

# **Chapter1**

# **Fundamentals of Big Data**

# **Analytics**

**Basanta Joshi, PhD**

Asst. Prof., Depart of Electronics and Computer Engineering  
Program Coordinator, MSc in Information and Communication Engineering  
Member, Laboratory for ICT Research and Development (LICT)  
Member, Research Management Cell (RMC)

Institute of Engineering  
[basanta@ioe.edu.np](mailto:basanta@ioe.edu.np)

<http://www.basantajoshi.com.np>

<https://scholar.google.com/citations?user=iocLiGcAAAAJ>  
[https://www.researchgate.net/profile/Basanta\\_Joshi2](https://www.researchgate.net/profile/Basanta_Joshi2)

# Presenter Profile - Dr. Bhuvan Unhelkar

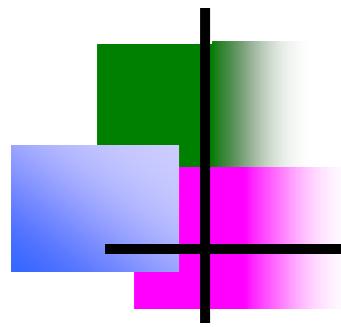
(BE, MBA, PhD, FACS, CBAP®, PSM)



- Professor of IT, USF (Sarasota-Manatee Campus)
  - Founder, MethodScience.com & PlatiFi.com
  - Courses: Adv Prog. Design (UML), Agile PM, Data & Security Analytics, NoSQL, Sr. Project
  - PhD from University of Technology, Sydney (UTS), 1997  
*“Effect of Granularity of OO Design in Modelling an Enterprise and its application to Financial Risk Management”*; Guide: Prof. **Brian Henderson-Sellers**
  - Author: **25 Books** (AI, Mobile, Agile, Green ICT, and Big Data Strategies for Agile Business )
  - Supervisor: **8 PhD Completions**;
  - **Fellow of the Australian Computer Society; IEEE Sr. member**, Life Member, Computer Society of India & BMA
  - Hon. Prof. Amity Univ. (India), Western Sydney Univ. (Australia)
  - Past President – Rotary club of Sarasota Sunrise; Rotary Club of St Ives, Paul Harris Fellow; AG); TiE;
  - [www.unhelkar.com](http://www.unhelkar.com) & [www.methodscience.com](http://www.methodscience.com)
- 
- I will be teaching in Collaboration with Prof. Bhuvan

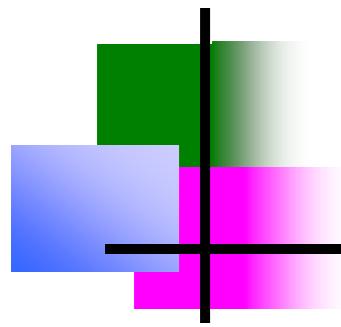


# What is Big Data?



---

why is there  
such *buzz* around  
**BIG DATA?**



---

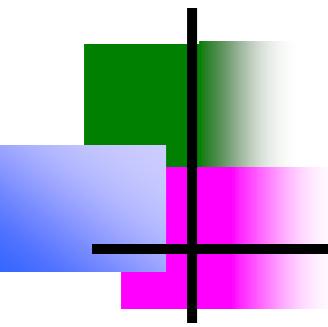
Isn't *Big Data*  
just big hype?



... How big is BIG?



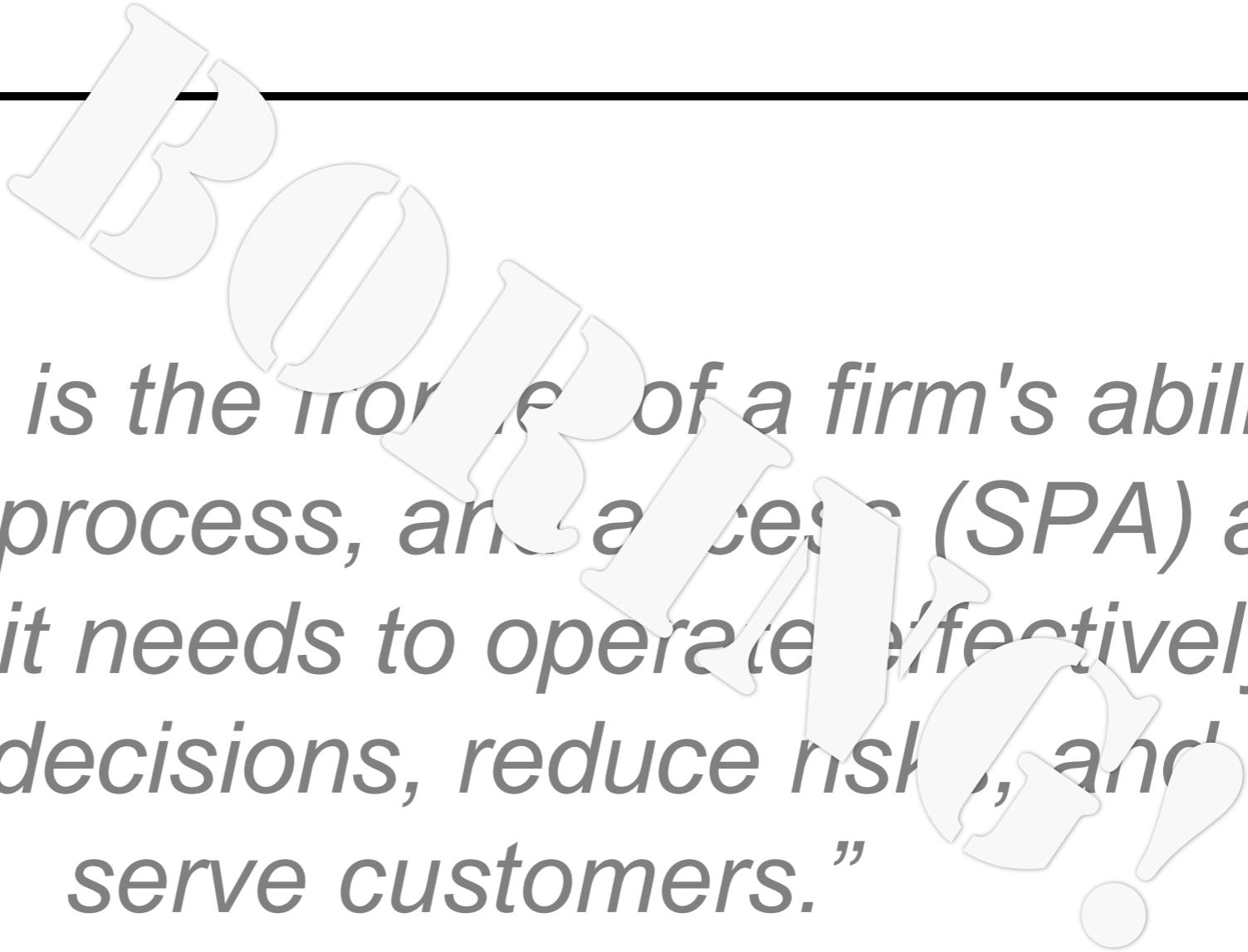
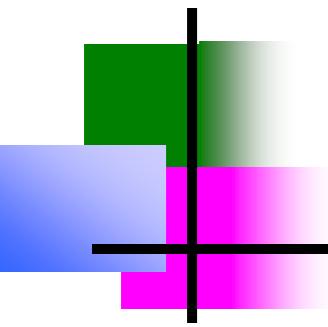
What is it that we are  
really talking about?



---

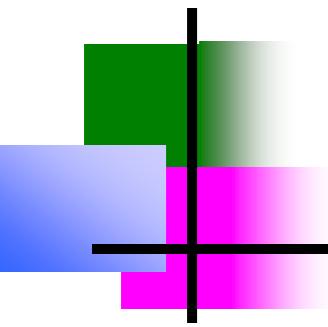
*“Big Data is the frontier of a firm's ability to store, process, and access (SPA) all the data it needs to operate effectively, make decisions, reduce risks, and serve customers.”*

-- Forrester



*“Big Data is the more of a firm's ability to store, process, and access (SPA) all the data it needs to operate effectively, make decisions, reduce risk, and serve customers.”*

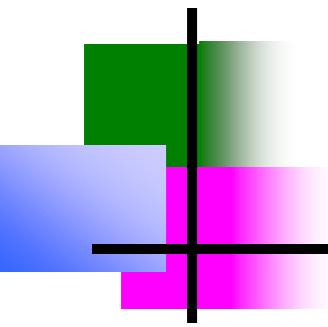
-- Forrester



---

*“Big Data in general is defined as high volume, velocity and variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making.”*

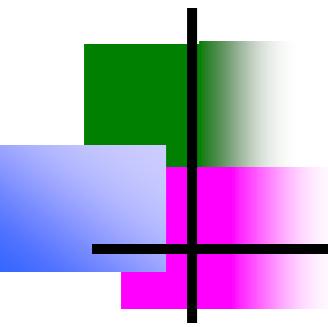
-- Gartner



---

*“Big Data in general is defined as high volume, velocity and variety information assets that demand cost effective, innovative forms of information processing for enhanced insight and decision making.”*

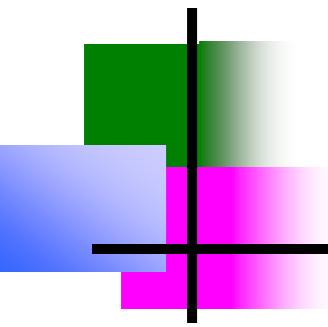
-- Gartner



---

*“Big data is data that exceeds the processing capacity of conventional database systems. The data is too big, moves too fast, or doesn't fit the strictures of your database architectures. To gain value from this data, you must choose an alternative way to process it.”*

-- O'Reilly

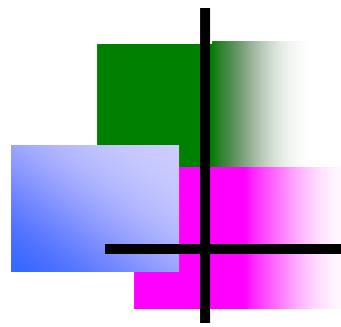


---

*“Big data is data that exceeds the processing capacity of conventional database systems. The data is too big, moves too fast, or doesn't fit the ‘rich’ resources of your database architecture. To gain value from this data, you must choose an alternative way of thinking.”*

— Matt Parker

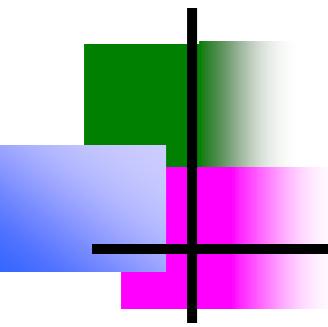
-- O'Reilly



---

*“Big data is the data characterized by 3 attributes: volume, variety and velocity.”*

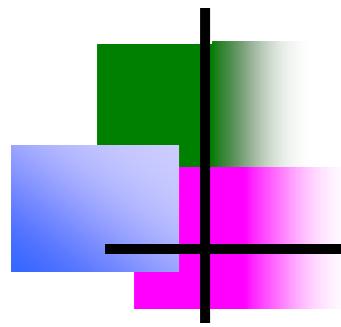
-- IBM



---

**RANDOM**  
*“Big data is the data characterized by 3  
attributed: volume, variety and velocity.”*  
**WORDS**

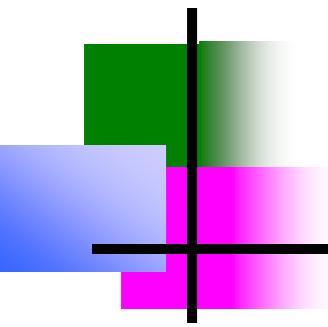
-- IBM



---

*“Big data is the data characterized by 4 key attributes: volume, variety, velocity and value.”*

-- Oracle

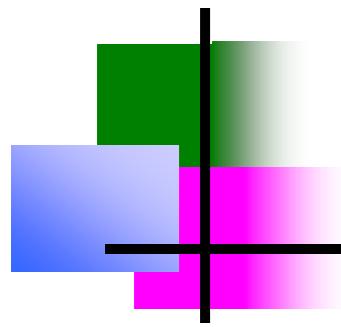


---

*“Big data is the data characterized by 4 key attributes: volume, velocity, variety, and value.”*

-- Oracle





---

Let's look at  
**Big Data**  
in a different way.



What was your  
first computer?





What was its  
“Big Data” limit?



**DevOps Borat**

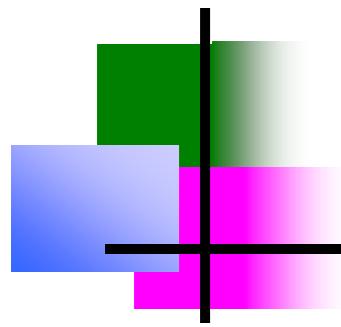
@DEVOPS\_BORAT

Small Data is when is fit in RAM.  
Big Data is when is crash because  
is not fit in RAM.

2/6/13, 8:22 AM



...



---

Let's try again...

Byte : one grain of rice



Byte

Byte : one grain of rice

Kilobyte : cup of rice

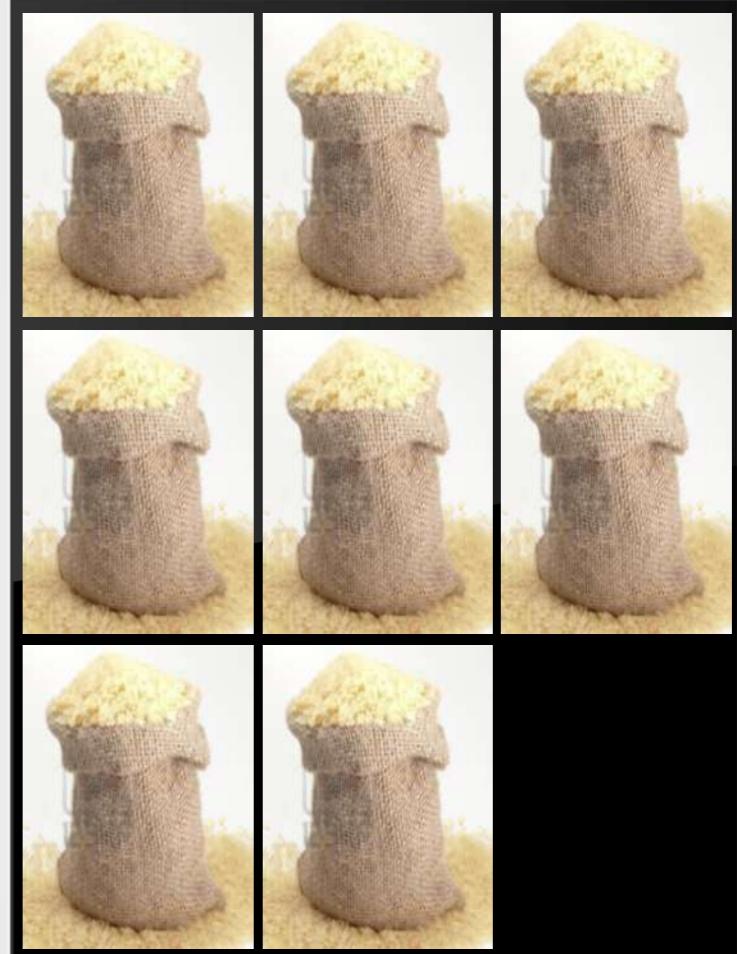


Kilobyte

Byte : one grain of rice

Kilobyte : cup of rice

Megabyte : 8 bags of rice



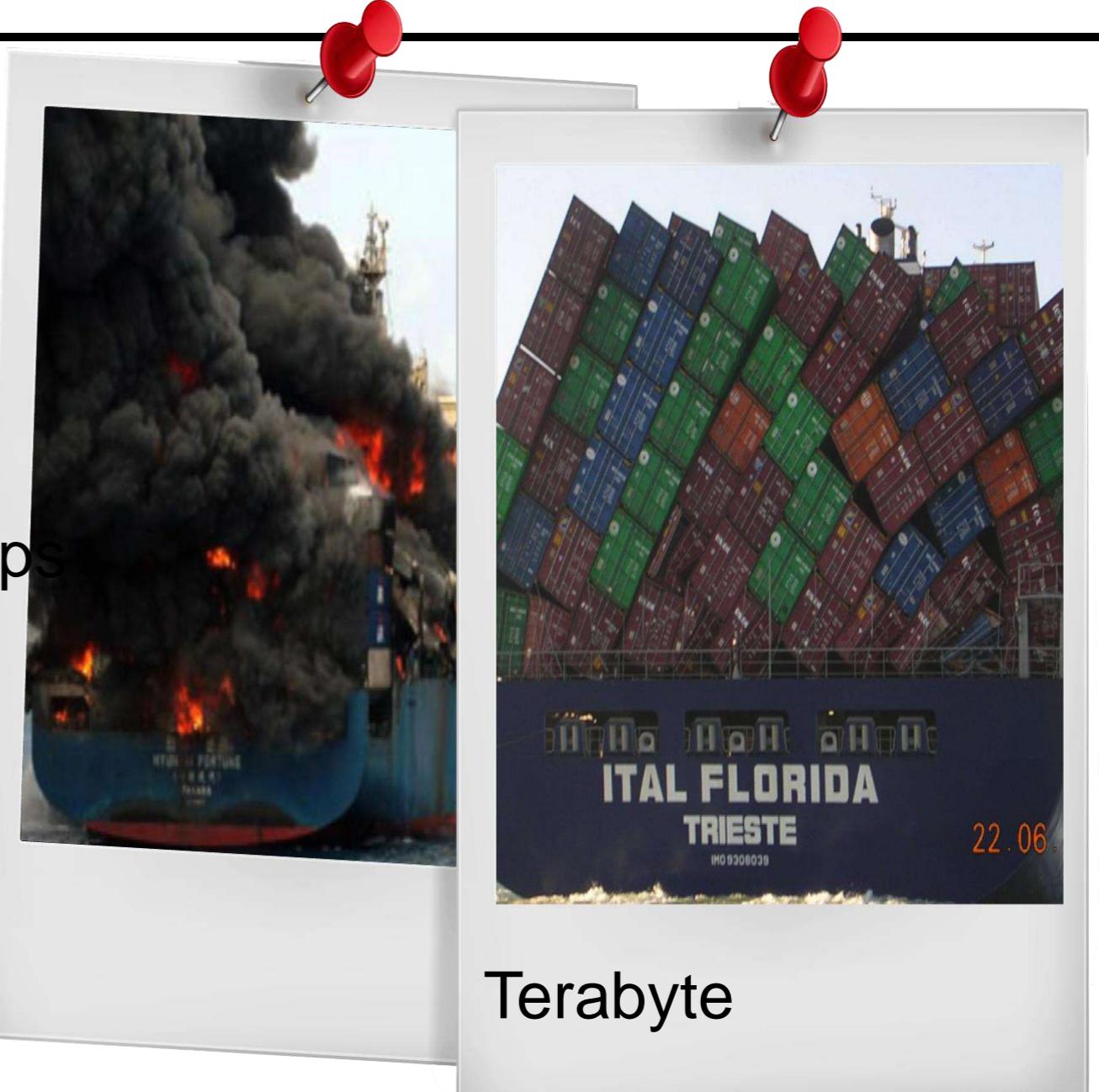
Megabyte

Byte : one grain of rice  
Kilobyte : cup of rice  
Megabyte : 8 bags of rice  
**Gigabyte** : 3 Semi trucks

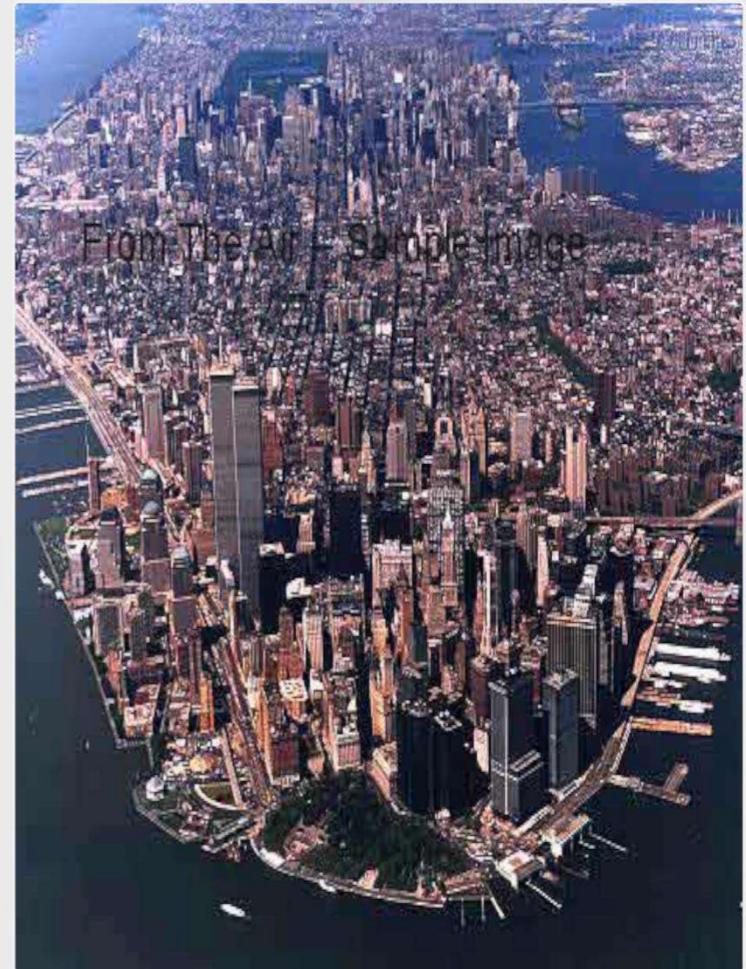


**Gigabyte**

Byte : one grain of rice  
Kilobyte : cup of rice  
Megabyte : 8 bags of rice  
Gigabyte : 3 Semi trucks  
Terabyte : 2 Container Ships

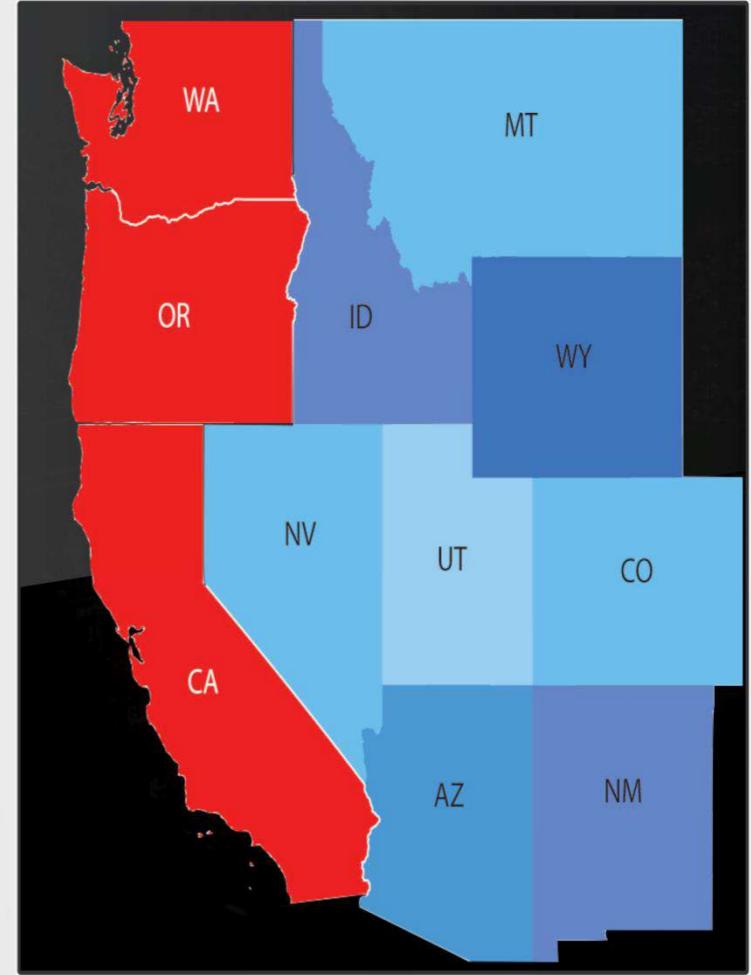


Byte : one grain of rice  
Kilobyte : cup of rice  
Megabyte : 8 bags of rice  
Gigabyte : 3 Semi trucks  
Terabyte : 2 Container Ships  
Petabyte : Blankets Manhattan



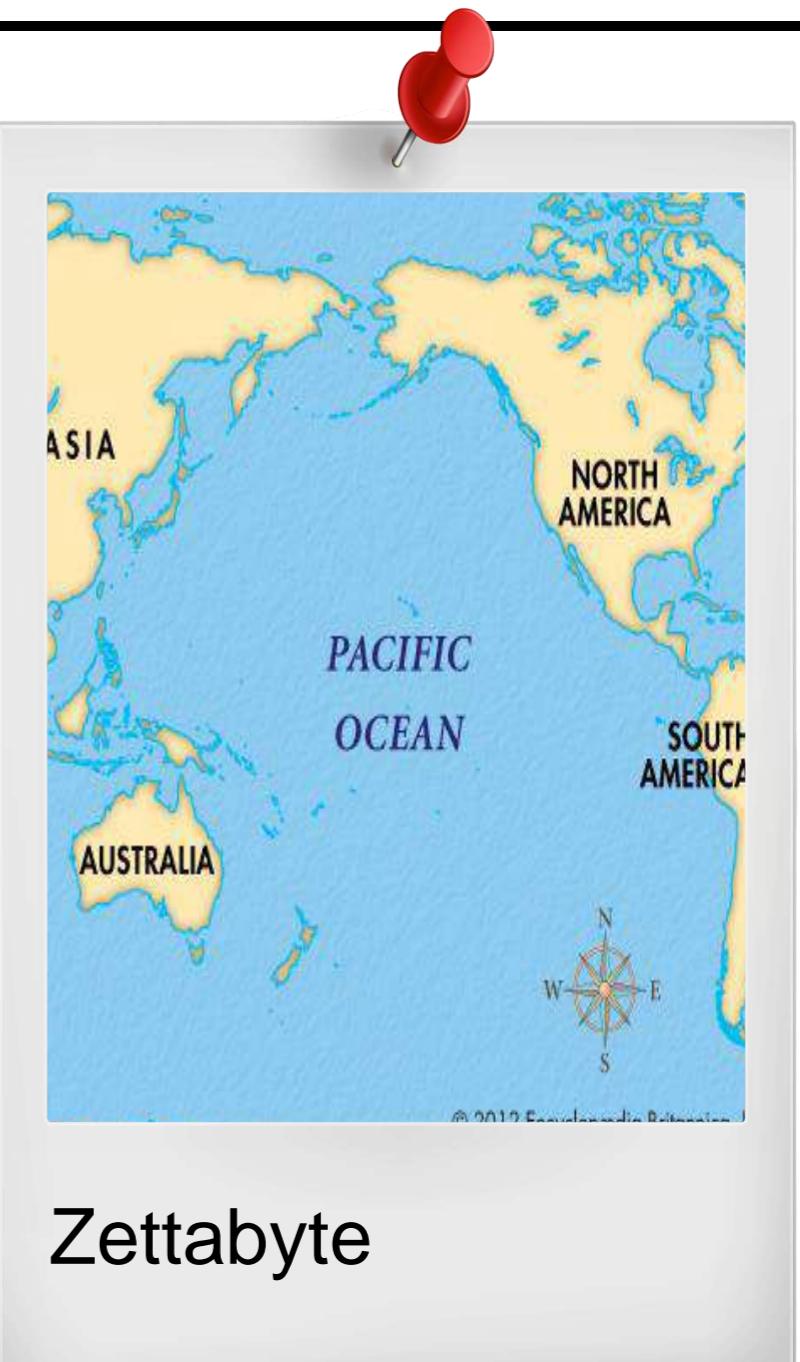
Petabyte

Byte : one grain of rice  
Kilobyte : cup of rice  
Megabyte : 8 bags of rice  
Gigabyte : 3 Semi trucks  
Terabyte : 2 Container Ships  
Petabyte : Blankets Manhattan  
**Exabyte** : Blankets west coast states

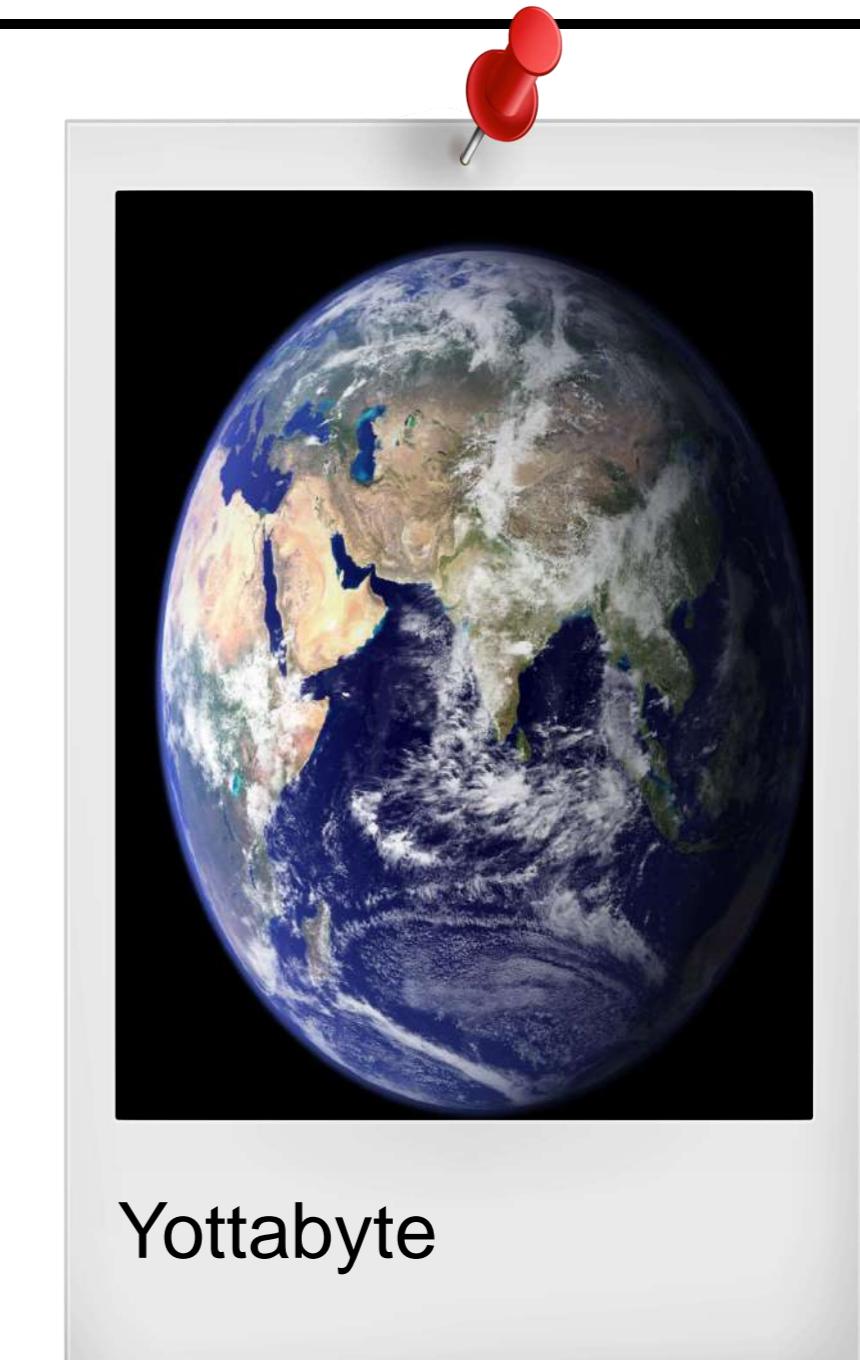


**Exabyte**

Byte : one grain of rice  
Kilobyte : cup of rice  
Megabyte : 8 bags of rice  
Gigabyte : 3 Semi trucks  
Terabyte : 2 Container Ships  
Petabyte : Blankets Manhattan  
Exabyte : Blankets west coast states  
Zettabyte : Fills the Pacific Ocean



Byte : one grain of rice  
Kilobyte : cup of rice  
Megabyte : 8 bags of rice  
Gigabyte : 3 Semi trucks  
Terabyte : 2 Container Ships  
Petabyte : Blankets Manhattan  
Exabyte : Blankets west coast states  
Zettabyte : Fills the Pacific Ocean  
**Yottabyte** : A EARTH SIZE RICE BALL!



**Yottabyte**



Megabyte : 8 bags of rice

Gigabyte : 3 Semi trucks

Terabyte : 2 Container Ships

Petabyte : Blankets Manhattan

Exabyte : Blankets west coast states

Zettabyte : Fills the Pacific Ocean

Yottabyte : A EARTH SIZE RICE BALL!

Byte : one grain of rice



Hobbyist

Kilobyte : cup of rice



Terabyte : 2 Container Ships

Petabyte : Blankets Manhattan

Exabyte : Blankets west coast states

Zettabyte : Fills the Pacific Ocean

Yottabyte : A EARTH SIZE RICE BALL!

Byte : one grain of rice



Hobbyist

Kilobyte : cup of rice



Desktop

~~Megabyte~~ : 8 bags of rice

Gigabyte : 3 Semi trucks



Exabyte : Blankets west coast states

Zettabyte : Fills the Pacific Ocean

Yottabyte : A EARTH SIZE RICE BALL!

Byte : one grain of rice



Hobbyist

Kilobyte : cup of rice



Desktop

Megabyte : 8 bags of rice

Gigabyte : 3 Semi trucks

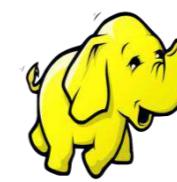
Terabyte : 2 Container Ships

Petabyte : Blankets Manhattan



Internet

Yottabyte : A EARTH SIZE RICE BALL!





Byte : one grain of rice

Kilobyte : cup of rice

Megabyte : 8 bags of rice

Gigabyte : 3 Semi trucks

Terabyte : 2 Container Ships

Petabyte : Blankets Manhattan

~~Yottabyte : A EARTH SIZE RICE BALL!~~

**YAHOO!**

amazon.com

**ebay**

**Google**

Byte : one grain of rice

Kilobyte : cup of rice

Megabyte : 8 bags of rice

Gigabyte : 3 Semi trucks

Terabyte : 2 Container Ships

Petabyte : Blankets Manhattan

Exabyte : Blankets west coast states

Zettabyte : Fills the Pacific Ocean



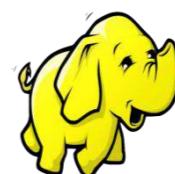
Hobbyist



Desktop



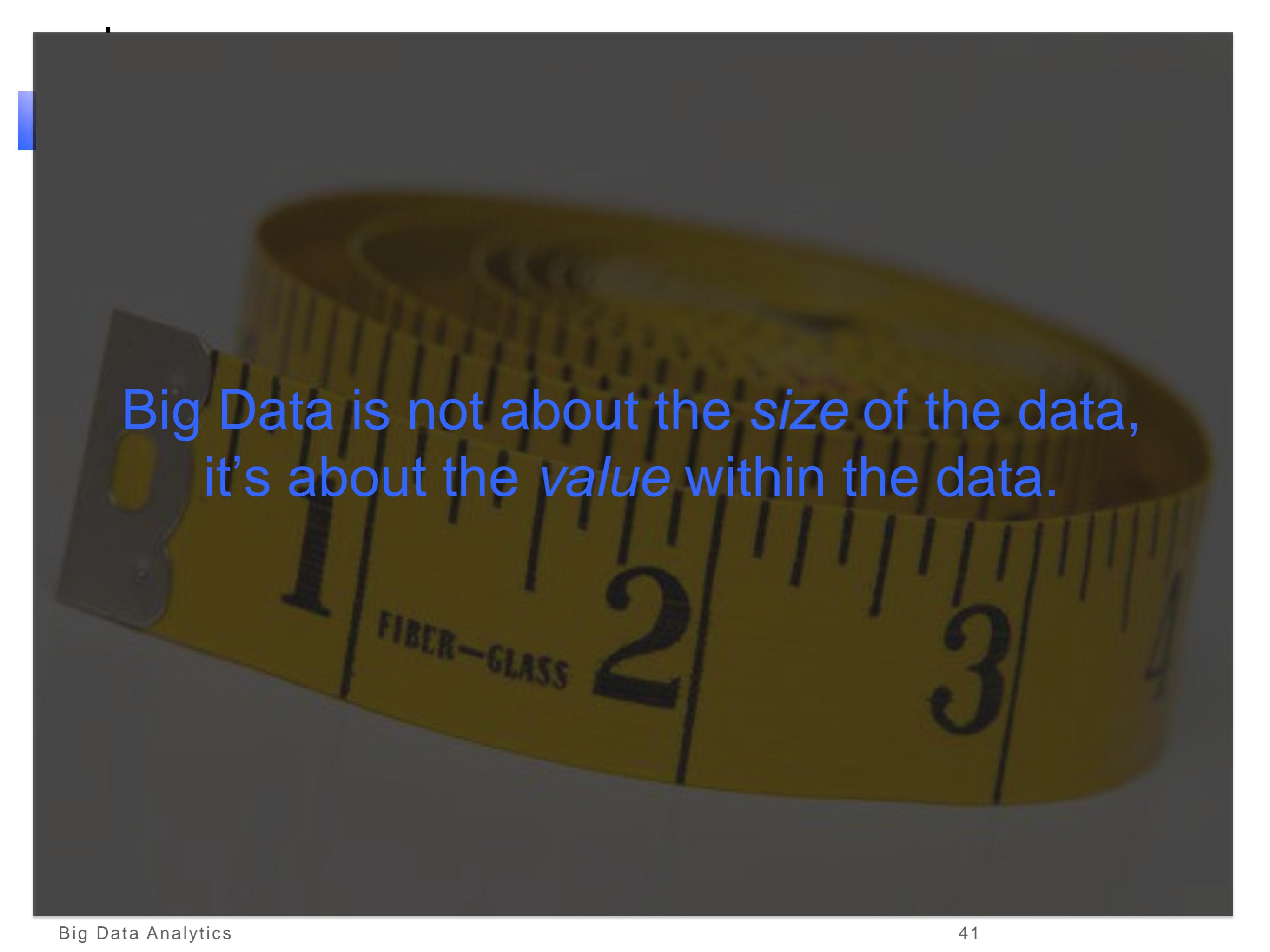
Internet



Big Data



*“Utah Governor Herbert on June 11, 2012 told the annual meeting of the National Governors’ Association that the NSA’s 1,500,000 square foot data center being built outside Salt Lake City will be the first facility to house a yottabyte of data.”*



Big Data is not about the size of the data,  
it's about the *value* within the data.

A close-up photograph of a person's hand holding a ring. The ring features a prominent, ornate skull and crossbones emblem in the center, surrounded by a circular pattern of small diamonds or similar stones. The band of the ring is also decorated with intricate metalwork. The background is dark and out of focus.

So, what is  
value?



Market Intelligence?  
Business Intelligence?  
Secret Intelligence?



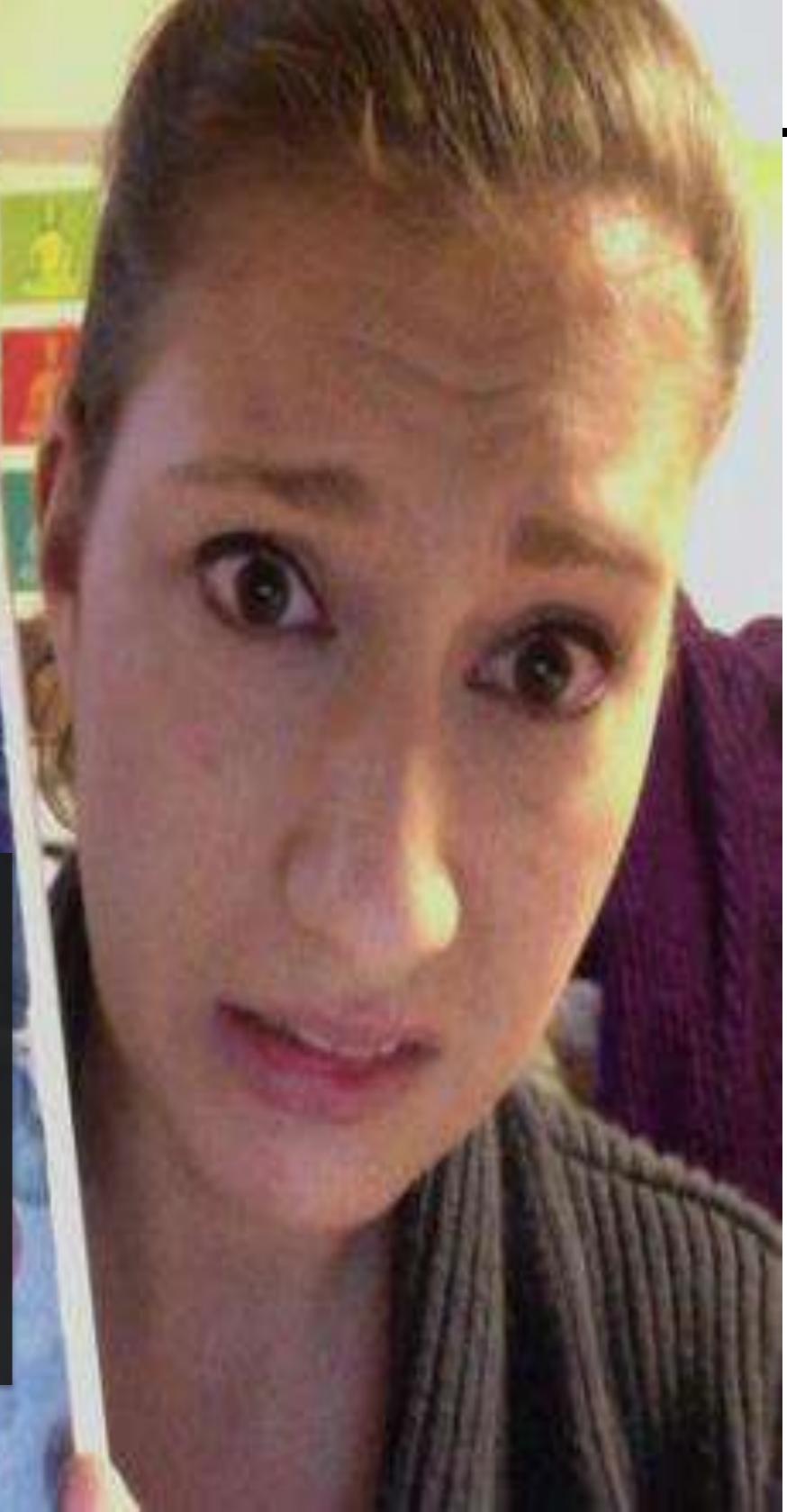
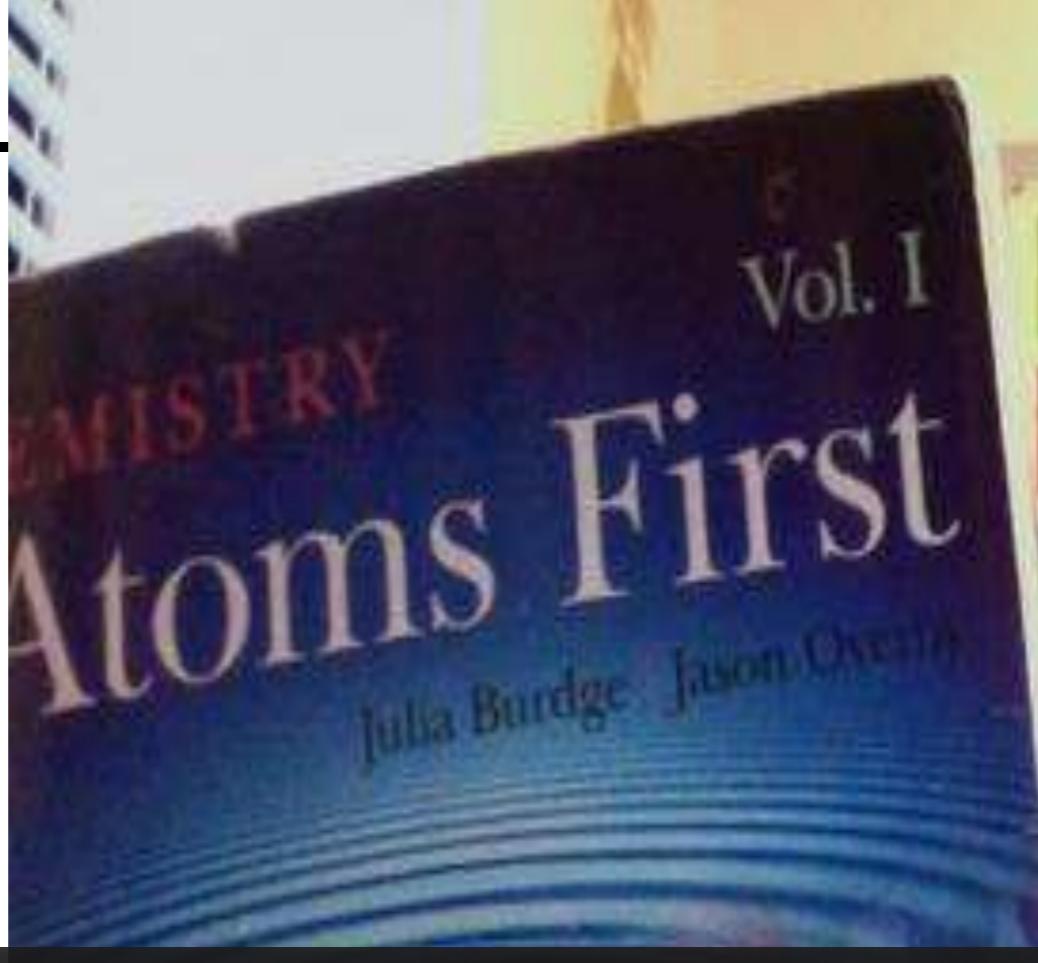
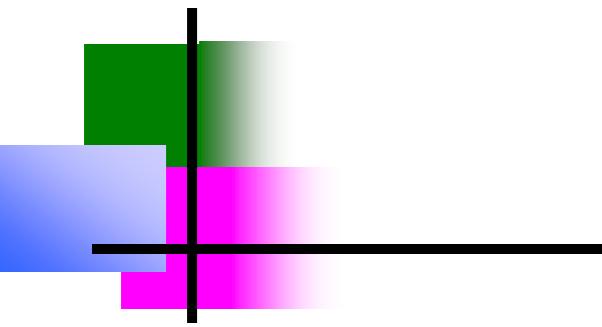
*Our society is leaving  
behind a digital footprint.*



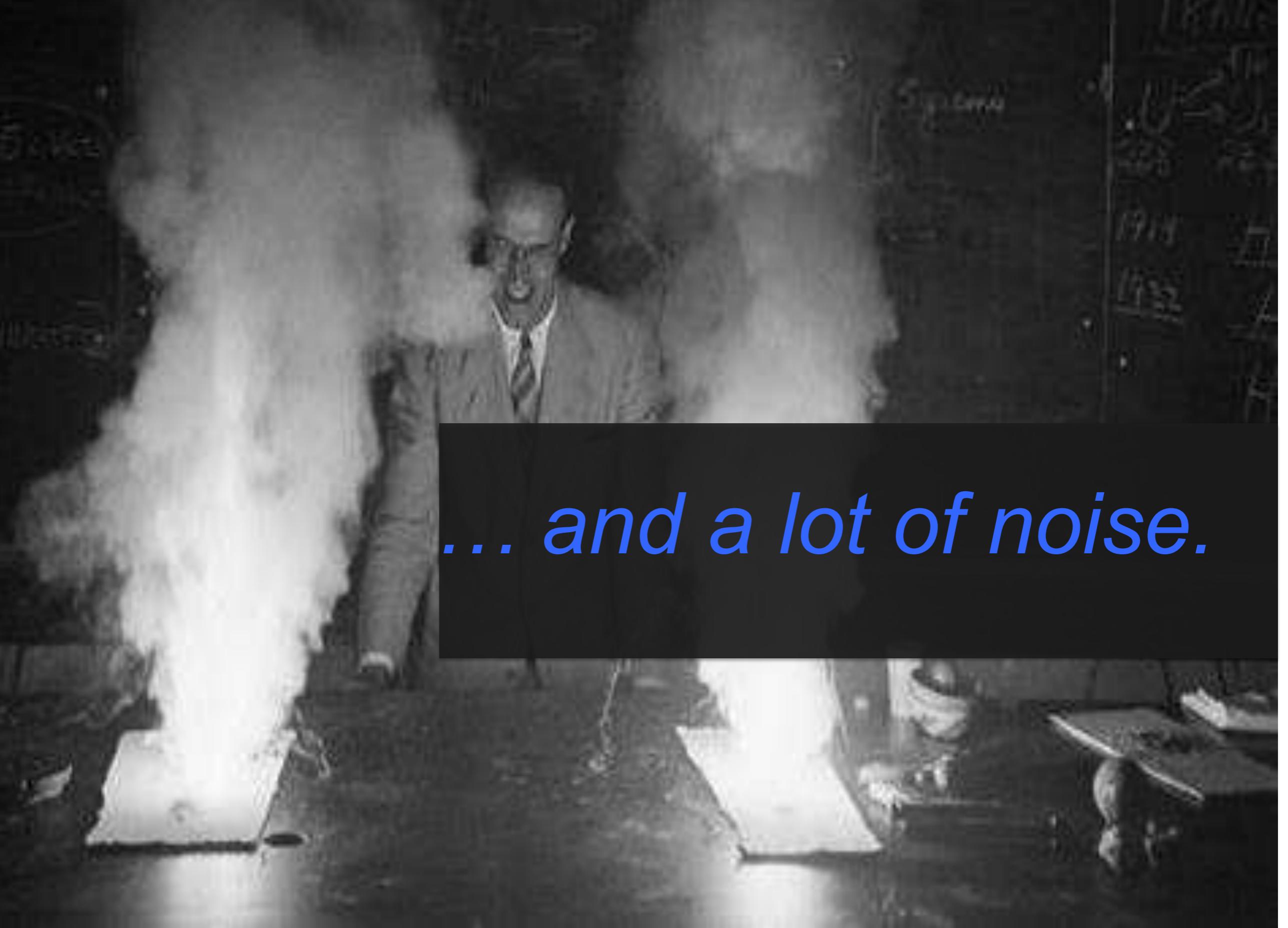
*People are living online and we all are expressing our attitudes, likes and dislikes, our opinions and perspectives.*

*We are generating huge amounts of data.*





*Data with a  
lot of information.*



*... and a lot of noise.*



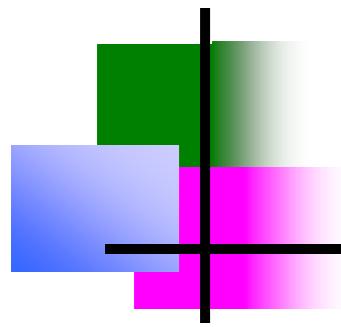
*The ability to hear the signal  
from the noise is the key...*



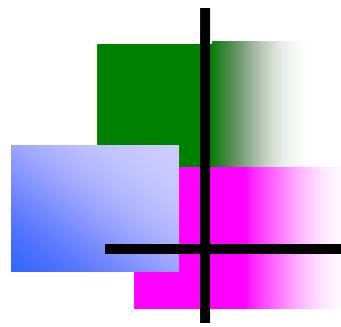
*to unlocking the human conversation  
that is taking place around us.*



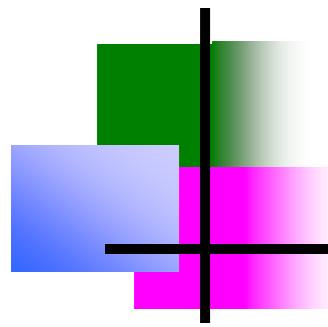
*Listening to this  
conversation  
can lead us to...*



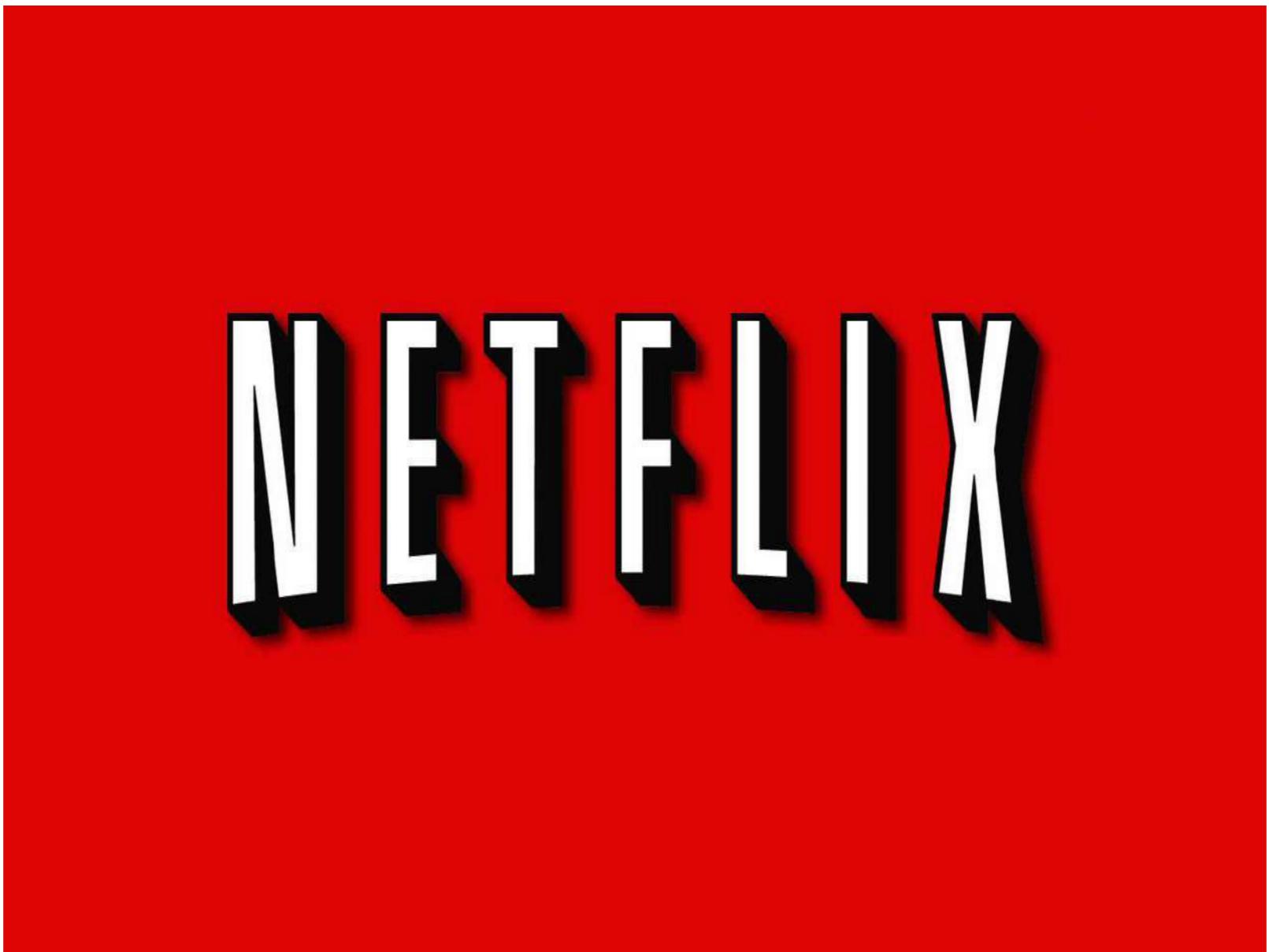
# *Unexpected Discoveries*



# Case Study



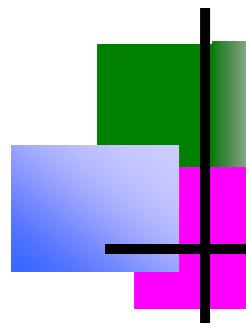
---



**NETFLIX**



***“House of Cards” is one of the first major test cases of this Big Data-driven creative strategy. For almost a year, Netflix executives have told us that their detailed knowledge of Netflix subscriber viewing preferences clinched their decision to license a remake of the popular and critically well regarded 1990 BBC miniseries. Netflix’s data indicated that the same subscribers who loved the original BBC production also gobbled down movies starring Kevin Spacey or directed by David Fincher. Therefore, concluded Netflix executives, a remake of the BBC drama with Spacey and Fincher attached was a no-brainer, to the point that the company committed \$100 million for two 13-episode seasons.***



---

# Did it work?



**David Armano**  
@armano



Just started. So far. Awesome.  
#houseofcards #GetGlue  
[getglue.com/tv\\_shows/house...](http://getglue.com/tv_shows/house...)

2/16/13, 7:02 PM

### House of Cards

Ruthless Congressman Francis Underwood and his ambitious wife Claire will stop at nothing to ascend the ranks of power. This wicked political drama slithers through the back halls of greed...



**The Studio Executive**  
@studioexec1

Watching #HouseofCards. Kevin Spacey's utterly brilliant as the slimy conniving politician with the quick wit. The man has such range.

2/24/13, 12:15 PM



**The Atlantic**  
@TheAtlantic



The Real History Behind the Politics of #HouseOfCards  
[theatlantic.com/politics/2013/02/the-real-history-behind-the-politics-of-house-of-cards/339577/](http://theatlantic.com/politics/2013/02/the-real-history-behind-the-politics-of-house-of-cards/339577/)

2/22/13, 5:00 AM



**Beau Willimon**  
@BeauWillimon



Wow- According to IMDB at least, #HouseofCards "Most Popular TV Show in the World" right now.  
THANK YOU FANS  
[@netflix">webpronews.com/house-of-cards... @netflix](http://webpronews.com/house-of-cards...)

2/15/13, 3:35 PM

12 RETWEETS 10 FAVORITES



**AnikaChapin**  
@AnikaChapin

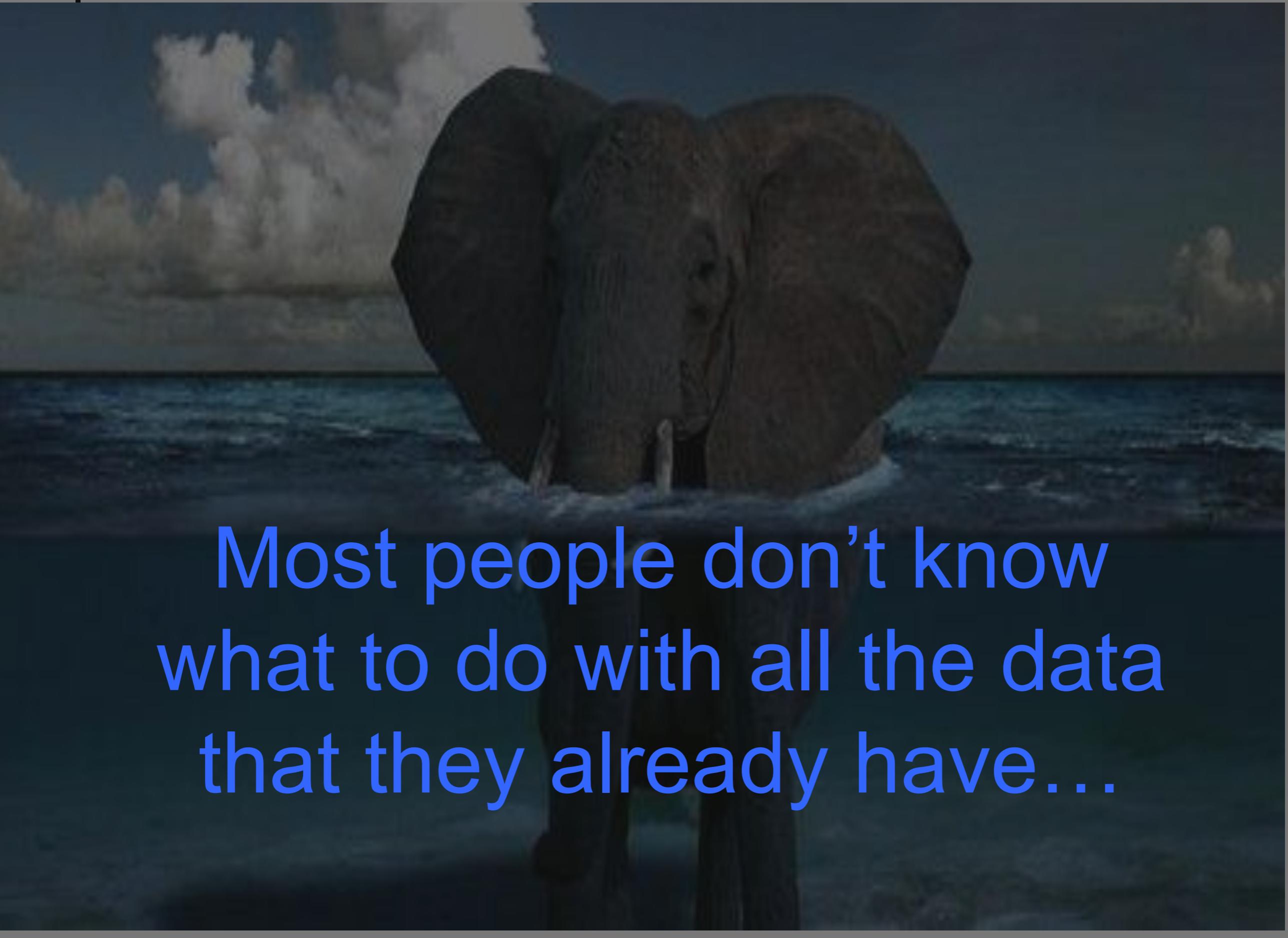
I've finished #HouseofCards and I don't know what to do with myself now. Maybe I'll start evilly manipulating people to fill the void.

2/24/13, 6:51 AM





*How do you  
mine (and mind)  
all this data?*



Most people don't know  
what to do with all the data  
that they already have...

A photograph of a young tree with green leaves growing out of a large, light-colored ceramic pot. The pot sits on a dark, textured surface. In the background, there's a blurred view of a landscape with hills and a body of water under a cloudy sky.

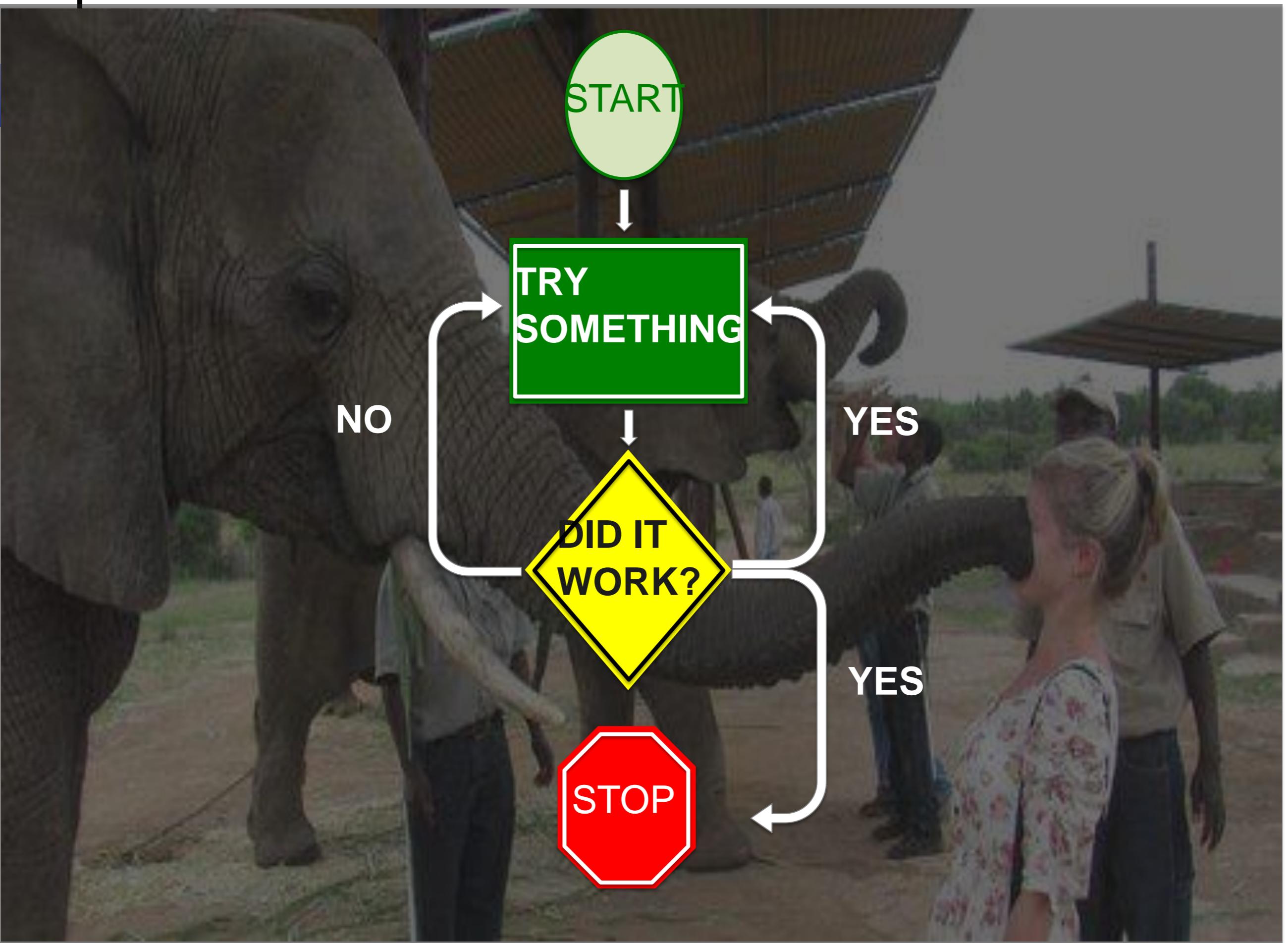
*Get Big*

by starting

*small*



Focus on  
*Business Impact.*

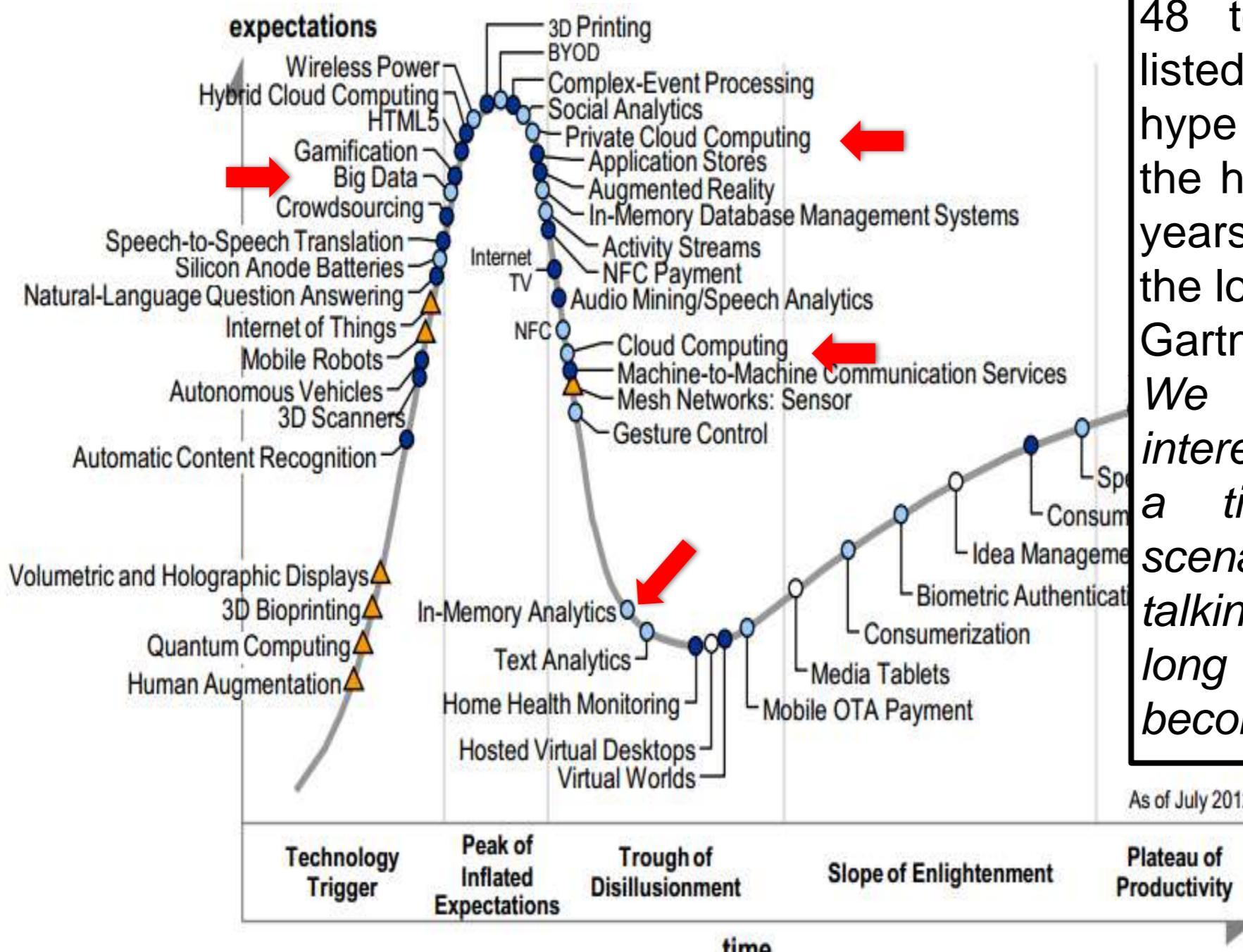




**Big Data isn't *big*,  
if you *know* how to  
*use it.***

# **SMART DATA**

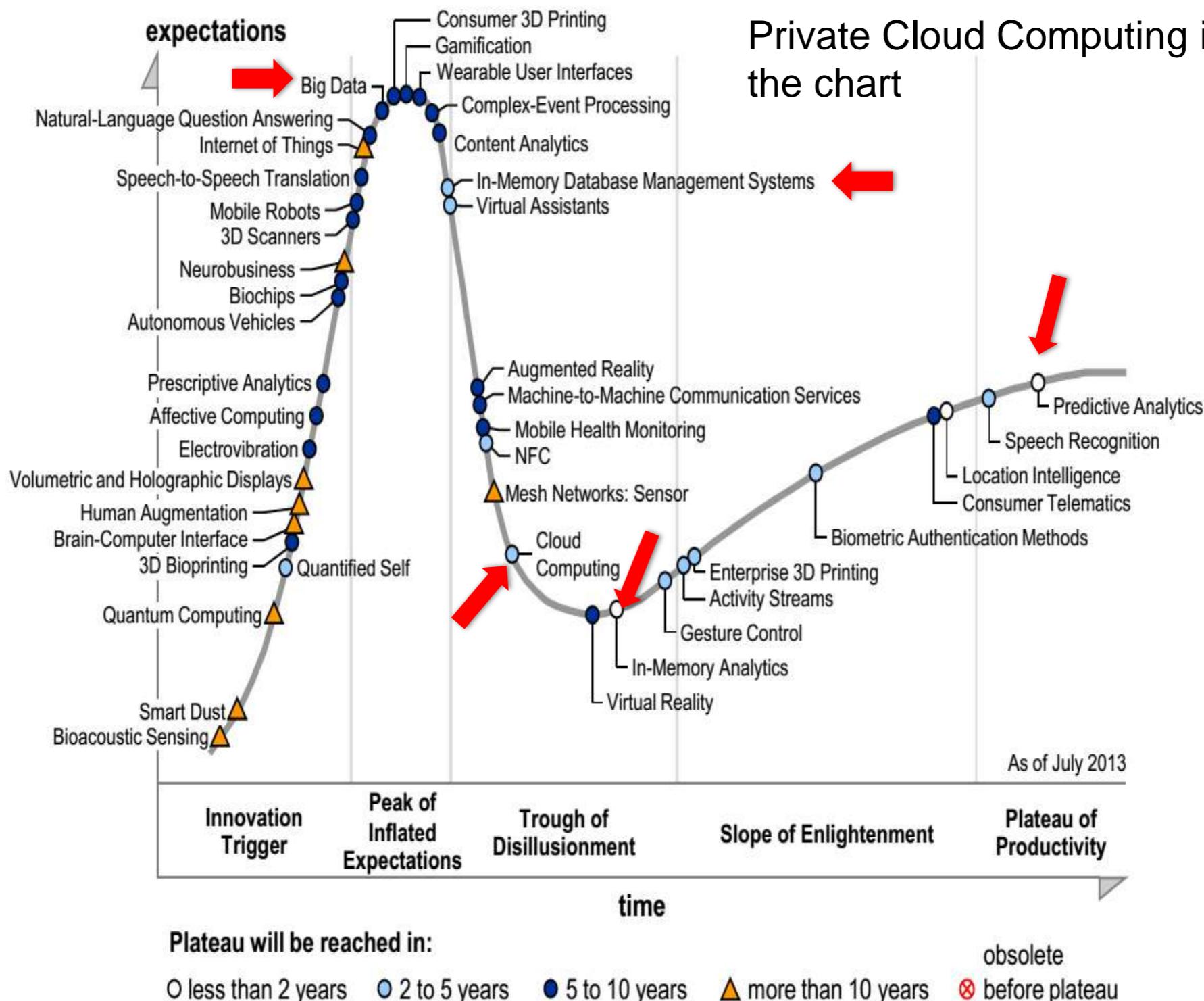
# Emerging Technologies Hype Cycle 2012



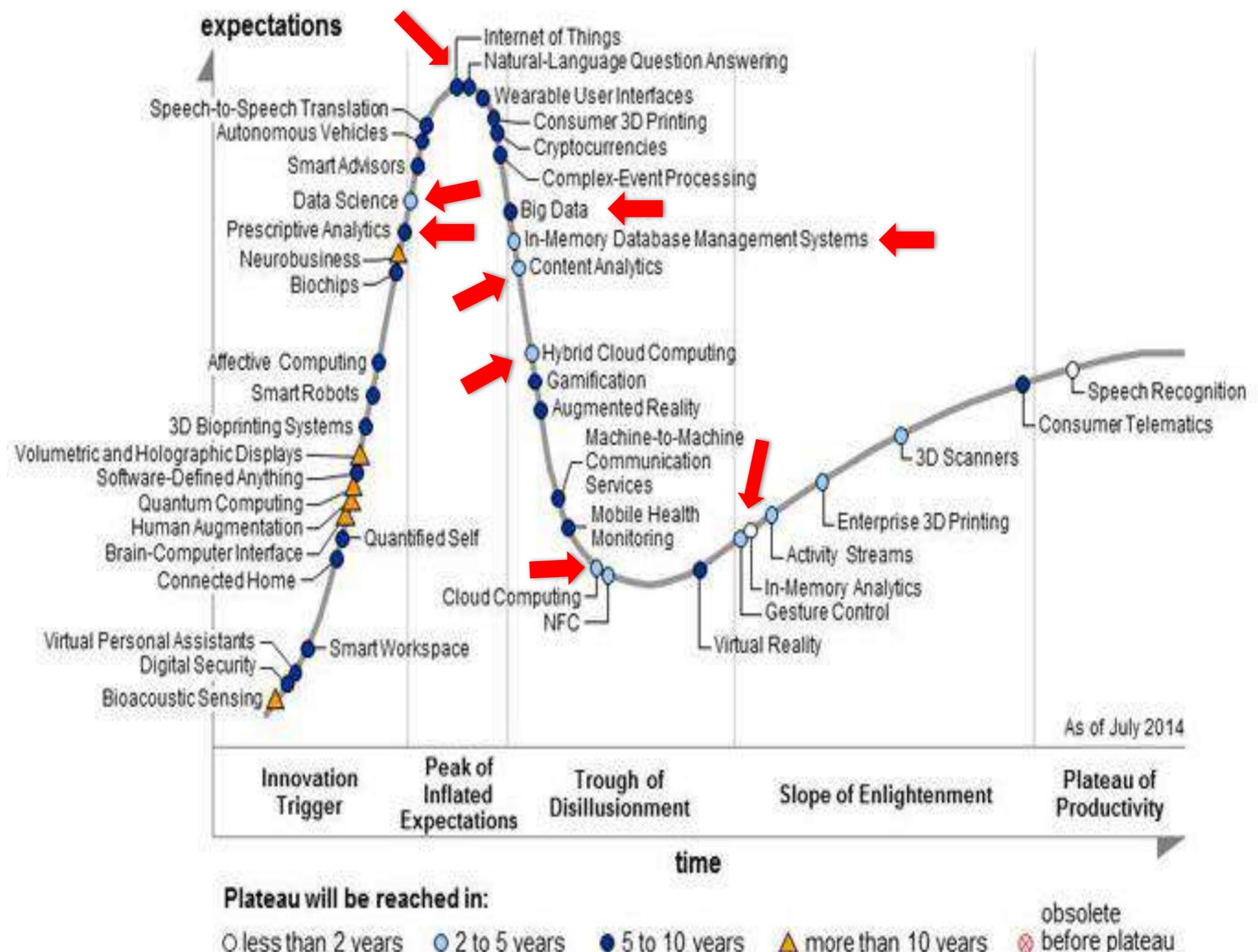
48 technologies are listed in this year's hype cycle which is the highest in last ten years. Year 2008 was the lowest (27). Gartner Says in 2012: *We are at an interesting moment — a time when the scenarios we've been talking about for a long time are almost becoming reality.*

As of July 2012

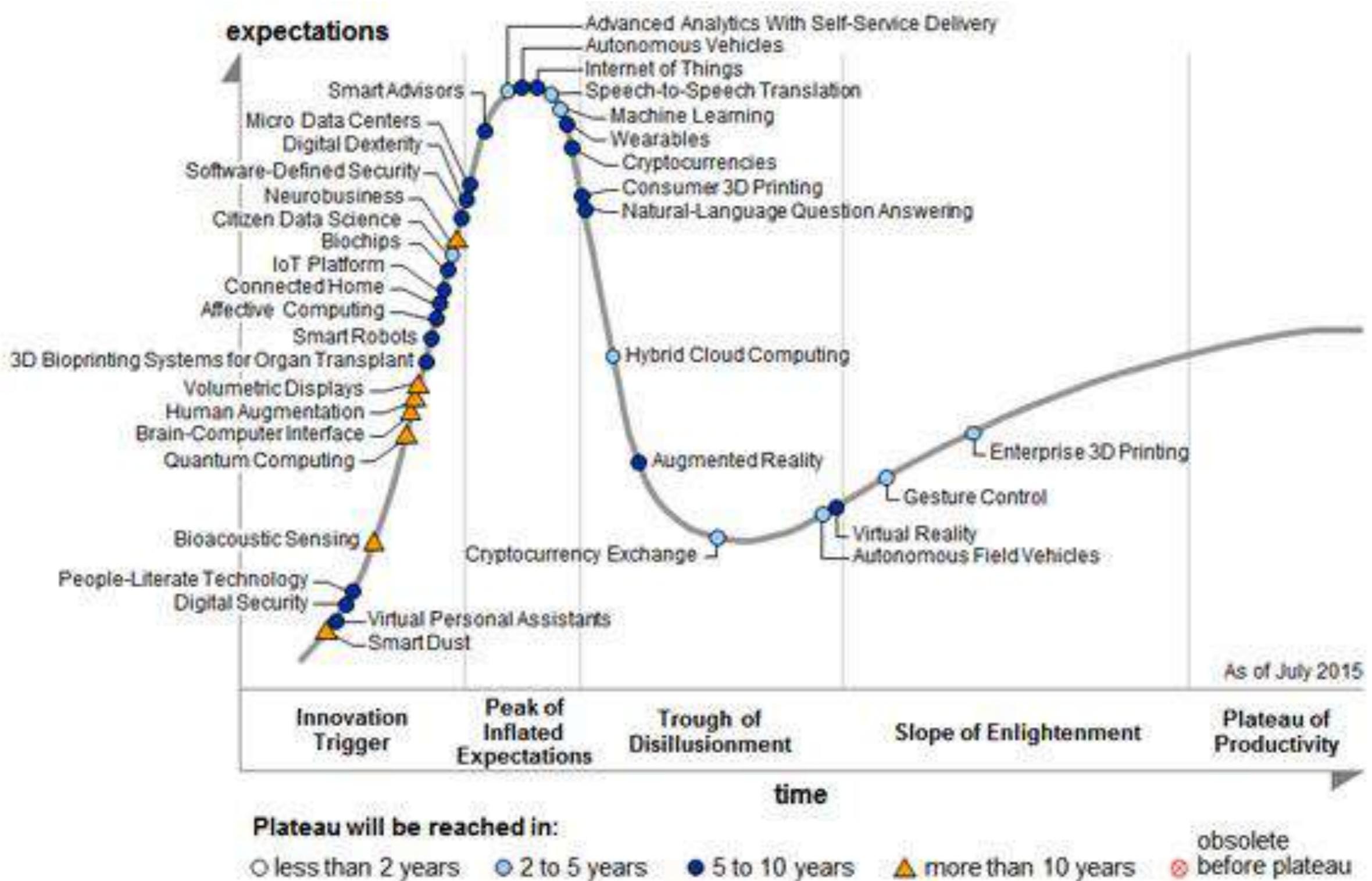
# Emerging Technologies Hype Cycle 2013



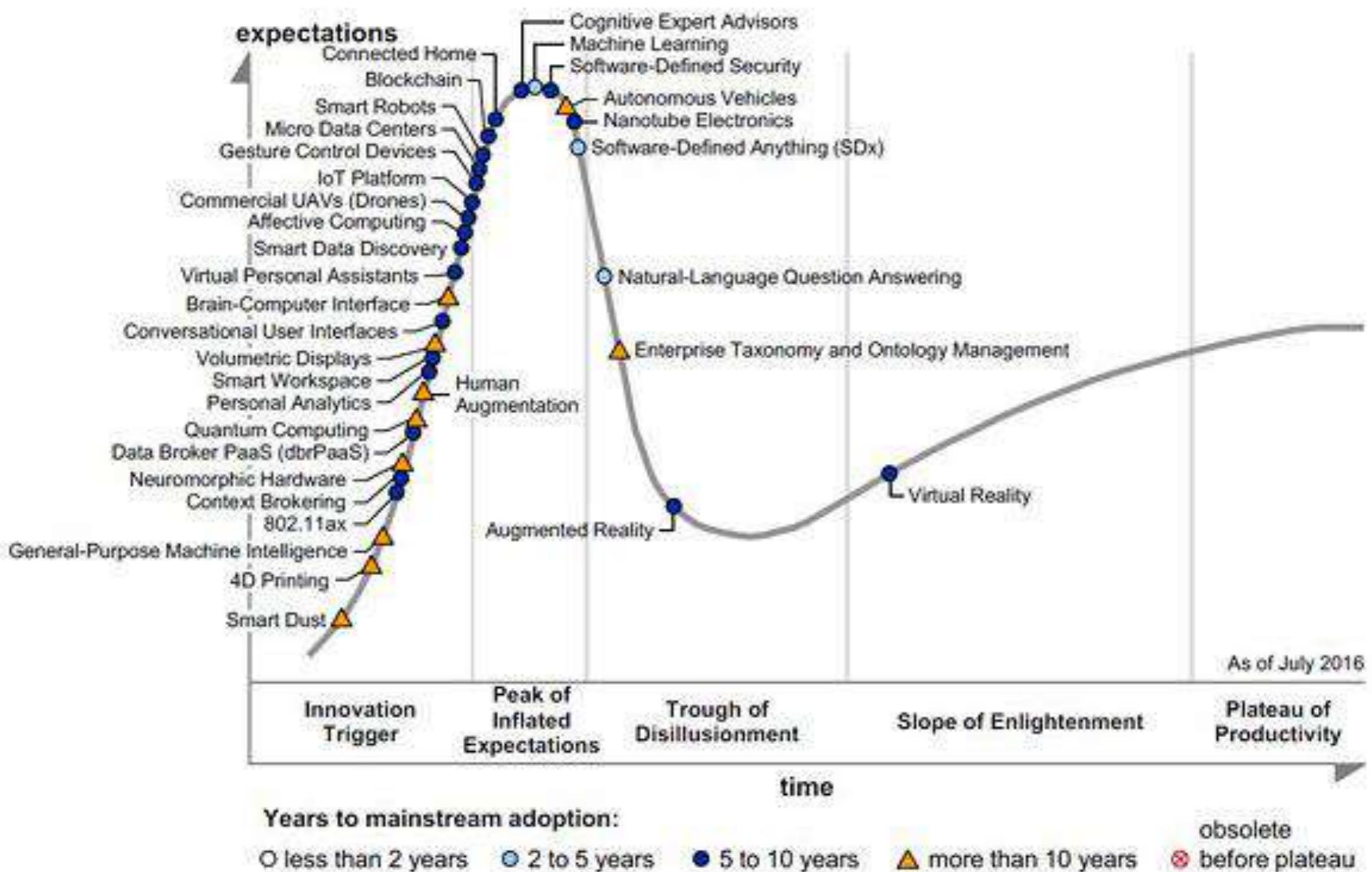
# Emerging Technologies Hype Cycle 2014

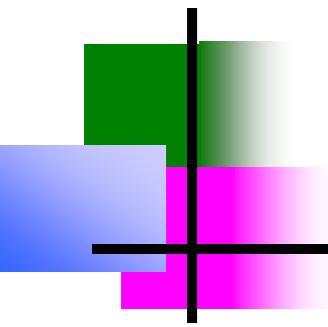


# Emerging Technologies Hype Cycle 2015



# Emerging Technologies Hype Cycle 2016





# What's Big Data?

---

No single definition; here is from Wikipedia:

- **Big data** is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications.
  - The challenges include **capture, curation, storage, search, sharing, transfer, analysis, and visualization**.
- The trend to larger data sets is due to the additional information derivable from analysis of a single large set of related data, as compared to separate smaller sets with the same total amount of data, allowing correlations to be found to "spot business trends, determine quality of research, prevent diseases, link legal citations, combat crime, and determine real-time roadway traffic conditions."

# Big data world then



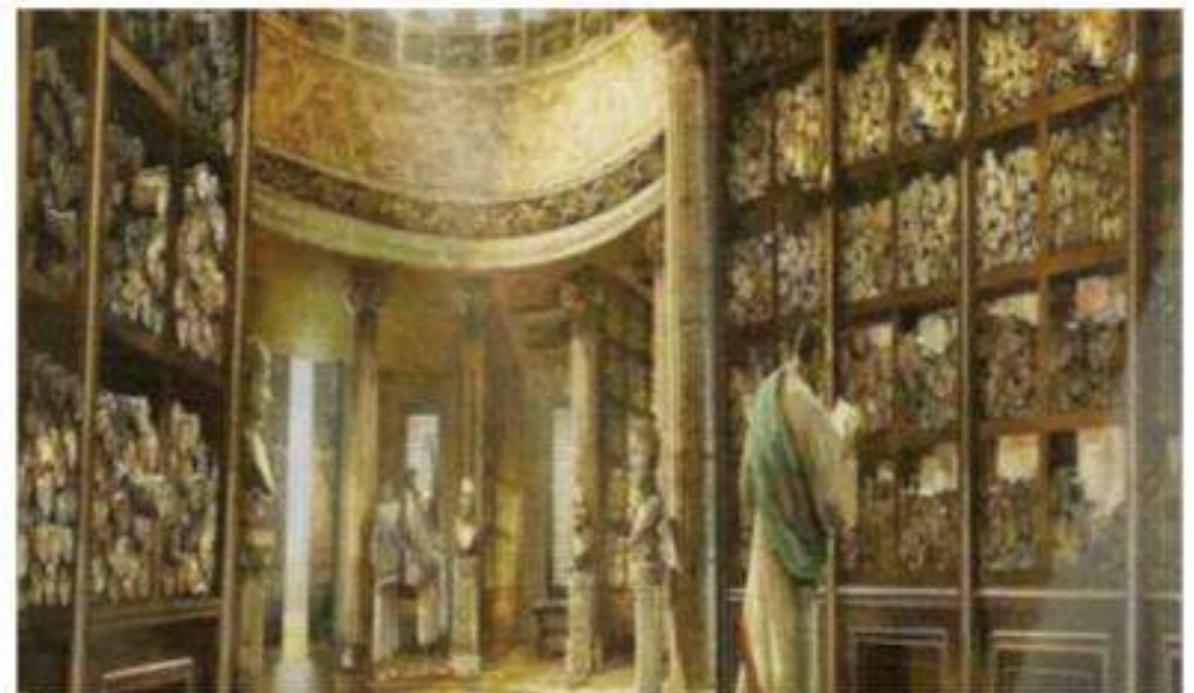
# Big data world then

“Big” is a Relative Term  
(Context dependent)  
IBM 5MB Hard Drive 1956



# Big data world then

- The library at Alexandria was the center of data in the ancient world
- Created in 300 BC by Ptolemy II of Egypt
- People from around the known world would congregate and spend their entire lives studying



**The ‘Data Lake’ of Antiquity**

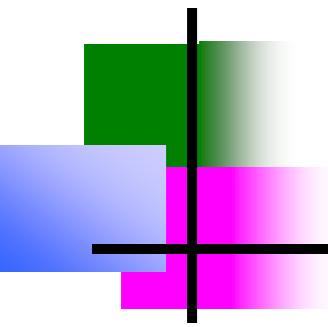
[www.extentia.com](http://www.extentia.com)

# Big data world then



**Two men operating a mainframe computer, circa 1960. It's amazing how today's smartphone holds so much more data than this huge 1960's relic. (Photo by Pictorial Parade/Archive Photos)**

Big Data Analytics by Vikram Neerugatti



# Big Data Statistics 2020

---

- Every person will generate 1.7 megabytes in just a second.
- Facebook has gained around 2.7 billion active monthly users and generates 4 petabytes of data per day
- Facebook stated that 3.14 billion people were using at least one of the company's core products (Facebook, WhatsApp, Instagram, or Messenger) each month.
- YouTube currently counts 2 billion monthly active users and **500 hours of content are uploaded to the platform every minute**
- Amazon has 150 million mobile users
- Twitter users send more than 540,000 tweets every minute.
- Over 2.5 quintillion bytes of data is generated worldwide every day.
- The amount of global data sphere subject to data analysis will grow to 5.2 zettabytes by 2025.
- By 2021, insight-driven businesses are predicted to take \$1.8 trillion annually from their less-informed peers.
- Data-driven organizations are 23 times more likely to acquire customers than their peers.
- Businesses are spending \$187 billion on big data and analytics in 2019.
- 91.6% of firms worldwide confirm an increased pace in investment in big data in 2019.

# Describing Data Size

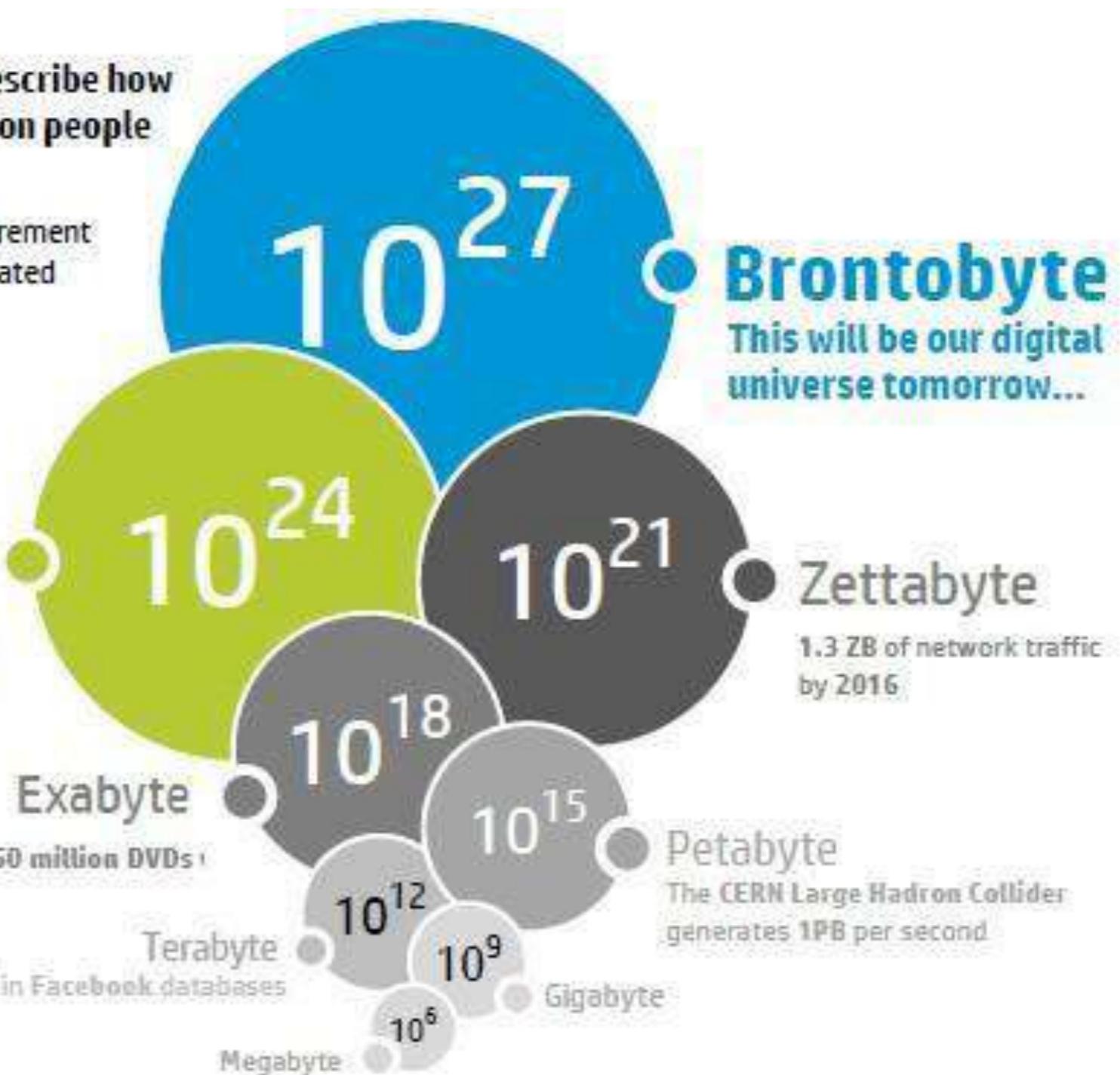
Today data scientist uses **Yottabytes** to describe how much government data the NSA or FBI have on people altogether.

In the near future, **Brontobyte** will be the measurement to describe the type of sensor data that will be generated from the IoT (Internet of Things)

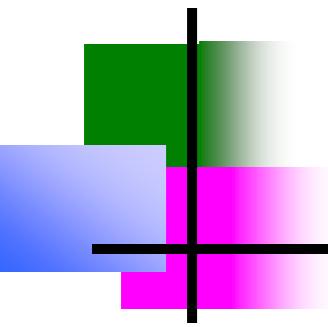
**Yottabyte**  
This is our digital universe today  
= 250 trillion of DVDs

1 EB of data is created on the internet each day = 250 million DVDs!

500TB of new data per day are ingested in Facebook databases



<https://twitter.com/paolopisani/>

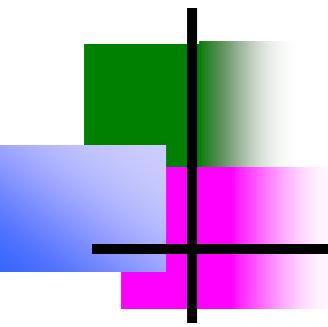


# Big Data

---

## WHY??

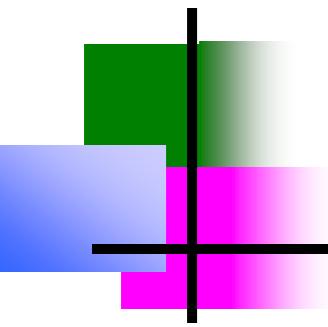
- “Retrieval of information”.
- “Need of past history”
- “Science and research”
- “Simulation and modeling”
- “Forecasting”
- “Increased population”
- “.....Many more....”



