BIG DATA: TECHNOLOGIES AND APPLICATIONS

9. Big Data Architecture, Hadoop, and Spark

Il-Yeol Song, Ph.D.
College of Computing & Informatics
Drexel University
Philadelphia, PA 19104

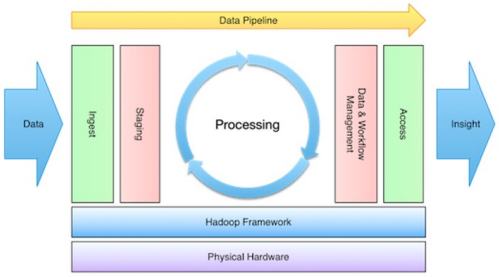
Big Data Technologies

- Big Data requires *scalable* technologies to efficiently process them within tolerable elapsed times.
  - Big data warehousing, data virtualization, data lake
  - *Scale-out Architecture*
  - *Parallel processing frameworks (MPP, MapReduce, Hadoop, Spark, and their ecosystems)*
  - Cloud Databases
  - NoSQL Databases
  - NewSQL Databases
  - Polyglot Persistence/Multimodel databases
  - IOT
  - Big Data Analytics (AI, Machine Learning, Data Science)

Big Data Processing

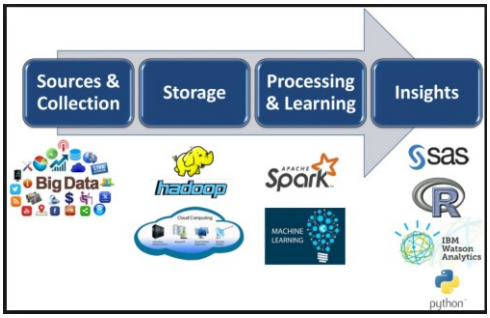
- Processing Big Data requires new technologies
  - Needs a ***scalable*** architecture
    - Use the scale-out architecture
  - Needs to meet the needs of 5Vs
- Hadoop and Spark are two most popular parallel processing frameworks for processing big data.
  - Open-sourced
- Map-Reduce is a parallel processing pattern,
  - Originally created by Google
  - Embedded in Hadoop and Spark

Big Data Processing Architecture

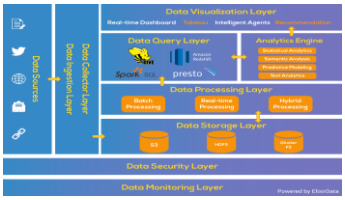


Source: http://blog.cloudera.com/blog/2014/09/getting-started-with-big-data-architecture/

Big Data Processing



Big Data Layered Architecture



Source: https://www.xenonstack.com/blog/ingestion-processing-data-for-big-data-iot-solutions

- The layered architecture of Big Data ensures a secure flow of data.
- This type of splitting ensures that each layer is performing a specific functionality.

Big Data Processing Technologies

- In this lecture, we cover:
  - Scale-up and Scale-out Architecture
  - MapReduce
  - Hadoop and its ecosystems
  - Spark
  - Big Data Architecture

Distributed Parallel Processing: Needs

- Computing (for analysis)
  - The amount of work required is greater than the capacity of a single CPU. Thus we need multiple CPUs and parallelism.
- Data Storage
  - Data are too big to store in one node and too slow to read from a singe node.
  - The obvious solution is to store in and read from multiple nodes simultaneously and in a distributed fashion.

⇒ Needs an architecture to easily perform distributed parallel processing

Distributed Parallel Processing: Challenges

There are several problems to work with multiple machines

- Coordination among multiple nodes
- Hardware failure → replication

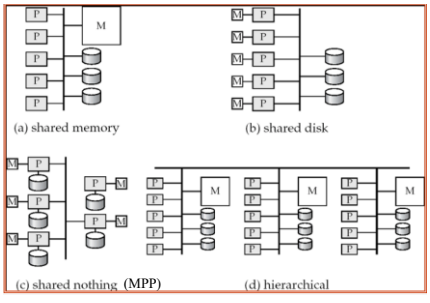
However, the developers do **not** want to think about these complexities

Image Source : http://hirendave.tech/programmers/tech-humor-a-programmers-revenge-deal-with-frustration-at-work/

How is big data handled?

