes with two components: JDBC Drivers and HIVE Command Line.
- JDBC, along with ODBC drivers work on establishing the data storage permissions and connection whereas HIVE Command line helps in the processing of queries.

Components of Hadoop

- **Mahout**
- Mahout, allows Machine Learnability to a system or application. Machine Learning, as the name suggests helps the system to develop itself based on some patterns, user/environmental interaction or on the basis of algorithms.
- It provides various libraries or functionalities such as collaborative filtering, clustering, and classification which are nothing but concepts of Machine learning. It allows invoking algorithms as per our need with the help of its own libraries.

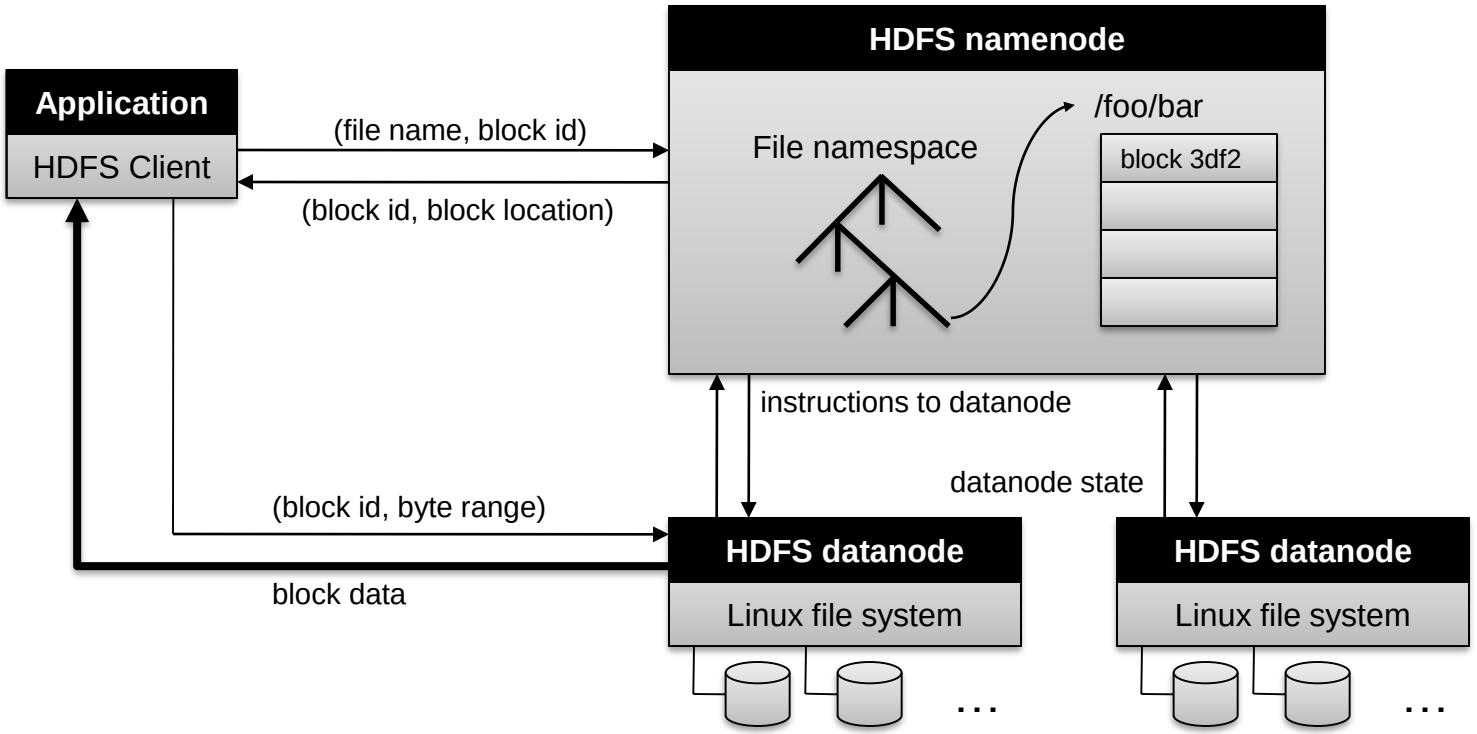
Components of Hadoop

- **Apache Spark**
- It's a platform that handles all the process consumptive tasks like batch processing, interactive or iterative real-time processing, graph conversions, and visualization, etc.
- It consumes in memory resources hence, thus being faster than the prior in terms of optimization.
- Spark is best suited for real-time data whereas Hadoop is best suited for structured data or batch processing, hence both are used in most of the companies interchangeably.
- **Apache HBase**
- It's a NoSQL database which supports all kinds of data and thus capable of handling anything of Hadoop Database. It provides capabilities of Google's BigTable, thus able to work on Big Data sets effectively.
- At times where we need to search or retrieve the occurrences of something small in a huge database, the request must be processed within a short quick span of time. At such times, HBase comes handy as it gives us a tolerant way of storing limited data.

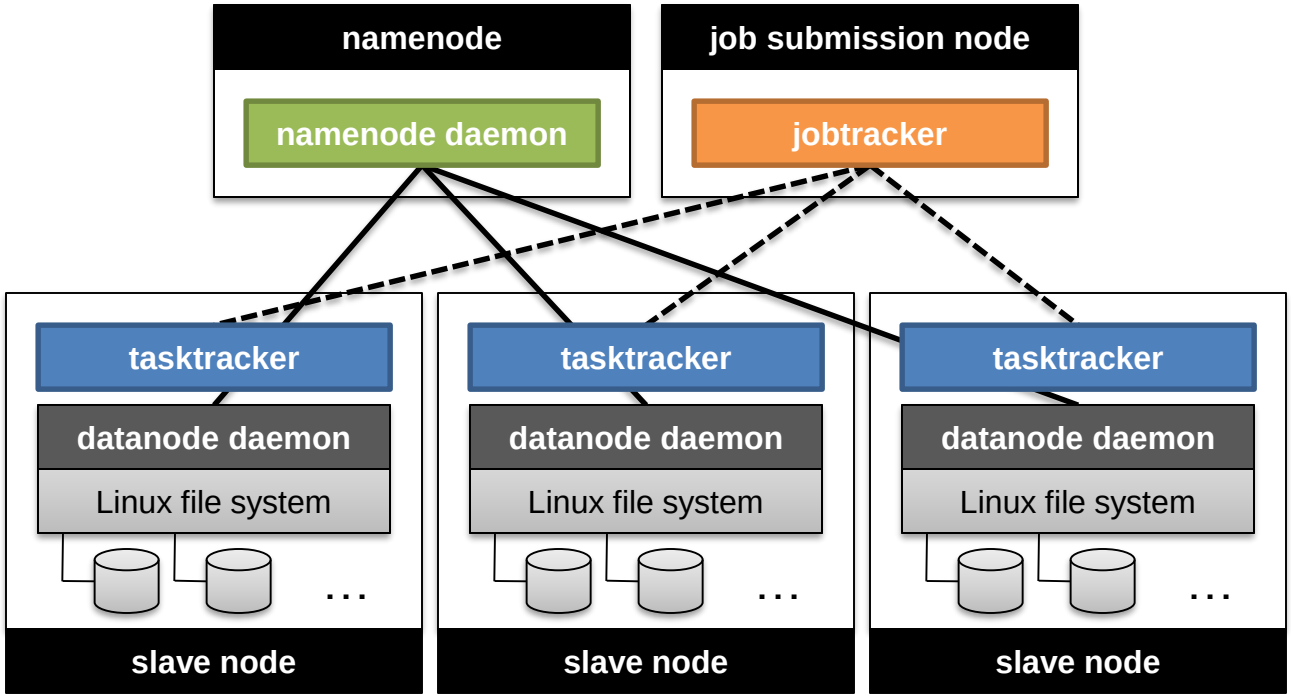
Components of Hadoop

- **Solr, Lucene:**
  - These are the two services that perform the task of searching and indexing with the help of some java libraries, especially Lucene is based on Java which allows spell check mechanism, as well. However, Lucene is driven by Solr.
- **Zookeeper:**
  - There was a huge issue of management of coordination and synchronization among the resources or the components of Hadoop which resulted in inconsistency, often. Zookeeper overcame all the problems by performing synchronization, inter-component based communication, grouping, and maintenance.
- **Oozie**
  - Oozie simply performs the task of a scheduler, thus scheduling jobs and binding them together as a single unit. There is two kinds of jobs .i.e Oozie workflow and Oozie coordinator jobs. Oozie workflow is the jobs that need to be executed in a sequentially ordered manner whereas Oozie Coordinator jobs are those that are triggered when some data or external stimulus is given to it.