# Big Data

---

- *Big data is a term applied to a new generation of software, applications, and system and storage architecture.*
- *It designed to provide business value from unstructured data.*
- *Big data sets require advanced tools, software, and systems to capture, store, manage, and analyze the data sets,*
- *All in a timeframe Big data preserves the intrinsic value of the data.*
- *Big data is now applied more broadly to cover commercial environments.*



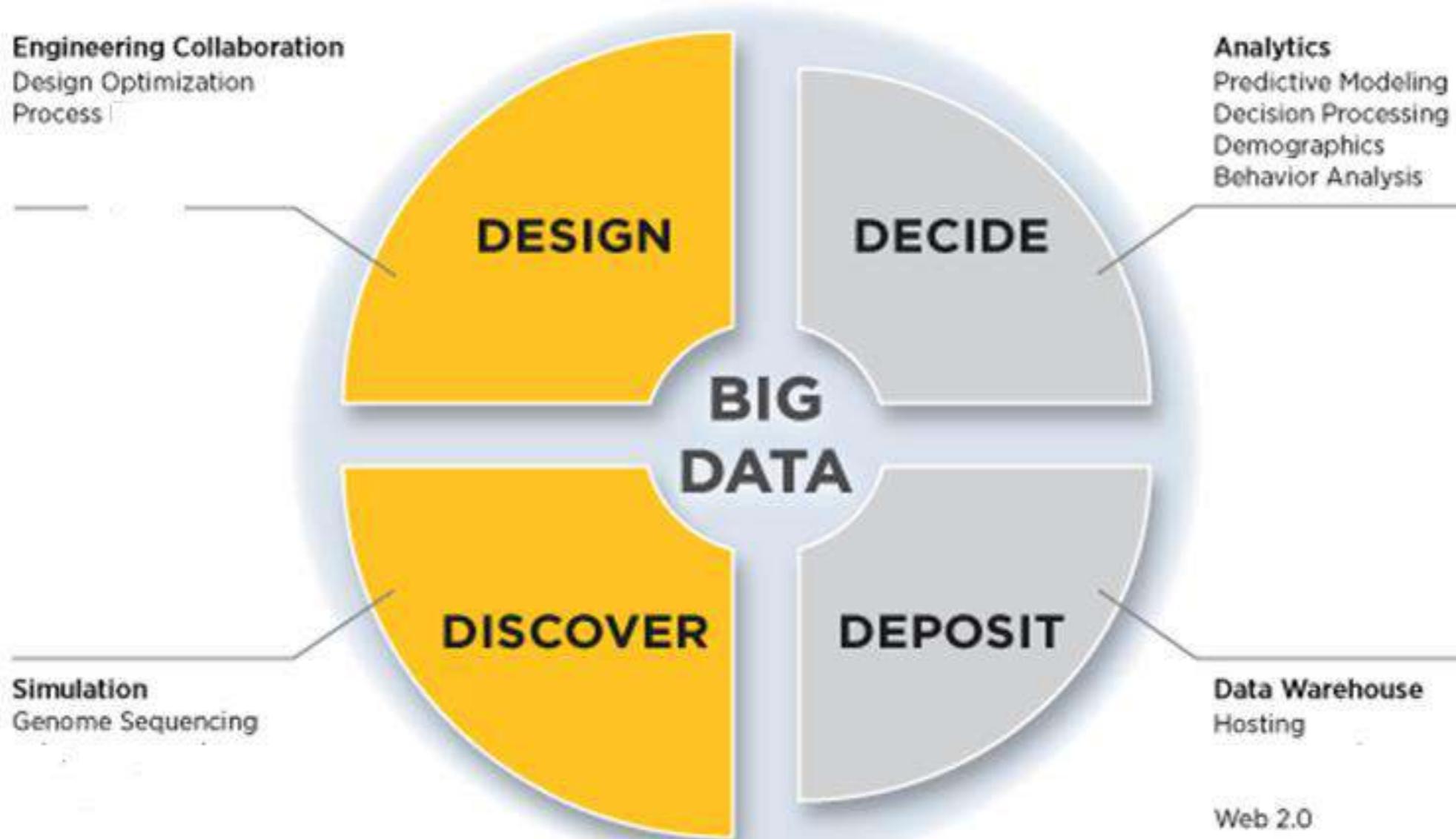
# Big data

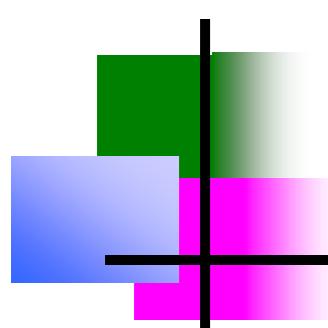
---

- *Four distinct applications segments comprise the big data market.*
- *each with varying levels of need for performance and scalability.*
- *The four big data segments are:*
  - *1) Design (engineering collaboration)*
  - *2) Discover (core simulation – supplanting physical experimentation)*
  - *3) Decide (analytics).*
  - *4) Deposit (Web 2.0 and data warehousing)*

# Big Data

## Big Data Application Segments





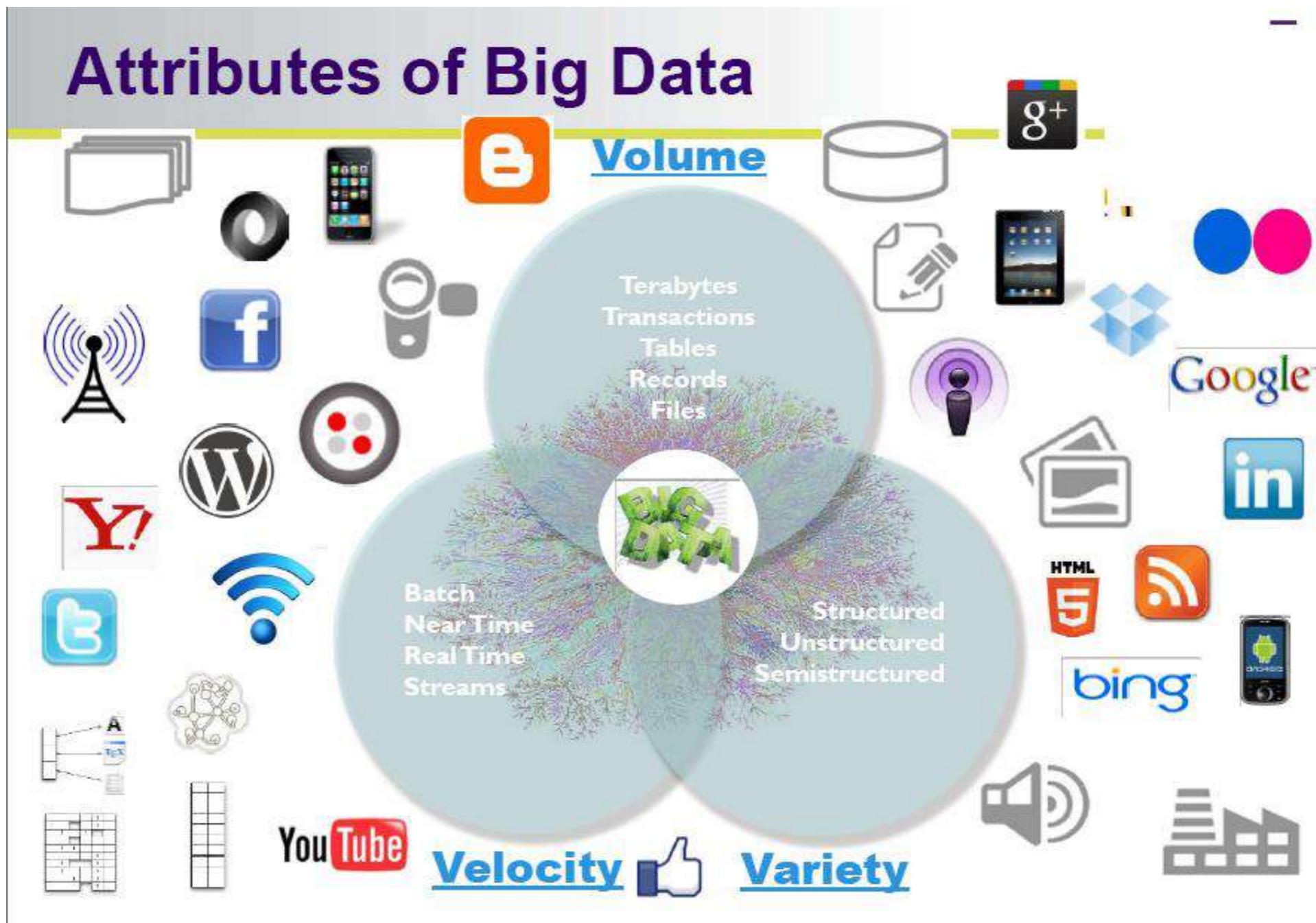
# Big Data

## “Data Driven” Web 2.0 onwards.

---



# Big Data



# Big Data

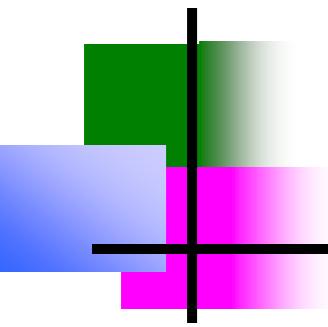
Enterprise + Big Data = Big Opportunity



# Big Data

## The Big Data Opportunity

<b>Financial Services</b> 	<b>Healthcare</b> 
<b>Retail</b> 	<b>Web/Social/Mobile</b> 
<b>Manufacturing</b> 	<b>Government</b> 



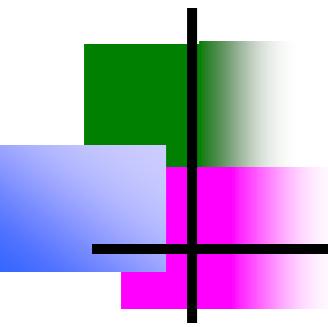
# Big Data

---

## Ten Common Big Data Problems

---

1. Modeling true risk
2. Customer churn analysis
3. Recommendation engine
4. Ad targeting
5. Transaction analysis
6. Analyzing network data to predict failure
7. Threat analysis
8. Trade surveillance
9. Search quality
10. Data “sandbox”



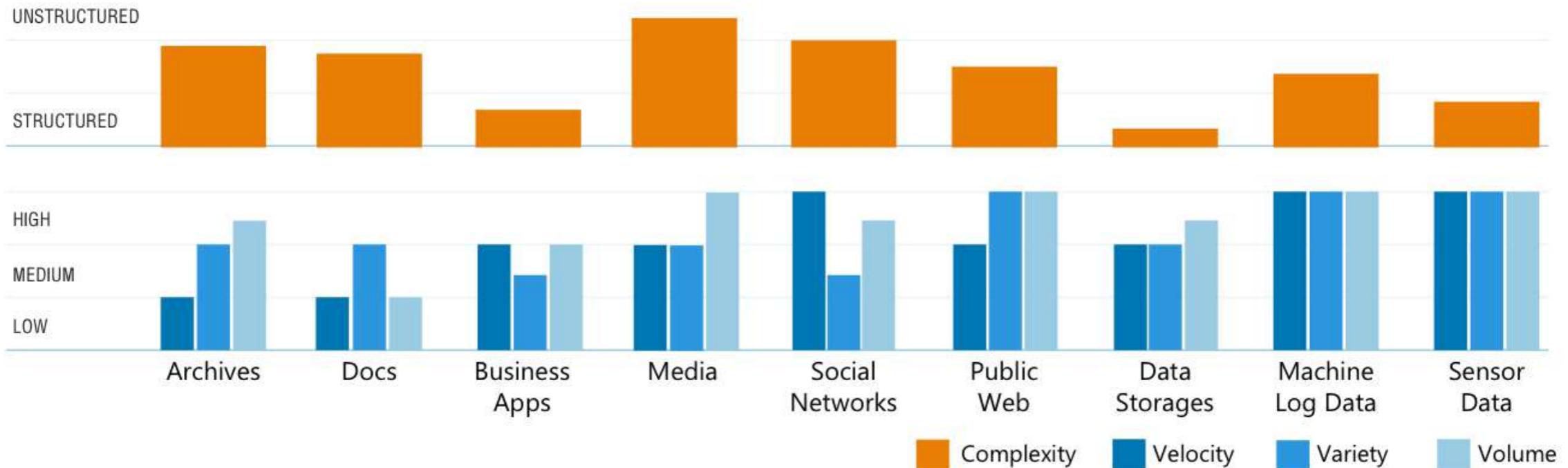
# Big Data Challenges

---

## Top 5 Big Data Challenges

1. Deciding what data is relevant
  2. Cost of technology infrastructure
  3. Lack of skills to analyze the data
  4. Lack of skills to manage big data projects
  5. Lack of business support
-

# Big Data challenges



## Archives

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



## Media

Images, video, audio etc.



## Data Storages

RDBMS, NoSQL, Hadoop, file systems etc.



## Docs

XLS, PDF, CSV, HTML, JSON etc.



## Social Networks

Twitter, Facebook, Google+, LinkedIn etc.



## Machine Log Data

Application logs, event logs, server data, CDRs, clickstream data etc.



## Business Apps

CRM, ERP systems, HR, project management etc.



## Public Web

Wikipedia, news, weather, public finance etc



## Sensor Data

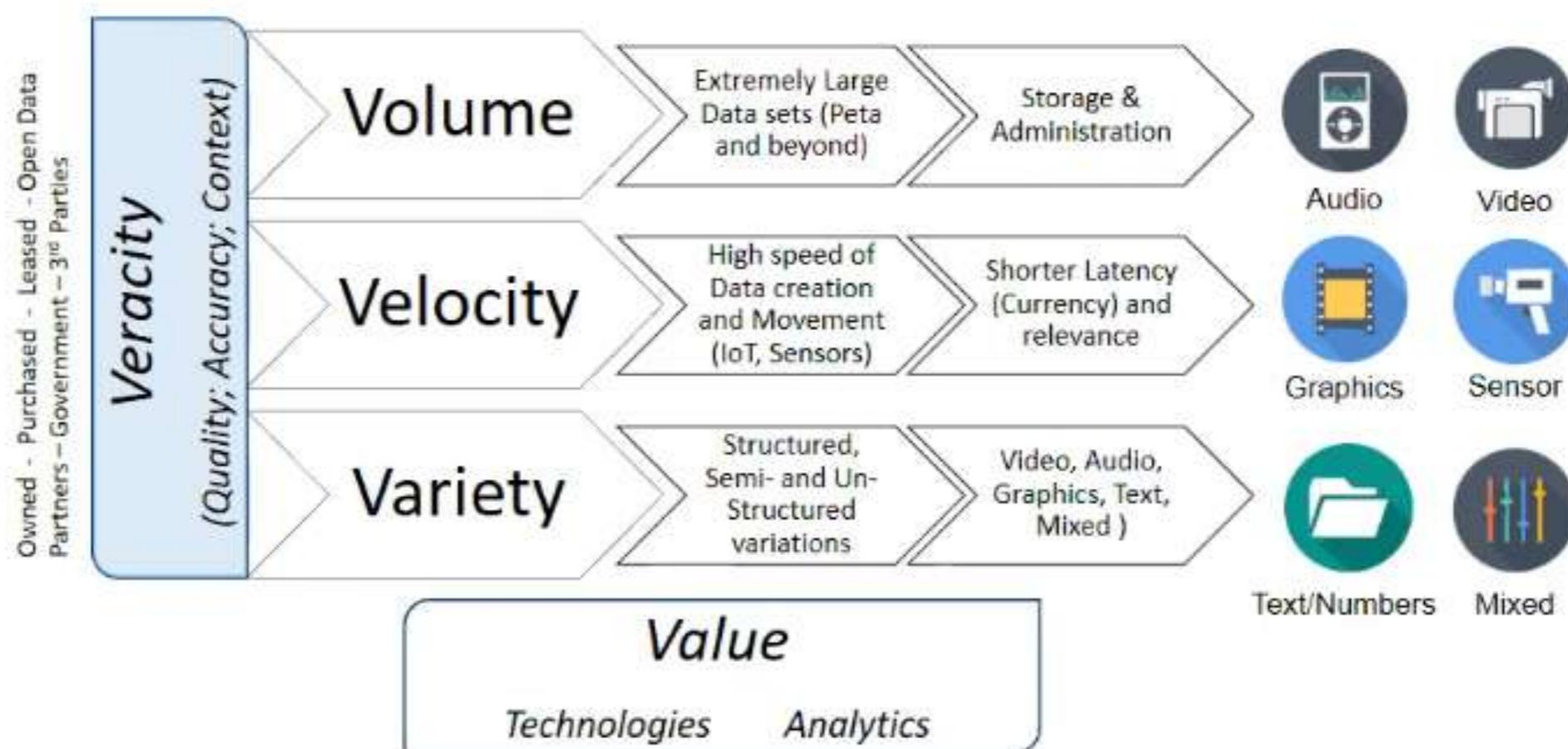
Smart electric meters, medical devices, car sensors, road cameras etc.

# Applications



# V's of Big data

Figure 3.3 : Detailed Characteristics of Big Data's 3+1+1 Vs and the Types and Categories of data. (The fifth V for Value is the focus of BDFAB)



Each DATA characteristic impacts the way Big Data Strategies and corresponding Solutions are designed, developed and used

# Sources of Big data



? TBs  
of data every  
day



12+  
TBs  
of tweet data  
every day

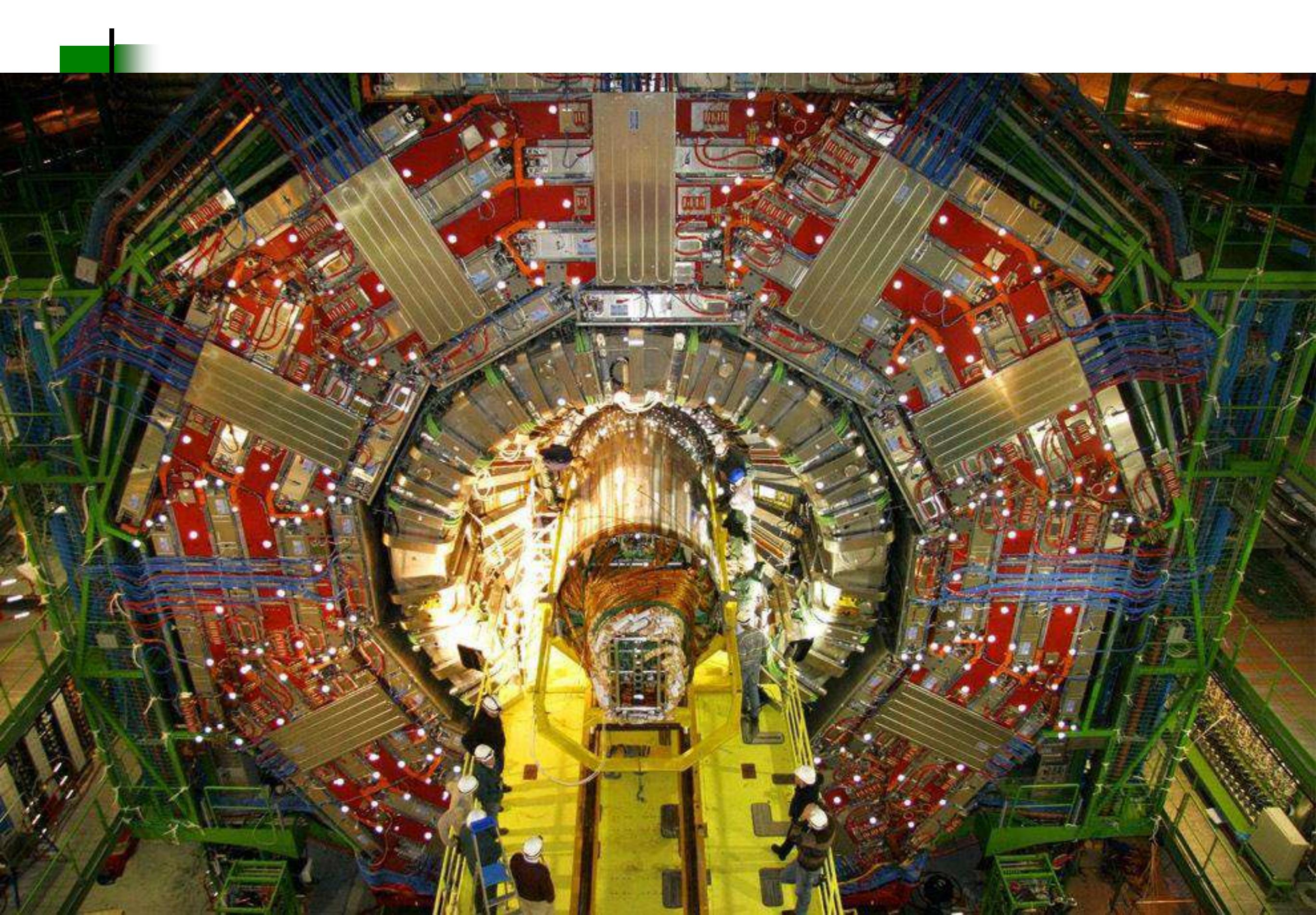
25+  
TBs  
of log  
data every  
day

76 million  
smart meters in  
2009... 200M by  
2014

30 billion  
RFID tags today  
(1.3B in 2005)



4.6  
billi  
on  
camera  
phones  
world  
wide  
  
100s  
of  
milli  
ons  
of  
GPS  
enabl  
ed  
bitio  
n  
devices  
sold  
annually  
people on  
the Web  
by end  
2011



# The Earthscope

- The Earthscope is the world's largest science project. Designed to track North America's geological evolution, this observatory records data over 3.8 million square miles, amassing 67 terabytes of data. It analyzes seismic slips in the San Andreas fault, sure, but also the plume of magma underneath Yellowstone and much, much more.

[http://www.msnbc.msn.com/id/44363598/ns/technology\\_and\\_science-future\\_of\\_technology/#.TmetOdQ-ul](http://www.msnbc.msn.com/id/44363598/ns/technology_and_science-future_of_technology/#.TmetOdQ-ul)



# Real-time/Fast Data



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



**Scientific instruments**  
(collecting all sorts of data)



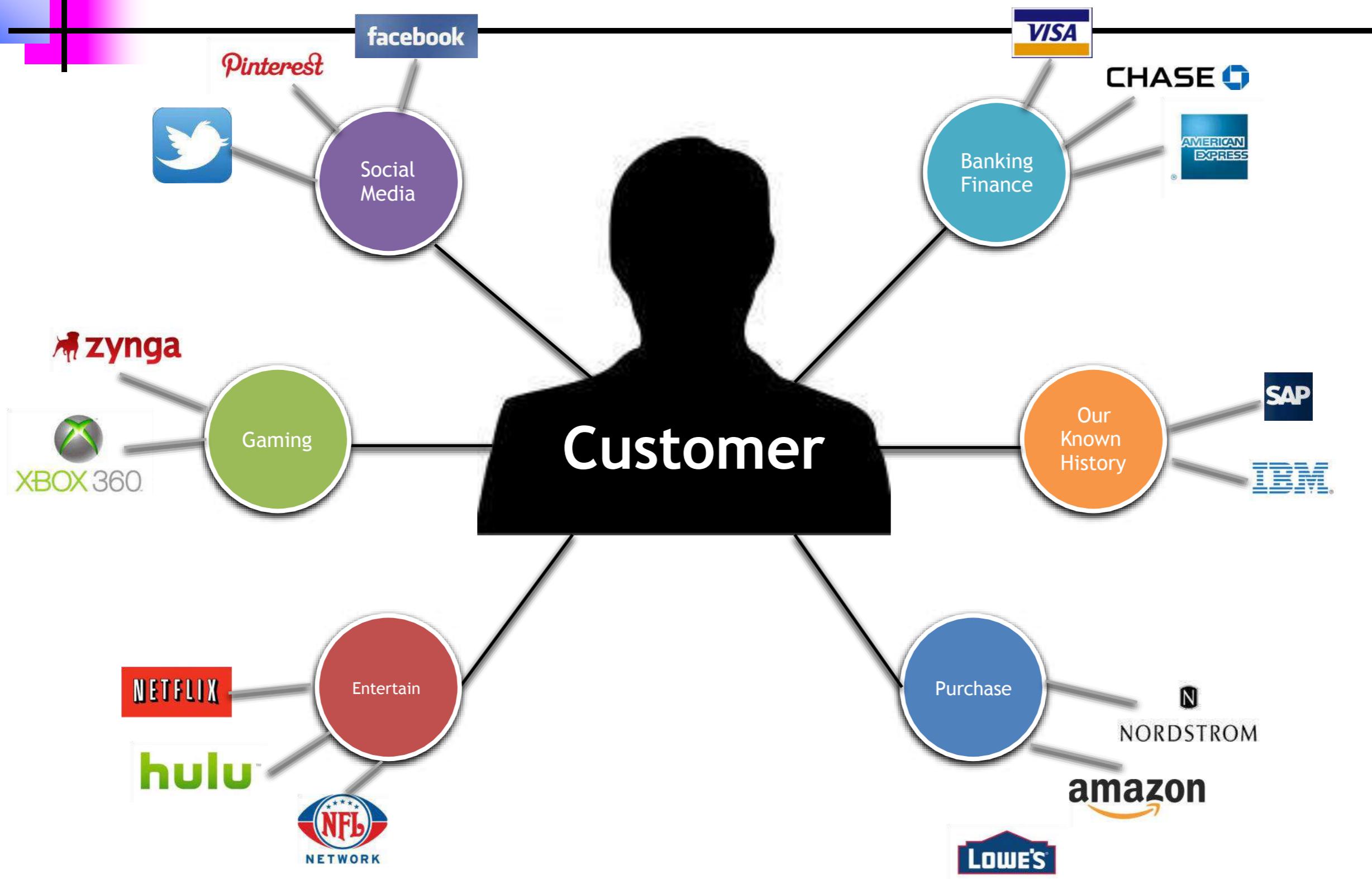
**Mobile devices**  
(tracking all objects all the time)

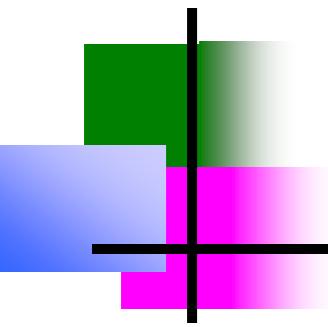


**Sensor technology and networks**  
(measuring all kinds of data)

- 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

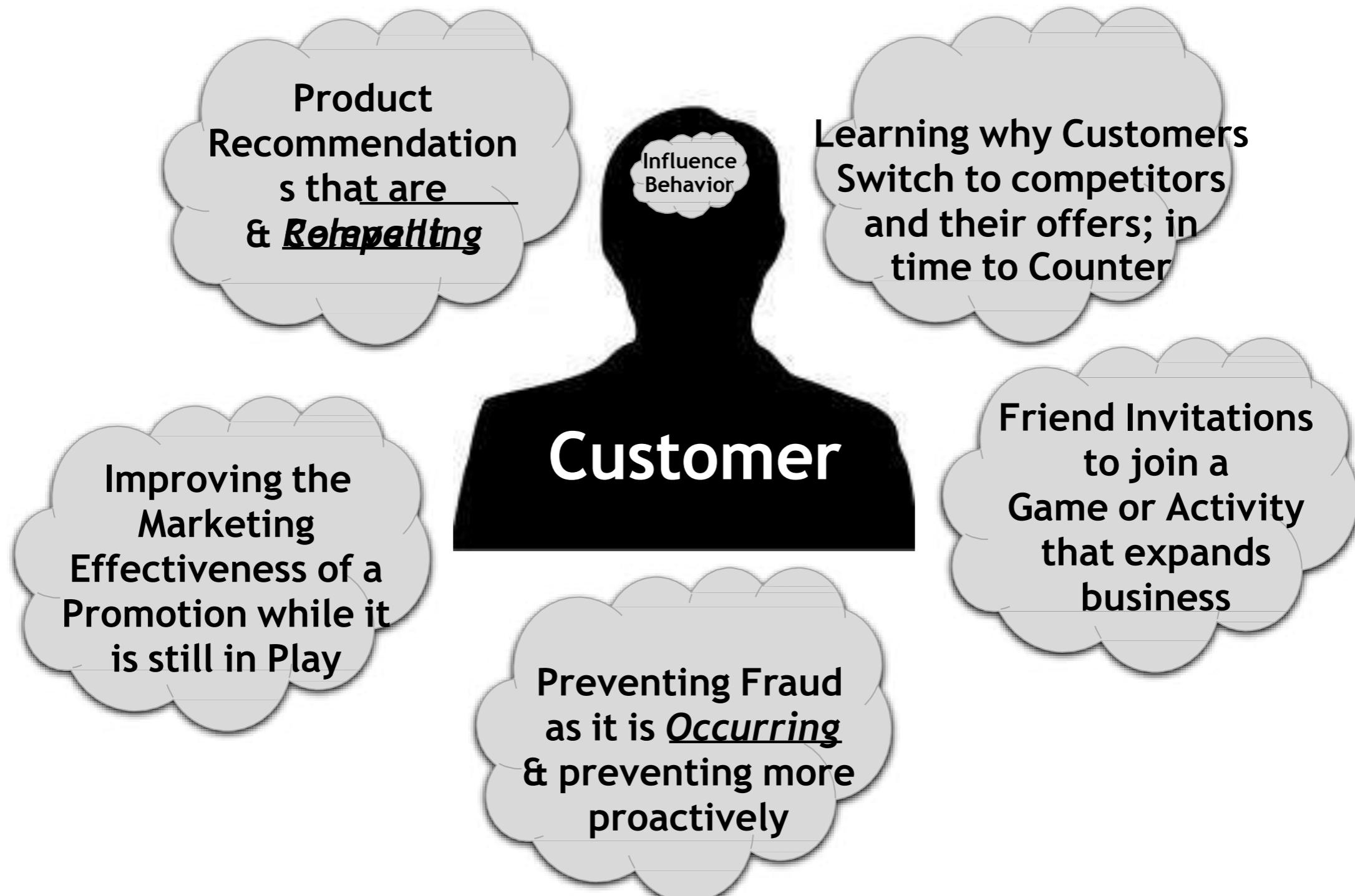
# A Single View to the Customer



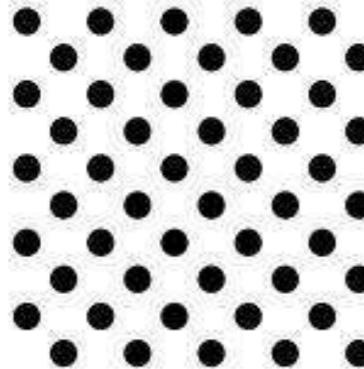
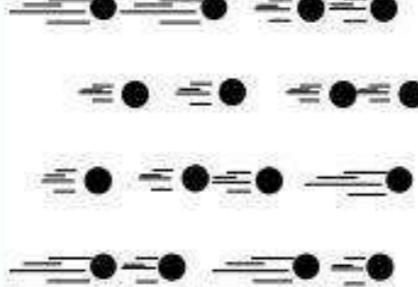
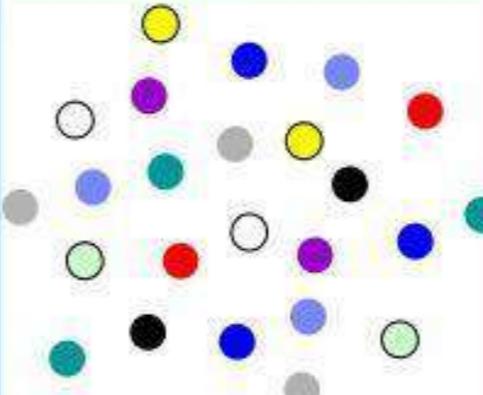
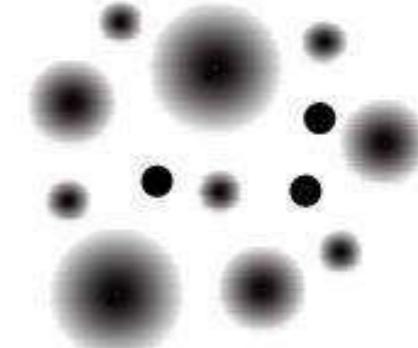


# Real-Time Analytics/Decision Requirement

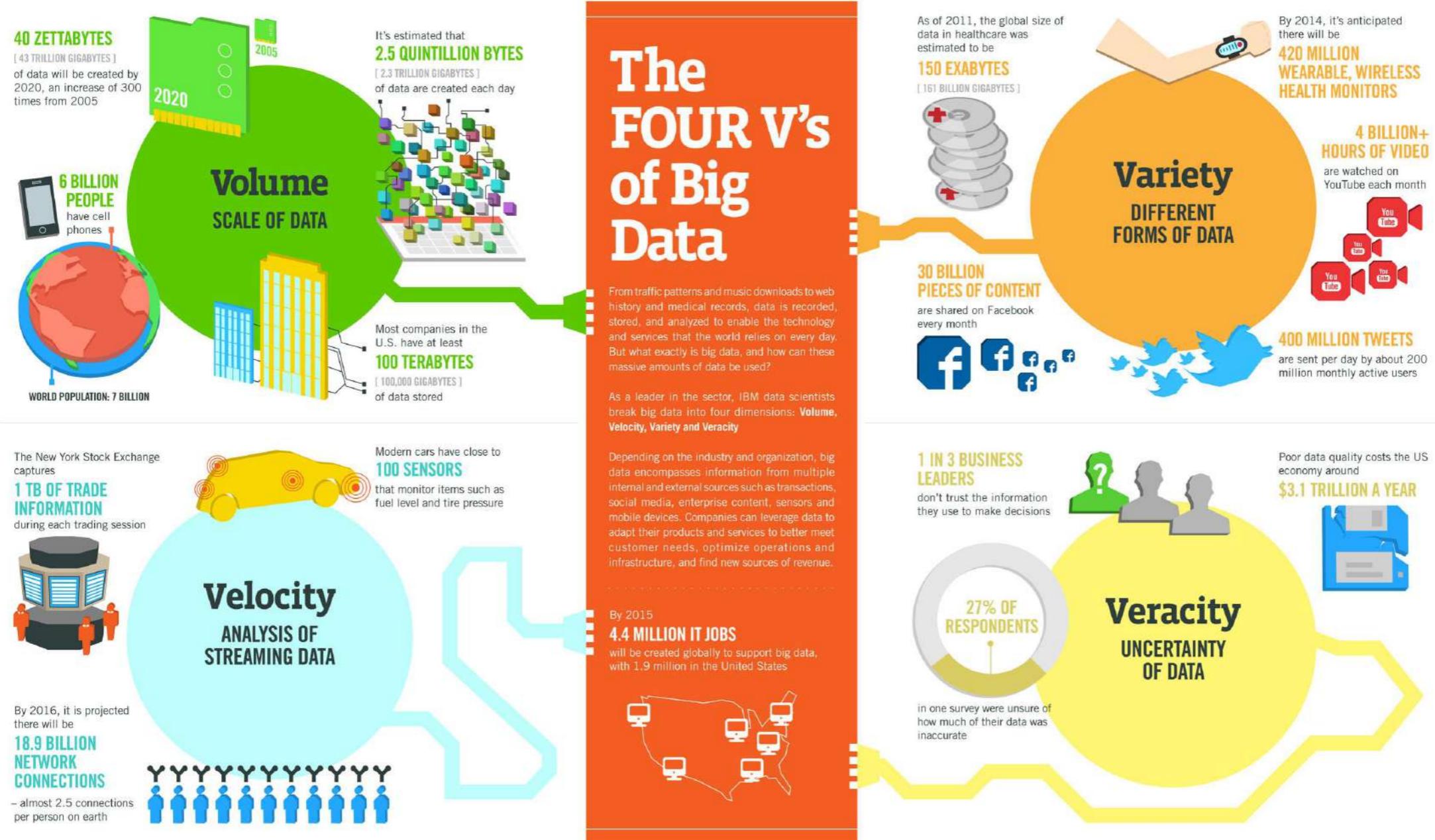
---



# Some Make it 4V's

Volume	Velocity	Variety	Veracity*
			
<b>Data at Rest</b>	<b>Data in Motion</b>	<b>Data in Many Forms</b>	<b>Data in Doubt</b>
Terabytes to exabytes of existing data to process	Streaming data, milliseconds to seconds to respond	Structured, unstructured, text, multimedia	Uncertainty due to data inconsistency & incompleteness, ambiguities, latency, deception, model approximations

# V's of Big data



IBM

# V's of Big data

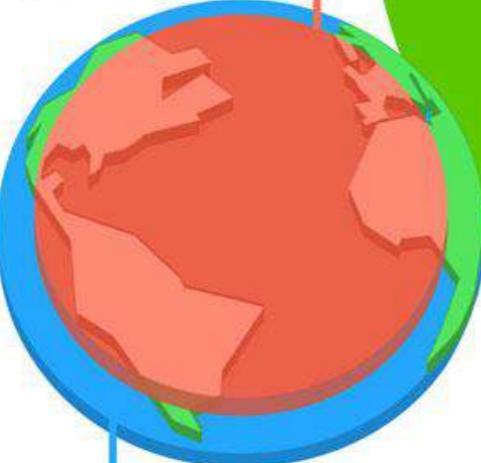
**40 ZETTABYTES**

[ 43 TRILLION GIGABYTES ]  
of data will be created by  
2020, an increase of 300  
times from 2005



**6 BILLION  
PEOPLE**

have cell  
phones



WORLD POPULATION: 7 BILLION



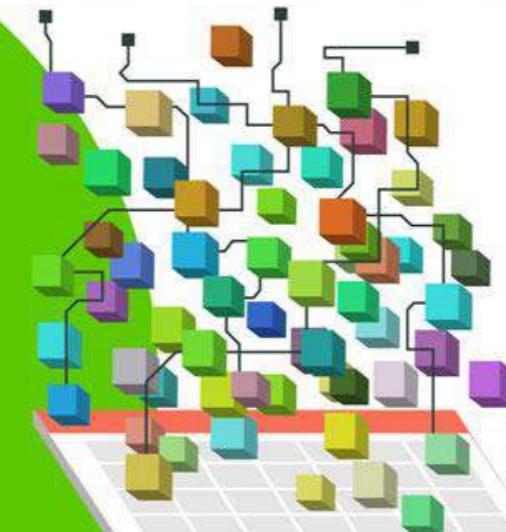
## Volume SCALE OF DATA



It's estimated that

**2.5 QUINTILLION BYTES**

[ 2.3 TRILLION GIGABYTES ]  
of data are created each day



Most companies in the  
U.S. have at least

**100 TERABYTES**

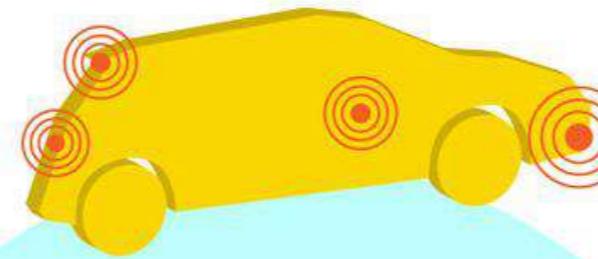
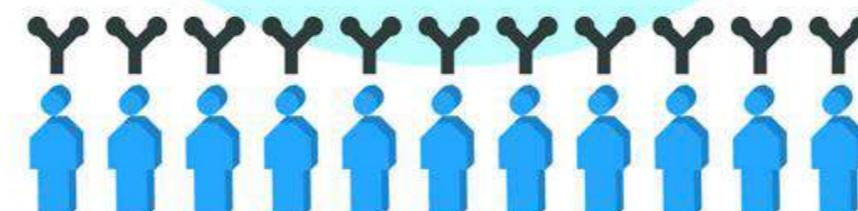
[ 100,000 GIGABYTES ]  
of data stored

# V's of Big data

The New York Stock Exchange captures  
**1 TB OF TRADE INFORMATION** during each trading session



By 2016, it is projected there will be  
**18.9 BILLION NETWORK CONNECTIONS**  
– almost 2.5 connections per person on earth



Modern cars have close to **100 SENSORS** that monitor items such as fuel level and tire pressure



## Velocity

ANALYSIS OF STREAMING DATA

v's of Big data  
420 million wearable,  
wireless health monitors  
4 billion hours of video are watched on YouTube each month  
400 million tweets are sent per day 200 million mo

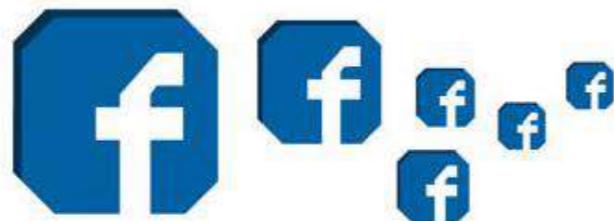
As of 2011, the global size of data in healthcare was estimated to be

**150 EXABYTES**  
[ 161 BILLION GIGABYTES ]



**30 BILLION PIECES OF CONTENT**

are shared on Facebook every month



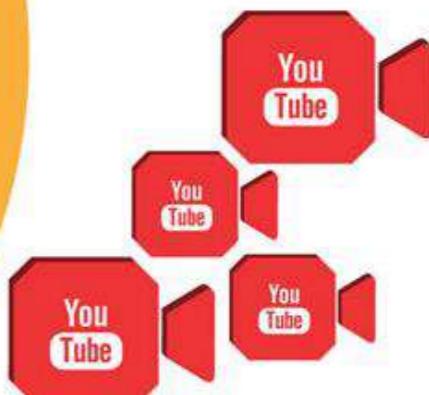
## Variety DIFFERENT FORMS OF DATA



By 2014, it's anticipated there will be

**420 MILLION WEARABLE, WIRELESS HEALTH MONITORS**

**4 BILLION+ HOURS OF VIDEO**  
are watched on YouTube each month



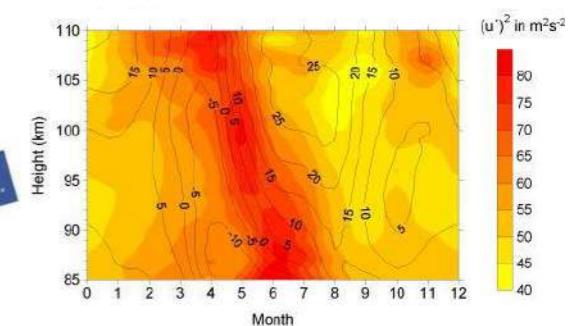
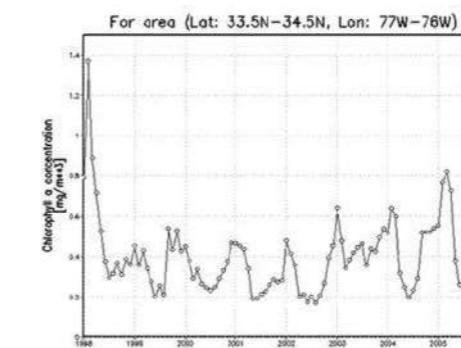
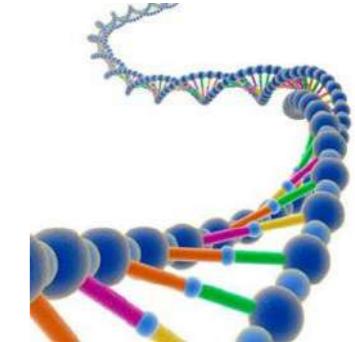
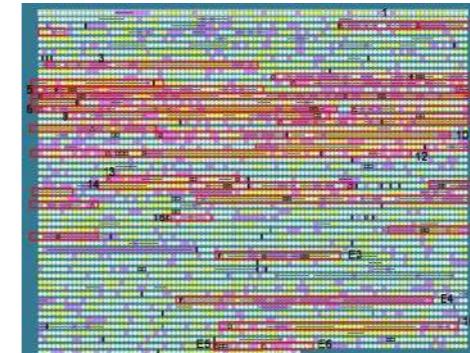
**400 MILLION TWEETS**

are sent per day by about 200 million monthly active users

# Variety (Complexity)

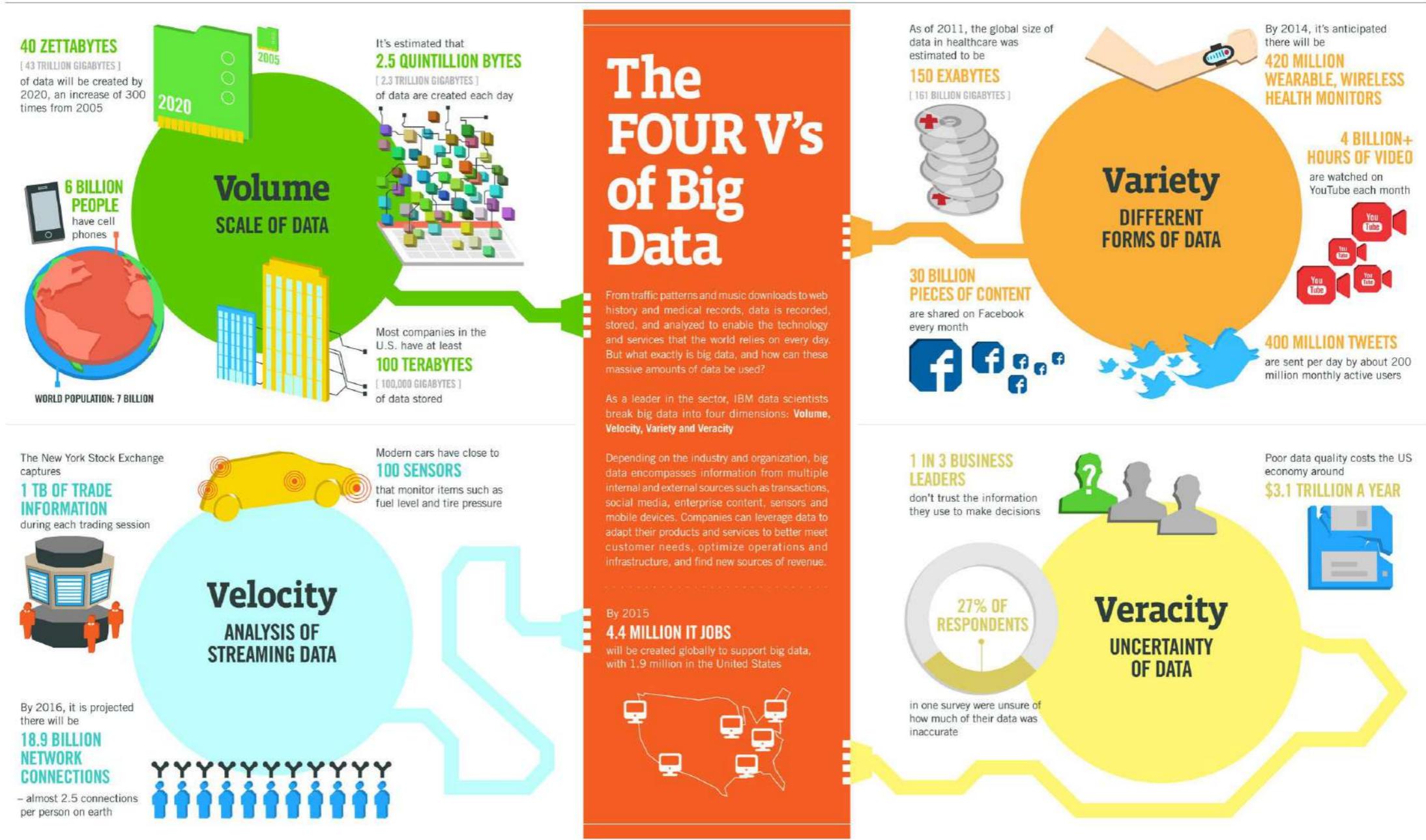
- Relational Data (Tables/Transaction/Legacy Data)
- Text Data (Web)
- Semi-structured Data (XML)
- Graph Data
  - Social Network, Semantic Web (RDF), ...
- Streaming Data
  - You can only scan the data once
- A single application can be generating/collecting many types of data

Big Public Data (online, weather, finance, etc)



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

# V's of Big data



# V's of Big data

## 1 IN 3 BUSINESS LEADERS

don't trust the information they use to make decisions



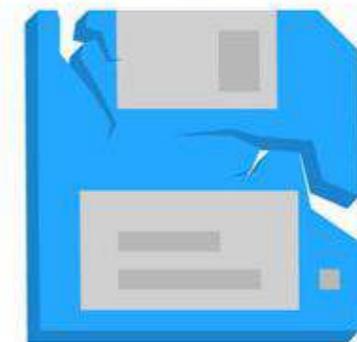
27% OF RESPONDENTS

in one survey were unsure of how much of their data was inaccurate

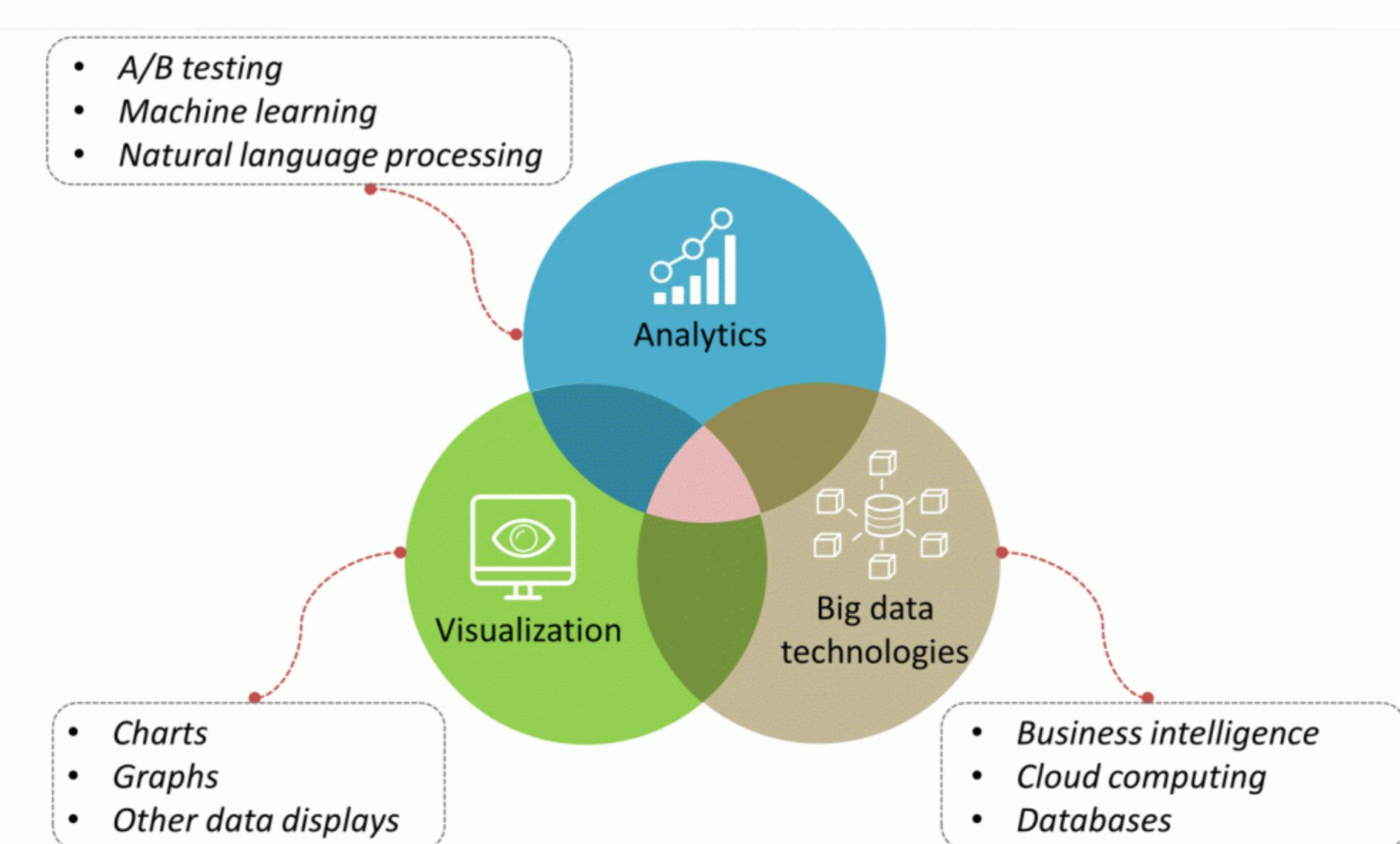
**Veracity**  
**UNCERTAINTY OF DATA**

Poor data quality costs the US economy around

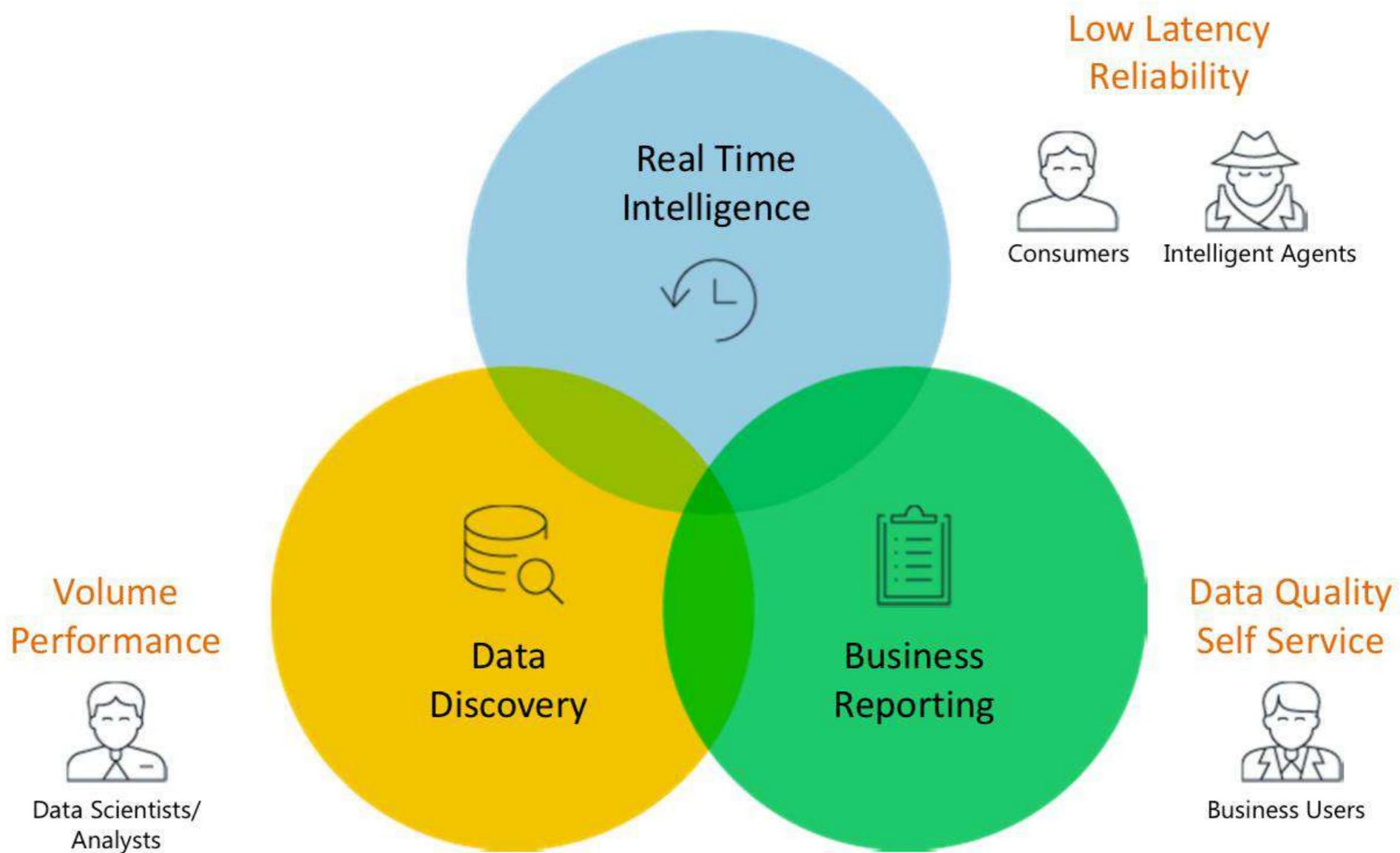
**\$3.1 TRILLION A YEAR**



# Technologies for Big data

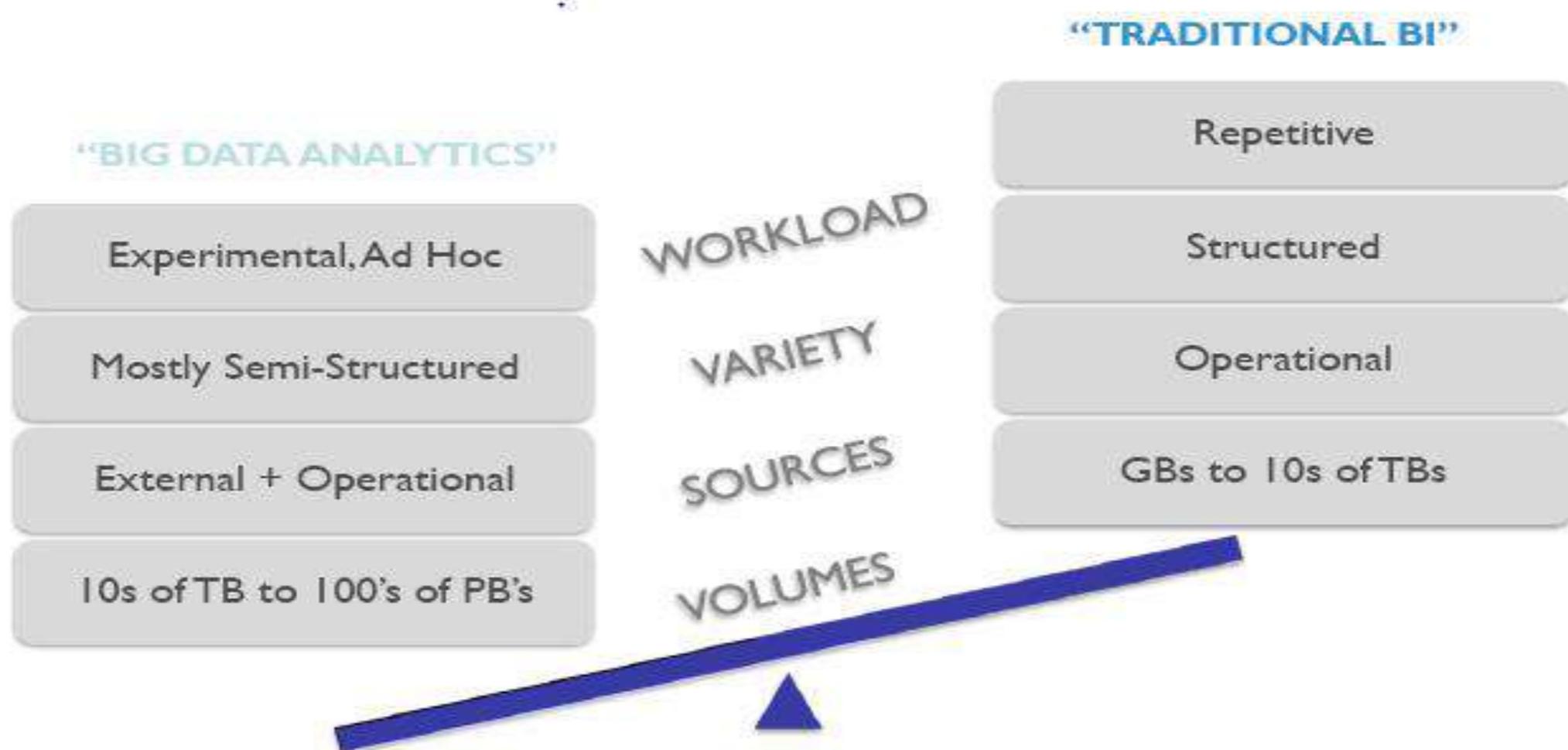


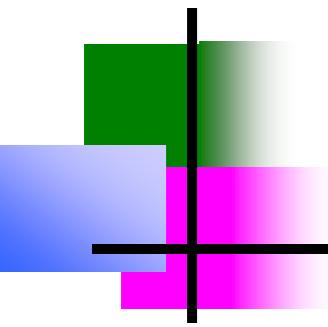
# Big Data Analytics Use Cases



# Big Data

## Big Data Is Different than Business Intelligence

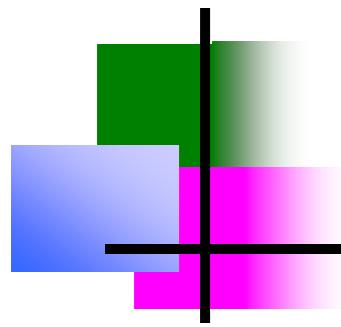




# Big Data Analytics

---

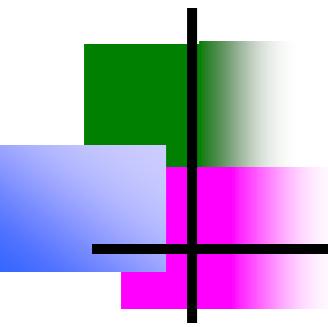
- *Big data analytics is the process of examining large amounts of data of a variety of types.*
- *The primary goal of big data analytics is to help companies make better business decisions.*
- *analyze huge volumes of transaction data as well as other data sources that may be left untapped by conventional business intelligence (BI) programs.*



# Data Analytics

---

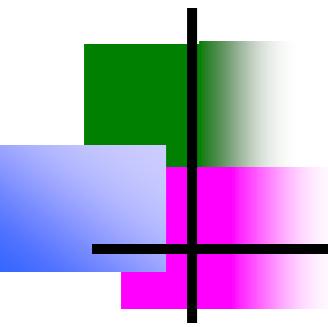
- Big data Consist of
    - uncovered hidden patterns.
    - Unknown correlations and other useful information.
- Such information can provide business benefits.
- more effective marketing and increased revenue.



# Data Analytics

---

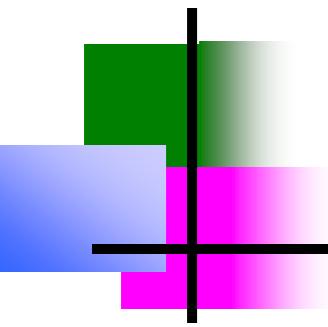
- Big data analytics can be done with the software tools commonly used as part of advanced analytics disciplines.
  - such as predictive analysis and data mining.
- 
- But the unstructured data sources used for big data analytics may not fit in traditional data warehouses.
  - Traditional data warehouses may not be able to handle the processing demands posed by big data.



# Data Analytics

---

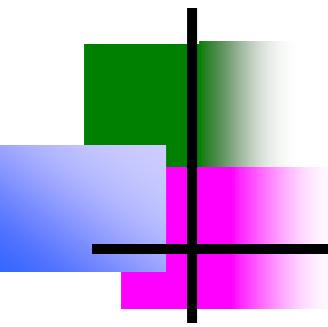
- *The technologies associated with big data analytics include NoSQL databases, Hadoop and MapReduce.*
- *Known about these technologies form the core of an open source software framework that supports the processing of large data sets across clustered systems.*
- *big data analytics initiatives include*
  - *internal data analytics skills*
  - *high cost of hiring experienced analytics professionals,*
  - *challenges in integrating Hadoop systems and data warehouses*



# Data Analytics

---

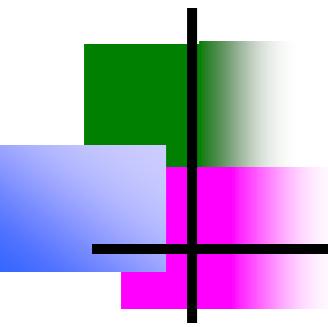
- *Big Analytics delivers competitive advantage in two ways compared to the traditional analytical model.*
- *First, Big Analytics describes the efficient use of a simple model applied to volumes of data that would be too large for the traditional analytical environment.*
- *Research suggests that a simple algorithm with a large volume of data is more accurate than a sophisticated algorithm with little data.*



# Data Analytics

---

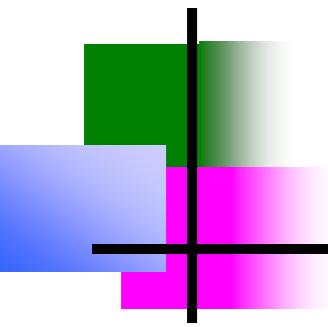
- *Big Analytics supporting the following objectives for working with Big Data Analytics:*
- *1. Avoid sampling / aggregation;*
- *2. Reduce data movement and replication;*
- *3. Bring the analytics as close as possible to the data.*
- *4. Optimize computation speed.*



# Data Analytics

---

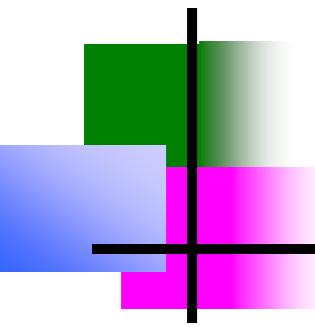
- *The term “analytics” refers to the use of information technology to harness statistics, algorithms and other tools of mathematics to improve decision-making.*
- *Guidance for analytics must recognize that processing of data may not be linear.*
- *May involve the use of data from a wide array of sources.*
- *Principles of fair information practices may be applicable at different points in analytic processing.*
- *Guidance must be sufficiently flexible to serve the dynamic nature of analytics and the richness of the data to which it is applied.*



# The Power and Promise of Analytics

---

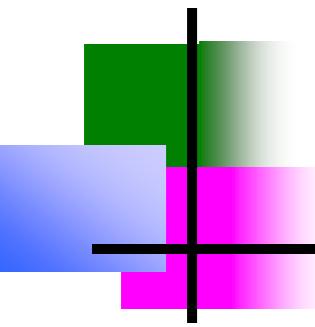
- *Big Data Analytics to Improve Network Security.*
- *Security professionals manage enterprise system risks by controlling access to systems, services and applications defending against external threats.*
- *protecting valuable data and assets from theft and loss.*
- *monitoring the network to quickly detect and recover from an attack.*
- *Big data analytics is particularly important to network monitoring, auditing and recovery.*
- *Business Intelligence uses big data and analytics for these purposes.*



# The Power and Promise of Analytics

---

- *Reducing Patient Readmission Rates (Medical data)*
- *big data to address patient care issues and to reduce hospital readmission rates.*
- *The focus on lack of follow-up with patients, medication management issues and insufficient coordination of care.*
- *Data is preprocessed to correct any errors and to format it for analysis.*

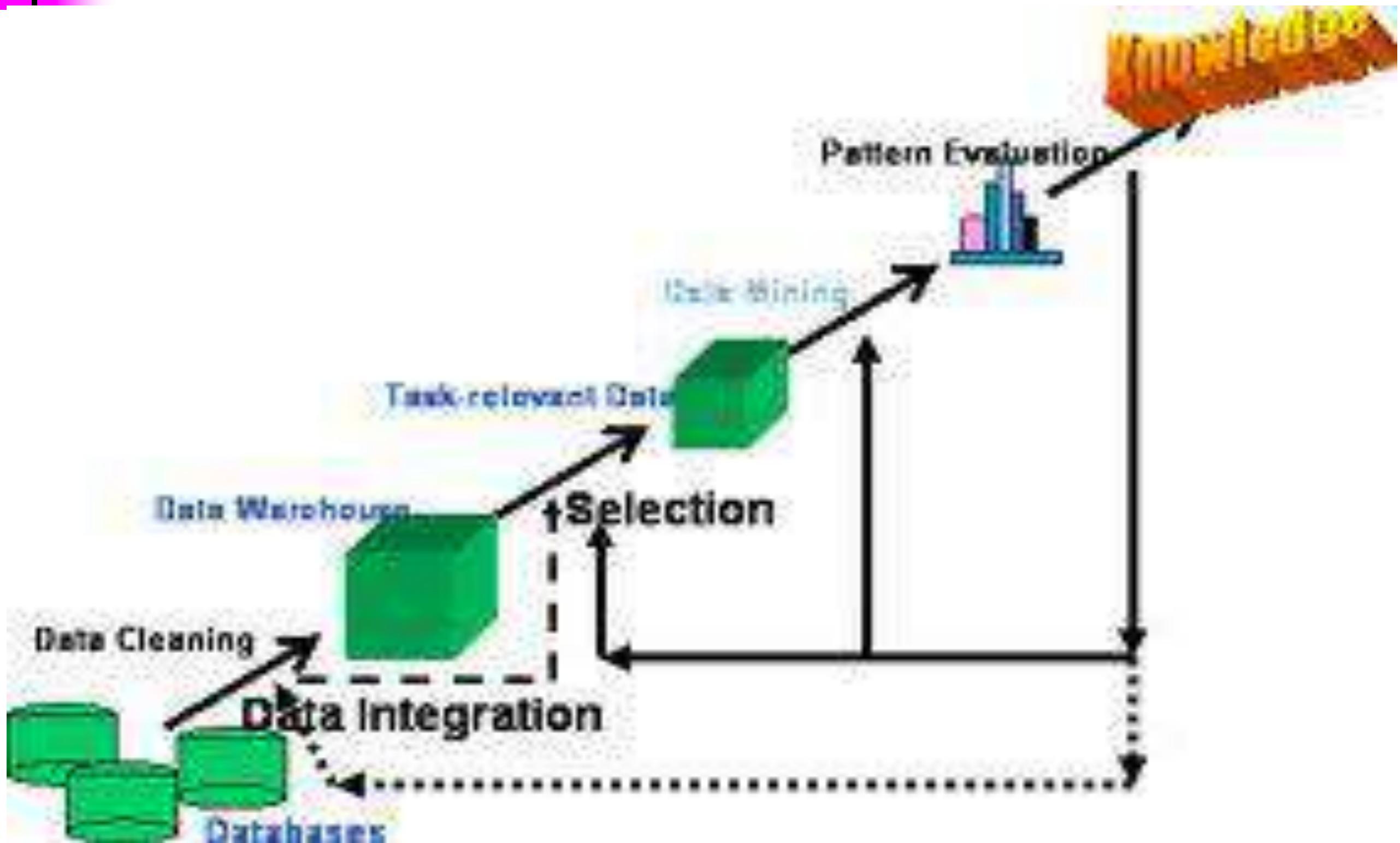


# The Power and Promise of Analytics

---

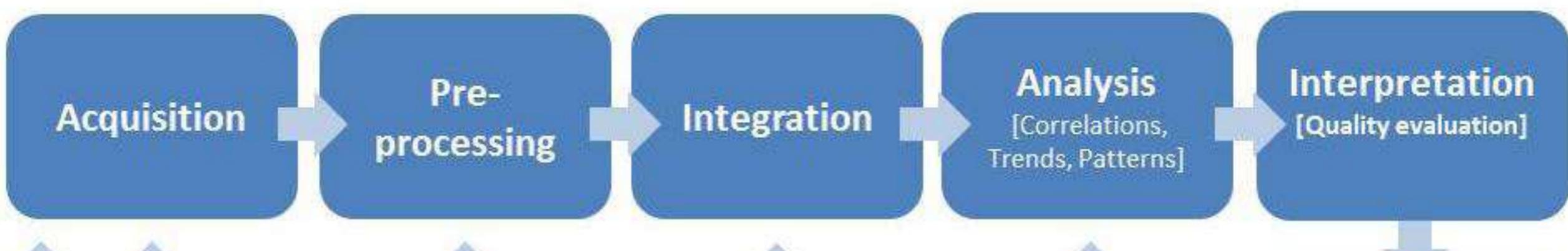
- Analytics to Reduce the Student Dropout Rate (Educational Data)
- Analytics applied to education data can help schools and school systems better understand how students learn and succeed.
- Based on these insights, schools and school systems can take steps to enhance education environments and improve outcomes.
- Assisted by analytics, educators can use data to assess and when necessary re-organize classes, identify students who need additional feedback or attention.
- Direct resources to students who can benefit most from them.
- etc.....

# Process of Data Analytics (KDD process)



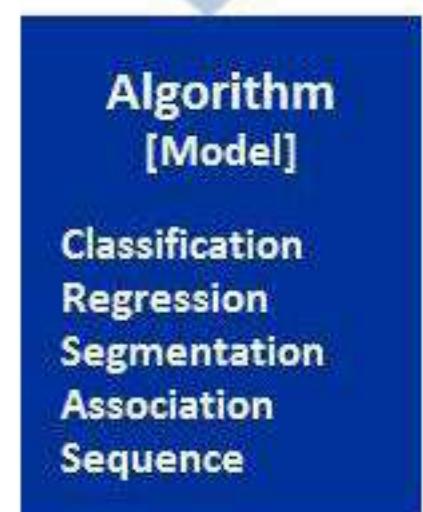
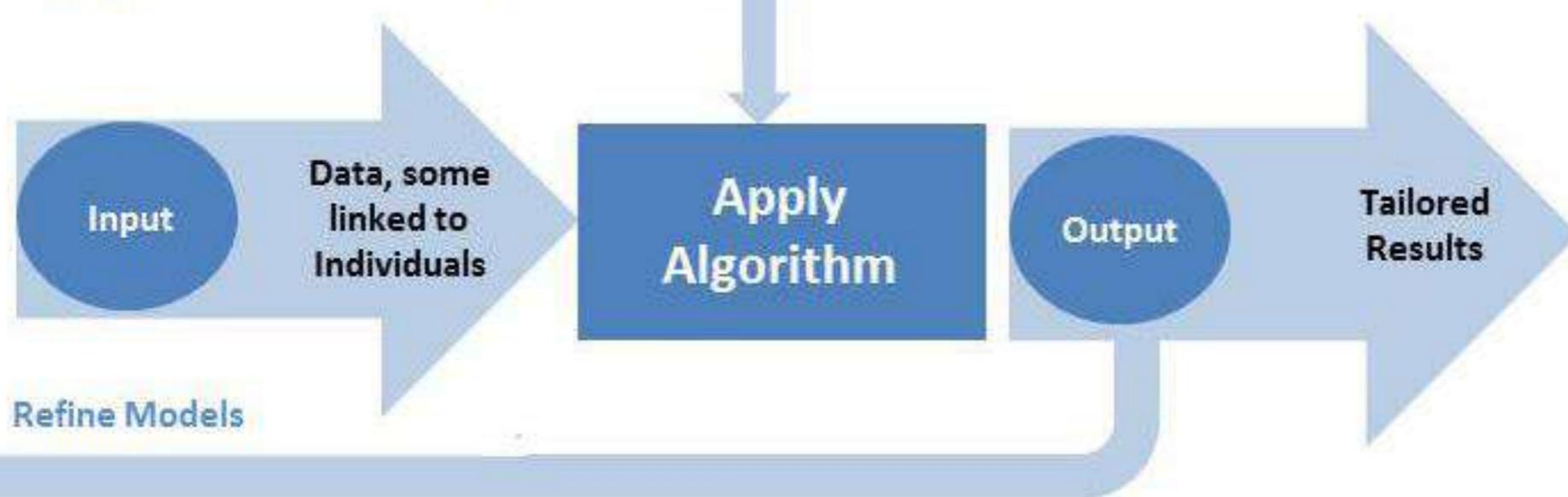
# The Process of Analytics

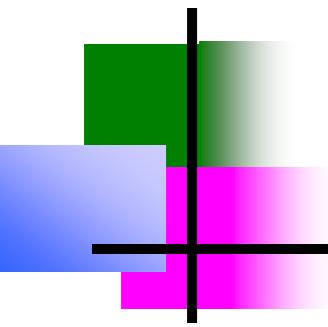
## Discovery (phase 1)



Iterative process: Quality Not Sufficient or New Questions Arise

## Application (phase 2)

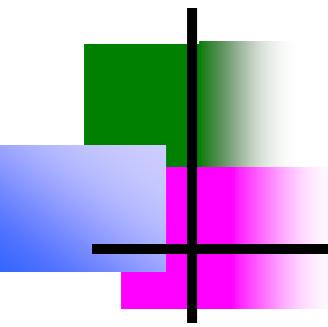




# The Process of Analytics (Phase-1)

---

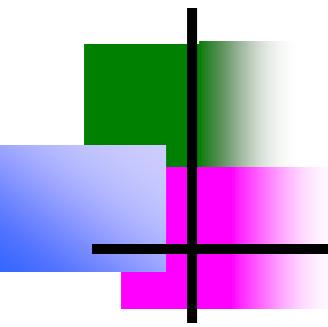
- This knowledge discovery phase involves
  - *gathering data to be analyzed.*
  - *pre-processing it into a* format that can be used.
  - consolidating (more certain) it for analysis, analyzing it to discover what it may reveal.
  - and interpreting it to understand the processes by which the data was analyzed and how conclusions were reached.



# The Process of Analytics (Phase-1)

---

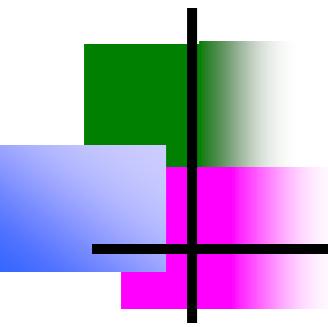
- *Acquisition –(process of getting something)*
- *Data acquisition involves collecting or acquiring data for analysis.*
- *Acquisition requires access to information and a mechanism for gathering it.*



# The Process of Analytics (Phase-1)

---

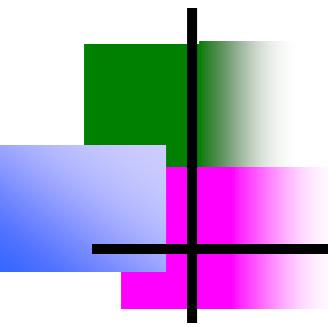
- *Pre-processing* -:
- *Data is structured and entered into a consistent format that can be analyzed.*
- *Pre-processing is necessary if analytics is to yield trustworthy (**able to trusted**), useful results.*
- *places it in a standard format for analysis.*



# The Process of Analytics (Phase-1)

---

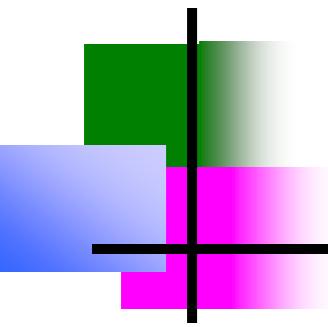
- *Integration* :-
- *Integration involves consolidating data for analysis.*
  - *Retrieving relevant data from various sources for analysis*
  - *eliminating redundant data or clustering data to obtain a smaller representative sample.*
  - *clean data into its data warehouse and further organizes it to make it readily useful for research.*
  - *distillation into manageable samples.*



# The Process of Analytics (Phase-1)

---

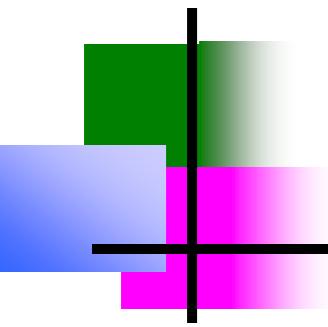
- *Analysis –; Knowledge discovery involves*
  - *searching for relationships between data items in a database, or exploring data in search of classifications or associations.*
  - *Analysis can yield descriptions (where data is mined to characterize properties) or predictions (where a model or set of models is identified that would yield predictions).*
  - *Analysis based on interpretation, organizations can determine whether and how to act on them.*



# The Process of Analytics (Phase-1)

---

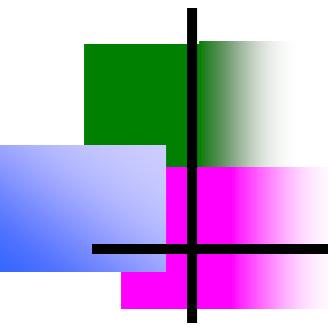
- *Interpretation* :-
  - *Analytic processes are reviewed by data scientists to understand results and how they were determined.*
  - *Interpretation involves retracing methods, understanding choices made throughout the process and critically examining the quality of the analysis.*
  - *It provides the foundation for decisions about whether analytic outcomes are trustworthy*



# The Process of Analytics (Phase-1)

---

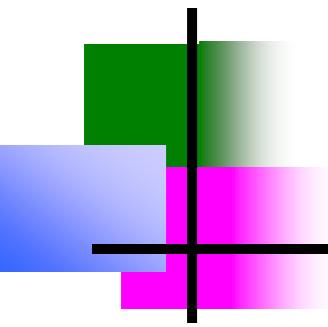
- *The product of the knowledge discovery phase is an algorithm. Algorithms can perform a variety of tasks:*
- *Classification algorithms categorize discrete variables (such as classifying an incoming email as spam).*
- *Regression algorithms calculate continuous variables (such as the value of a home based on its attributes and location).*
- *Segmentation algorithms divide data into groups or clusters of items that have similar properties (such as tumors found in medical images).*
- *Association algorithms find correlations between different attributes in a data set (such as the automatically suggested search terms in response to a query).*
- *Sequence analysis algorithms summarize frequent sequences in data (such as understanding a DNA sequence to assign function to genes and proteins by comparing it to other sequences).*



# The Process of Analytics (Phase-2)

---

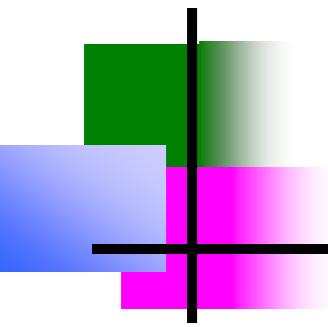
- *Application*
  - *Associations discovered amongst data in the knowledge phase of the analytic process are incorporated into an algorithm and applied.*
  - *for example, classify individuals according to certain criteria, and in doing so determine their suitability to engage in a particular activity.*
  - *In the application phase organizations reap (collect) the benefits of knowledge discovery.*
  - *Through application of derived algorithms, organizations make determinations upon which they can act.*



# Goals for Analytics Guidance

---

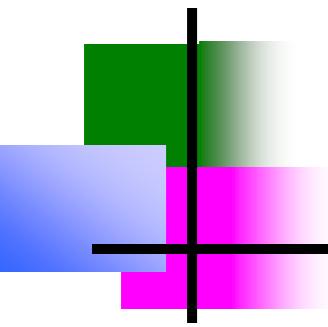
- *Recognize and reflect the two-phased nature of analytic processes.*
  - *Traditional methods of data analysis usually involve identification of a question and analysis of data in search of answers to that question.*
  - *Use of advanced analytics with big data upends that approach by making it possible to find patterns in data through knowledge discovery. Rather than approach data with a predetermined question.*
  - *The results of this analysis may be unexpected.*
  - *Moreover, this research may suggest further questions for analysis or prompt exploration of data to identify additional insights, through iterative analytic processing.*



# Goals for Analytics Guidance

---

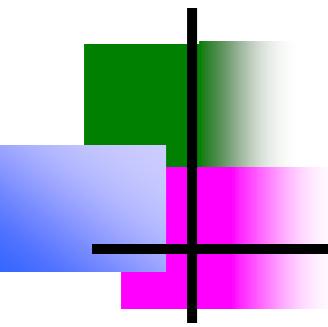
- *Provide guidance for companies about how to establish that their use of data for knowledge discovery is a legitimate business purpose.*
  - allow for processing of data for a legitimate business purpose, but provide little guidance about how organizations establish legitimacy and demonstrate it to the appropriate oversight body.
  - Guidance for analytics would articulate the criteria against which legitimacy is evaluated and describe how organizations demonstrate to regulators or other appropriate authorities the steps they have taken to support it.



# Goals for Analytics Guidance

---

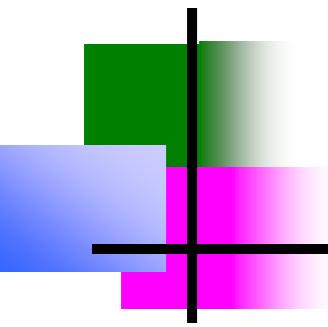
- *Emphasize the need to establish accountability through an internal privacy program that relies upon the identification and mitigation of the risks the use of data for analytics may raise for individuals.*
  - *how fair information practices are applied, it is important that organizations implement an internal privacy program that involves credible assessment of the risks data processing may raise.*
  - *Risk mitigation may involve de-identification and pseudo-nominisation of data, as well as other controls to prevent re-identification of the original data subject.*



# Goals for Analytics Guidance

---

- *Take into account that analytics may be an iterative process using data from a variety of sources.*
  - analytics is not necessarily a linear process. Insights yielded by analytics may be identified as flawed or lacking, and data scientists may in response re-develop an algorithm or re-examine the appropriateness of the data for its intended purpose and prepare it for further analysis.
  - Knowledge discovery may reveal that data could provide additional insights, and researchers may choose to explore them further. Data used for analytics may come from an organization's own stores, but may also be derived from public records.
  - Data entered into the analytic process may also be the result of earlier processing.

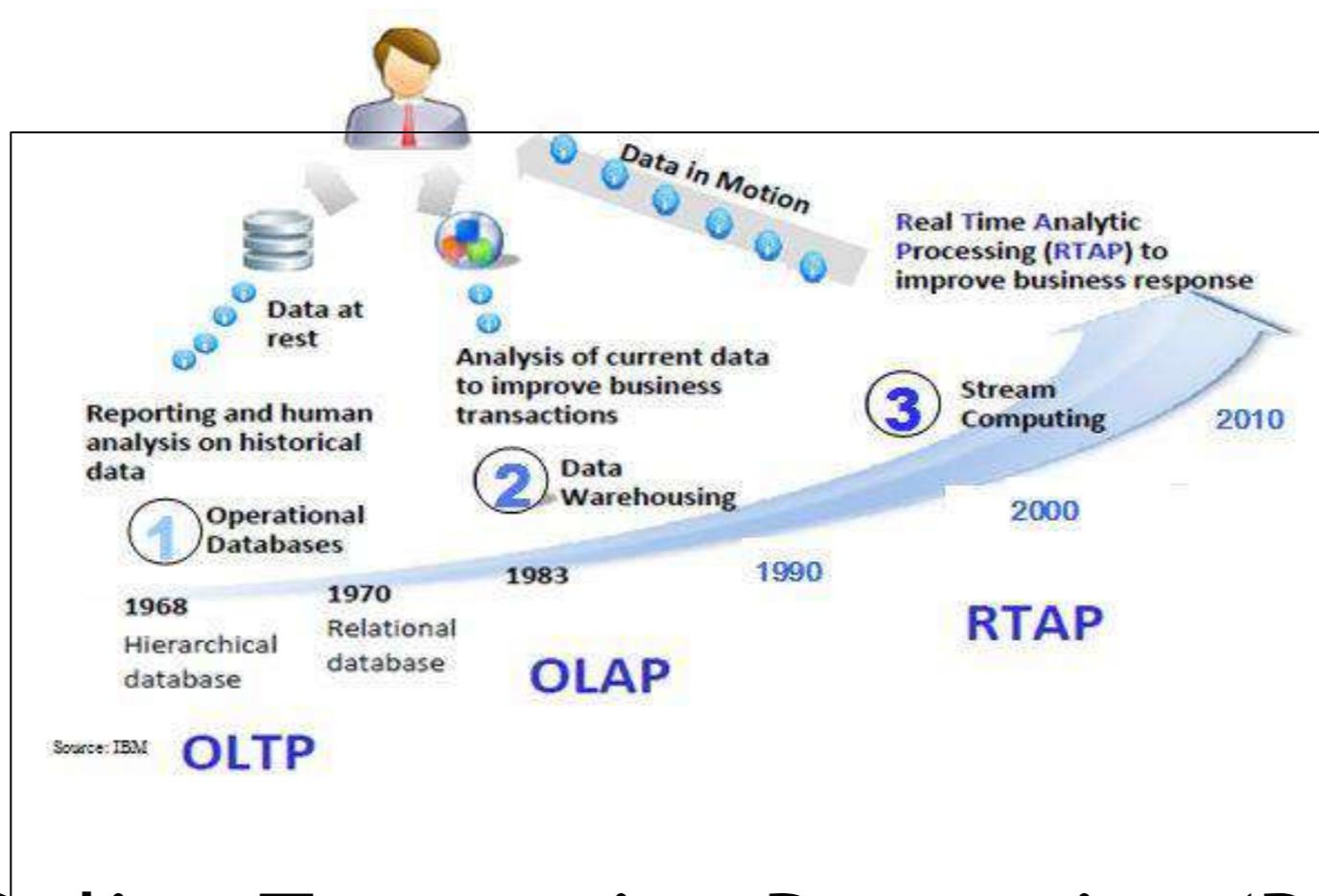


# Data Analytics

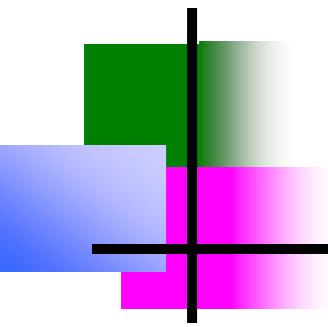
---

- *Conclusion*
  - *Analytics and big data hold growing potential to address longstanding issues in critical areas of business, science, social services, education and development. If this power is to be tapped responsibly, organizations need workable guidance that reflects the realities of how analytics and the big data environment work.*
  - *Such guidance must be grounded on the consensus of*
    - *international stakeholders.*
    - *data protection authorities and regulators.*
    - *business leaders.*
    - *academics and experts.*
    - *and civil society.*
- *Thoughtful, practical guidance can release and enhance the power of data to address societal questions in urgent need of answers.*
- *A trusted dialogue to arrive at that guidance will be challenging, but cannot wait.*

# Harnessing Big Data



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

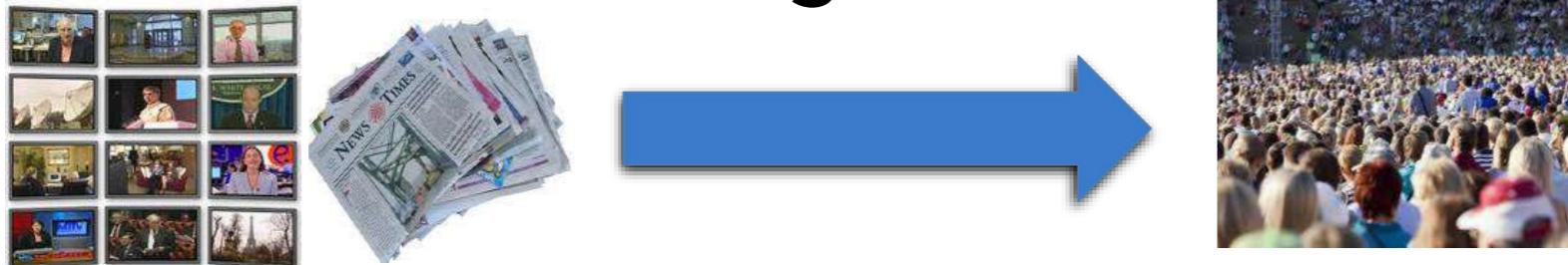


# The Model Has Changed...

---

- The Model of Generating/Consuming Data has Changed

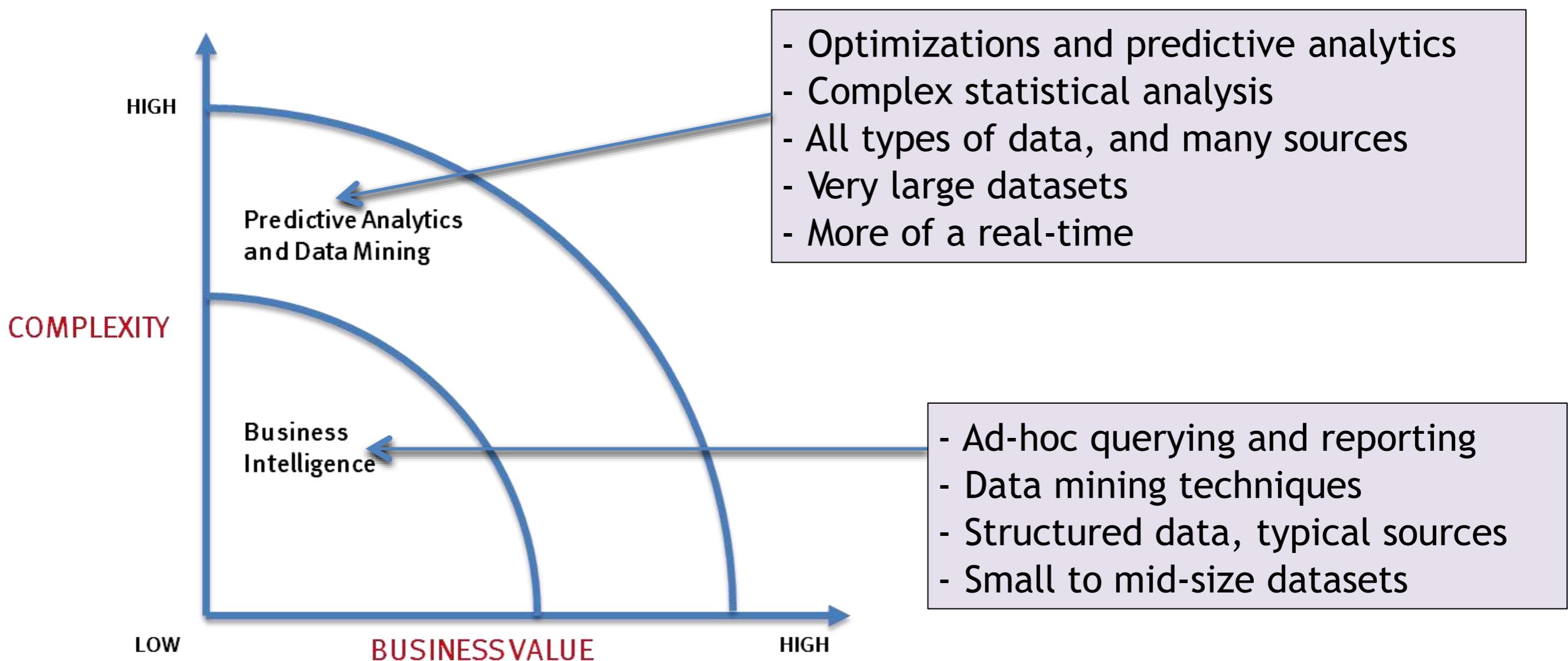
**Old Model:** Few companies are generating data, all others are consuming data



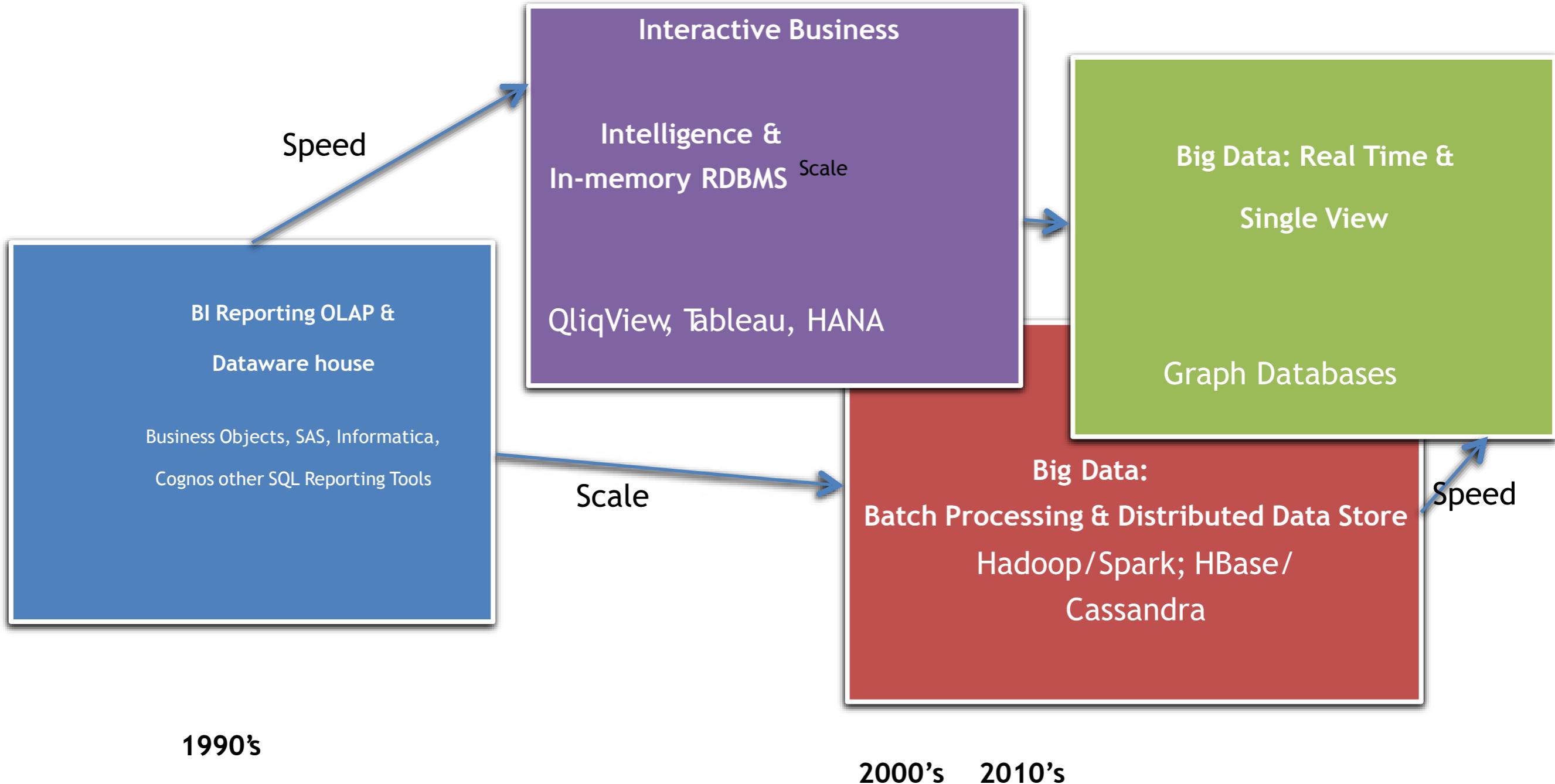
**New Model:** all of us are generating data, and all of us are consuming data