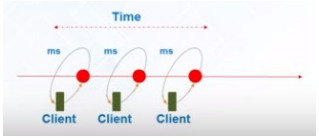
- Big data processing is done through ***parallelization***
  - Break input data into multiple streams and process each stream by different processors
  - Efficient parallel processing requires specialized technology
  - Not every process can be completely parallelized, but there are many tasks that are can be efficiently parallelized ("Embarrassingly parallel")
- Two common types of parallelism:
  - Split tasks on one computer across multiple cores (limited)
  - Divide tasks on *clusters* across multiple machines (scalable)

Parallel Processing Architecture



Batch vs Real-Time Processing

Analytics based on the data collected over a period of time is ***Batch Analytics***



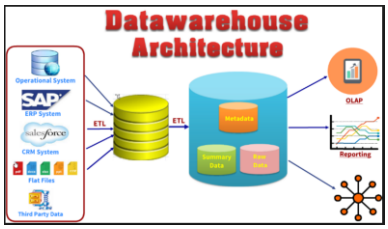
Analytics based on immediate for instant result is ***Real-time (Stream)Analytics***

Use Cases of Real-time Analytics

The Scaling Architecture Models

- Scaling Model
  - **Scale up: buy a more powerful server**
    - ✓**The traditional Relational database approach**
  - **Scale out: Add more commodity HWs**
    - ✓ Use the shared-nothing architecture
    - ✓Instead of buying a bigger server, add more commodity servers
    - ✓Hadoop, NoSQL databases, big data architecture

Traditional Relational Database Architecture

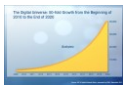


For Big Data, ETL and analytics cause performance bottleneck

Scale-Up Architecture

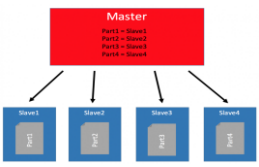
- **Buy a more powerful server:**

  - Add faster disks to physically read the data faster.
  - Increase RAM for row caching
  - Use more powerful CPU's for better transform performance
  - Increase or optimize networking performance for faster data delivery to the front end and analytics
- Cannot keep scaling-up because data grows exponentially.



The Scale-Out Architecture

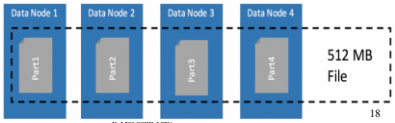
- **Add more commodity HWs**



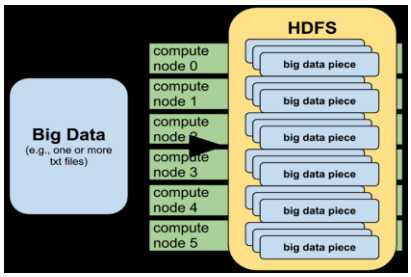
- **Master node (Name node)** manages what piece of data is stored on which slave
- **Slave (work or data) node** stores data and process queries

HDFS: Hadoop Distributed File System

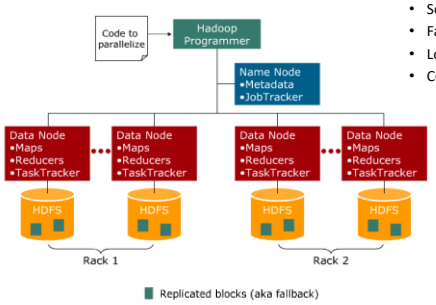
- HDFS is based on GFS (Google File System).
- GFS is a *chunk*-based distributed file system that supports fault-tolerance by data partitioning and replication.
- Data are automatically chopped up into 128MB in Hadoop 2.X (64 MB in Hadoop 1.X) blocks and stored into data nodes
- Fault-tolerance: blocks are automatically replicated to some other nodes (Default: 3 in HDFS)
- Scalability: Node can be easily added as needed
- Load balancing by a Master Node
- Cost-down