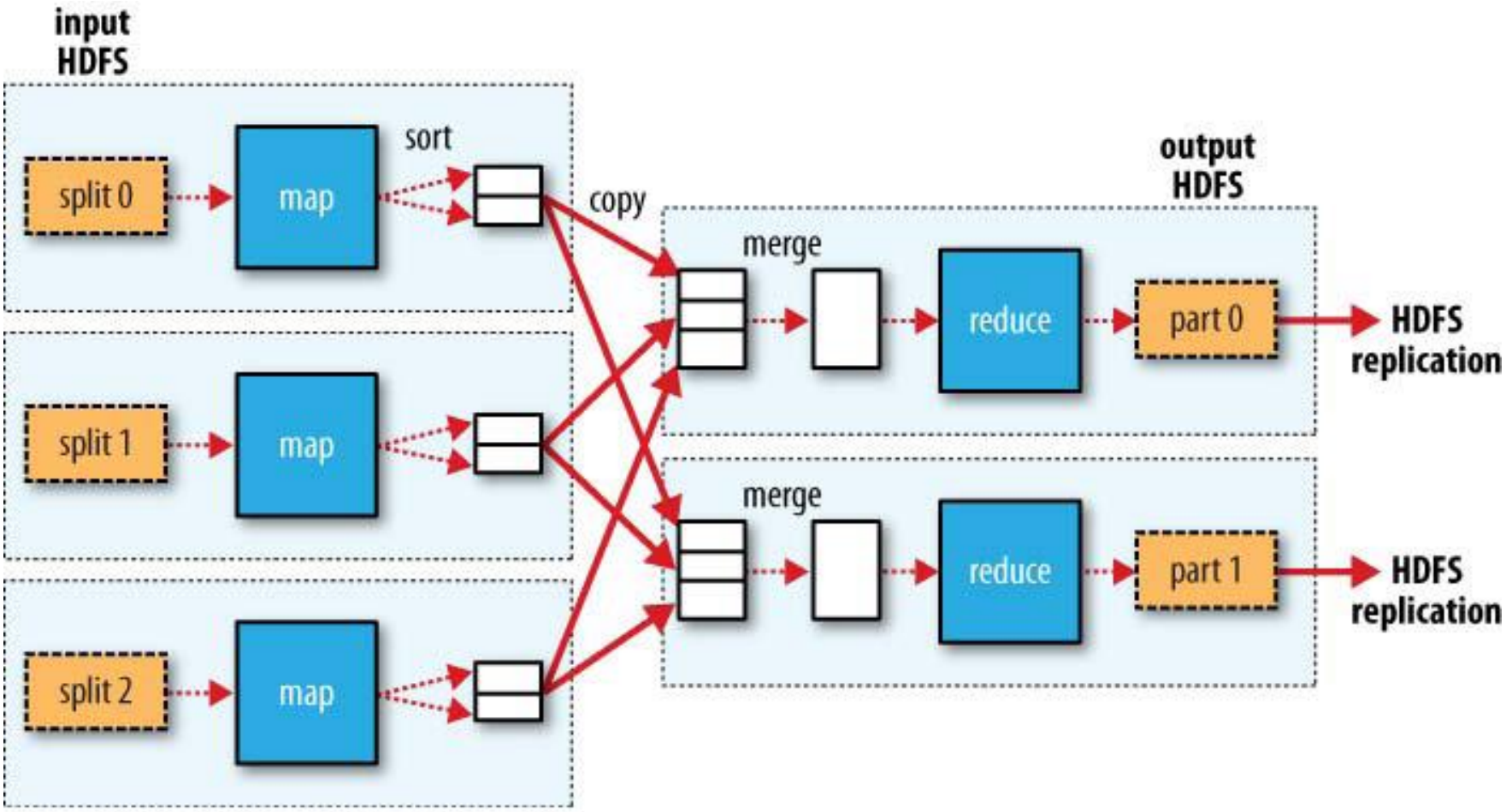
Components of Hadoop

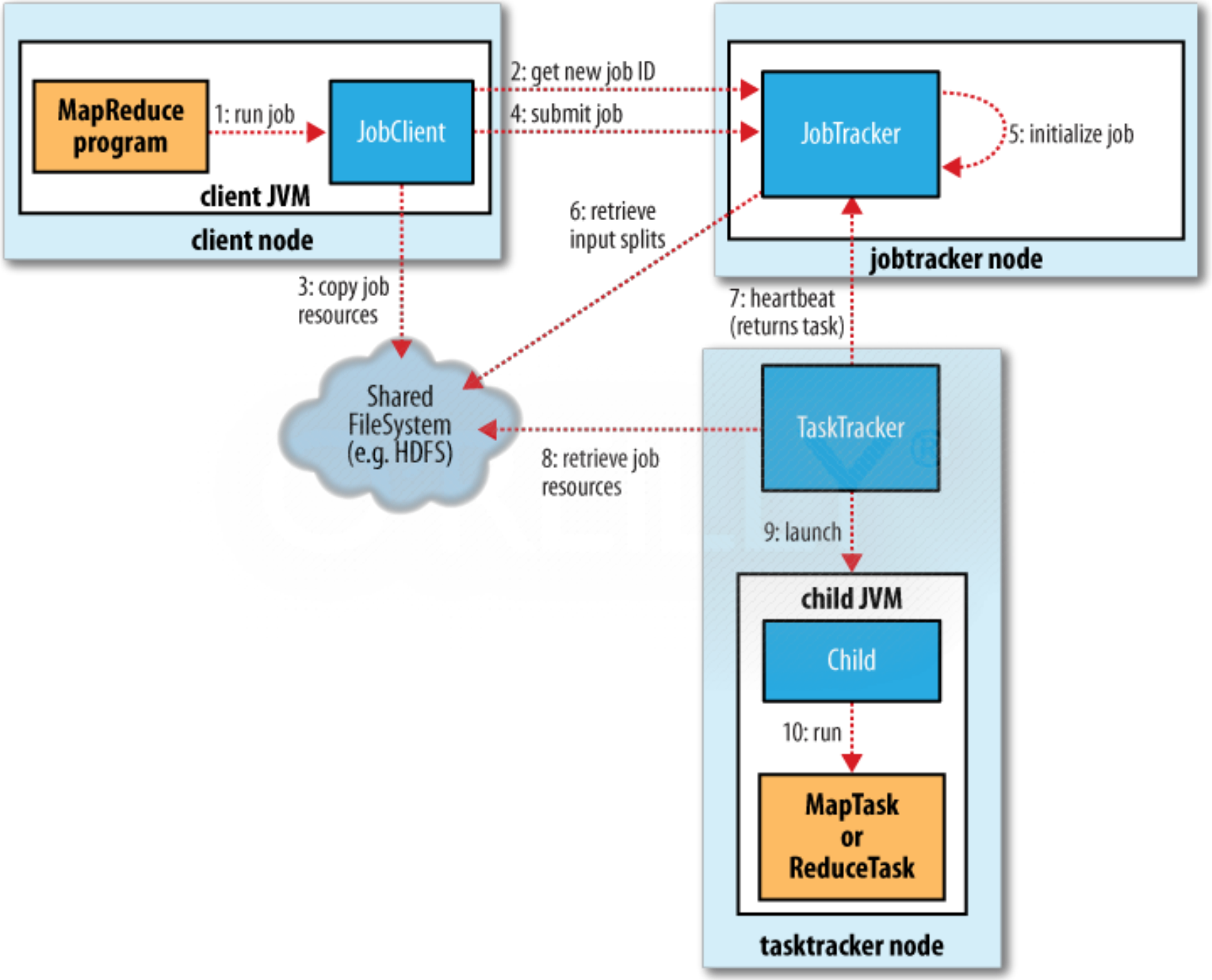


Components of Hadoop



Components of Hadoop




MapReduce program
1: run job
JobClient
client JVM
client node
2: get new job ID
4: submit job
JobTracker
5: initialize job
jobtracker node
6: retrieve input splits
3: copy job resources
7: heartbeat (returns task)
Shared FileSystem (e.g. HDFS)
8: retrieve job resources
TaskTracker
9: launch
child JVM
Child
10: run
MapTask or ReduceTask
tasktracker node

Features of Hadoop

- Hadoop is Open Source
  - Hadoop is an open-source project, which means its source code is available free of cost for inspection, modification, and analyses that allows enterprises to modify the code as per their requirements.
- Hadoop cluster is Highly Scalable
  - Hadoop cluster is scalable means we can add any number of nodes (horizontal scalable) or increase the hardware capacity of nodes (vertical scalable) to achieve high computation power. This provides horizontal as well as vertical scalability to the Hadoop framework.

Features of Hadoop

- Hadoop provides Fault Tolerance
  - Fault tolerance is the most important feature of Hadoop. HDFS in Hadoop 2 uses a replication mechanism to provide fault tolerance.
  - It creates a replica of each block on the different machines depending on the replication factor (by default, it is 3). So if any machine in a cluster goes down, data can be accessed from the other machines containing a replica of the same data.
  - Hadoop 3 has replaced this replication mechanism by erasure coding. Erasure coding provides the same level of fault tolerance with less space. With Erasure coding, the storage overhead is not more than 50%.

Features of Hadoop

- Hadoop provides High Availability
  - This feature of Hadoop ensures the high availability of the data, even in unfavorable conditions.
- Hadoop is very Cost-Effective
  - Since the Hadoop cluster consists of nodes of commodity hardware that are inexpensive, thus provides a cost-effective solution for storing and processing big data. Being an open-source product, Hadoop doesn't need any license.
- Hadoop is Faster in Data Processing
  - Hadoop stores data in a distributed fashion, which allows data to be processed distributedly on a cluster of nodes. Thus it provides lightning-fast processing capability to the Hadoop framework.
- Hadoop is based on Data Locality concept
  - Hadoop is popularly known for its data locality feature means moving computation logic to the data, rather than moving data to the computation logic. This features of Hadoop reduces the bandwidth utilization in a system.