# What's driving Big Data

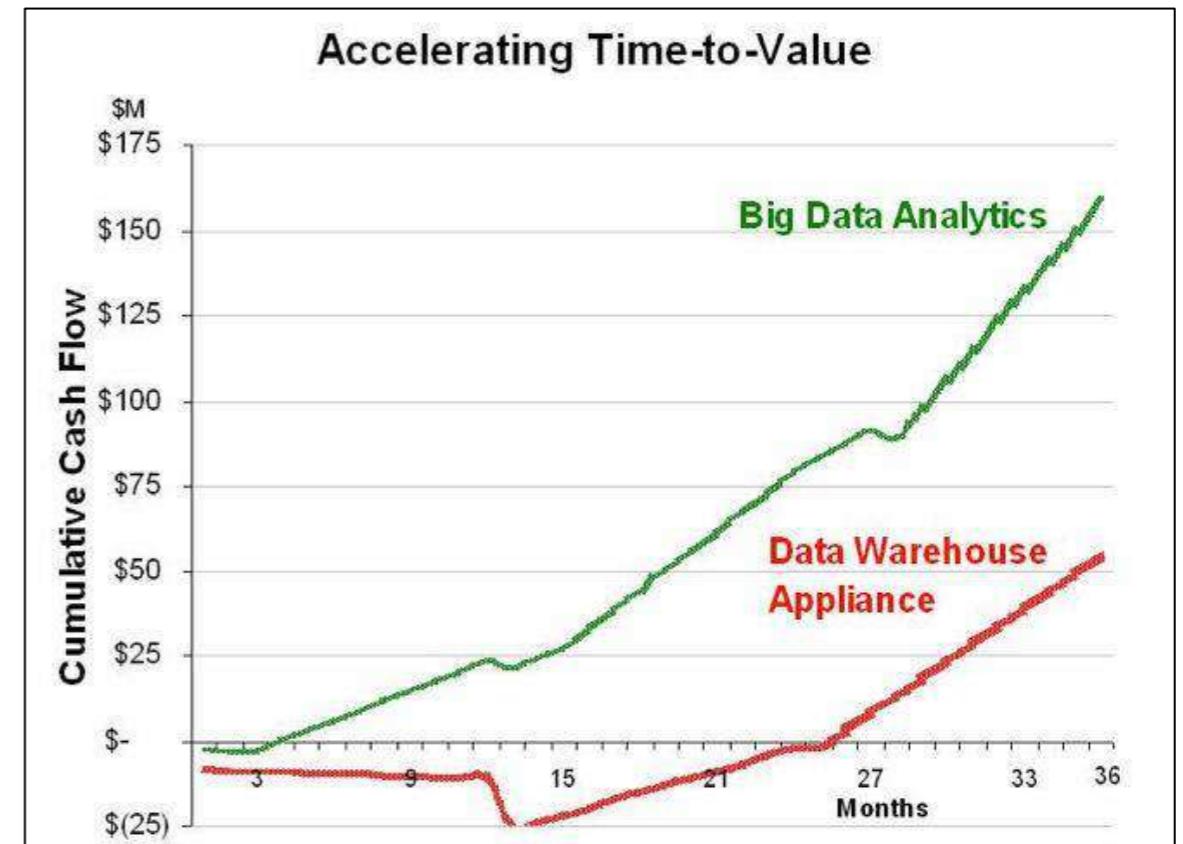


# THE EVOLUTION OF BUSINESS INTELLIGENCE



# Big Data Analytics

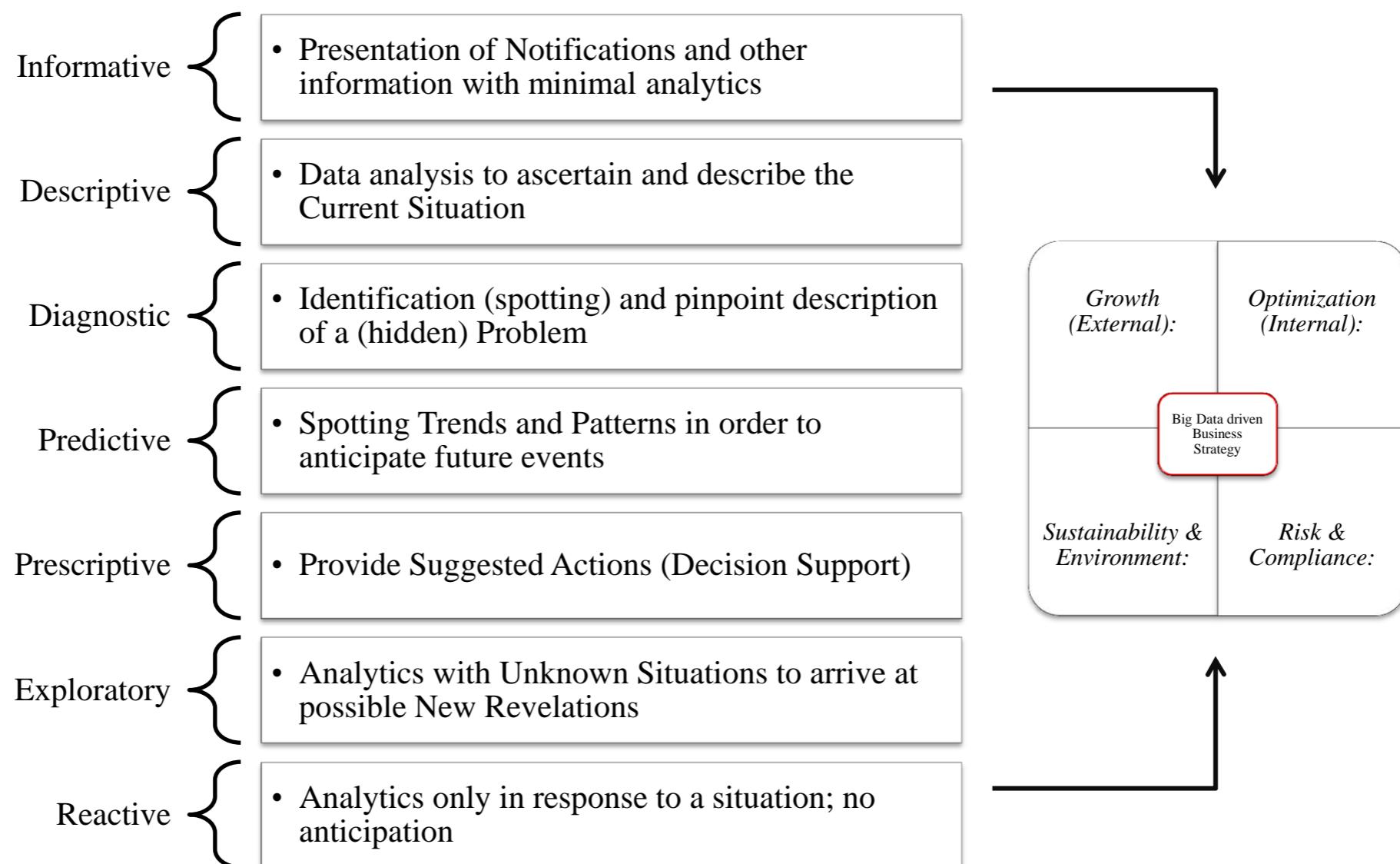
- Big data is more real-time in nature than traditional DW applications
- Traditional DW architectures (e.g. Exadata, Teradata) are not well-suited for big data apps
- Shared nothing, massively parallel processing, scale out architectures are well-suited for big data apps

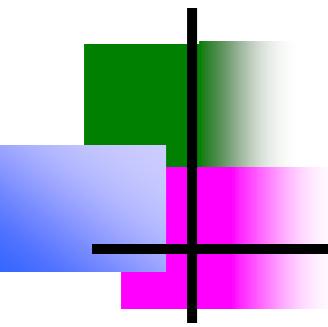


# Big Data Analytics



# Figure 3.11: Various Analytics Categories provide Agile Business Values (and form basis of Business Strategies)



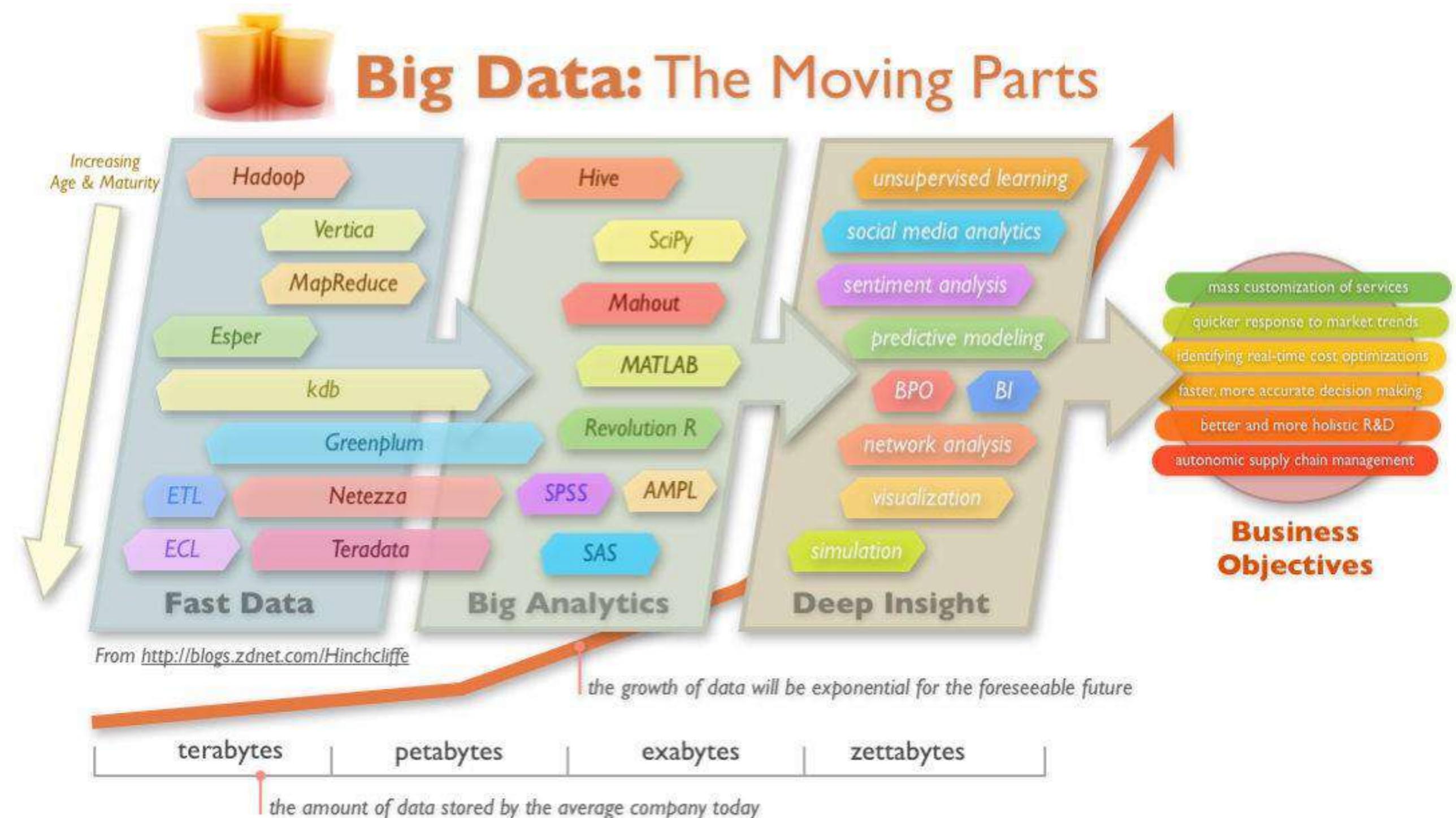


# Big Data Analytics

---

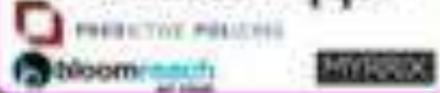
	Traditional Analytics (BI)	vs	Big Data Analytics
Focus on	<ul style="list-style-type: none"><li>• Descriptive analytics</li><li>• Diagnosis analytics</li></ul>		<ul style="list-style-type: none"><li>• <b>Predictive analytics</b></li><li>• <b>Data Science</b></li></ul>
Data Sets	<ul style="list-style-type: none"><li>• Limited data sets</li><li>• Cleansed data</li><li>• Simple models</li></ul>		<ul style="list-style-type: none"><li>• Large scale data sets</li><li>• More types of data</li><li>• Raw data</li><li>• Complex data models</li></ul>
Supports	<b>Causation:</b> what happened, and why?		<b>Correlation:</b> new insight More accurate answers

# Big Data Technology



# Big Data Landscape

## Vertical Apps



## Log Data Apps



## Ad/Media Apps



## Business Intelligence



## Analytics and Visualization



## Data As A Service



## Analytics Infrastructure



## Operational Infrastructure



## Infrastructure As A Service



## Structured Databases



## Technologies



# Big Data Landscape

## Infrastructure

### NoSQL Databases



### NewSQL Databases



### MPP Databases



### Crowdsourcing



## Analytics

### Analytics Solutions



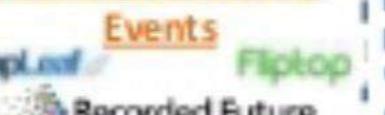
### Statistical Computing



### Sentiment Analysis



### Location / People / Events



### Real-Time



## Applications

### Ad Optimization



### Publisher Tools



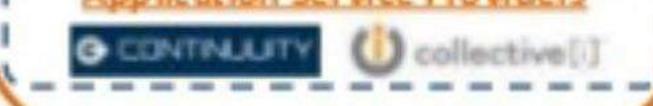
### Marketing



### Industry Applications



### Application Service Providers



## Data Sources

### Data Marketplaces



### Personal Data



## Cross Infrastructure / Analytics

### Microsoft VMware amazon

### ORACLE METAMARKETS TERADATA

## Open Source Projects

### Framework



### Query / Data Flow



### riak Data Access



### mongoDB



### Coordination / Workflow



### Real - Time



### Statistical Tools



### Machine Learning



### Cloud Deployment

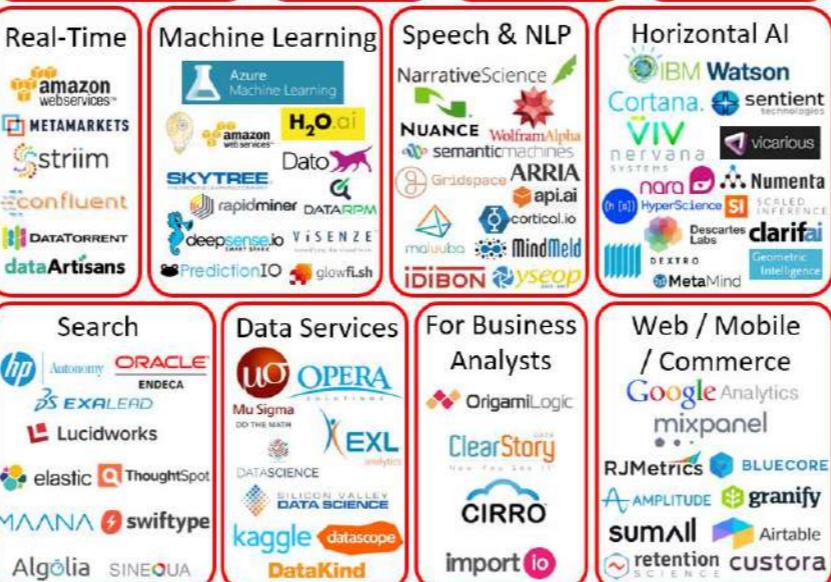
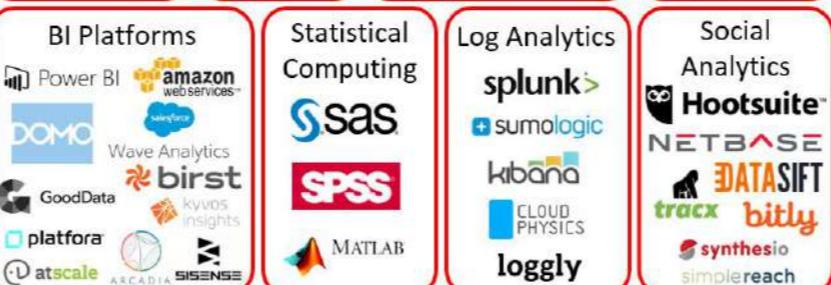
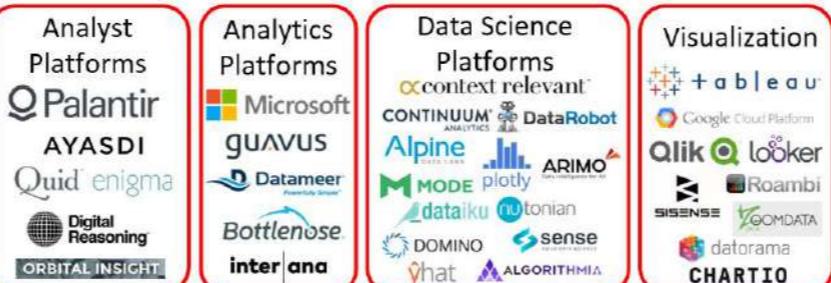


# **Big Data Landscape 2016 (Version 3.0)**

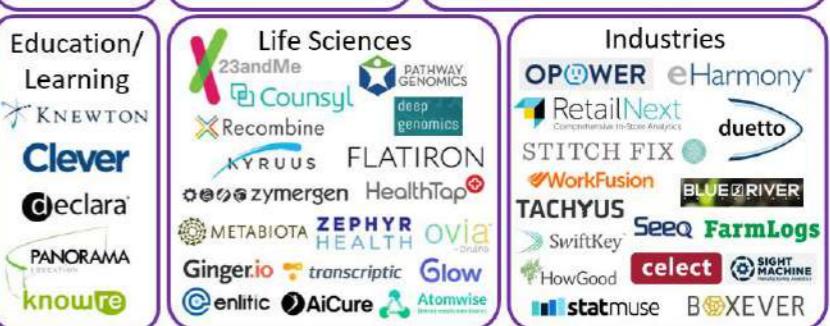
## Infrastructure



## Analytics



## Applications



## Cross-Infrastructure/Analytics



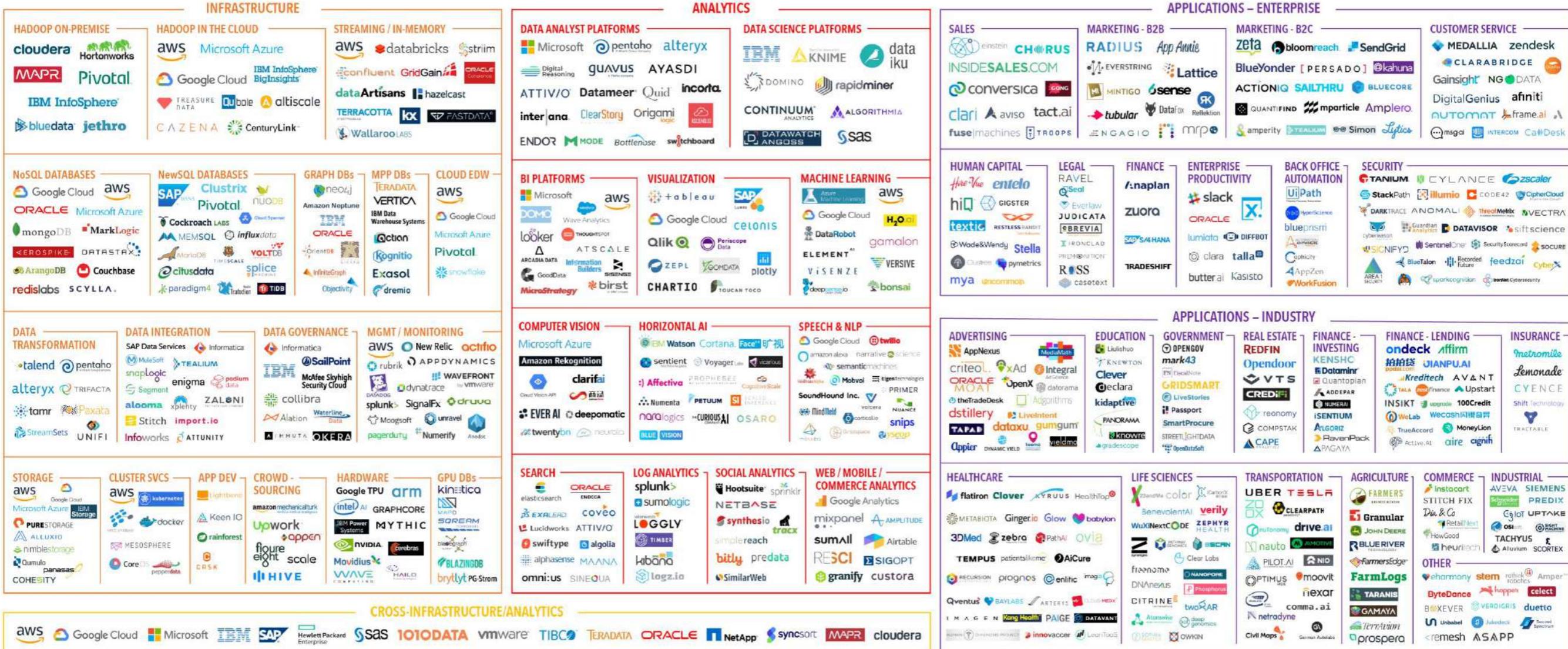
Open Source



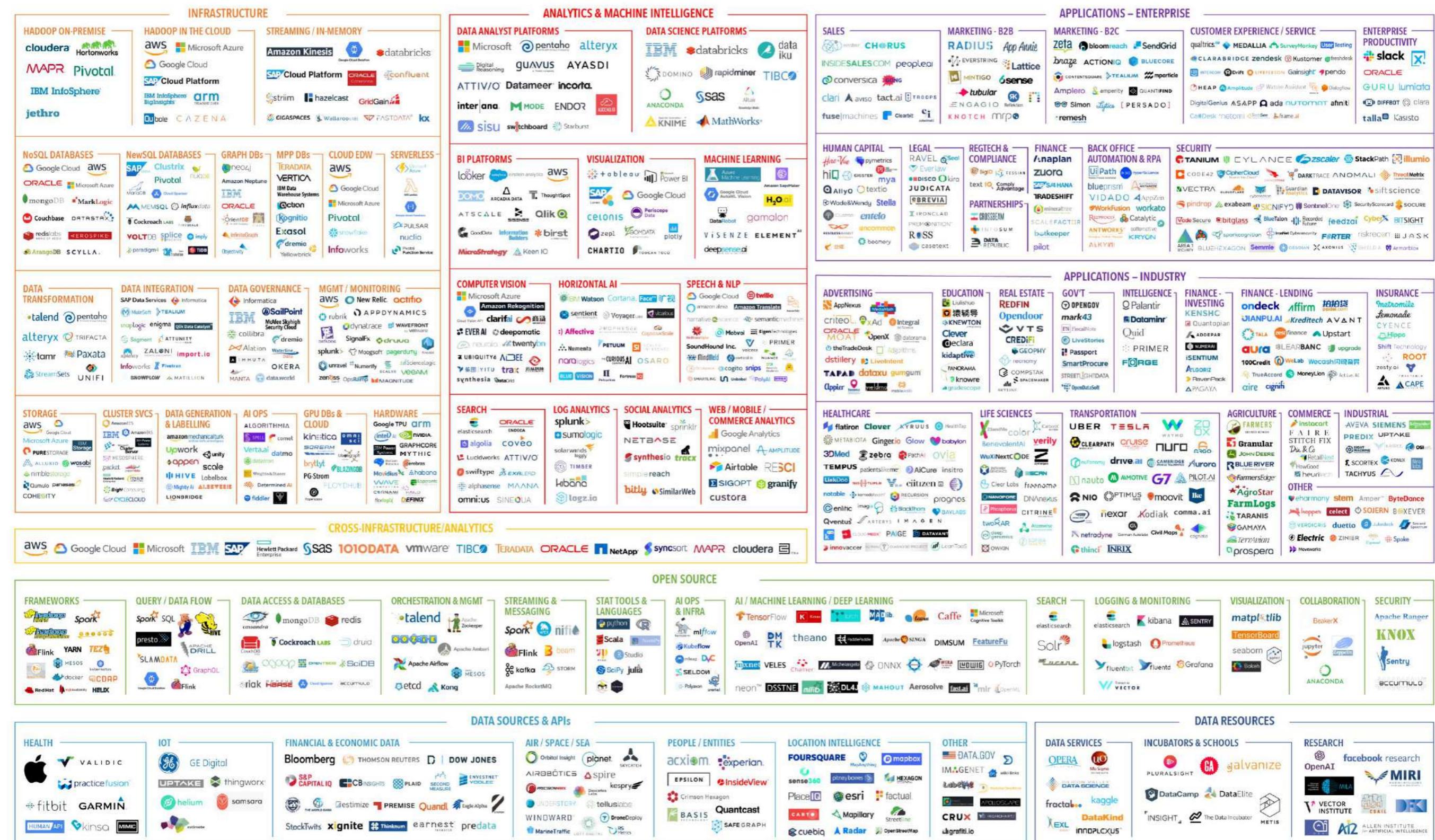
## Data Sources & APIs



BIG DATA & AI LANDSCAPE 2018



DATA & AI LANDSCAPE 2019



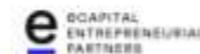
# 2020 AI GERMAN STARTUP LANDSCAPE

247 AI STARTUPS

CONTRIBUTORS:

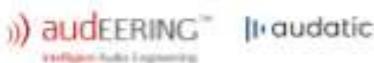


EARLYBIRD



## ENTERPRISE INTELLIGENCE

### Audio Data



### Visual Data



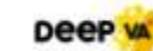
ModuLabs



NavVis



IDnow



deep VA



VISION IMPULS



deevio



owl



EyeEm



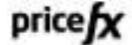
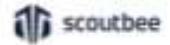
LANA



inspirient



### Market Data



### Sensor Data



maito



ZOLITRON



KONUX



TERAKI



MOTIONTAG

### Geo Data



GEOSPIN



Eagle AI

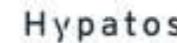


Targomo

### Text Data



THINGS  
THINKING



Hypatos



lengoo



curiosity.ai



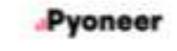
Fileeee



gini



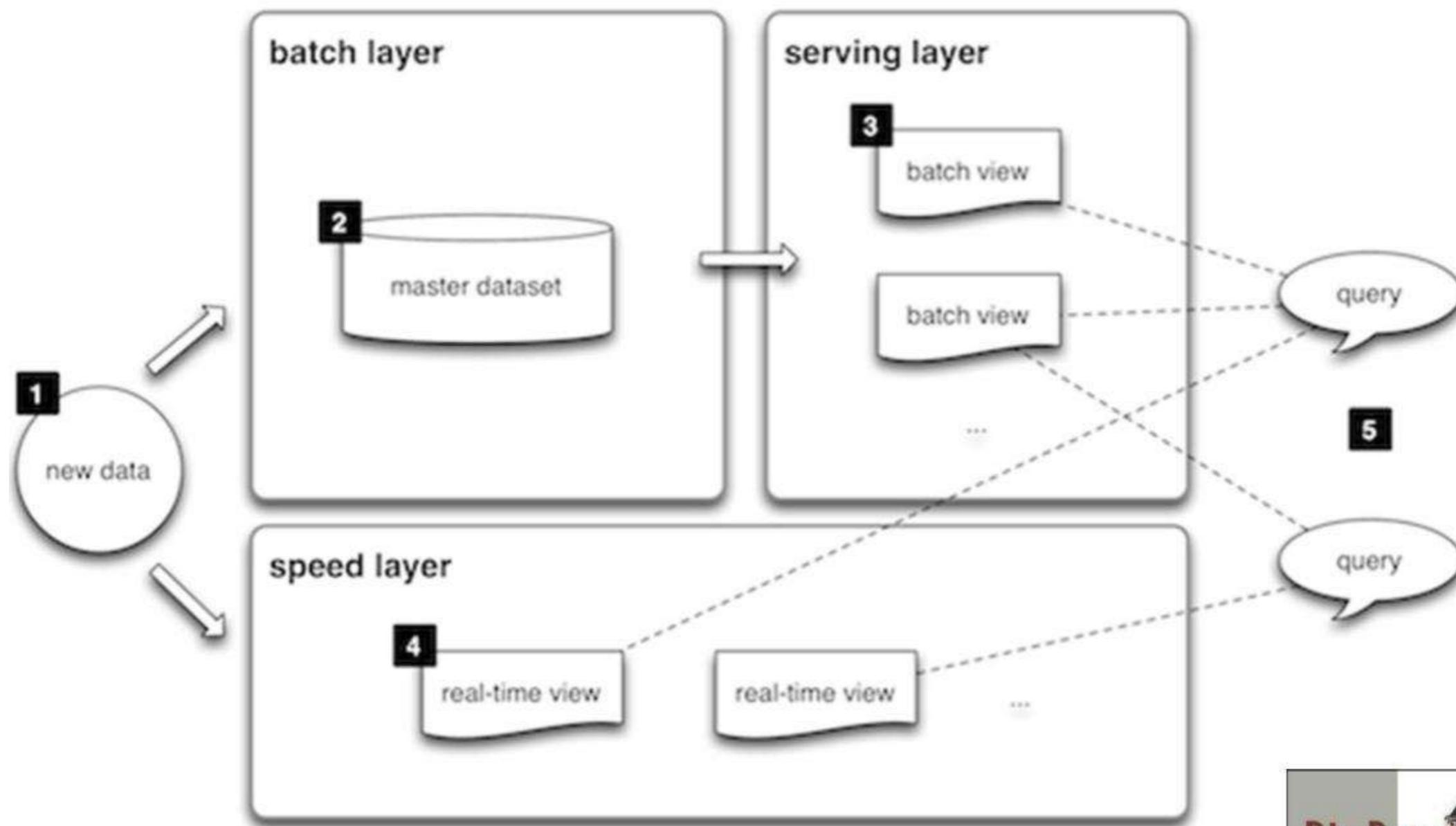
LATERAL



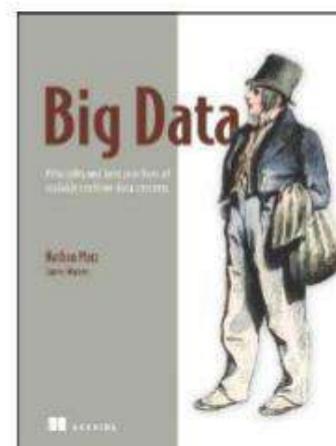
Pyioneer



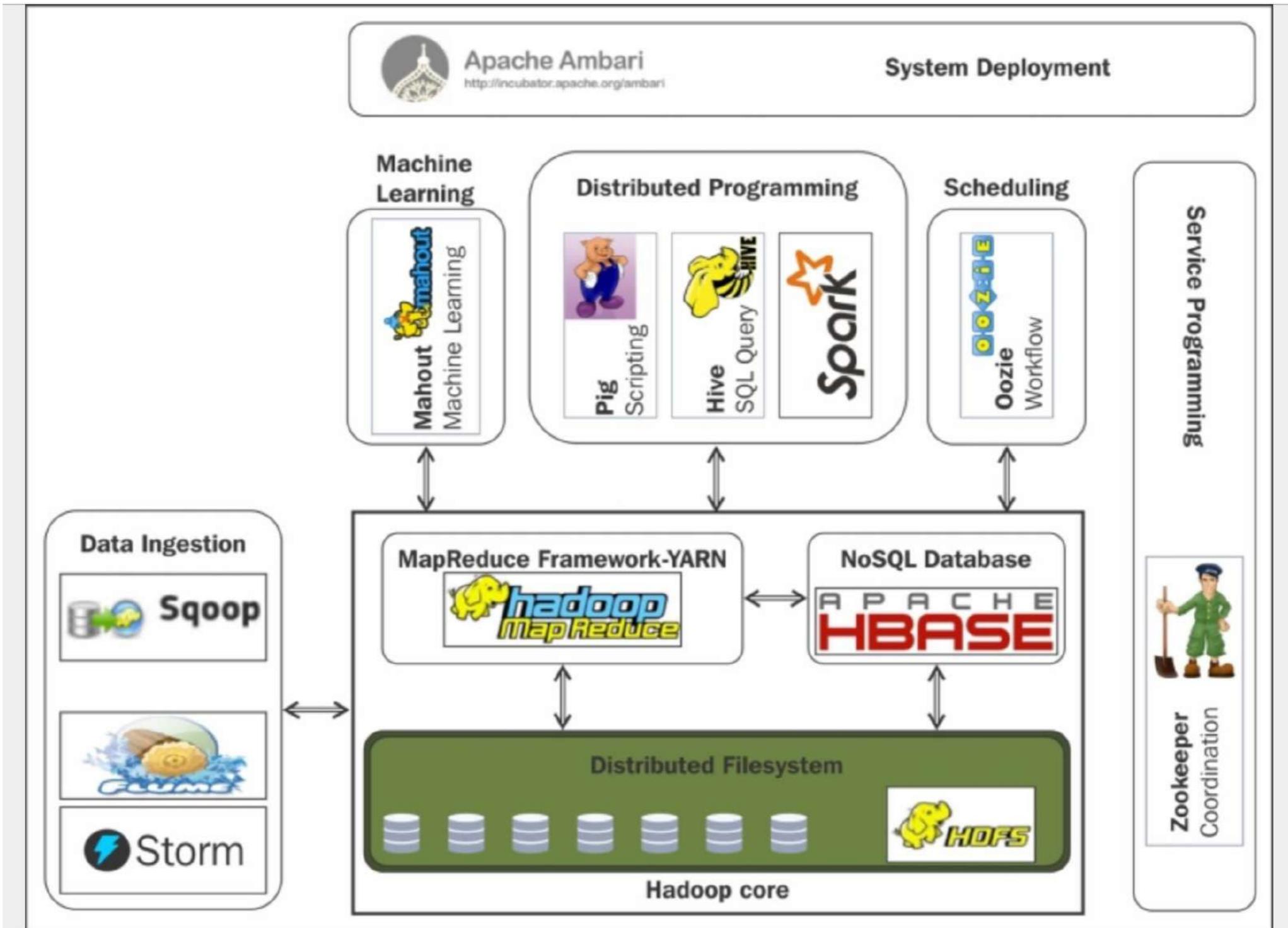
EVANA



*Source:*

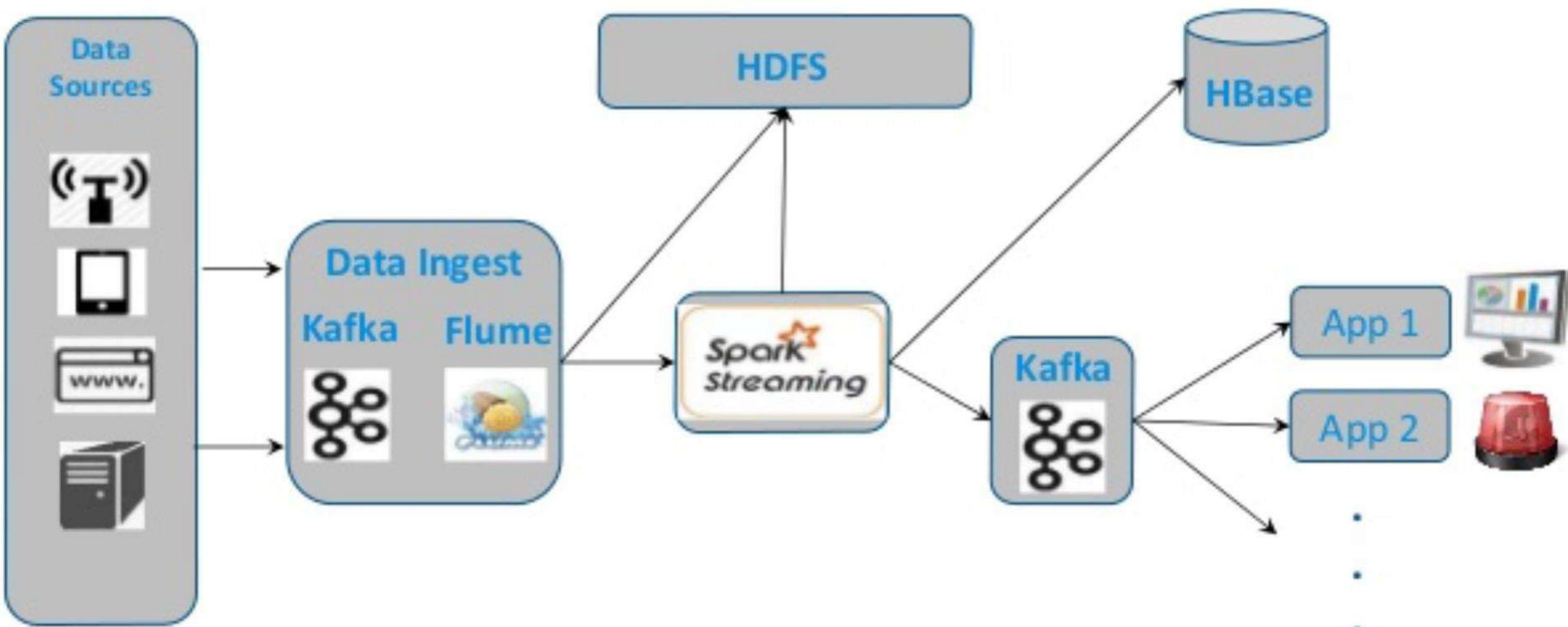


# Hadoop Ecosystem

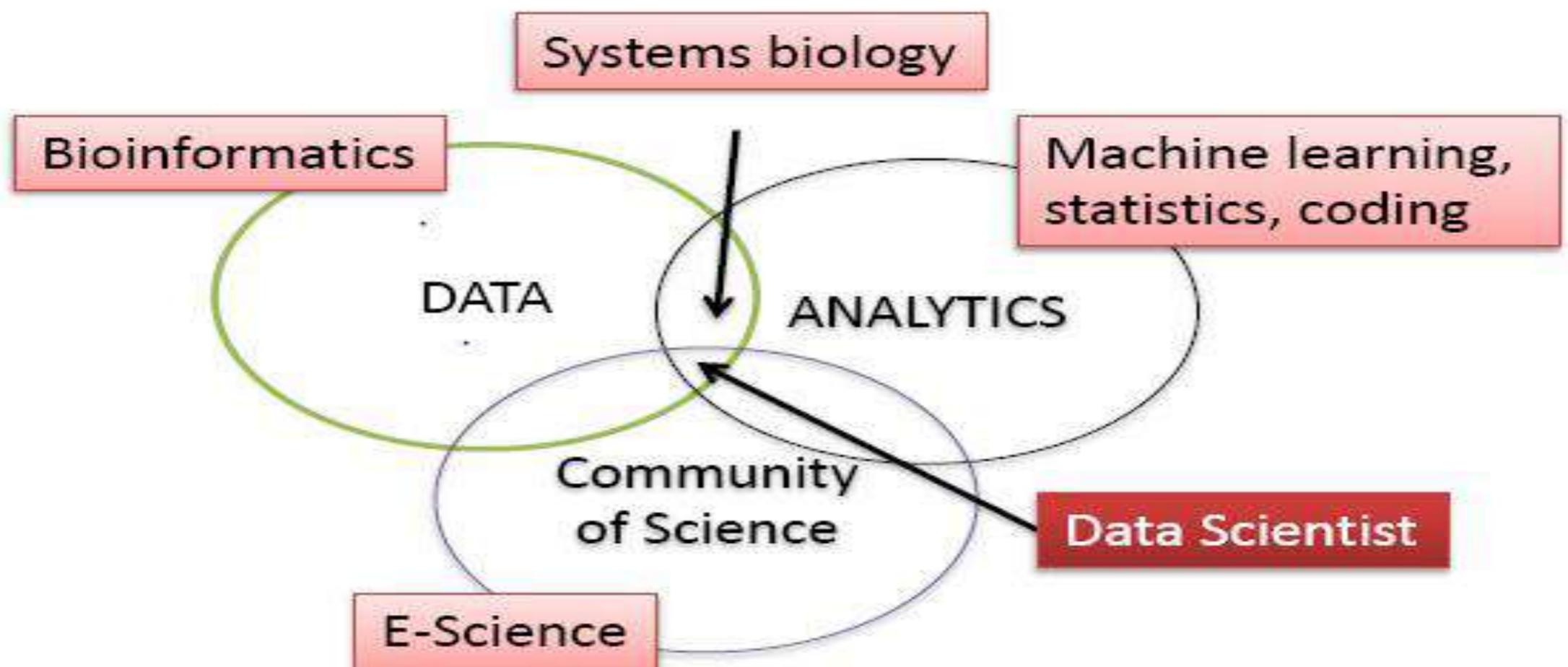


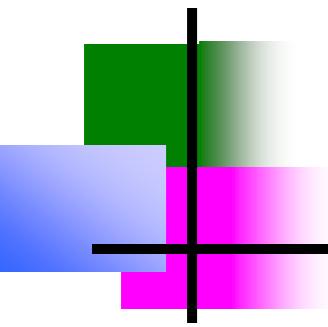
# Stream Processing

## Architecture



# Data Scientist

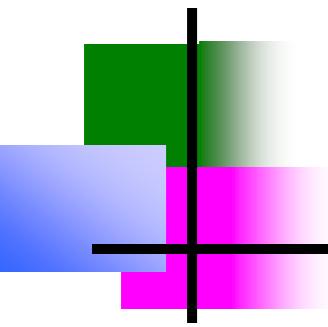




# Data Scientist

---

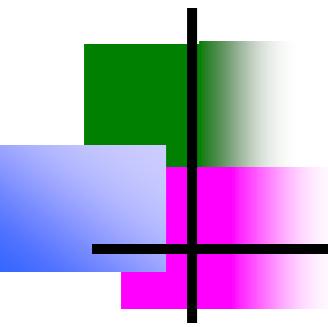
- *Data scientist includes*
- *Data capture and Interpretation*
- *New analytical techniques*
- *Community of Science*
- *Perfect for group work*
- *Teaching strategies*



# Data scientist

---

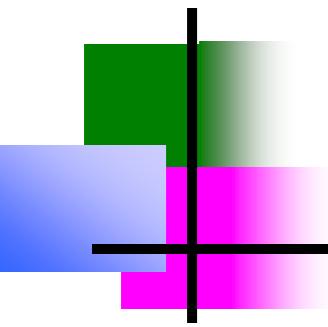
- *Data scientists require a wide range of skills:*
  - *Business domain expertise and strong analytical skills*
  - *Creativity and good communications.*
  - *Knowledgeable in statistics, machine learning and data visualization*
  - *Able to develop data analysis solutions using modeling/analysis methods and languages such as MapReduce, R, SAS, etc.*
  - *Adept at data engineering, including discovering and mashing/blending large amounts of data.*



# Data scientist

---

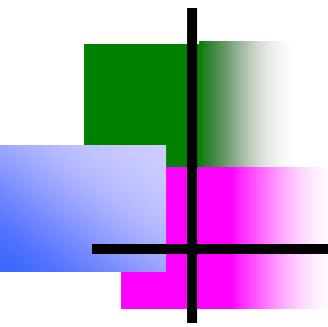
- *Data scientists use an investigative computing platform*
  - *to bring un-modeled data.*
  - *multi-structured data, into an investigative data store for experimentation.*
  - *deal with unstructured, semi-structured and astructured data from various source.*



# Data scientist

---

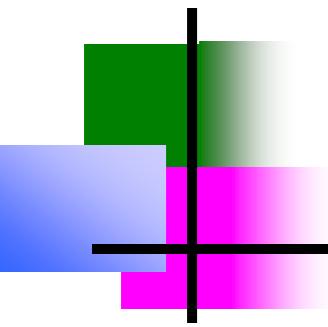
- *Data scientist helps broaden the business scope of investigative computing in three areas:*
- *New sources of data – supports access to multi-structured data.*
- *New and improved analysis techniques – enables sophisticated analytical processing of multi-structured data using techniques such as Map-Reduce and in-database analytic functions.*
- *Improved data management and performance – provides improved price/performance for processing multi-structured data using non-relational systems such as Hadoop, relational DBMSs, and integrated hardware/software.*



# Role of Data scientist

---

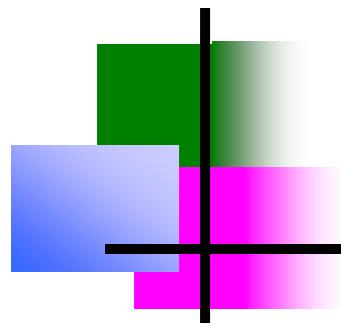
- *Goal of data analytics is the role of data scientist*
  - *Recognize and reflect the two-phased nature of analytic processes.*
  - *Provide guidance for companies about how to establish that their use of data for knowledge discovery is a legitimate business purpose.*
  - *Emphasize the need to establish accountability through an internal privacy program that relies upon the identification and mitigation of the risks the use of data for analytics may raise for individuals.*
  - *Take into account that analytics may be an iterative process using data from a variety of sources.*



# Current trend in Big data Analytics

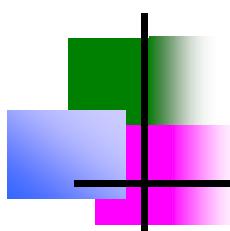
---

- Iterative process (Discovery and Application)
- In general:
- Analyze the unstructured data (Data analytics)
- development of algorithm (Data analytics)
- Data Scrub (Data engineer)
- Present structured data (relationship, association)
- Data refinement (Data scientist)
- Process data using distributed engine. E.g. HDFS (S/W engineer) and write to No-SQL DB (Elasticsearch, Hbase, MongoDB, Cassandra, etc)
- Visual presentation in Application sw.
- QC verification.
- Client release.



---

Thank you !!!



# Chapter2

# Technologies for handing of Big Data

Basanta Joshi, PhD

Asst. Prof., Depart of Electronics and Computer Engineering  
Program Coordinator, MSc in Information and Communication Engineering  
Member, Laboratory for ICT Research and Development (LICT)

Member, Research Management Cell (RMC)

Institute of Engineering

basanta@ioe.edu.np

<http://www.basantajoshi.com.np>

<https://scholar.google.com/citations?user=iocLiGcAAAAJ>

[https://www.researchgate.net/profile/Basanta\\_Joshi2](https://www.researchgate.net/profile/Basanta_Joshi2)



# Big Data In the Cloud

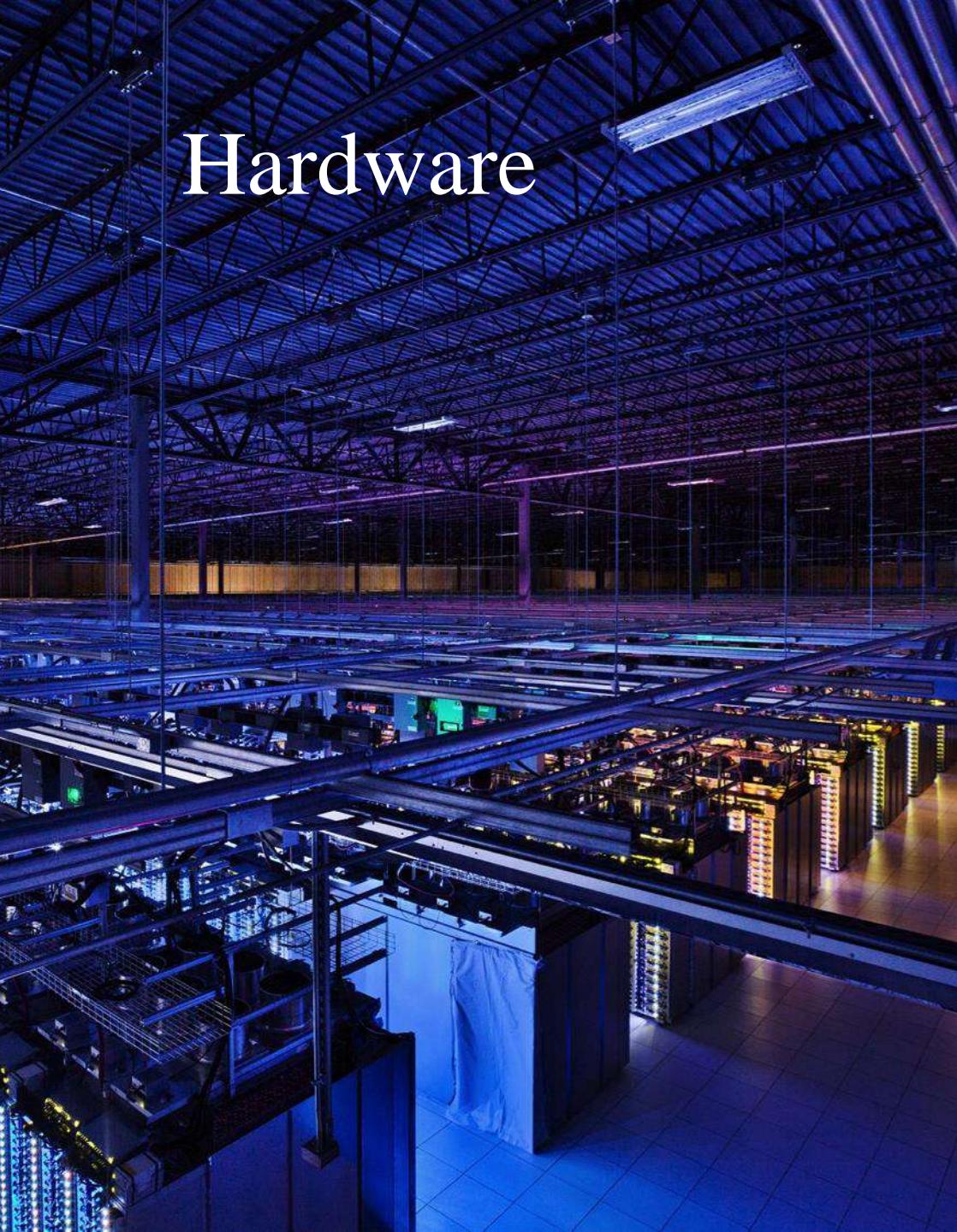


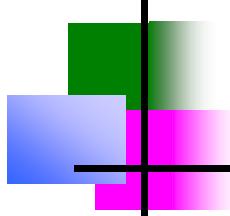
Slides from **Matei Zaharia** [matei@cs.stanford.edu](mailto:matei@cs.stanford.edu)

# Cloud Computing, Big Data



# Hardware

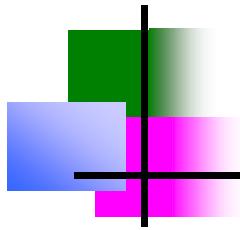




# Google 1997

---





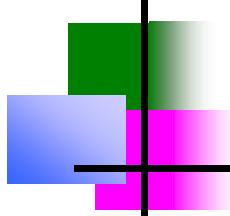
# Data, Data, Data

---

“...**Storage space** must be used efficiently to store indices and, optionally, the documents themselves. The indexing system must process **hundreds of gigabytes** of data efficiently...”

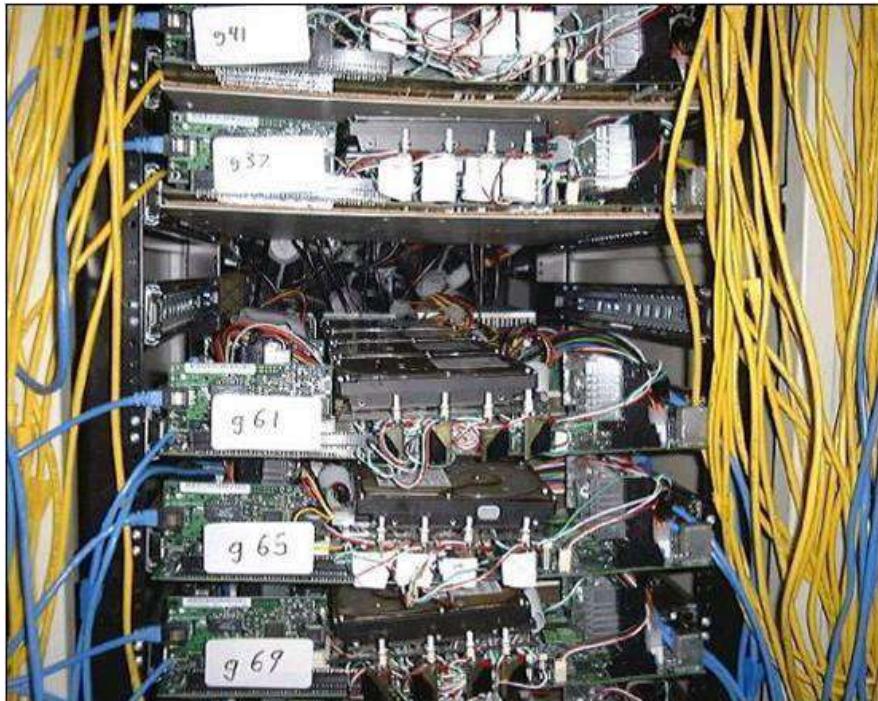
## The Anatomy of a Large-Scale Hypertextual Web Search Engine

Sergey Brin and Lawrence Page



# Google 2001

---



Commodity CPUs

Lots of disks

Low bandwidth network

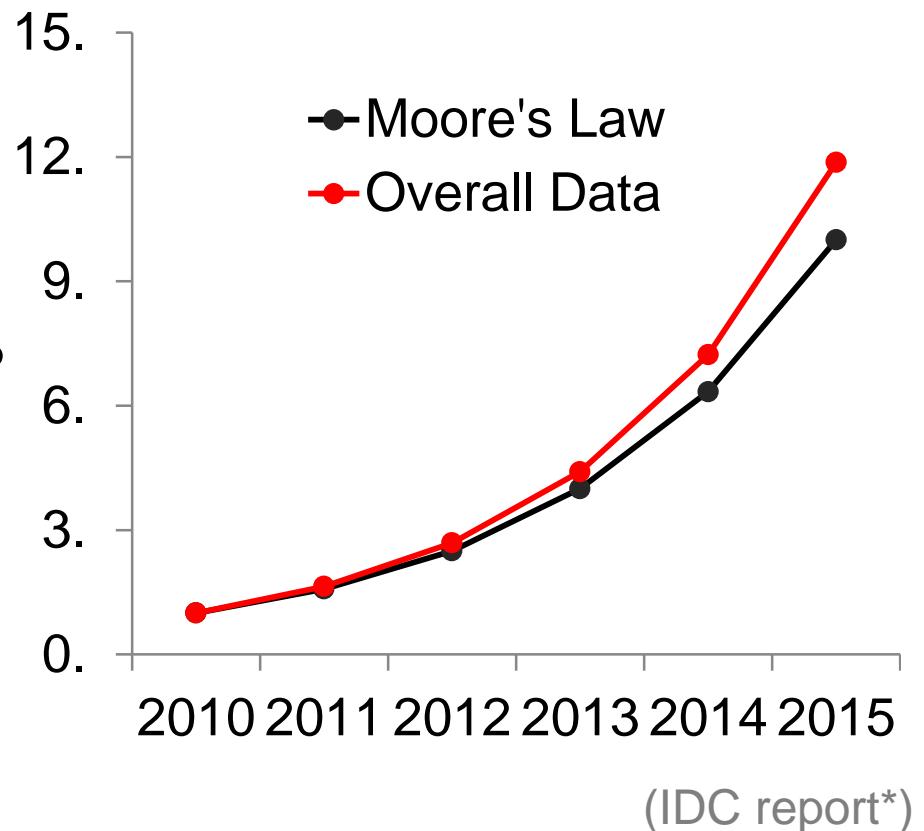
Cheap !

# Datacenter evolution

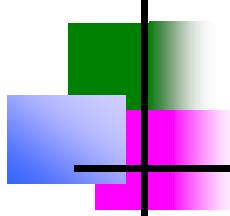
Facebook's daily logs: 60 TB

1000 genomes project: 200 TB

Google web index: 10+ PB



Slide from Ion Stoica

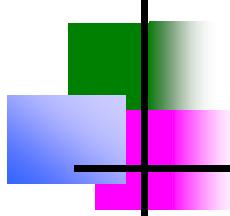


# Datacenter Evolution

---



Google data centers in The Dalles,  
Oregon



# Datacenter Evolution

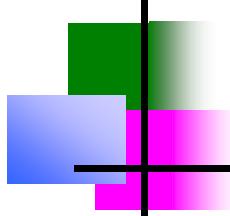
---

Capacity:  
~10000 machines



Bandwidth:  
12-24 disks per  
node

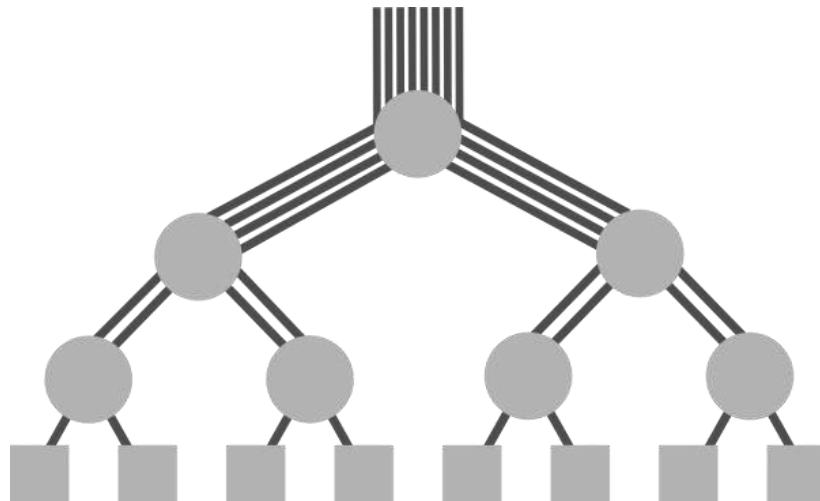
Latency:  
256GB RAM cache



# Datacenter Networking

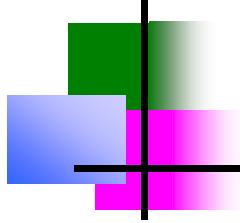
---

Initially tree topology  
Over subscribed links



Fat tree, Bcube, VL2 etc.

Lots of research to get  
full bisection bandwidth



# Datacenter Design

---

Goals

Power usage effectiveness (PUE)

Cost-efficiency

Custom machine design



Open Compute Project  
(Facebook)

# Datacenters → Cloud Computing

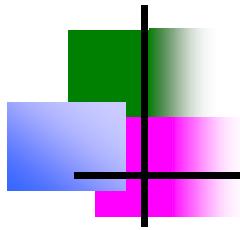
“...long-held dream of computing as a utility...”



## Above the Clouds: A Berkeley View of Cloud Computing

Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz,  
Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia  
*(Comments should be addressed to [abovetheclouds@cs.berkeley.edu](mailto:abovetheclouds@cs.berkeley.edu))*

UC Berkeley Reliable Adaptive Distributed Systems Laboratory \*  
<http://radlab.cs.berkeley.edu/>

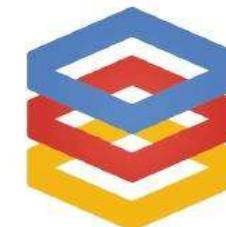


# From Mid 2006

---

Rent virtual computers in the “Cloud”

On-demand machines, spot  
pricing



Google Compute Engine

100's of  
Enterprise  
Customers

# About GigaSpaces



# Big Data In The Cloud

2.7 ZB

Global Digital Data

0.5

Petabytes

Two years tweets

66%

Will or plan to use Big  
Data in the cloud

43%

think that data  
analytics could be improved in their  
organization if data analytics was part of  
**cloud services**



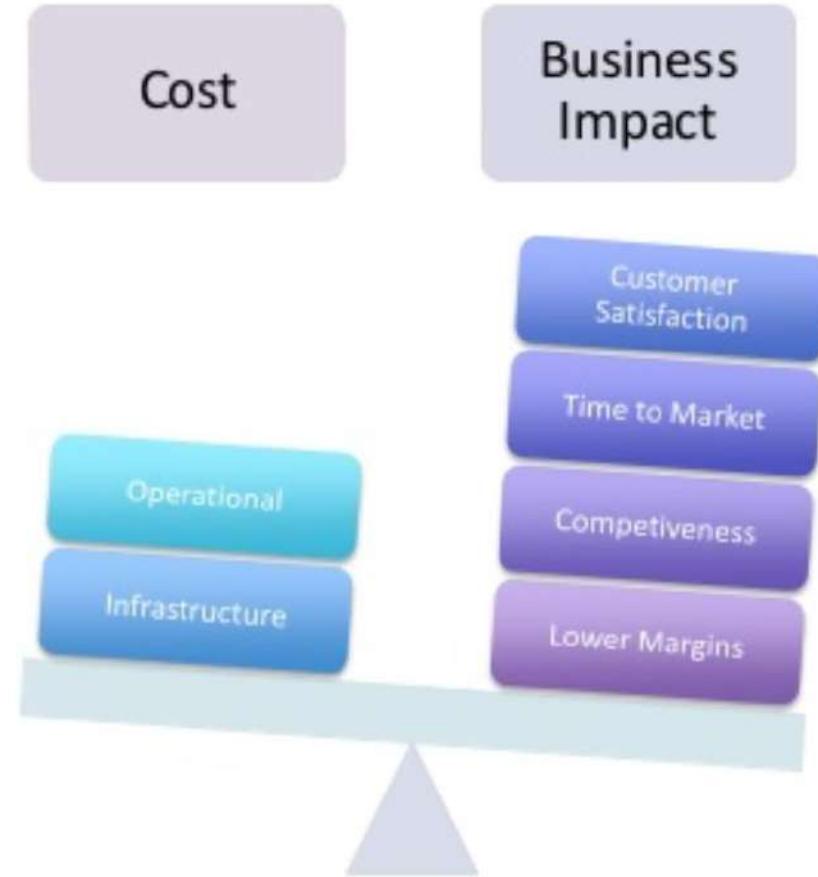
## The Challenges..

Ever Growing Data

Deeper Correlation

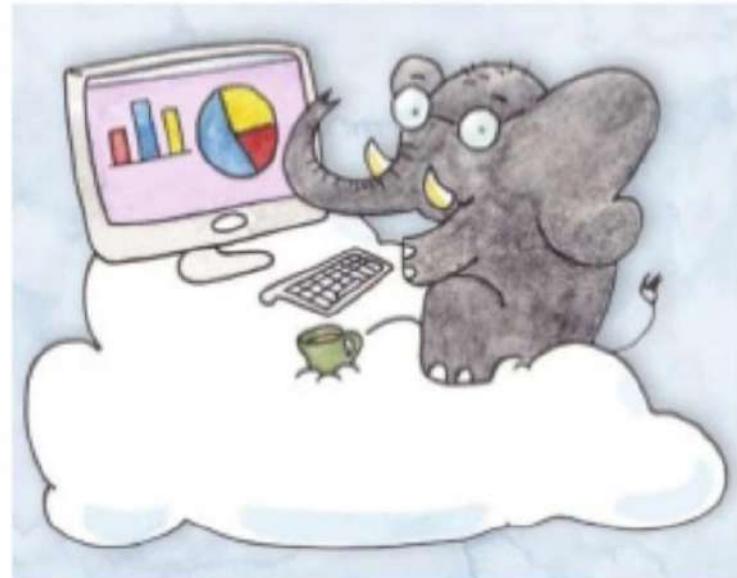
Tight Performance

# The Challenge



# The Solution

Big Data  
in the **Cloud**



# Big Data in the Cloud- 3 Reasons



Holger Kisker

Forbes

- **Skills**
  - Do you really need/want this all in-house?
- **Huge amounts of external data.**
  - Does it make sense to move and manage all this data behind your firewall?
- **Focus on the value of your data**
  - Instead of big data management.

# Managing Big Data on the Cloud



- Auto start VMs
- Install and configure app components
- Monitor
- Repair
- (Auto) Scale
- Burst...

# Big Data in the Cloud..

Reduce the  
Infrastructure  
Cost



Choose the Right Cloud  
for the Job

Running Bare-Metal for  
high I/O workloads, Public  
cloud for sporadic  
workloads

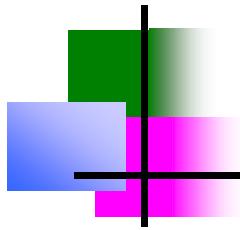
# Big Data in the Cloud ..

## Reducing The Operational Complexity



- Consistent Management
- Automation Through the Entire Stack



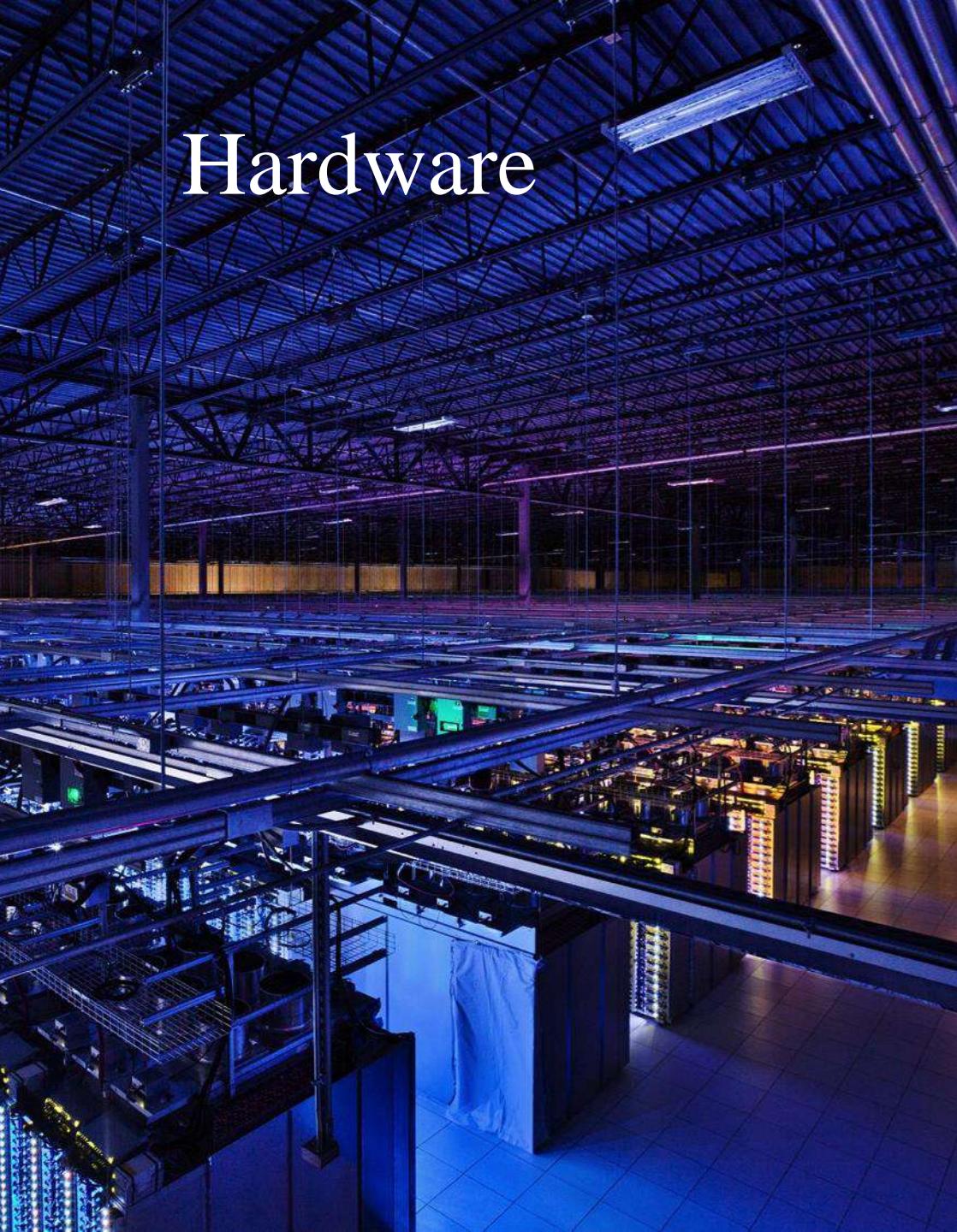


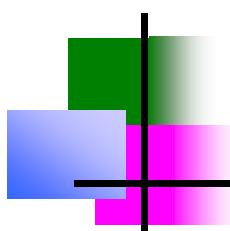
# Amazon EC2

Machine	Memory (GB)	Compute Units (ECU)	Local Storage (GB)	Cost / hour
t1.micro	0.615	2	0	\$0.02
m1.xlarge	15	8	1680	\$0.48
cc2.8xlarge	60.5	88 (Xeon 2670)	3360	\$2.40

1 ECU = CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor

# Hardware





# Hopper vs. Datacenter

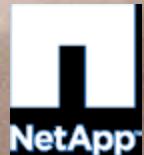
	Hopper	Datacenter <sup>2</sup>
Nodes	6384	1000s to 10000s
CPUs (per node)	2x12 cores	~2x6 cores
Memory (per node)	32-64GB	~48-128GB
Storage (overall)	~4 PB	120-480 PB
Interconnect	~ 66.4 Gbps	~10Gbps

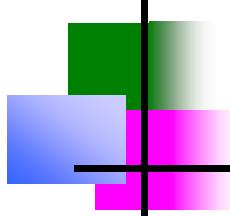
<sup>2</sup><http://blog.cloudera.com/blog/2013/08/how-to-select-the-right-hardware-for-your-new-hadoop-cluster/>  
Big Data Analytics



# Paradigm Shift in Computing

## Azure Services Platform



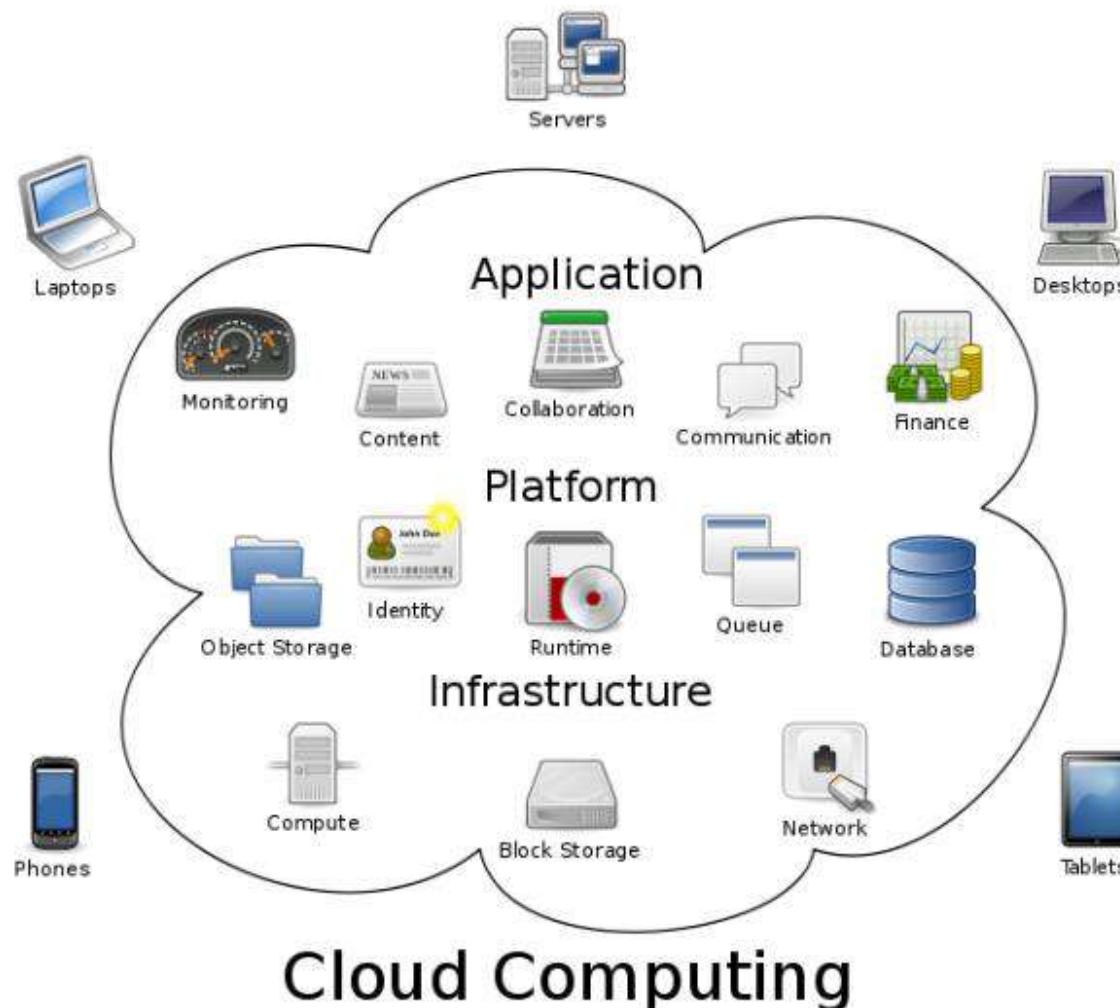


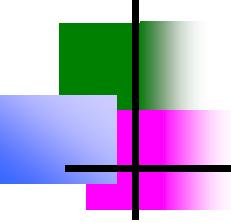
# What is Cloud Computing?

---

- 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, ...

# What is Cloud Computing?





# Cloud Computing: History

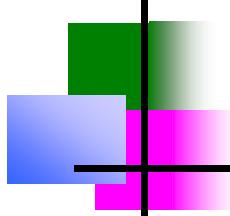
---

“

If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility... The computer utility could become the basis of a new and important industry.

”

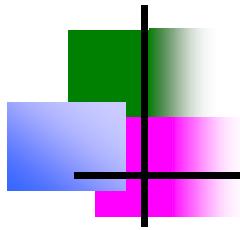
—[John McCarthy](#), speaking at the MIT Centennial in 1961<sup>[2]</sup>



# Cloud Computing: Why Now?

---

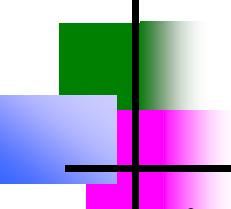
- Experience with very large datacenters
  - Unprecedented economies of scale
  - Transfer of risk
- Technology factors
  - Pervasive broadband Internet
  - Maturity in Virtualization Technology
- Business factors
  - Minimal capital expenditure
  - Pay-as-you-go billing model



# Benefits

---

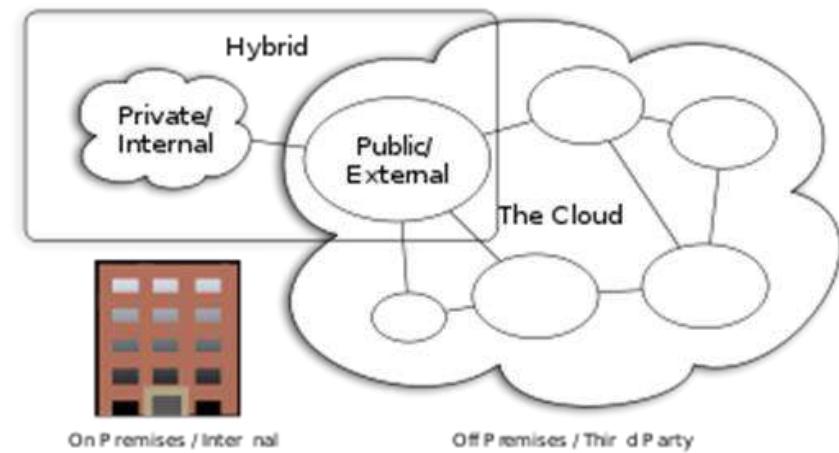
- Cost & management
  - Economies of scale, “out-sourced” resource management
- Reduced Time to deployment
  - Ease of assembly, works “out of the box”
- Scaling
  - On demand provisioning, co-locate data and compute
- Reliability
  - Massive, redundant, shared resources
- Sustainability
  - Hardware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

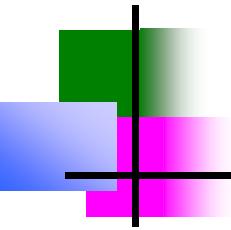


# Types of Cloud Computing

---

- **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 organisations.
- **Hybrid Cloud:** Organisations host some critical, secure applications in private clouds. The not so critical applications are hosted in the public cloud
  - **Cloud bursting:** the organisation 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](#).

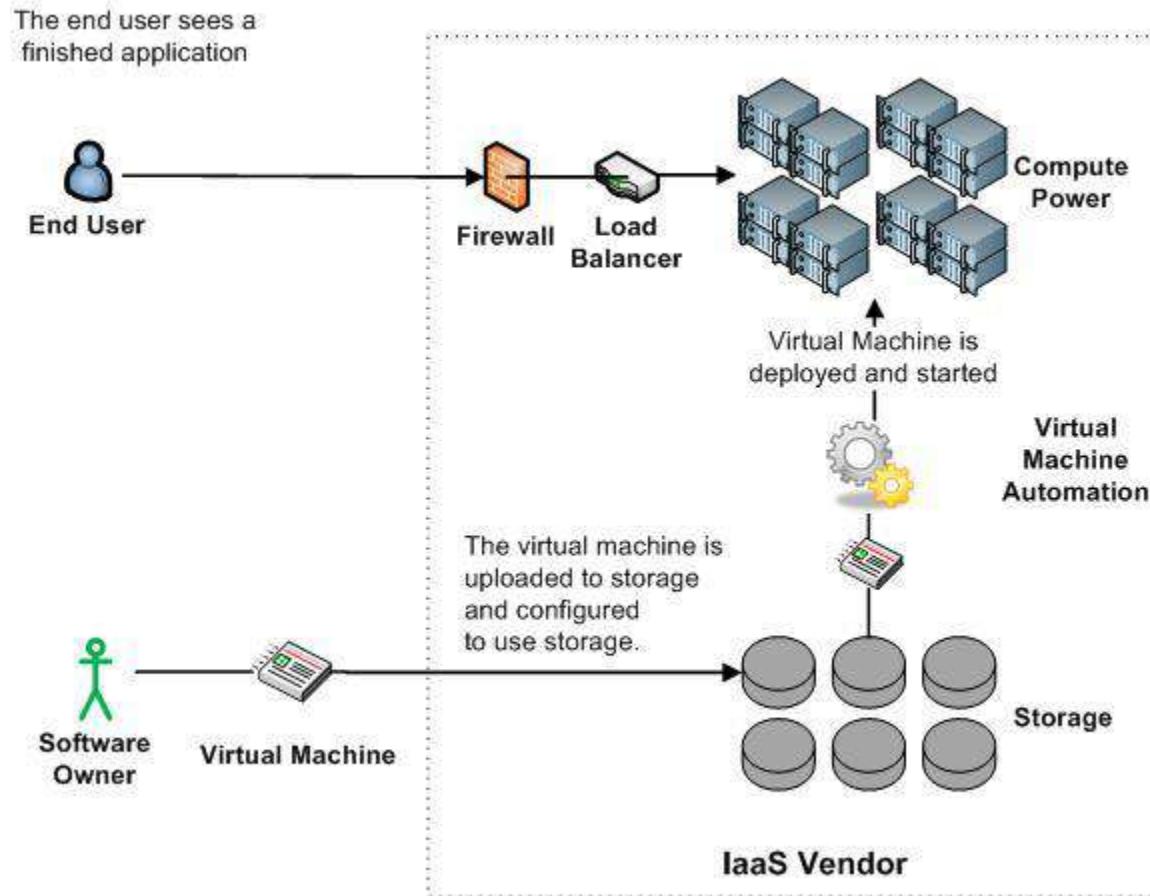
- **Platform as a Service (PaaS)**

- 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](#).

# Infrastructure as a Service (IaaS)



# 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

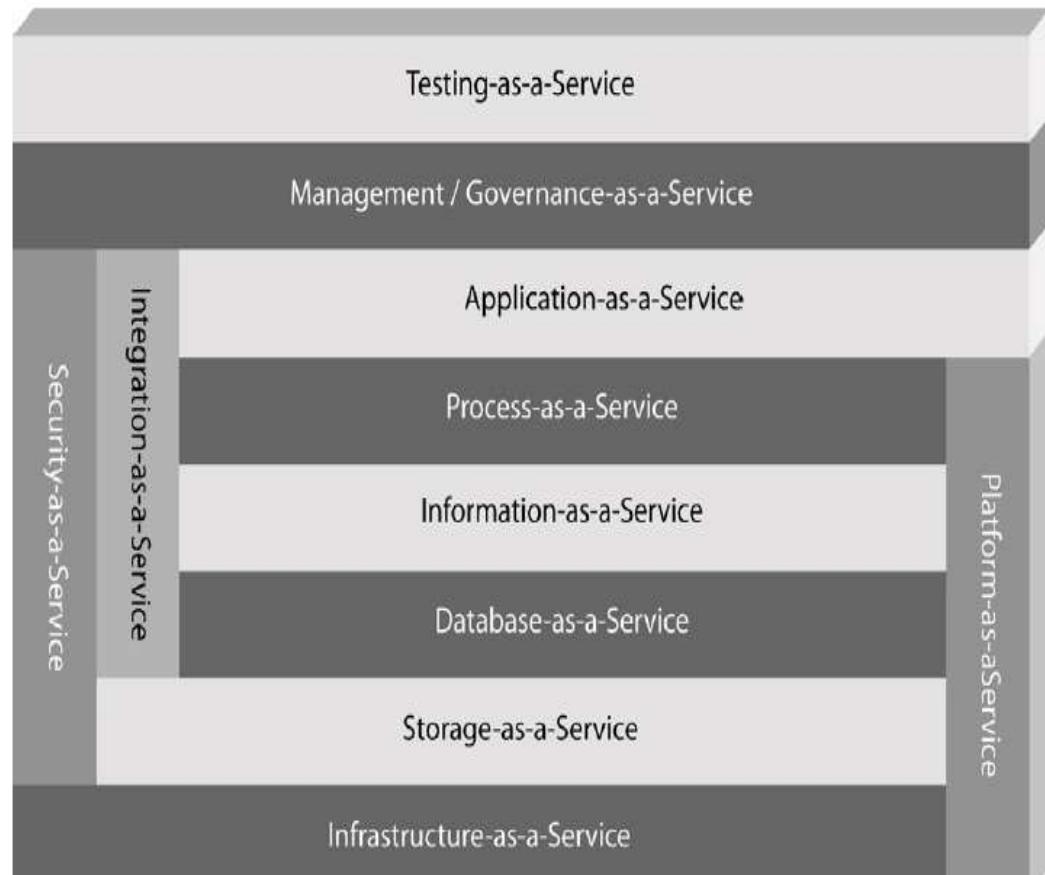
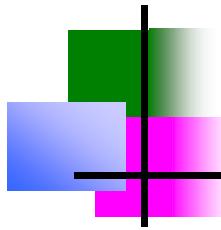


Figure 1: The patterns or categories of cloud computing providers allow you to use a discrete set of services within your architecture.

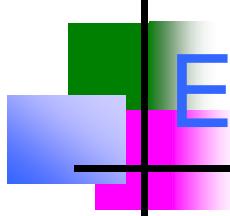
InfoWorld Cloud Computing Deep Dive



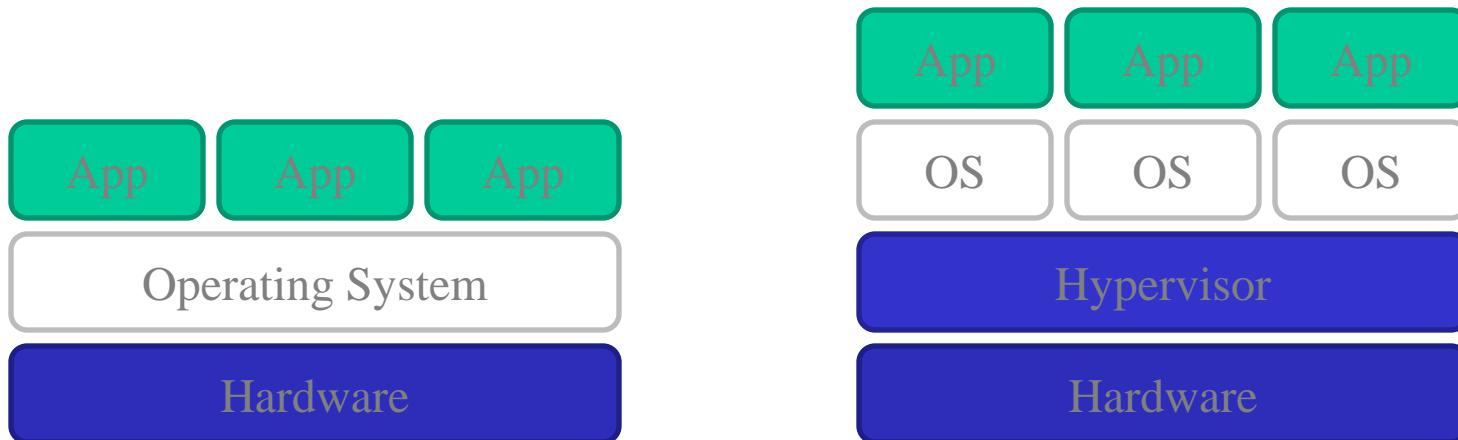
# Key Ingredients in Cloud Computing

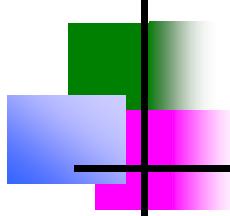
---

- 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 Service)
- Web Services in Cloud



# Enabling Technology: Virtualization

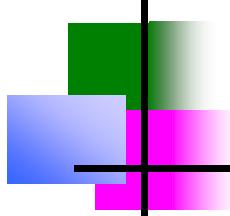




# Everything as a Service

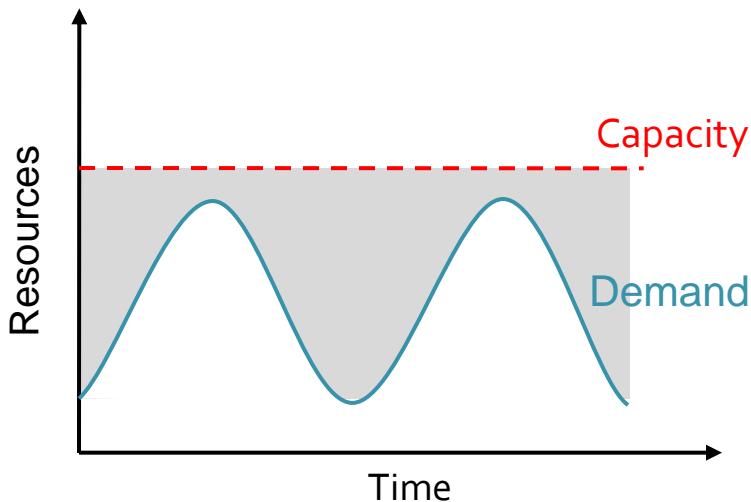
---

- Utility computing = Infrastructure as a Service (IaaS)
  - Why buy machines when you can rent cycles?
  - Examples: Amazon's EC2, Rackspace
- Platform as a Service (PaaS)
  - Give me nice API and take care of the maintenance, upgrades, ...
  - Example: Google App Engine
- Software as a Service (SaaS)
  - Just run it for me!
  - Example: Gmail, Salesforce

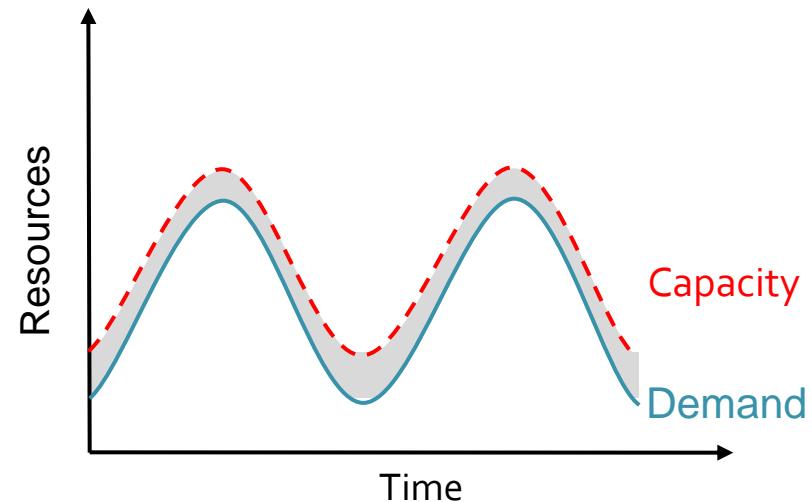


# Economics of Cloud Users

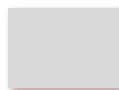
- Pay by use instead of provisioning for peak



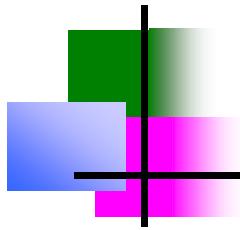
Static data center



Data center in the cloud



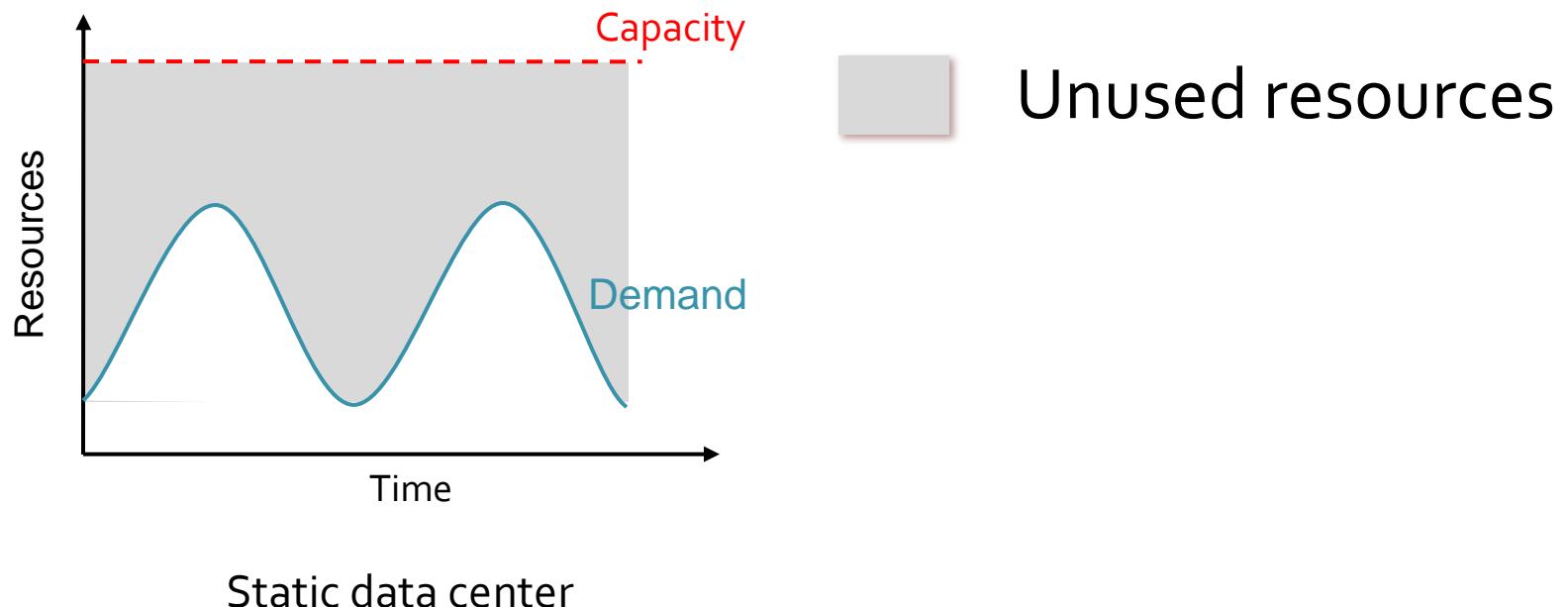
Unused resources

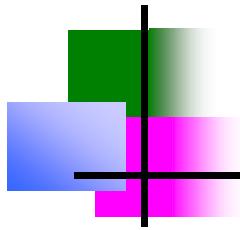


# Economics of Cloud Users

---

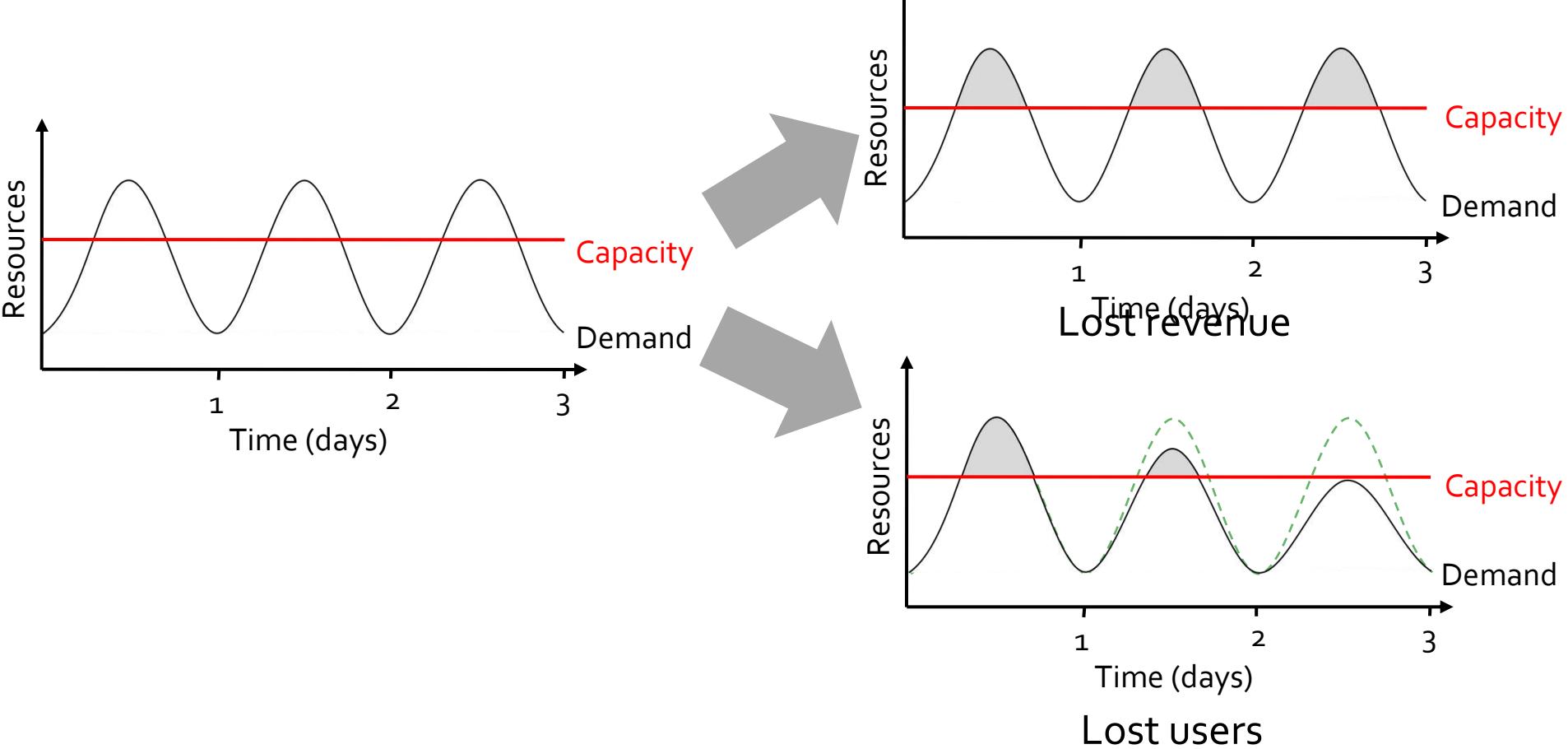
- Risk of over-provisioning: underutilization

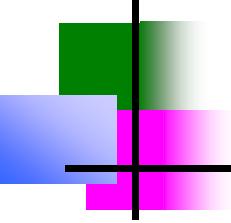




# Economics of Cloud Users

- Heavy penalty for under-provisioning

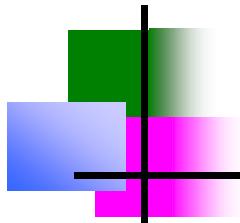




# The Big Picture

---

- Unlike the earlier attempts:
  - Distributed Computing
  - Distributed Databases
  - Grid Computing
- Cloud Computing is likely to persist:
  - Organic growth: Google, Yahoo, Microsoft, and Amazon
  - Poised to be an integral aspect of National Infrastructure in US and other countries



# Cloud Reality

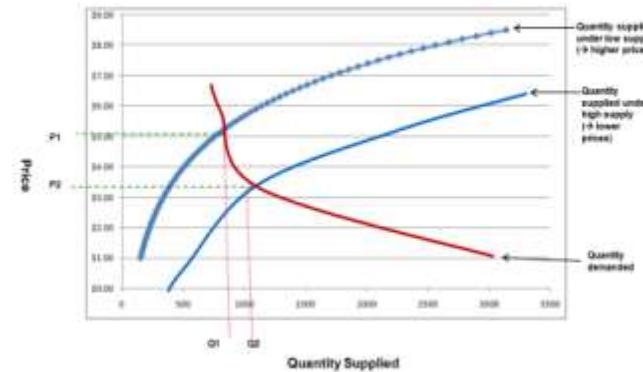
---

- Facebook Generation of Application Developers
- Animoto.com:
  - Started with 50 servers on Amazon EC2
  - Growth of 25,000 users/hour
  - Needed to scale to 3,500 servers in 2 days  
(RightScale@SantaBarbara)
- Many similar stories:
  - RightScale
  - Joyent
  - ...