HDFS: Hadoop Distributed File System



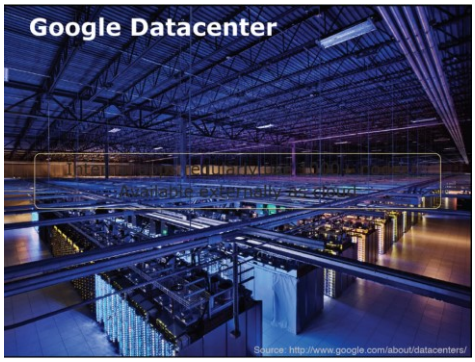
HDFS: Hadoop Distributed File System



- Scalability
- Fault-tolerant
- Load balancing
- Cost-down

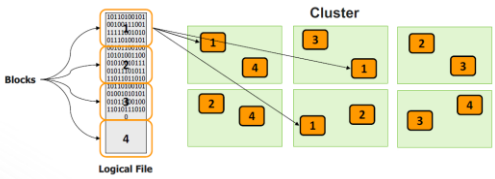
The Yahoo Hadoop Cluster

As of 2012, Yahoo! has one of the largest publicly announced Hadoop deployments at 42,000 nodes across several clusters utilizing 350 petabytes of raw storage.



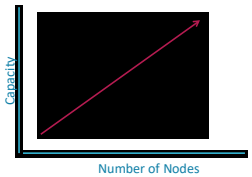
HDFS: Hadoop Distributed File System

- Replication and Fault-tolerant
  - Distribute 3 copies **randomly** across the cluster



Scalability

- **Increasing load results in a graceful decline in performance**
  - Not failure of the system
- **Adding nodes adds capacity proport i onally**



Map/Reduce Parallel Processing Framework

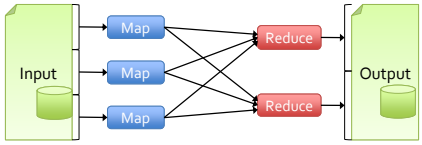
- A parallel programming model for big data processing
  - Developed by Google (2004):
    - To index web pages
    - The output was used to create the page ranking for Google Search
  - Scalable by using the scale-out architecture
  - Idea: parallelize *independent* tasks
    - » **MAP** phase:
      - » Read data from HDFS and perform MAP function
    - » **REDUCE** phase:
      - » Perform the computation and store the results to HDFS
  - Hadoop is an open source platform that implements Map/Reduce on the top of HDFS (Hadoop Distributed File System)

Map/Reduce

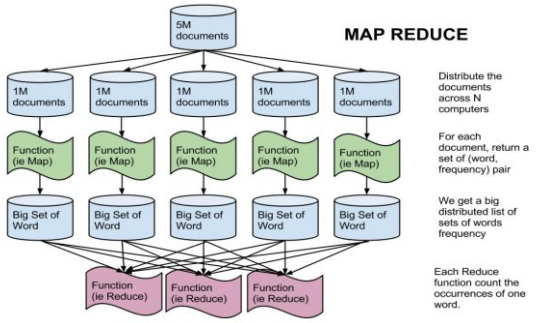
- map(key, val) is run on each item in set
  - emits new-key / new-val pairs
- reduce(key, vals) is run for each unique key emitted by map()
  - emits final output

MR in Hadoop Framework

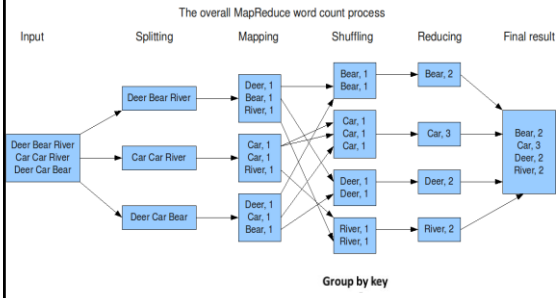
(Hadoop transforms data flowing from a stable storage to another stable storage