Features of Hadoop

- Hadoop is Easy to use
  - Hadoop is easy to use as the clients don't have to worry about distributing computing. The processing is handled by the framework itself.
- Hadoop ensures Data Reliability
  - In Hadoop due to the replication of data in the cluster, data is stored reliably on the cluster machines despite machine failures.
  - The framework itself provides a mechanism to ensure data reliability by Block Scanner, Volume Scanner, Disk Checker, and Directory Scanner. If your machine goes down or data gets corrupted, then also your data is stored reliably in the cluster and is accessible from the other machine containing a copy of data.

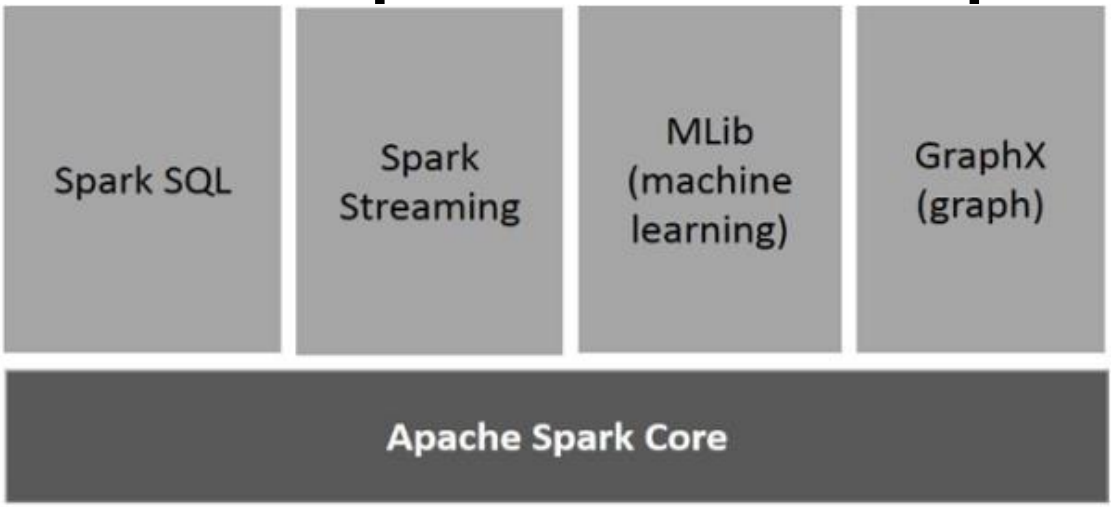
What is Spark?

- Spark is a fast cluster computing technology, designed for fast computation.
- It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations.
- It includes interactive queries and stream processing.
- The main feature of Spark is its in-memory cluster computing that increases the processing speed of an application.

History of Spark?

- Spark is one of Hadoop's sub projects that was developed in 2009 in UC Berkeley's AMPLab.
- It was Open Sourced in 2010 under a BSD license.
- It was donated to Apache software foundation in 2013.
- Apache Spark has become a top level Apache project since Feb-2014.

The Components of Spark



- Apache Spark Core
  - Spark Core is the underlying general execution engine for spark platform that all other functionality is built upon.
  - It provides In-Memory computing and referencing datasets in external storage systems.

The Components of Spark

- Spark SQL
  - Spark SQL is a component on top of Spark Core that introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data.
- Spark Streaming
  - Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of data.

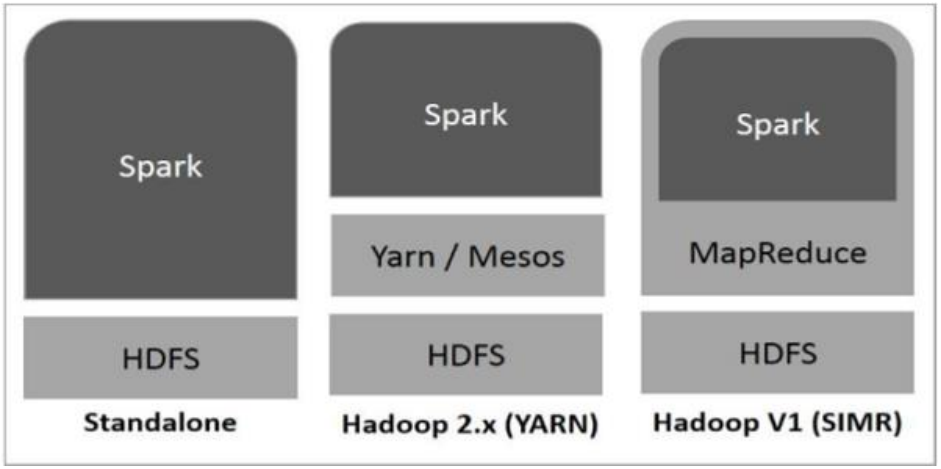
The Components of Spark

- MLlib (Machine Learning Library)
  - MLlib is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture.
  - It is, according to benchmarks, done by the MLlib developers against the Alternating Least Squares (ALS) implementations. Spark MLlib is nine times as fast as the Hadoop disk-based version of Apache Mahout (before Mahout gained a Spark interface).
- GraphX
  - GraphX is a distributed graph-processing framework on top of Spark. It provides an API for expressing graph computation that can model the user-defined graphs by using Pregel abstraction API. It also provides an optimized runtime for this abstraction.