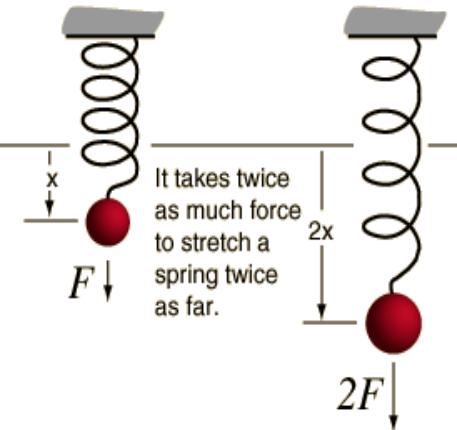
# Cloud Challenges: Elasticity



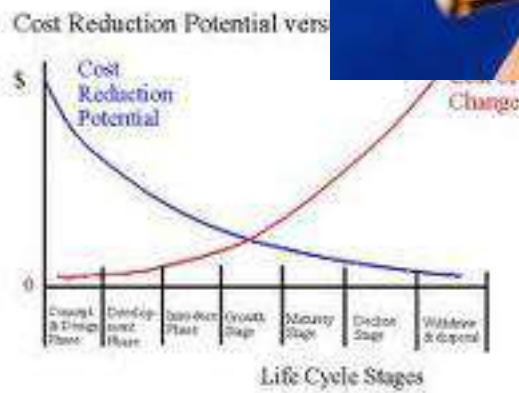
Equilibrium Prices Under Low Price Elasticity



Hooke's Law:  
 $F_{spring} = -kx$   
Spring constant  $k$



# Cloud Challenges: Differential Pricing Models

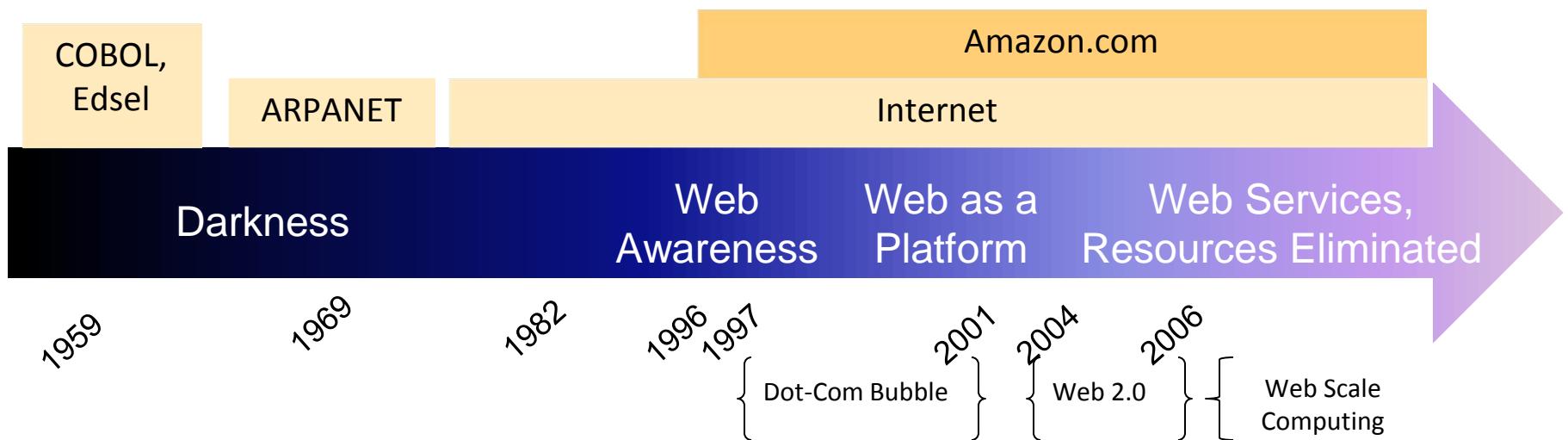


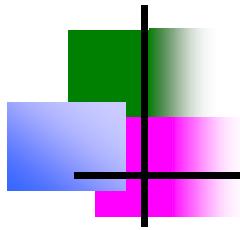
\* Adapted from CIO.com Exhibit 2, p. 15



# The Obligatory Timeline Slide

## (Mike Culver @ AWS)





# AWS

---

- Elastic Compute Cloud – EC2 (IaaS)
- Simple Storage Service – S3 (IaaS)
- Elastic Block Storage – EBS (IaaS)
- SimpleDB (SDB) (PaaS)
- Simple Queue Service – SQS (PaaS)
- CloudFront (S3 based Content Delivery Network – PaaS)
- Consistent AWS Web Services API

# What does Azure platform offer to developers?

## Your Applications



Service Bus

Workflow

Access Control

...

Compute

Storage

Manage

...



Database

Analytics

Reporting

...



Identity

Contacts

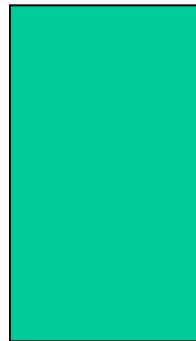
Devices

...



# Google's AppEngine vs Amazon's EC2

Python  
BigTable  
Other API's



VMs  
Flat File Storage

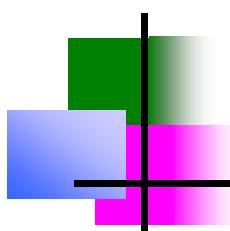


## AppEngine:

- Higher-level functionality  
(e.g., automatic scaling)
- More restrictive  
(e.g., respond to URL only)
- Proprietary lock-in

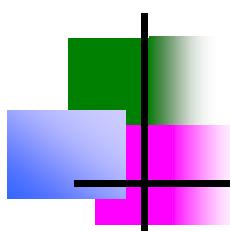
## EC2/S3:

- Lower-level functionality
- More flexible
- Coarser billing model



---

Thank you !!!



# **Chapter2**

# **Technologies for handing of Big Data**

## **GFS Vs HDFS**

**Basanta Joshi, PhD**

Asst. Prof., Depart of Electronics and Computer Engineering  
Program Coordinator, MSc in Information and Communication Engineering  
Member, Laboratory for ICT Research and Development (LICT)

Member, Research Management Cell (RMC)

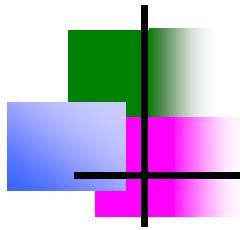
Institute of Engineering

[basanta@ioe.edu.np](mailto:basanta@ioe.edu.np)

<http://www.basantajoshi.com.np>

<https://scholar.google.com/citations?user=iocLiGcAAAAJ>

[https://www.researchgate.net/profile/Basanta\\_Joshi2](https://www.researchgate.net/profile/Basanta_Joshi2)



# Outline

---

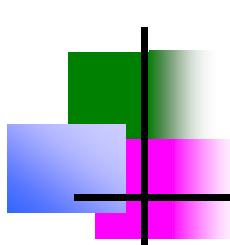
1. Why do we need a Distributed File System?

2. What is a Distributed File System?

3. Google File System (GFS)

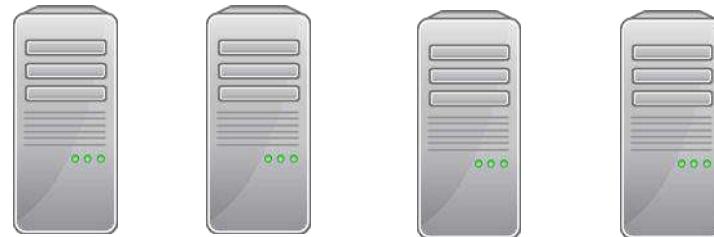
4. Hadoop Distributed File System (HDFS)

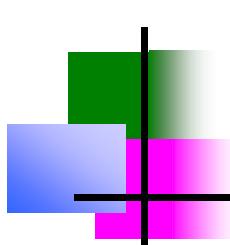
5. GFS Vs HDFS



# Why do we need a Distributed File System?

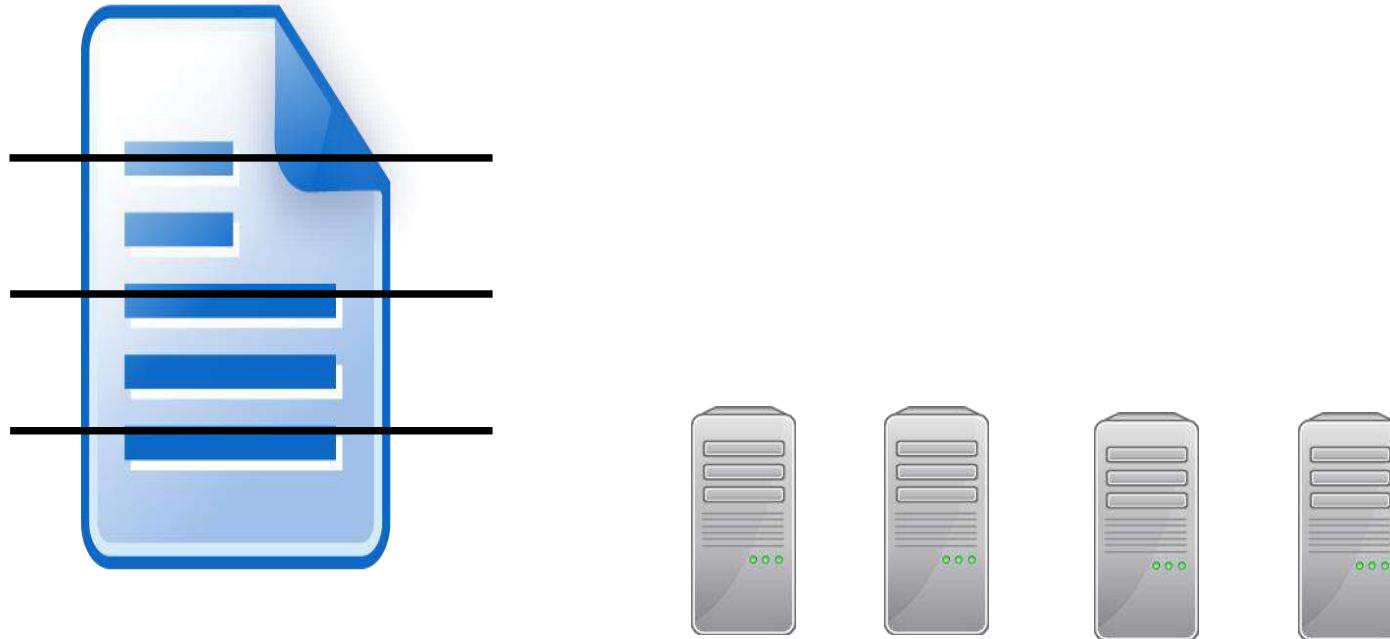
---



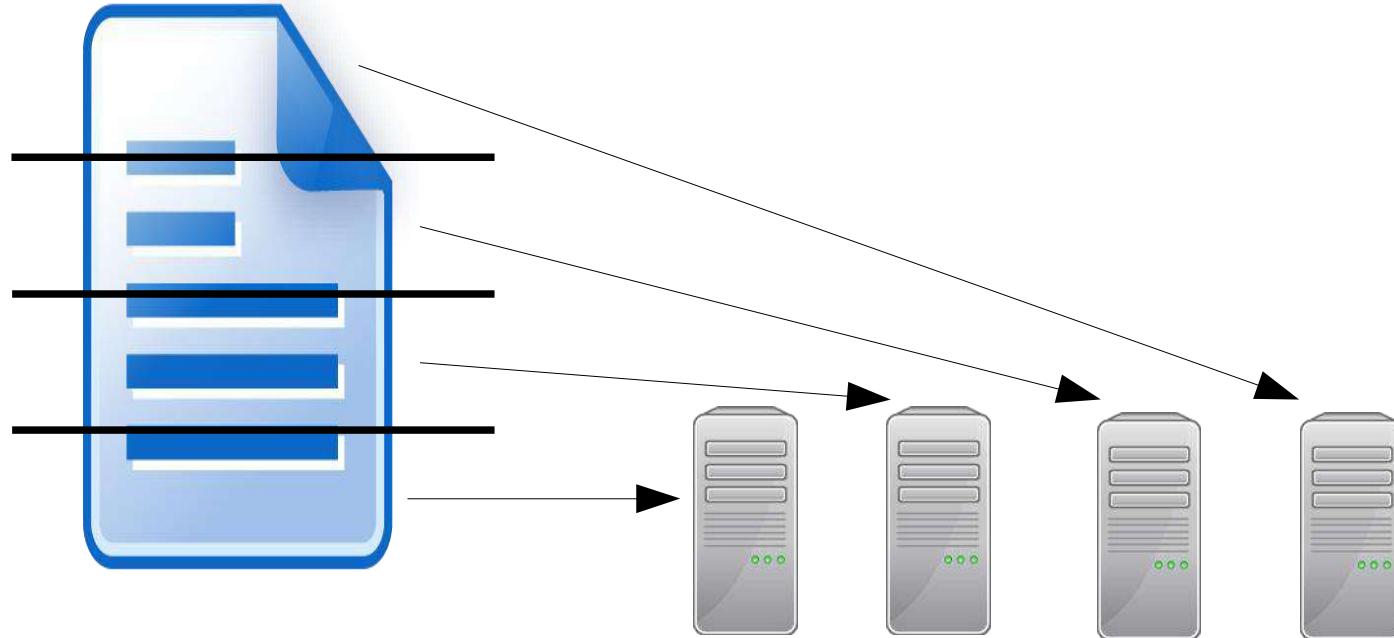


# Why do we need a Distributed File System?

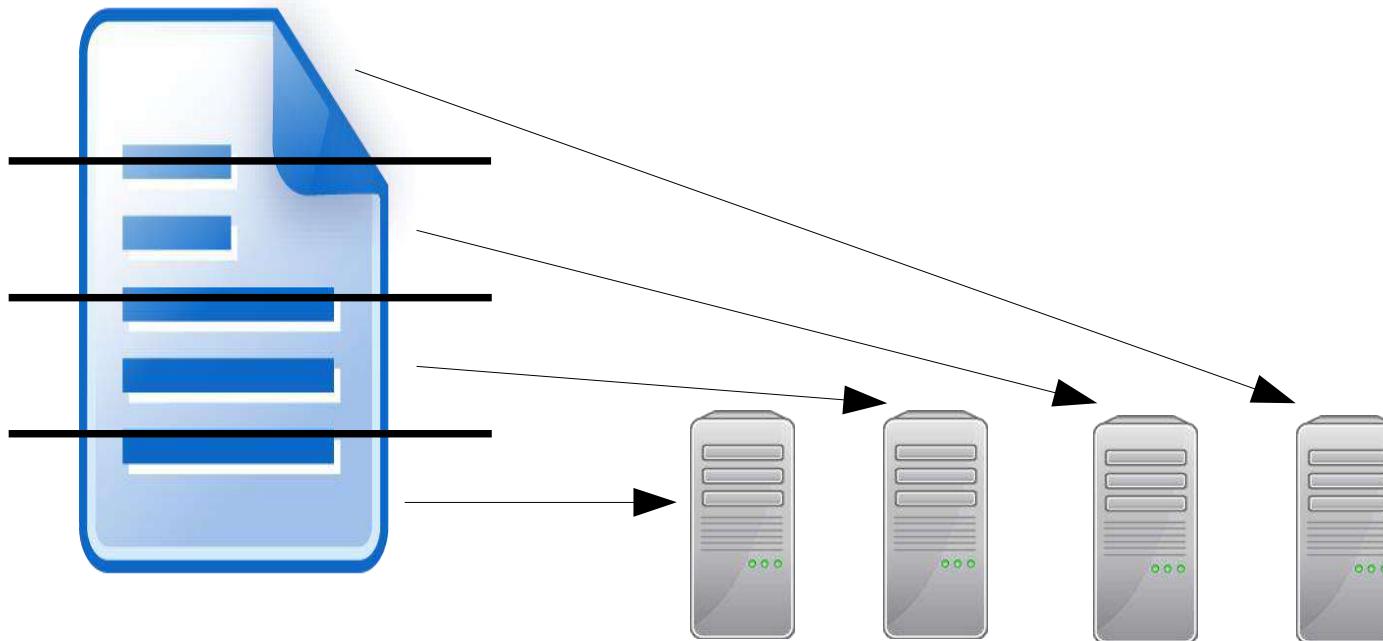
---



# Why do we need a Distributed File System?



# Why do we need a Distributed File System?

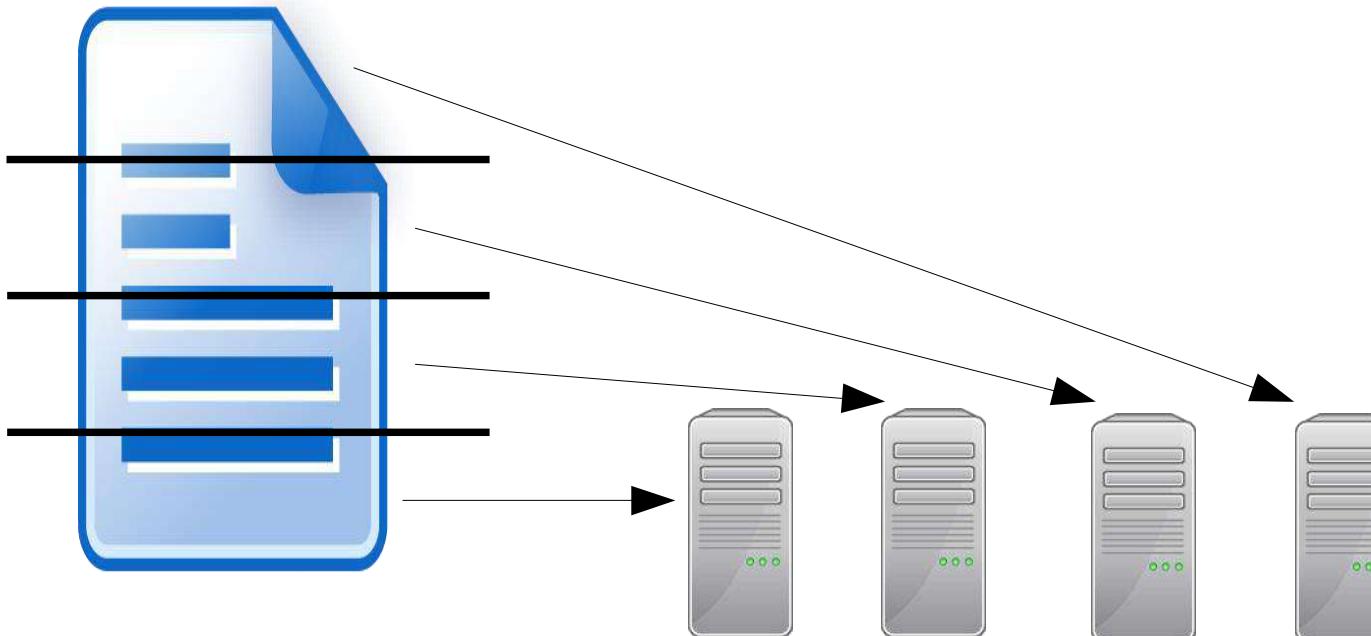


Read???

- Whole File?
- Specific part?



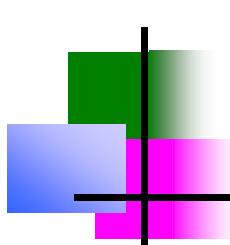
# Why do we need a Distributed File System?



Write???

- Append to the end of the file?
- Insert content in the middle?



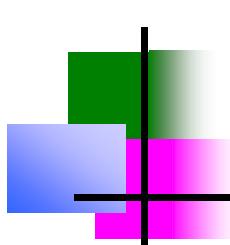


# Why do we need a Distributed File System?

---

We want to:

- △ Read large data fast
- △ **scalability**: perform multiple parallel reads and writes

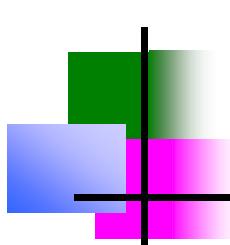


# Why do we need a Distributed File System?

---

We want to:

- △ Read large data fast
- △ scalability: perform multiple parallel reads and writes
  
- △ Have the files available even if one computer crashes
- △ fault tolerance: replication

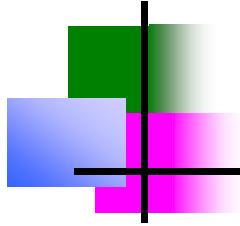


# Why do we need a Distributed File System?

---

We want to:

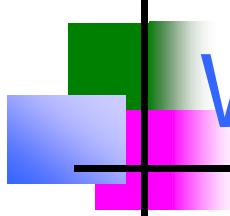
- △ Read large data fast
- △ scalability: perform multiple parallel reads and writes
- △ Have the files available even if one computer crashes
- △ fault tolerance: replication
- △ Hide parallelization and distribution details
- △ transparency: clients can access it like a local filesystem



# Outline

---

1. Why do we need a Distributed File System?
2. What is a Distributed File System?
3. GFS and HDFS
4. Hadoop Distributed File System (HDFS)

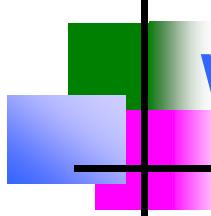


# What is a Distributed File System?

---

## DEFINITIONS:

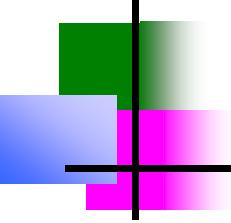
- A **Distributed File System** ( DFS ) is simply a classical model of a file system distributed across multiple machines. The purpose is to promote sharing of dispersed files.
- This is an area of active research interest today.
- The resources on a particular machine are **local** to itself. Resources on other machines are **remote**.
- A file system provides a service for clients. The server interface is the normal set of file operations: create, read, etc. on files.



# What is a Distributed File System?

---

- Distributed file systems support the sharing of information in the form of files throughout the intranet.
- A distributed file system enables programs to store and access remote files exactly as they do on local ones, allowing users to access files from any computer on the intranet.
- Recent advances in higher bandwidth connectivity of switched local networks and disk organization have lead high performance and highly scalable file systems.



# What is a Distributed File System?

---

## Definitions

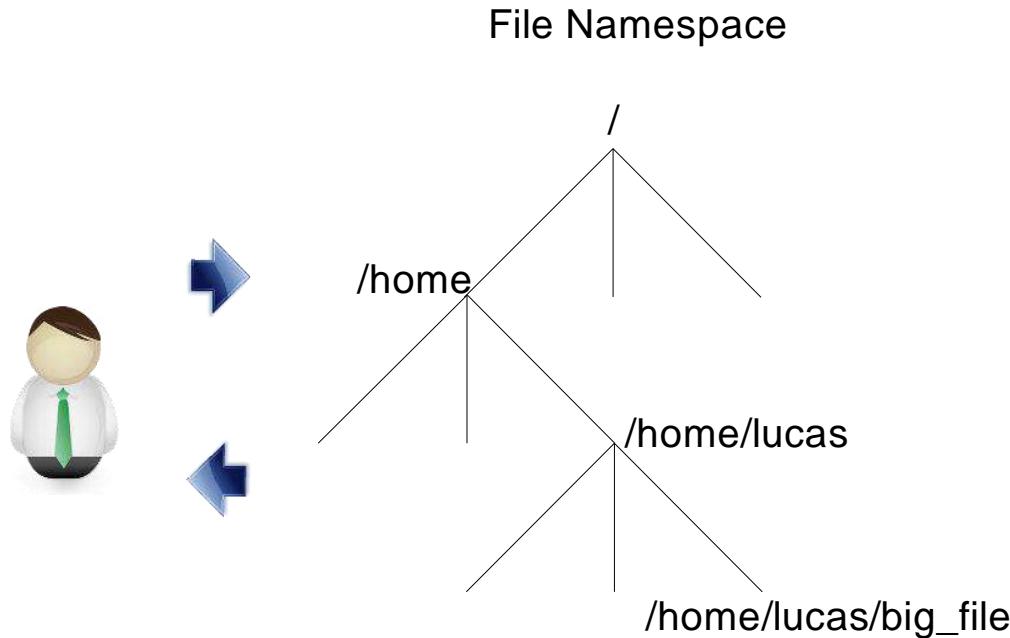
Clients, servers, and storage are dispersed across machines. Configuration and implementation may vary -

- a) Servers may run on dedicated machines, OR
- b) Servers and clients can be on the same machines.
- c) The OS itself can be distributed with the file system a part of that distribution.
- d) A distribution layer can be interposed between a conventional OS and the file system.

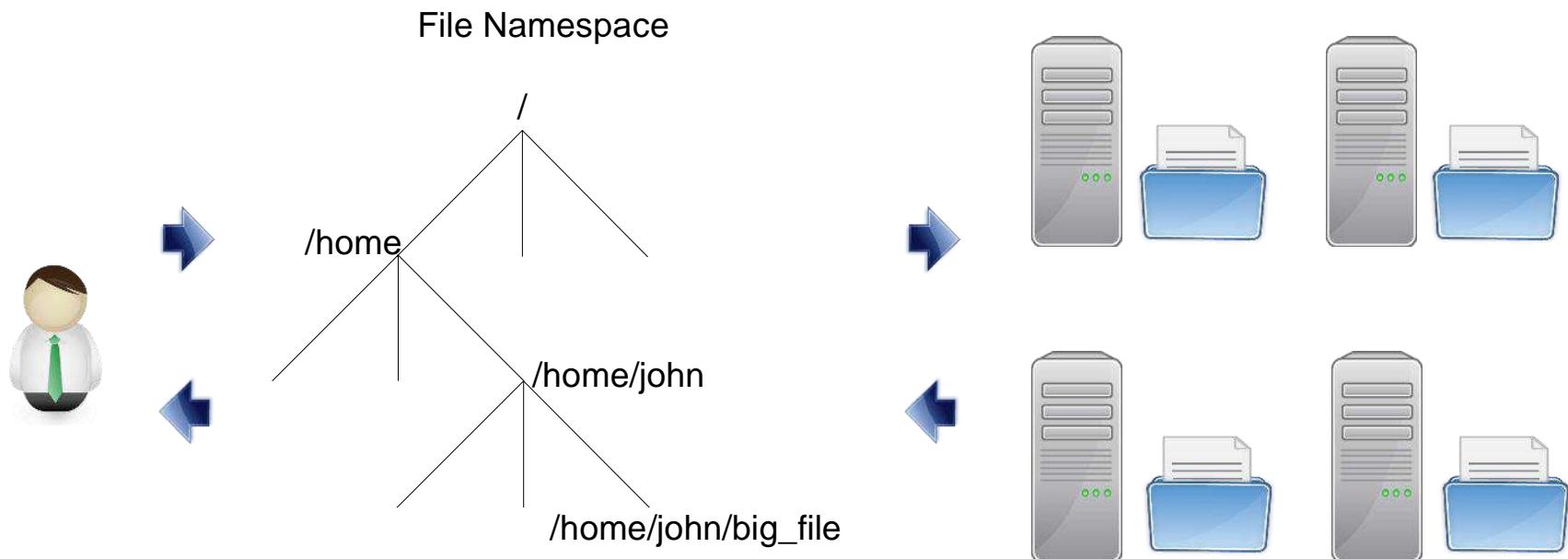
Clients should view a DFS the same way they would a centralized FS; the distribution is hidden at a lower level.

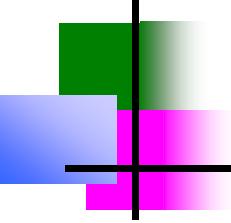
Performance is concerned with throughput and response time.

# What is a Distributed File System?



# What is a Distributed File System?

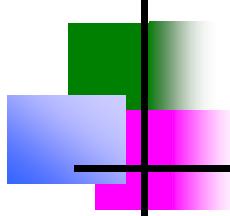




# Examples

---

- ▲ Windows Distributed File System (DFS; Microsoft, 1996)
- ▲ GFS (Google, 2003)
- ▲ Lustre (Cluster File Systems, 2003)
- ▲ BeeGFS (Fraunhofer, 2005)
- ▲ HDFS (Apache Software Foundation, 2006)
- ▲ GlusterFS (Red Hat, 2007)
- ▲ Ceph (Inktank/Red Hat, 2007)
- ▲ MooseFS (Core Technology/Gemius, 2008)
- ▲ MapR File System (MapR Technologies, 2010)

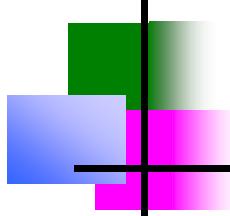


# Components

---

A typical distributed filesystem contains the following components

- △ Clients - they interface with the user

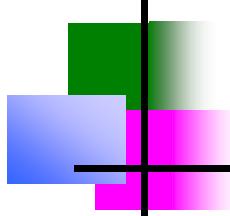


# Components

---

A typical distributed filesystem contains the following components

- ▲ Clients - they interface with the user
- ▲ Chunk nodes - stores chunks of files

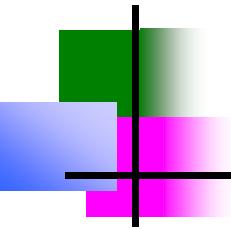


# Components

---

A typical distributed filesystem contains the following components

- ▲ Clients - they interface with the user
- ▲ Chunk nodes - stores chunks of files
- ▲ Master node - stores which parts of each file are on which chunk node



# Presentation

---

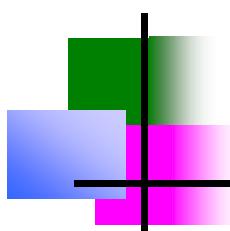
<b>SN</b>	<b>FN</b>	<b>LN</b>	<b>Title</b>	<b>Date</b>
1	Shishir	Subedi	The Google File System	10/11/2020
2	Bibek	Shrestha	The Hadoop Distributed File System	10/11/2020
3	Sagarman	Shrestha	Bigtable: A Distributed Storage System for Structured Data	10/18/2020

# GFS Vs HDFS

Hadoop Distributed File System HDFS	Google File System GFS
Cross Platform	Linux
Developed in Java environment	Developed in C,C++ environment
Initially it was developed by Yahoo and now its an <b>open source Framework</b>	It was developed & still owned by Google
It has Name node and Data Node	It has Master-node and Chunk server

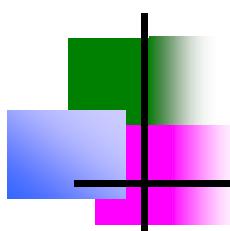
# GFS Vs HDFS

Hadoop Distributed File System HDFS	Google File System GFS
Cross Platform	Linux
Developed in Java environment	Developed in C,C++ environment
Initially it was developed by Yahoo and now its an <b>open source Framework</b>	It was developed & still owned by Google
It has Name node and Data Node	It has Master-node and Chunk server
<b>128 MB</b> will be the default block size	<b>64 MB</b> will be the default block size
Name node receive heartbeat from Data node	Master node receive heartbeat from Chunk server
Commodity hardware are used	Commodity hardware are used
"Write Once and Read Many" times model	Multiple writer , multiple reader model
Deleted files are renamed into particular folder and then it will removed via garbage	Deleted files are not reclaimed immediately and are renamed in hidden name space and it will deleted after three days if it's not in use
Edit Log is maintained	Operational Log is maintained
Only append is possible	Random file write possible



---

Thank you !!!



# **Chapter3**

# **Understanding Big Data**

# **Technology Foundations**

## **Big Data Stack**

**Basanta Joshi, PhD**

Asst. Prof., Depart of Electronics and Computer Engineering

Program Coordinator, MSc in Information and Communication Engineering

Member, Laboratory for ICT Research and Development (LICT)

Member, Research Management Cell (RMC)

Institute of Engineering

[basanta@ioe.edu.np](mailto:basanta@ioe.edu.np)

<http://www.basantajoshi.com.np>

<https://scholar.google.com/citations?user=iocLiGcAAAAJ>

[https://www.researchgate.net/profile/Basanta\\_Joshi2](https://www.researchgate.net/profile/Basanta_Joshi2)

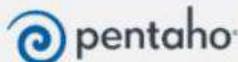
# Big data Stack

VISUALIZATION  
LAYER AND API'S



RESTful API

ANALYTICAL TOOLS



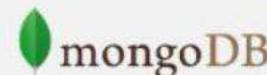
ANALYSIS LAYER



PROCESSING  
LAYER



STORAGE LAYER



REAL TIME  
EVENT ENGINE



INTEGRATION LAYER  
(ETL)



SOURCES



Logs

XML



CAPA DE GESTIÓN



HERRAMIENTAS  
ADMINISTRACIÓN



HERRAMIENTAS  
MONITORIZACIÓN



Nagios

HERRAMIENTAS  
DIAGNÓSTICO



# What We Need?

- Store
- Join
- Index
- Analytics
- Aggregate
- Visualize

# Challenge

The challenge in big data analytics is to

- dig deeply
- quickly (real time?)
- and widely

# "ilities" or NFR?

- Availability
- Scalability
- Security
- Performance
- ...

# Solution?

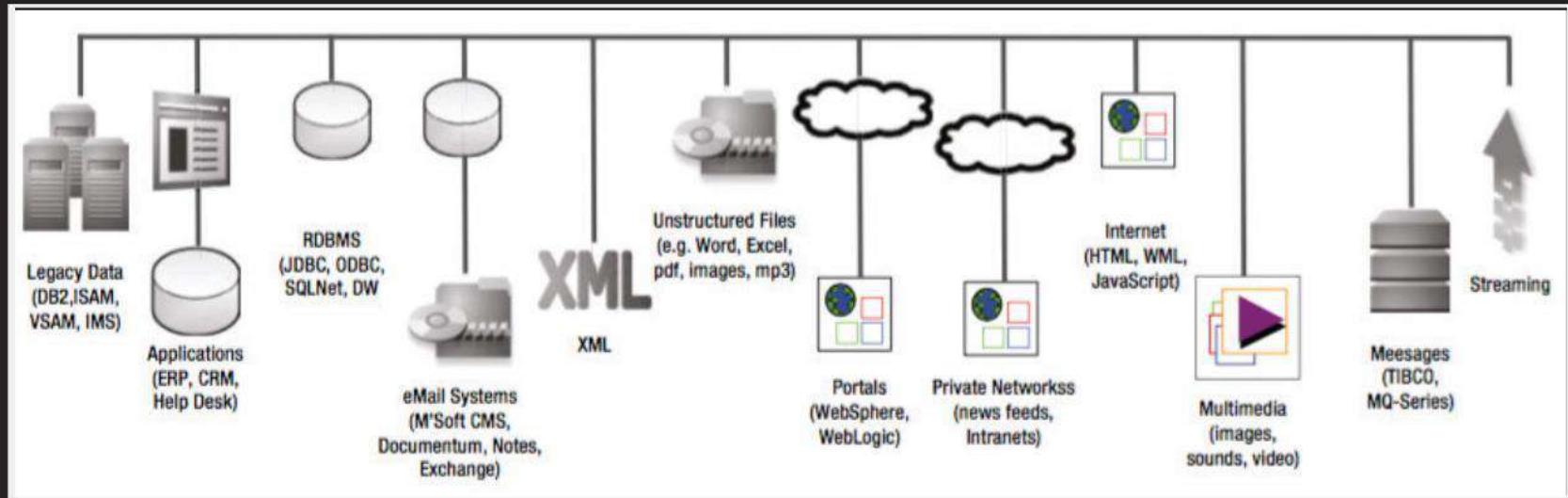
# Big Data Tech Stack

# What're essential components?

# Data Sources

Multiple internal  
& external  
data sources

Creates a  
data lake



## **Legacy Data Sources**

**HTTP/HTTPS web services**

**RDBMS**

**FTP**

**JMS/MQ based services**

**Text/flat file/csv logs**

**XML data sources**

**IM Protocol requests**

## **New Age Data Sources**

### **High Volume Sources**

1. Switching devices data
2. Access point data messages
3. Call data record due to exponential growth in user base
4. Feeds from social networking sites

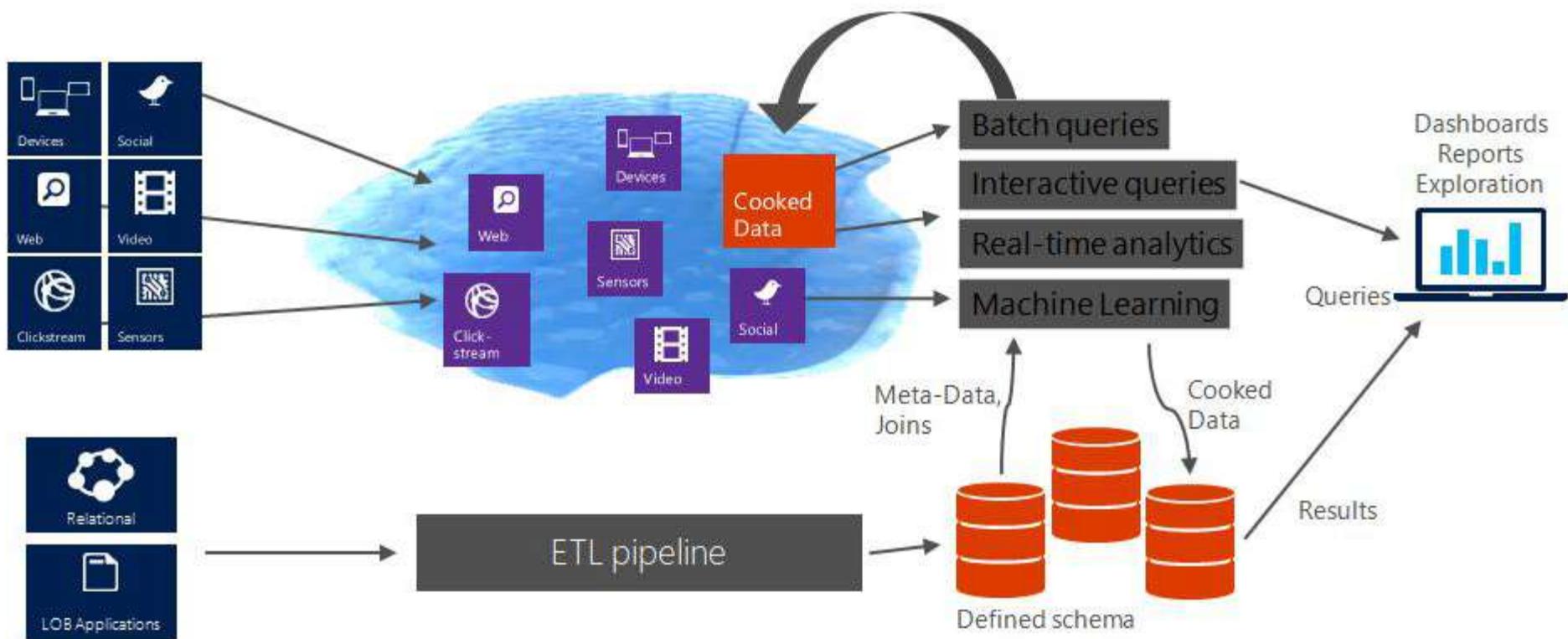
### **Variety of Sources**

1. Image and video feeds from social Networking sites
2. Transaction data
3. GPS data
4. Call center voice feeds
5. E-mail
6. SMS

### **High Velocity Sources**

1. Call data records
2. Social networking site conversations
3. GPS data
4. Call center - voice-to-text feeds

# The data lake and warehouse



Different  
Volume, Variety,  
Velocity

Aim is to create  
a **funnel** after  
proper **validation**  
and **cleaning**

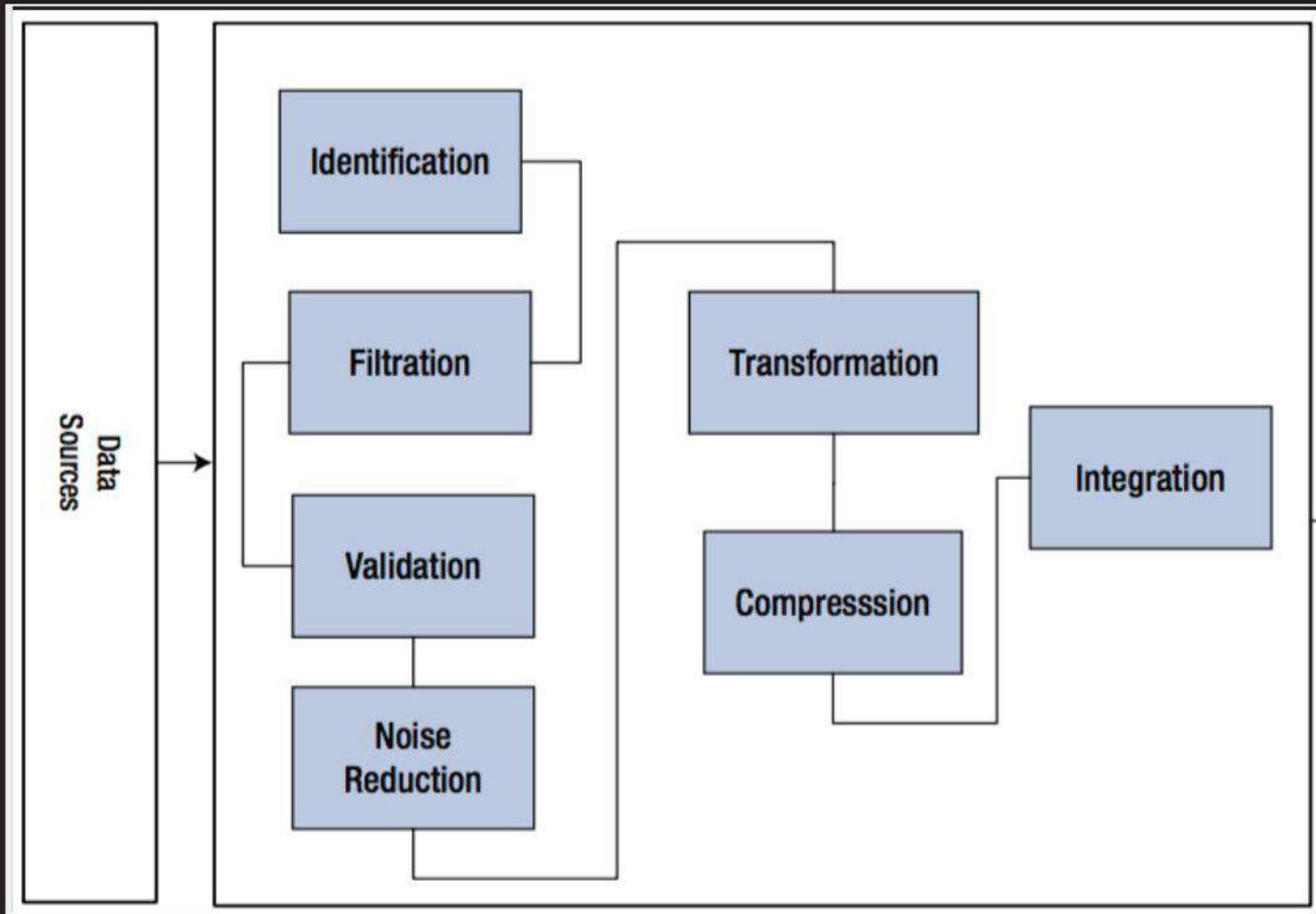
# Ingestion Layer

**Signal-to-Noise  
ratio  
10:90**

separate the  
noise from  
relevant info

# It has capability to

- Validate
- Cleanse
- Transform
- Reduce
- Integrate



# Distributed Storage Layer

# Fault tolerance Parallelization

# HDFS

## massively scalable distributed file system

# HDFS

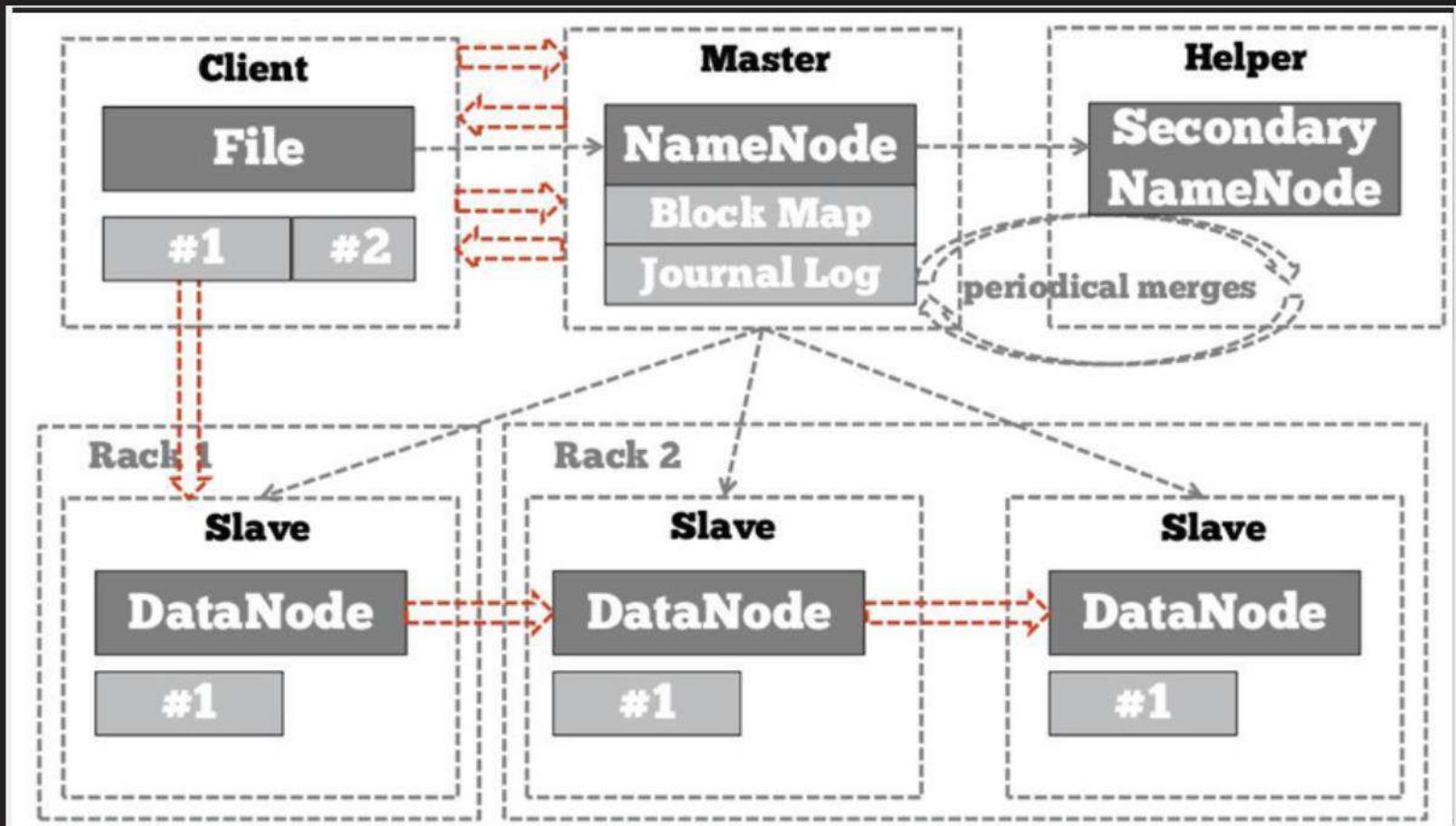
**Intended for**

- **large files**
- **batch inserts**

*Write once, read many times*



# HDFS Architecture

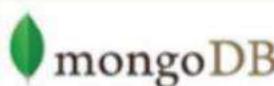
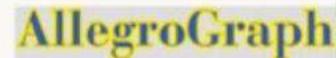


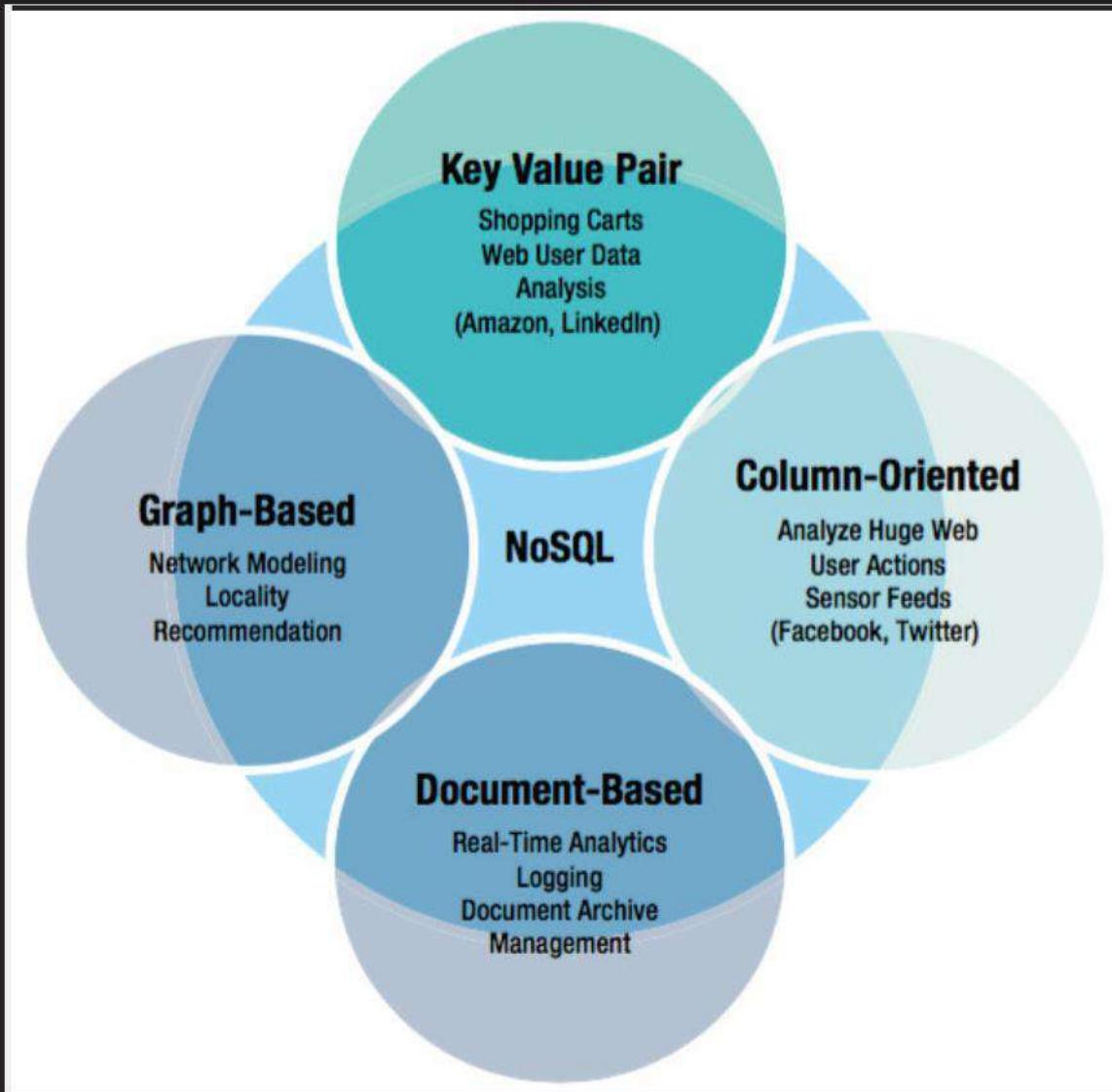
Non-relational,  
distributed data?

# NoSQL

# CAP theorem

## Consistency, Availability, Partition Tolerance

Key-Value Data Stores	Column-oriented Data Stores	Document Data Stores	Graph Data Stores
    	    Windows Azure  	   terrastore 	    



# Ingestion to DFS

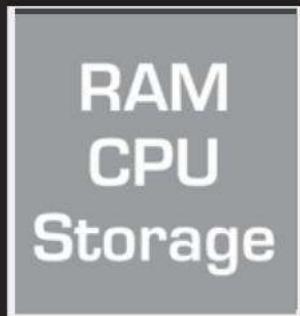
## Sqoop, Flume, MapReduce, ETL

# Infrastructure & Platform Layer

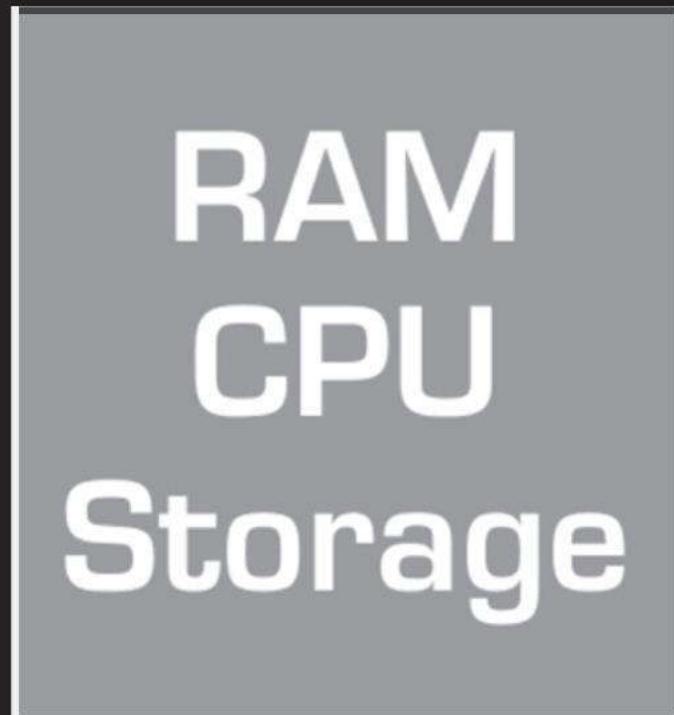
# Computing & Scalability

# Hadoop?

# Vertical Scaling



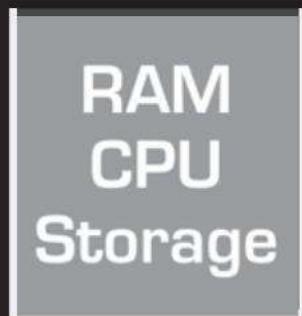
# Vertical Scaling



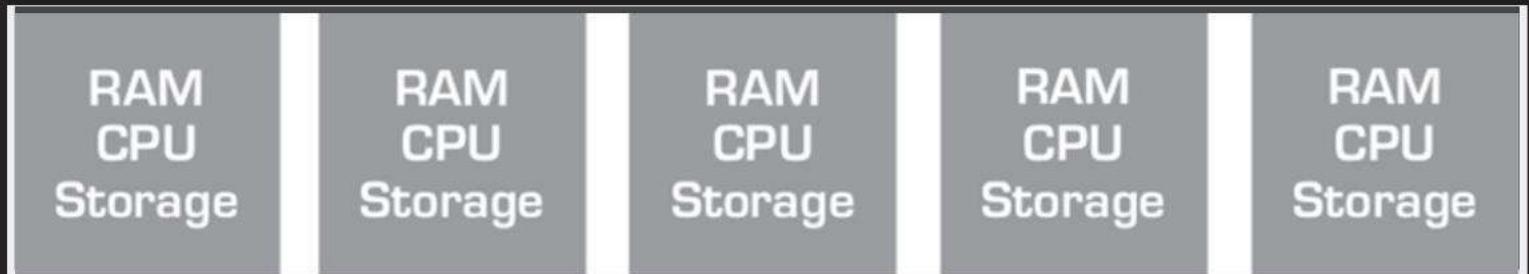
# Vertical Scaling

RAM  
CPU  
Storage

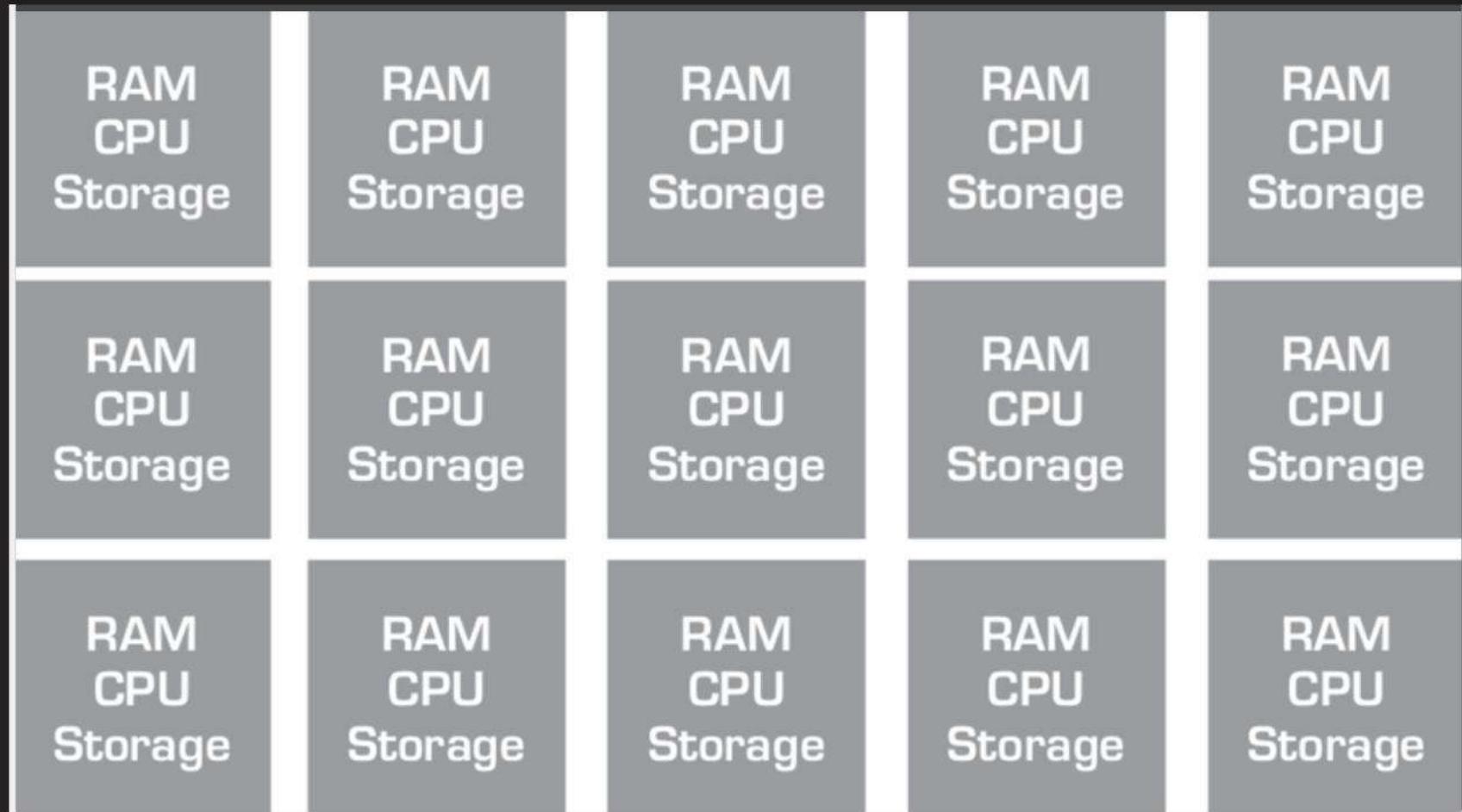
# Horizontal Scaling

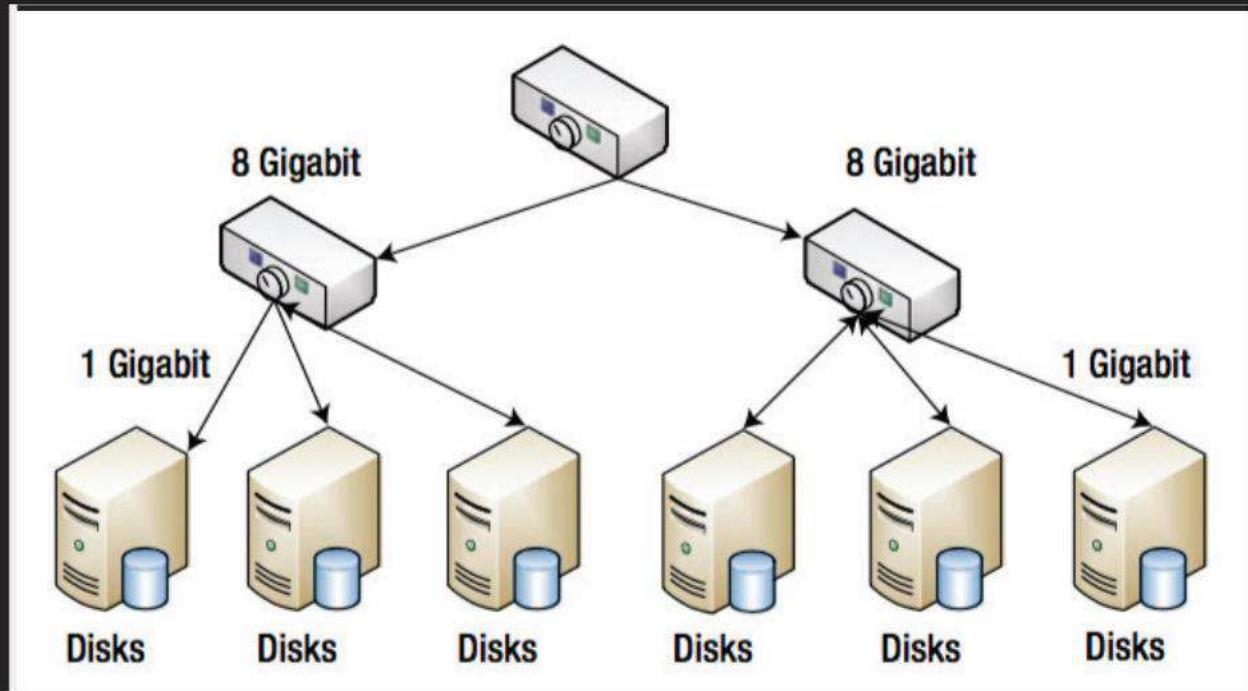


# Horizontal Scaling



# Horizontal Scaling

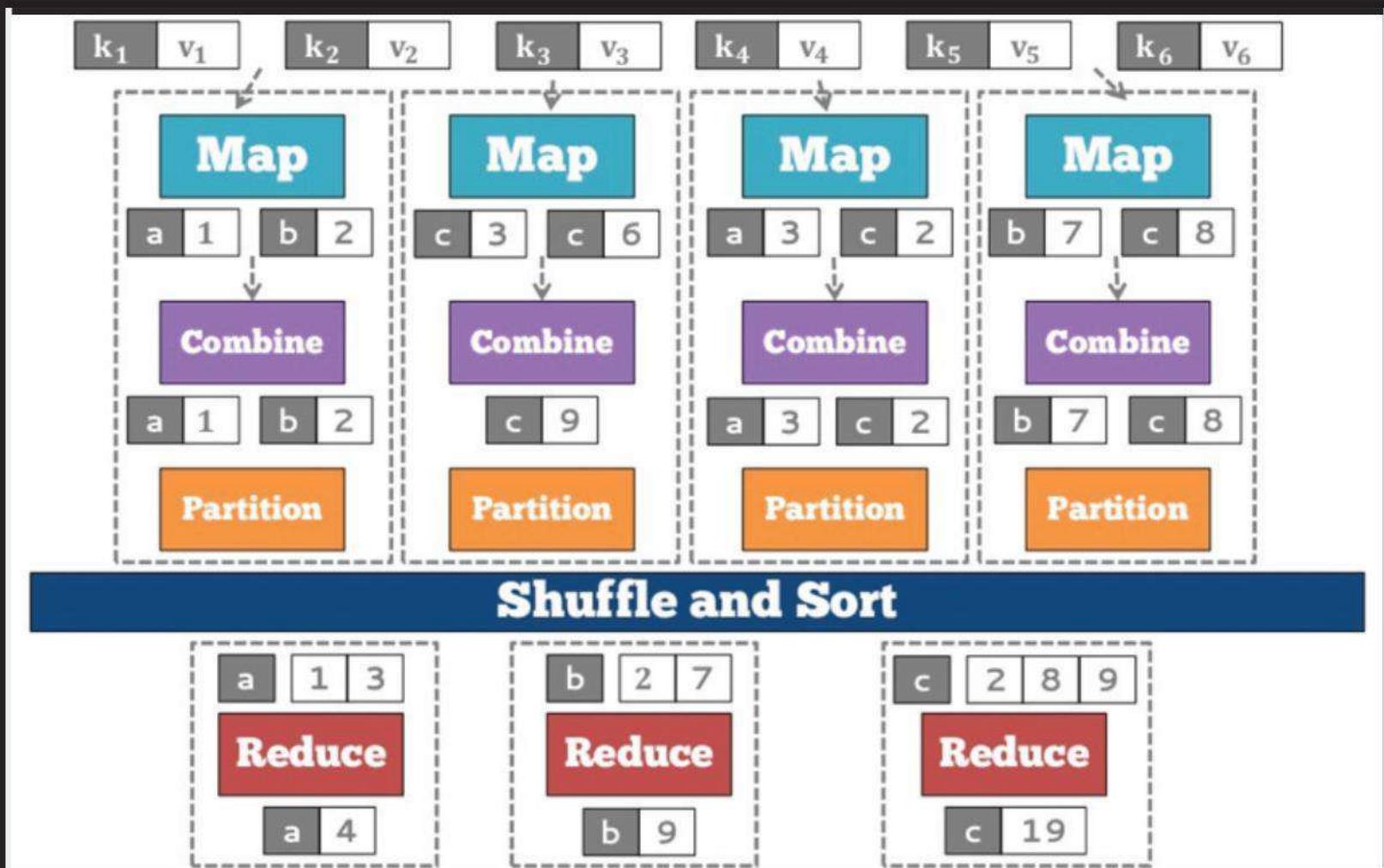


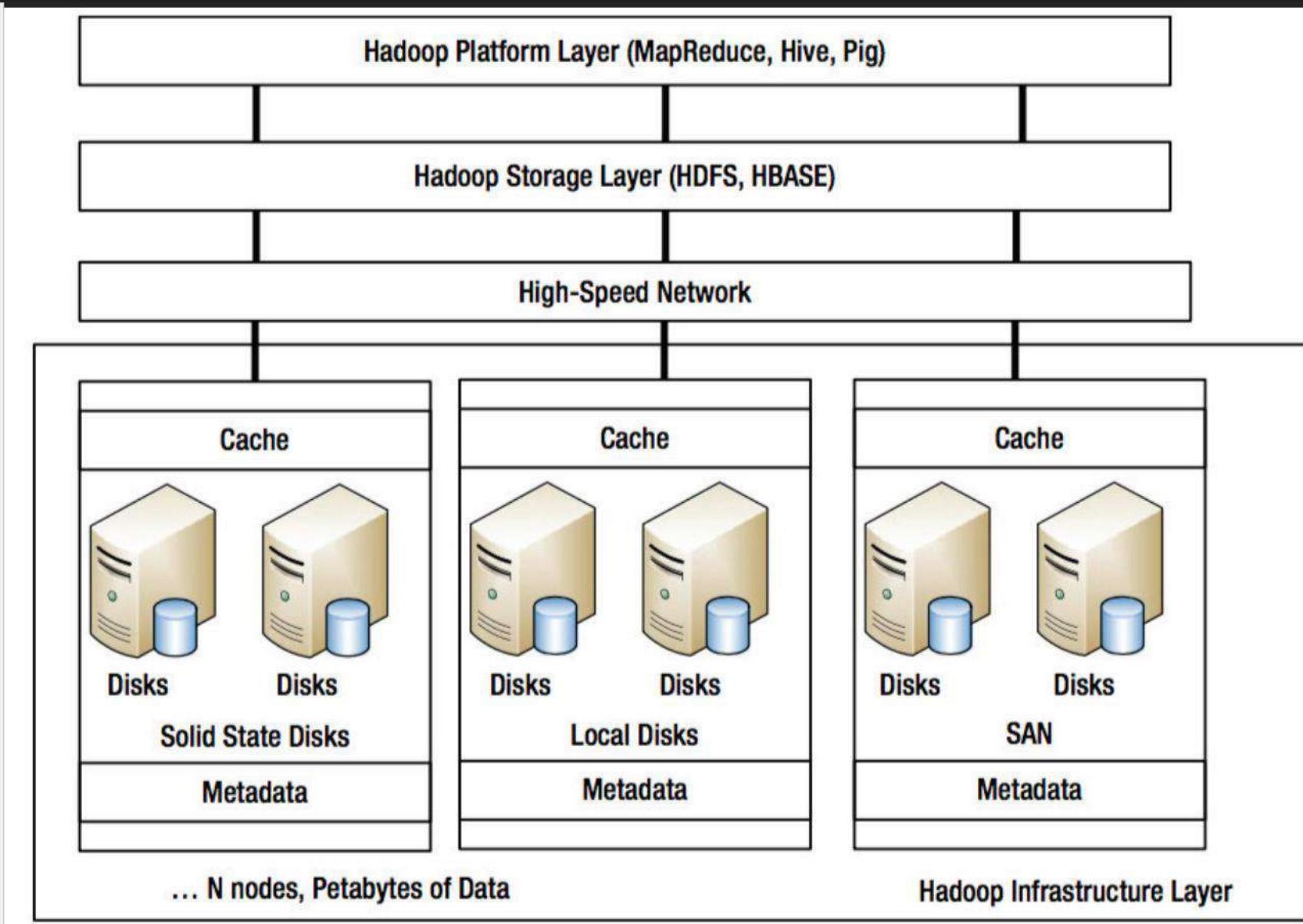


# MapReduce

is the main computation paradigm

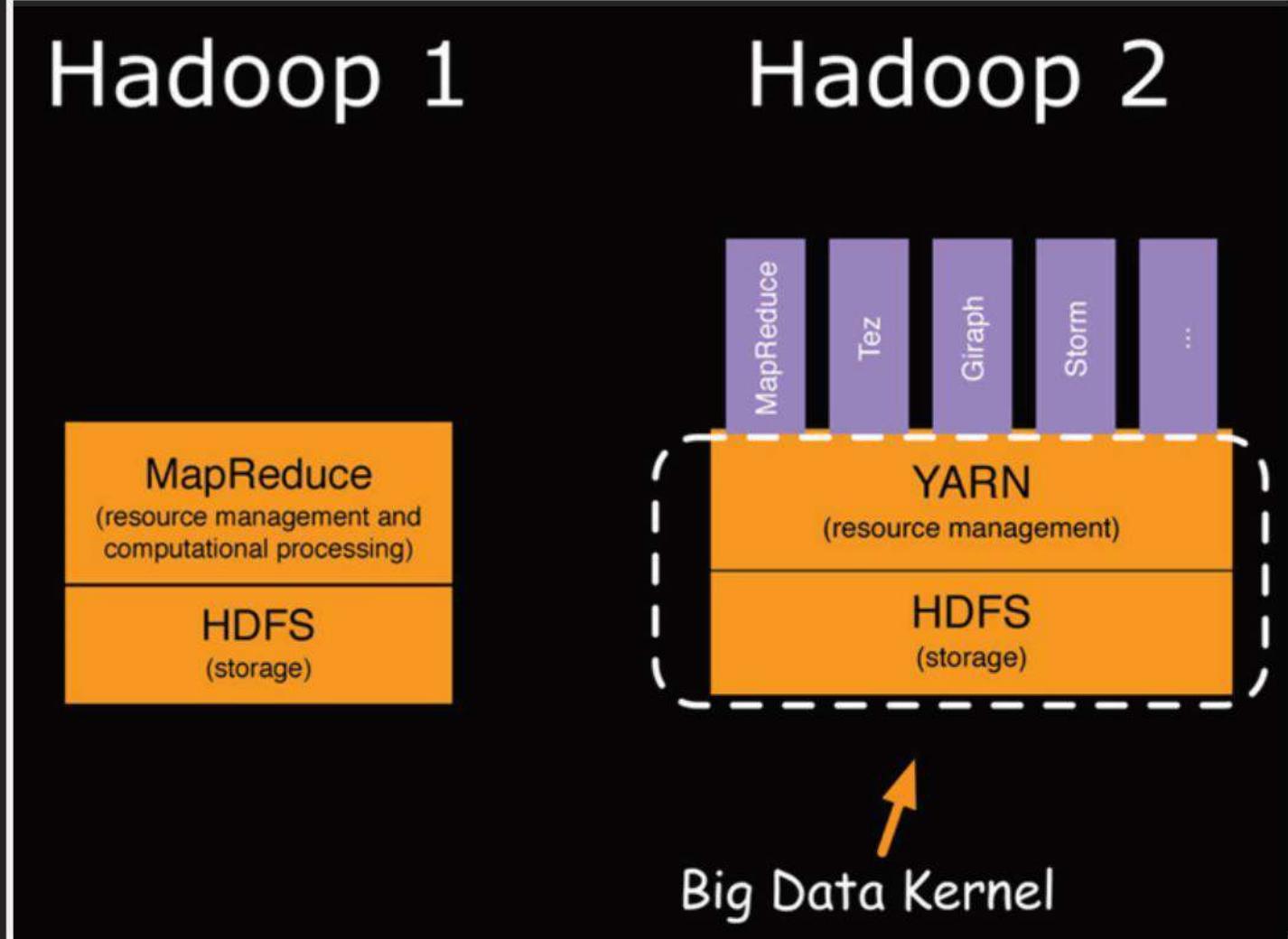
# MapReduce



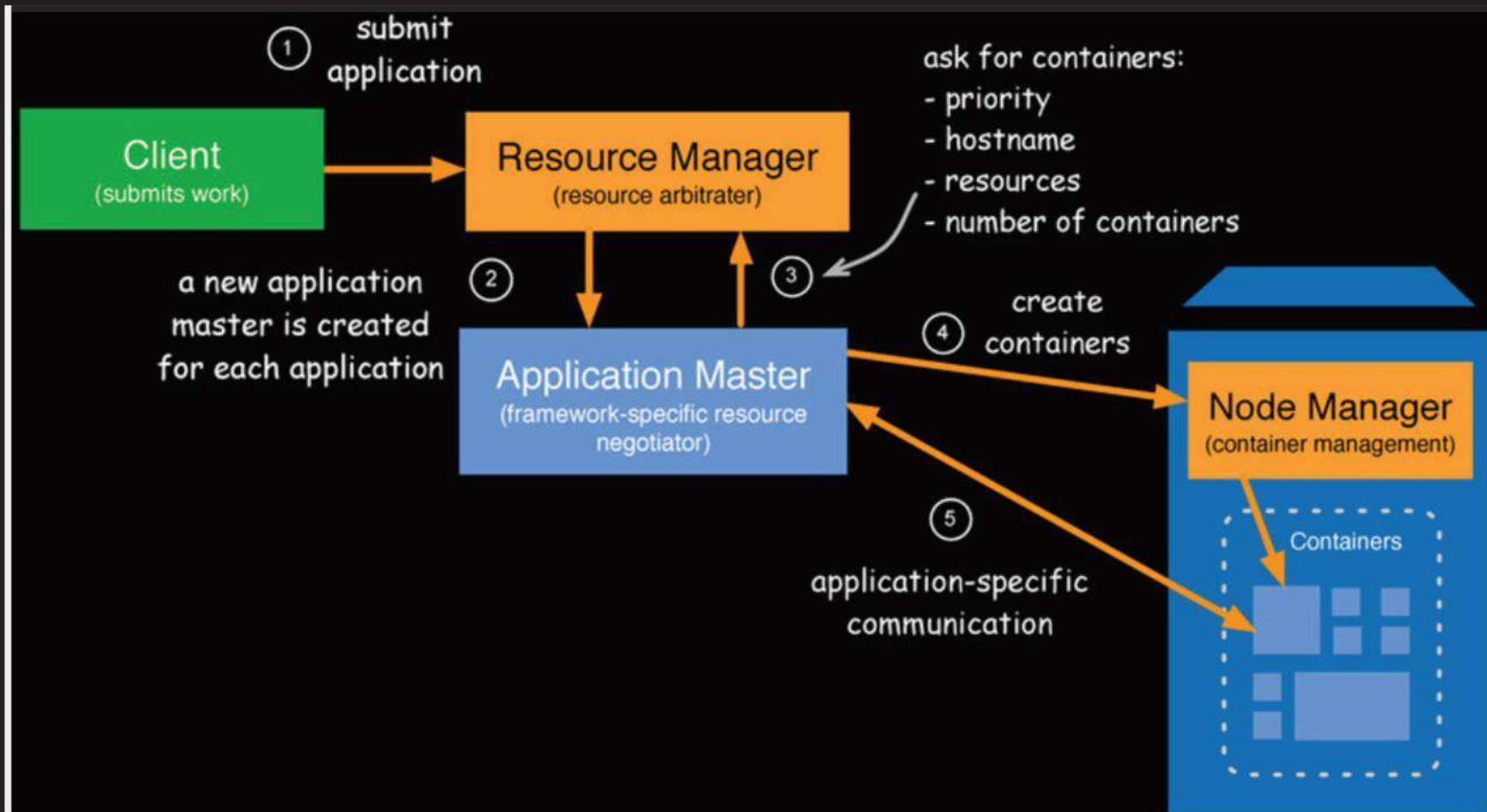


# Hadoop 2

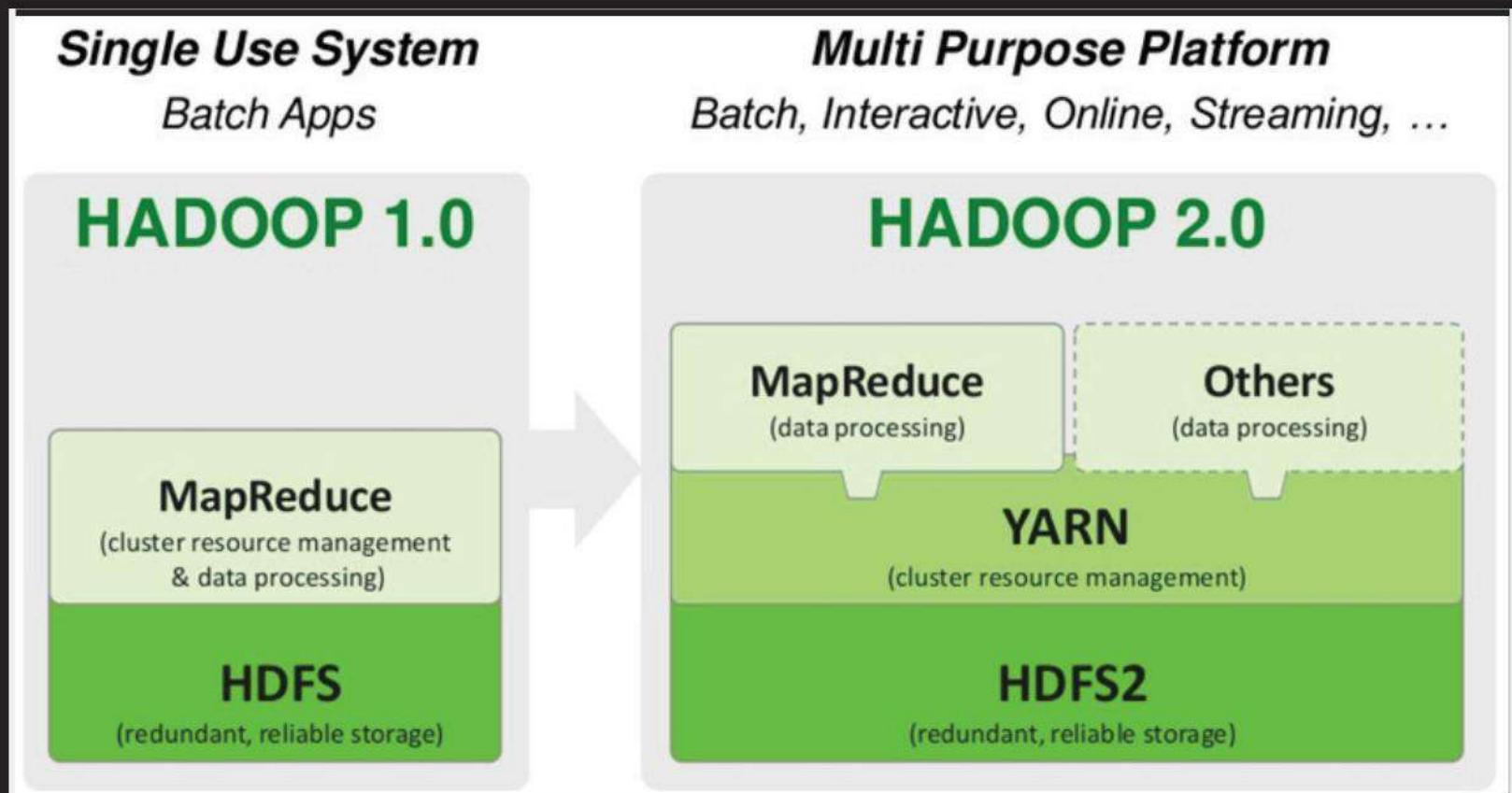
# What's new?



# What's new?



# H1 vs. H2



**One cluster,  
distributed storage,  
distributed scheduler,  
many types of applications.**

# Blueprints

- NoSQL with HBase
- Stream Processing with Storm/Spark
- Graph Processing with Giraph
- SQL on Hadoop with Impala
- Columnar Data Formats

# Security Layer

# Data need to be **protected**

- Meet compliance requirements
- Individual's privacy

Proper  
authorization and  
authentication  
needed

# What can we do?

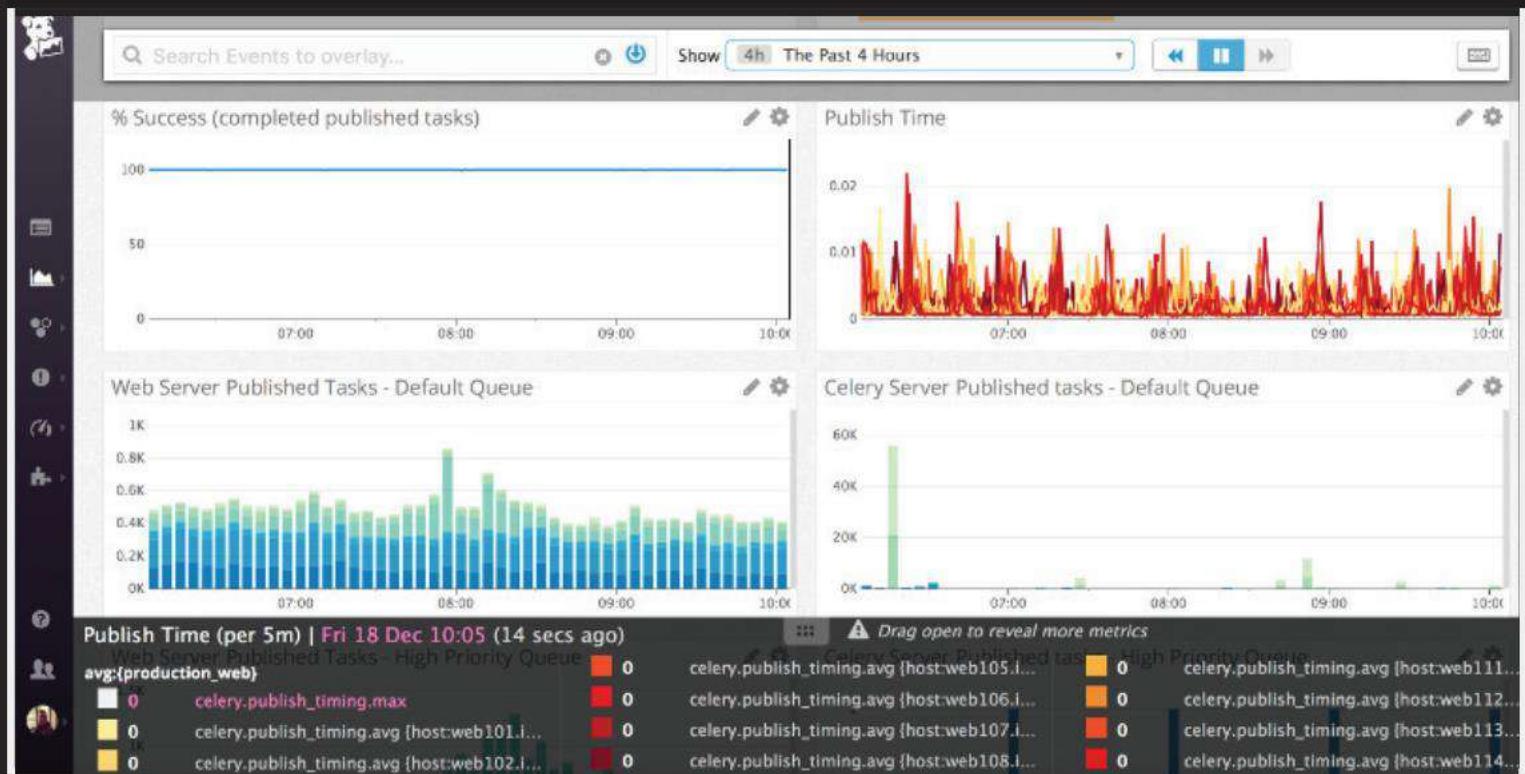
- Authentication protocol like Kerberos
- Enable file layer encryption
- Use SSL, certificates and trusted keys
- Provision with Chef, Puppet or Ansible like tools
- Log all the communication for detecting anomalies
- Monitor whole system

# Monitoring Layer

**Get a complete  
picture  
of our Big Data tech stack**

Satisfy SLAs with  
min downtime

# DataDog



# New Relic (Overview)



# New Relic (Databases)



# Analytics Engine

# Co-Existence with Traditional BI

- Data warehouse in the traditional way
- Distributed MR processing on big data stores

**Mediate data in either direction**  
**i.e use **Hive/HBase** with Sqoop**

**Real-time analysis can leverage  
low-latency NoSQL stores**

i.e Cassandra, Vertica, ...

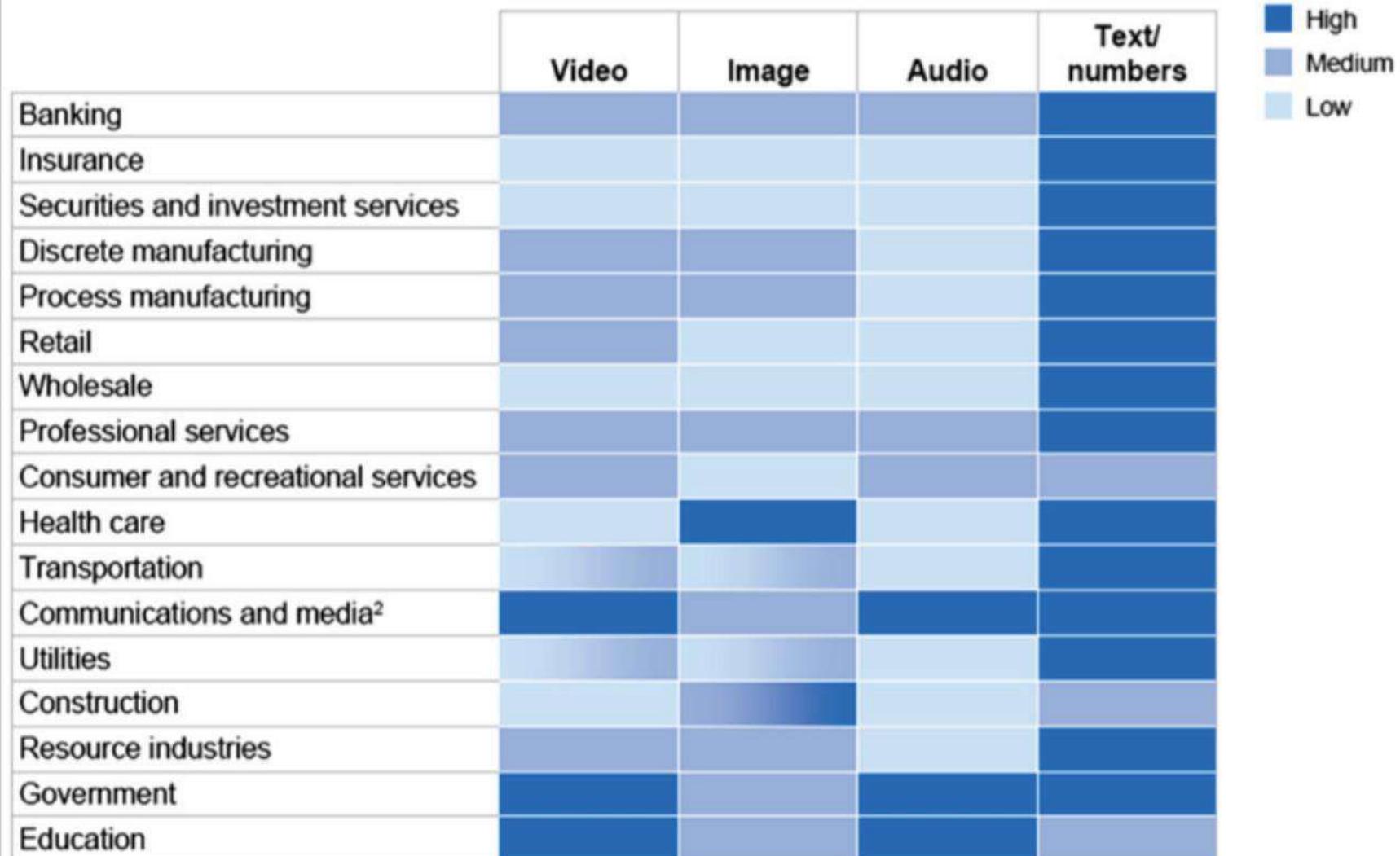
**R may be used for complex  
statistical algorithms**

# Search Engines

Huge volume and  
variety of data

**“needle in a  
haystack”**

## The type of data generated and stored varies by sector<sup>1</sup>



1 We compiled this heat map using units of data (in files or minutes of video) rather than bytes.

2 Video and audio are high in some subsectors.

Need blazing **fast** search  
mechanism  
to **index** and **search** for big data  
analytics

## **Result Display**

## **Query Processing**

## **User Management**

## **Search Functions**

Spelling

Stemming

Faceting

Highlighting

Tagging

Parsing

Semantics

Pertinence

## **Search Engine**

Indexing

Crawling

Elastic Search,  
Solr, ...

# Real-time Processing

# In memory?

**Apache Spark™** is a fast and general engine for large-scale data processing.

Apache Spark

Spark  
SQL

Spark  
Streaming

MLlib  
(machine  
learning)

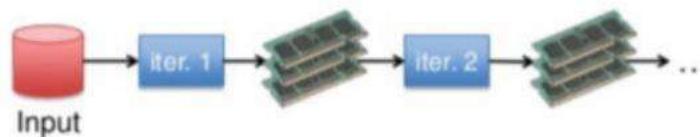
GraphX  
(graph)

Apache Spark

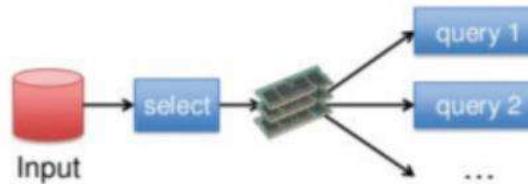


# Spark In-memory Processing

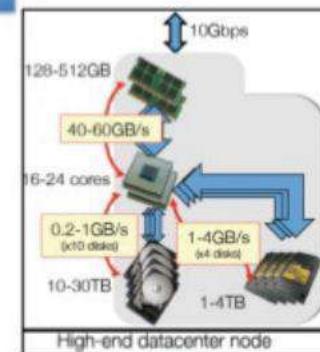
Iterative:



Interactive:



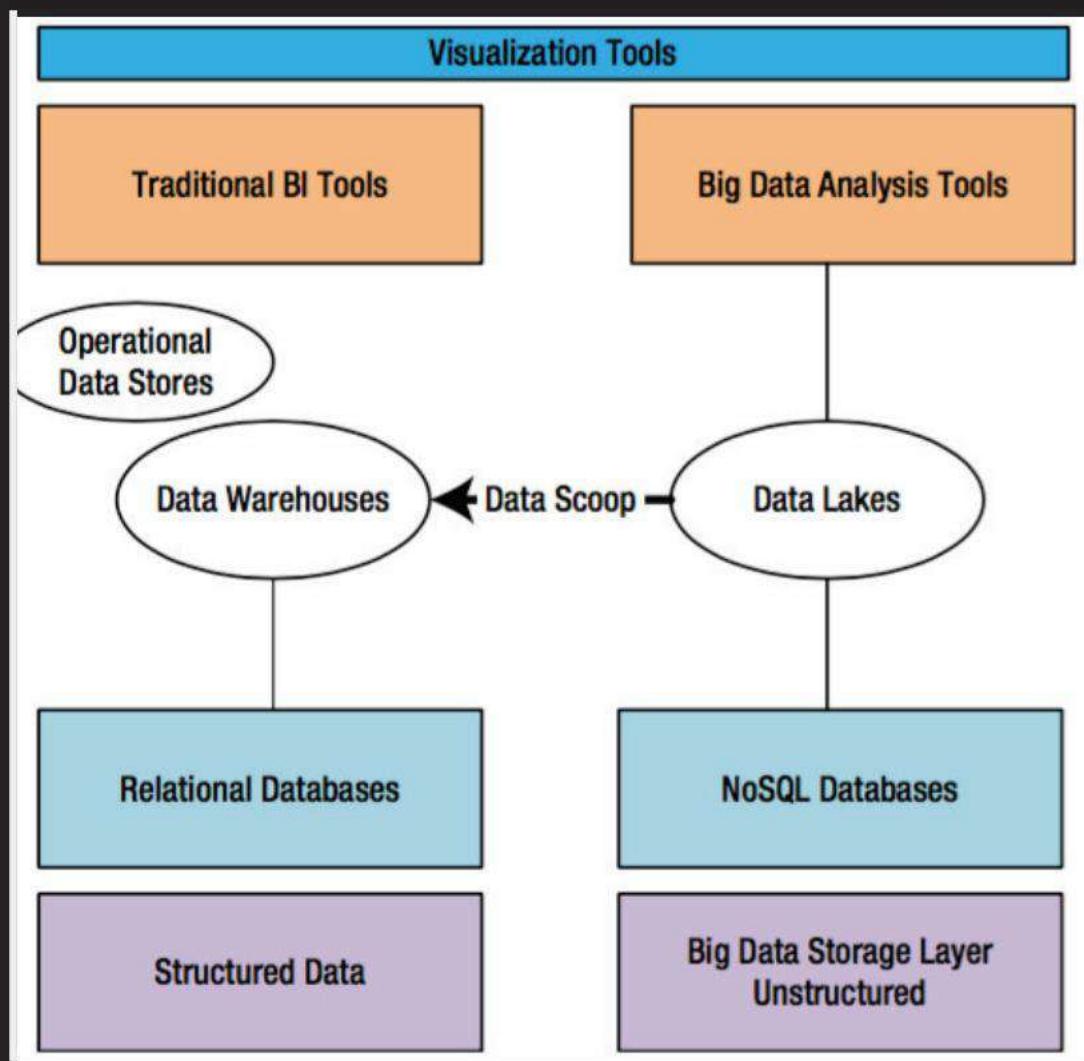
- 1.Extract a working set
- 2.Cache it
- 3.Query it repeatedly



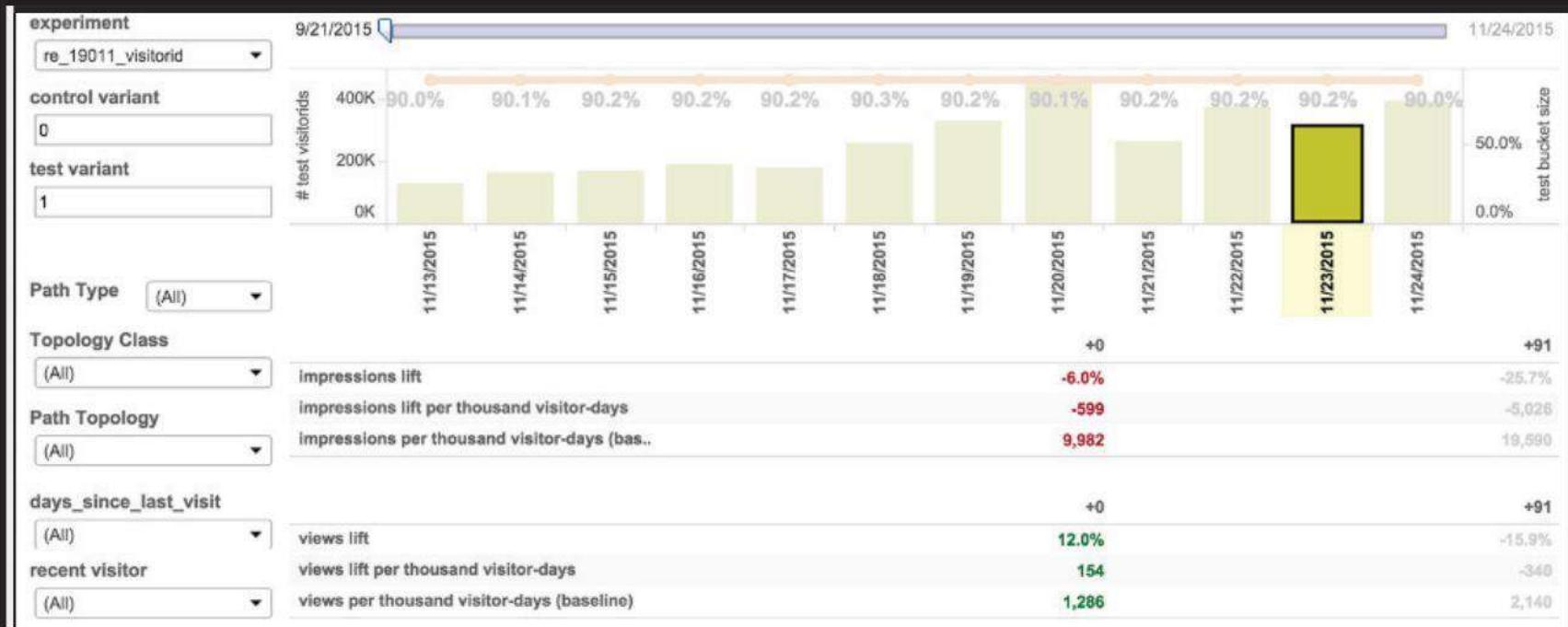
**Storm, Kinesis,  
Flink, ...**

# Visualization Layer

**Gain insight faster**  
**Look at different aspects of**  
**data visually**



# Tableau



# ChartIO

## Learning at Udemy

**357,772**

Days of Content Consumed  
in 2015

*Students consumed an incredible 352,000 days of content on Udemy in 2015; average consumption also went up across the platform.*

monthly average min consumed

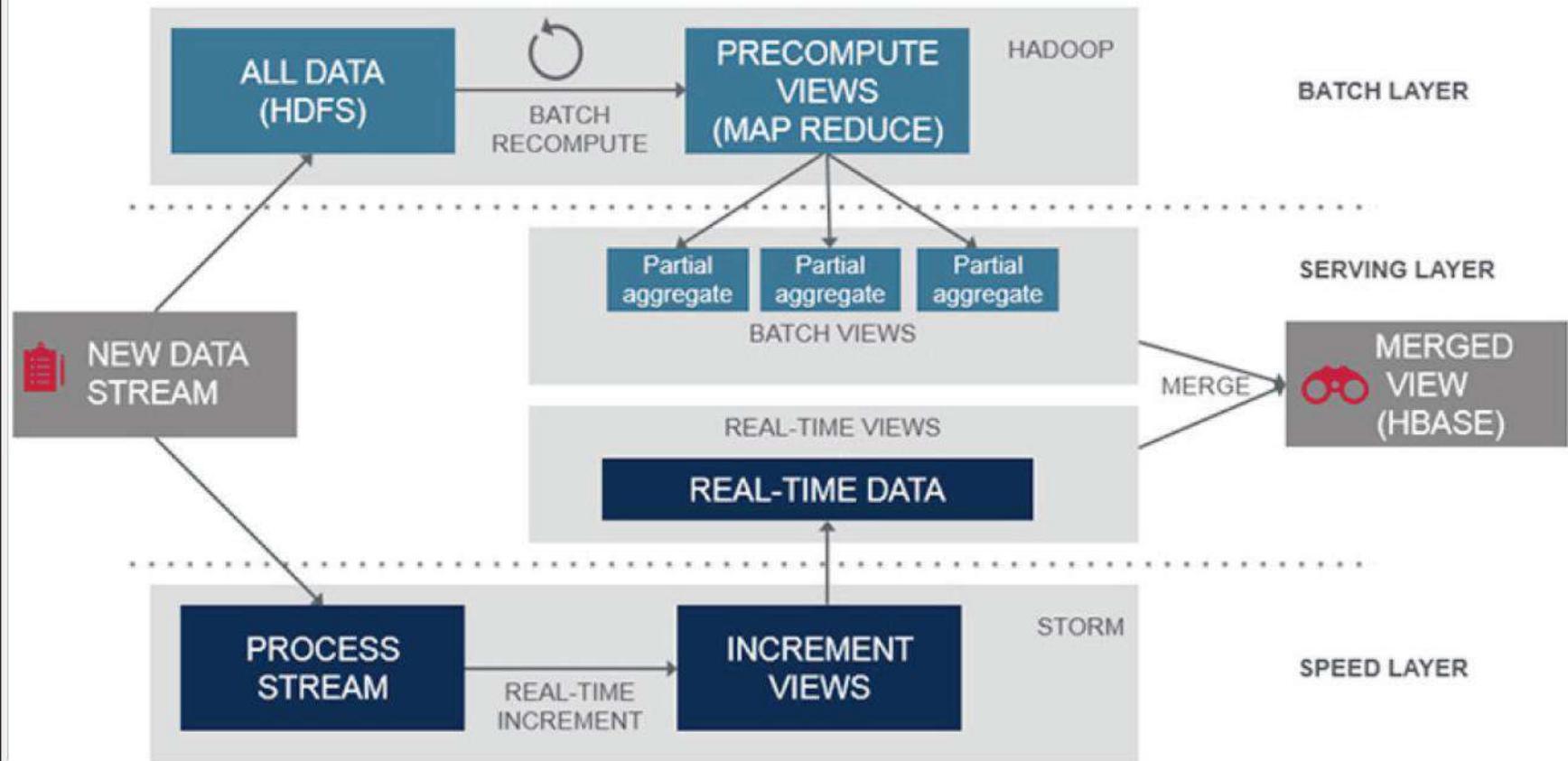


Total Minutes Watched by Udemy Students



# Lambda Architecture

# Lambda Architecture



Lambda Architecture / MapR

# Don't forget

There is no  
"One Size Fits All"  
solution

We need  
**Continuous  
Development**



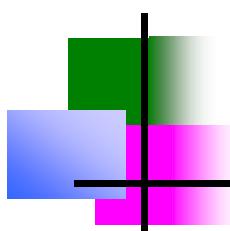
*Get Big*  
by starting

*small*



Focus on  
*Business Impact.*

# Thank You :)



# **Chapter3**

# **Understanding Big Data**

# **Technology Foundations**

## **Lamda Architecture**

**Basanta Joshi, PhD**

Asst. Prof., Depart of Electronics and Computer Engineering

Program Coordinator, MSc in Information and Communication Engineering

Member, Laboratory for ICT Research and Development (LICT)

Member, Research Management Cell (RMC)

Institute of Engineering

[basanta@ioe.edu.np](mailto:basanta@ioe.edu.np)

<http://www.basantajoshi.com.np>

<https://scholar.google.com/citations?user=iocLiGcAAAAJ>

[https://www.researchgate.net/profile/Basanta\\_Joshi2](https://www.researchgate.net/profile/Basanta_Joshi2)



# The world is changing !

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

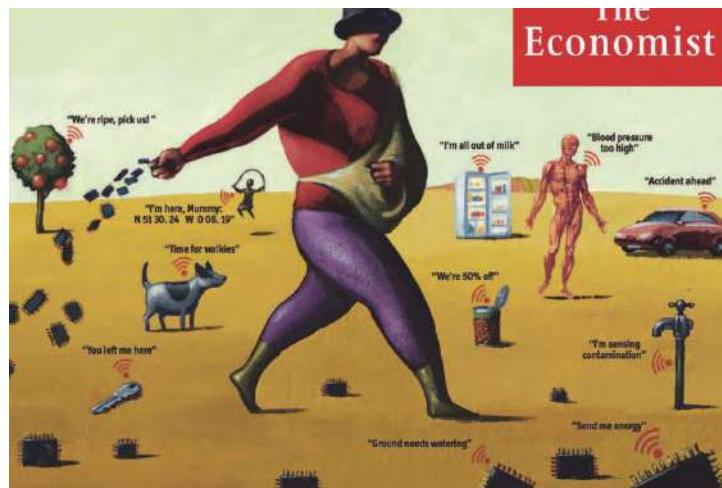




# ■ Internet Of Things – Sensors are/will be everywhere

There are more devices tapping into the internet than people on earth

How do we prepare our systems/architecture for the future?