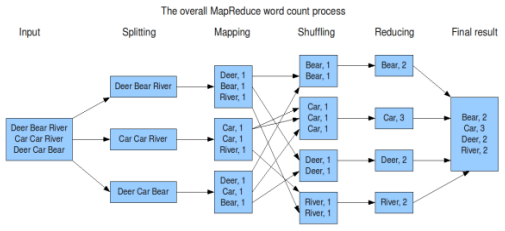
Steps of MapReduce



Word Count Problem

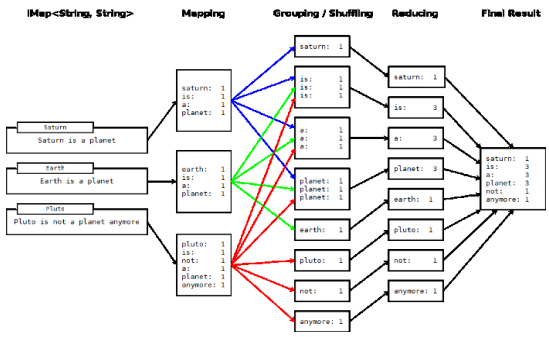


Data is in the form of (key, value).
e.g., Deer is key and 1 is a value.



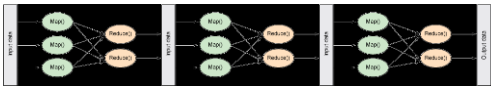
- The input for MapReduce is a list of (key1, value1) pairs and Map() is applied to each pair to compute intermediate key-value pairs, (key2, value2).
- The intermediate key-value pairs are then grouped together on the keyequality basis, i.e. (key2, list(value2)).
- For each key2, Reduce() works on the list of all values, then produces zero or more aggregated results.

Find Top 10 Words



Complex MapReduce Model

- For complex work, chain M/R jobs together
  - Use a higher level language or APIs for you



MapReduce & Hadoop in Data Analytics Life Cycle

- MapReduce represents activities:
  - Phase 1 and 2 of the lifecycle in acquire/parse and filter stages for ETL
  - Phase 4 with model building

Source: Data Science And Big Data Analytics Course Student Guide, Education Services, October 2012

Hadoop & Map/Reduce

- Map/Reduce is a parallel programming framework with automatic parallelization
- Apache Hadoop is an open source platform that :
  - Implements Map/Reduce algorithm
  - Supports extreme **scalability** using low cost commodity hardware
  - **Provide Fault tolerance**
  - Process and store large amounts of structured, unstructured and semi-structured data
- Developed by Doug Cutting and Mike Cafarella (2004), later hired by Yahoo

Two Core Components of Hadoop 1

(1) MapReduce: A scalable parallel processing framework

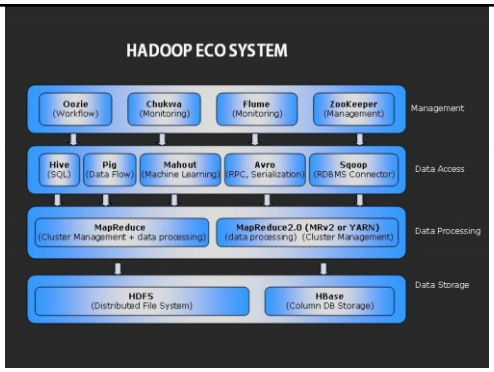
(2) HDFS: distributed, **scalable**, and **fault-tolerant** file-system written in Java in a scale-out architecture

Why Hadoop (I)?

- The ability to handle **big data** quickly.
- **Low cost**: open source and commodity hardware
- **Distributed** computing power
- **Scalability** by adding nodes
- **Storage flexibility**: no preprocessing; no ETL, **structured and unstructured, multimedia** data
- **Triple storage**: automatic redirection to a duplicated copy if hardware failure occurs

Why Hadoop (II)?

- It is open source and free, but also available from vendors like IBM, Cloudera, and EMC Greenplum with administrative tools, support and maintenance.
- Hadoop currently is a read-only system once data has been written into the HDFS, users cannot easily insert, delete or modify individual data, as in RDBs.
- Supported by a large and active ecosystems
  - E.g., Hive, Hbase, Pig, Storm, etc.