Features of Apache Spark

- Speed
  - Spark helps to run an application in Hadoop cluster, up to 100 times faster in memory, and 10 times faster when running on disk.
  - This is possible by reducing number of read/write operations to disk. It stores the intermediate processing data in memory.
- Supports multiple languages
  - Spark provides built-in APIs in Java, Scala, or Python. Therefore, you can write applications in different languages. Spark comes up with 80 high-level operators for interactive querying.
- Advanced Analytics
  - Spark not only supports ‘Map’ and ‘reduce’. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.

Spark Built on Hadoop



- Standalone:
  - Spark Standalone deployment means Spark occupies the place on top of HDFS (Hadoop Distributed File System) and space is allocated for HDFS, explicitly.
  - Here, Spark and MapReduce will run side by side to cover all spark jobs on cluster.

Spark Built on Hadoop

- Hadoop Yarn:
  - Hadoop Yarn deployment means, simply, spark runs on Yarn without any pre-installation or root access required. It helps to integrate Spark into Hadoop ecosystem or Hadoop stack. It allows other components to run on top of stack.
- Spark in MapReduce (SIMR):
  - Spark in MapReduce is used to launch spark job in addition to standalone deployment. With SIMR, user can start Spark and uses its shell without any administrative access.

Spark vs Hadoop

- With multiple big data frameworks available, choosing the right one is a challenge.
- The key difference:
  - Spark can do it in-memory.
  - Hadoop MapReduce has to read from and write to a disk.
- Spark may be up to 100 times faster.
- Hadoop MapReduce is able to work with far larger data sets than Spark.

Spark vs Hadoop

- Advantages of Hadoop Mapreduce
- Linear processing of huge data sets:
  - Hadoop MapReduce allows parallel processing of huge amounts of data.
  - In case the resulting dataset is larger than available RAM, Hadoop MapReduce may outperform Spark.
- Economical solution, if no immediate results are expected:
  - MapReduce is a good solution if the speed of processing is not critical. For instance, if data processing can be done during night hours, it makes sense to consider using Hadoop MapReduce.

Spark vs Hadoop

- Advantages of Spark
- Fast data processing:
  - In-memory processing makes Spark faster than Hadoop MapReduce – up to 100 times for data in RAM and up to 10 times for data in storage.
- Iterative processing:
  - If the task is to process data again and again – Spark defeats Hadoop MapReduce. Spark's Resilient Distributed Datasets (RDDs) enable multiple map operations in memory, while Hadoop MapReduce has to write interim results to a disk.
- Near real-time processing:
  - If a business needs immediate insights, then they should opt for Spark and its in-memory processing.

Spark vs Hadoop

- Graph processing:
  - Spark's computational model is good for iterative computations that are typical in graph processing. And Apache Spark has GraphX – an API for graph computation.
- Machine learning:
  - Spark has MLlib – a built-in machine learning library, while Hadoop needs a third-party to provide it. MLlib has out-of-the-box algorithms that also run in memory.
- Joining datasets:
  - Due to its speed, Spark can create all combinations faster, though Hadoop may be better if joining of very large data sets that requires a lot of shuffling and sorting is needed.

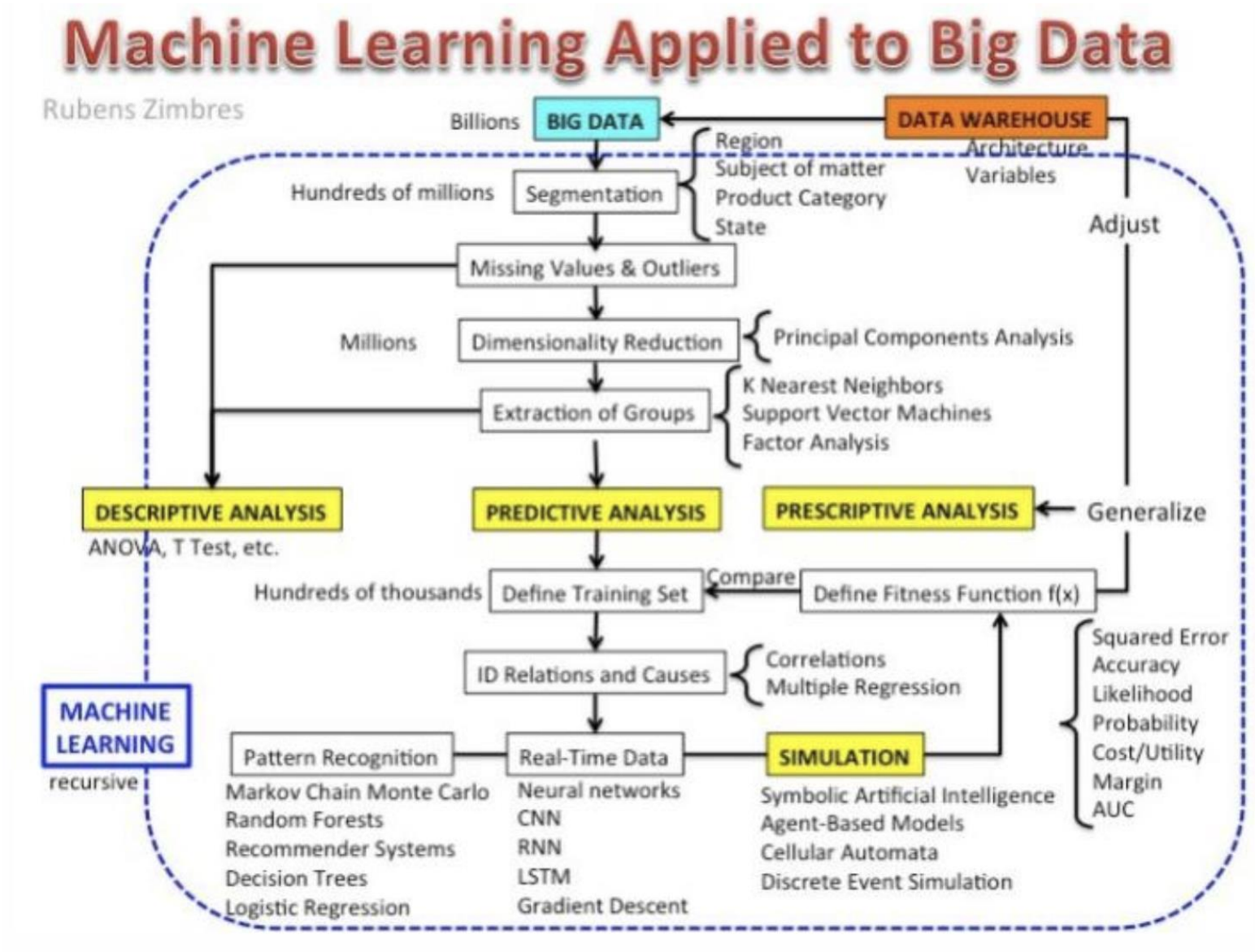
Application of Hadoop & Spark

- **Customer segmentation:**
  - Analyzing customer behavior and identifying segments of customers that demonstrate similar behavior patterns will help businesses to understand customer preferences and create a unique customer experience.
- **Risk management**:
  - Forecasting different possible scenarios can help managers to make right decisions by choosing non-risky options.
- **Real-time fraud detection:**
  - After the system is trained on historical data with the help of machine-learning algorithms, it can use these findings to identify or predict an anomaly in real time that may signal of a possible fraud.
- **Industrial big data analysis:**
  - It's also about detecting and predicting anomalies, but in this case, these anomalies are related to machinery breakdowns. A properly configured system collects the data from sensors to detect pre-failure conditions.