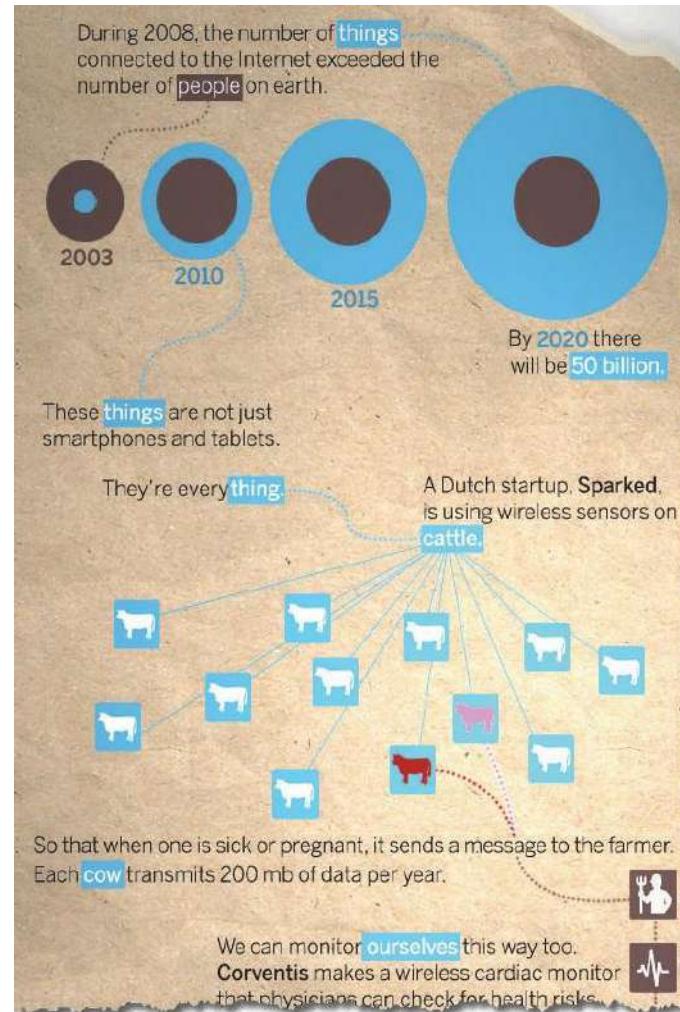


2014 © Trivadis

DOAG 2014 | Big Data und Fast Data - Lambda Architektur und deren Umsetzung 19.11.2014

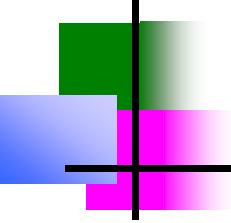
Source: The Economist

8



Source:  
Cisco

**trivadis**  
makes IT easier.



# 2020 IoT Insights

---

- With **1.3 billion projected subscriptions by 2023**, IoT is about to experience another boost by the 5G technology.
  - By 2022, **Google Home will have the largest IoT devices market share**, at 48%.
  - The average number of connected devices **per household in 2020 will be 50**.
  - By 2021, **35 billion IoT devices** will be installed around the world.
  - The number of connected devices **in 2020 is predicted to hit 50 billion**.
- 
- <https://techjury.net/blog/how-many-iot-devices-are-there/#gref>

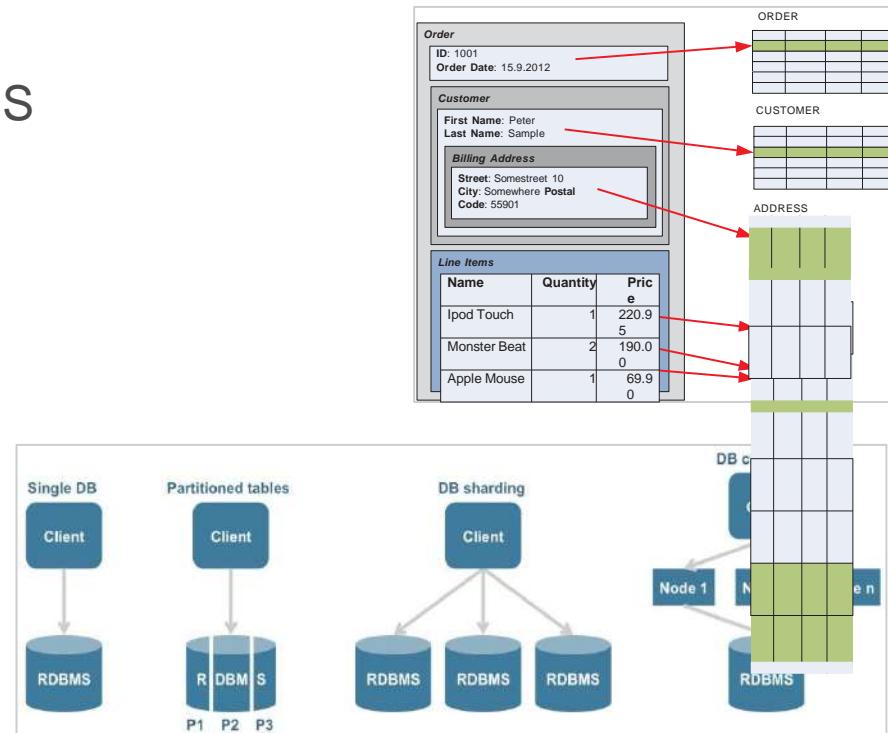


# The world is changing new data stores

Not  
Only SQL OR ~~SQL~~

Problem of traditional (R)DBMS  
approach  
complex object  
graph

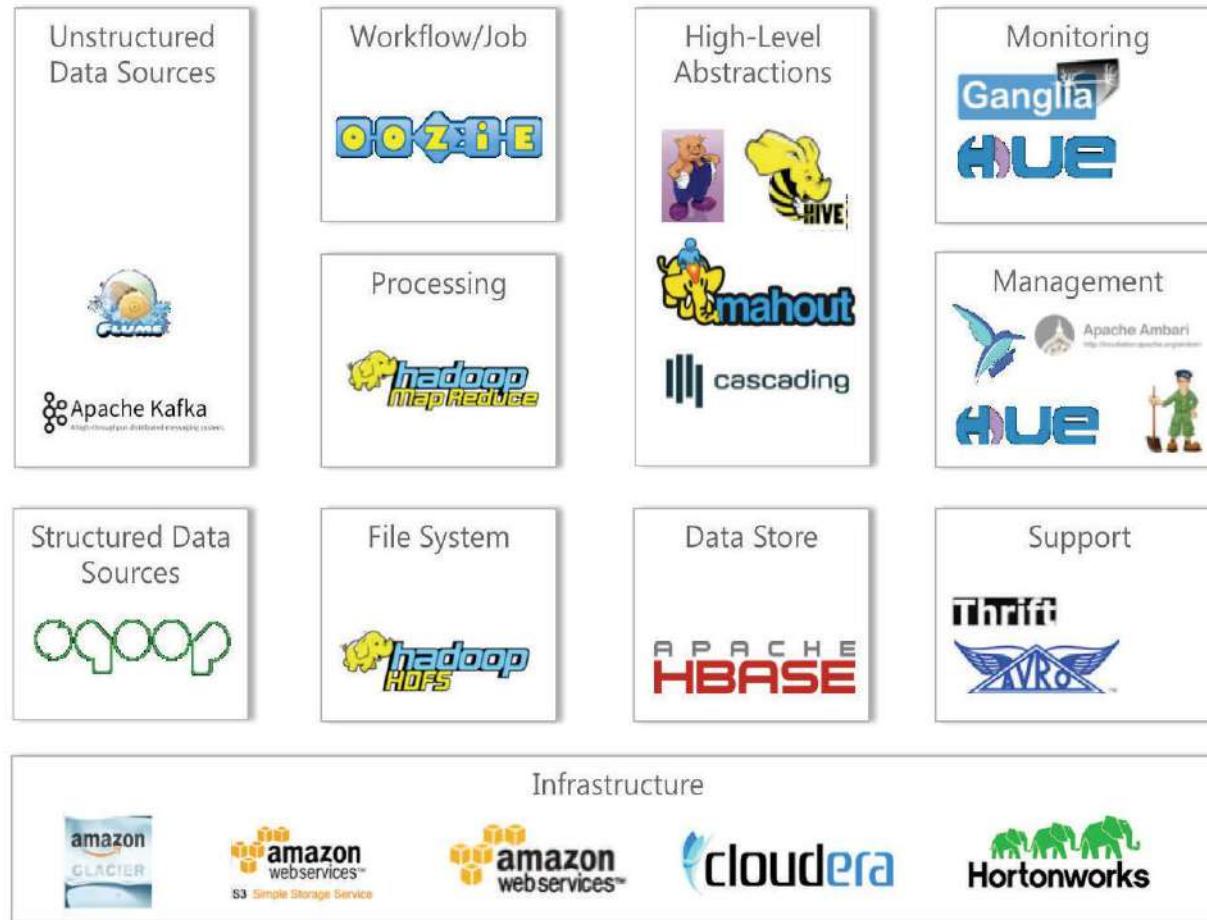
- Schema evolution
- Semi-structured data
- Scaling



Polyglot persistence

- Using multiple data storage technologies (RDBMS + NoSQL + NewSQL + In-Memory)

# ■ The world is changing ! New platforms evolving (i.e. Hadoop Ecosystem)



## ■ Data as an Asset – Store everything?



Data is just too valuable to delete!  
**We must store anything!**

**It depends ...**  
Big Data technologies allow to  
**store the raw information** from new and existing data sources so that you can later use it to create **new data-driven products**, which you haven't thought about today!



**Nonsense!**  
Just store the data you know you need today!





# AGENDA

1. Big Data and Fast Data, what is it?

**2. Architecting (Big) Data Systems**

3. The Lambda Architecture

4. Use Case and the Implementation

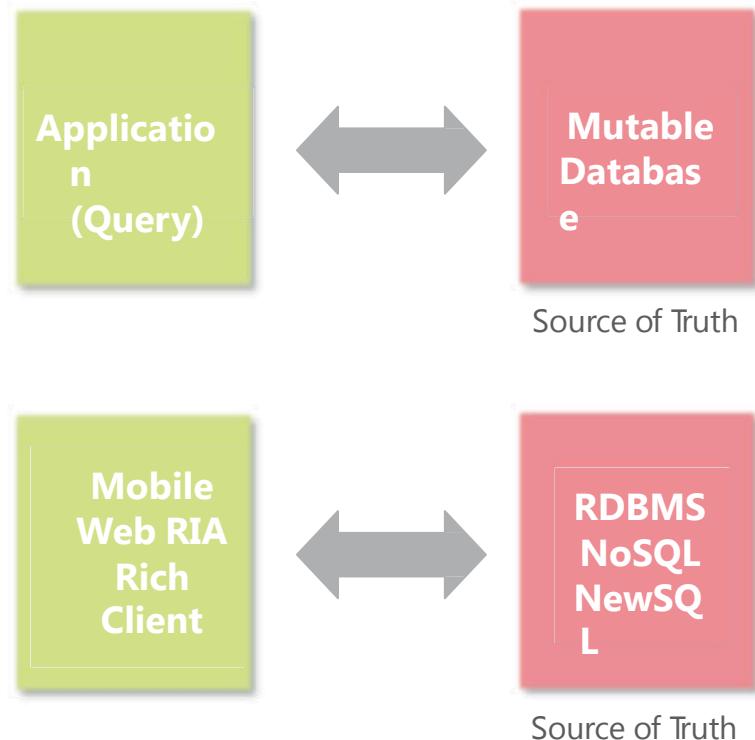
5. Summary and Outlook



## What is a data system?

- A (data) system that manages the **storage and querying of data** with a **lifetime measured in years** encompassing **every version** of the application to ever exist, every **hardware failure** and every **human mistake** ever made.
- A data system answers **questions based on information** that was **acquired in the past**

## ■ How do we build (data) systems today – Today's Architectures



Source of Truth **is mutable!**

- CRUD pattern

What is the problem with this?

- Lack of Human Fault Tolerance

- Potential loss of information/ data



# Lack of Human Fault Tolerance

Bugs will be deployed to production over the lifetime of a data system

Operational mistakes will be made

Humans are part of the overall system

- Just like hard disks, CPUs, memory, software
- design for human error like you design for any other fault

Examples of human error

- Deploy a bug that increments counters by two instead of by one
- Accidentally delete data from database
- Accidental DOS on important internal service

Worst two consequences: **data loss** or **data corruption**

As long as an error **doesn't lose or corrupt** good data, you can **fix** what went wrong



# Lack of Human Fault Tolerance – Immutability vs. Mutability

An immutable system captures historical records of events

Each event happens at a particular time and is always true

The **U** and **D** in CRUD

A mutable system updates the current state of the world

**Mutable** systems inherently lack human fault-tolerance

The diagram illustrates the mutation of a database table. It starts with an initial state (left) showing two rows: Guido in Berne and Albert in Zurich. An arrow points to an intermediate state (middle), where a new row for Guido in Basel is added with a timestamp of 1.4.2013. Another arrow points to a final state (right), where the row for Guido in Basel has been updated to Guido in Basel, and the row for Albert in Zurich has been updated to Albert in Zurich.

Name	City	Timestamp
Guido	Berne	1.8.1999
Albert	Zurich	10.5.1988

→

Name	City	Timestamp
Guido	Berne	1.8.1999
Albert	Zurich	10.5.1988
Guido	Basel	1.4.2013

→

Name	City
Guido	Berne
Albert	Zurich

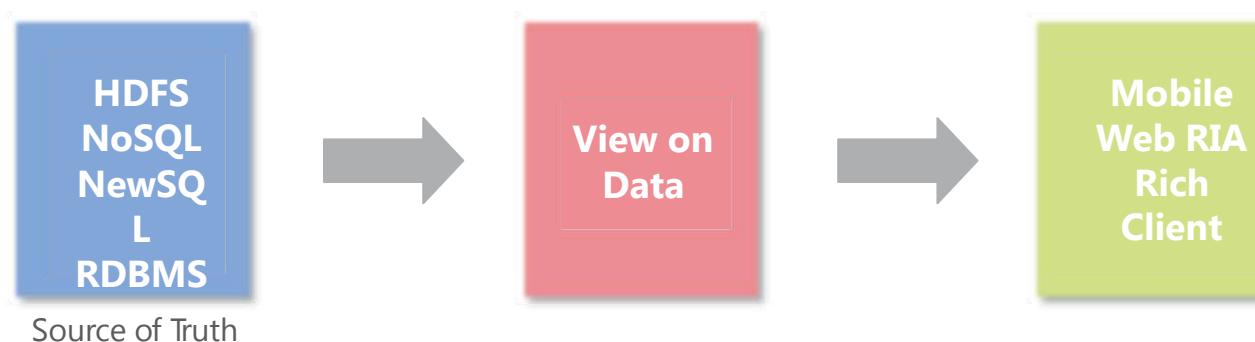
Immutability restricts the range of errors causing data loss/data

corruption Vastly more **human fault-tolerant**

**Conclusion:** Your **source of truth** should always be **immutable**

## ■ A different kind of architecture with immutable source of truth

Instead of using our traditional approach ! why not building data systems like this



## ■ How to create the views on the Immutable data?

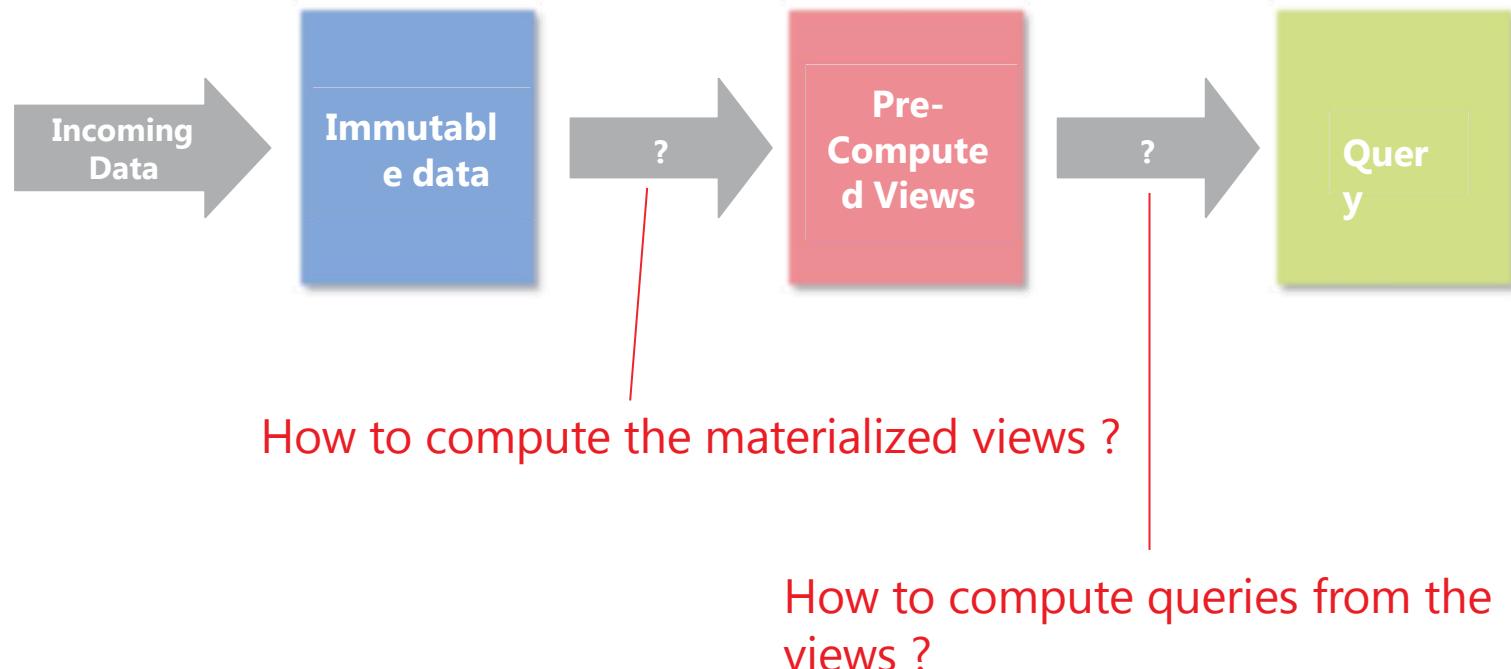
On the fly



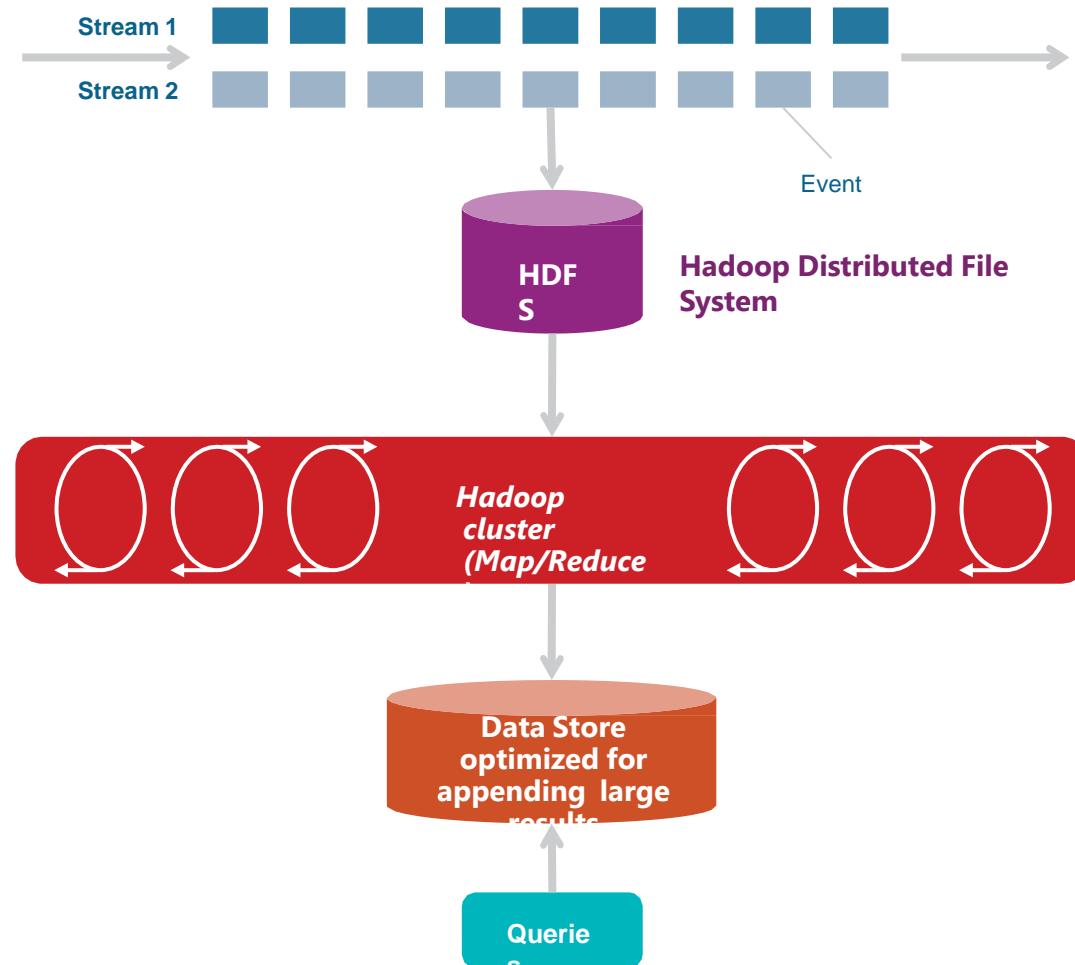
Materialized, i.e. Pre-computed



# (Big) Data Processing



# ■ Today Big Data Processing means Batch Processing !



# Big Data Processing - Batch

## Favorite Product List

1.2.13	Add	iPAD 64GB
10.3.13	Add	Sony RX-100
11..3.13	Add	Canon GX-10
11.3.13	Remove	Sony RX-100
12.3.13	Add	Nikon S-100
14.4.13	Add	BoseQC-15
15.4.13	Add	MacBook Pro
		15
20.4.13	Remove	Canon GX10

**Raw information =>  
data**



**Current Favorite Product List**  
iPAD 64GB  
Nikon S-100  
BoseQC-15  
MacBook Pro  
15

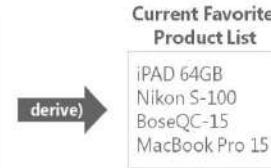


**Current Product Count**  
4

**Information =>  
derived**

# ■ Big Data Processing – Batch

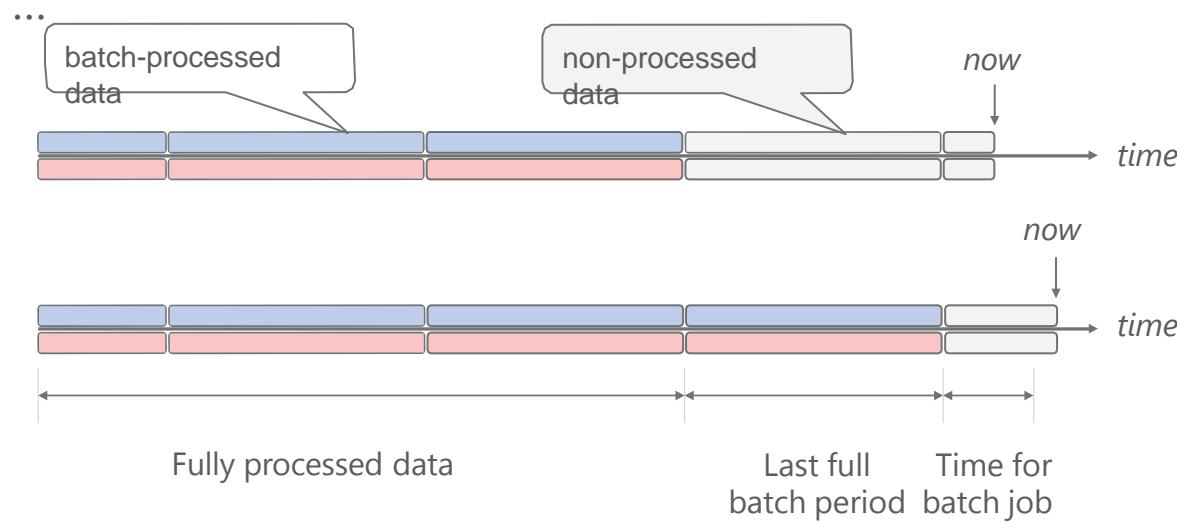
Favorite Product List Changes		
1.2.13	Add	iPAD 64GB
10.3.13	Add	Sony RX-100
11..3.13	Add	Canon GX-10
11.3.13	Remove	Sony RX-100
12.3.13	Add	Nikon S-100
14.4.13	Add	BoseQC-15
15.4.13	Add	MacBook Pro 15
20.4.13	Remove	Canon GX10
Now	Add	Canon Scanner



derive



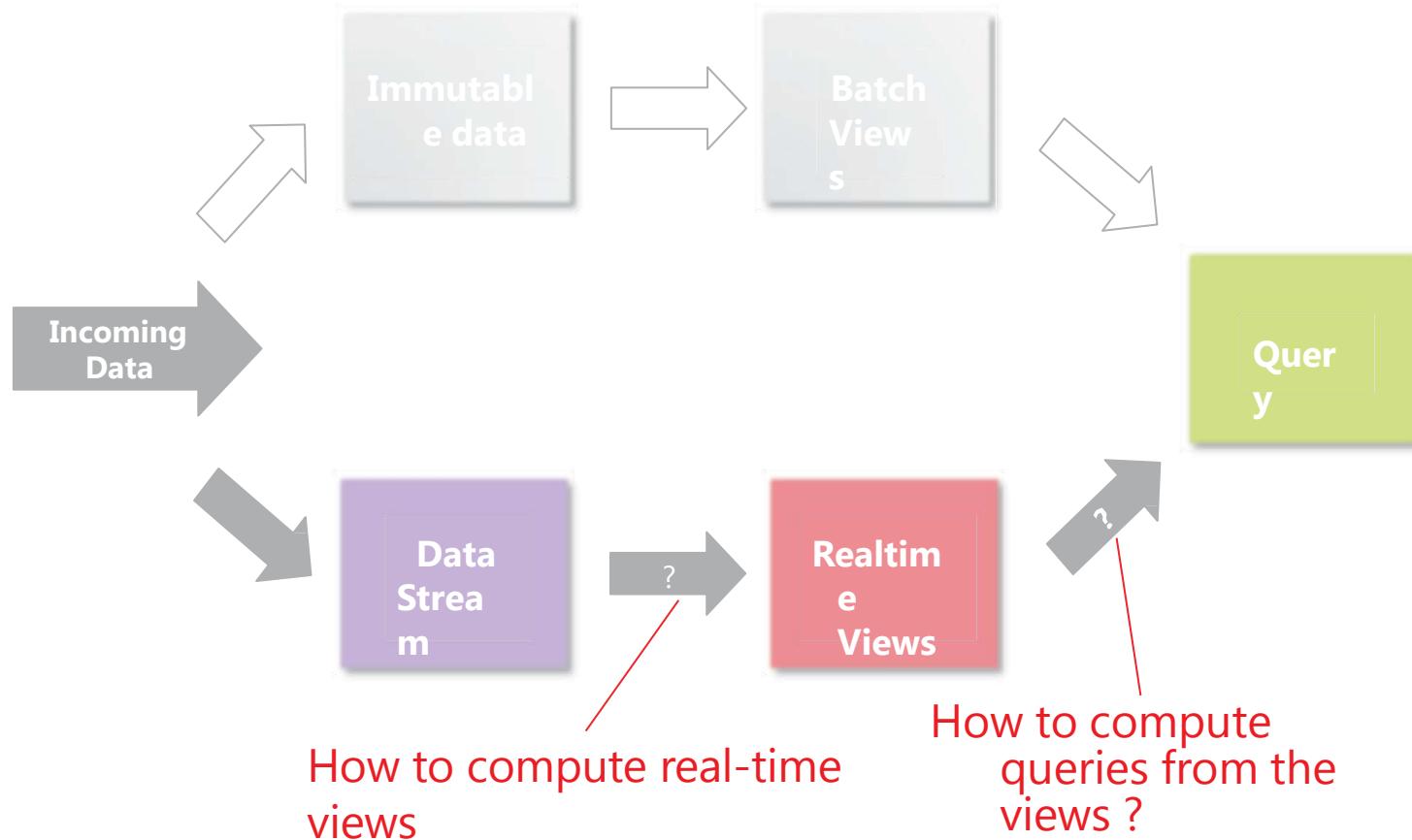
But we are not done yet



- Using only batch processing, leaves you always with a portion of non- processed data.

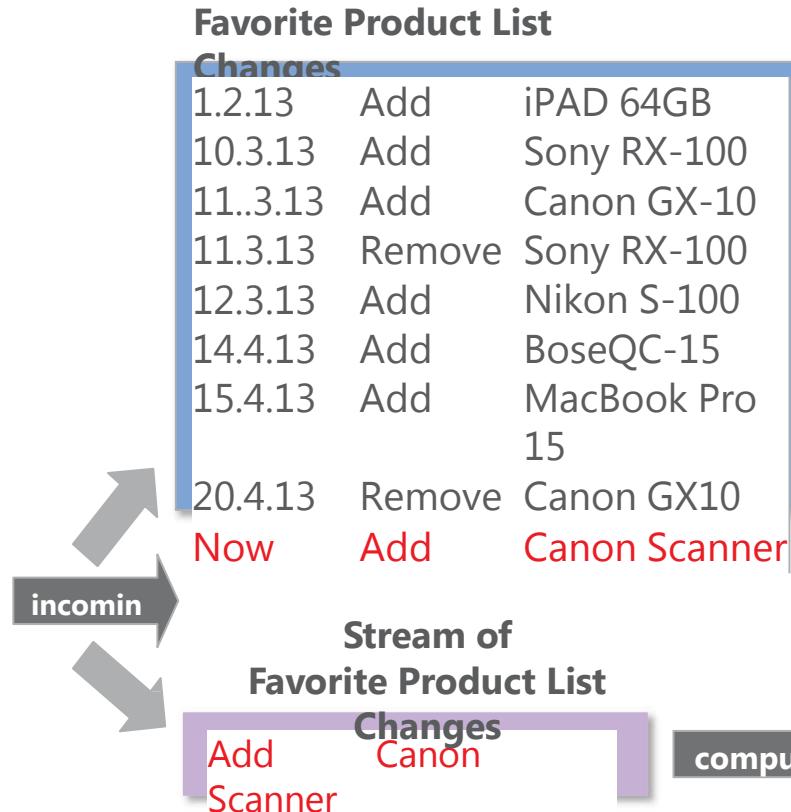
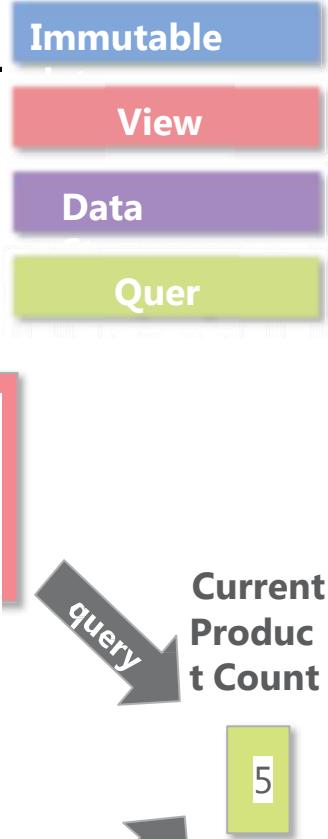


# Big Data Processing - Adding Real-Time

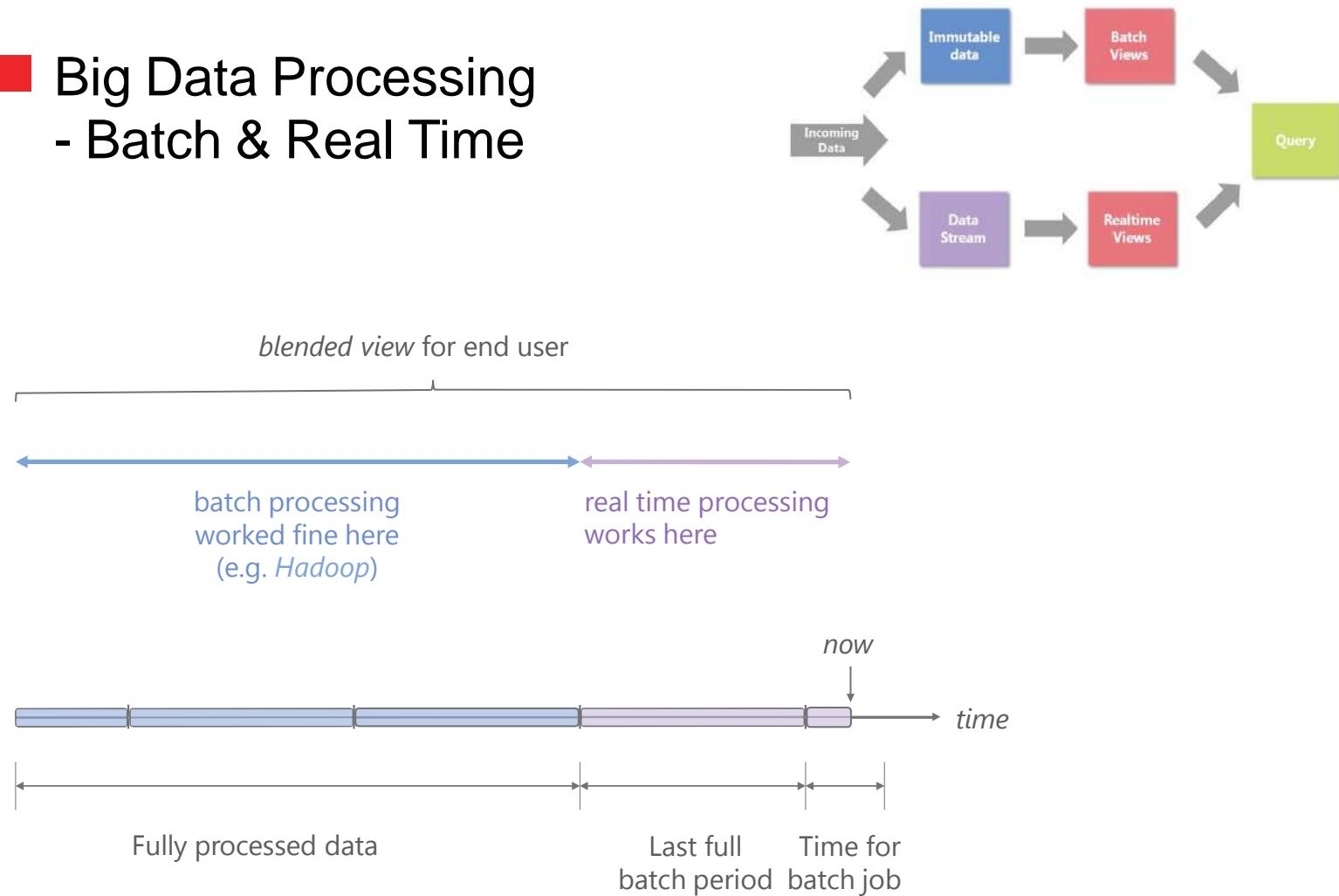




# Big Data Processing - Adding Real-Time



# ■ Big Data Processing - Batch & Real Time



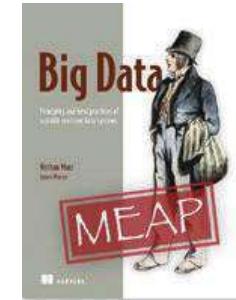
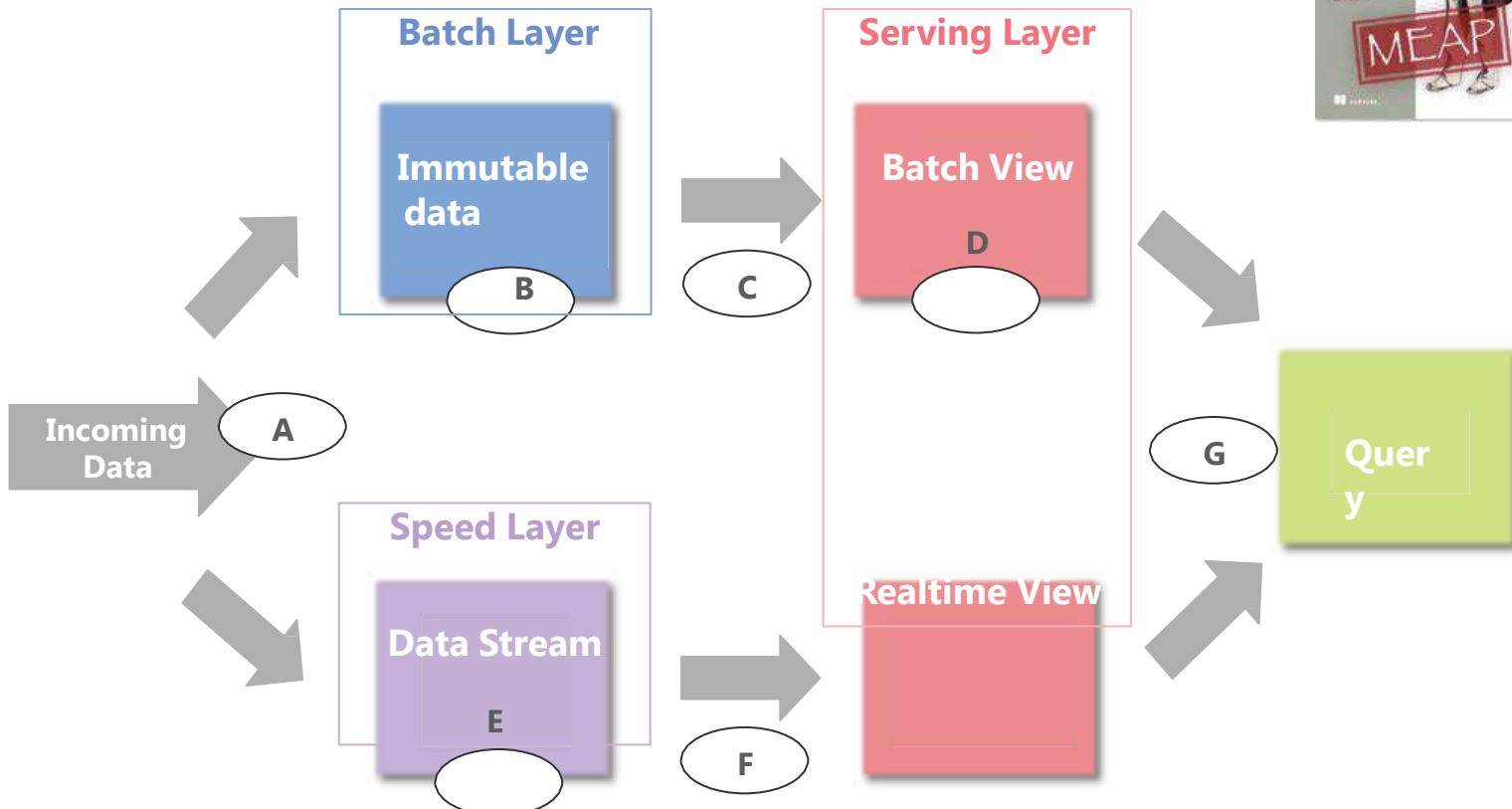
Adapted from Ted Dunning (March 2012):  
<http://www.youtube.com/watch?v=7PcmbI5aC20>



# AGENDA

1. Big Data and Fast Data, what is it?
2. Architecting (Big) Data Systems
- 3. The Lambda Architecture**
4. The Use Case and the Implementation
5. Summary and Outlook

# Lambda Architecture



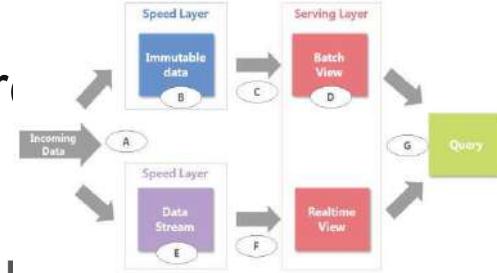
Lambda => Query = function(all data)

27

2014 © Trivadis

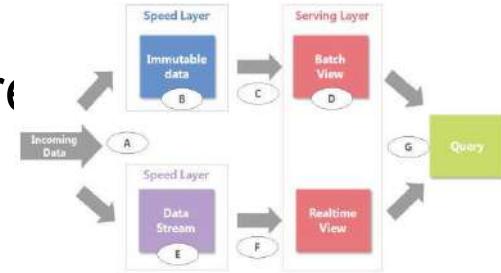
DOAG 2014 | Big Data und Fast Data - Lambda Architektur und deren Umsetzung 19.11.2014

# Lambda Architecture



- A. All data is sent to **both** the **batch and speed layer**.
- B. Master data set is an **immutable, append-only** set of data
- C. Batch layer **pre-computes** query functions from scratch, result is called Batch Views. Batch layer **constantly re-computes** the batch views.
- D. Batch views are **indexed** and **stored** in a **scalable database** to get particular values very quickly. Swaps in new batch views when they are available
- E. Speed layer **compensates** for the high latency of updates to the Batch Views
- F. Uses fast **incremental algorithms** and read/write databases to produce real- time views
- G. Queries are resolved by getting results from **both** batch and real- time views

# Lambda Architecture



## Batch

Stores the immutable constantly growing dataset  
Computes arbitrary views from this dataset using BigData technologies (can take hours)  
Can be always recreated

## Speed

Computes the views from the constant stream of data it receives  
Needed to compensate for the high latency of the batch layer  
Incremental model and views are transient

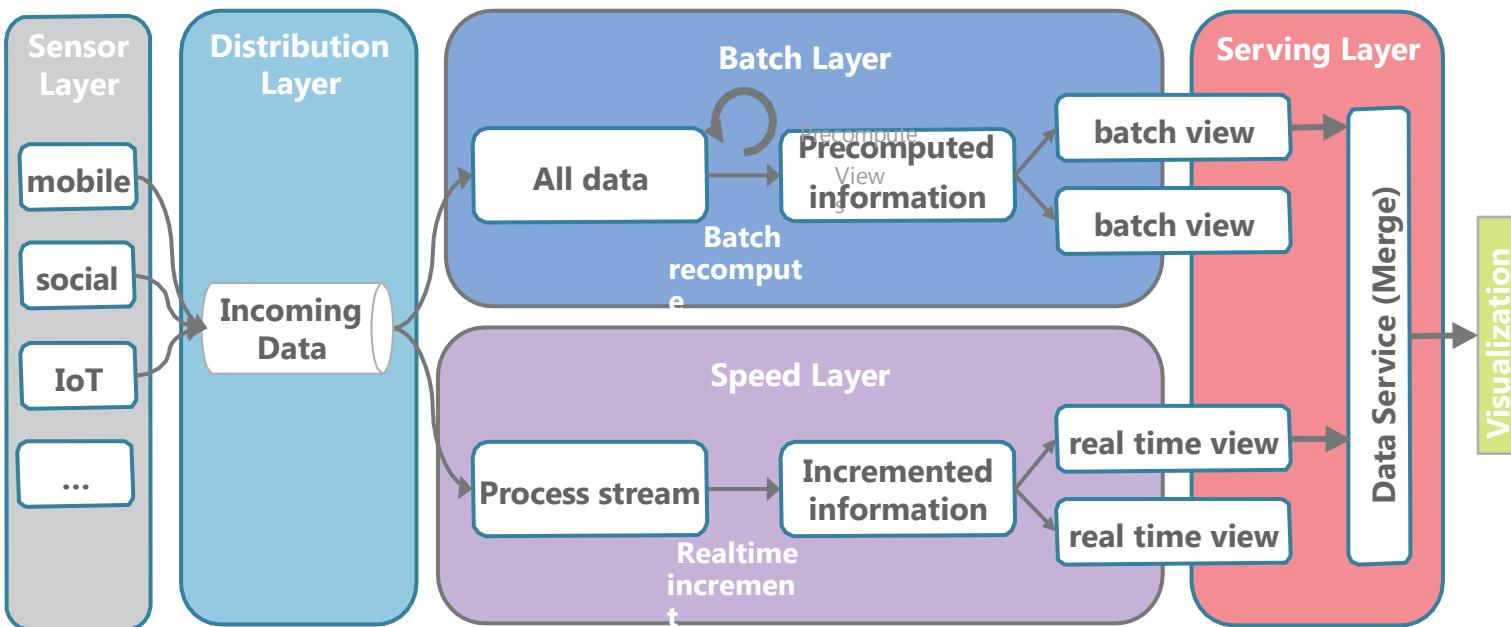
## Serving

Responsible for indexing and exposing the pre-computed batch views so that they can be queried

Exposes the incremented real-time views

Merges the batch and the real-time views into a consistent result

# Lambda Architecture



Adapted from: Marz, N. & Warren, J. (2013) Big Data. Manning.



# AGENDA

1. Big Data and Fast Data, what is it?
2. Architecting (Big) Data Systems
3. The Lambda Architecture
- 4. Use Case and the Implementation**
5. Summary and Outlook



# ■ Project Definition

- Build a platform for analyzing Twitter communications in retrospective and in real-time
- Scalability and ability for future data fusion with other information is a must
- Provide a Web-based access to the analytical information
- Invest into new, innovative and not widely-proven technology
  - PoC environment, a pre-invest for future systems

# Anatomy of a tweet

Time

Space

Content

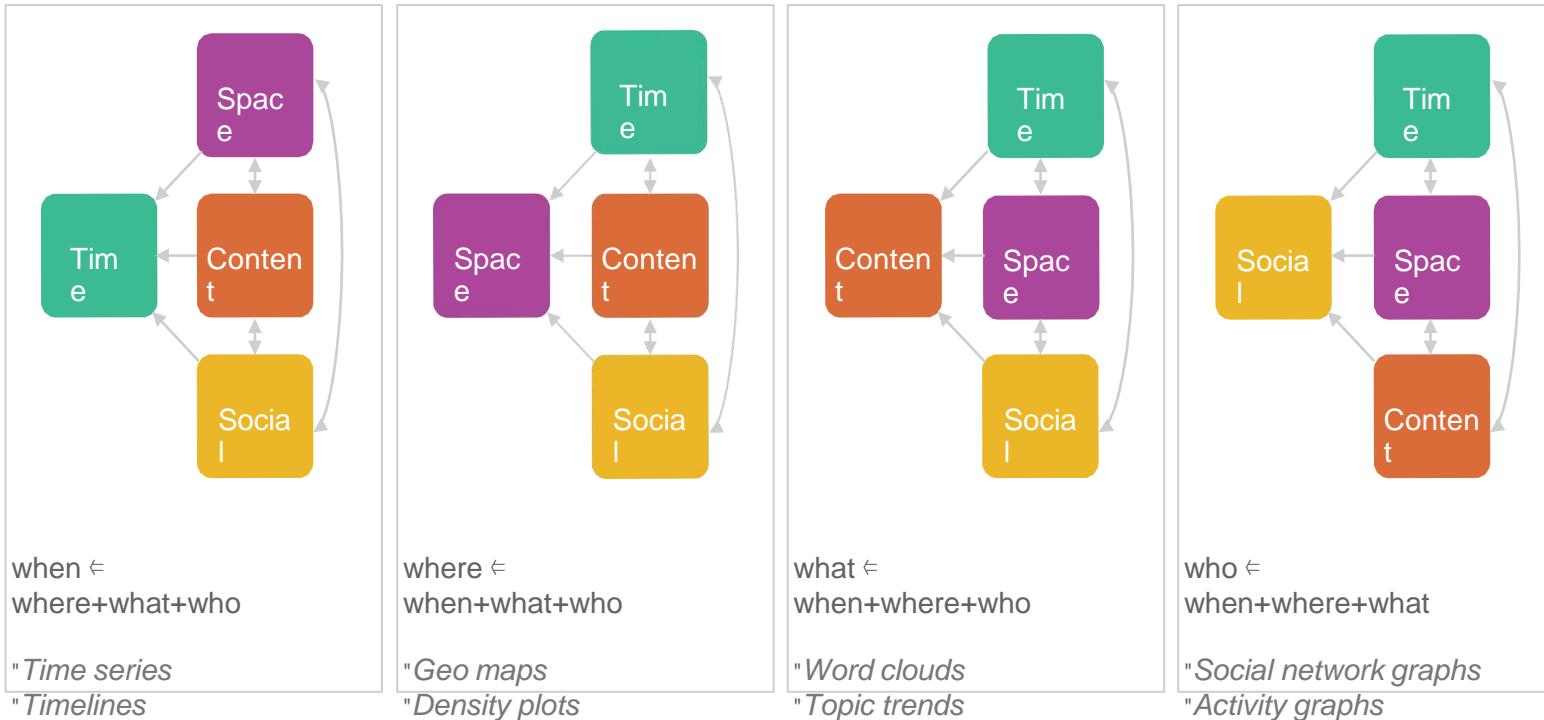
Social

Technic

```
{  
  "created_at": "Sun Aug 18 14:29:11 +0000 2013",  
  "id": 369103686938546176, "id_str": "369103686938546176",  
  
  "text": "Baloncesto preparaci\u00f3n Eslovenia, Rajoy derrota a Merkel.",  
  "#quelsepash",  
  "source": "\u003ca href=\"http://twitter.com/download/iphone\"  
  rel=\"nofollow\"\u003eTwitter for iPhone\u003c/a\u003e",  
  "truncated": false,  
  "in_reply_to_status_id": null, "in_reply_to_status_id_str": null,  
  "in_reply_to_user_id": null, "in_reply_to_user_id_str": null,  
  "in_reply_to_screen_name": null, "user": {  
    "id": 15032594, "id_str": "15032594",  
    "name": "Juan Carlos Romo\u2122",  
    "screen_name": "jcsromo", "location": "Sopuerta, Vizcaya", "url": null,  
    "description": "Portugalijo, saturado de todo, de baloncesto no. Twitter personal.",  
    "protected": false,  
    "followers_count": 1331, "friends_count": 1326, "listed_count": 31,  
    "created_at": "Fri Jun 06 21:21:22 +0000 2008", "favourites_count": 255,  
    "utc_offset": 7200, "time_zone": "Madrid", "geo_enabled": true, "verified": false,  
    "statuses_count": 22787, "lang": "es", "contributors_enabled": false,  
    "is_translator": false,  
    ...  
    "profile_image_url_https": "https://si0.twimg.com/profile_images/2649762203/  
    be4973d9eb457a45077897879c47c8b7_normal.jpeg",  
  
  "profile_banner_url": "https://pbs.twimg.com/profile_banners/15032594/  
    1371570460",  
  "profile_link_color": "#2FC2EF",  
  "profile_sidebar_border_color": "#FFFFFF", "profile_sidebar_fill_color": "#252429",  
  "profile_text_color": "#666666", "profile_use_background_image": true,  
  "default_profile": false, "default_profile_image": false, "following": null,  
  "follow_request_sent": null, "notifications": null},  
  
  "geo": {  
    "type": "Point", "coordinates": [43.28261499, -2.96464655],  
    "coordinates": {"type": "Point", "coordinates": [-2.96464655, 43.28261499]},  
    "place": {"id": "cd43ea85d651af92",  
    "url": "https://api.twitter.com/1.1/geo/id/cd43ea85d651af92.json",  
    "place_type": "city",  
    "name": "Bilbao", "full_name": "Bilbao, Vizcaya", "country_code": "ES",  
    "country": "Espa\u00f1a",  
    "bounding_box": {"type": "Polygon", "coordinates": [[[[-2.9860102, 43.2136542],  
    [-2.9860102, 43.2901452], [-2.8803248, 43.2901452], [-2.8803248, 43.2136542]]]}},  
    "attributes": {}},  
  
  "contributors": null, "retweet_count": 0, "favorite_count": 0,  
  
  "entities": {"hashtags": [{"text": "quelsepash", "indices": [58, 70]}], "symbols": [],  
  "urls": []},  
  
  "user_mentions": [], "favorited": false, "retweeted": false, "filter_level": "medium",  
  
  "lang": "es"  
}
```



# Views on Tweets in four dimensions

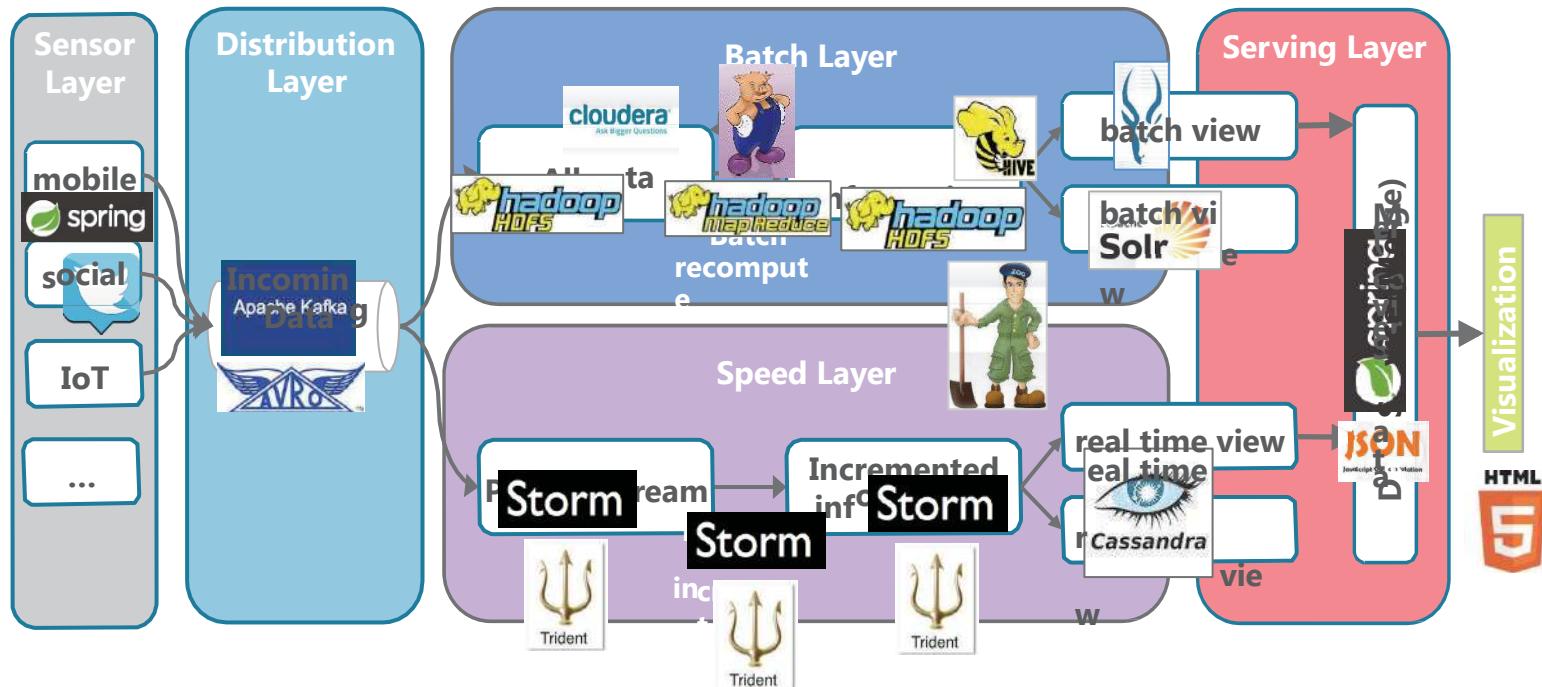


# Accessing Twitter

Quelle	Limitierungen	Zugang
Twitter's Search API	3200 / user 5000 / keyword 180 Anfragen / 15 Minuten	gratis
Twitter's Streaming API	1%-40% des Volumens	gratis
DataSift	keine	0.15 -0.20\$ / unit
Gnip	keine	Auf Anfrage

# Lambda Architecture

Open Source Frameworks for implementing a Lambda Architecture



# Lambda Architecture in Action

## Twitter Horsebird Client (hbc)

### Distribution

- Twitter Java API over Streaming API
- 

**Spring Framework**  
Popular Java Framework used to modularize part of the logic (sensor and serving layer)

## Apache Kafka

- Simple messaging framework based on file system to distribute information to both batch and speed layer

## Apache Avro

- Serialization system for efficient cross-language RPC and persistent data storage

## JSON

- open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs.

## Cloudera

Distribution of Apache Hadoop: HDFS, MapReduce, Hive, Flume, Pig, Impala

## Cloudera

### Impala

- distributed query execution engine that runs against data stored in HDFS and HBase

## Apache Zookeeper

- Distributed, highly available coordination service. Provides primitives such as distributed locks

## Apache Storm &

### Trident

- distributed, fault-tolerant realtime computation system

## Apache

### Cassandra

- distributed database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure

# Facts & Figures

14 active twitter feeds

• ~ 14 million tweets/day (> 5 billion tweets/year)

• ~ 8 GB/day raw data, compressed (2 DVDs)

• 66 GB storage capacity / day  
(replication & views/results included)

Cluster of 10 nodes

• ~100 processors

• ~40 TB HD capacity in total; 46% used

• >500 GB RAM

Currently in total

• 2.7 TB Raw Data

• 1.1 TB Pre-Processed data in Impala

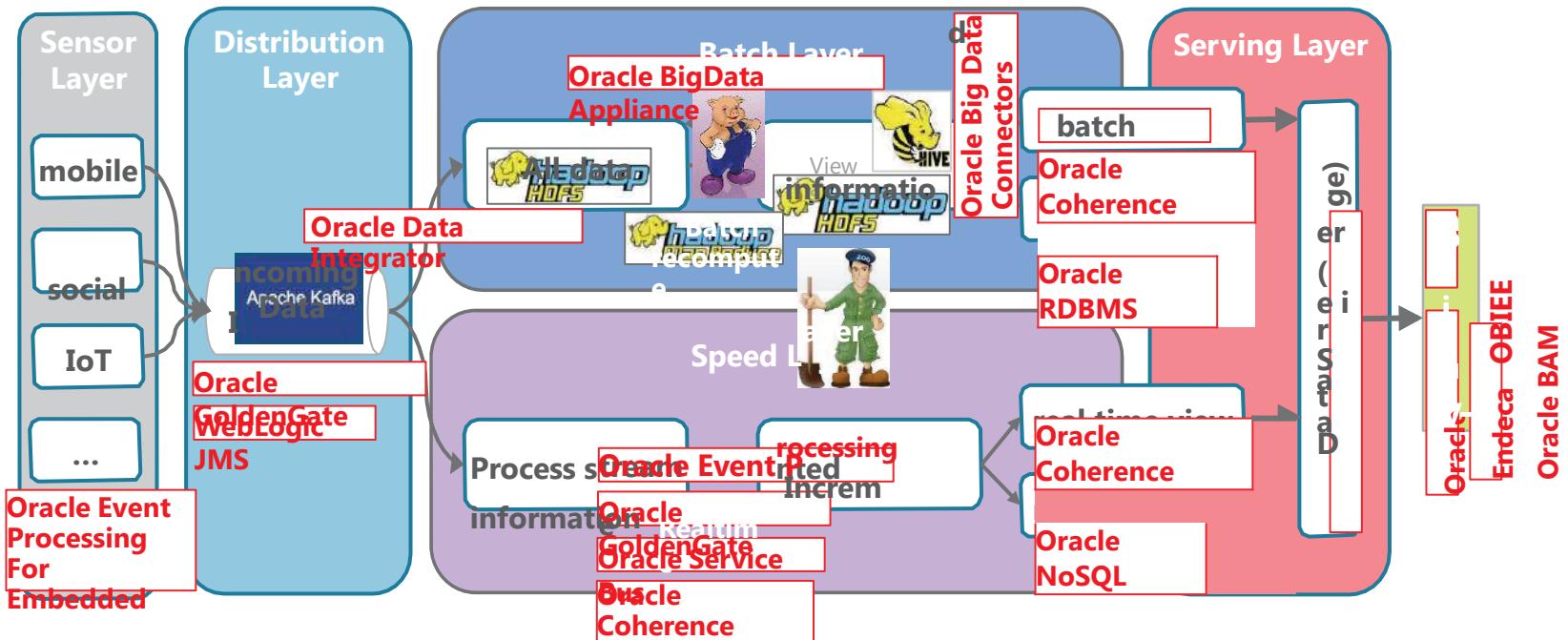
• 1 TB Solr indices for full text search

Cloudera 4.7.0 with Hadoop, Pig, Hive, Impala and Solr

Kafka 0.7, Storm 0.9, DataStax Enterprise Edition

# Lambda Architecture with Oracle Product Stack

Possible implementation with Oracle Product stack





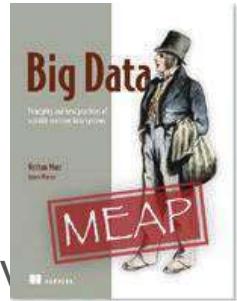
# AGENDA

- 1. Big Data and Fast Data, what is it?**
- 2. Architecting (Big) Data Systems**
- 3. The Lambda Architecture**
- 4. Use Case and the Implementation**
- 5. Summary and Outlook**





## Summary – The lambda architecture



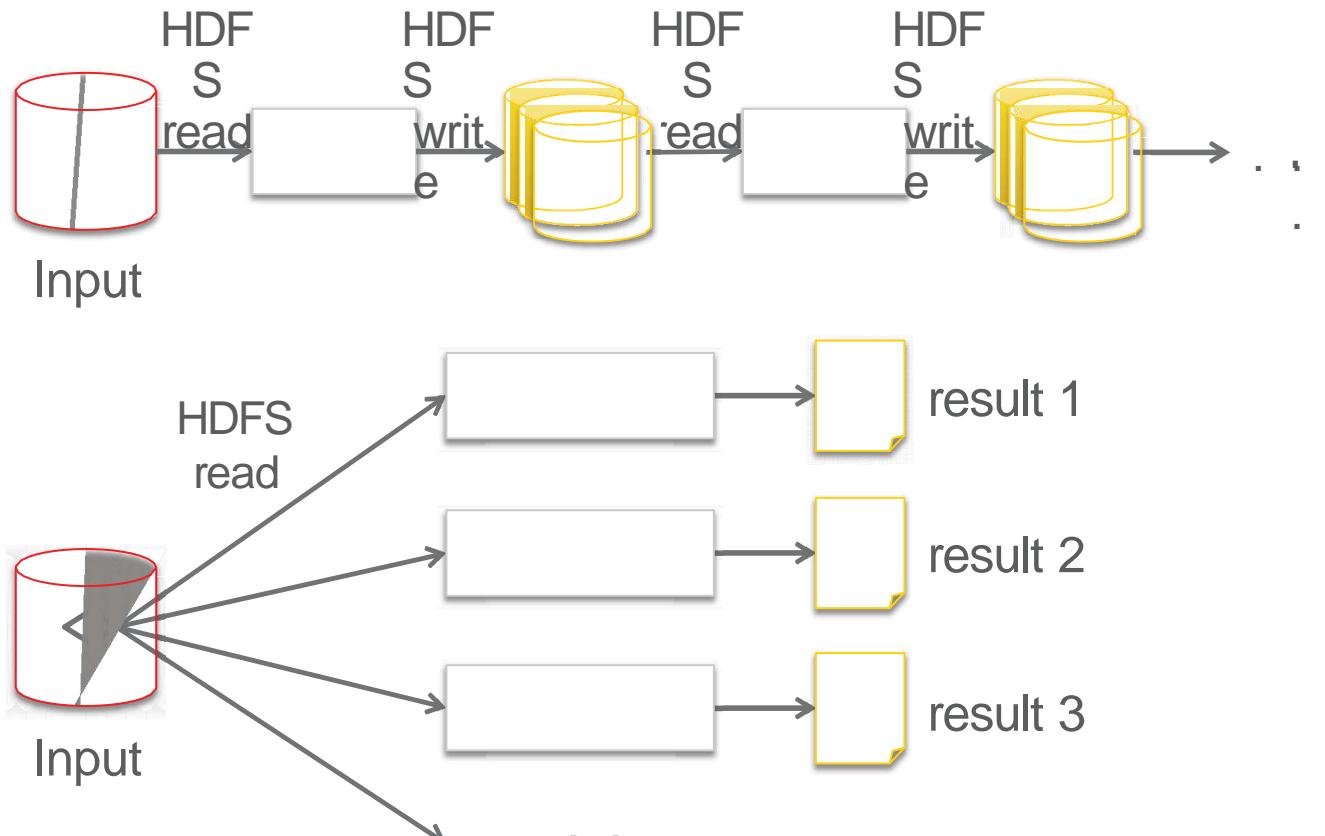
- Can discard batch views and real-time views and recreate everything from scratch
- Mistakes corrected via re-computation
- Scalability through platform and distribution
- Data storage layer optimized independently from query resolution layer
- Still in a early stage !. But a very interesting idea!
  - Today a zoo of technologies are needed => Infrastructure group might not like it
  - Better with so-called Hadoop distributions and Hadoop V2 (YARN)



# Alternative Approaches – Motivation

## Data Sharing in Map Reduce

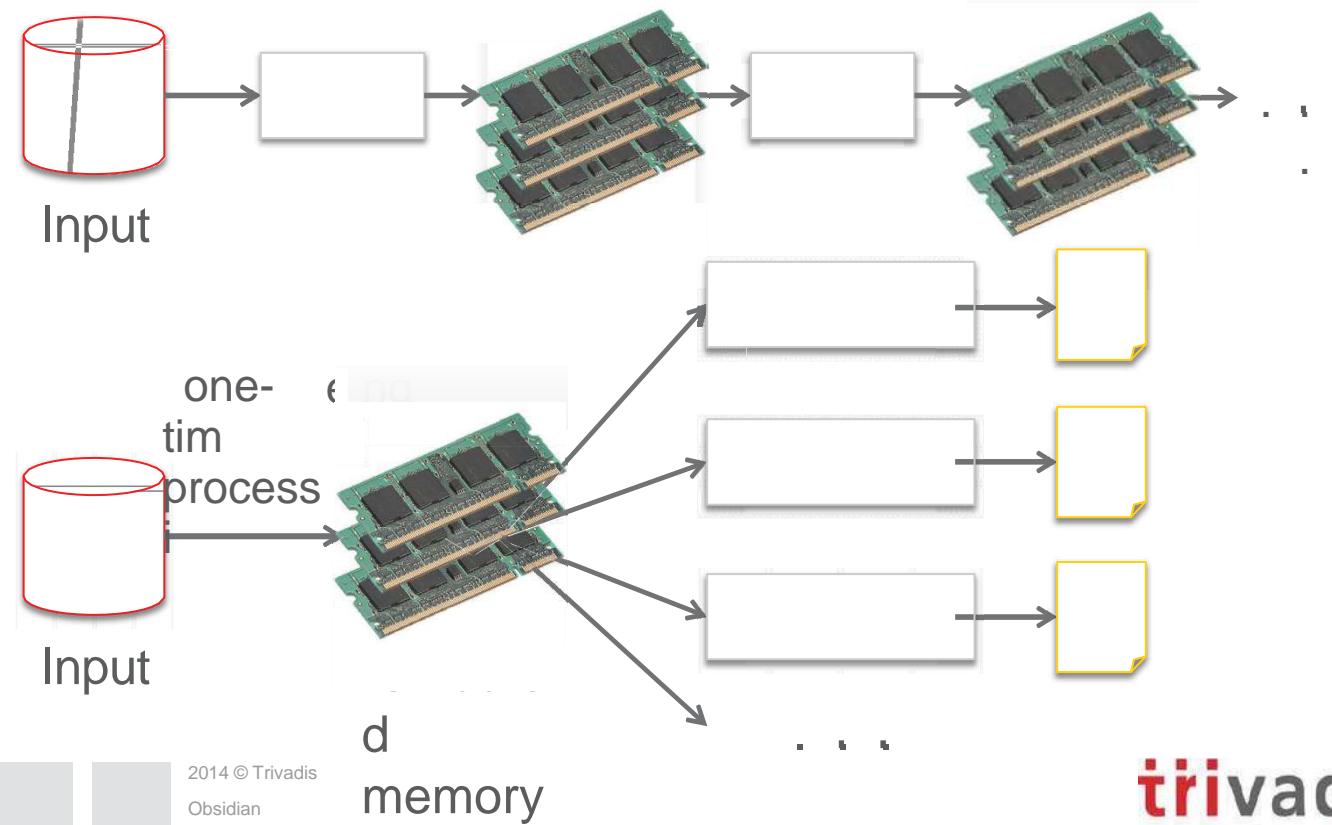
!



# ■ Alternative Approaches – Motivation

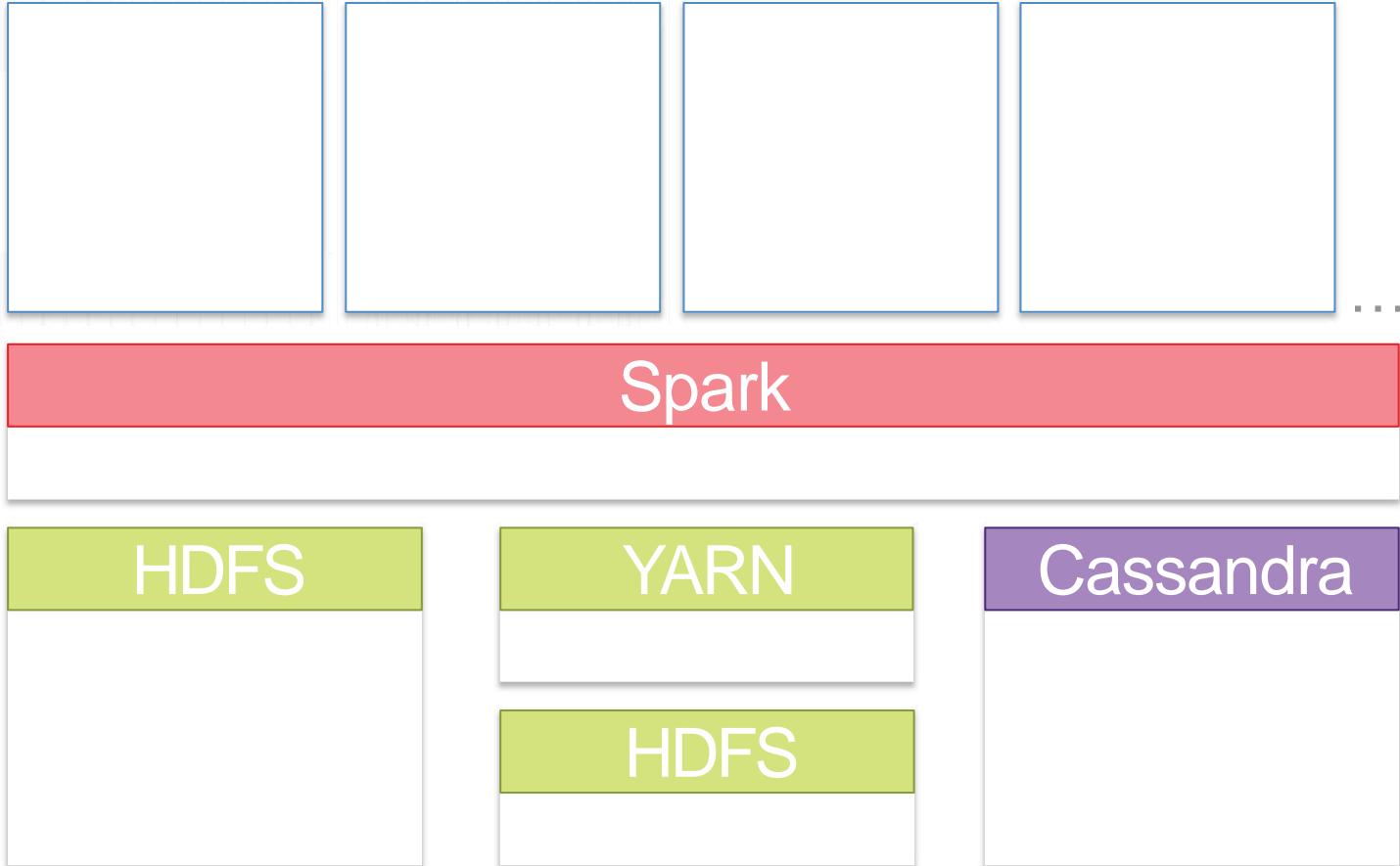
What we would like

!

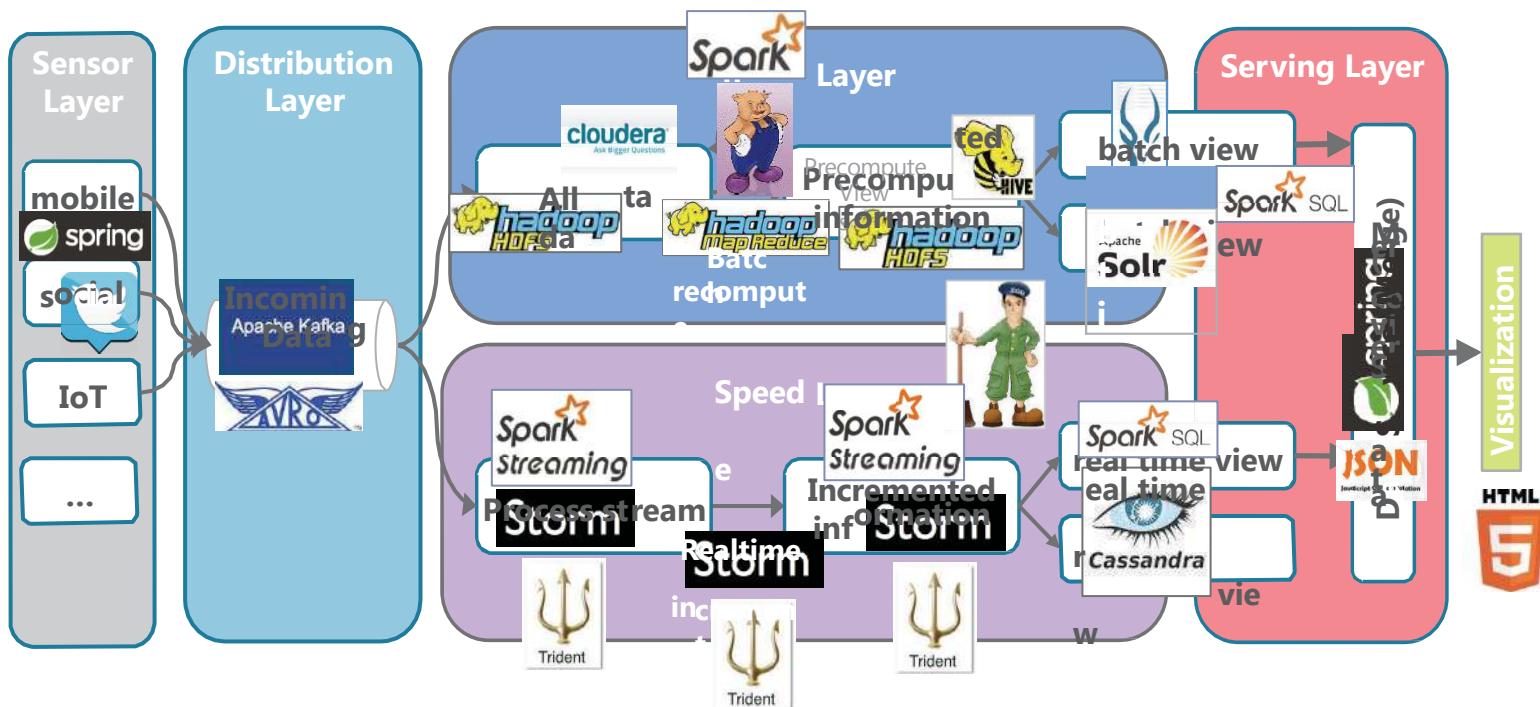




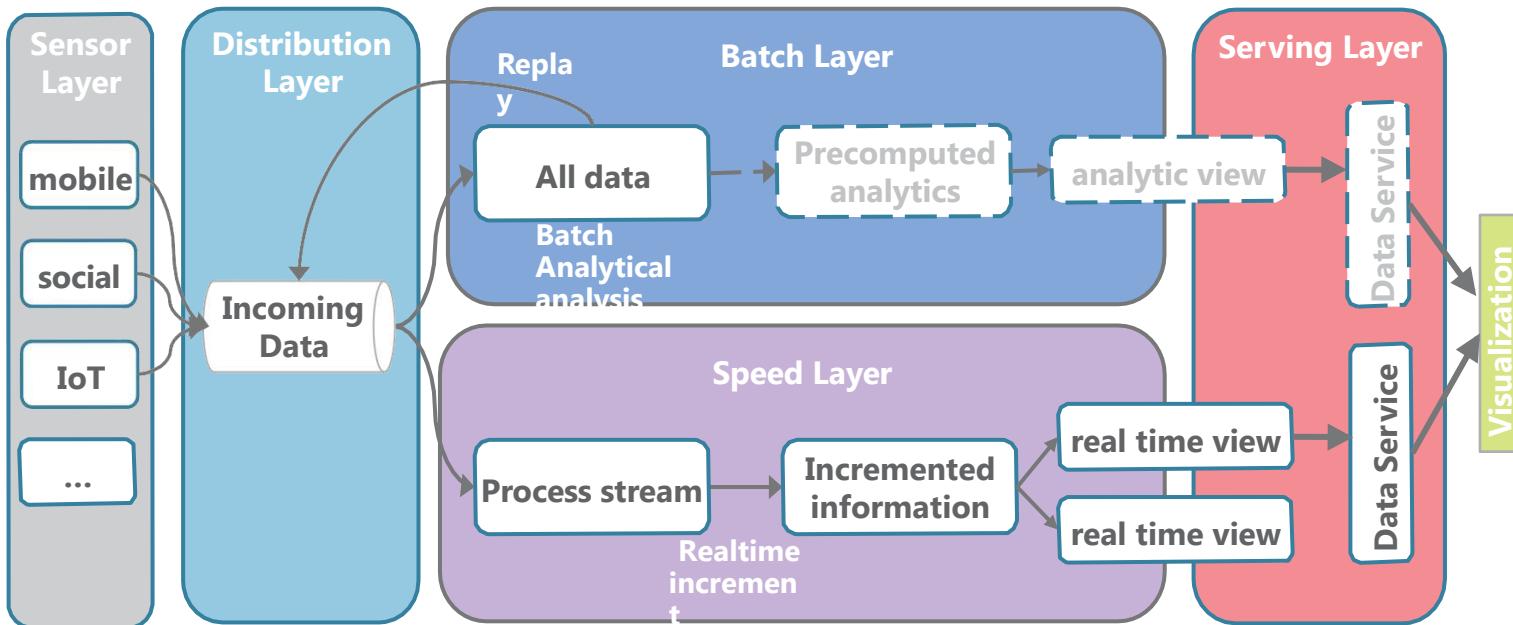
## Alternatives – Apache Spark



# ■ Alternative Technologies – Apache Spark



# “Kappa Architecture”



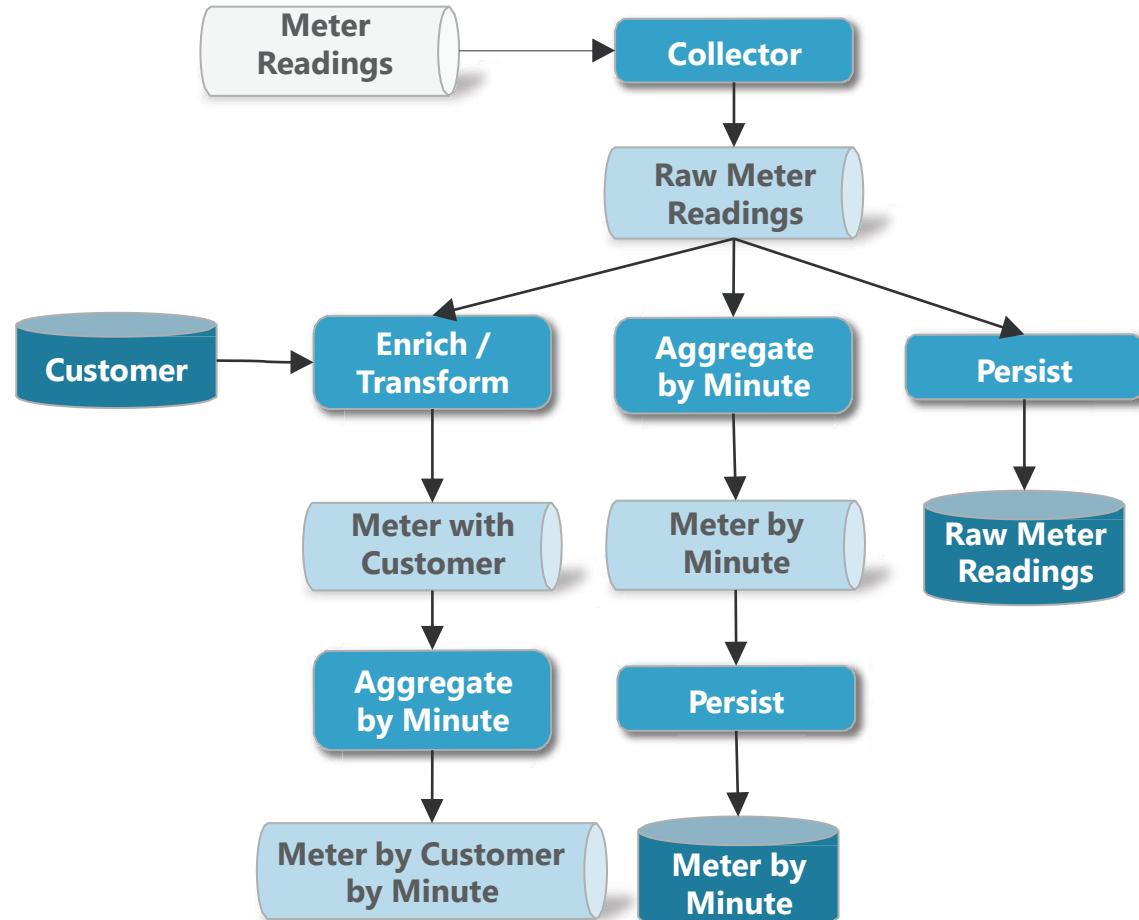
Adapted from: Marz, N. & Warren, J. (2013) Big Data. Manning.

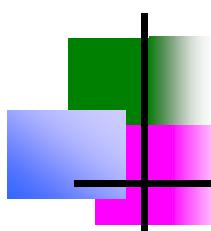
# Unified Log Processing Architecture

Stream processing allows for computing feeds off of other feeds

Derived feeds are no different than original feeds they are computed off

Single deployment of “Unified Log” but logically different feeds





---

**Thank You :)**



PRESNTED BY

O'REILLY®

cloudera®

# Hadoop Application Architectures: Fraud Detection

Strata + Hadoop World, London 2016

[tiny.cloudera.com/app-arch-london](http://tiny.cloudera.com/app-arch-london)

[tiny.cloudera.com/app-arch-questions](http://tiny.cloudera.com/app-arch-questions)

Gwen Shapira | @gwenshap

Jonathan Seidman | @jseidman

Ted Malaska | @ted\_malaska

Mark Grover | @mark\_grover

[strataconf.com](http://strataconf.com)

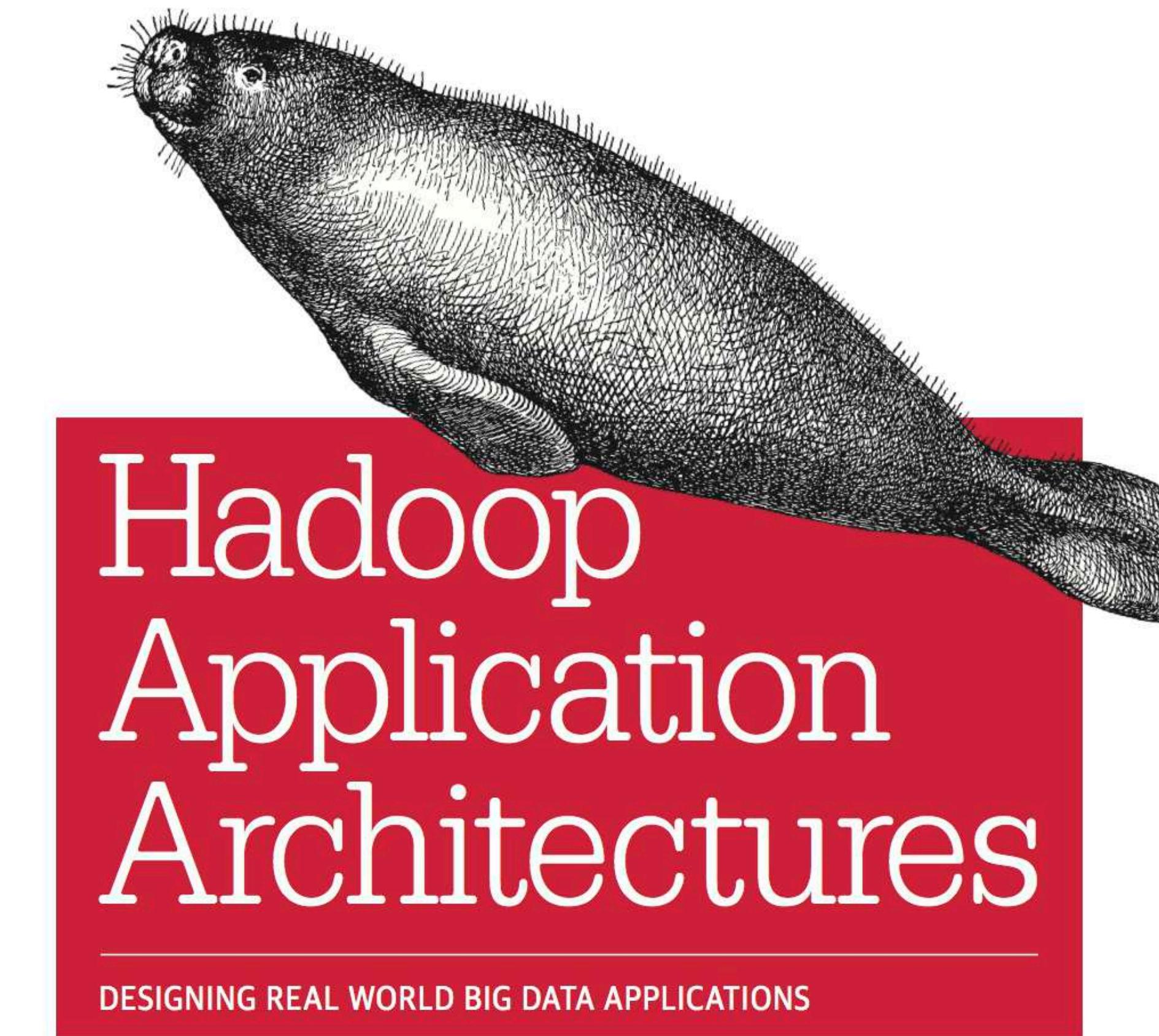
#StrataHadoop

# Logistics

- Break at 10:30 – 11:00 AM
- Questions at the end of each section
- Slides at [tiny.cloudera.com/app-arch-london](http://tiny.cloudera.com/app-arch-london)
- Code at <https://github.com/hadooparchitecturebook/fraud-detection-tutorial>

# About the book

- @hadooparchbook
- hadooparchitecturebook.com
- github.com/hadooparchitecturebook
- slideshare.com/hadooparchbook



Mark Grover, Ted Malaska,  
Jonathan Seidman & Gwen Shapira

# About the presenters

## Ted Malaska

- Principal Solutions Architect at Cloudera
- Previously, lead architect at FINRA
- Contributor to Apache Hadoop, HBase, Flume, Avro, Pig and Spark

## Jonathan Seidman

- Senior Solutions Architect/ Partner Enablement at Cloudera
- Contributor to Apache Sqoop.
- Previously, Technical Lead on the big data team at Orbitz, co-founder of the Chicago Hadoop User Group and Chicago Big Data

# About the presenters

## Gwen Shapira

- System Architect at Confluent
- Committer on Apache Kafka, Sqoop
- Contributor to Apache Flume

## Mark Grover

- Software Engineer on Spark at Cloudera
- Committer on Apache Bigtop, PMC member on Apache Sentry(incubating)
- Contributor to Apache Spark, Hadoop, Hive, Sqoop, Pig, Flume

# Strata+ Hadoop

---

WORLD

PRES EN TED BY

O'REILLY®

cloudera®

# Case Study Overview

## Fraud Detection

[strataconf.com](http://strataconf.com)

#StrataHadoop

# Credit Card Transaction Fraud

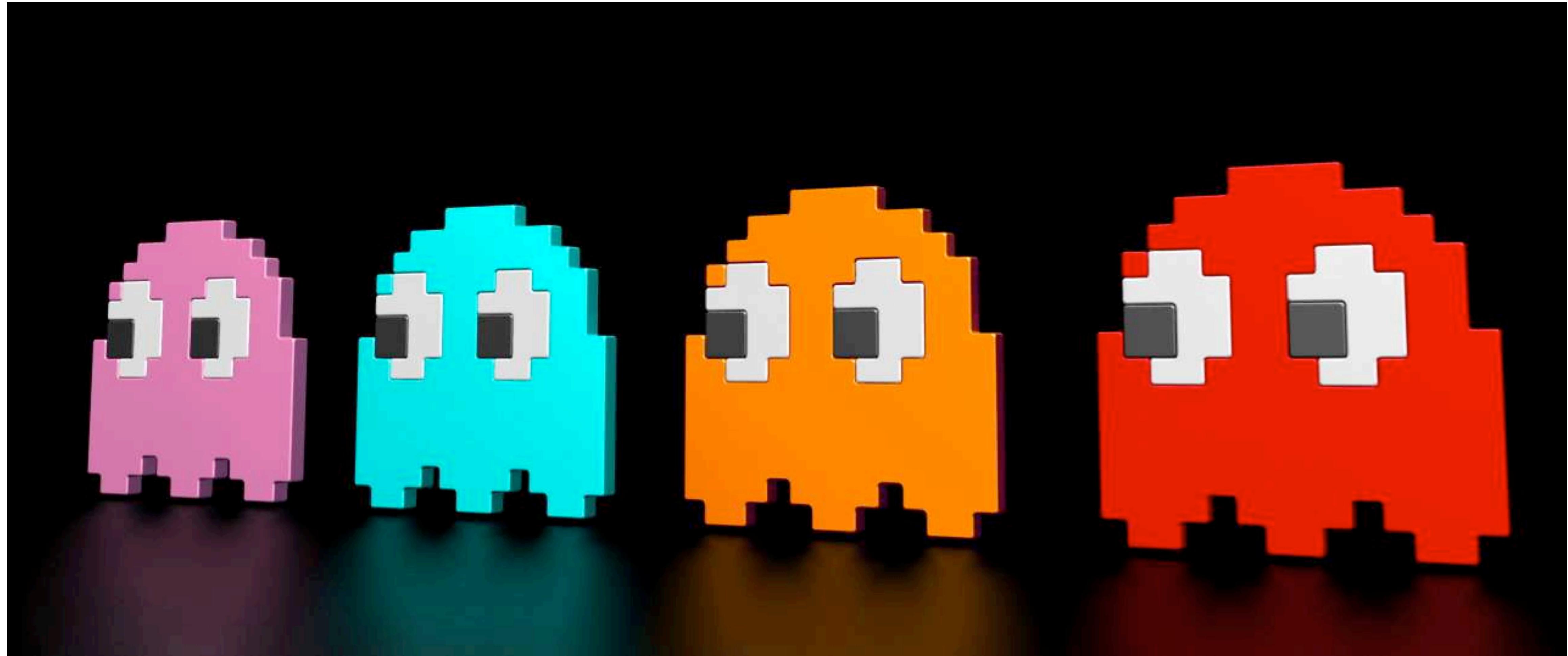


#StrataHadoop

Questions? [tiny.cloudera.com/app-arch-questions](http://tiny.cloudera.com/app-arch-questions)

Strata + Hadoop  
WORLD

# Video Game Strategy



#StrataHadoop

Questions? [tiny.cloudera.com/app-arch-questions](http://tiny.cloudera.com/app-arch-questions)

Strata + Hadoop  
WORLD

# Health Insurance Fraud



#StrataHadoop

Questions? [tiny.cloudera.com/app-arch-questions](http://tiny.cloudera.com/app-arch-questions)

Strata+Hadoop  
WORLD

# Network anomaly detection



#StrataHadoop

Questions? [tiny.cloudera.com/app-arch-questions](http://tiny.cloudera.com/app-arch-questions)

Strata + Hadoop  
WORLD

# Strata+ Hadoop

---

WORLD

PRESNTED BY

O'REILLY®

cloudera®

**Ok, enough with the  
use-cases**

Can we move on?