Hadoop Ecosystem

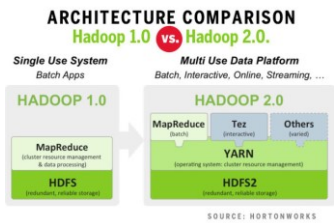
- **HBase**: A Non-relational columnar NoSQL DB that uses HDFS
- **Hive**: *SQL-like* access to data;
- **Pig**: a scripting language to perform ETL for data stored in HDFS
- **Mahout**: scalable machine learning algorithms
- **Storm**: a distributed real time computation system for stream data (Twitter uses Storm to identify trends in near real time)

Other Hadoop Components

- **Flume**
  - A software that collects, aggregates and moves large amounts of streaming data into HDFS.
- **Sqoop**
  - A connection and transfer mechanism that moves data between Hadoop and relational databases.
- **Oozie**
  - A Hadoop job scheduler.
- **Zookeeper**
  - An application that coordinates distributed processes.

Hadoop 1 and Hadoop 2

- Hadoop 1 (MapReduce1) vs. Hadoop 2 (YARN or MapReduce 2)
  - Hadoop1 (2005) : Single Use System (**batch** applications only)
  - Hadoop2 (2012): Multi Use Data Platform (**batch, interactive, online, streaming, graph**, etc.) **Extends to non M/R applications**



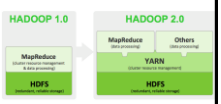
Source: http://www.networkworld.com/article/2369327/software/comparing-the-top-hadoop-distributions.html

Three Core Components of Hadoop 2

(1) MapReduce: A scalable parallel processing framework

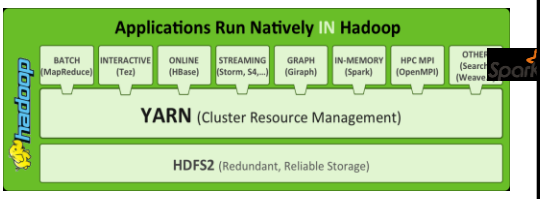
(2) HDFS: distributed, scalable, and fault-tolerant file-system written in Java

(3) YARN (Yet Another Resource Negotiator):
A resource management framework for **scheduling and handling resource requests** from distributed applications.



Source: http://en.wikipedia.org/wiki/Apache_Hadoop

Tools Running on YARN Layer in Hadoop 2



Source: http://blog.andreamostosi.name/2014/03/hadoop-vs-berkeley/

Hadoop Use Cases

Why & where Hadoop is used / not used

- What Hadoop is good for:
  - Massive amounts of data through parallelism
  - A variety of data (structured, unstructured, semi-structured)
  - Inexpensive commodity hardware
- Hadoop is not good for:
  - Not to process transactions (random access)
  - Not good when work cannot be parallelized
  - Not good for low latency data access
  - Not good for processing lots of small files
  - Not good for intensive calculations with little data

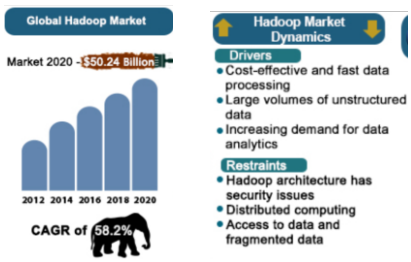
Hadoop Use Cases

- **IBM Watson (2011)**
  - To educate Watson, Hadoop was utilized to process various data sources such as encyclopedias, dictionaries, news wire feeds, literature, and the entire contents of Wikipedia .
  - For each clue provided during the game, Watson had to perform the following tasks in less than three seconds
    - Deconstruct the provided clue into words and phrases
    - Establish the grammatical relationship between the words and the phrases
    - Create a set of similar terms to use in Watson's search for a response
    - Use Hadoop to coordinate the search for a response across terabytes of data
    - Determine possible responses and assign their likelihood of being correct
    - Actuate the buzzer
    - Provide a syntactically correct response in English