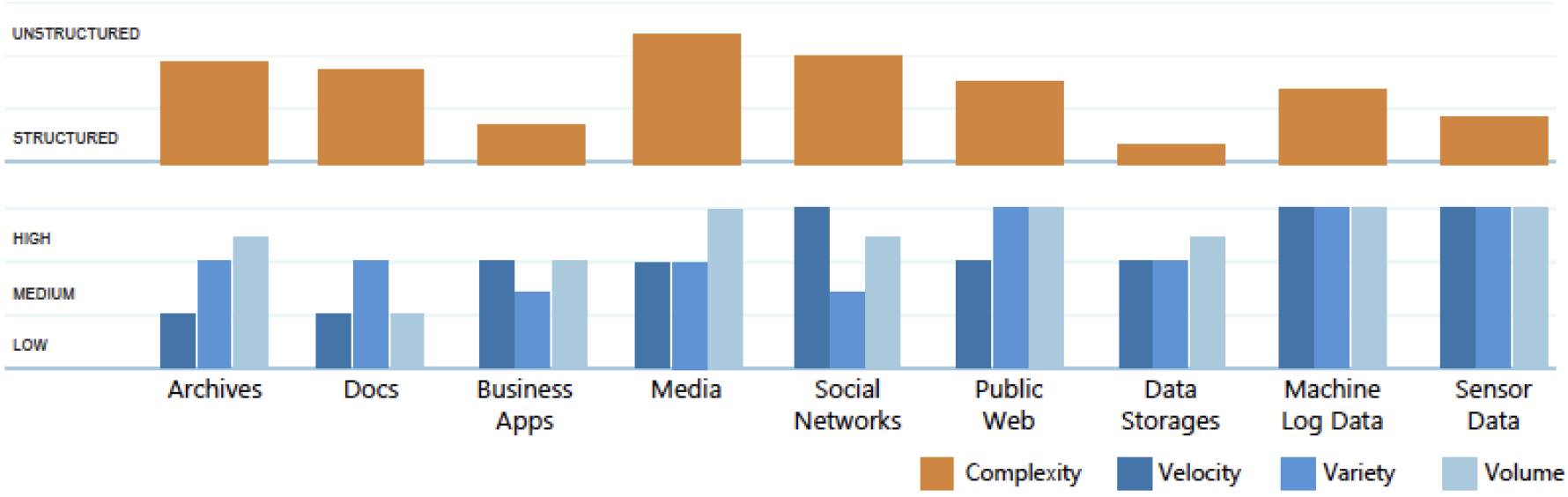
Which one to Choose?

- Linear processing of huge datasets is the advantage of Hadoop MapReduce, while Spark delivers fast performance, iterative processing, real-time analytics, graph processing, machine learning and more.
- In many cases Spark may outperform Hadoop MapReduce.
- Spark is fully compatible with the Hadoop eco-system and works smoothly with Hadoop Distributed File System, Apache Hive, etc.

Big Data & Machine Learning



Big Data & Machine Learning Challenges



Archives
Scanned documents, statements, medical records, e-mails etc..

Docs
XLS, PDF, CSV, HTML, JSON etc.

Business Apps
CRM, ERP systems, HR, project management etc.

Media
Images, video, audio etc.

Social Networks
Twitter, Facebook, Google+, LinkedIn etc.

Public Web
Wikipedia, news, weather, public finance etc

Data Storages
RDBMS, NoSQL, Hadoop, file systems etc.

Machine Log Data
Application logs, event logs, server data, CDRs, clickstream data etc.

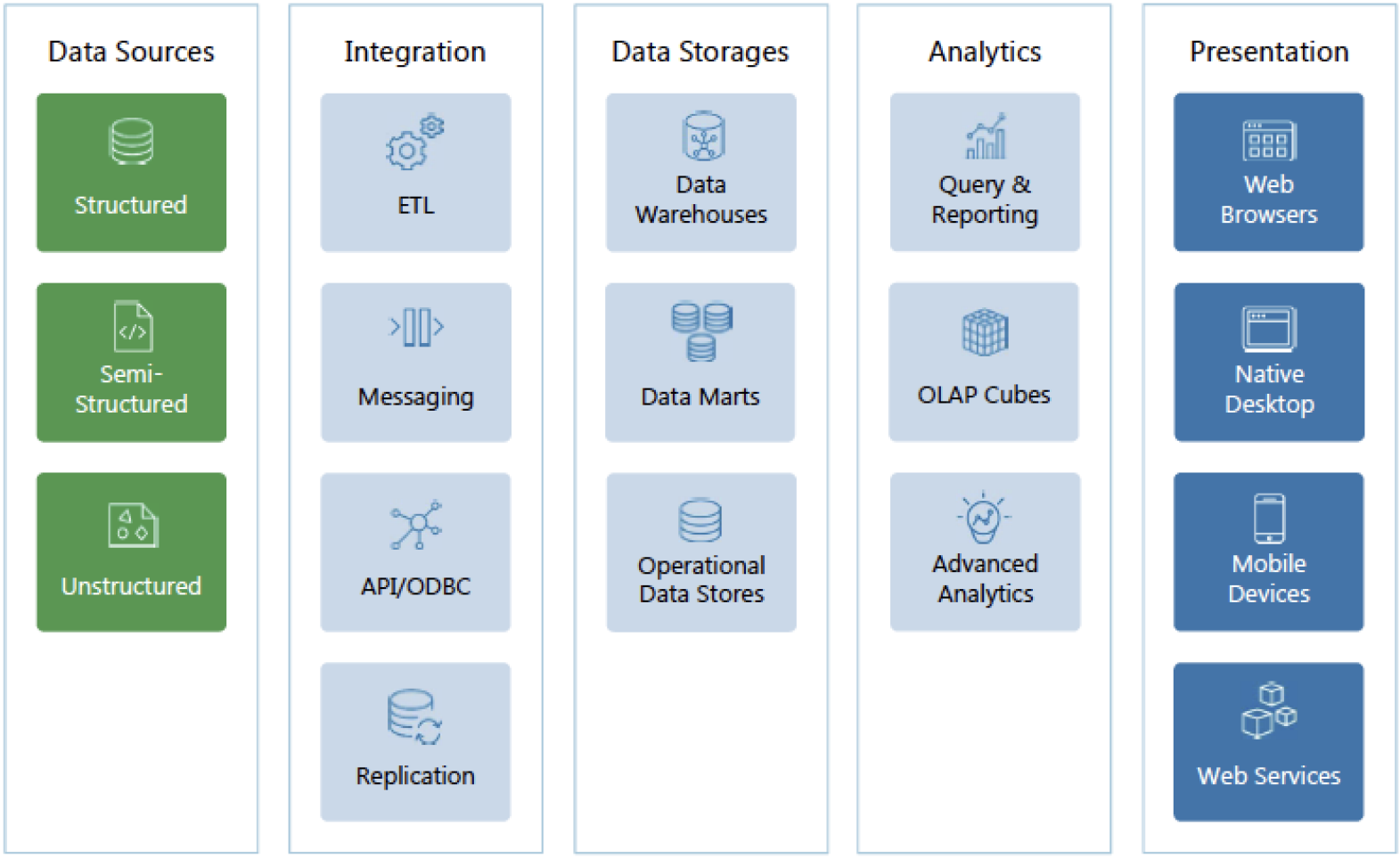
Sensor Data
Smart electric meters, medical devices, car sensors, road cameras etc.