[strataconf.com](http://strataconf.com)

#StrataHadoop

# How Do We React

- Human Brain at Tennis
  - *Muscle Memory*
  - *Fast Thought*
  - *Slow Thought*



# Back to network anomaly detection

---

- Muscle memory – for quick action (e.g. reject events)
  - Real time alerts
  - Real time dashboard
- Platform for automated learning and improvement
  - Real time (fast thought)
  - Batch (slow thought)
- Slow thought
  - Audit trail and analytics for human feedback

# Strata+ Hadoop

---

WORLD

PRESENTED BY

O'REILLY®

cloudera®

## Use case

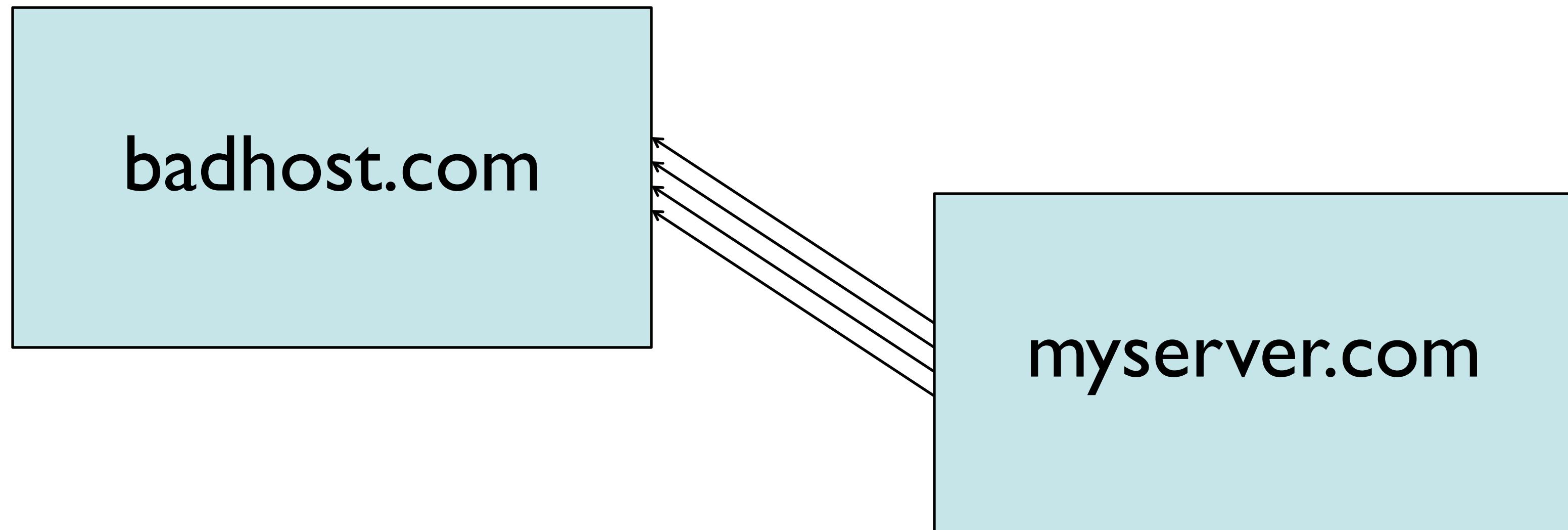
### Network Anomaly Detection

[strataconf.com](http://strataconf.com)

#StrataHadoop

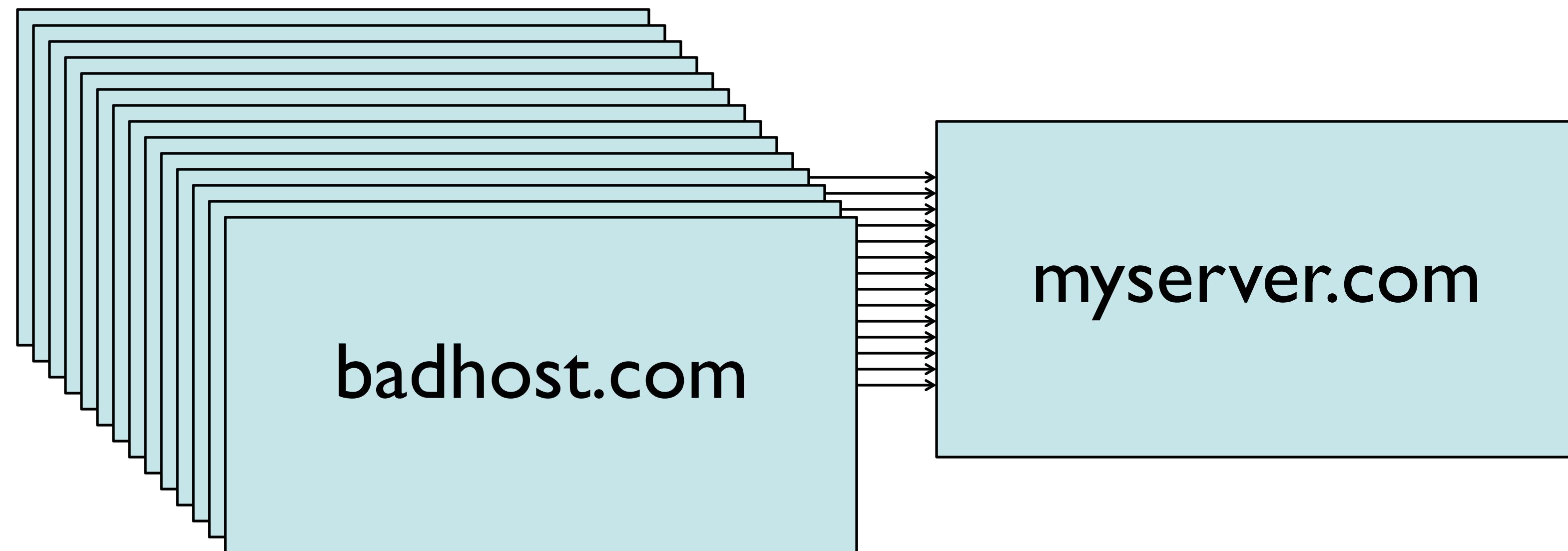
# Use Case – Network Anomaly Detection

- “Beaconing” – Malware “phoning home”.



# Use Case – Network Anomaly Detection

- Distributed Denial of Service attacks.



# Use Case – Network Anomaly Detection

- Network intrusion attempts – attempted or successful break-ins to a network.



# Use Case – Network Anomaly Detection

---

All of these require the ability to detect deviations from known patterns.

# Other Use Cases

Could be used for any system requiring detection of anomalous events:



Credit card transactions



Market transactions



Internet of Things

Etc...

# Input Data – NetFlow

---

- Standard protocol defining a format for capturing network traffic flow through routers.
- Captures network data as *flow records*. These flow records provide information on source and destination IP addresses, traffic volumes, ports, etc.
- We can use this data to detect normal and anomalous patterns in network traffic.

# Strata+ Hadoop

---

WORLD

PRESNTED BY

O'REILLY®

cloudera®

[strataconf.com](http://strataconf.com)

#StrataHadoop

# Why is Hadoop a great fit?

# Hadoop

- In traditional sense
  - HDFS (append-only file system)
  - MapReduce (really slow but robust execution engine)
- Is **not** a great fit

# Why is Hadoop a Great Fit?

- But the Hadoop ecosystem is!
- More than just MapReduce + YARN + HDFS

# Volume

- Have to maintain millions of device profiles
- Retain flow history
- Make and keep track of automated rule changes

# Velocity

- Events arriving concurrently and at high velocity
- Make decisions in real-time

# Variety

- Maintain simple counters in profile (e.g. throughput thresholds, purchase thresholds)
- Iris or finger prints that need to be matched

# Challenges of Hadoop Implementation



#StrataHadoop

Questions? [tiny.cloudera.com/app-arch-questions](http://tiny.cloudera.com/app-arch-questions)

Strata+Hadoop  
WORLD

# Challenges of Hadoop Implementation



# Challenges - Architectural Considerations

- Storing state (for real-time decisions):
  - Local Caching? Distributed caching (e.g. Memcached)? Distributed storage (e.g. HBase)?
- Profile Storage
  - HDFS? HBase?
- Ingestion frameworks (for events):
  - Flume? Kafka? Soop?
- Processing engines (for background processing):
  - Storm? Spark? Trident? Flume interceptors? Kafka Streams?
- Data Modeling
- Orchestration
  - Do we still need it for real-time systems?

# Strata+ Hadoop

---

WORLD

PRESNTED BY

O'REILLY®

cloudera®

[strataconf.com](http://strataconf.com)

#StrataHadoop

# Case Study Requirements

## Overview

# But First...

# Real-Time? Near Real-Time? Stream Processing?

#StrataHadoop

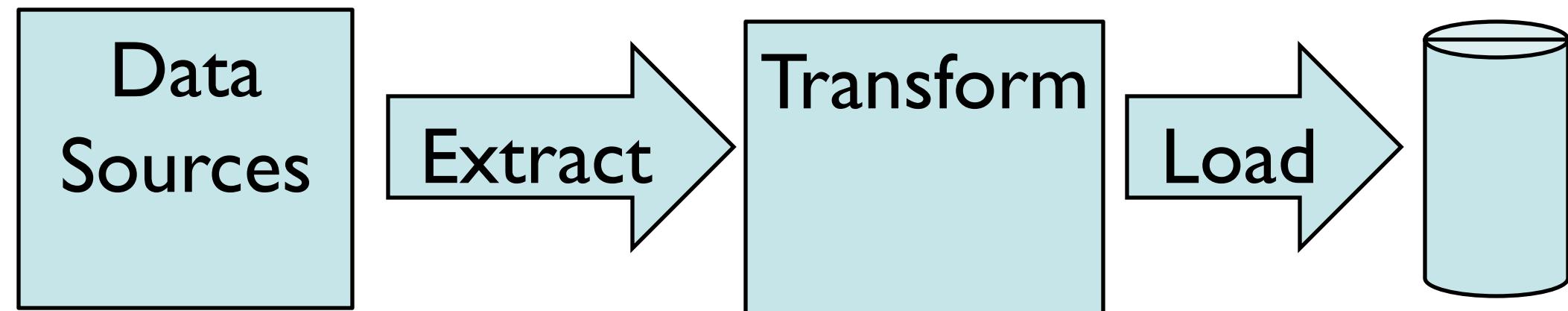
Questions? [tiny.cloudera.com/app-arch-questions](http://tiny.cloudera.com/app-arch-questions)

Strata + Hadoop  
WORLD

# Some Definitions

- Real-time – well, not really. Most of what we're talking about here is...
- Near real-time:
  - Stream processing – continual processing of incoming data.
  - Alerting – immediate (well, as fast as possible) responses to incoming events. Getting closer to real-time.

Typical Hadoop Processing – Minutes to Hours



Fraud Detection

seconds

Stream  
Processing

milliseconds

Alerts  
!

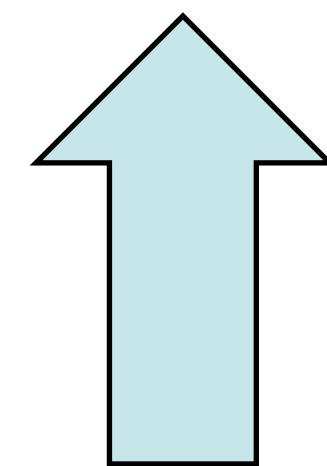
# Requirements – Storage/Modeling

- Need long term storage for large volumes of profile, event, transaction, etc. data.



# Requirements – Storage/Modeling

- Need to be able to quickly retrieve specific profiles and respond quickly to incoming events.



# Requirements – Storage/Modeling

- Need to store sufficient data to analyze whether or not fraud is present.



# Requirements – Alerts

- Need to be able to respond to incoming events quickly (milliseconds).
- Need high throughput and low latency.
- Processing requirements are minimal.
- Two types of alerts: “atomic”, and “atomic with context”



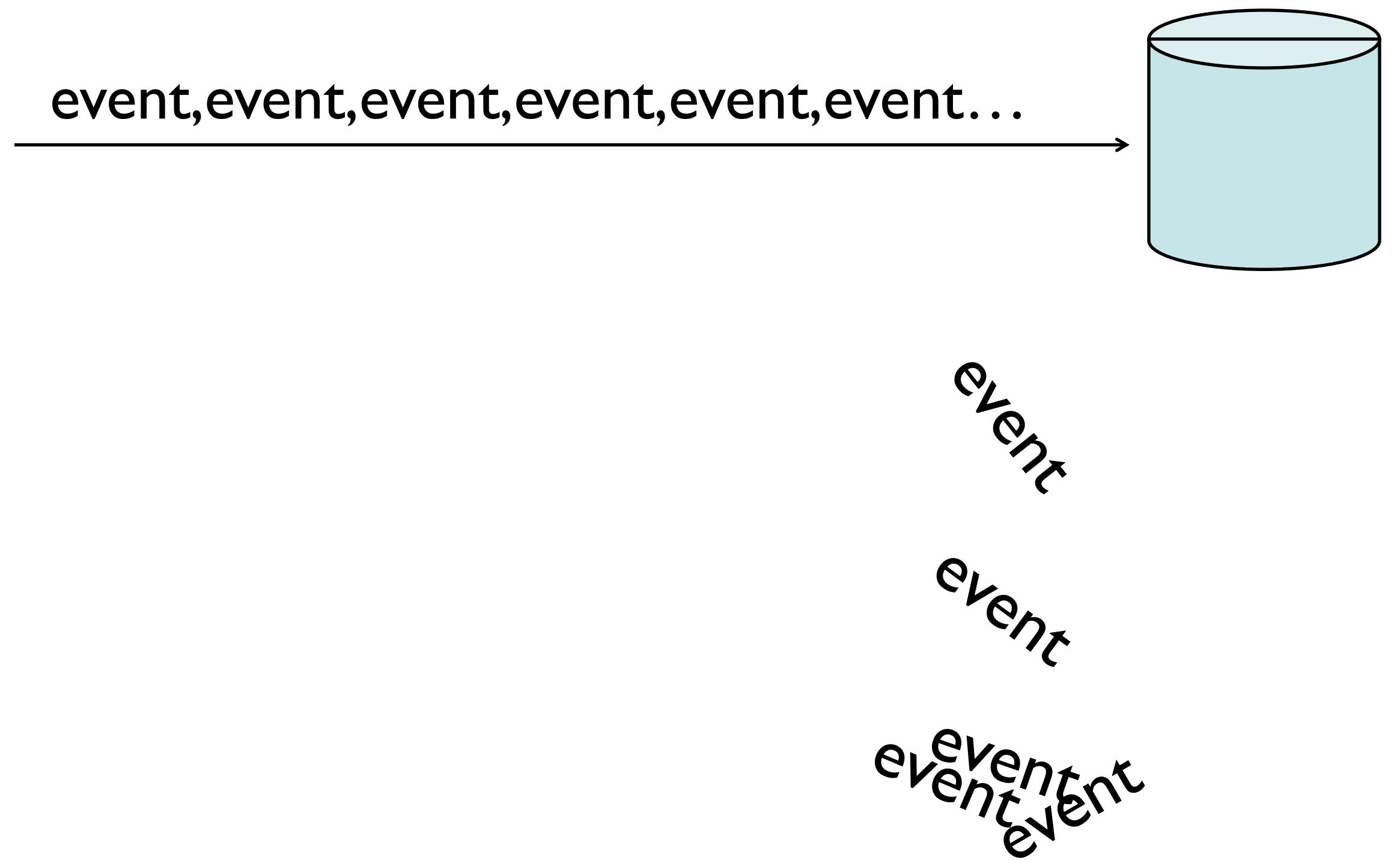
# Requirements – Ingestion



Photo Credit: USGS - Sirenia Project

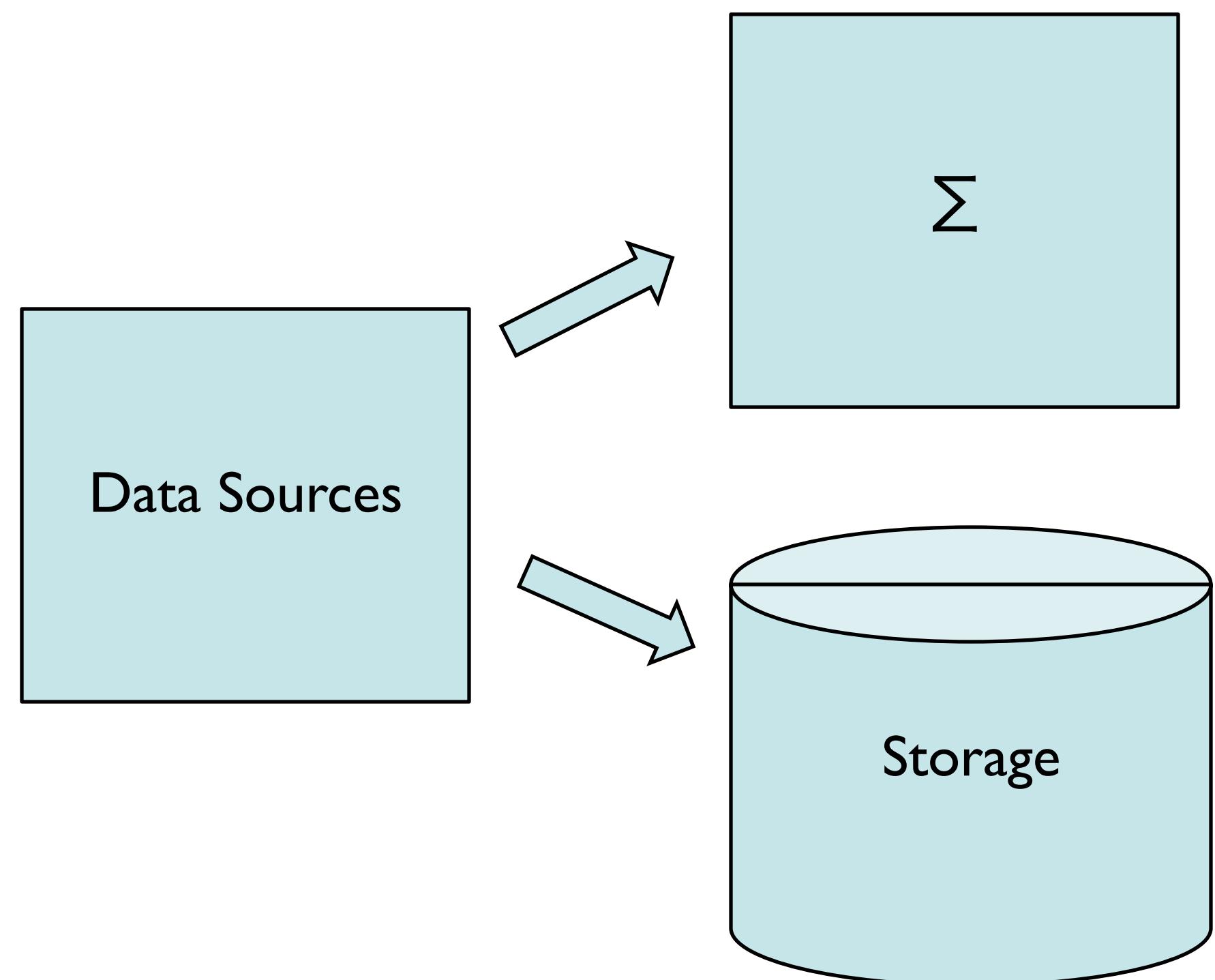
# Requirements – Ingestion

- Reliable – we don't want to lose events.



# Requirements – Ingestion

- Support for multiple targets.



# Requirements – Ingestion

- Needs to support high throughput of large volumes of events.

# Requirements – Stream Processing

- A few seconds to minutes response times.
- Need to be able to detect trends, thresholds, etc. across profiles.
- Quick processing more important than 100% accuracy.



# Requirements – Batch Processing

- Non real-time, “off-line”, exploratory processing and reporting.
- Data needs to be available for analysts and users to analyze with tools of choice – SQL, MapReduce, etc.
- Results need to be able to feed back into NRT processing to adjust rules, etc.



"Parmigiano reggiano factory". Licensed under CC BY-SA 3.0 via Commons - [https://commons.wikimedia.org/wiki/File:Parmigiano\\_reggiano\\_factory.jpg](https://commons.wikimedia.org/wiki/File:Parmigiano_reggiano_factory.jpg#/media/File:Parmigiano_reggiano_factory.jpg)

# Strata+ Hadoop

---

WORLD

PRES EN TED BY

O'REILLY®

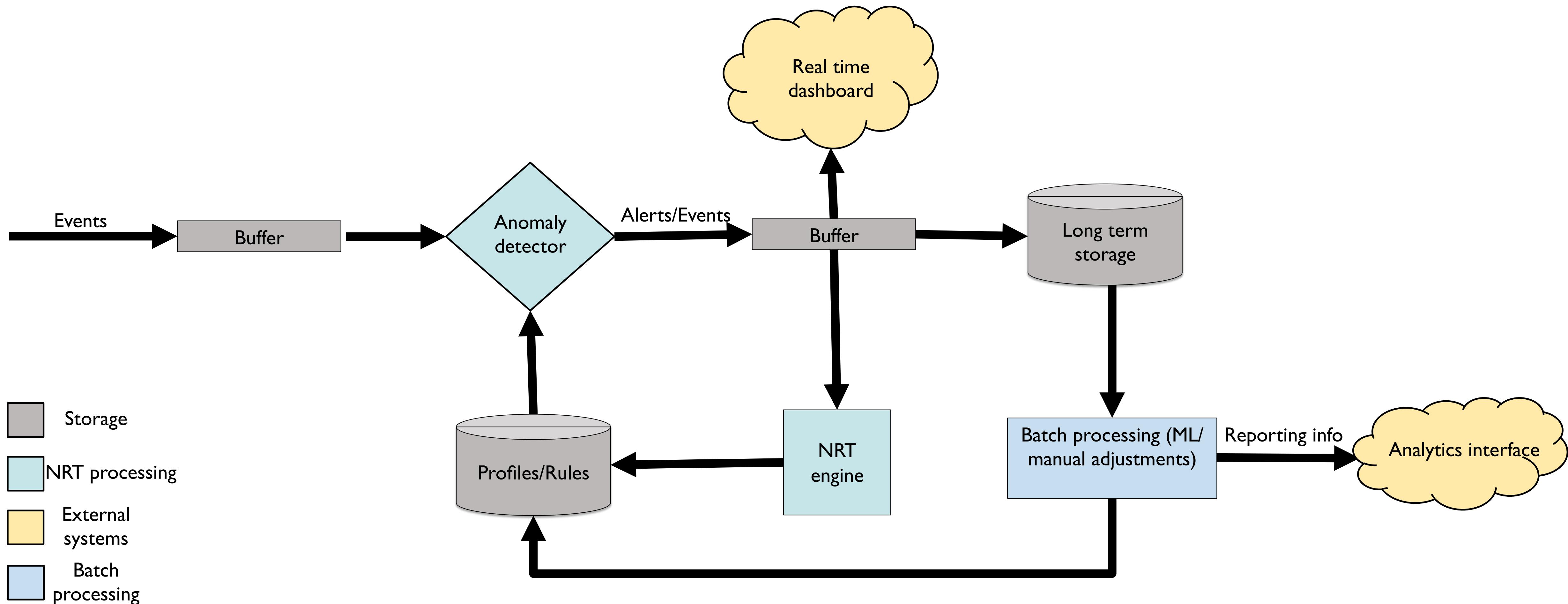
cloudera®

# High level architecture

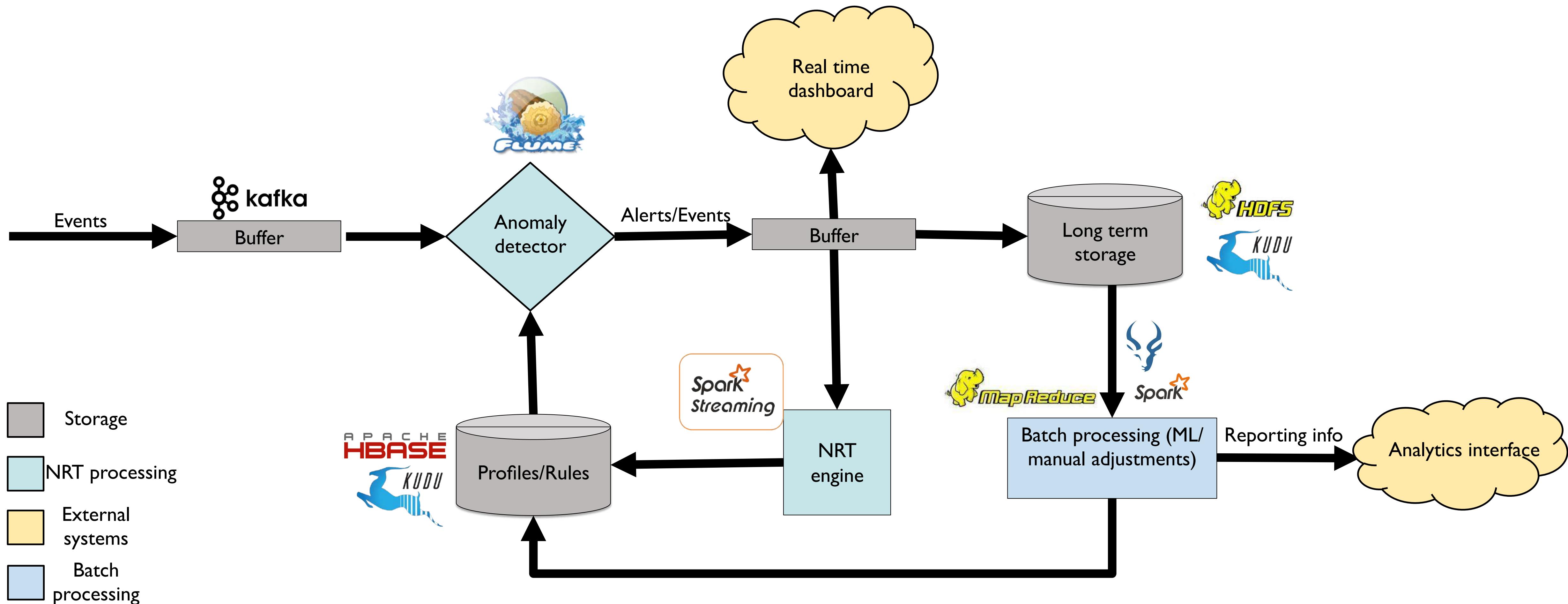
[strataconf.com](http://strataconf.com)

#StrataHadoop

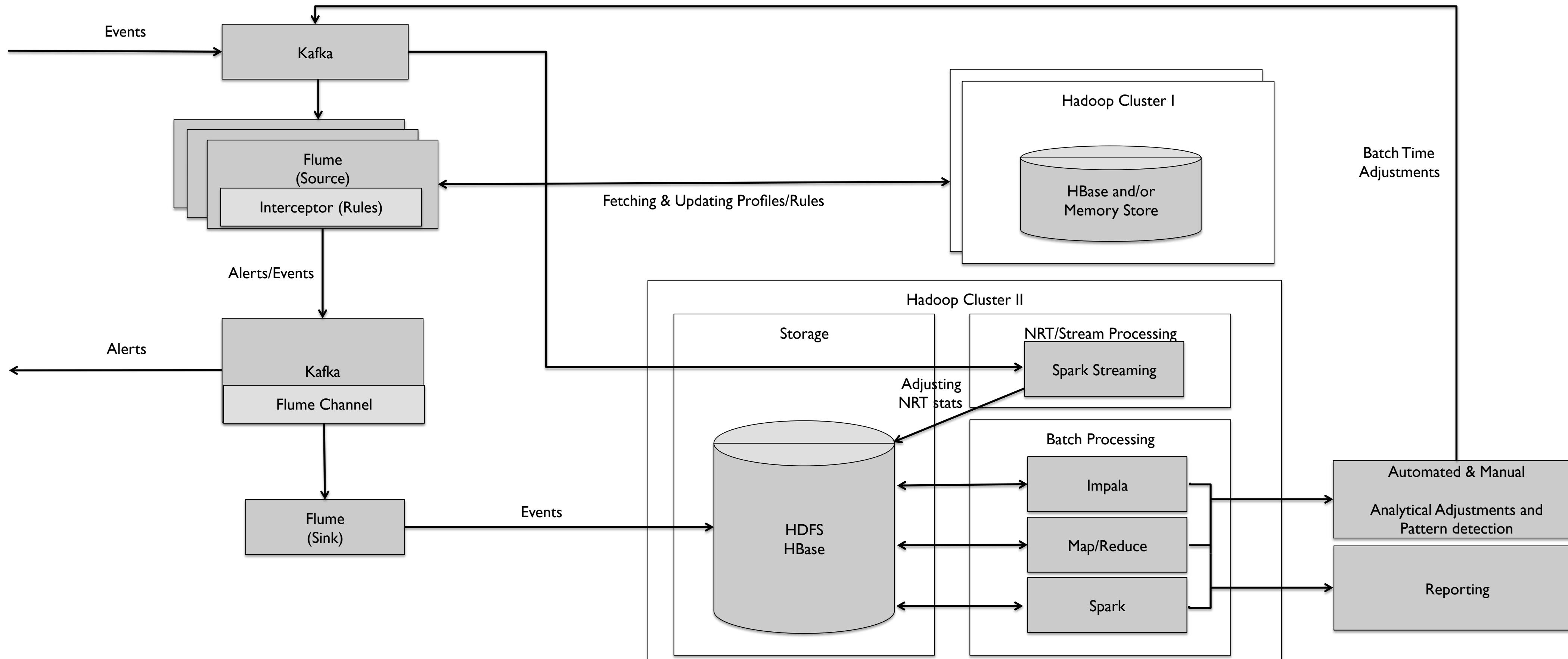
# High level architecture



# High level architecture



# Full Architecture



# Strata+ Hadoop

---

WORLD

PRES EN TED BY

O'REILLY®

cloudera®

# Storage Layer

## Considerations

[strataconf.com](http://strataconf.com)

#StrataHadoop

# Storage Layer Considerations

- Two likely choices for long-term storage of data:



# Data Storage Layer Choices



- Stores data directly as files
- Fast scans
- Poor random reads/writes
- Stores data as Hfiles on HDFS
- Slow scans
- Fast random reads/writes

# Storage Considerations

- Batch processing and reporting requires access to all raw and processed data.
- Stream processing and alerting requires quickly fetching and saving profiles.

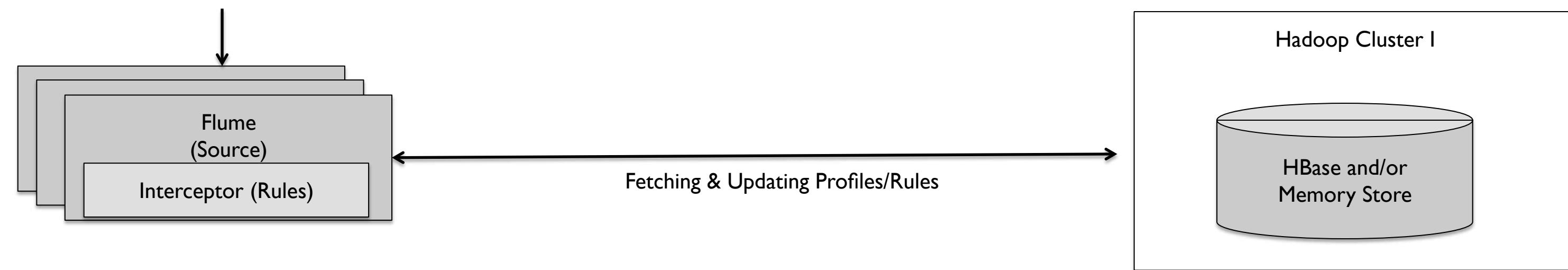
# Data Storage Layer Choices



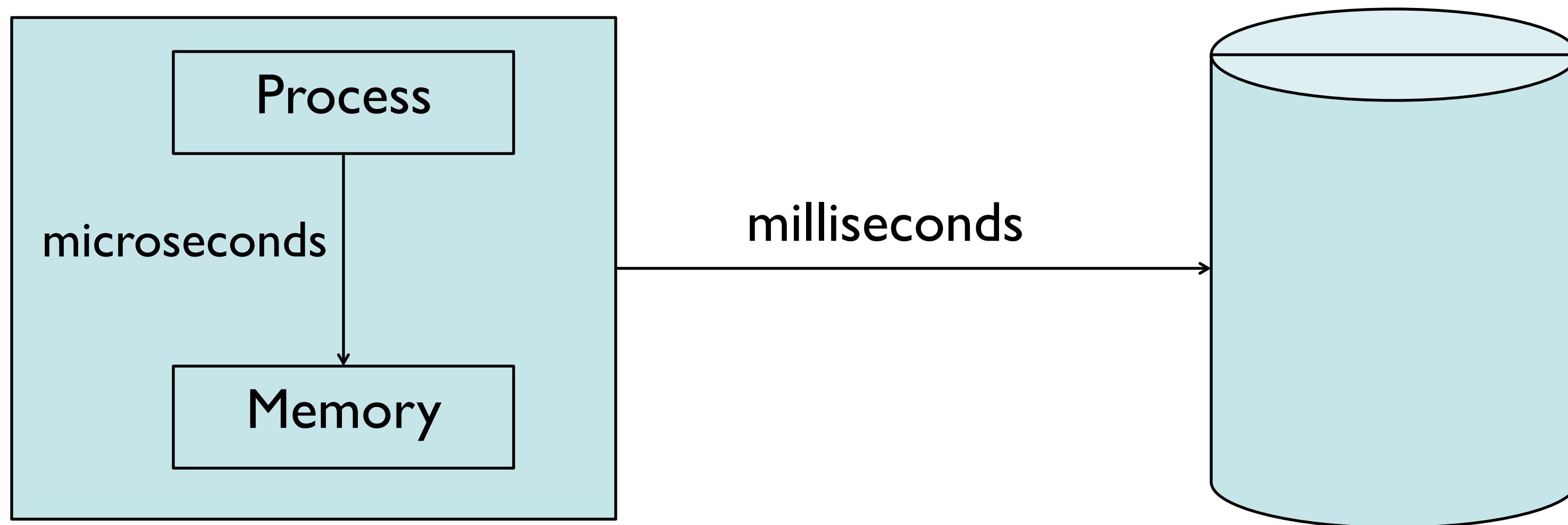
- For ingesting raw data.
- Batch processing, also some stream processing.
- For reading/writing profiles.
- Fast random reads and writes facilitates quick access to large number of profiles.

# But...

## Is HBase fast enough for fetching profiles?



# What About Caching?



# Caching Options

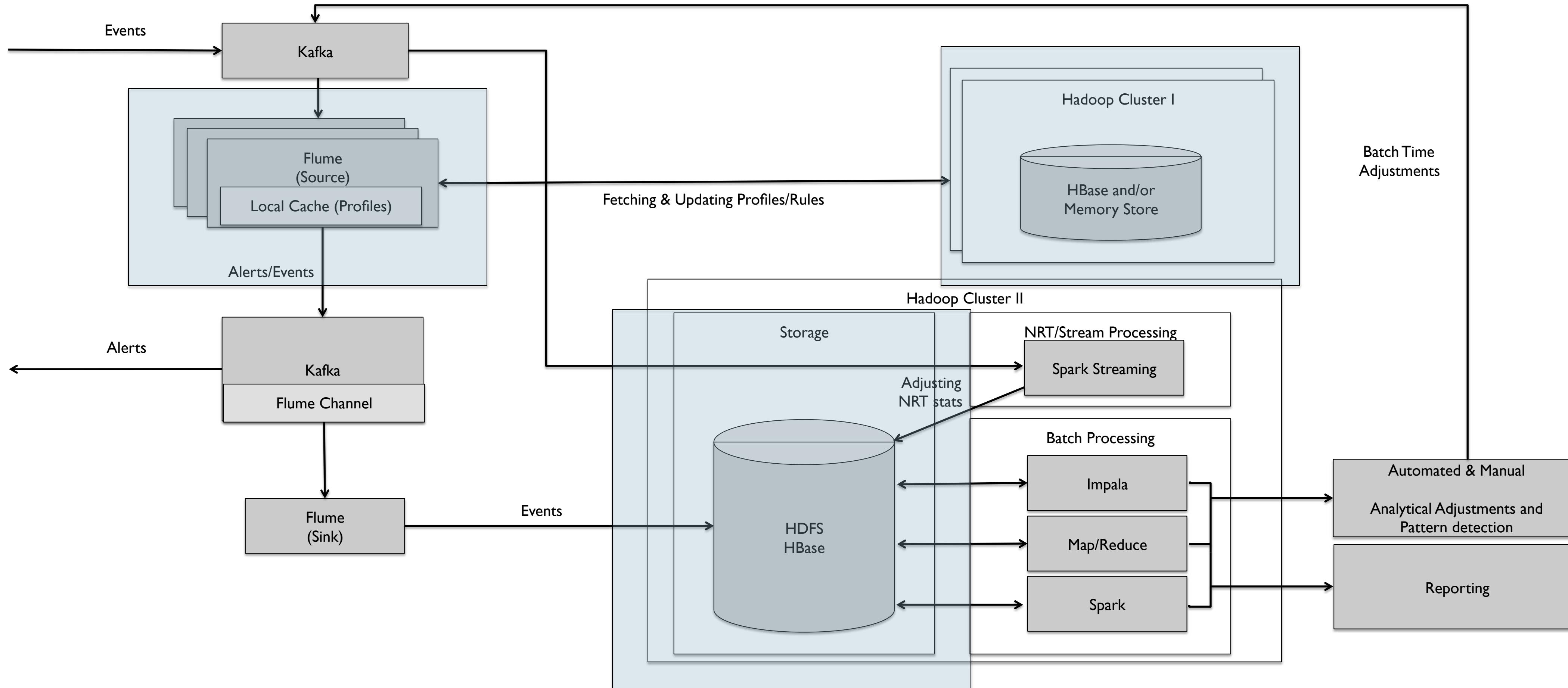
- Local in-memory cache (on-heap, off-heap).
- Remote cache
  - Distributed cache (Memcached, Oracle Coherence, etc.)
  - HBase BlockCache

# Caching – HBase BlockCache

---

- Allows us to keep recently used data blocks in memory.
- Note that this will require recent versions of HBase and specific configurations on our cluster.

# Our Storage Choices



# Strata+ Hadoop

---

WORLD

PRES EN TED BY

O'REILLY®

cloudera®

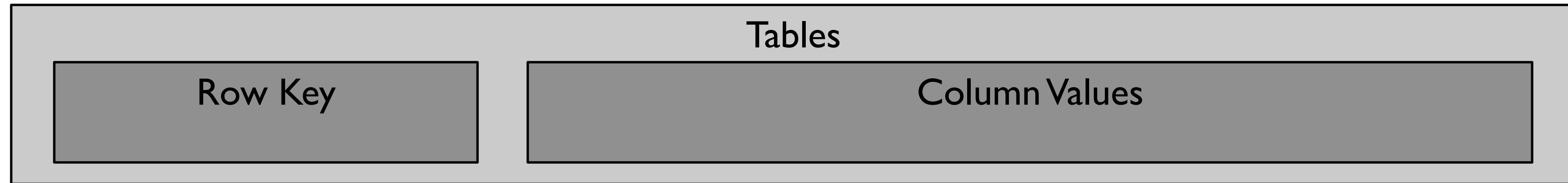
# HBase Data Modeling

## Considerations

[strataconf.com](http://strataconf.com)

#StrataHadoop

# HBase Data Modeling Considerations



# HBase Data Modeling Considerations

- Tables
  - *Minimize # of Tables*
  - *Reduce/Remove Joins*
  - *# Region*
  - *Split policy*

# HBase Data Modeling Considerations

- RowKeys
  - *Location Location Location*
  - *Salt is good for you*

# HBase Data Modeling Considerations



The image shows the Cloudera website header. It features the Cloudera logo with the tagline "Ask Bigger Questions". On the right side, there is a search bar with a magnifying glass icon. Below the search bar are five navigation links: COMMUNITY, DOCUMENTATION, DOWNLOADS, TRAINING, and BLOGS.

Hadoop & Big Data

Our Customers

FAQs

Blog

## How-to: Scan Salted Apache HBase Tables with Region-Specific Key Ranges in MapReduce

by Justin Kestelyn (@kestelyn) | June 24, 2015 |  no comments

**Thanks to Pengyu Wang, software developer at FINRA, for permission to republish this post.**

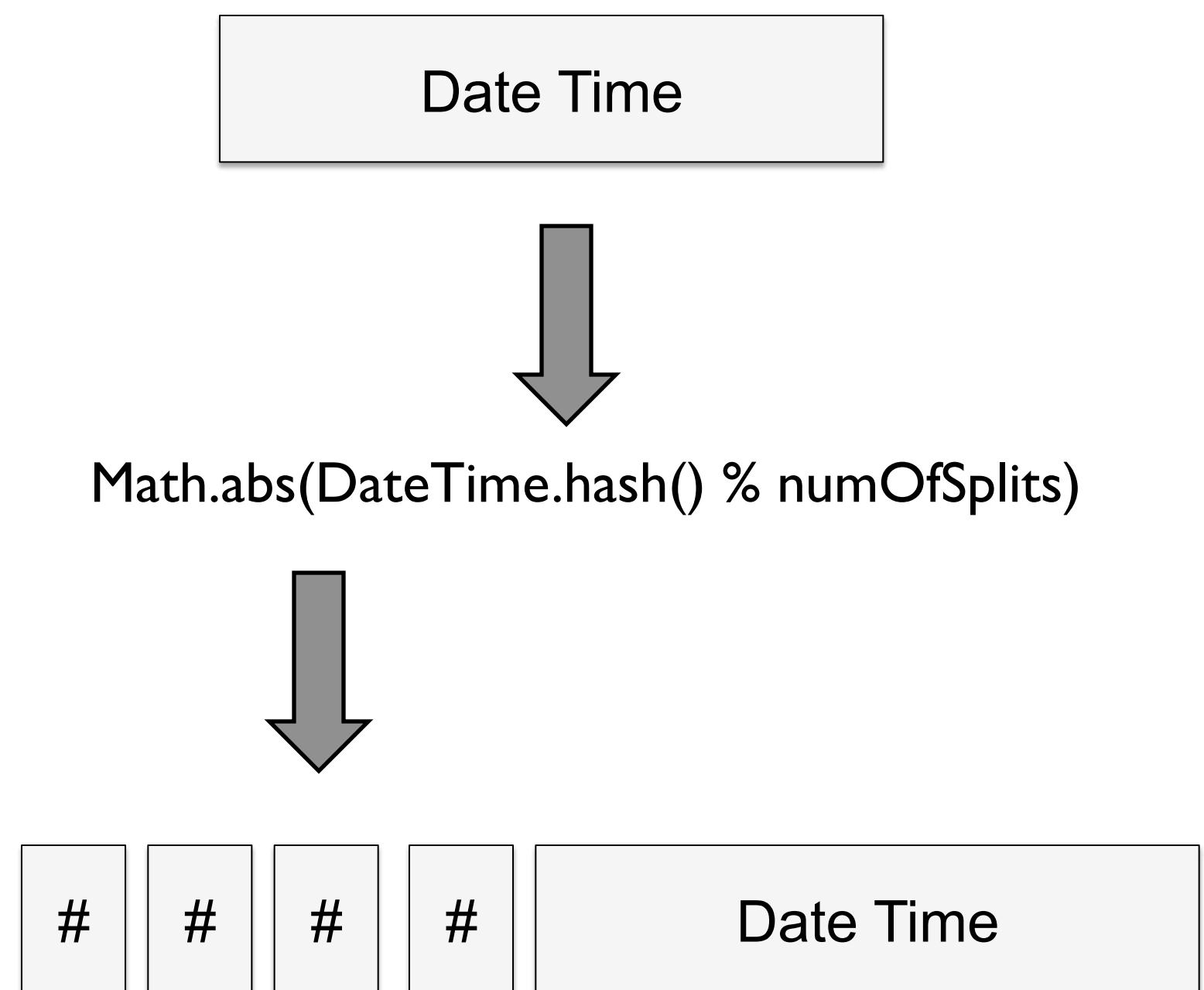
Salted Apache HBase tables with pre-split is a proven effective HBase solution to provide uniform workload distribution across RegionServers and prevent hot spots during bulk writes. In this design, a row key is made with a logical key plus salt at the beginning. One way of generating salt is by calculating  $n / \text{number of regions}$  modulo on the

#StrataHadoop

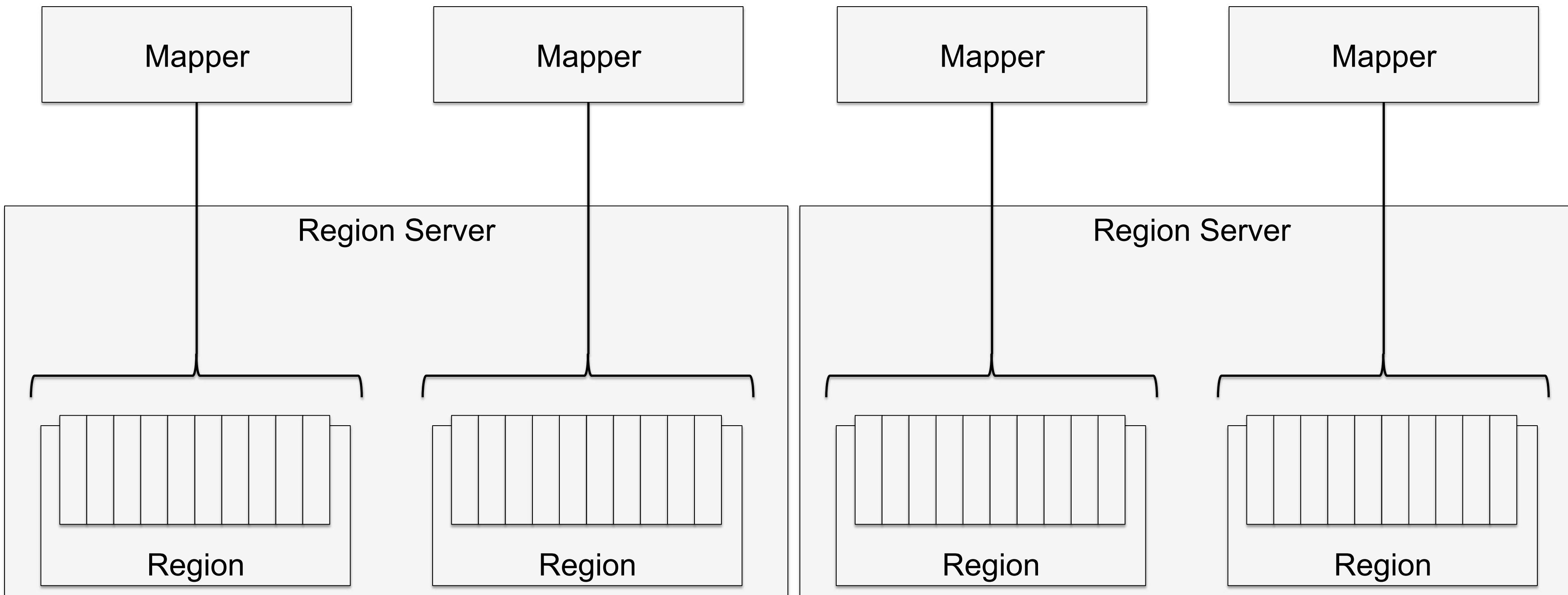
Questions? [tiny.cloudera.com/app-arch-questions](http://tiny.cloudera.com/app-arch-questions)

Strata+Hadoop  
WORLD

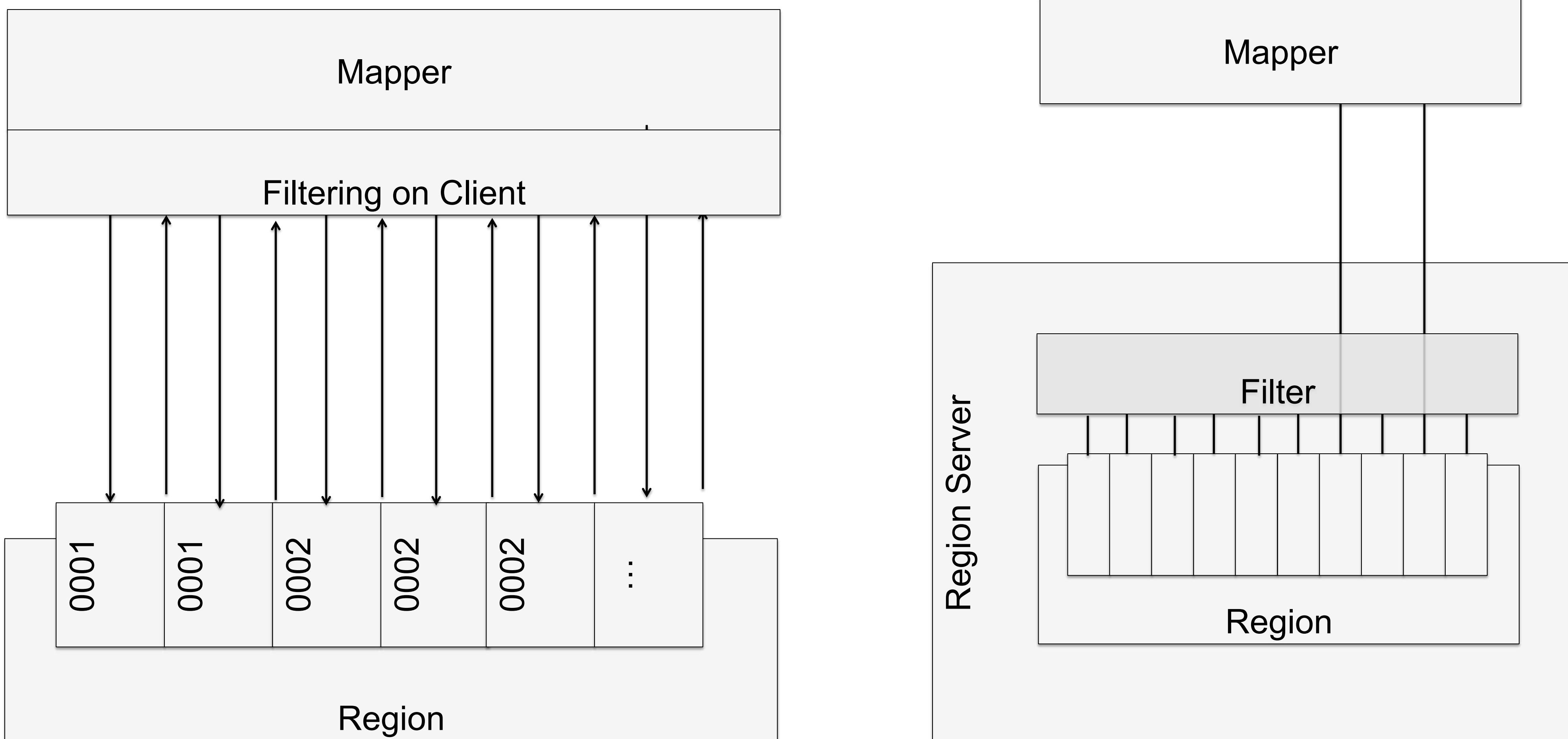
# What is a Salt



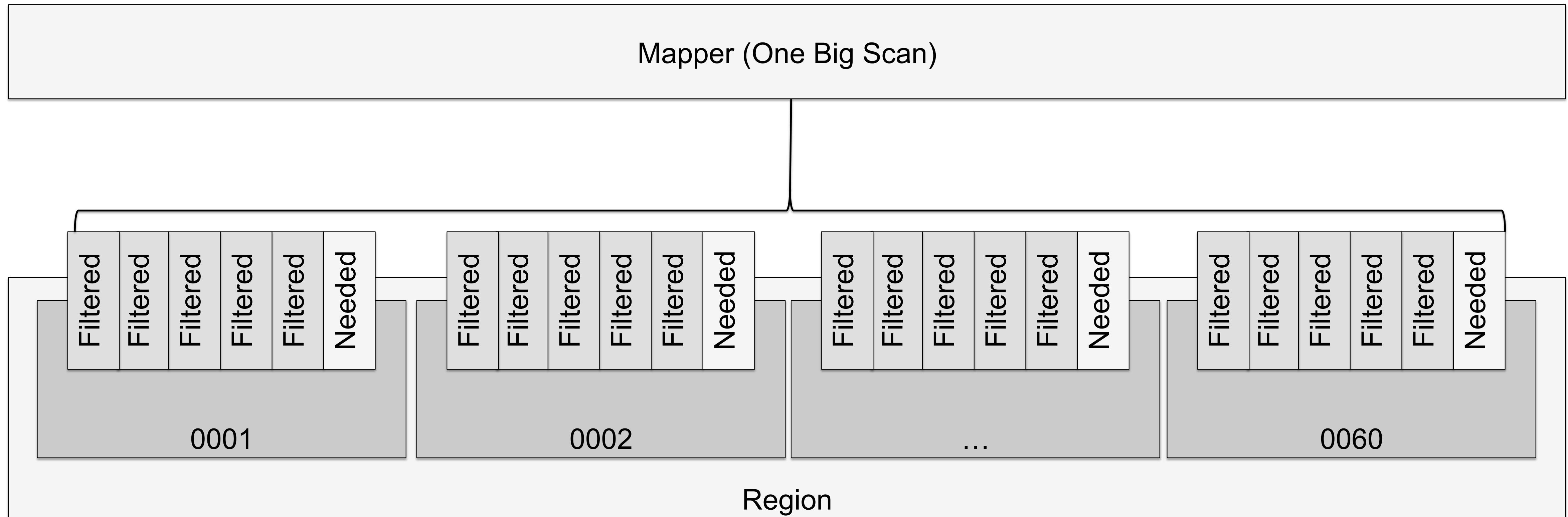
# Scan a Salted Table



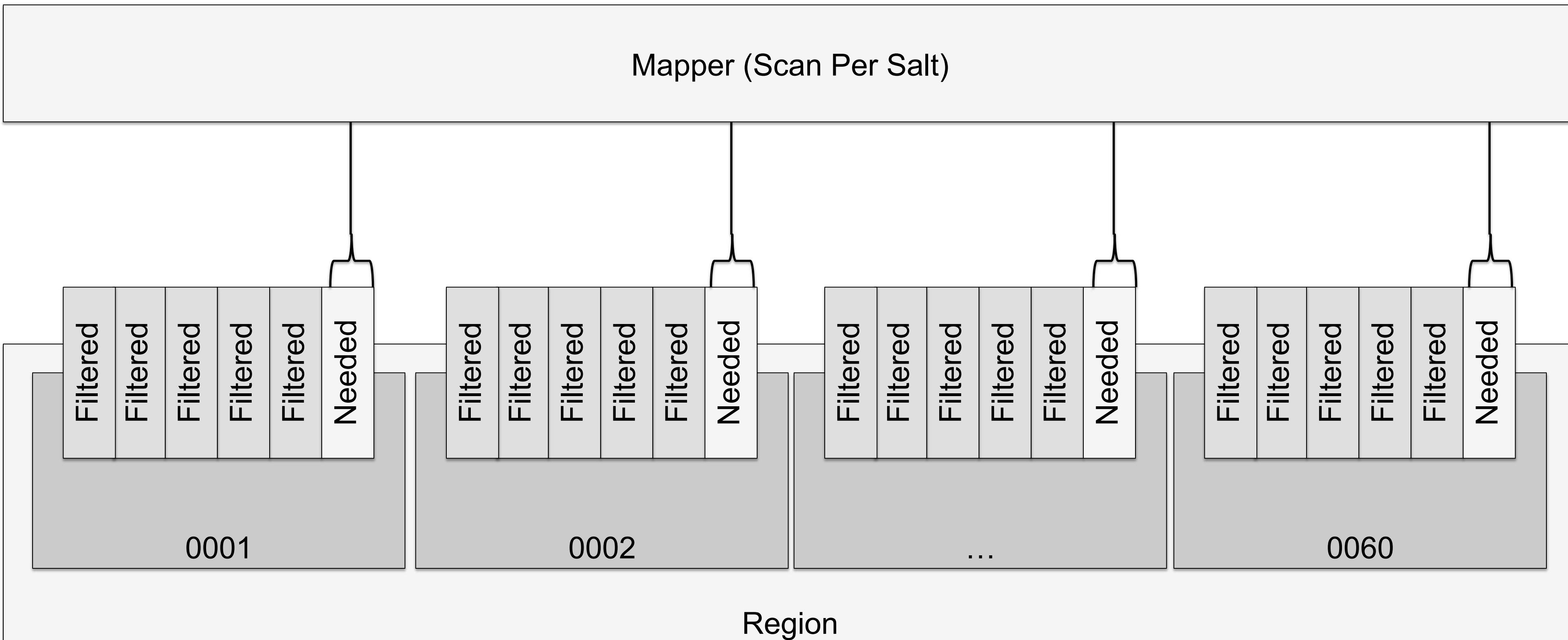
# Scan a Salted Table



# Scan a Salted Table



# Scan a Salted Table



# HBase Data Modeling Considerations

- Columns
  - *Mega Columns*
  - *Single Value Columns*
  - *Millions of Columns*
  - *Increment Values*

# Strata+ Hadoop

---

WORLD

PRESENTED BY

O'REILLY®

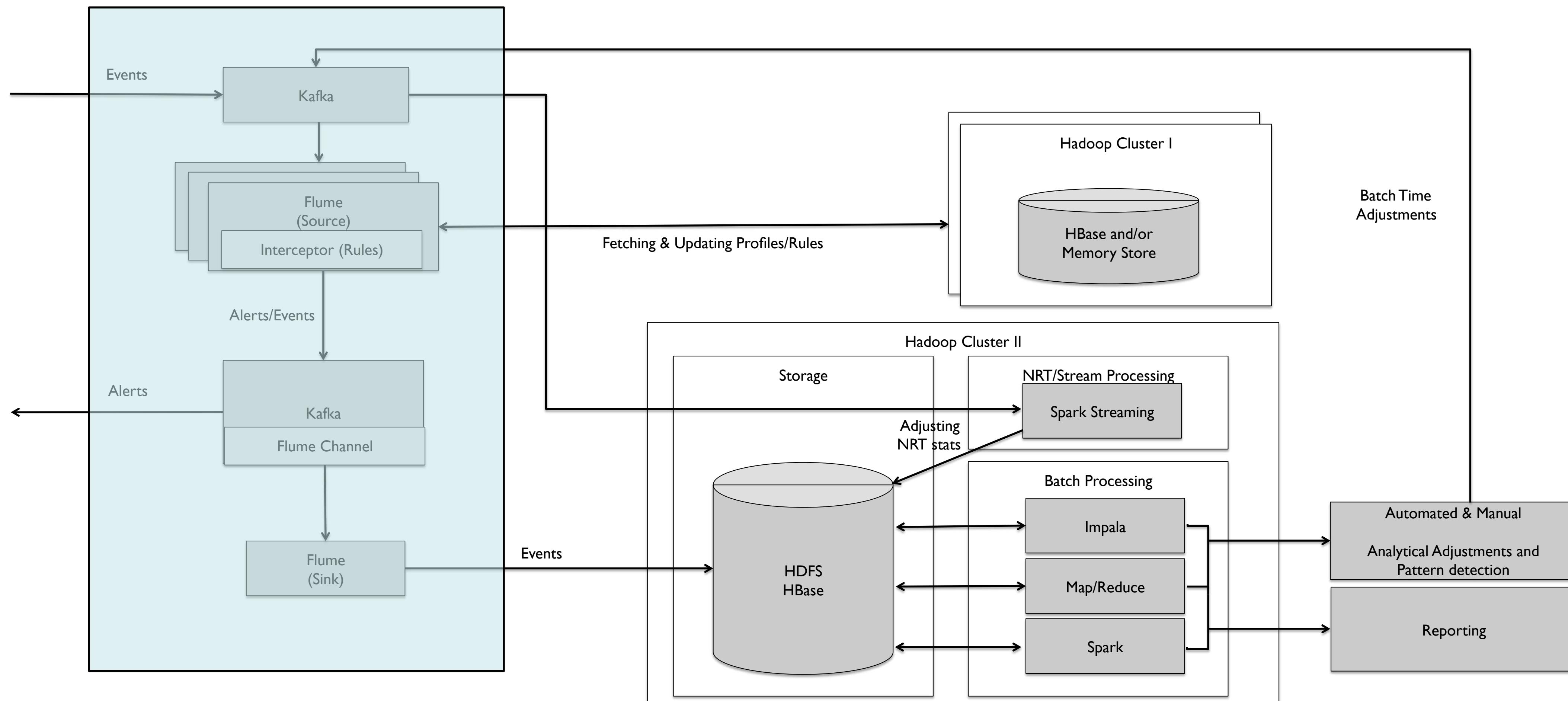
cloudera®

# Ingest and Near Real- Time Alerting

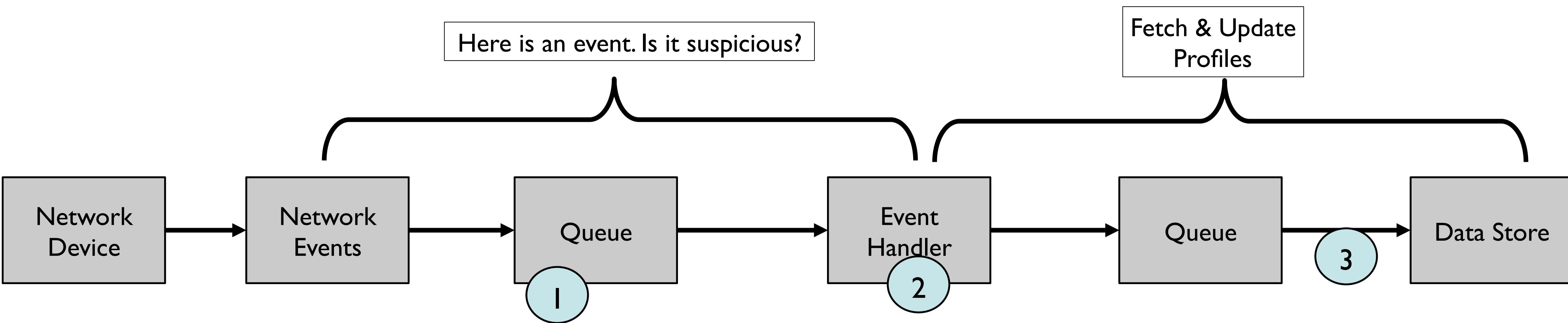
## Considerations

[strataconf.com](http://strataconf.com)

#StrataHadoop



# The basic workflow



# The Queue

- What makes Apache Kafka a good choice?
  - Low latency
  - High throughput
  - Partitioned and replicated
  - Easy to plug to applications

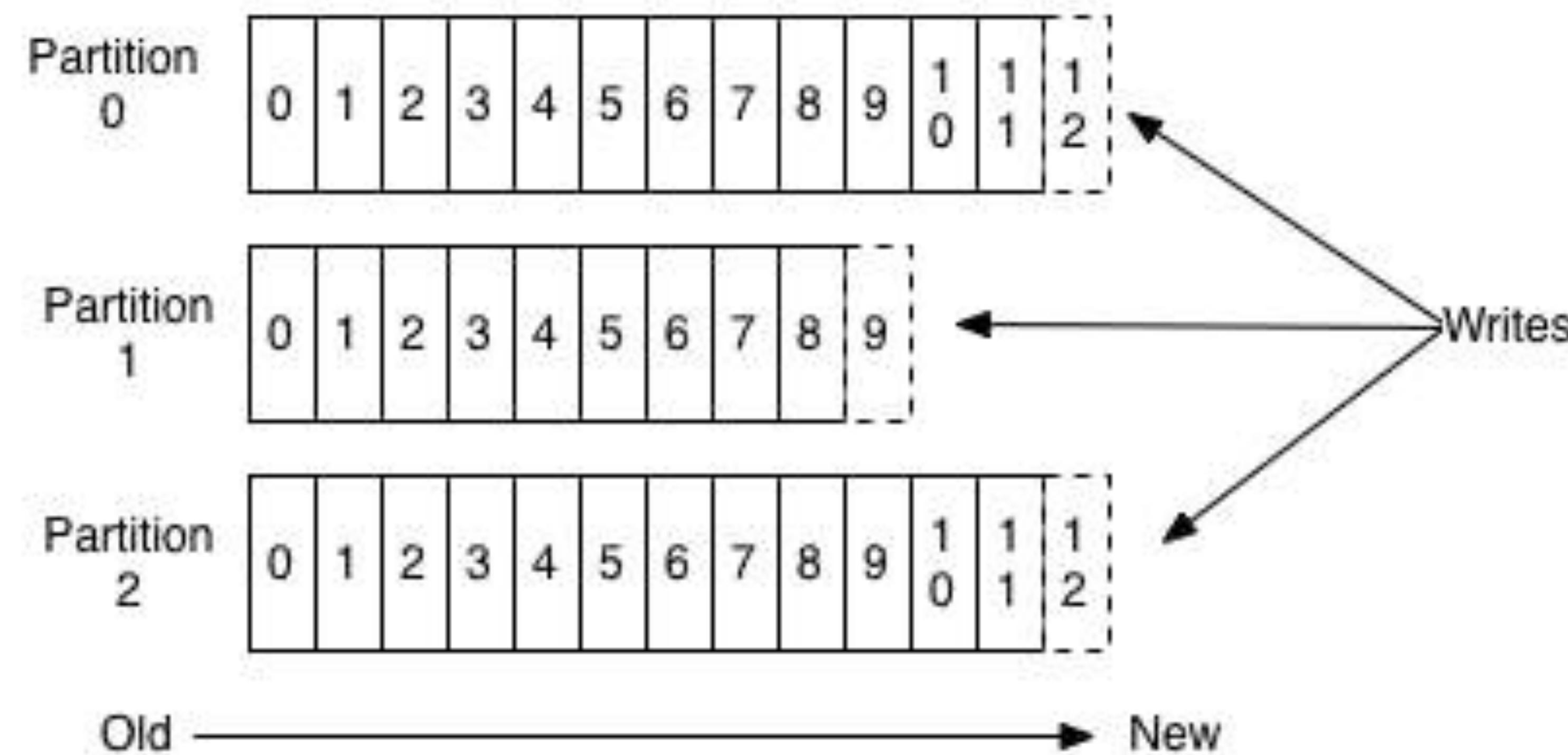


# The Basics

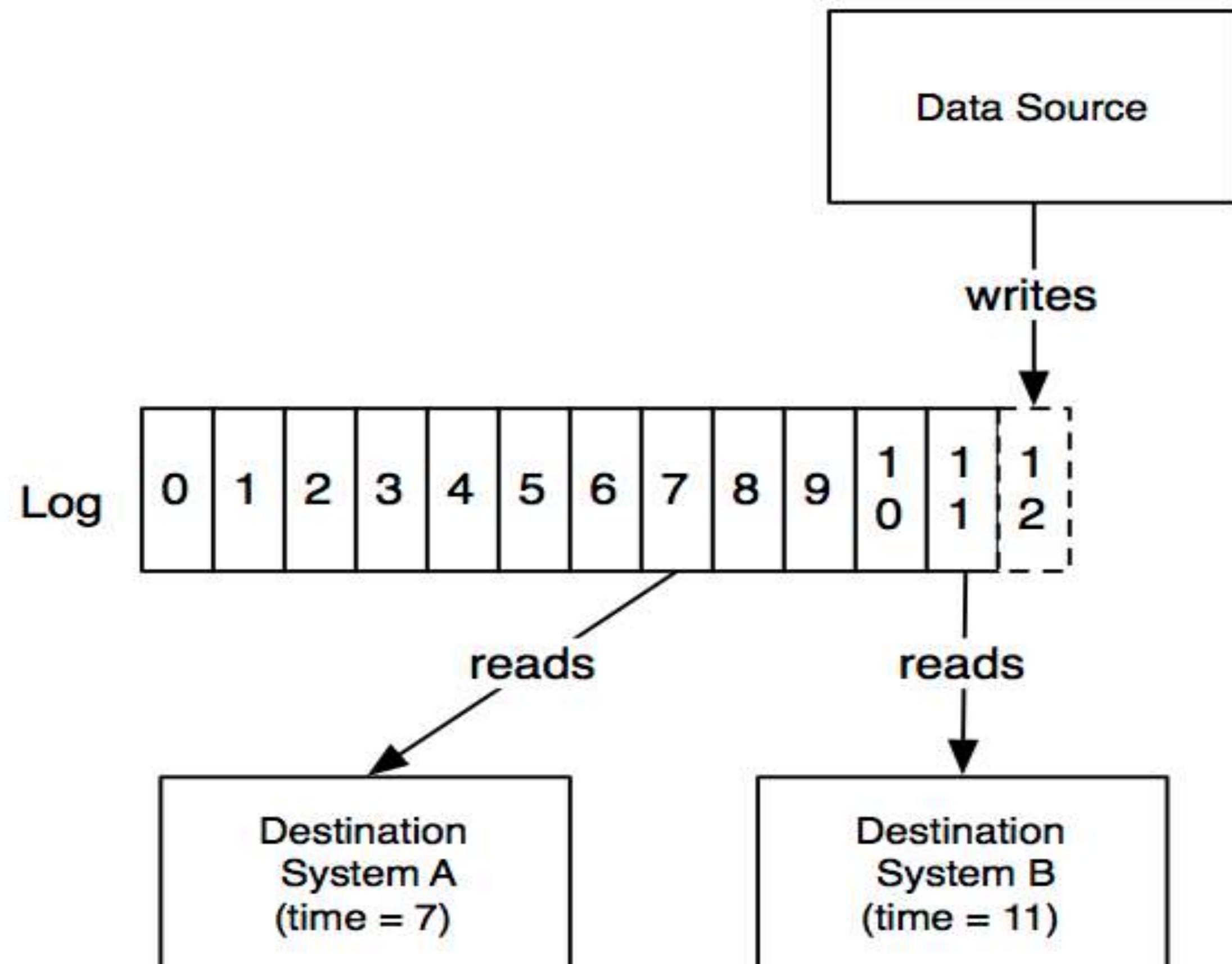
- Messages are organized into **topics**
- **Producers** push messages
- **Consumers** pull messages
- Kafka runs in a cluster. Nodes are called **brokers**

# Topics, Partitions and Logs

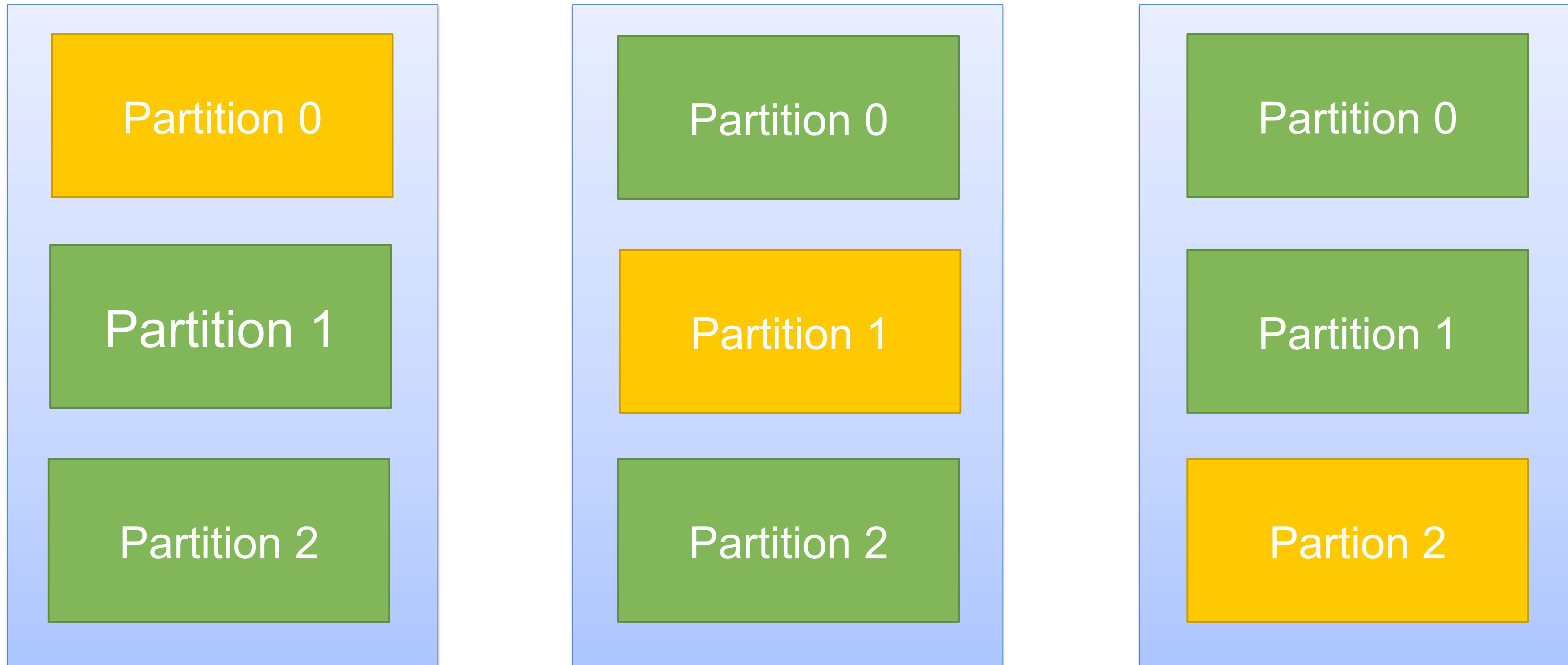
## Anatomy of a Topic



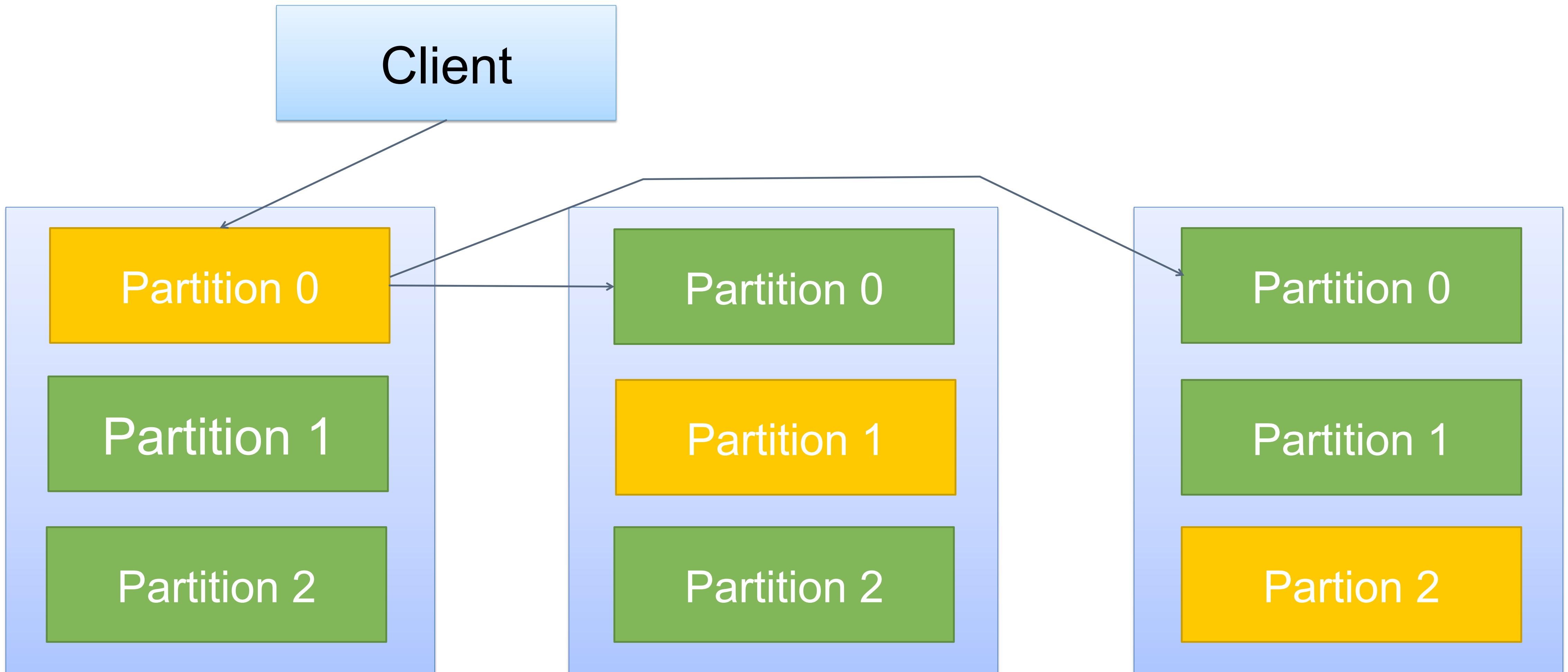
# Each partition is a log



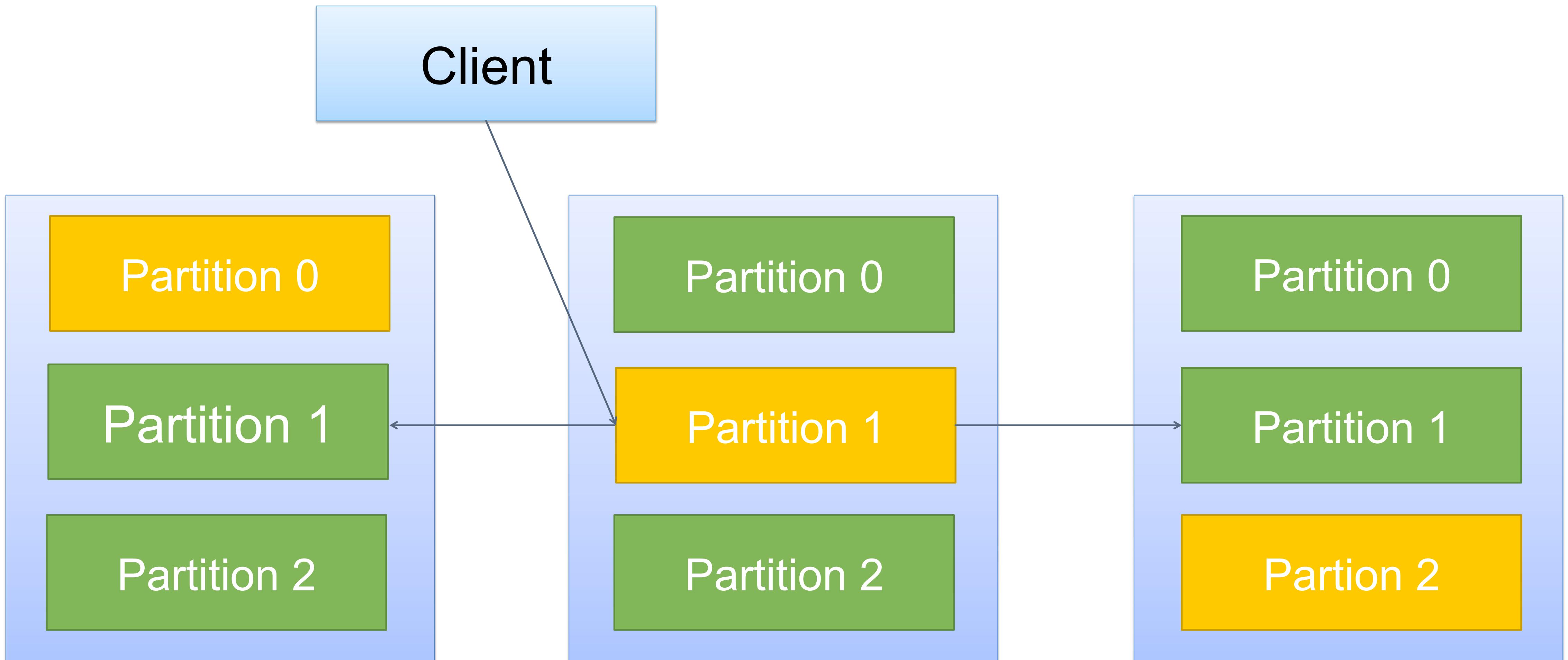
# Each Broker has many partitions



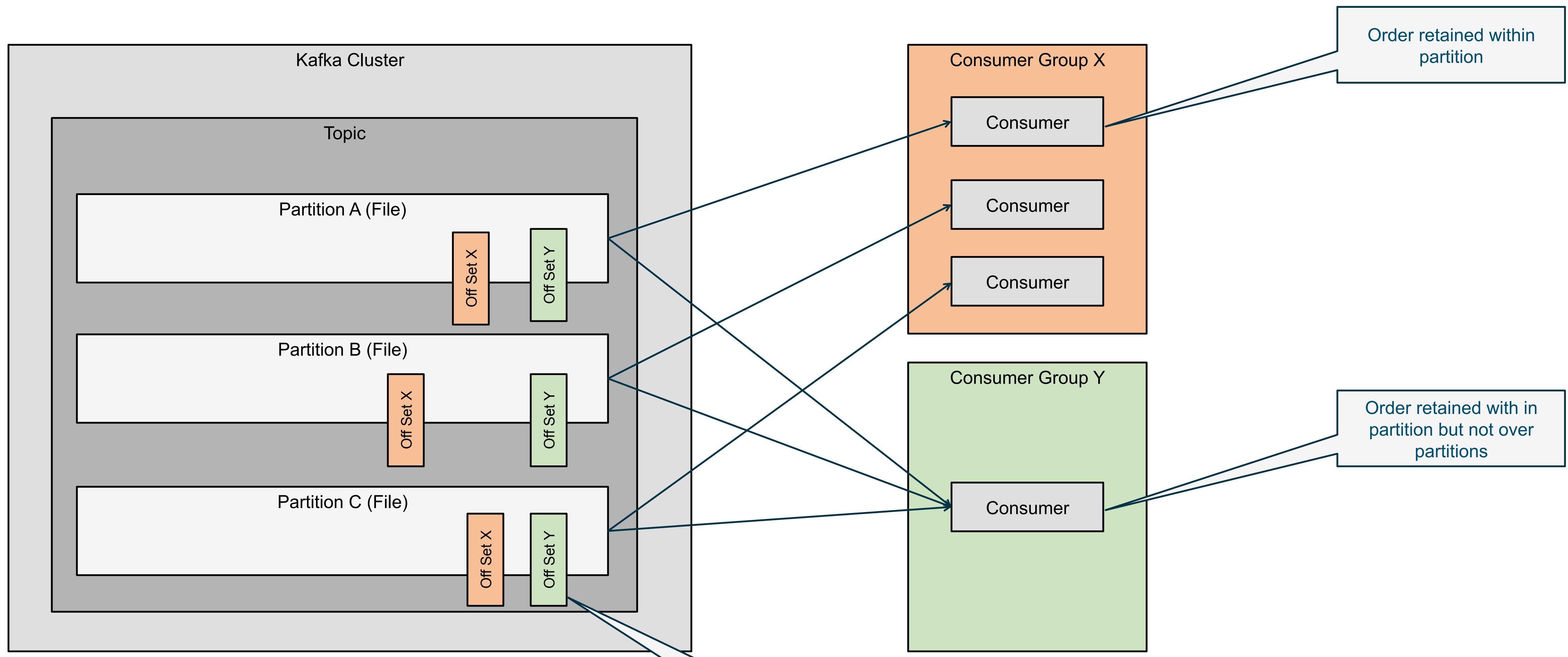
# Producers load balance between partitions



# Producers load balance between partitions

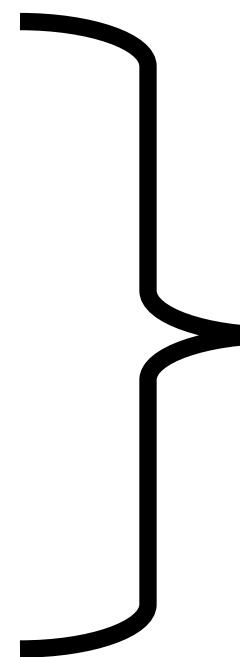


# Consumers



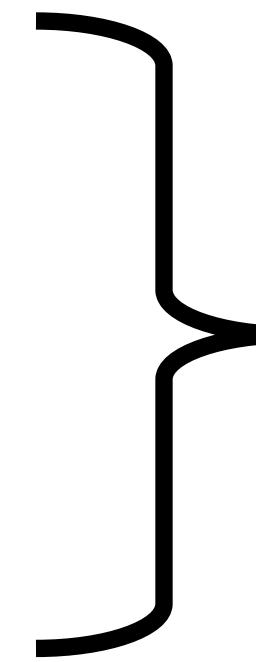
# In our use-case

- Topic for profile updates
- Topic for current events
- Topic for alerts
- Topic for rule updates



Partitioned by device ID

# The event handler

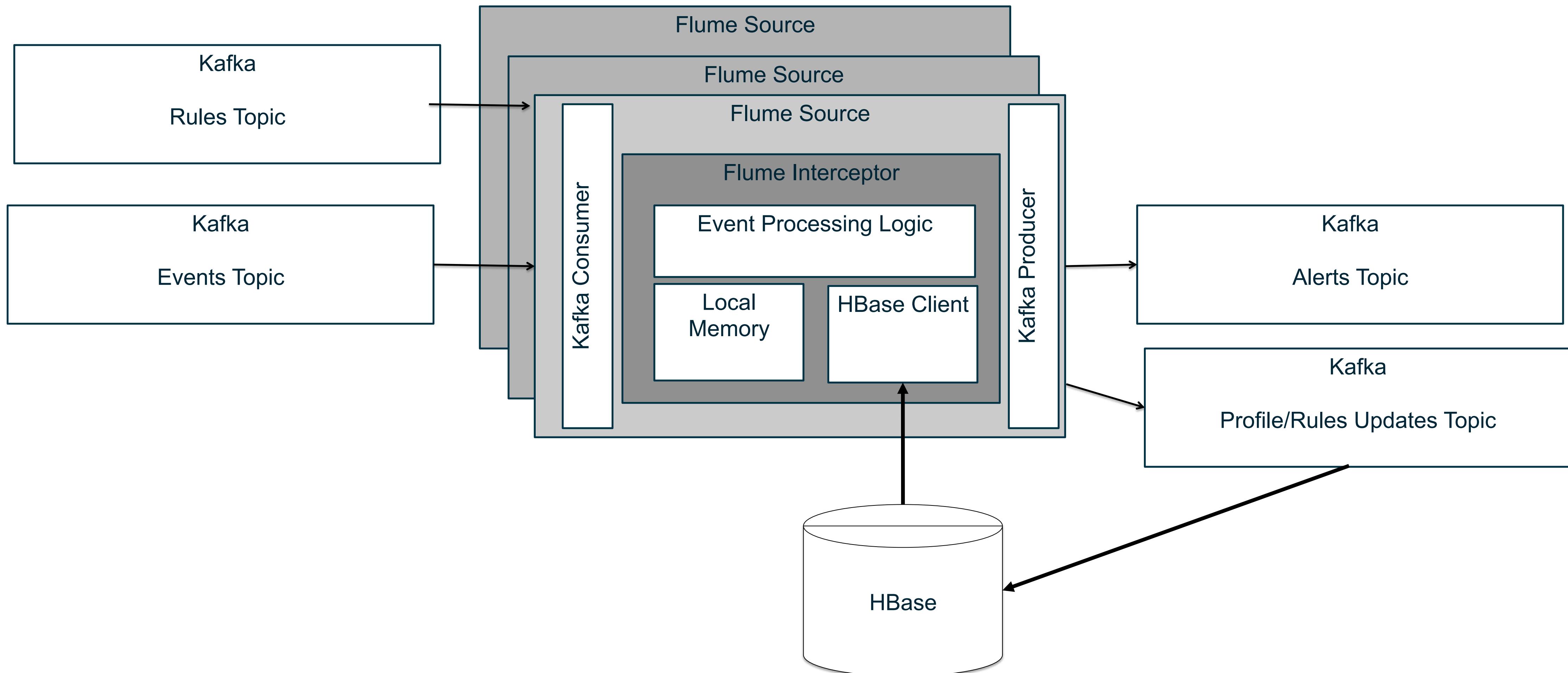
- Very basic pattern:
    - Consume a record
    - “do something to it”
    - Produce zero, one, or more results
- 
- Kafka Processor Pattern

# Easier than you think

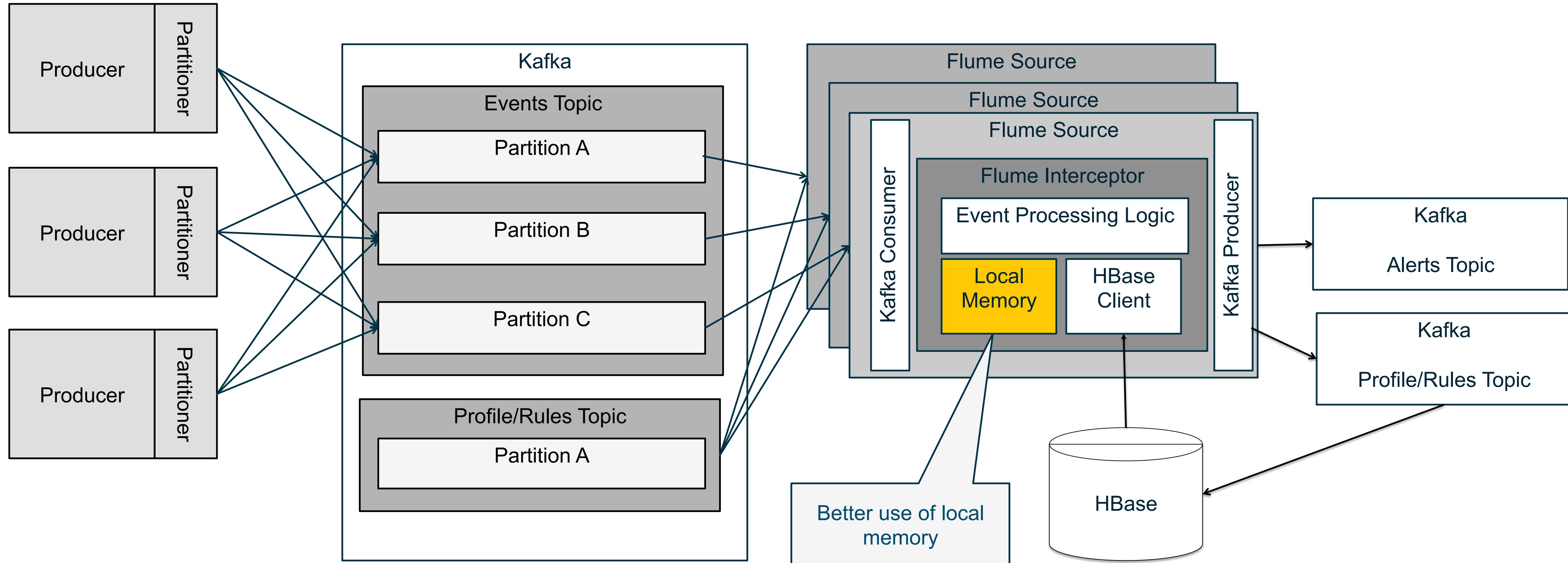
- Kafka provides load balancing and availability
  - Add processes when needed – they will get their share of partitions
  - You control partition assignment strategy
  - If a process crashes, partitions will get automatically reassigned
  - You control when an event is “done”
  - Send events safely and asynchronously
- So you focus on the use-case, not the framework

So easy a PHP developer  
can do it

# Inside The Processor



# Scaling the Processor



# Ingest – Gateway to deep analytics

- Data needs to end up in:
  - SparkStreaming
    - Can read directly from Kafka
  - HBase
  - HDFS



# Considerations for Ingest

- **Async** – nothing can slow down the immediate reaction
  - Always ingest the data from a queue in separate thread or process to avoid write-latency
- **Push vs Pull**
- **Useful end-points** – integrates with variety of data sources and sinks
- Supports **standard formats** and possibly some transformations
  - Data format inside the queue can be different than in the data store
    - For example Avro -> Parquet

# More considerations - Safety

- **Reliable** – data loss is not acceptable
  - Either “at-least-once” or “exactly-once”
  - When is “exactly-once” possible?
- **Error handling** – reasonable reaction to unreasonable data
- **Security** – Authentication, authorization, audit, lineage

# Good Patterns

- Ingest all things. Events, evidence, alerts
  - It has potential value
  - Especially when troubleshooting
  - And for data scientists
- Make sure you take your schema with you
  - Yes, your data probably has a schema
  - Everyone accessing the data (web apps, real time, stream, batch) will need to know about it
  - So make it accessible in a centralized schema registry

# Choosing Ingest

## Flume

Our choice for  
this example

- Part of Hadoop
- Key-value based
- Supports HDFS and HBase
- Easy to configure
- Serializers provide conversions
- At-least once delivery guarantee
- Proven in huge production environments

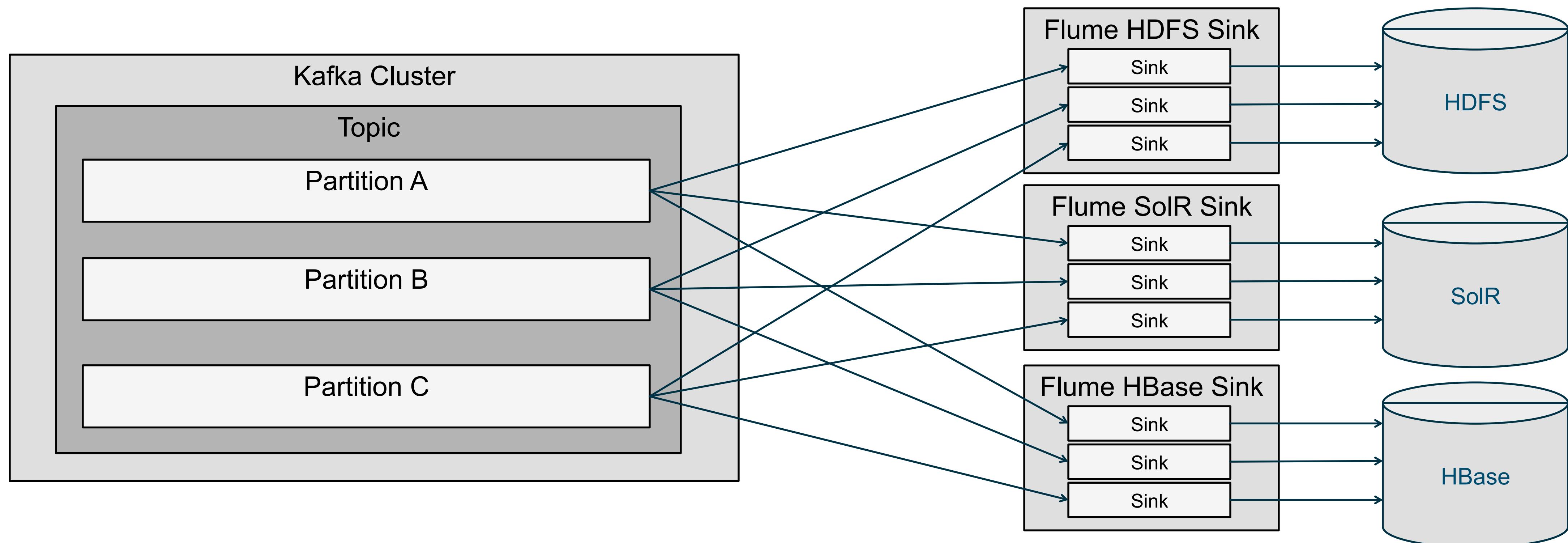
## Kafka Connect

- Part of Apache Kafka
- Schema preserving
- Many connectors
- Well integrated support for Parquet, Avro and JSON
- Exactly-once delivery guarantee
- Easy to manage and scale

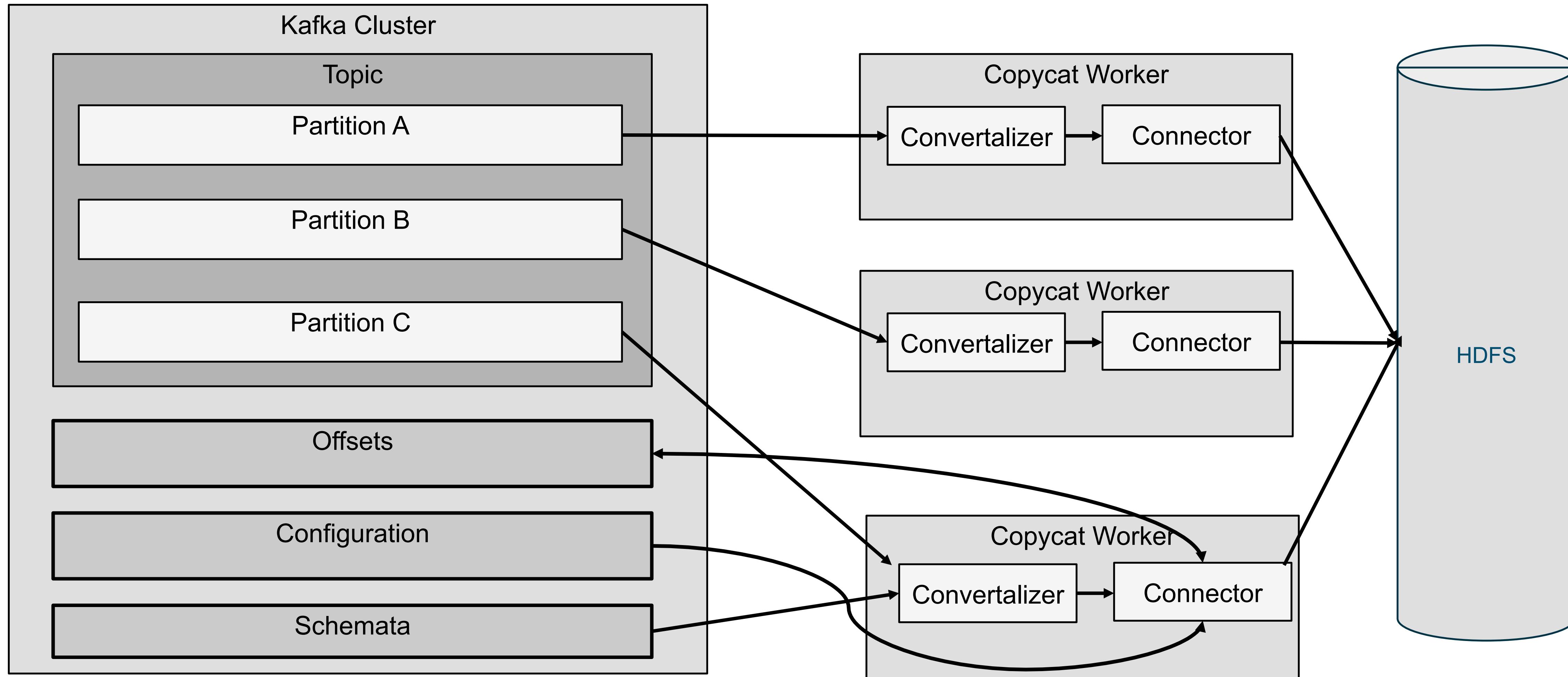
## Spark Streaming

- Part of Hadoop
- Very strong transformation story
- Not as simple as Flume or Connect
- Supports HDFS and HBase
- Smart Kafka Integration
- Exactly-once guarantee from Kafka

# Ingestion - Flume



# Ingestion – Kafka Connect

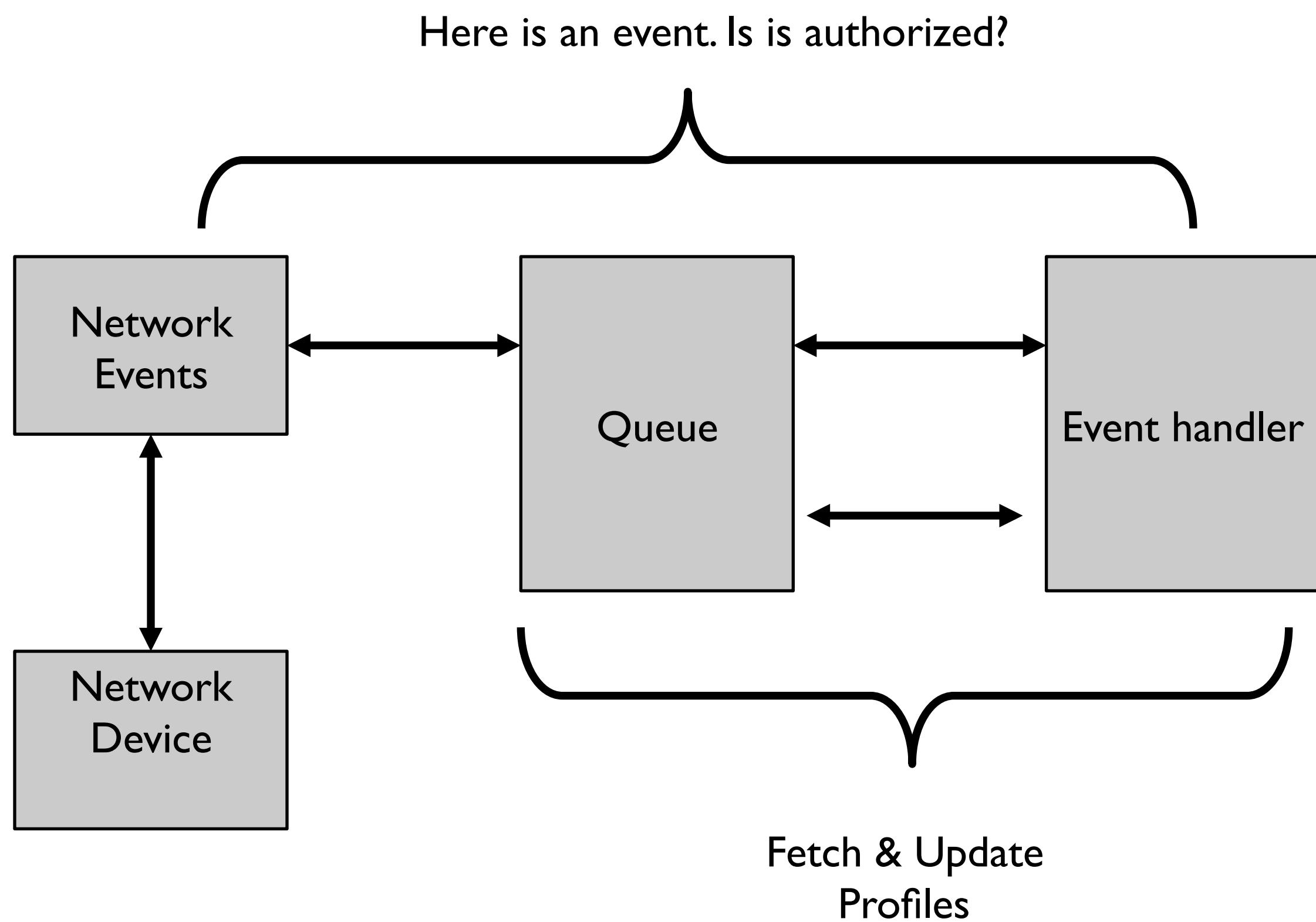


# Alternative Architectures

- What if...