Hadoop Use Cases

- **LinkedIn (**Acquired by Microsoft by $26.2 billion in 2016)
  - LinkedIn is an online professional network of 433 million users in 200 countries as of early 2016.
  - LinkedIn provides several free and subscription-based services, such as company information pages, job postings, talent searches, social graphs of one's contacts, personally tailored news feeds, and access to discussion groups, including a Hadoop users group.
  - LinkedIn utilizes Hadoop for the following purposes
    - Process daily production database transaction logs
    - Examine the users' activities such as views and clicks
    - Feed the extracted data back to the production systems
    - Restructure the data to add to an analytical database
    - Develop and test analytical models

Global Hadoop Market



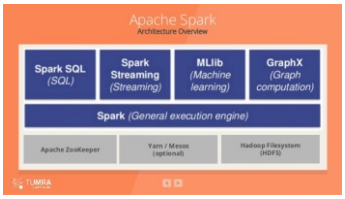
Source: Allied Market Research, 2015

Hadoop Summary

- **Hadoop Consists of M/R paradigm and HDFS**
- The main idea of the MapReduce model is to hide details of parallel execution and allow users to focus only on data processing strategies.
- HDFS
  - HDFS stores files in chunks which are physically stored on multiple compute nodes
  - HDFS still presents data to users and applications as single continuous files despite the above fact
- **Map-reduce is ideal for operating on very large, flat (unstructured) datasets and perform trivially parallel operations on them.**

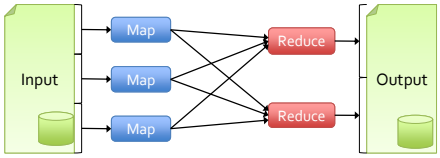
Apache Spark

- A in-memory centric computing platform for real-time analytical processing:
  - Runs up to 100X faster than Hadoop for iterative applications
  - Good for iterative or real-time processing (ML, streaming, graph, etc)



Motivation

Hadoop transforms data flowing from a stable storage to another stable storage



Using memory improves interactive, real-time processing

Spark History and Features

- Originally developed by Matei Zaharia at Berkeley (2009)
- Open-source released in early 2014
- DataBrick was founded to commercialize Spark
  - Spark is written in Scala, but has nice Python and R abstractions.
  - Spark does not necessarily replace Hadoop or HDFS.
  - Sparks does a lot with intermediate data in memory.
  - Also, it's fault-tolerant, with resilient distributed datasets (RDDs)
  - RDDs are efficient, since faults don't mean "start over."
  - Also, keeping data in memory can make Spark much faster. Since memory is limited, Spark will write to disk if necessary.

Spark Updates Hadoop

- Hardware had advanced since Hadoop started:
  - Very large RAMs, Faster networks (10Gb+)
  - Bandwidth to disk not keeping up
- MapReduce is awkward for some key big data workloads:
  - Low latency dispatch (e.g., quick queries)
  - Iterative algorithms (e.g., ML, Graph…)
  - Streaming data ingest
- Enhance programmability: Scala, Python, Java APIs

Easy and Fast Big Data

- **Easy to Develop**
  - Rich APIs in Java, Scala, Python
  - Interactive shell

2-10× less code

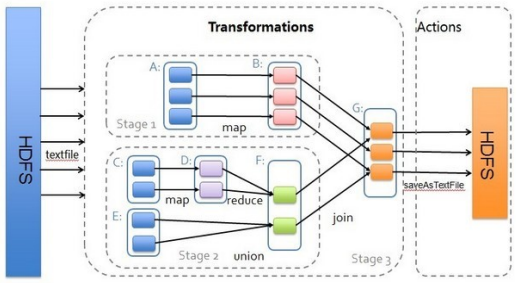
- **Fast to Run**
  - General execution graphs
  - In-memory storage

Up to 10× faster on disk, 100× in memory

Apache Spark

- **Why Spark is faster than Hadoop ?**
  - **Spark is geared toward in-memory processing,** while Hadoop is very disk-dependent
    - Well suited for **iterative algorithms that require multiple passes** over in-memory data as in machine learning algorithms
    - Stores **RDD** (Resilient Distributed Dataset) in memory, which can be recovered from failures.
- In addition to "map" and "reduce", defines a large set of *operations* (transformations & actions)
  - Transformations (e.g. map, filter, groupBy, union, join)
    - Produce an RDD that contains the result
  - Actions (e.g. count, reduce, collect, save): used to write
  - Operations can be arbitrarily combined in any order
- Spark can be used with Hadoop or without Hadoop.