Big Data & Machine Learning Challenges

	Traditional Analytics (BI)	vs	Big Data Analytics
Focus on	• Descriptive analytics • Diagnosis analytics		• **Predictive analytics** • **Data Science**
Data Sets	• Limited data sets • Cleansed data • Simple models		• Large scale data sets • More types of data • Raw data • Complex data models
Supports	**Causation:** what happened, and why?		**Correlation**: new insight More accurate answers

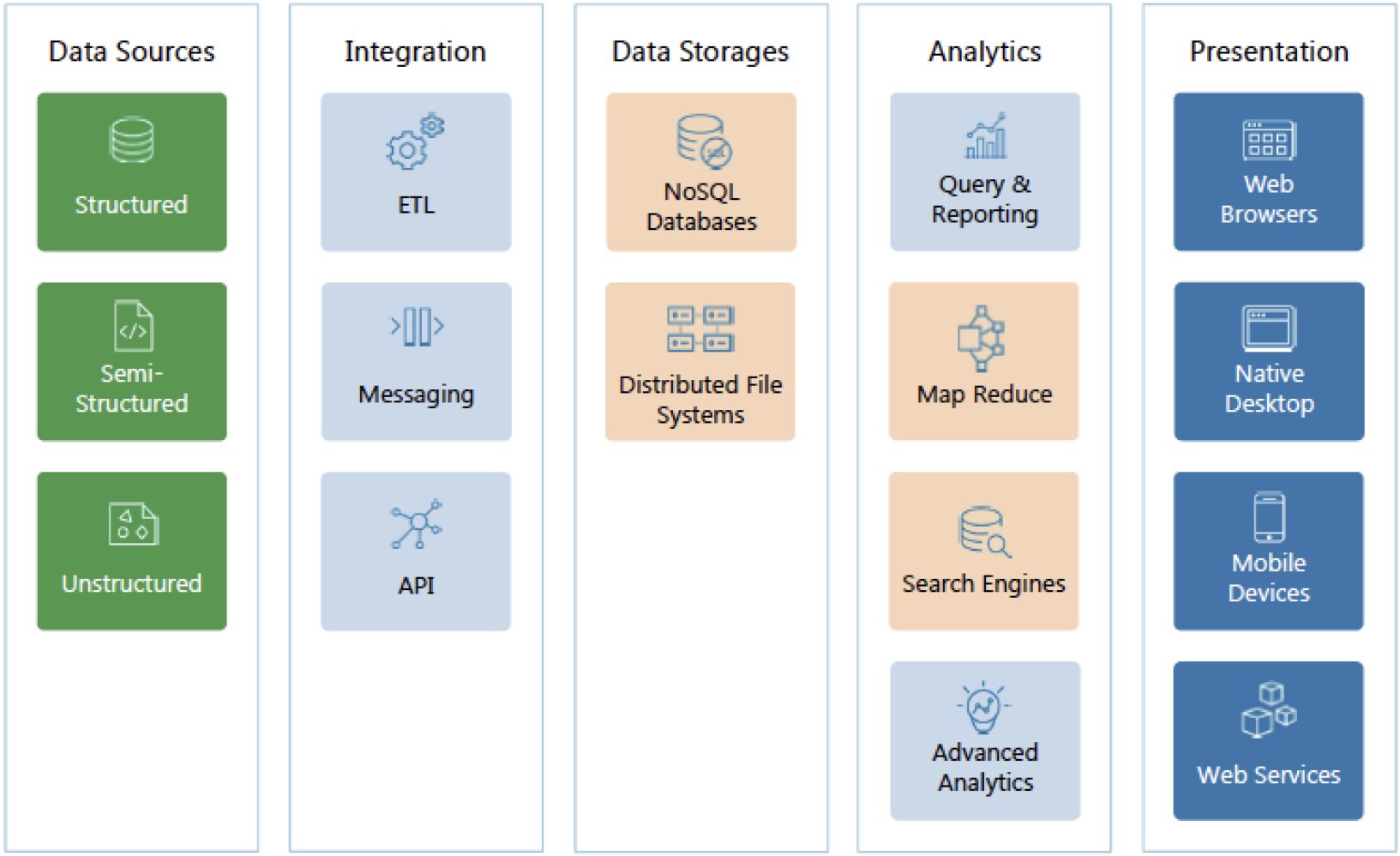
Big Data & Machine Learning Challenges

Relational Reference Architecture