# Do we really need HBase?

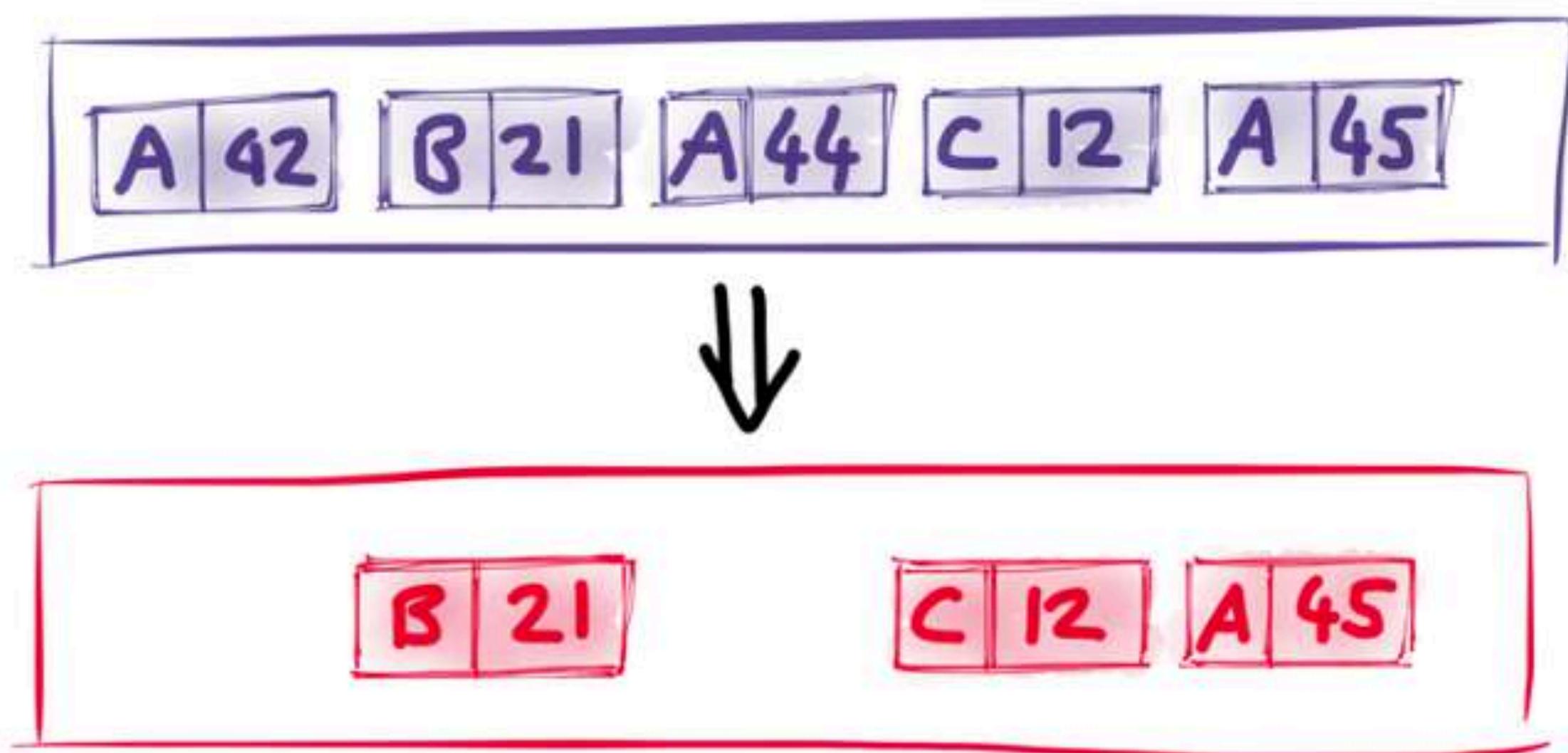
- What if... Kafka would store all information used for real-time processing?



# Log Compaction in Kafka

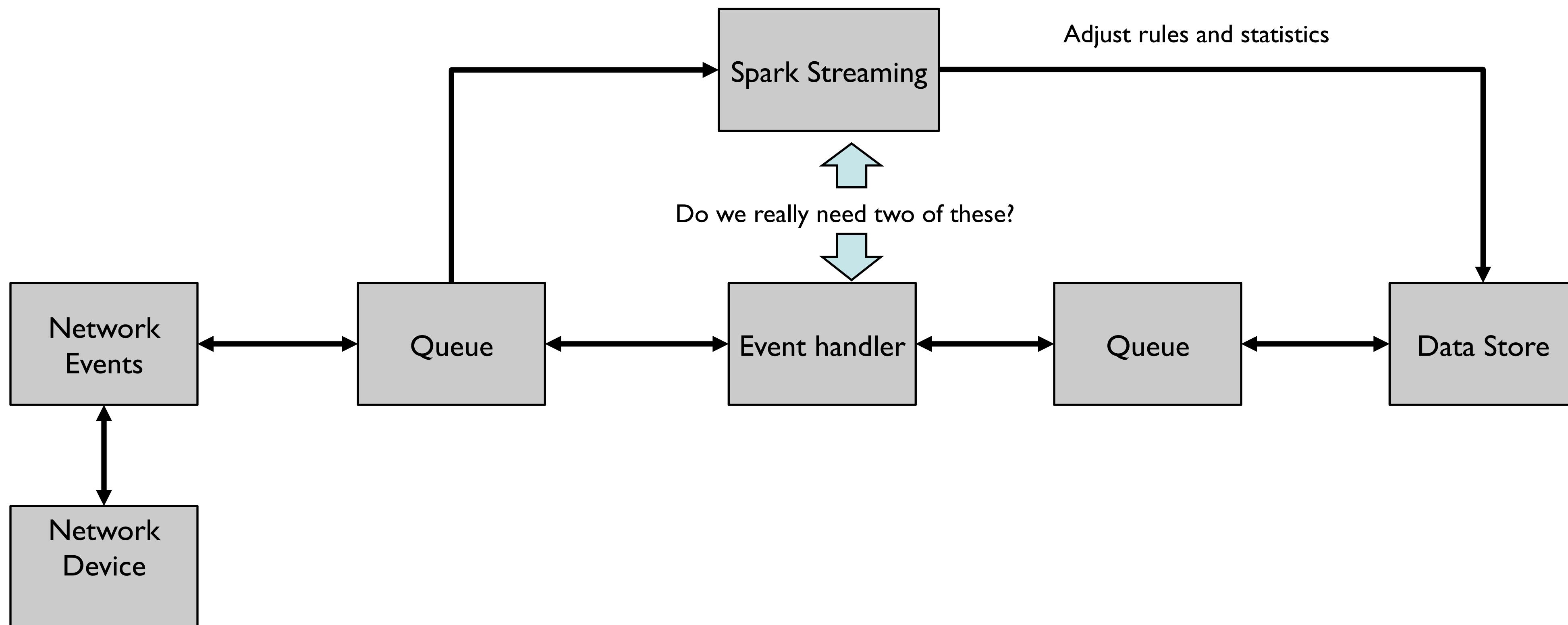
- Kafka stores a “change log” of profiles
- But we want the latest state
- log.cleanup.policy = {delete / compact}

Kafka changelog compaction



# Do we really need Spark Streaming?

- What if... we could do low-latency single event processing and complex stream processing analytics in the same system?



# Is complex processing of single events possible?

- The challenge:
  - Low latency vs high throughput
  - No data loss
  - Windows, aggregations, joins
  - Late data

# Solutions?

- Apache Flink
  - Process each event, checkpoint in batches
  - Local and distributed state, also checkpointer
- Apache Samza
  - State is persisted to Kafka (Change log pattern)
  - Local state cached in RocksDB
- Kafka Streams
  - Similar to Samza
  - But in simple library that is part of Apache Kafka.
  - Late data as updates to state

# You Probably Don't Need Orchestration

#StrataHadoop

Questions? [tiny.cloudera.com/app-arch-questions](http://tiny.cloudera.com/app-arch-questions)

Strata+Hadoop  
WORLD

# Strata+ Hadoop

---

WORLD

PRESENTED BY

O'REILLY®

cloudera®

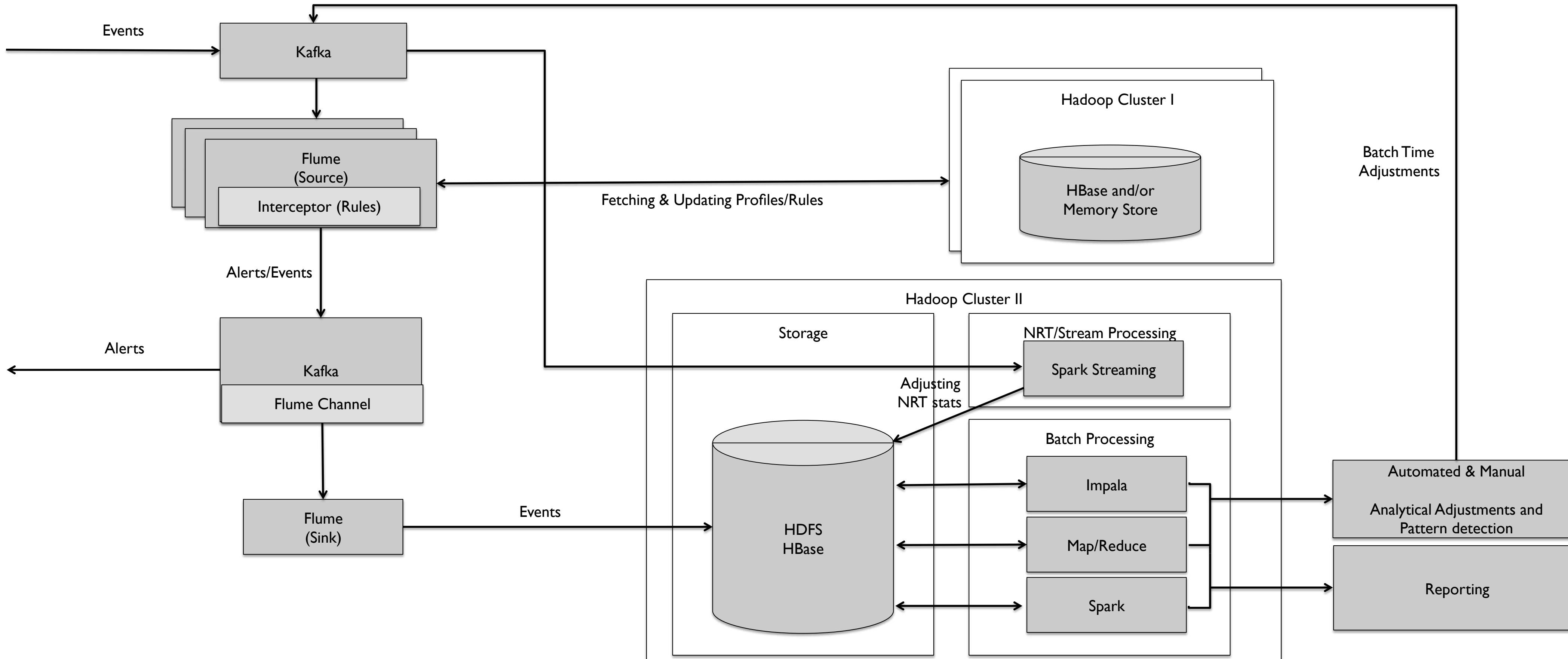
# Processing

## Considerations

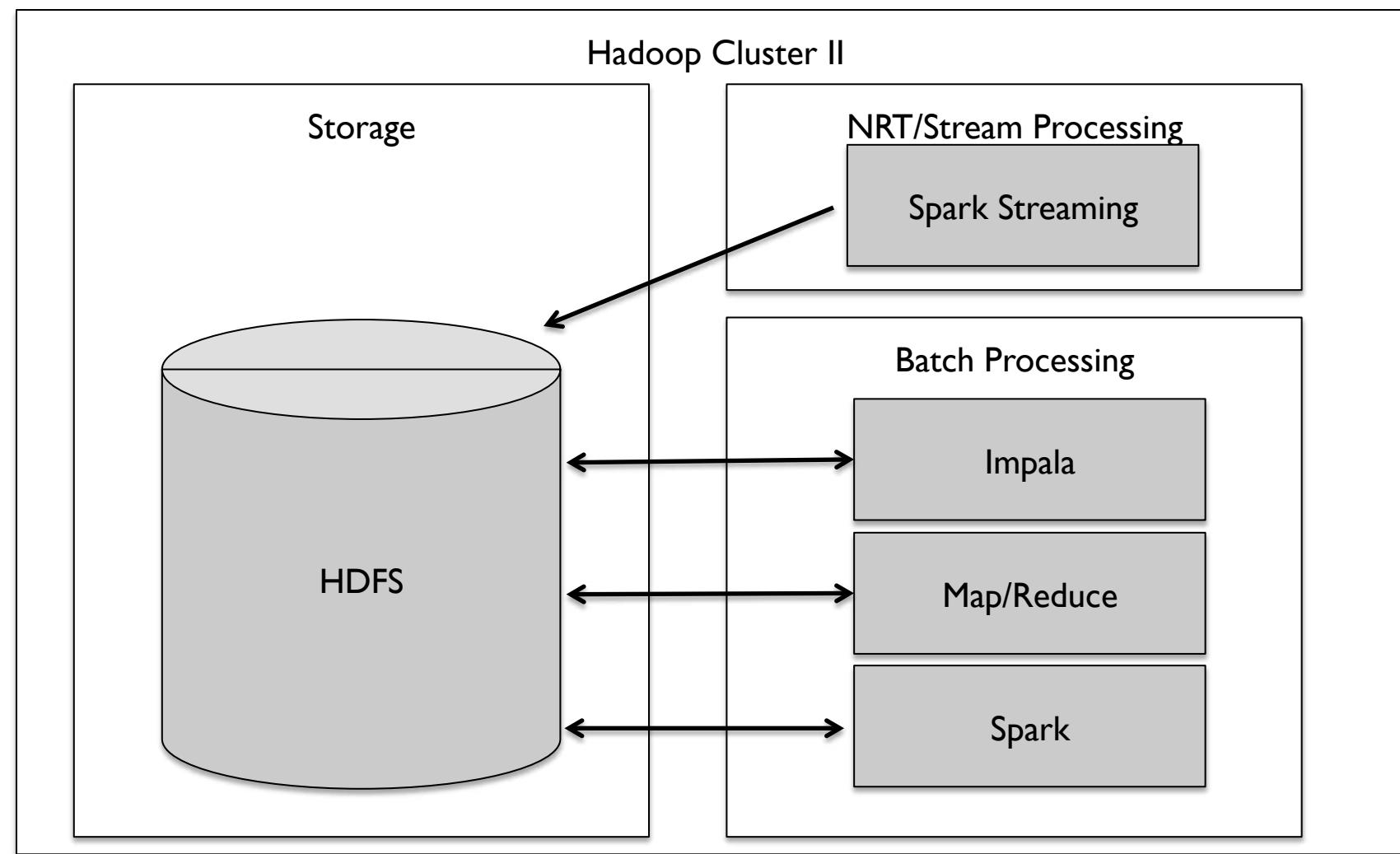
[strataconf.com](http://strataconf.com)

#StrataHadoop

# Reminder: Full Architecture

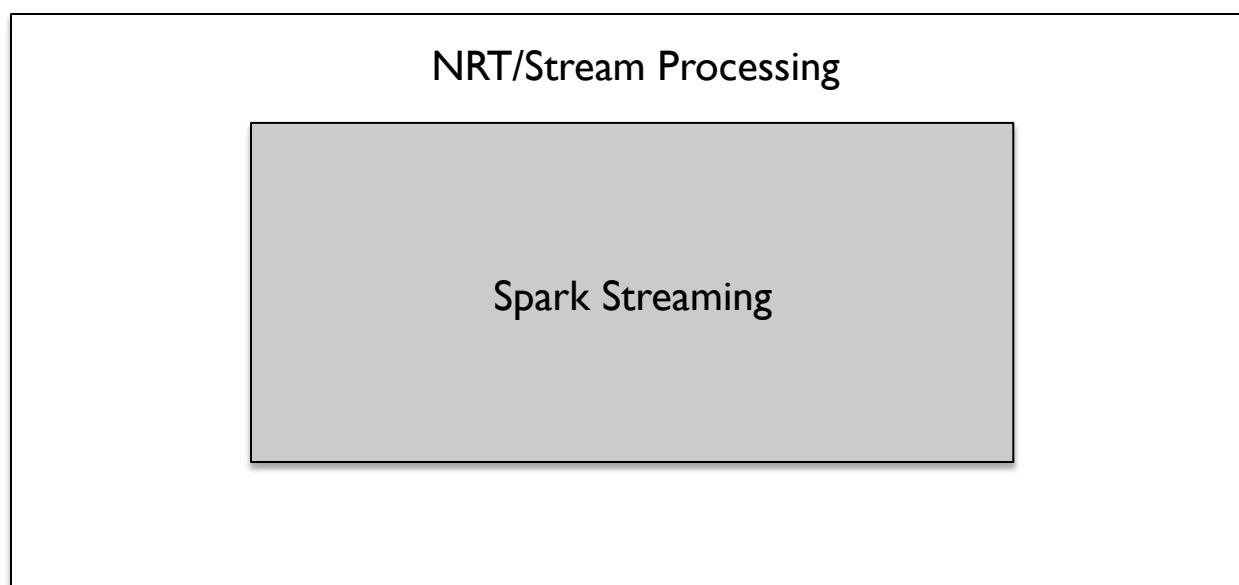


# Processing

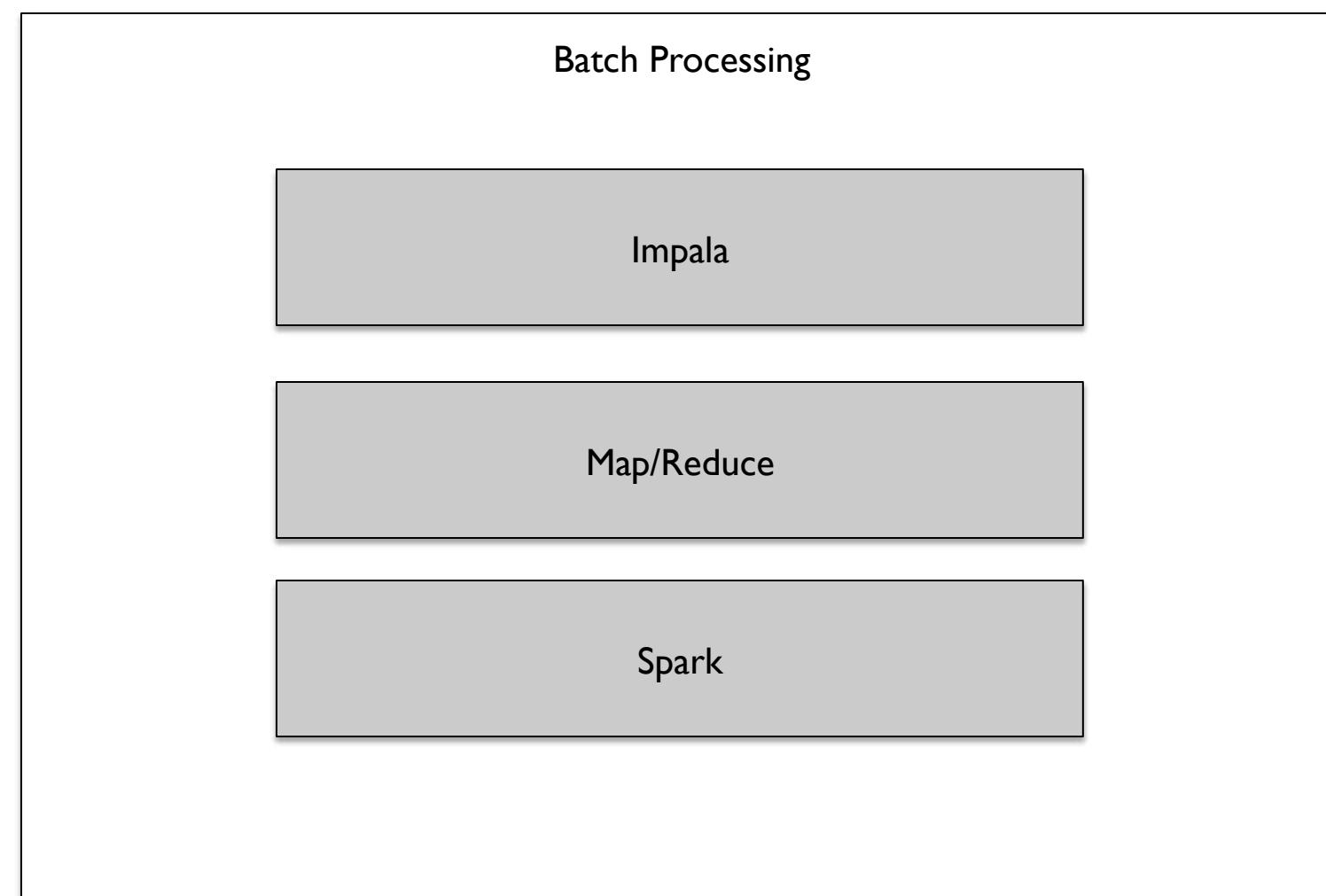


# Two Kinds of Processing

## Streaming



## Batch



# Strata+ Hadoop

---

WORLD

PRES EN TED BY

O'REILLY®

cloudera®

# Stream Processing Considerations

[strataconf.com](http://strataconf.com)

#StrataHadoop

# Stream Processing Options

1. Storm
2. Spark Streaming
3. KafkaStreams
4. Flink

# Why Spark Streaming?

- There's one engine to know.
- Micro-batching helps you scale reliably.
- Exactly once counting
- Hadoop ecosystem integration is baked in.
- No risk of data loss.
- It's easy to debug and run.
- State management
- Huge eco-system backing
- You have access to ML libraries.
- You can use SQL where needed.
- Wide-spread use and hardened

# Spark Streaming Example

1. val conf = new SparkConf().setMaster("local[2]")
2. val ssc = new StreamingContext(conf, Seconds(1))
3. val lines = ssc.socketTextStream("localhost", 9999)
4. val words = lines.flatMap(\_.split(" "))
5. val pairs = words.map(word => (word, 1))
6. val wordCounts = pairs.reduceByKey(\_ + \_)
7. wordCounts.print()
8. SSC.start()

# Spark Streaming Example

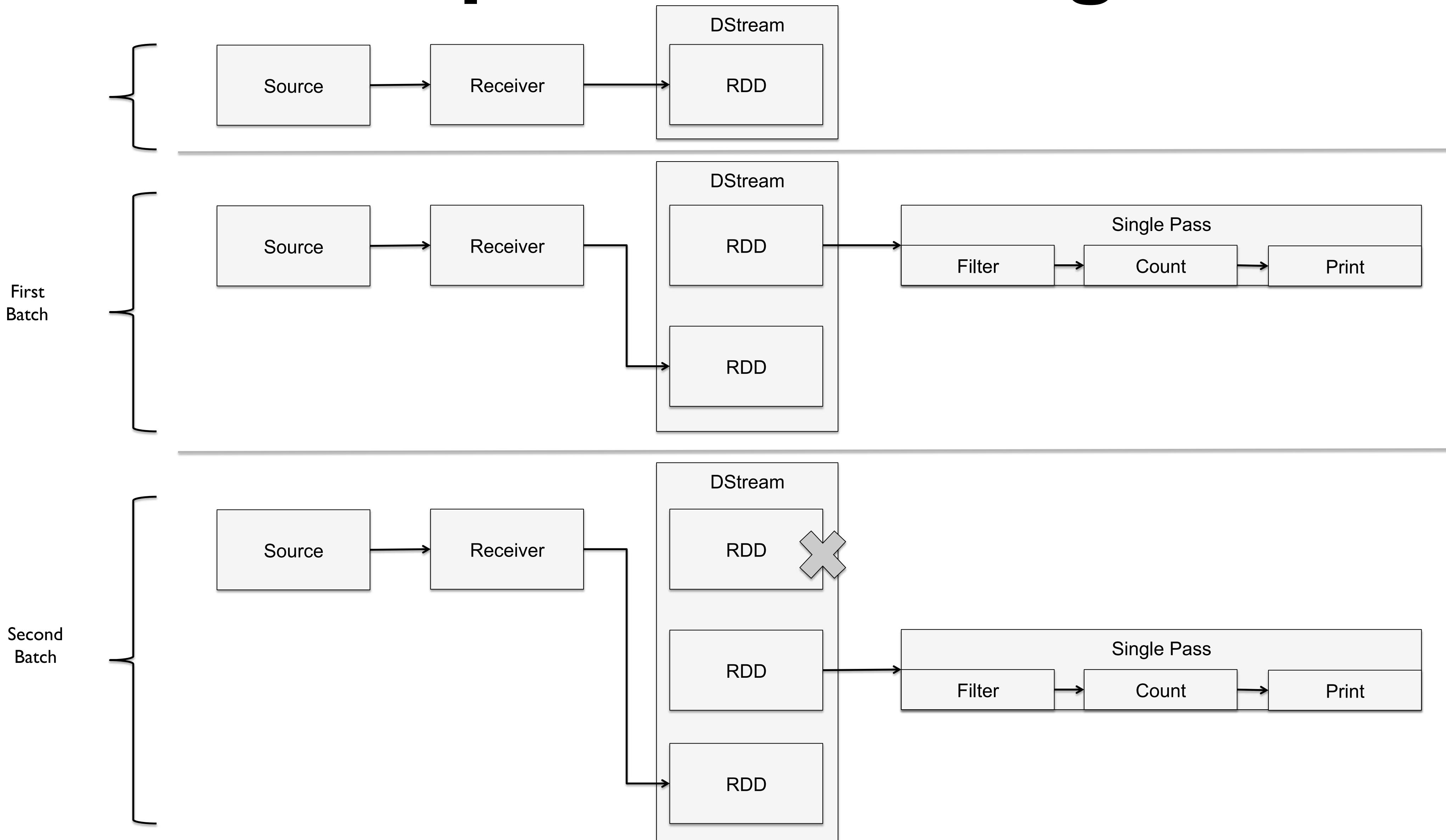
1. val conf = new SparkConf().setMaster("local[2]")
2. val sc = new SparkContext(conf)
3. val lines = sc.textFile(path, 2)
4. val words = lines.flatMap(\_.split(" "))
5. val pairs = words.map(word => (word, 1))
6. val wordCounts = pairs.reduceByKey(\_ + \_)
7. wordCounts.print()

# Spark SQL and Spark Streaming

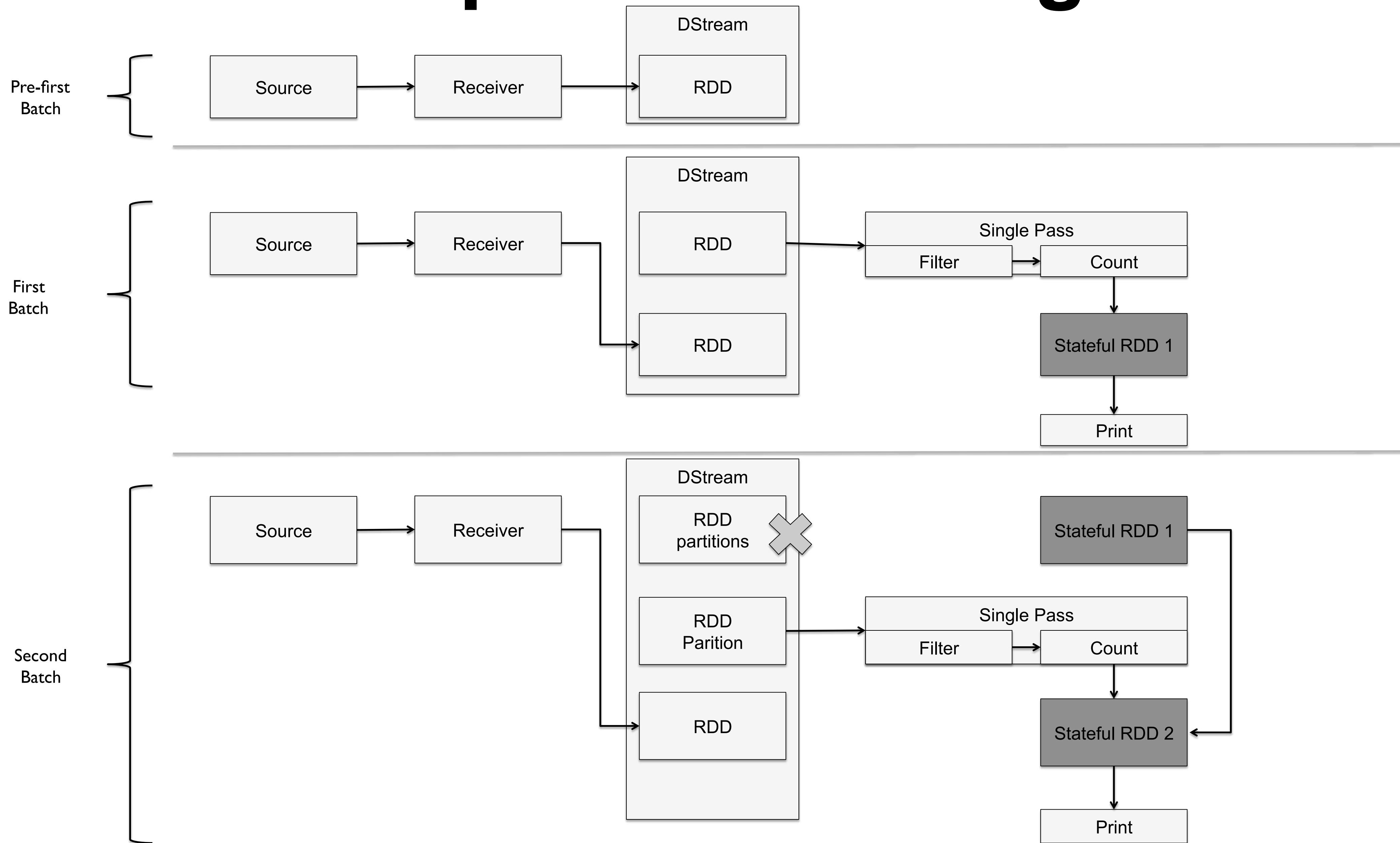
---

- Catalyst
  - Spark SQL's optimizer
- Bringing Catalyst and Spark SQL integration to streaming

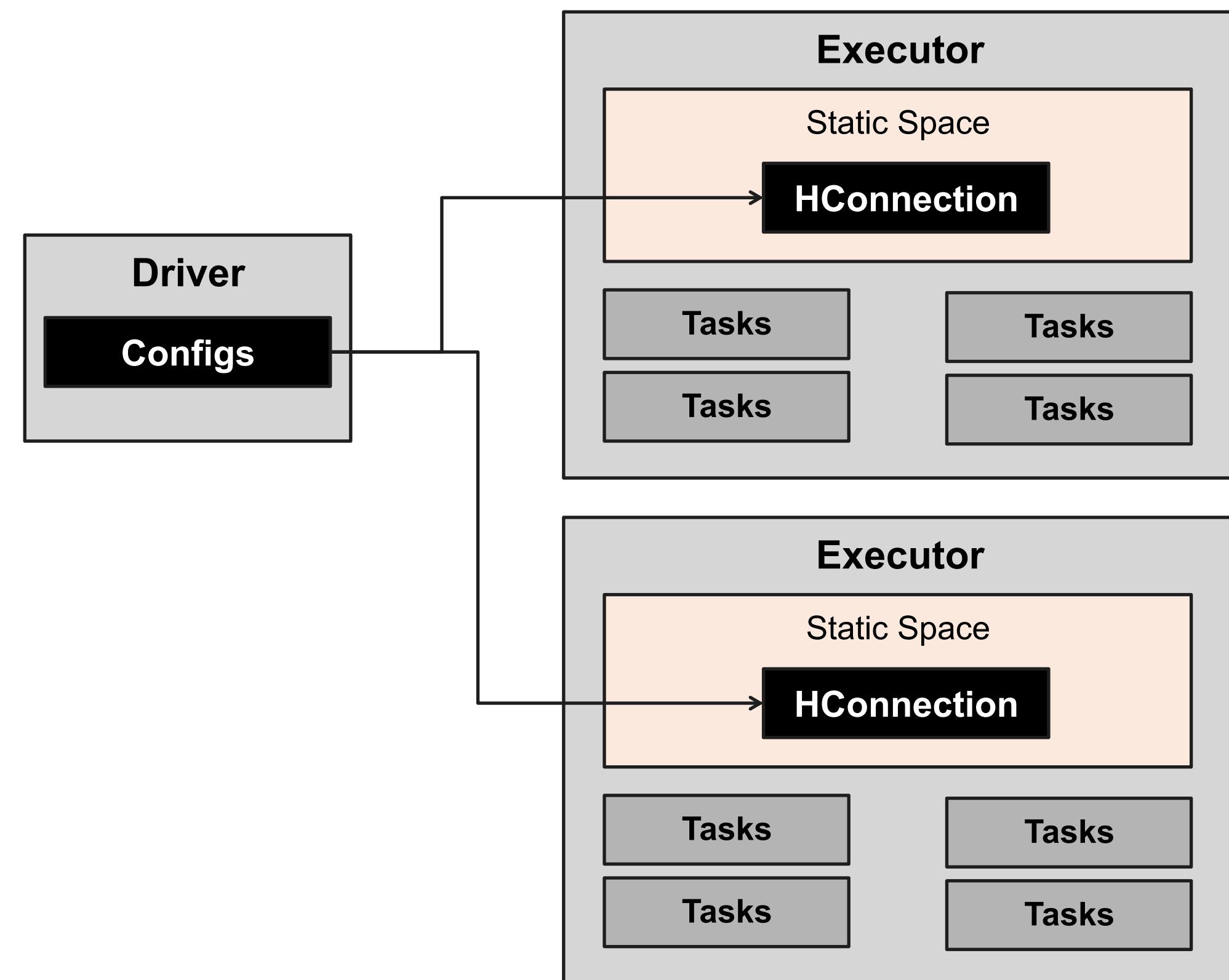
# Spark Streaming



# Spark Streaming



# Spark Streaming and HBase



# HBase-Spark Module

- BulkPut
- BulkDelete
- BulkGet
- BulkLoad (Wide and Tail)
- ForeachPartition(`It, Conn`)
- MapPartition(`it, Conn`)
- Spark SQL

# HBase-Spark Module

```
val hbaseContext = new HBaseContext(sc, config);

hbaseContext.bulkDelete[Array[Byte]](rdd,
    tableName,
    putRecord => new Delete(putRecord),
    4);
```

```
val hbaseContext = new HBaseContext(sc, config)

rdd.hbaseBulkDelete(hbaseContext,
    tableName,
    putRecord => new Delete(putRecord),
    4)
```

# HBase-Spark Module

```
val hbaseContext = new HBaseContext(sc, config)  
rdd.hbaseBulkDelete(tableName)
```

# HBase-Spark Module

```
val hbaseContext = new HBaseContext(sc, config)

rdd.hbaseForeachPartition(hbaseContext, (it, conn) => {

    val bufferedMutator = conn.getBufferedMutator(TableName.valueOf("t1"))

    ...

    bufferedMutator.flush()

    bufferedMutator.close()

})

val getRdd = rdd.hbaseMapPartitions(hbaseContext, (it, conn) => {

    val table = conn.getTable(TableName.valueOf("t1"))

    var res = mutable.MutableList[String]()

    ...

    }

)
```

# HBase-Spark Module

```
rdd.hbaseBulkLoad (tableName,  
                    t => {  
                      Seq( new KeyFamilyQualifier(t.rowKey, t.family,  
                        t.qualifier), t.value)).  
                      iterator  
                    },  
                    stagingFolder)
```

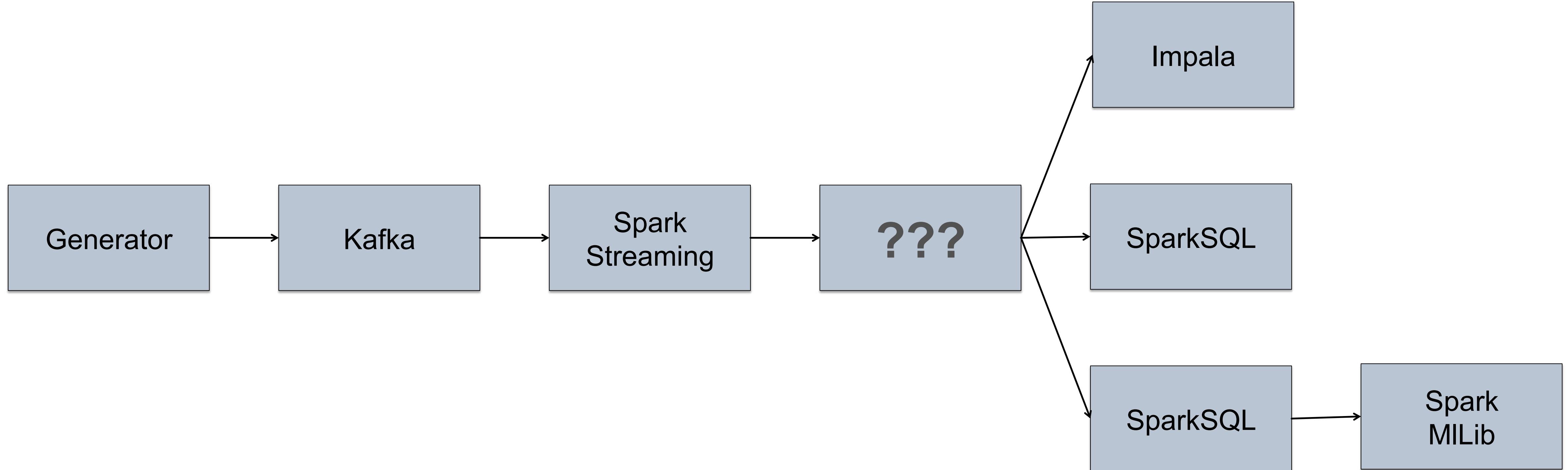
# HBase-Spark Module

```
val df = sqlContext.load("org.apache.hadoop.hbase.spark",
    Map("hbase.columns.mapping" -> "KEY_FIELD STRING :key, A_FIELD STRING c:a,
B_FIELD STRING c:b,",
    "hbase.table" -> "t1"))

df.registerTempTable("hbaseTmp")

sqlContext.sql("SELECT KEY_FIELD FROM hbaseTmp " +
    "WHERE " +
    "(KEY_FIELD = 'get1' and B_FIELD < '3') or " +
    "(KEY_FIELD <= 'get3' and B_FIELD = '8')).foreach(r => println(" - " + r))
```

# Demo Architecture





# KafkaStreams

---

- *Event-at-a-time processing*
- *State-full processing*
- *Windowing with out-of-order data*
- *Auto scaling / fault tolerant / reprocessing / etc*

# KafkaStreams Example

```
public static void main(String[] args) {  
    // specify the processing topology by first reading in a stream from a topic  
    Kstream<String, String> words = builder.stream("topic1");  
  
    // count the words in this stream as an aggregated table  
    Ktable<String, Long> counts = words.countByKey("Counts");  
  
    // write the result table to a new topic  
    counts.to("topic2")  
  
    // create stream processing instance and run in  
    KafkaStreams streams = new KafkaStreams(builder, config);  
    streams.start();  
}
```

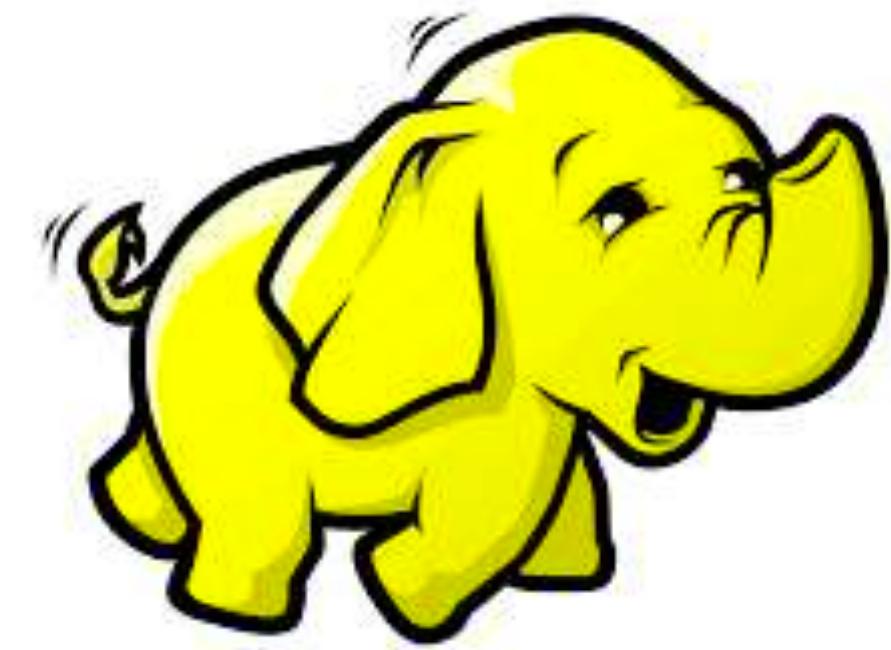
# Key KafkaStreams Ideas

---

- Operators (join, aggregate, source, sink) are nodes in DAG
- Stateful operators are persisted to Kafka – process states are a STREAM
- Data parallelism via topic partitions
- STREAM can be a changelog of a TABLE
- TABLE is a snapshot of a STREAM
- Flow is based on event time (when the event was created)

# New Hadoop Storage Option

Use case break up



## Unstructured Data

Deep Storage  
Scan Use Cases

## Structured Data

SQL + Scan Use Cases



## Fixed Columns Schemas

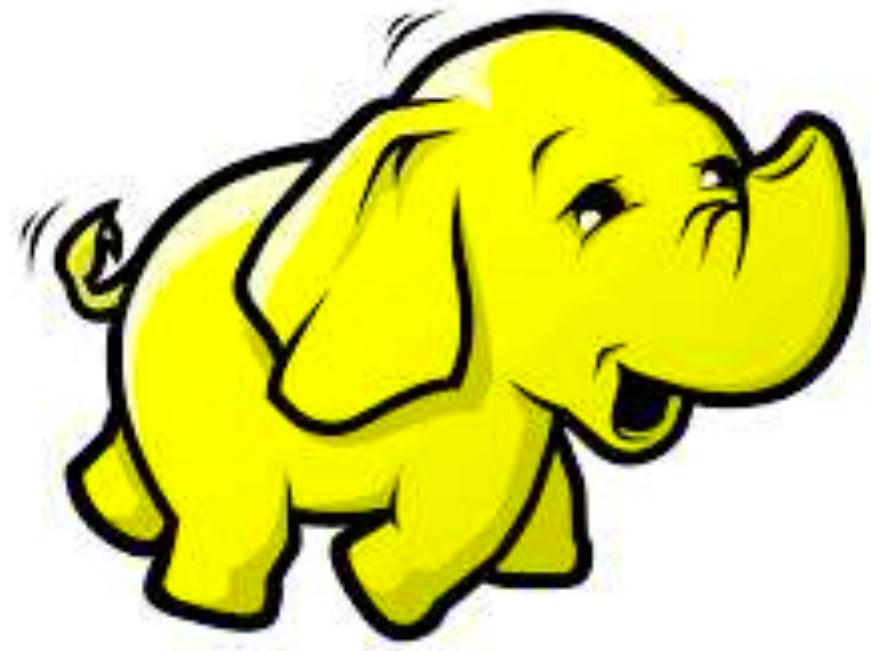
SQL + Scan Use Cases

## Any Type of Column Schemas

Gets / Puts / Micro Scans

# New Hadoop Storage Option

Use case break up



## Unstructured Data

Deep Storage  
Scan Use Cases

## Structured Data

SQL + Scan Use Cases

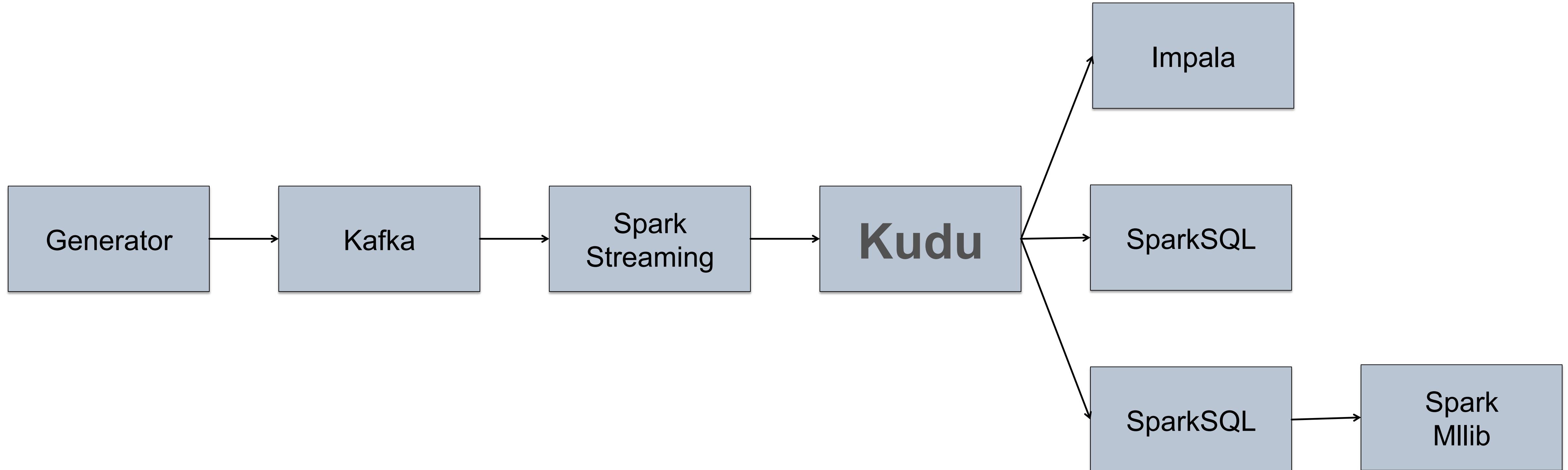
## Fixed Columns Schemas

SQL + Scan Use Cases

## Any Type of Column Schemas

Gets / Puts / Micro Scans

# Real-Time Analytics with Kudu



# Kudu and Spark

<https://github.com/tmalaska/SparkOnKudu>

Same Development From HBase-Spark is now on Kudu and Spark

- Full Spark RDD integration
- Full Spark Streaming integration
- Initial SparkSQL integration

# Strata+ Hadoop

---

WORLD

PRESENTED BY

O'REILLY®

cloudera®

# Processing

## Batch

[strataconf.com](http://strataconf.com)

#StrataHadoop

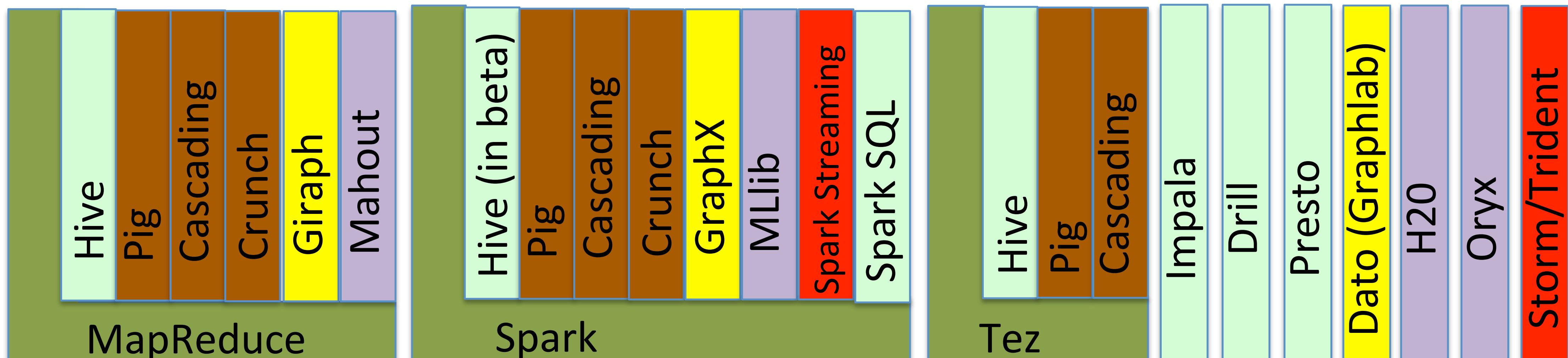
# Why batch in a NRT use case?

- For background processing
  - To be later used for quick decision making
- Exploratory analysis
  - Machine Learning
- Automated rule changes
  - Based on new patterns
- Analytics
  - Who's attacking us?

# Processing Engines (Past)

- Hadoop = HDFS (distributed FS) + MapReduce (Processing engine)

# Processing Engines (present)



General purpose execution engines

Abstraction engines

Graph processing engines

Storage managers

SQL engines

Real-time frameworks

Machine Learning engines

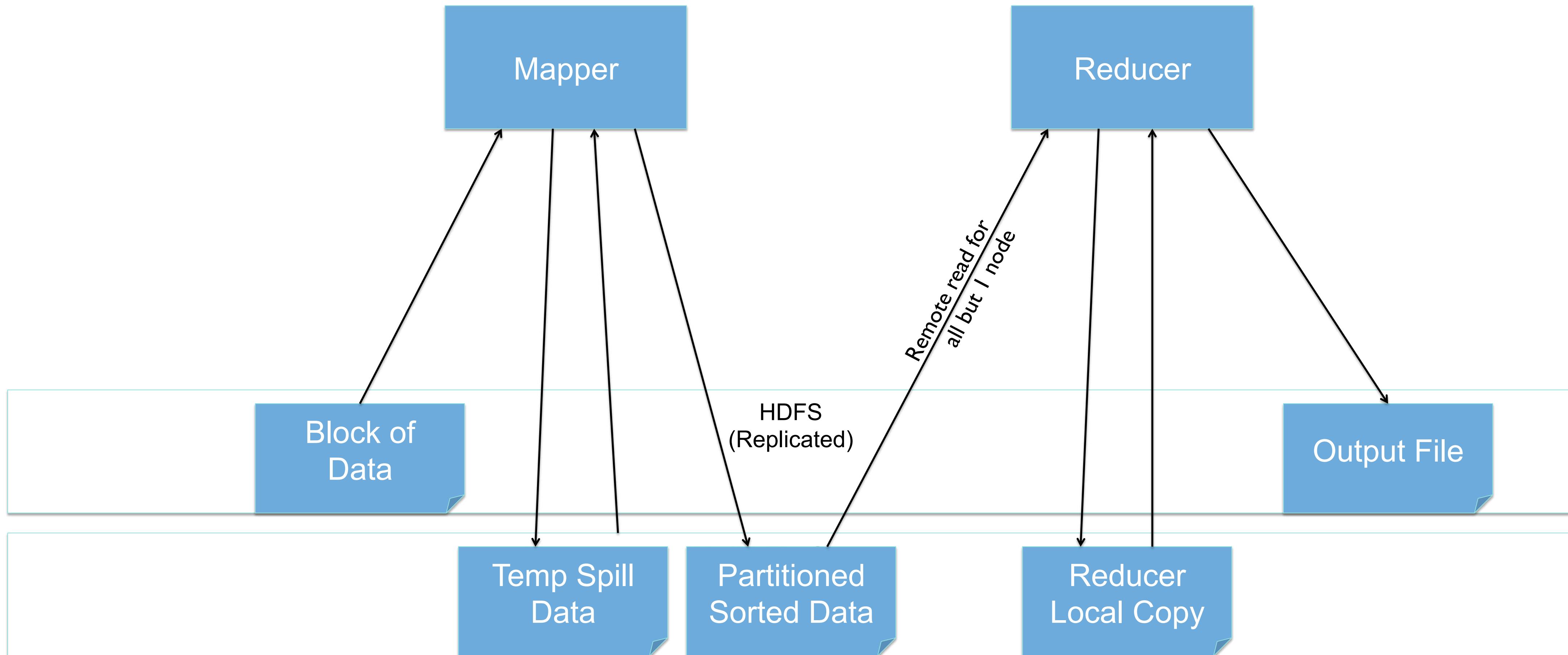
# Processing Engines

- MapReduce
- Abstractions
- Spark
- Impala

# MapReduce

- Oldie but goody
- Restrictive Framework / Innovative Workarounds
- Extreme Batch

# MapReduce Basic High Level



# Abstractions

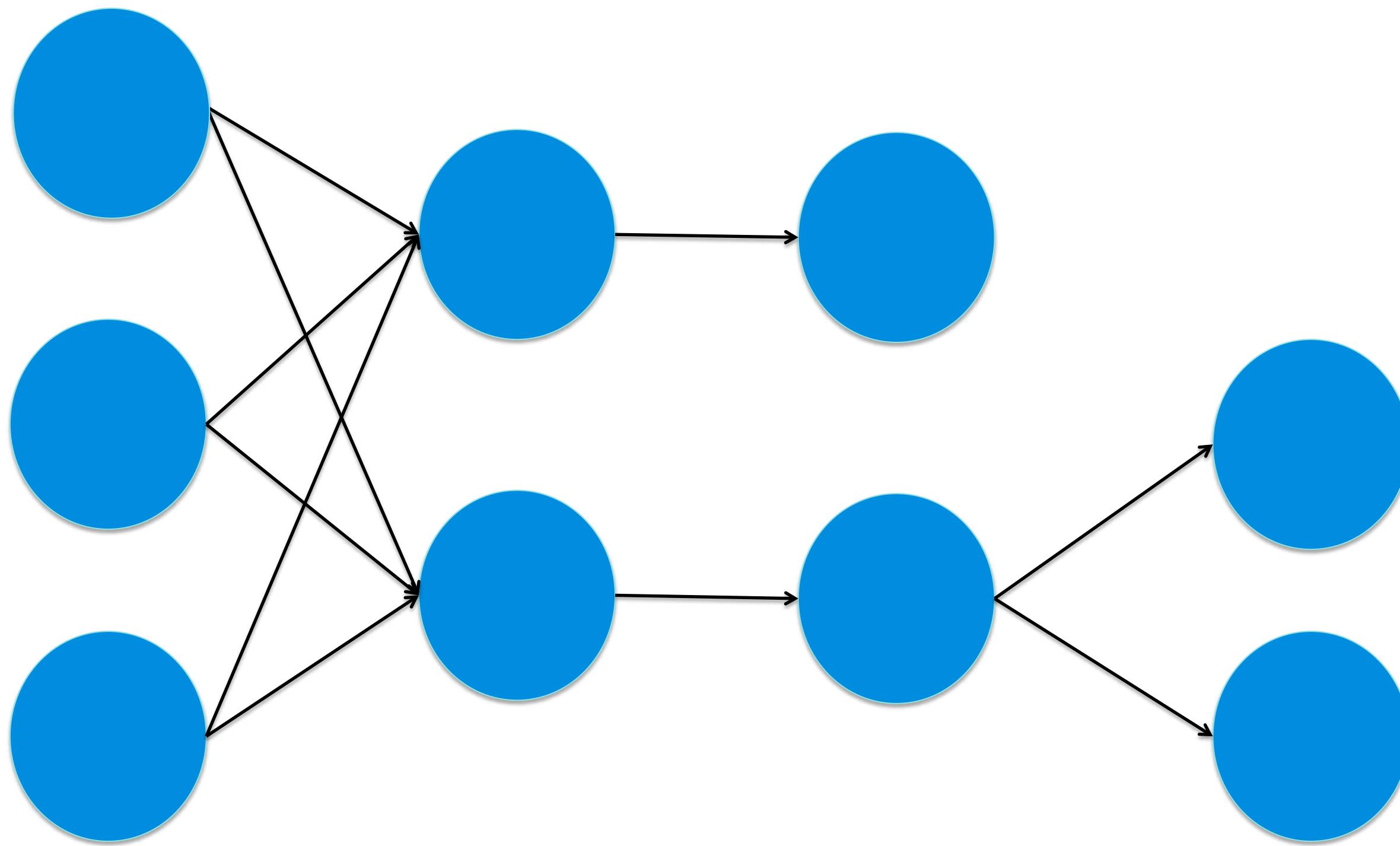
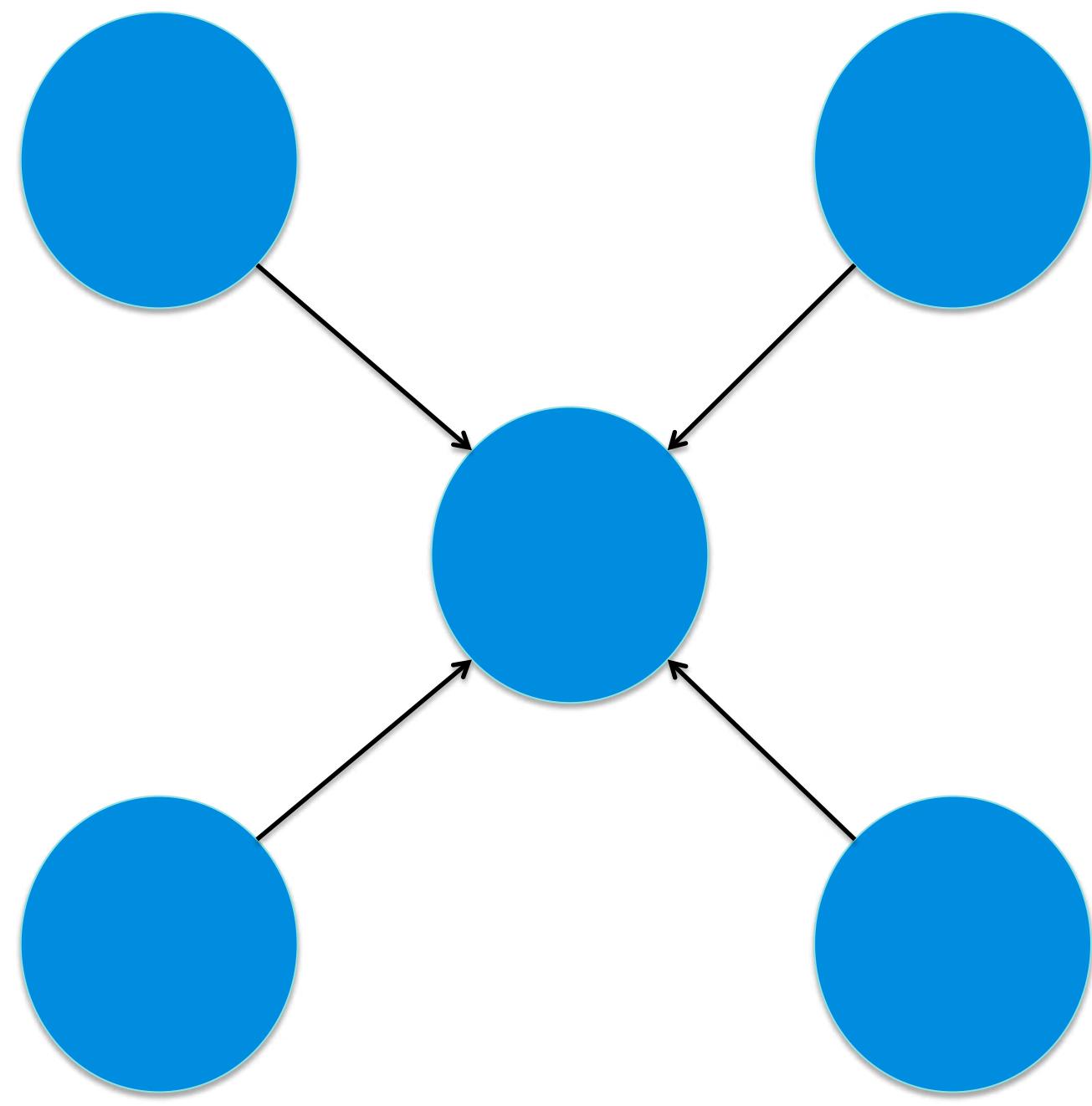
- SQL
  - Hive
- Script/Code
  - Pig: Pig Latin
  - Crunch: Java/Scala
  - Cascading: Java/Scala

# Spark

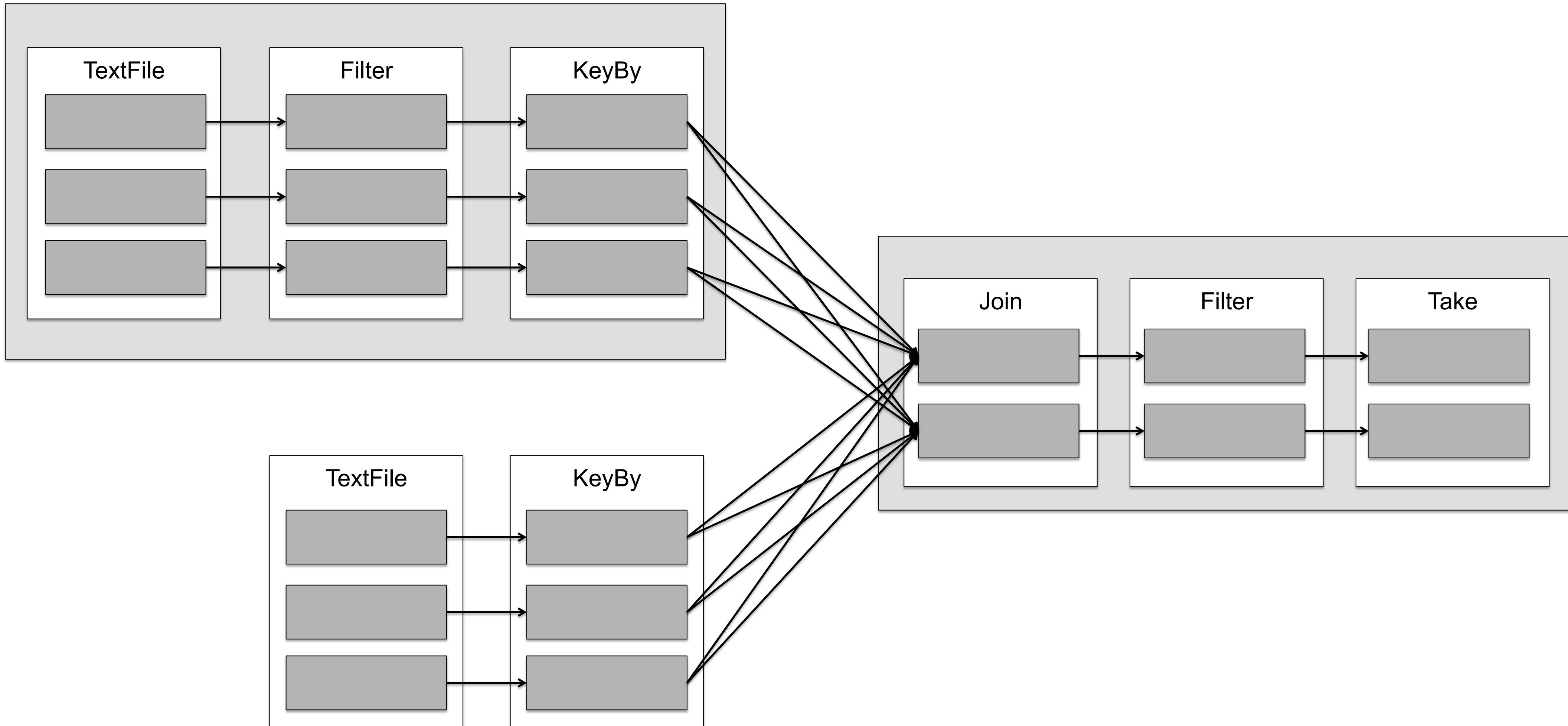
- The New Kid that isn't that New Anymore
- Easily 10x less code
- Extremely Easy and Powerful API
- Very good for iterative processing (machine learning, graph processing, etc.)
- Scala, Java, and Python support
- RDDs
- Has Graph, ML and SQL engines built on it
- R integration (SparkR)



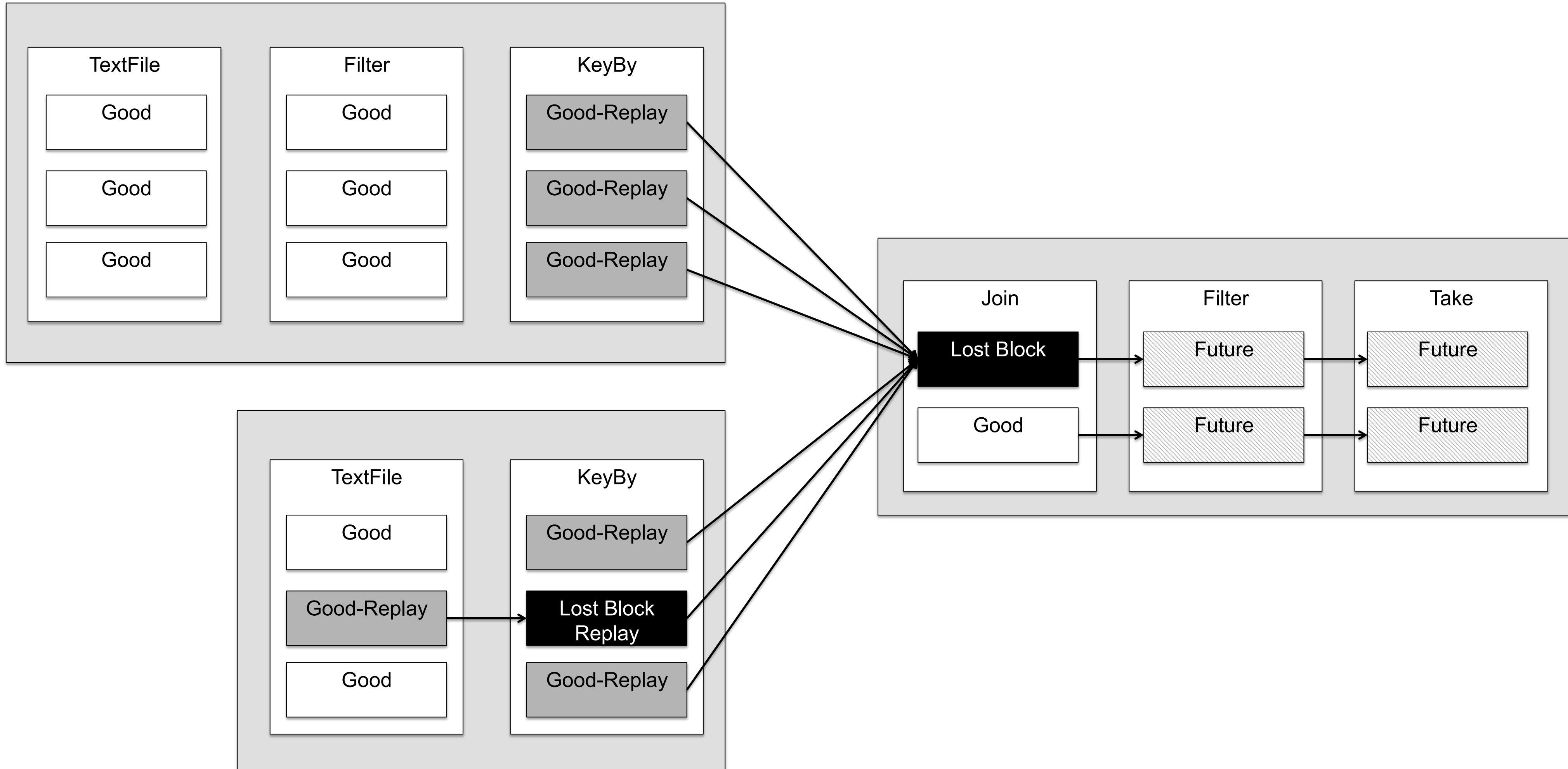
# Spark - DAG



# Spark - DAG



# Spark - DAG



# Is Spark replacing Hadoop?

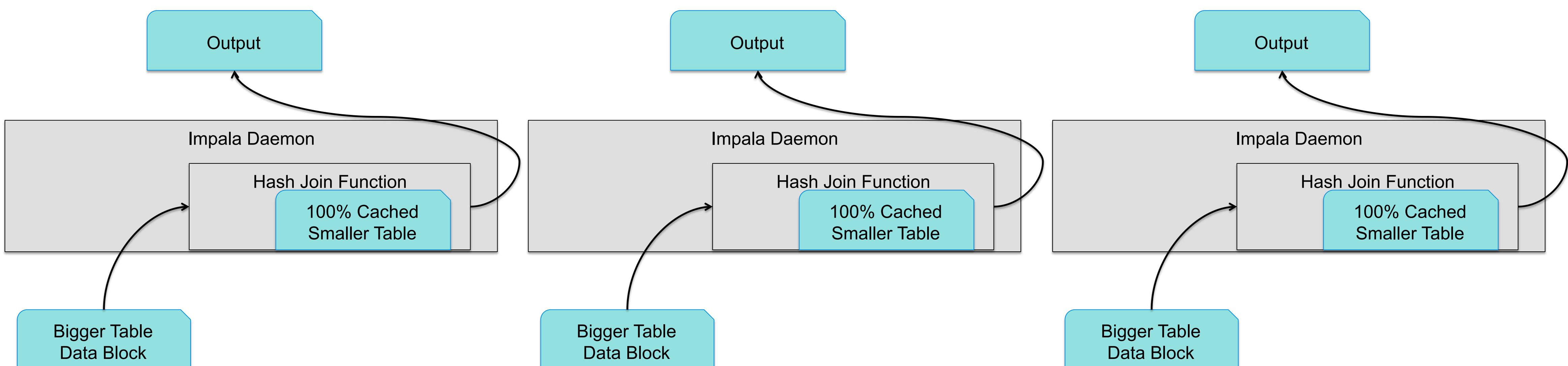
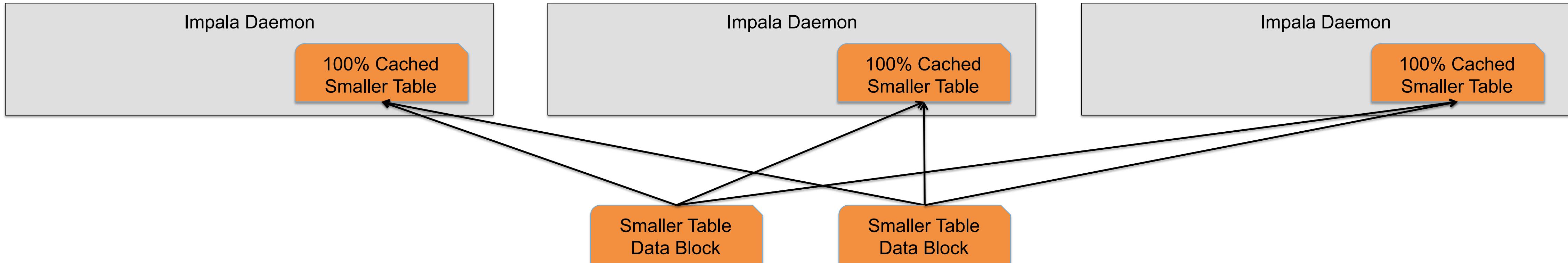
- Spark is an execution engine
  - Much faster than MR
  - Easier to program
- Spark is replacing MR

# Impala

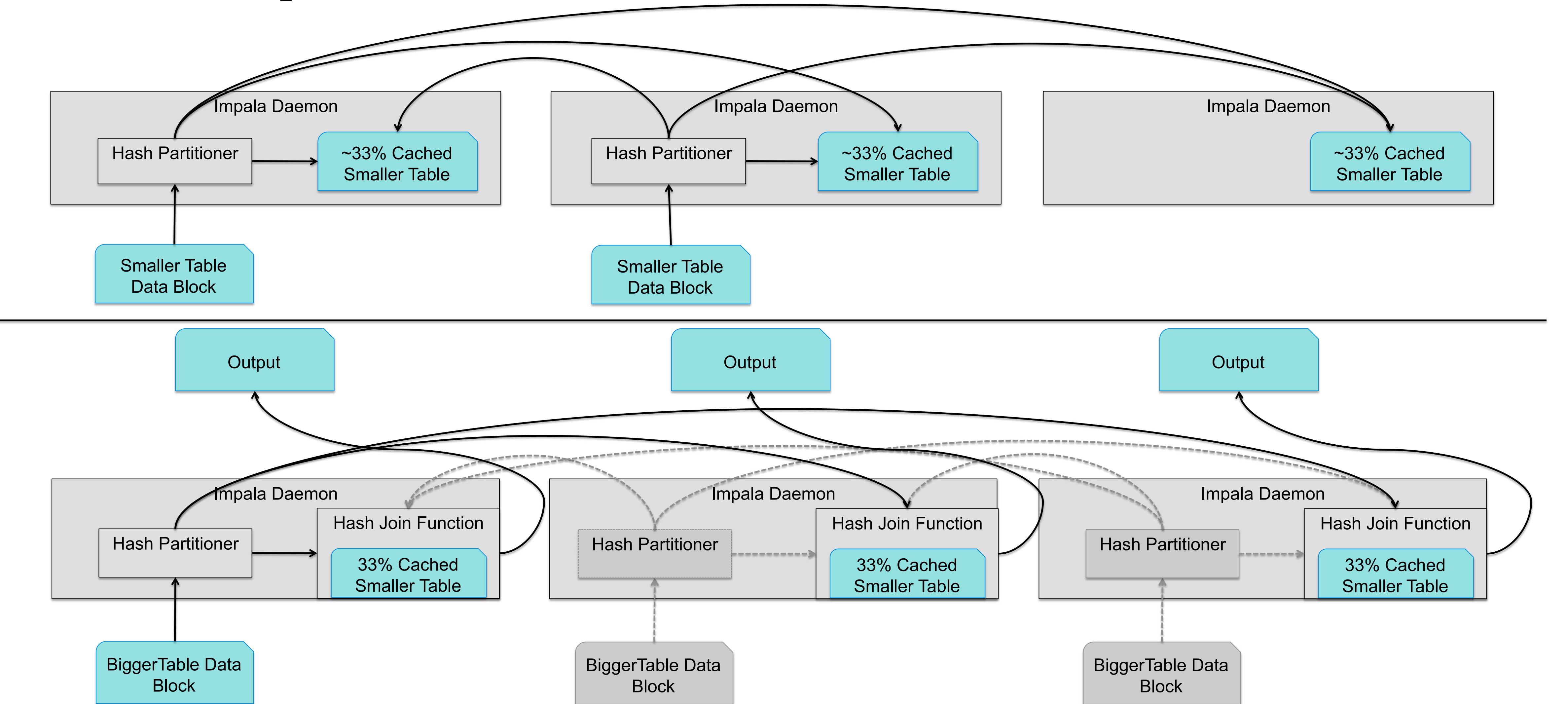


- MPP Style SQL Engine on top of Hadoop
- Very Fast
- High Concurrency
- Analytical windowing functions

# Impala – Broadcast Join



# Impala – Partitioned Hash Join



# Presto



- Facebook's SQL engine replacement for Hive
- Written in Java
- Doesn't build on top of MapReduce
- Very few commercial vendors supporting it

# Strata+ Hadoop

---

WORLD

PRESENTED BY

O'REILLY®

cloudera®

# Batch processing in our use-case

[strataconf.com](http://strataconf.com)

#StrataHadoop

# Batch processing in fraud-detection

- Reporting
  - How many events come daily?
  - How does it compare to last year?
- Machine Learning
  - Automated rules changes
- Other processing
  - Tracking device history and updating profiles

# Reporting

- Need a fast JDBC-compliant SQL framework
- Impala or Hive or Presto?
- We choose Impala
  - Fast
  - Highly concurrent
  - JDBC/ODBC support

# Machine Learning

- Options
  - MLLib (with Spark)
  - Oryx
  - H2O
  - Mahout
- We recommend MLLib because of larger use of Spark in our architecture.

# Other Processing

- Options:
  - MapReduce
  - Spark
- We choose Spark
  - Much faster than MR
  - Easier to write applications due to higher level API
  - Re-use of code between streaming and batch

# Strata+ Hadoop

---

WORLD

PRES EN TED BY

O'REILLY®

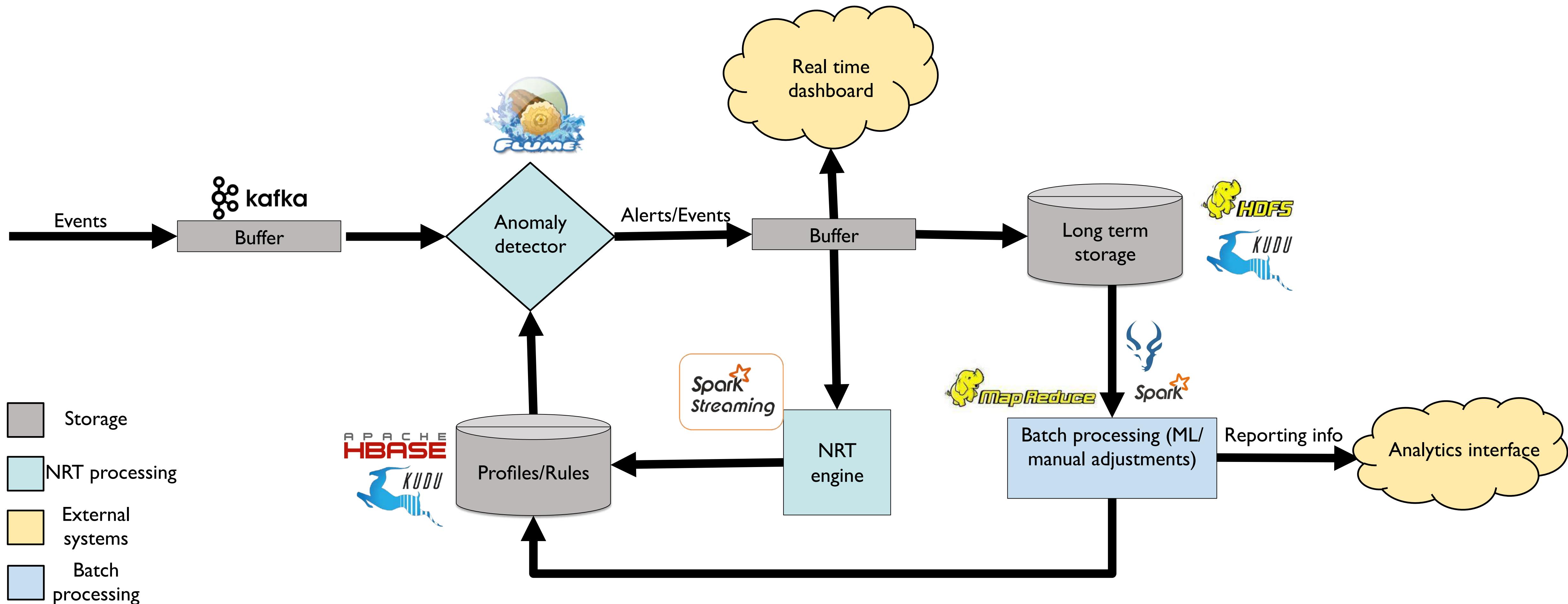
cloudera®

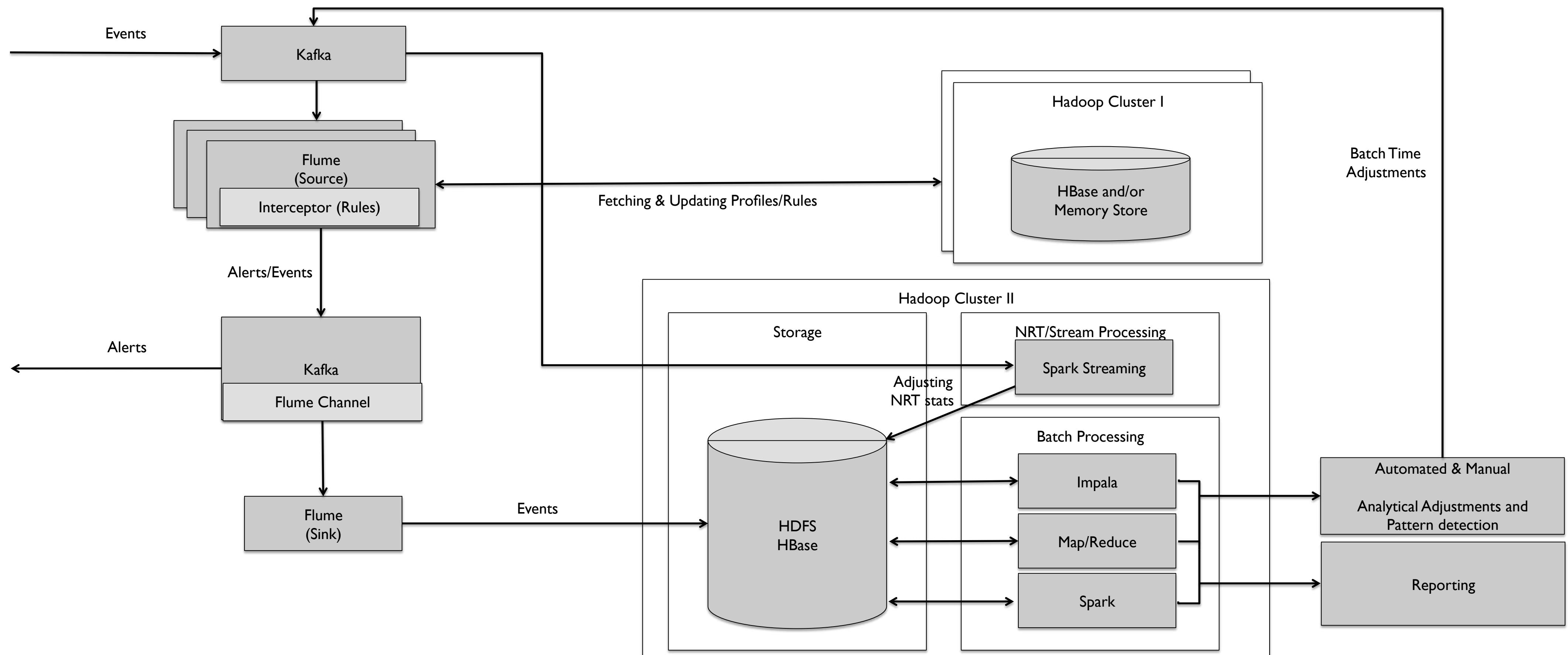
# Overall Architecture Overview

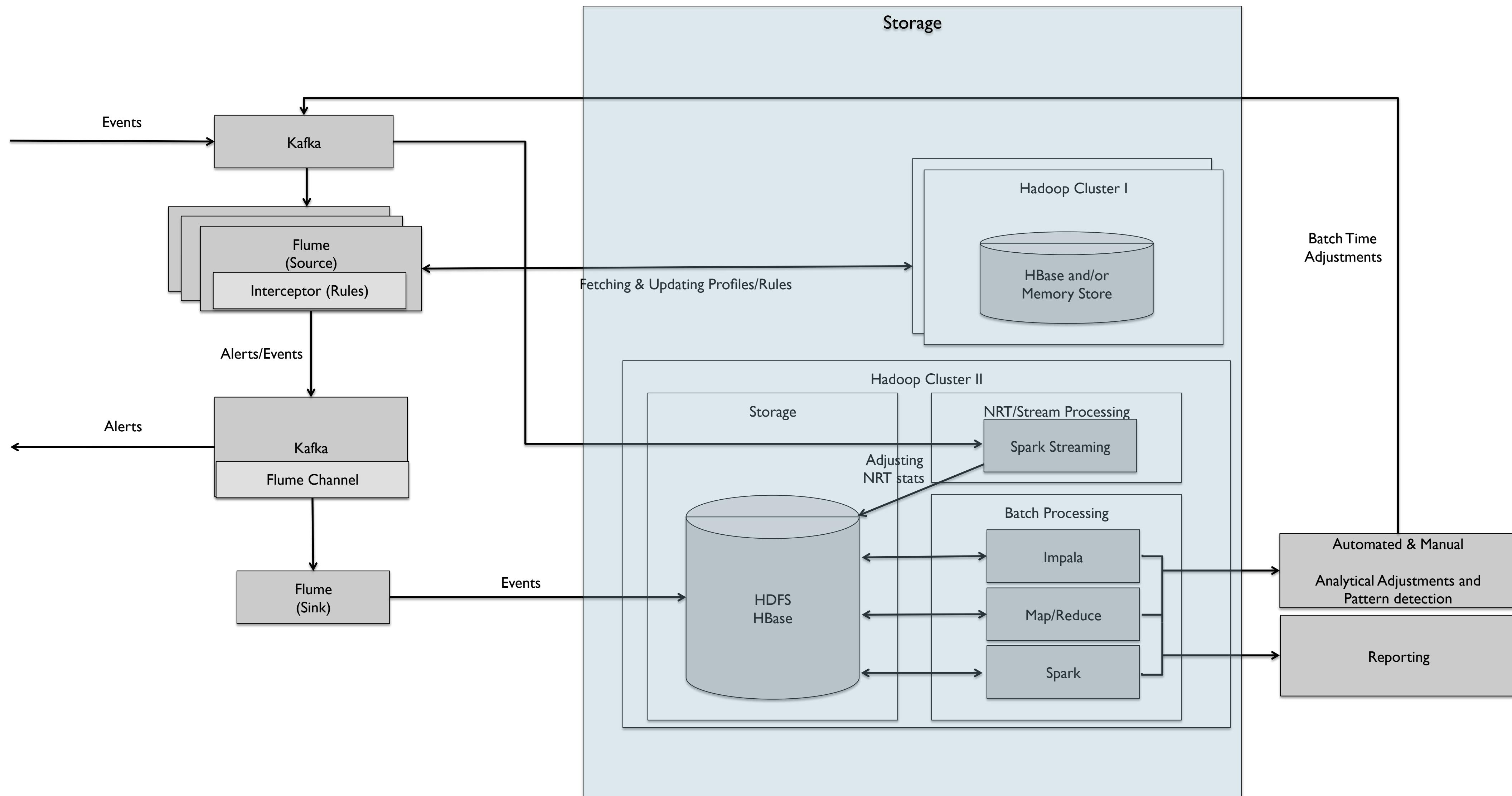
[strataconf.com](http://strataconf.com)