Spark: Transformations & Actions



On-Disk Sort Record:
Time to sort 100TB

2013 Record: Hadoop — 2100 machines, 72 minutes

2014 Record: Spark — 207 machines, 23 minutes

Also sorted 1PB in 4 hours
Query 2 TB of data in a fraction of a second

Source: Daytona GraySort benchmark, sortbenchmark.org

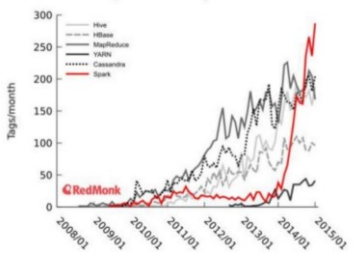
Spark Use Cases

Apache Spark Use Cases

- **Yahoo! Personalizes news pages for Web visitors**
  - Uses ML algorithms to figure out what individual users are interested in, and to categorize news stories as they arise to figure out what types of users would be interested in reading them.
- **Yahoo! runs analytics for advertising**
  - View and query their advertising analytic data collected in Hadoop
- **Conviva: the 2nd largest video streaming company**
  - Streams 4 billion video feeds per month
  - Uses Spark Streaming to learn network conditions in real time and control video traffic

Source: https://spark-summit.org/2014/talk/spark-use-case-at-telefonica-cbs

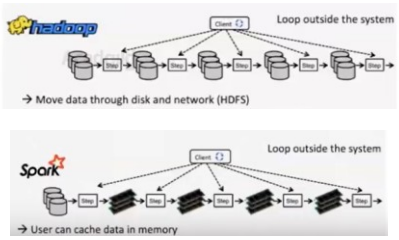
Adoption of Spark



Source: https://spark-summit.org/2014/talk/spark-use-case-at-telefonica-cbs

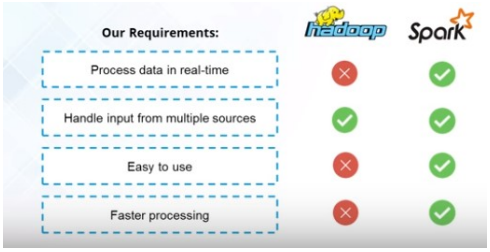
Hadoop and Spark

- Hadoop is disk-based, while Spark is memory-based



Hadoop and Spark Use Cases

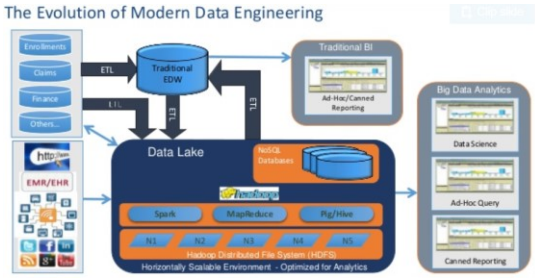
Hadoop implements Batch processing on Big Data.
It thus cannot deliver to our Real-Time use case needs.



Hadoop and Spark

- **Hadoop:**
  - Good for sequential batch processing
  - "Really" large data sets
  - Mature and widely commercially supported tools
- **Spark:**
  - Better for iterative processing for running ML algorithms or graph processing
  - Better for real-time processing
  - Easier to program than Hadoop
  - Hugh investment in tools going-on
  - Details, tutorials, videos: www.spark-project.org

Integrated Modern Big Data Architecture



Review

- How is batch analytics different from real-time analytics?
- What is MPP?
- What is the scale-out architecture? What are its major features?
- Why the scale-up architecture is not suitable for big data processing?
- What is HDFS? What are their important features?
- Can you describe Map/Reduce algorithm?
- What is the relationship between Map/Reduce and Hadoop?
- What is the purpose of the Namenode in HDFS?
- Briefly summarize Spark.
- What are major differences between Hadoop and Spark?
- What does the modern big data architecture look like?