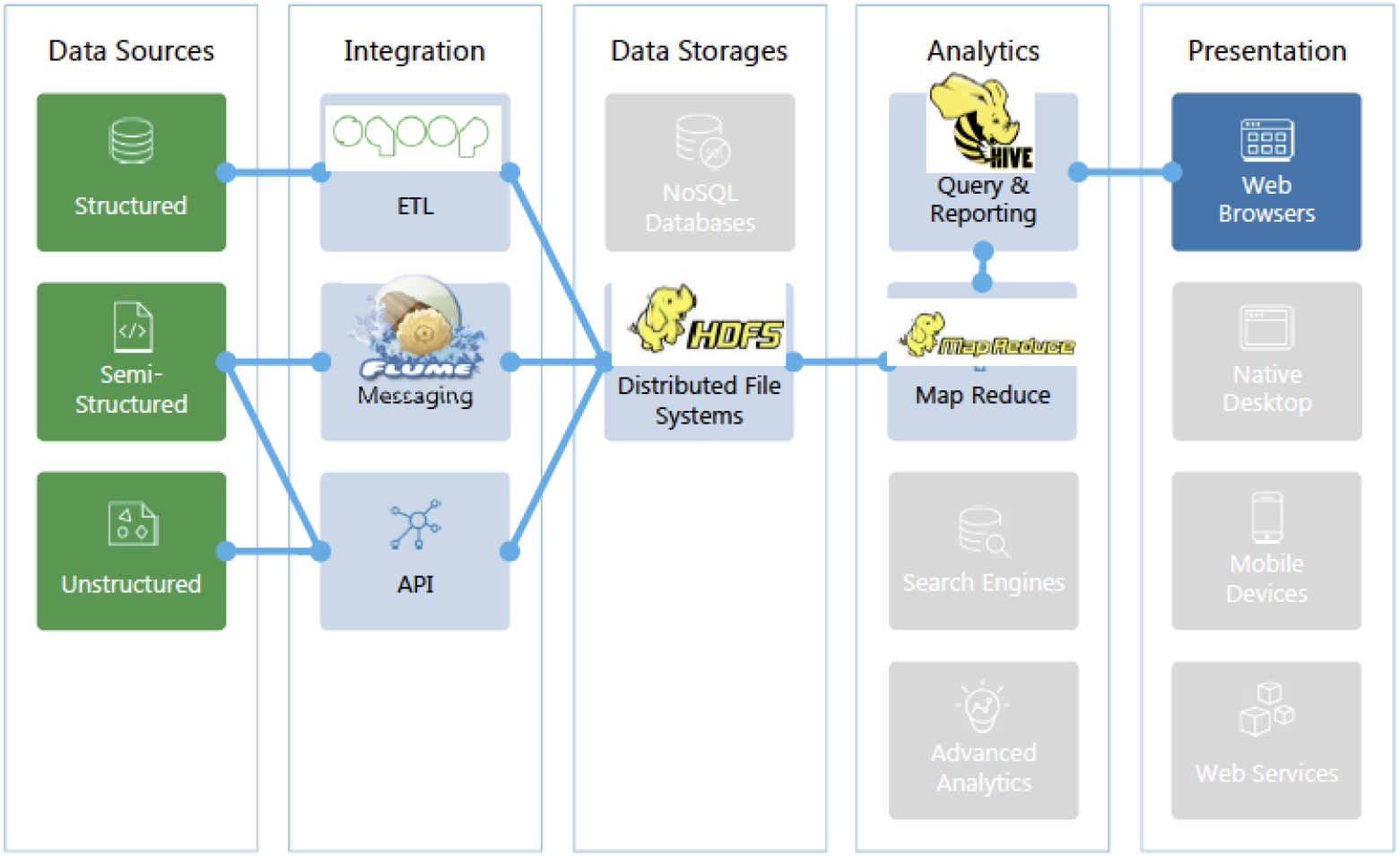
Big Data & Machine Learning Challenges

Non-Relational Reference Architecture



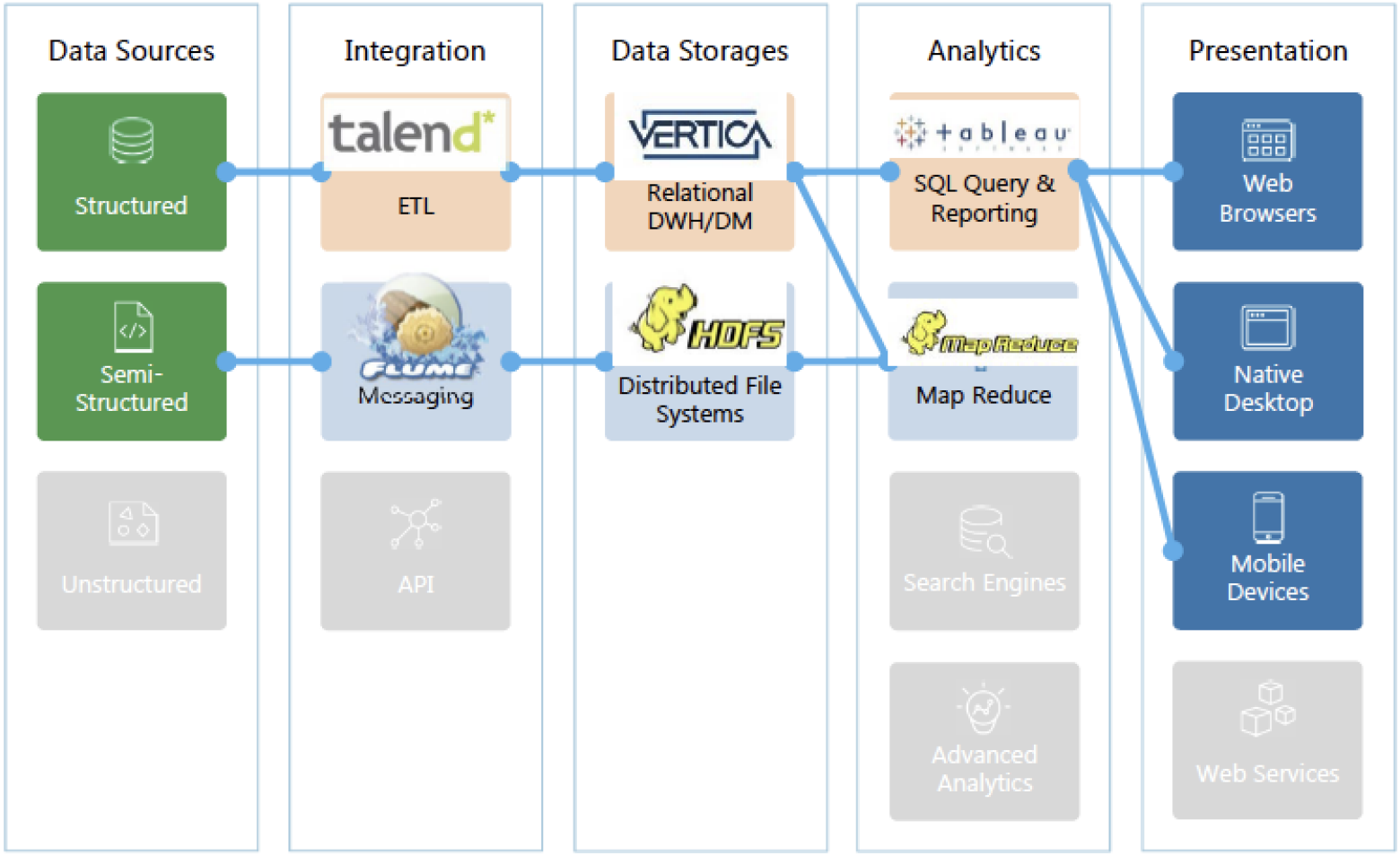
Big Data & Machine Learning Challenges

Data Discovery: Non-Relational Architecture

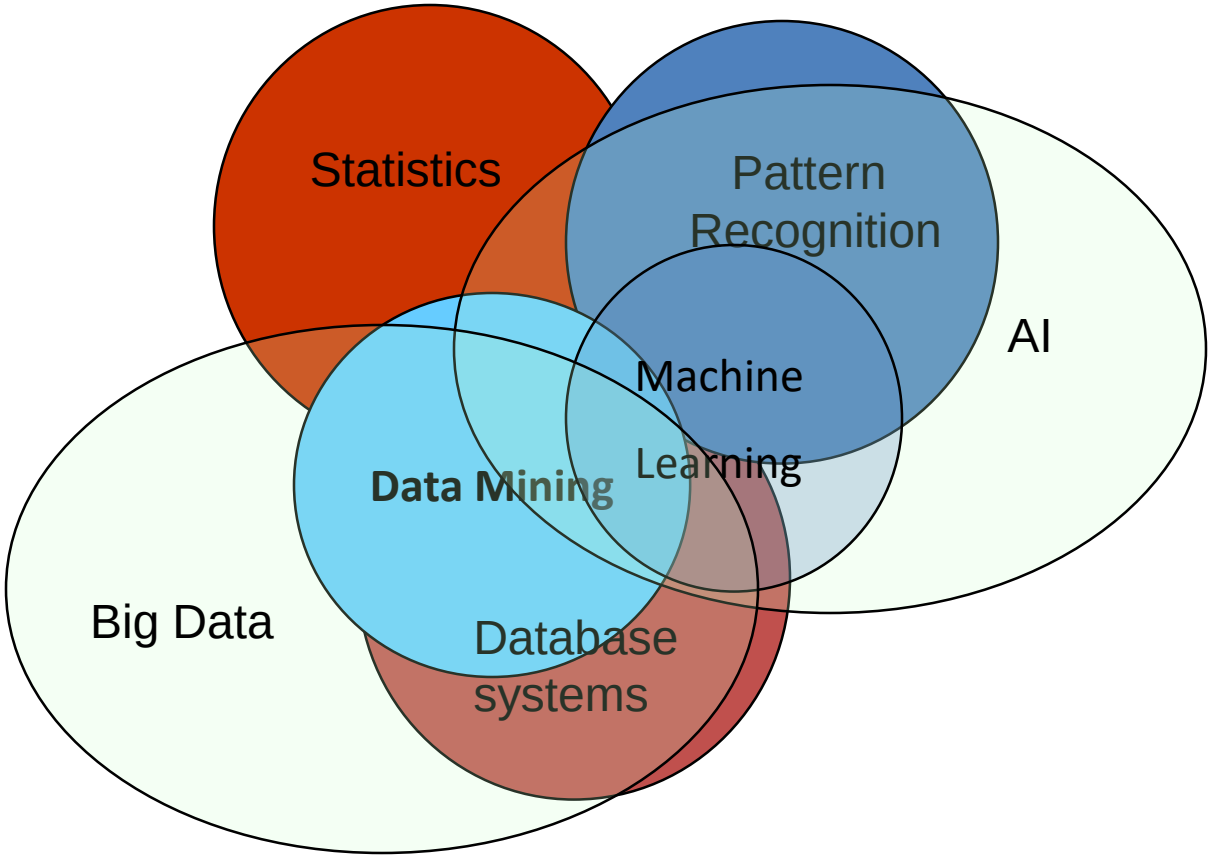


Big Data & Machine Learning Challenges

Business Reporting: Hybrid Architecture



Big Data & Machine Learning Challenges



Machine Learning with MapReduce
A case Study
k-means

Machine Learning with MapReduce

Algorithm 16.1: K-means Algorithm

K-means $(\boldsymbol{D}, k, \epsilon)$:

$t = 0$

Randomly initialize k centroids: $\boldsymbol{\mu}_1^t, \boldsymbol{\mu}_2^t, \ldots, \boldsymbol{\mu}_k^t$

repeat

  $t = t + 1$

  `// Cluster Assignment Step`

  foreach $\boldsymbol{x}_j \in \boldsymbol{D}$ **do**

    $j^* = \arg\min_i \{\|\boldsymbol{x}_j - \boldsymbol{\mu}_i^t\|^2\}$ `// Assign` $\boldsymbol{x}_j$ `to closest centroid`

    $C_{j^*} = C_{j^*} \cup \{\boldsymbol{x}_j\}$

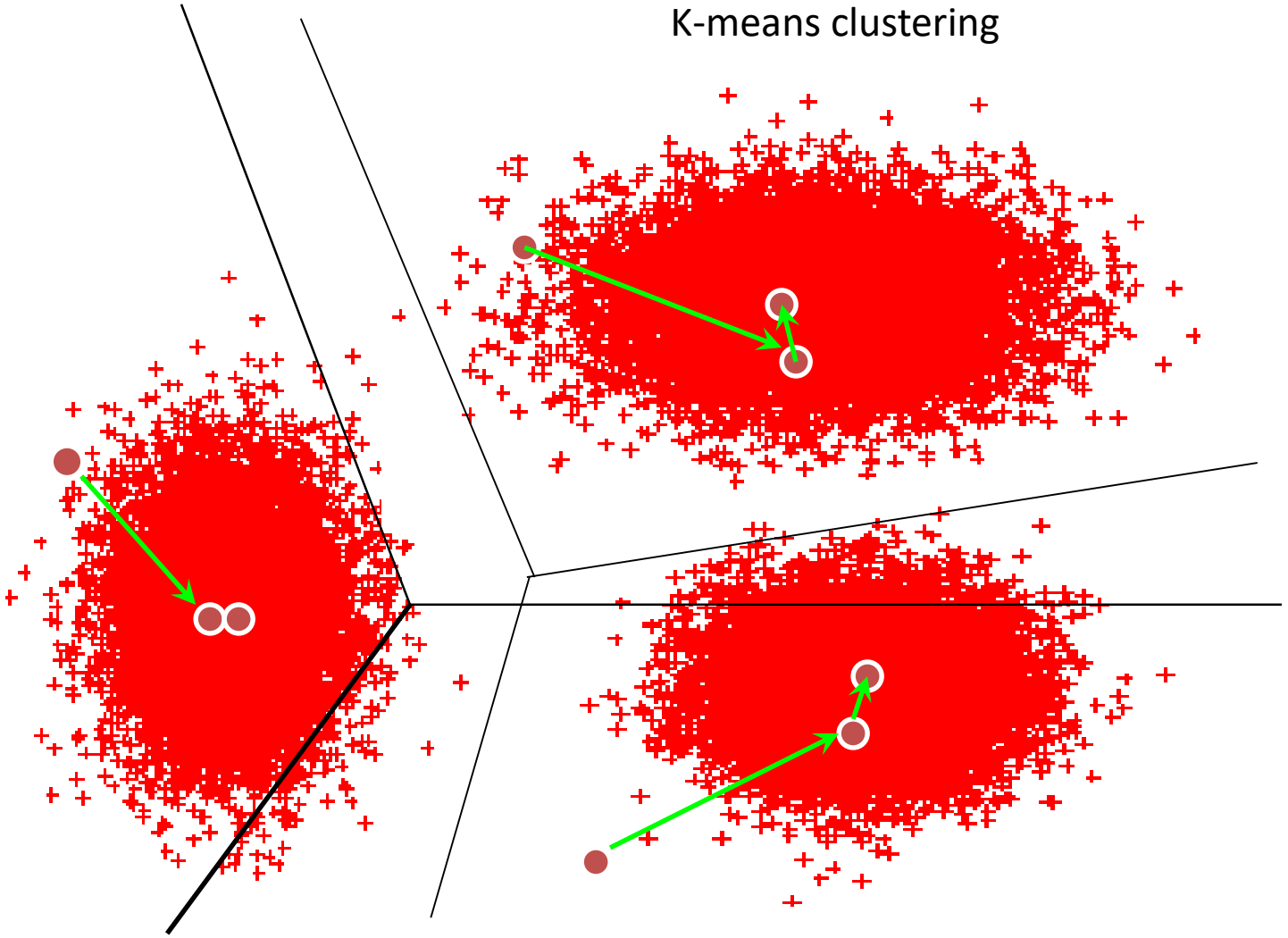
  `// Centroid Update Step`

  foreach $i = 1$ *to* k **do**

    $\boldsymbol{\mu}_i^t = \frac{1}{|C_i|} \sum_{\boldsymbol{x}_j \in C_i} \boldsymbol{x}_j$

until $\sum_{i=1}^k \|\boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1}\|^2 \leq \epsilon$

Machine Learning with MapReduce



How to MapReduce K-Means?

- Given *K*, assign the first *K* random points to be the initial cluster centers
- Assign subsequent points to the closest cluster using the supplied distance measure
- Compute the centroid of each cluster and iterate the previous step until the cluster centers converge within *delta*
- Run a final pass over the points to cluster them for output

K-Means Map/Reduce Design

- Driver
  - Runs multiple iteration jobs using mapper+combiner+reducer
  - Runs final clustering job using only mapper
- Mapper
  - Configure: Single file containing encoded Clusters
  - Input: File split containing encoded Vectors
  - Output: Vectors keyed by nearest cluster
- Combiner
  - Input: Vectors keyed by nearest cluster
  - Output: Cluster centroid vectors keyed by “cluster”
- Reducer (singleton)
  - Input: Cluster centroid vectors
  - Output: Single file containing Vectors keyed by cluster

Big Data & Machine Learning Challenges

- Large volumes of data are now available
- Platforms now exist to run computations over large datasets (Hadoop, HBase)
- Sophisticated analytics are needed to turn data into information people can use
- Active research community and proprietary implementations of “machine learning” algorithms
- The world needs scalable implementations of ML on distributed platforms

History of Mahout

- Summer 2007
  - Developers needed scalable ML
  - Mailing list formed
- Community formed
  - Apache contributors
  - Academia & industry
  - Lots of initial interest
- Project formed under Apache Lucene
  - January 25, 2008

ML Current Code Base

- Matrix & Vector library
  - Memory resident sparse & dense implementations
- Clustering
  - Canopy
  - K-Means
  - Mean Shift
- Collaborative Filtering
  - Taste
- Utilities
  - Distance Measures
  - Parameters

Other ML Code Bases

- Naïve Bayes
- Perceptron
- PLSI/EM
- Genetic Programming
- Dirichlet Process Clustering
- Clustering Examples
- Hama: for very large arrays