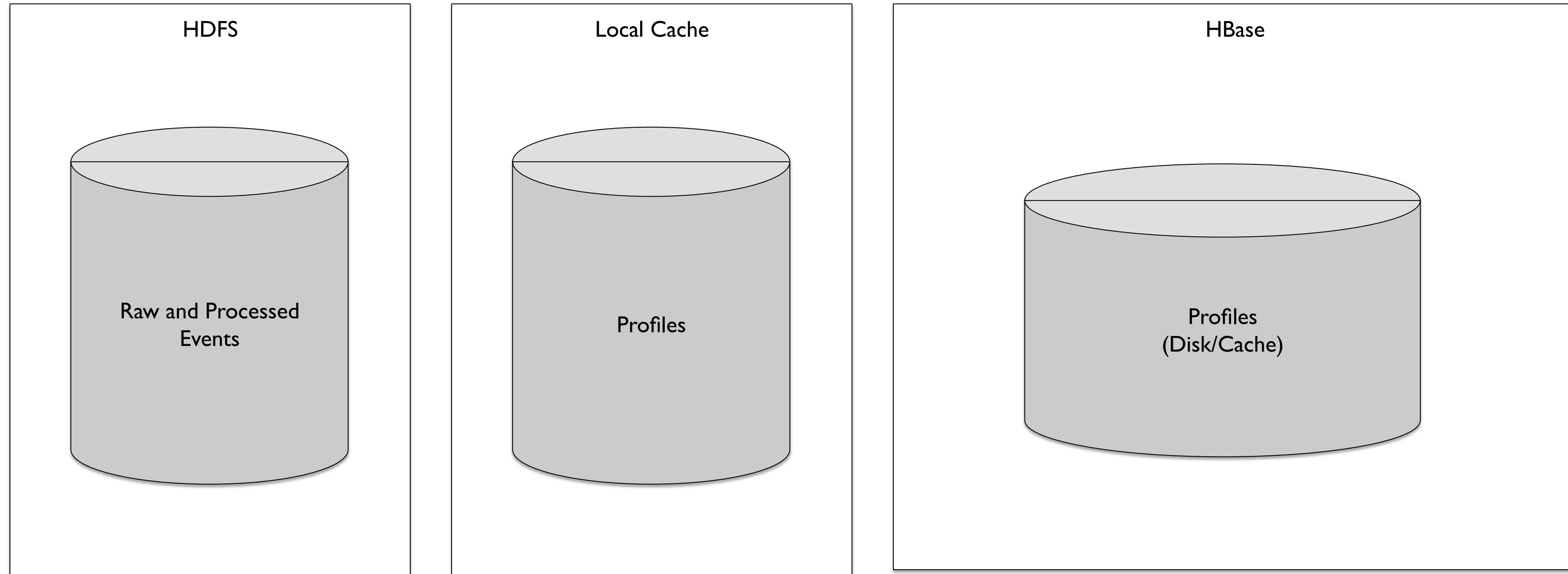
#StrataHadoop

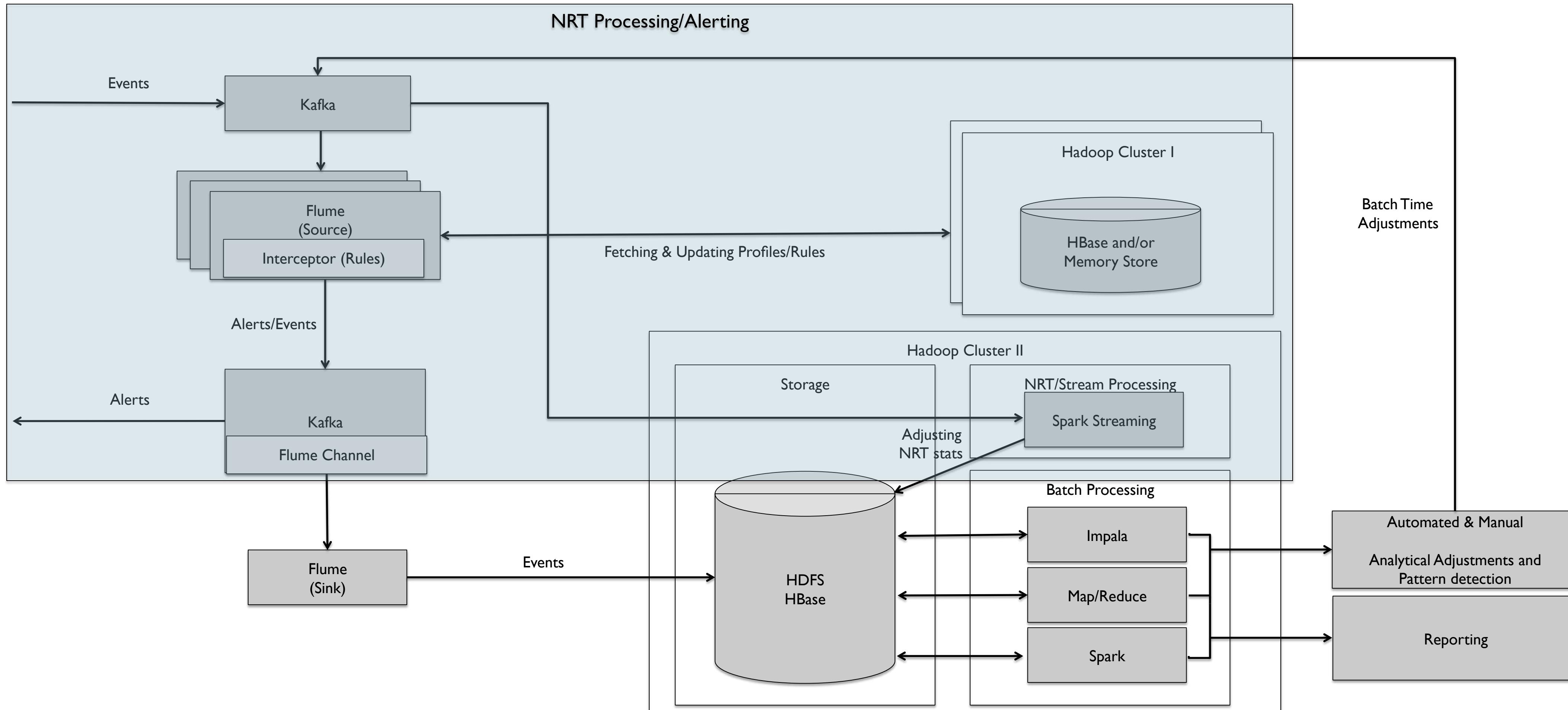
# High level architecture

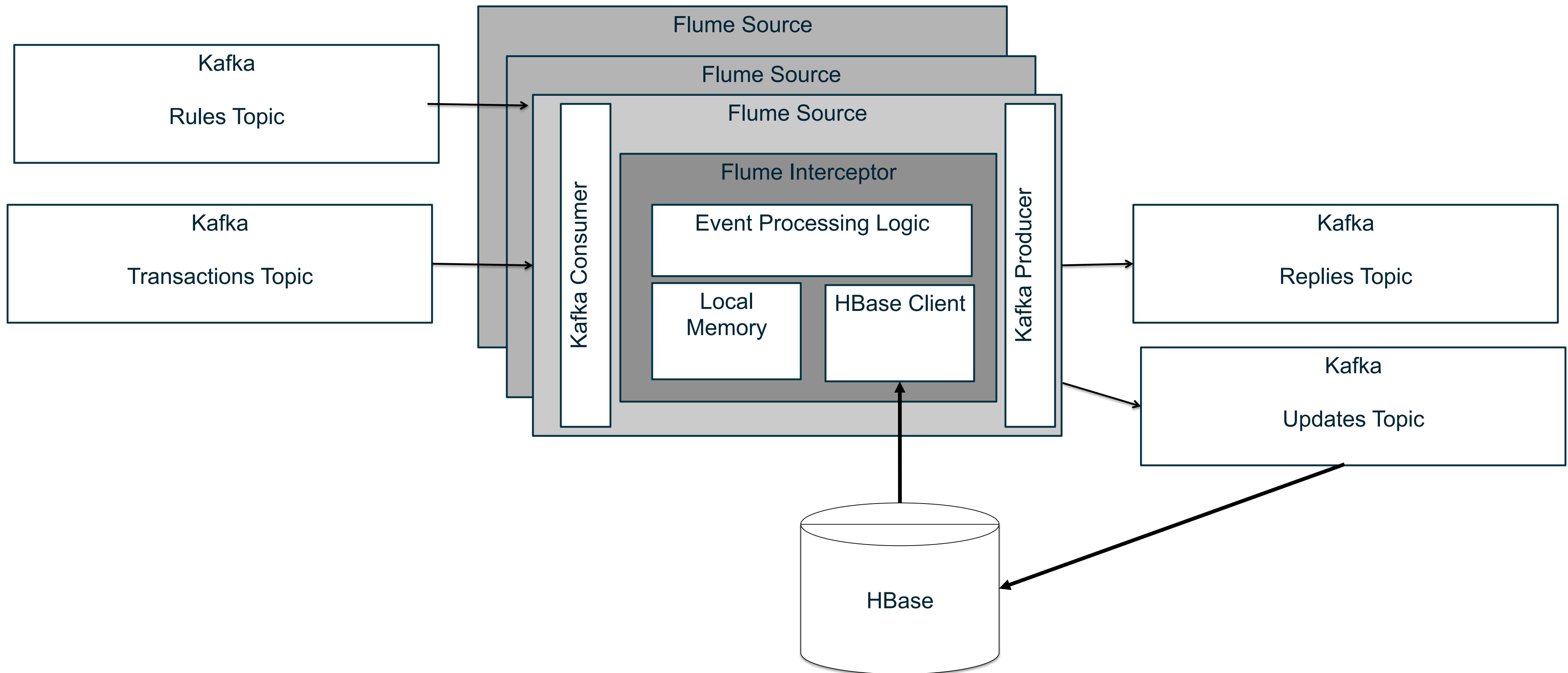


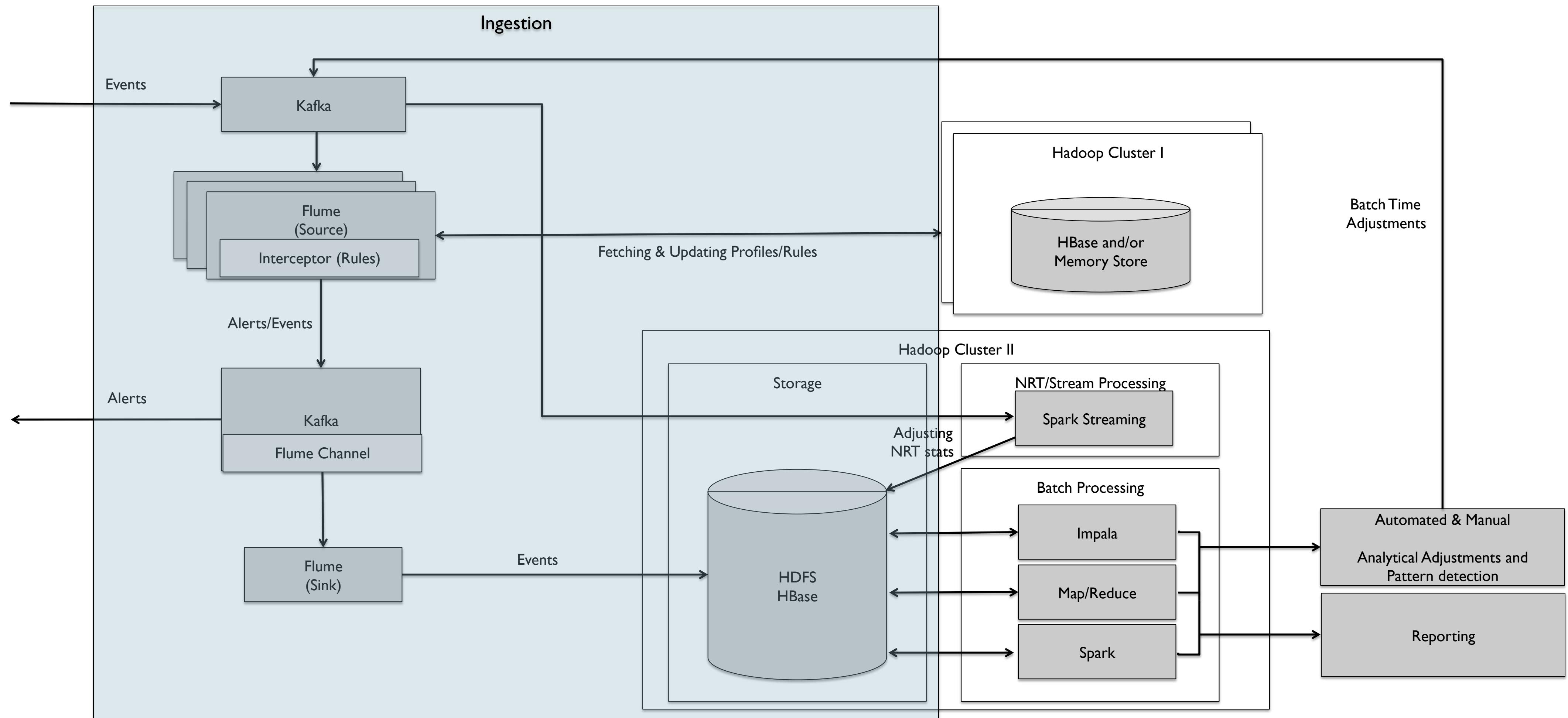


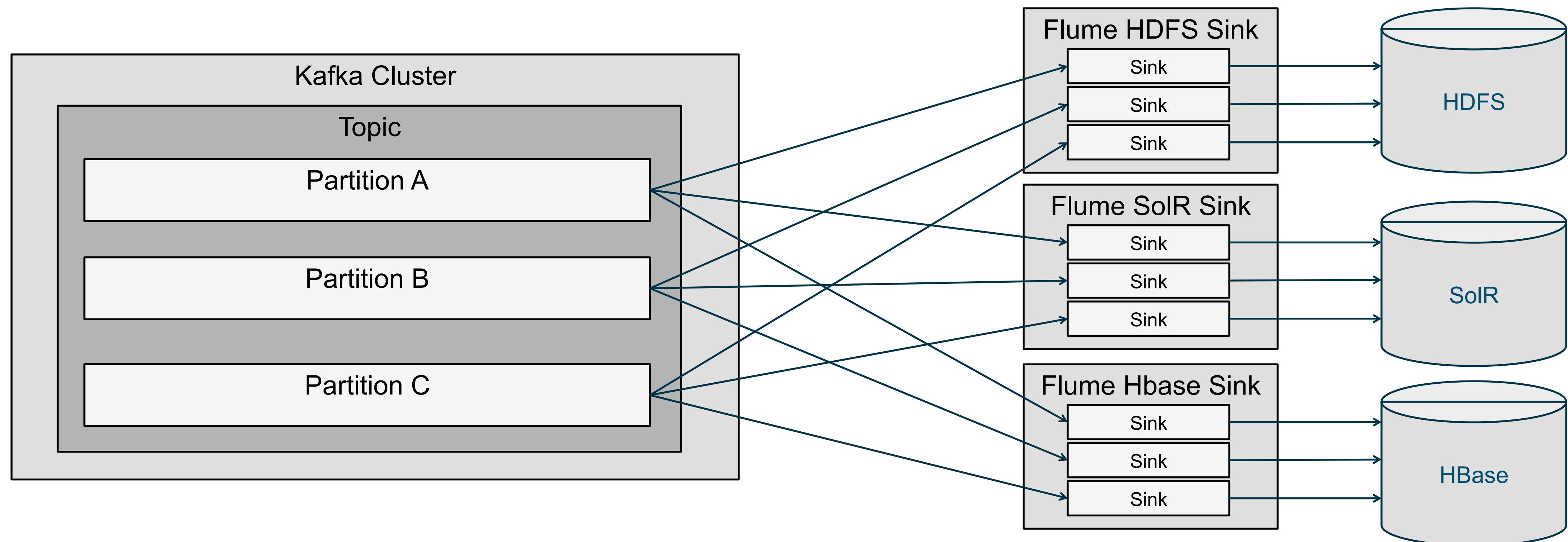


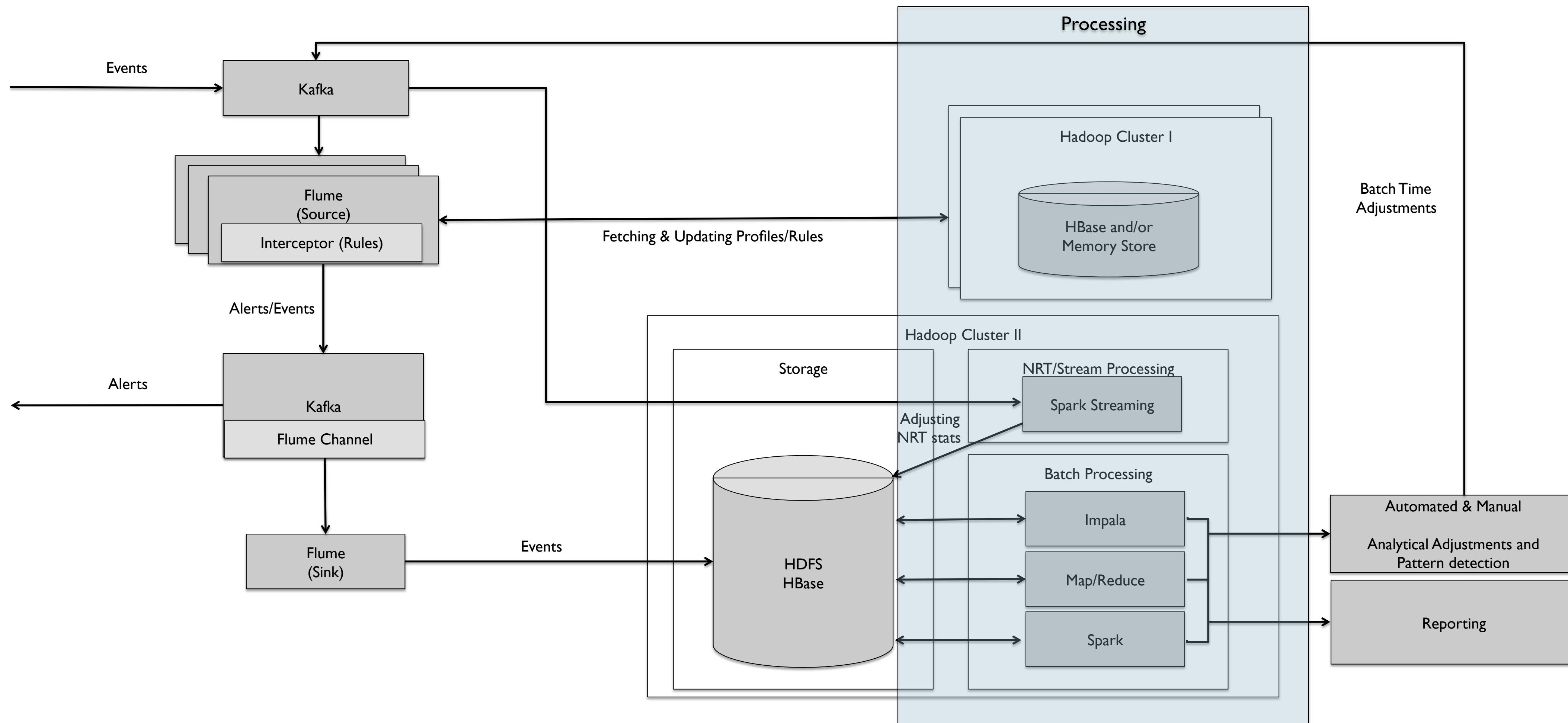


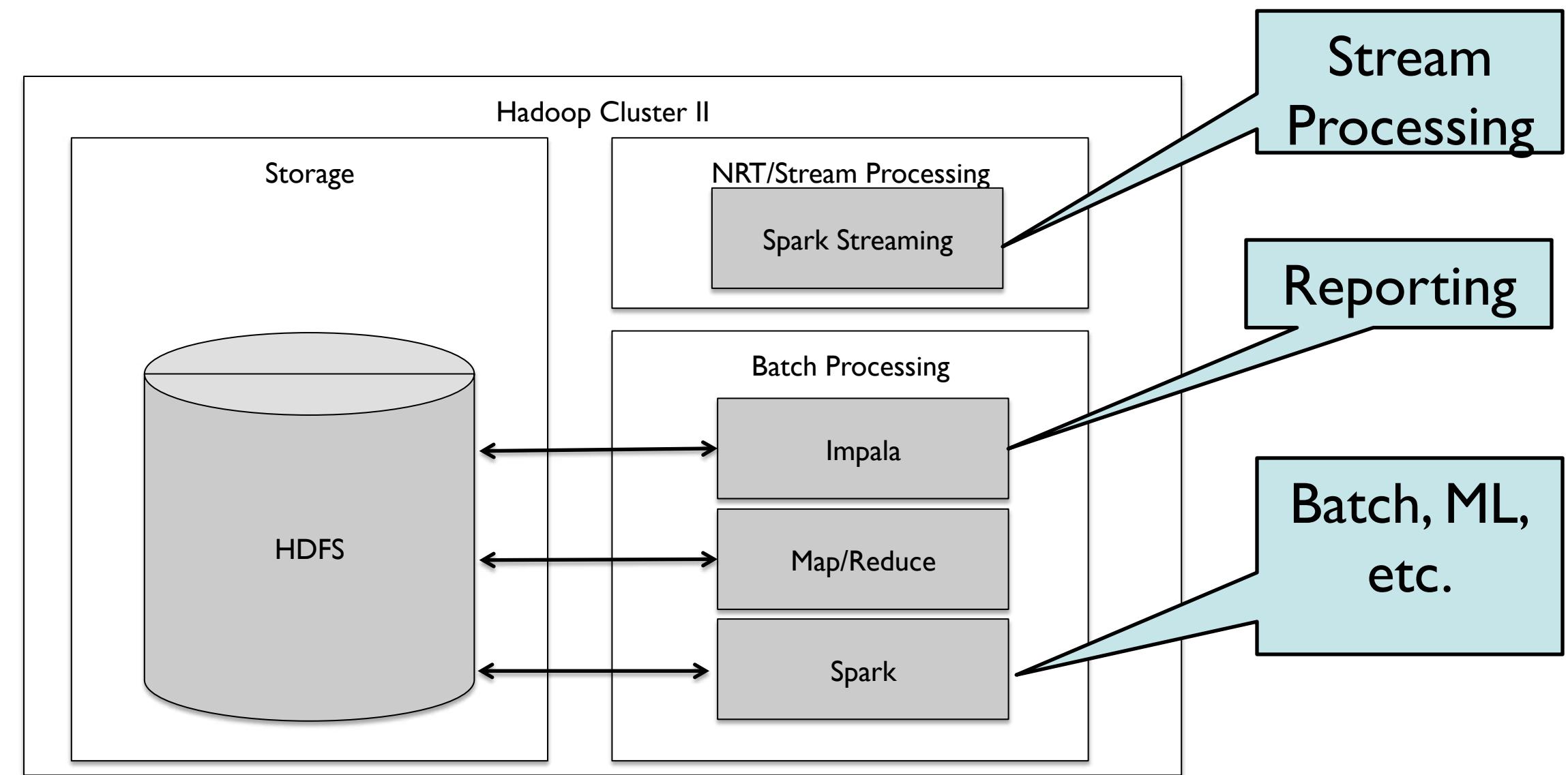












# Strata+ Hadoop

---

WORLD

PRESENTED BY

O'REILLY®

cloudera®

# Demo!

[strataconf.com](http://strataconf.com)

#StrataHadoop

# Strata+ Hadoop

---

WORLD

PRES EN TED BY

O'REILLY®

cloudera®

# Where else to find us?

[strataconf.com](http://strataconf.com)

#StrataHadoop

# Book Signing!

- This evening at 5:35 PM at the Cloudera booth.

# Other Sessions

- Ask Us Anything session (all) – Thursday, 2:05 pm
- Reliability guarantees in Kafka (Gwen) – Thursday, 12:05 pm
- Putting Kafka into overdrive (Gwen) – Thursday, 5:25 pm
- Introduction to Apache Spark for Java and Scala developers (Ted) – Friday, 2:05 pm

# Strata+ Hadoop

---

WORLD

PRES EN TED BY



[strataconf.com](http://strataconf.com)

#StrataHadoop

# Thank you!

@hadooparchbook  
[tiny.cloudera.com/app-arch-london](http://tiny.cloudera.com/app-arch-london)  
Gwen Shapira | @gwenshap  
Jonathan Seidman | @jseidman  
Ted Malaska | @ted\_malaska  
Mark Grover | @mark\_grover

edureka!



Hadoop Ecosystem

# Agenda for today's Session

## Hadoop Ecosystem

- HDFS -> Hadoop Distributed File System
- YARN -> Yet Another Resource Negotiator
- MapReduce -> Data processing using programming
- Spark -> In-memory Data Processing
- PIG, HIVE-> Data Processing Services using Query (SQL-like)
- HBase -> NoSQL Database
- Mahout, Spark MLlib -> Machine Learning
- Apache Drill -> SQL on Hadoop
- Zookeeper -> Managing Cluster
- Oozie -> Job Scheduling
- Flume, Sqoop -> Data Ingesting Services
- Solr & Lucene -> Searching & Indexing
- Ambari -> Provision, Monitor and Maintain cluster



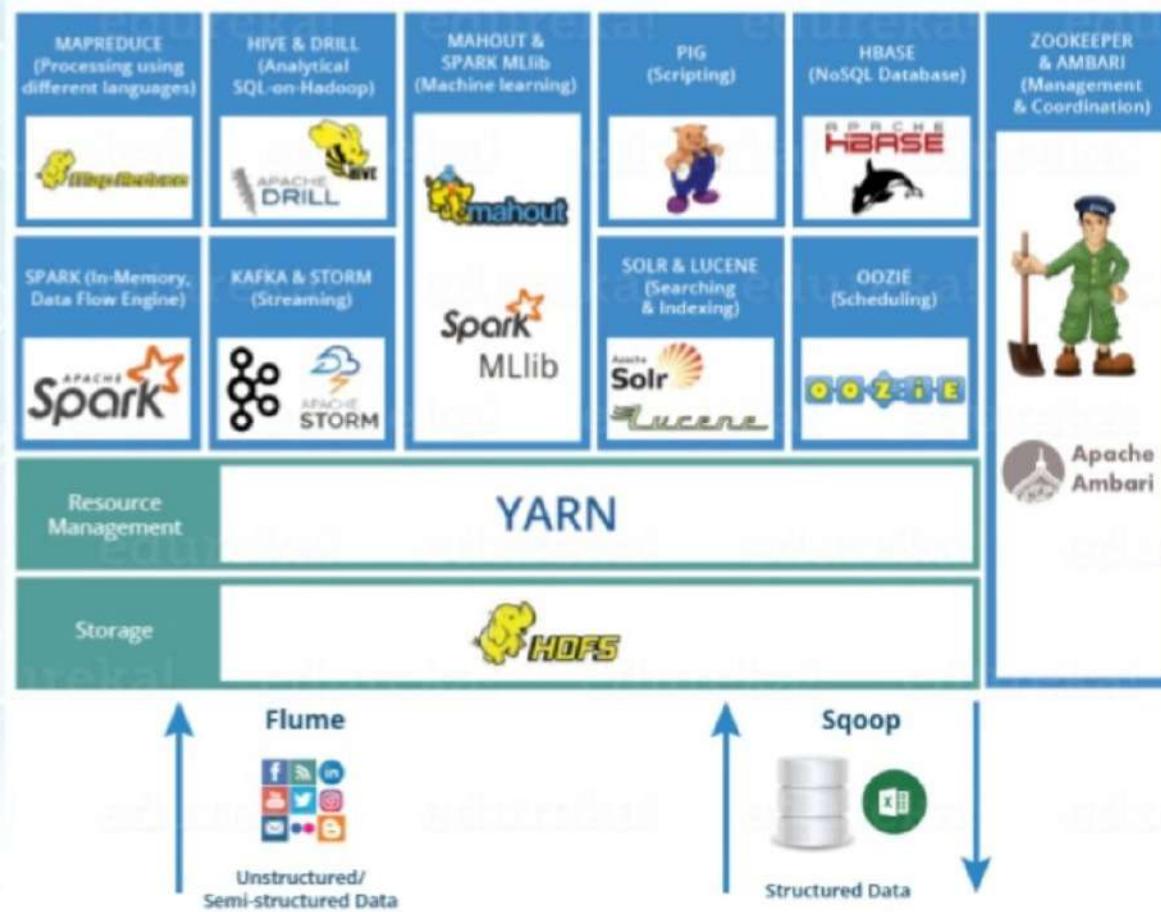
# Agenda for today's Session

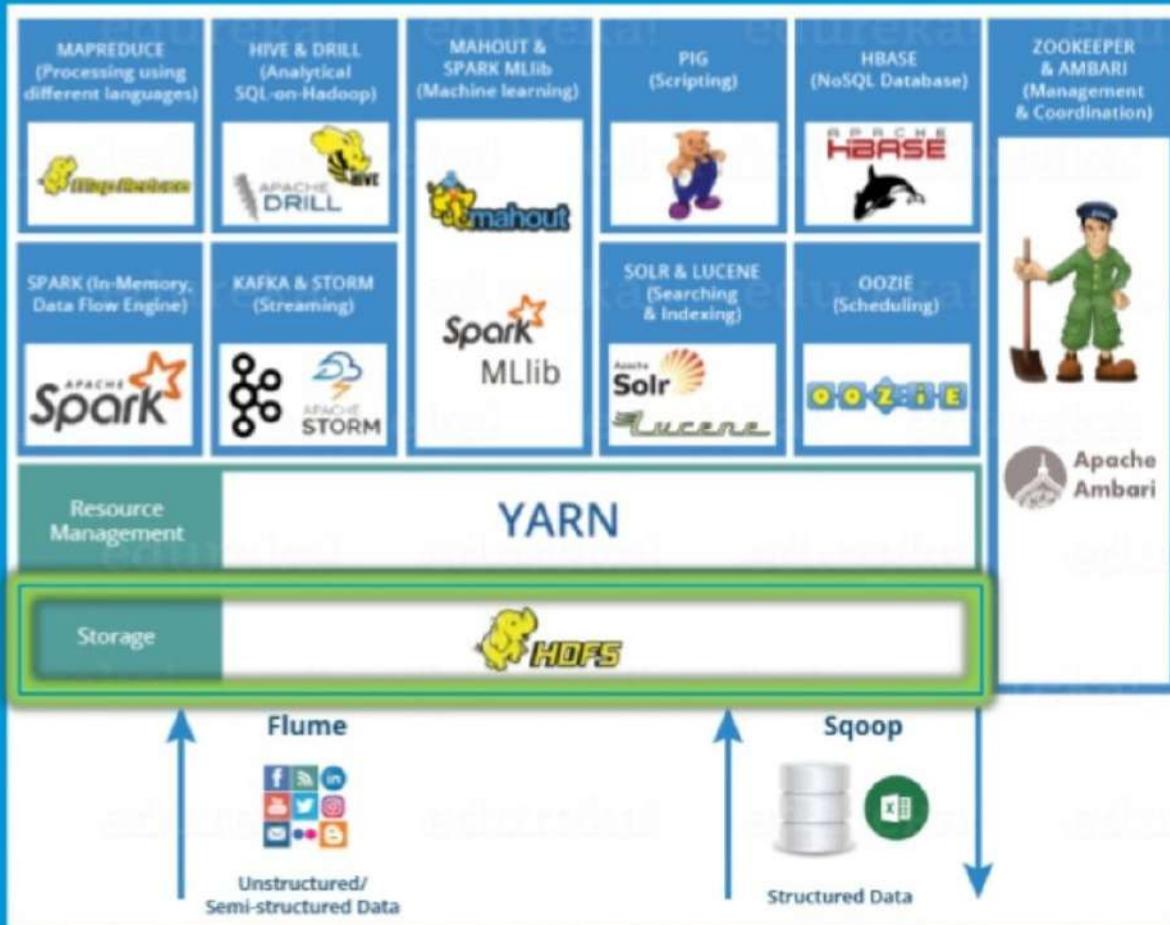
## Hadoop Ecosystem

- HDFS -> Hadoop Distributed File System
- YARN -> Yet Another Resource Negotiator
- MapReduce -> Data processing using programming
- Spark -> In-memory Data Processing
- PIG, HIVE-> Data Processing Services using Query (SQL-like)
- HBase -> NoSQL Database
- Mahout, Spark MLlib -> Machine Learning
- Apache Drill -> SQL on Hadoop
- Zookeeper -> Managing Cluster
- Oozie -> Job Scheduling
- Flume, Sqoop -> Data Ingesting Services
- Solr & Lucene -> Searching & Indexing
- Ambari -> Provision, Monitor and Maintain cluster



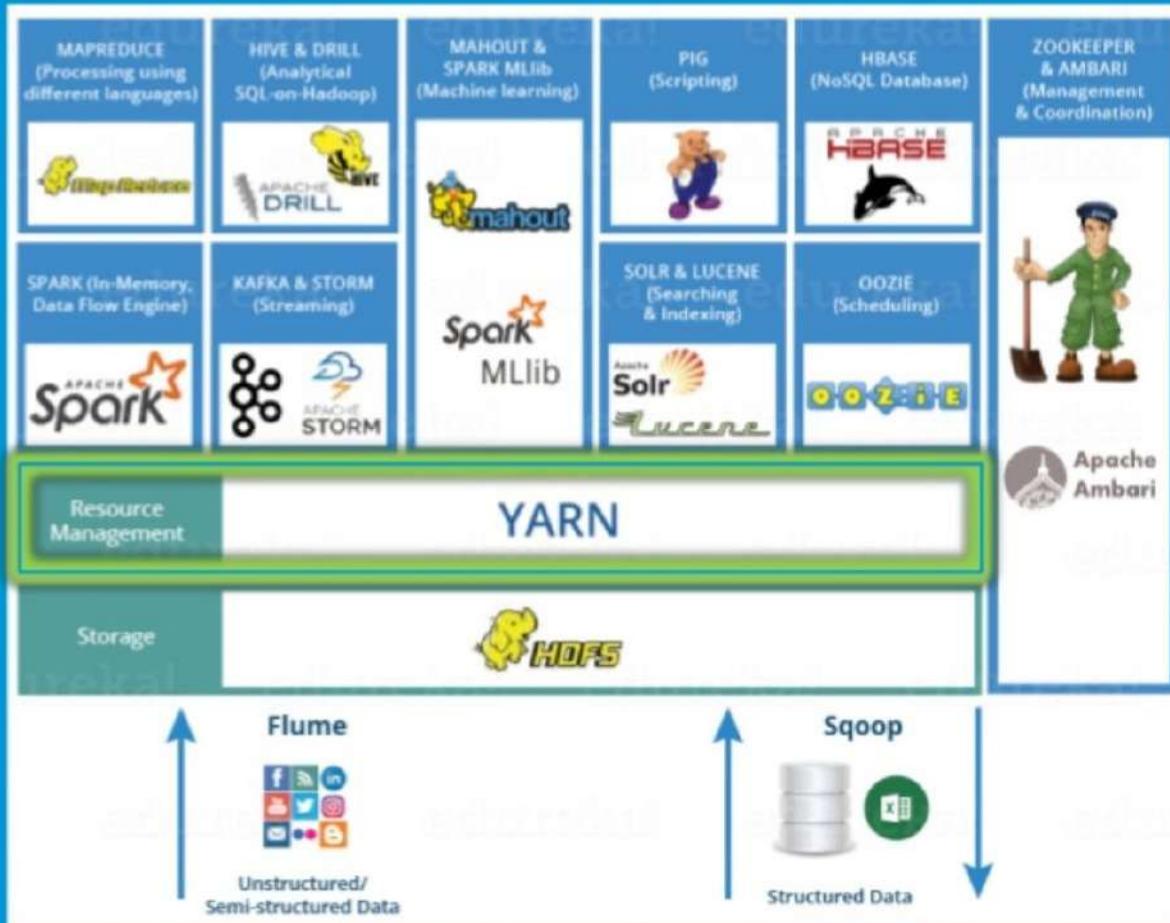
# Hadoop Ecosystem





- Stores different types of large data sets (i.e. structured, unstructured and semi structured data)
- HDFS creates a level of abstraction over the resources, from where we can see the whole HDFS as a single unit
- Stores data across various nodes and maintains the log file about the stored data (metadata)
- HDFS has two core components, i.e. NameNode and DataNode



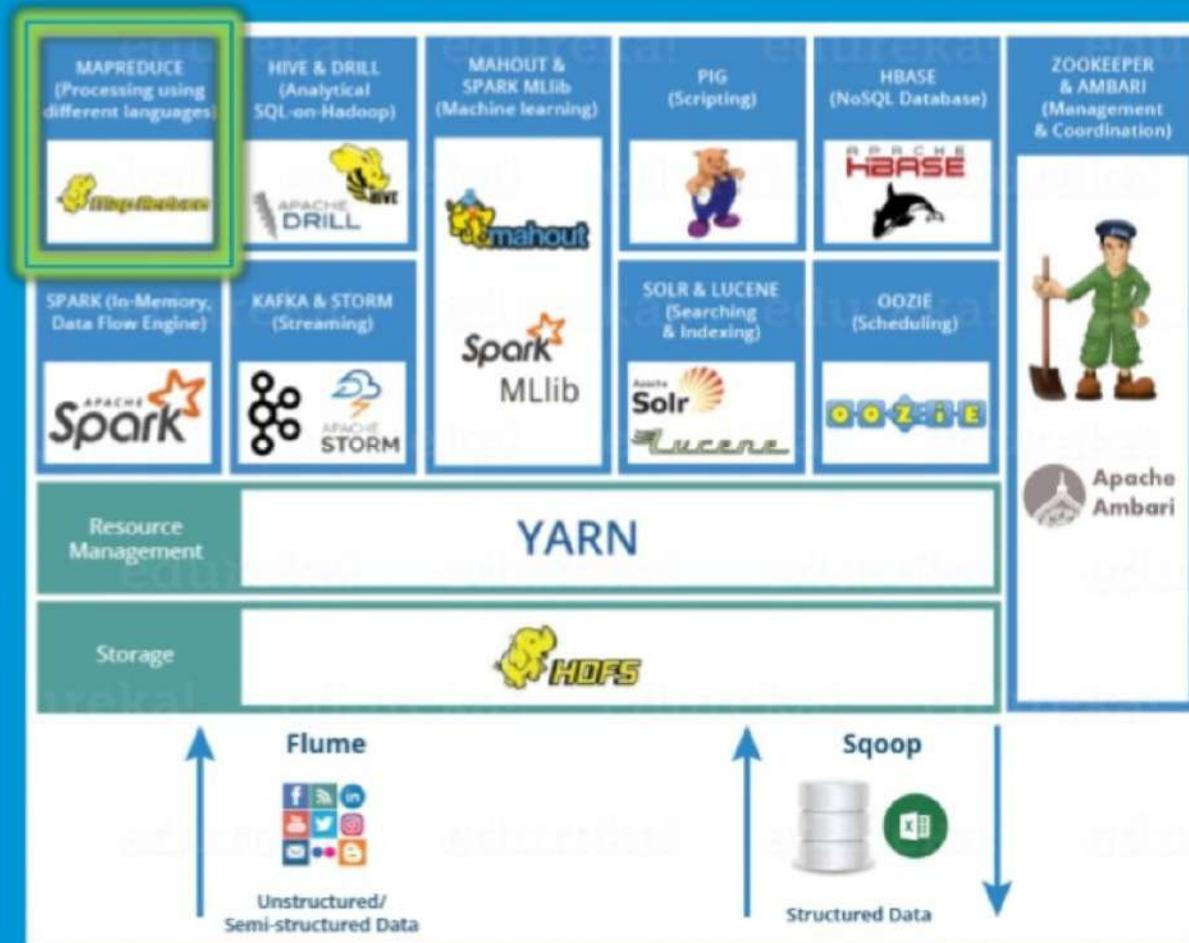


- Performs all your processing activities by allocating resources and scheduling tasks
- Two services: ResourceManager and NodeManager
- ResourceManager: Manages resources and schedule applications running on top of YARN
- NodeManager: Manages containers and monitors resource utilization in each container



- Performs all your processing activities by allocating resources and scheduling tasks
- Two services: ResourceManager and NodeManager
- ResourceManager: Manages resources and schedule applications running on top of YARN
- NodeManager: Manages containers and monitors resource utilization in each container

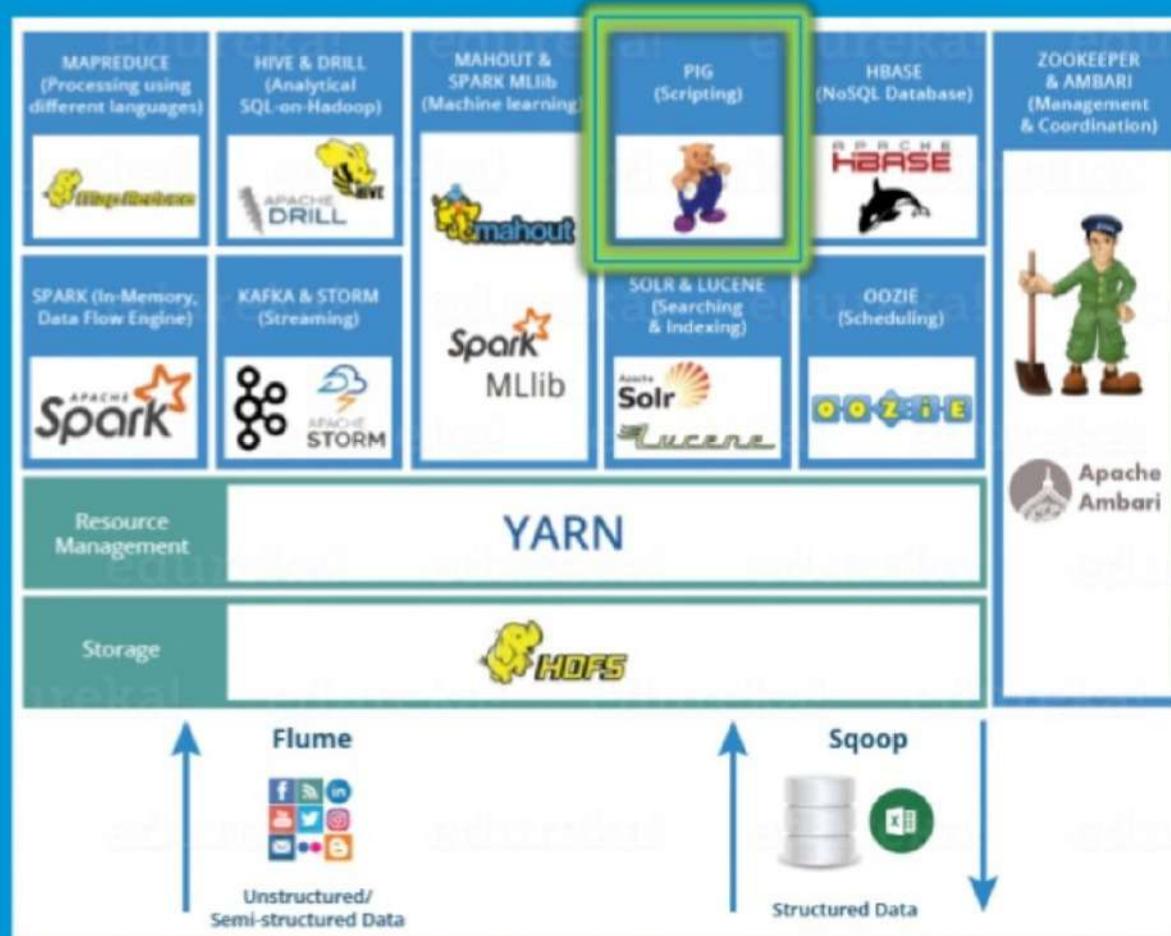




# MapReduce: Data Processing Using Programming [edureka!](#)

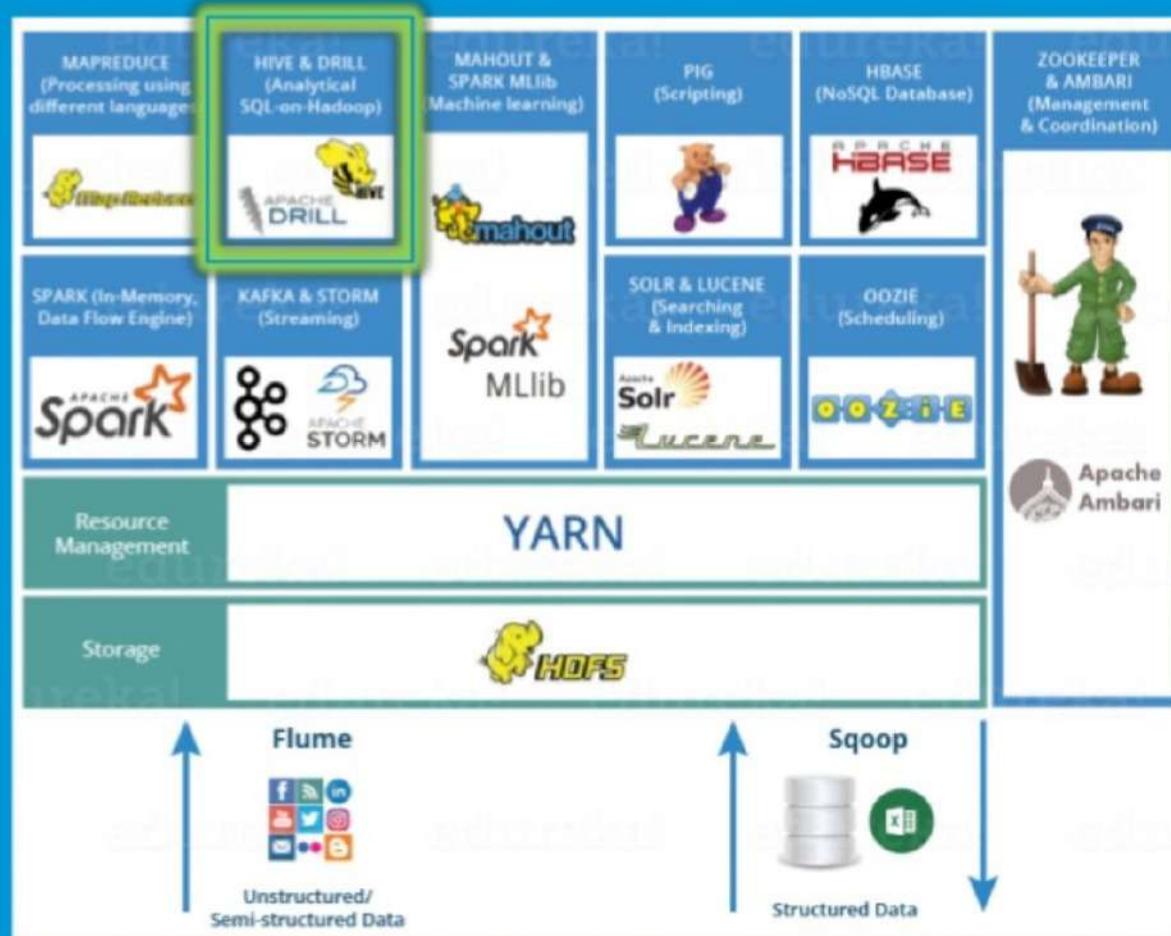
- Core component in a Hadoop Ecosystem for processing
- Helps in writing applications that processes large data sets using distributed and parallel algorithms
- In a MapReduce program, Map() and Reduce() are two functions
- Map function performs actions like filtering, grouping and sorting
- Reduce function aggregates and summarizes the result produced by map function







- PIG has two parts: Pig Latin, the language and the pig runtime, for the execution environment
- **1 line of pig latin = approx. 100 lines of Map-Reduce job**
- The compiler internally converts pig latin to MapReduce
- It gives you a platform for building data flow for ETL (Extract, Transform and Load)
- PIG first loads the data, then performs various functions like grouping, filtering, joining, sorting, etc. and finally dumps the data on the screen or stores in HDFS.

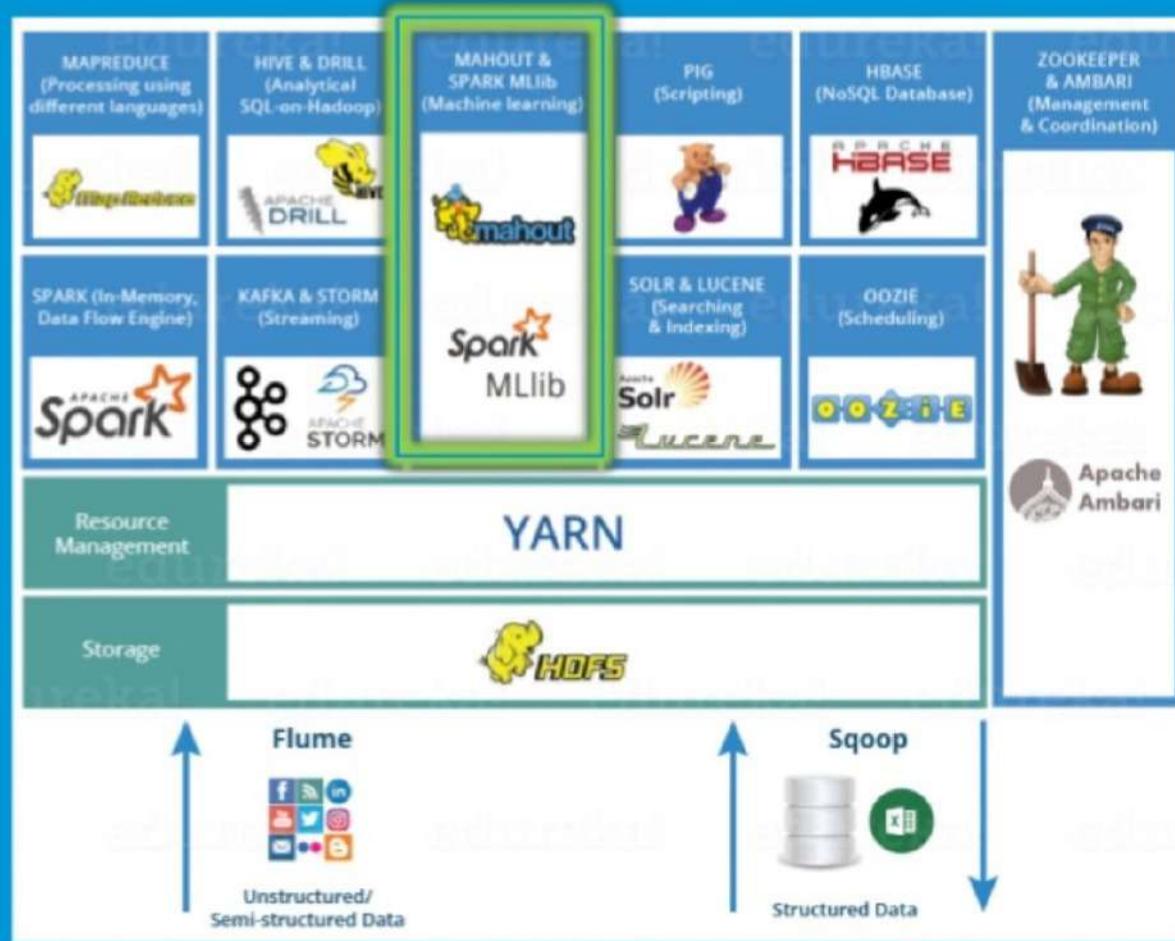




- A data warehousing component which analyses data sets in a distributed environment using SQL-like interface
- The query language of Hive is called Hive Query Language(HQL)
- 2 basic components: Hive Command Line and JDBC/ODBC driver
- Supports user defined functions (UDF) to accomplish specific needs



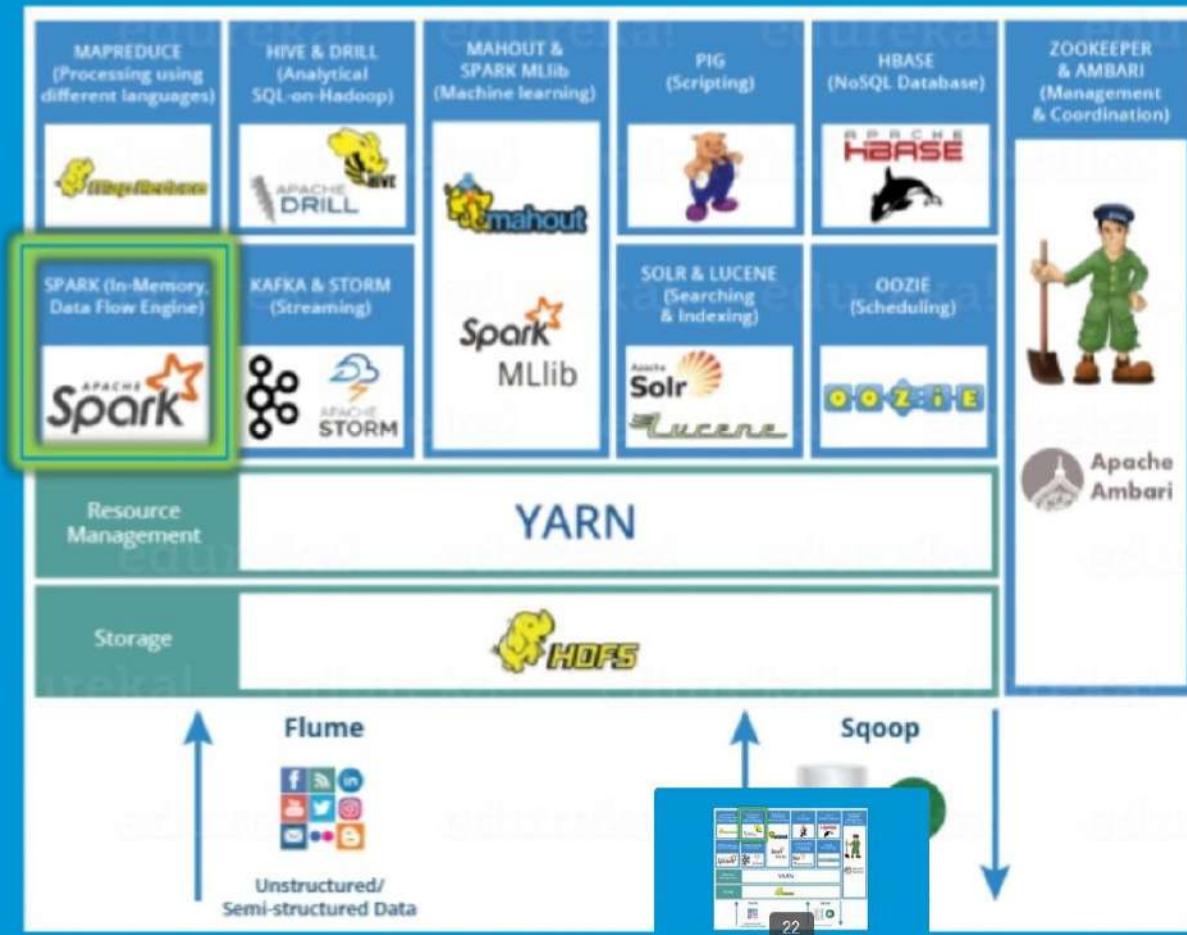
- A data warehousing component which analyses data sets in a distributed environment using SQL-like interface
- The query language of Hive is called Hive Query Language(HQL)
- 2 basic components: Hive Command Line and JDBC/ODBC driver
- Supports user defined functions (UDF) to accomplish specific needs



# Mahout: Machine Learning

- Provides an environment for creating machine learning applications
- It performs collaborative filtering, clustering and classification
- Provides a command line to invoke various algorithms.
- It has a predefined set of library which already contains different inbuilt algorithms for different use cases.





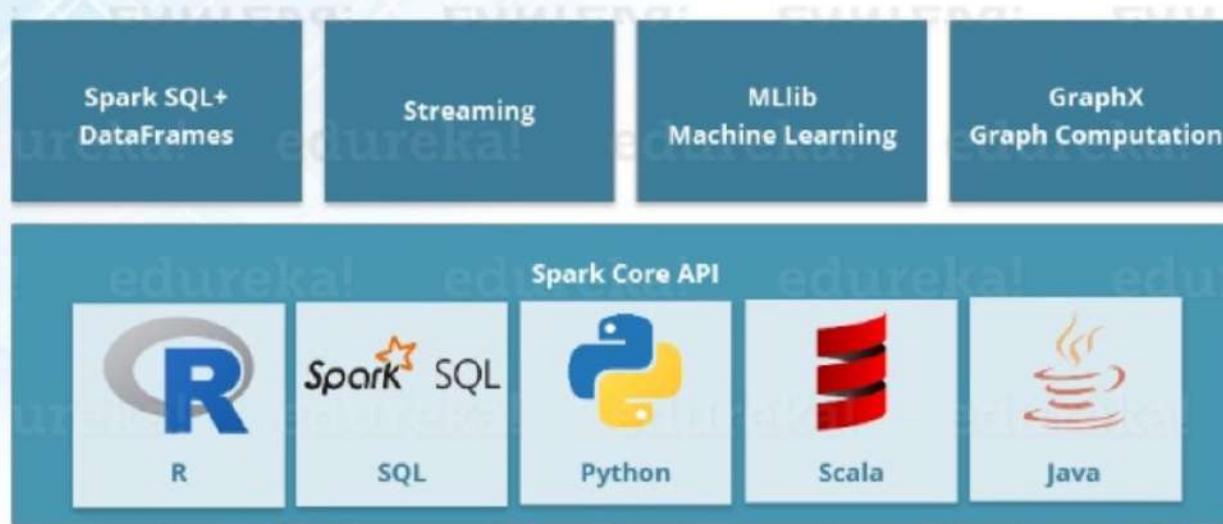
# Spark: In-memory Data Processing

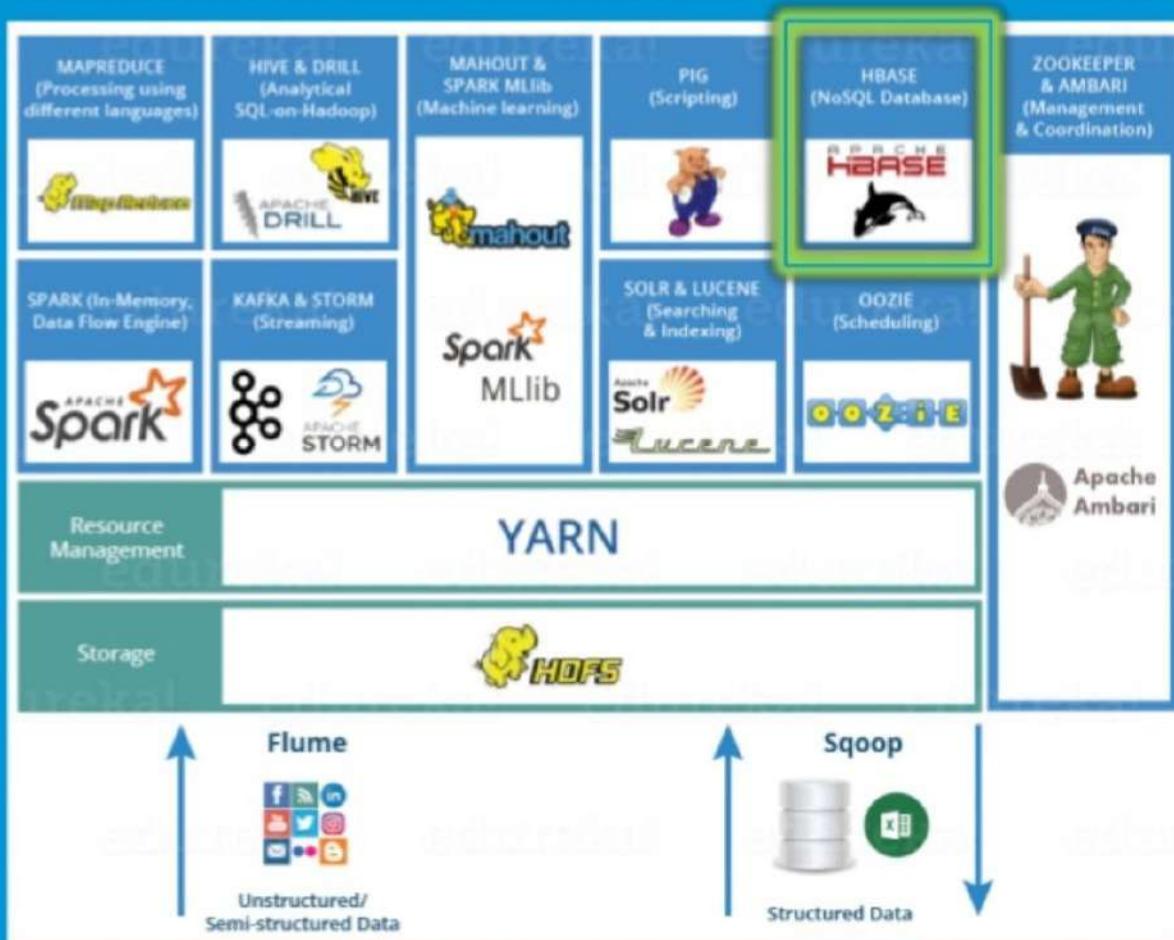
- A framework for real time data analytics in a distributed computing environment.
- Written in Scala and was originally developed at the University of California, Berkeley.
- It executes in-memory computations to increase speed of data processing over Map-Reduce.
- 100x faster than Hadoop for large scale data processing by exploiting in-memory computations

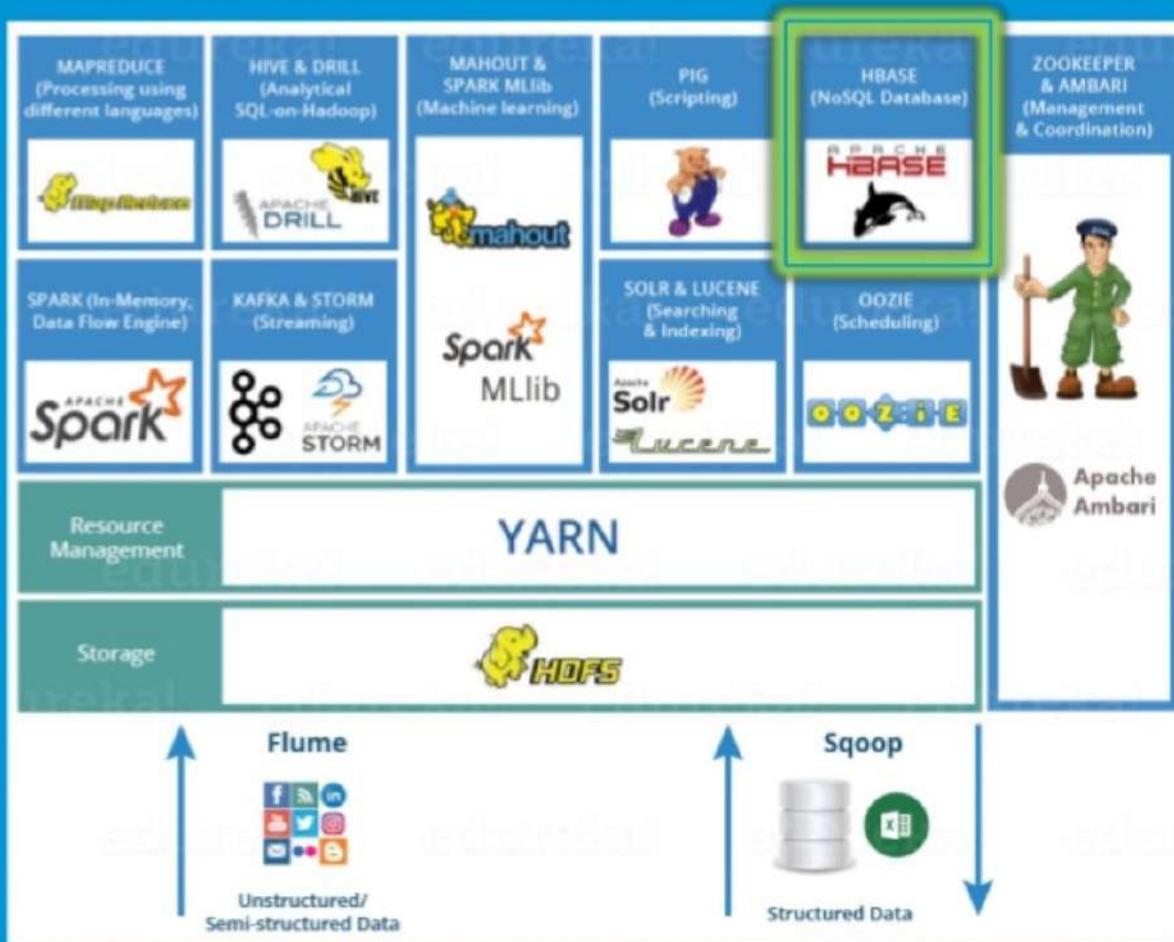


# Spark: In-memory Data Processing

- Spark comes packed with high-level libraries
- Provides various services like MLlib, GraphX, SQL + Data Frames, Streaming services
- Supports various languages like R, SQL, Python, Scala, Java
- Seamlessly integrates in complex workflow

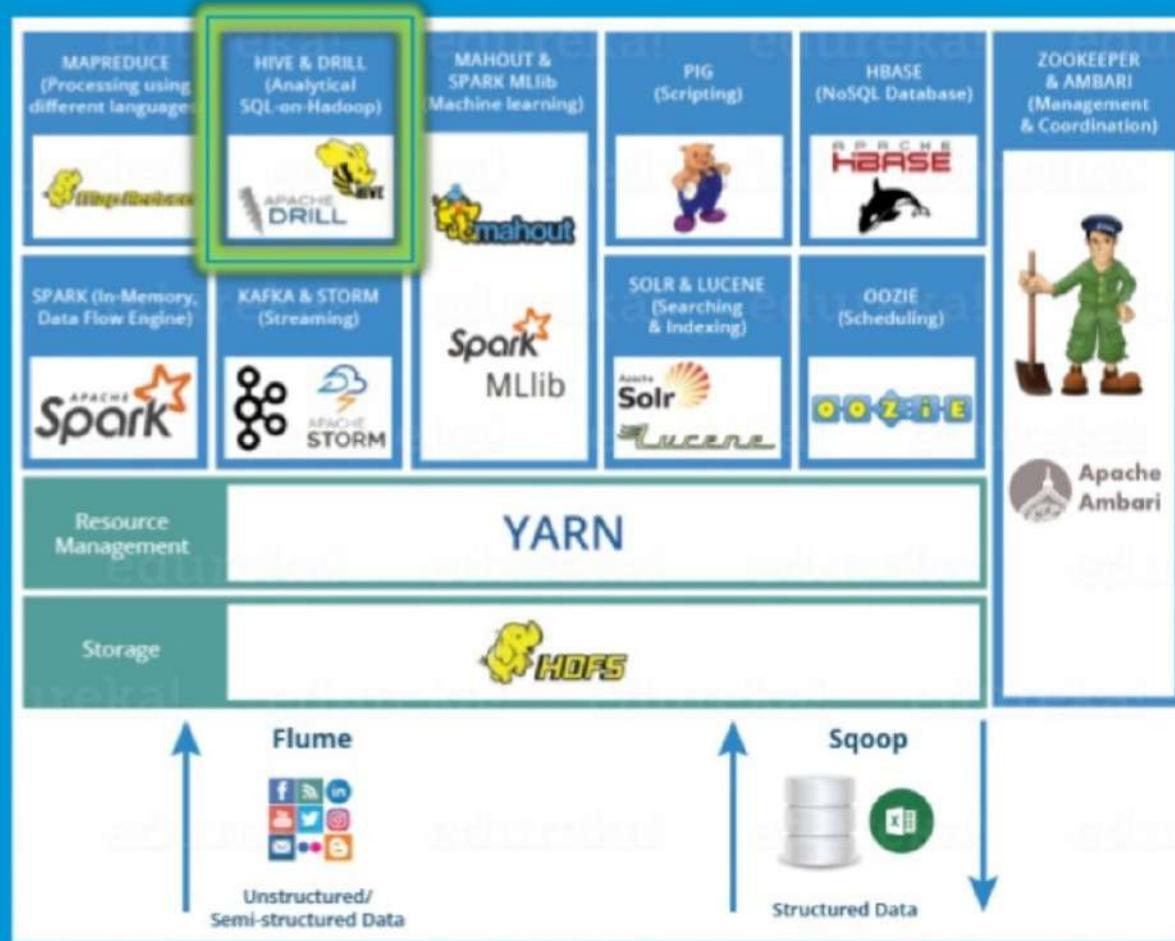






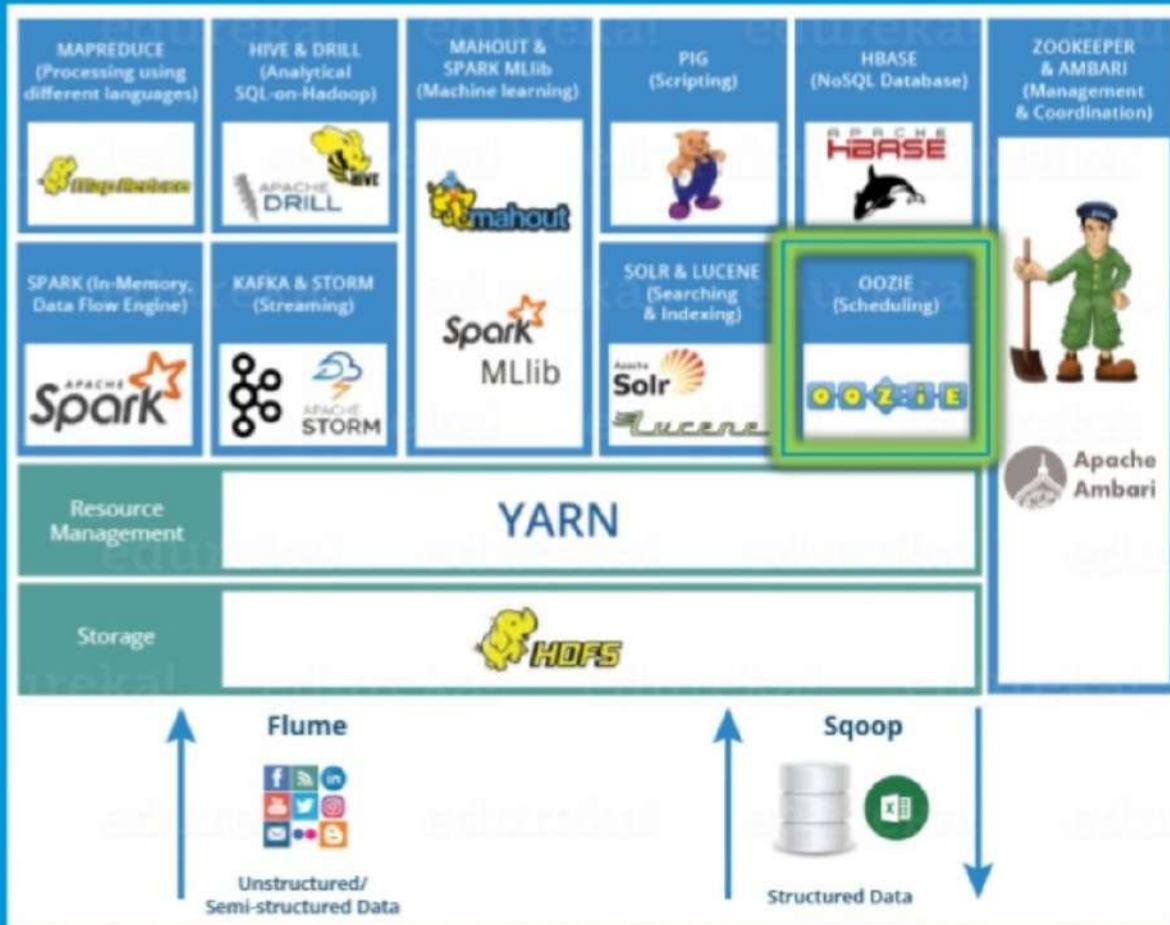


- An open source, non-relational distributed database  
- a **NoSQL** database
- Supports all types of data and that is why, it's capable of handling anything and everything
- It is modelled after Google's BigTable
- It gives us a fault tolerant way of storing sparse data
- It is written in Java, and HBase applications can be written in REST, Avro and Thrift APIs



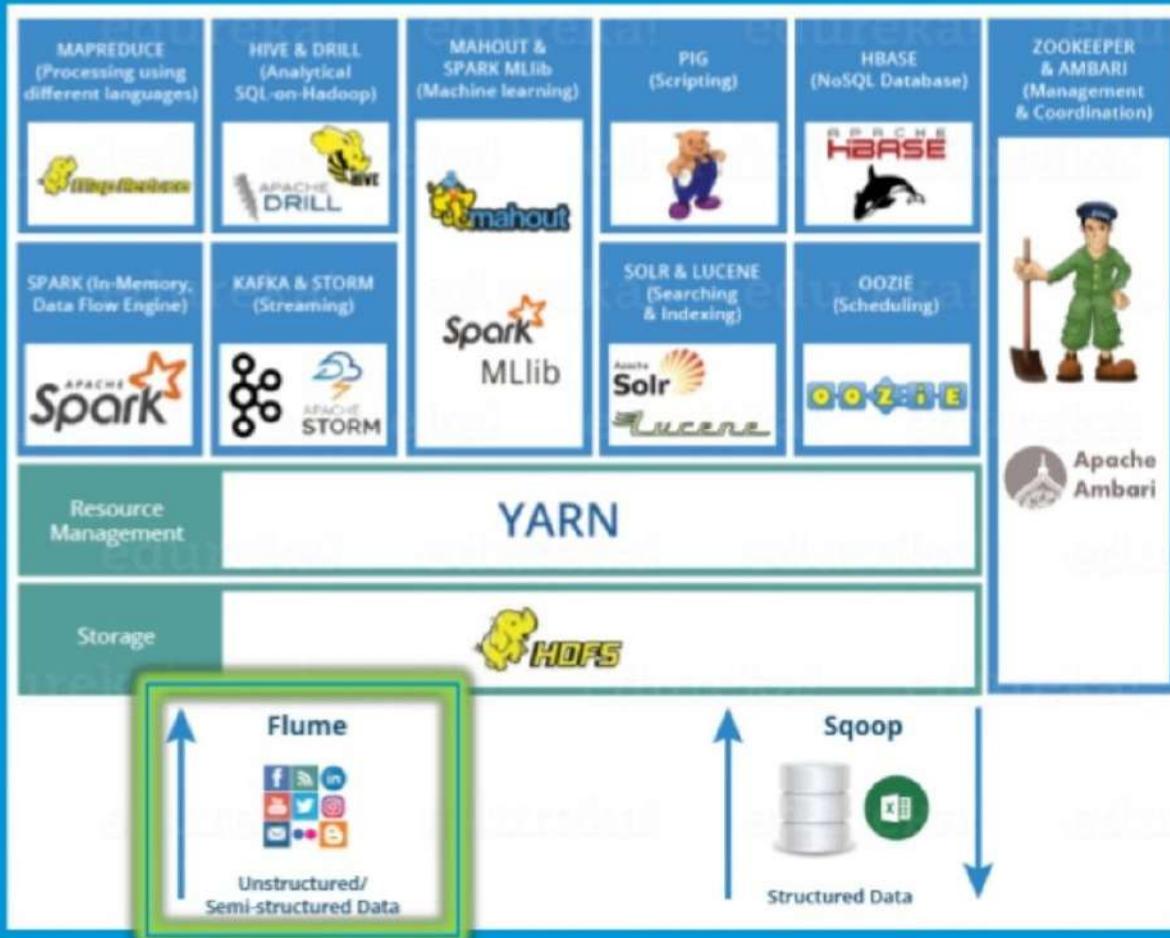
- An open source application which works with distributed environment to analyze large data sets
- Follows the ANSI SQL
- Supports different kinds NoSQL databases and file systems
- For example: Azure Blob Storage, Google Cloud Storage, HBase, MongoDB, MapR-DB HDFS, MapR-FS, Amazon S3, Swift, NAS and local files
- Combines a variety of data stores just by using a single query





- Oozie is a job scheduler in Hadoop ecosystem
- Two kinds of Oozie jobs: Oozie workflow and Oozie Coordinator
- **Oozie workflow:** Sequential set of actions to be executed
- **Oozie Coordinator:** Oozie jobs which are triggered when the data is made available to it or even triggered based on time



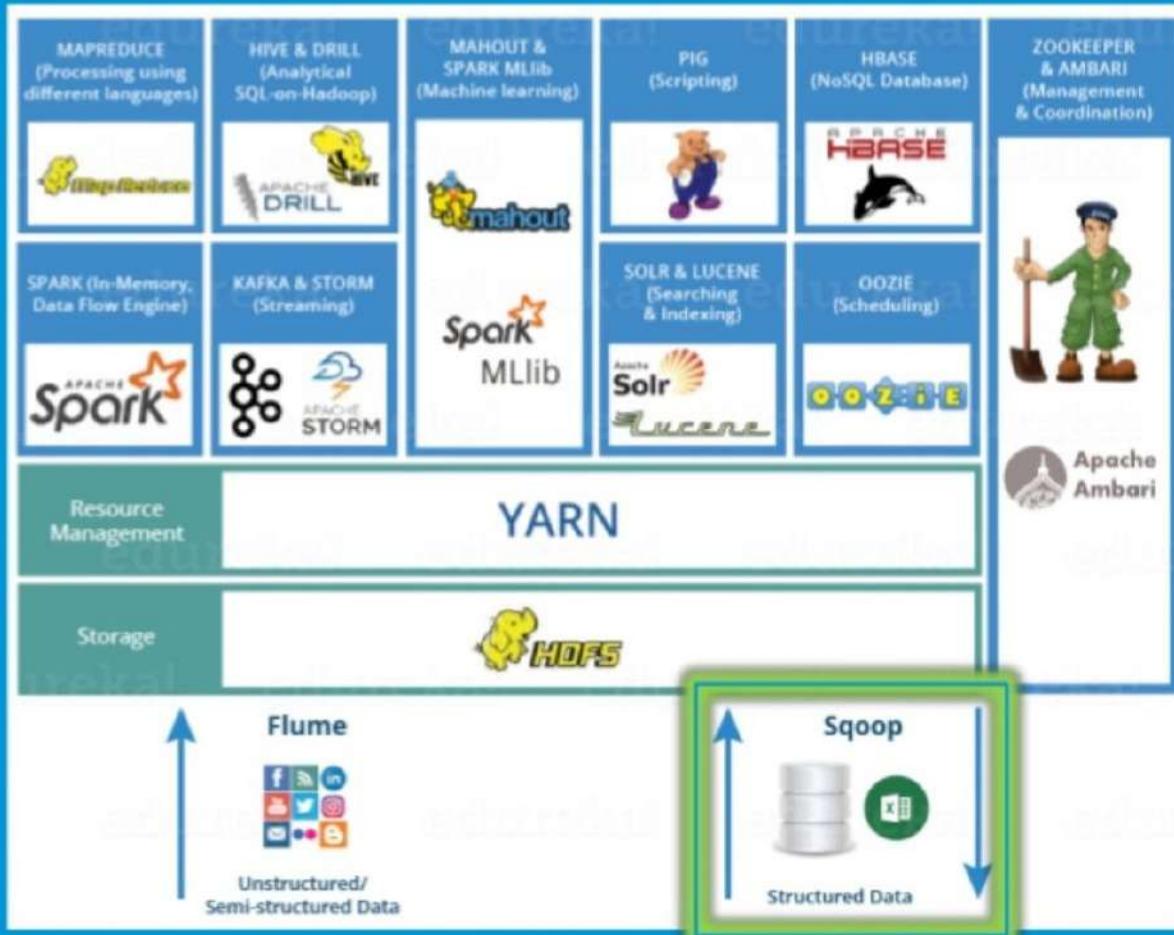


# Apache Flume: Data Ingesting Service

edureka!

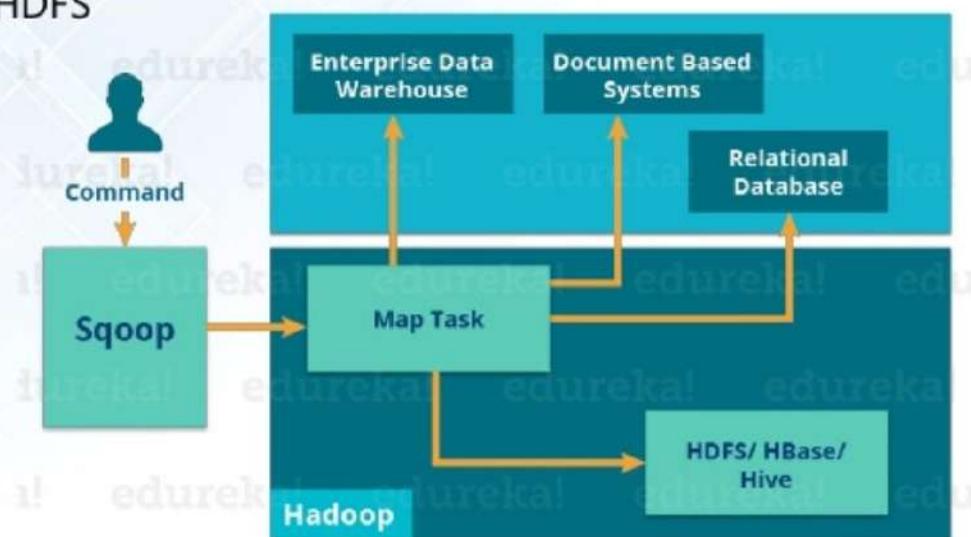
- Ingests unstructured and semi-structured data into HDFS.
- It helps us in collecting, aggregating and moving large amount of data sets.
- It helps us to ingest online streaming data from various sources like network traffic, social media, email messages, log files etc. in HDFS.

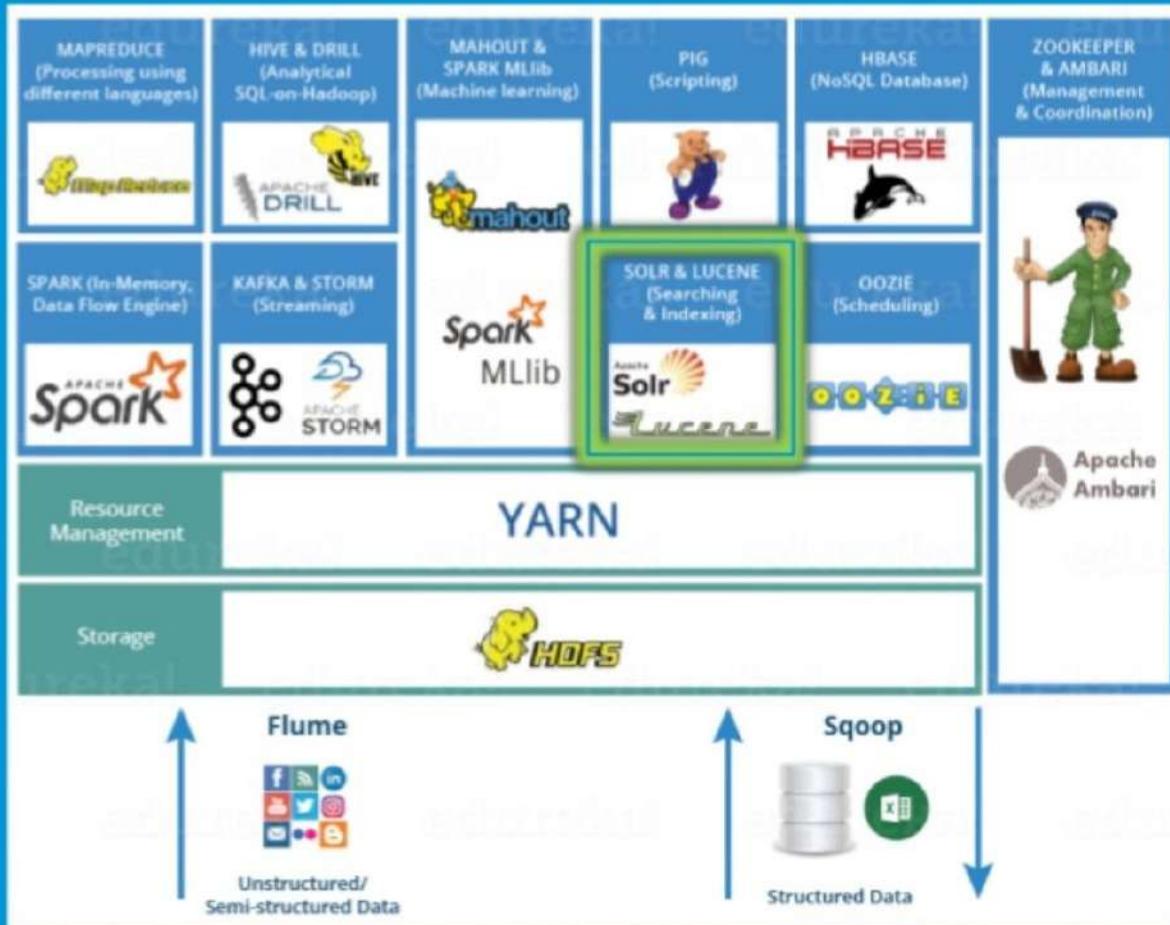




# Apache Sqoop: Data Ingesting Service

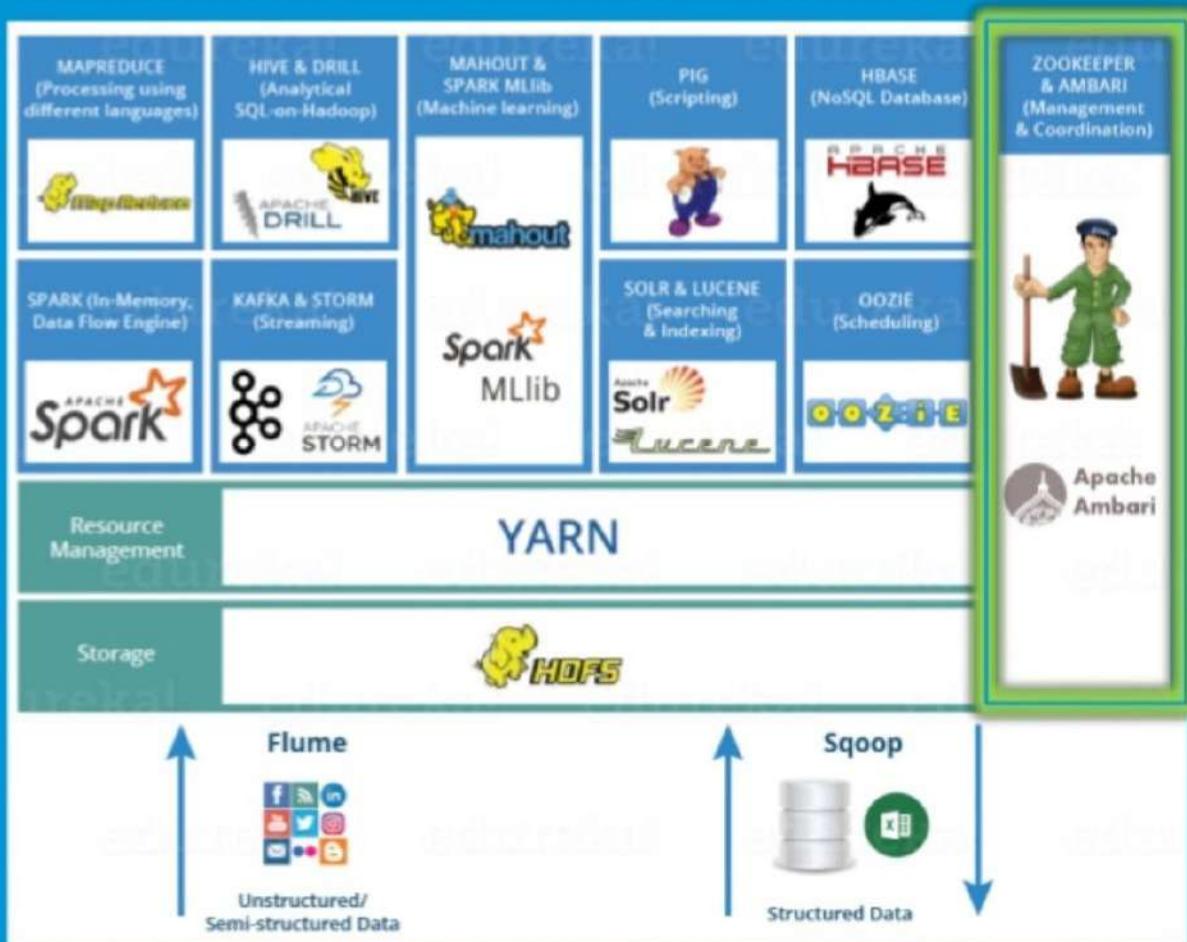
- Another data ingesting service
- Sqoop can import as well as export structured data from RDBMS
- Flume only ingests unstructured data or semi-structured data into HDFS





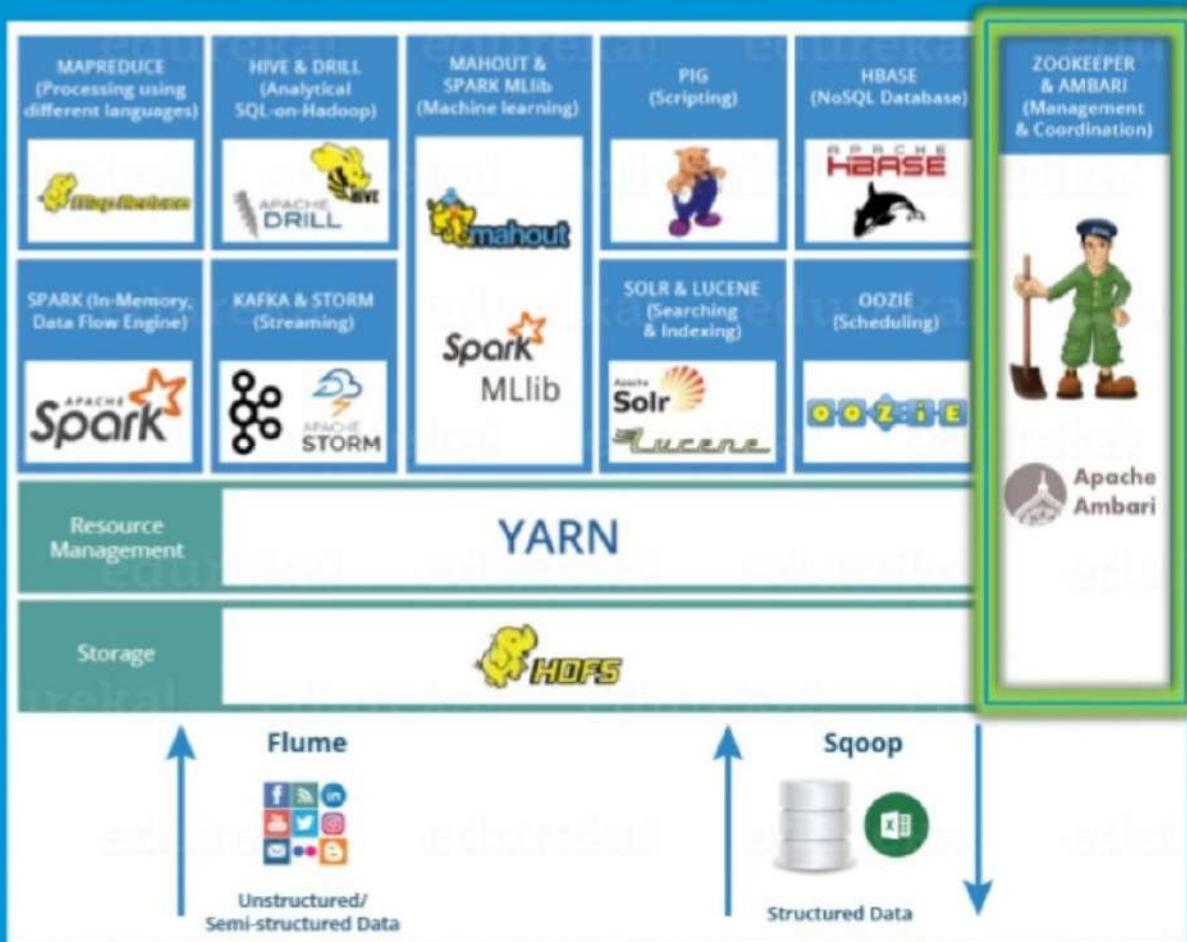
- Two services which are used for searching and indexing in Hadoop Ecosystem
- Apache Lucene is based on Java, which also helps in spell checking
- Apache Lucene is the engine, Apache Solr is a complete application built around Lucene
- Solr uses Apache Lucene Java search library for searching and indexing





- An open-source server which enables highly reliable distributed coordination
- Apache Zookeeper coordinates with various Hadoop services in a distributed environment
- Performs synchronization, configuration maintenance, grouping and naming





# Apache Ambari: Cluster Manager

edureka!

- Software for provisioning, managing and monitoring Apache Hadoop clusters
- Gives us step by step process for installing Hadoop services
- Handles configuration of Hadoop services
- Provides a central management service for starting, stopping and re-configuring Hadoop services
- Monitors health and status of the Hadoop cluster



Apache  
Ambari

# Apache Ambari: Cluster Manager

edureka!

- Software for provisioning, managing and monitoring Apache Hadoop clusters
- Gives us step by step process for installing Hadoop services
- Handles configuration of Hadoop services
- Provides a central management service for starting, stopping and re-configuring Hadoop services
- Monitors health and status of the Hadoop cluster



**Apache  
Ambari**

- **Hadoop Tutorial:** [www.edureka.co/blog/hadoop-tutorial](http://www.edureka.co/blog/hadoop-tutorial)
  
- **HDFS Ecosystem:** [www.edureka.co/blog/hadoop-ecosystem](http://www.edureka.co/blog/hadoop-ecosystem)



video

edureka!

# Thank You

For more information please visit our website  
[www.edureka.co](http://www.edureka.co)

# Using Big Data for Analytics

Basanta Joshi (PhD)  
Assistant Professor

Department of Electronics and Computer  
Engineering, Pulchowk Campus, Institute of  
Engineering (IOE), Pulchowk Campus, Lalitpur, Nepal  
[basanta@ioe.edu.np](mailto:basanta@ioe.edu.np) / [www.basantajoshi.com.np](http://www.basantajoshi.com.np)

# Types of Analytics

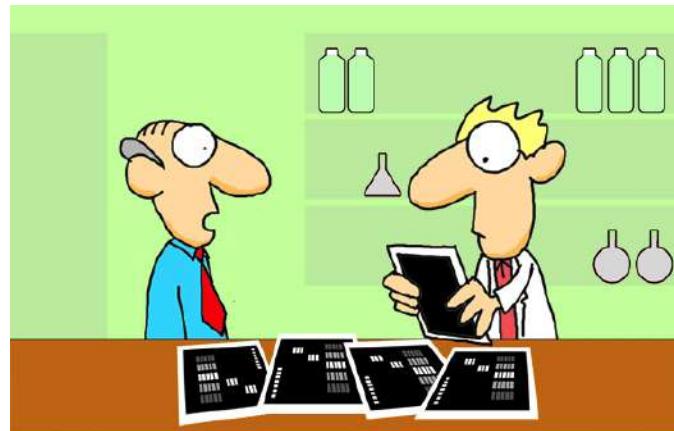
- **Descriptive**: A set of techniques for reviewing and examining the data set(s) to understand the data and analyze business performance.
- **Diagnostic**: A set of techniques for determine what has happened and why
- **Predictive**: A set of techniques that analyze current and historical data to determine what is most likely to (not) happen
- **Prescriptive**: A set of techniques for computationally developing and analyzing alternatives that can become courses of action – either tactical or strategic – that may discover the unexpected
- **Decisive**: A set of techniques for visualizing information and recommending courses of action to facilitate human decision-making when presented with a set of alternatives.

	Passive	Active	
Deductive	Descriptive		Diagnostic
Inductive	Predictive		Prescriptive

# Descriptive Analytics

- **Process:**
  - Identify the attributes, then assess/evaluate the attributes
  - Estimate the magnitude to correlate the relative contribution of each attribute to the final solution
  - Accumulate more instances of data from the data sources
  - If possible, perform the steps of evaluation, classification and categorization quickly
  - Yield a measure of adaptability within the OODA loop
- At some threshold, crossover into diagnostic and predictive analytics

<http://v1shal.com/content/25-cartoons-give-current-big-data-hype-perspective/>



"Data don't make any sense,  
we will have to resort to statistics."



*Copyright (except where referenced) 2014-2016*

*Stephen H. Kaisler, Frank Armour, Alberto Espinosa and William H. Money*

# Descriptive Analytics

NC

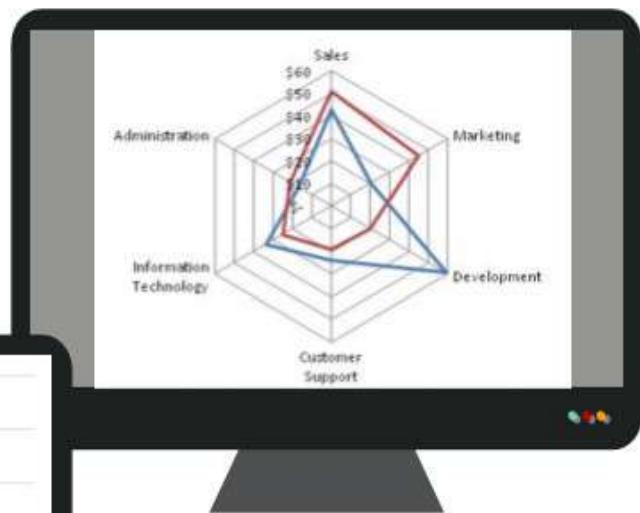
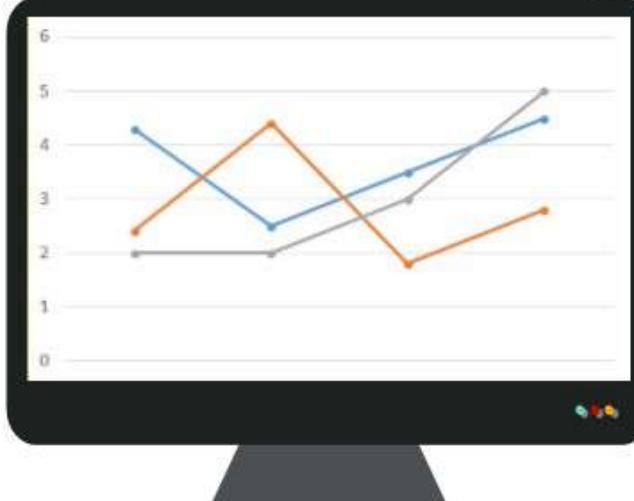
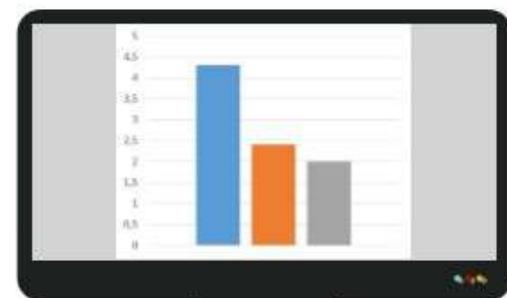
Descriptive analytics



Descriptive  
analytics

What is happening

Describe the  
data to get  
useful info



- KPIs
- Dashboards
- Visualizations

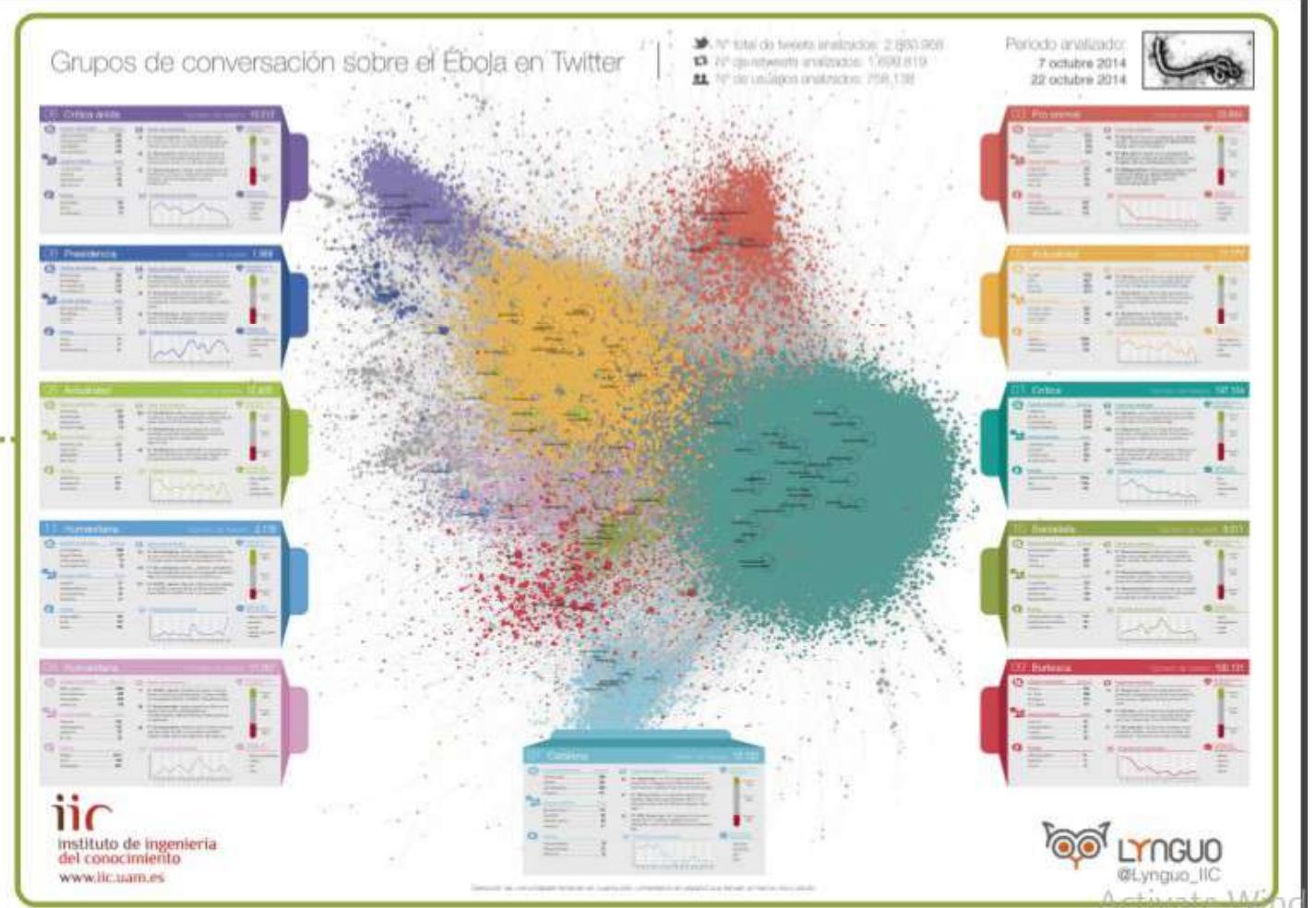
Activate Wind

# Descriptive Analytics



Descriptive analytics

What is happening



# Descriptive Analytics



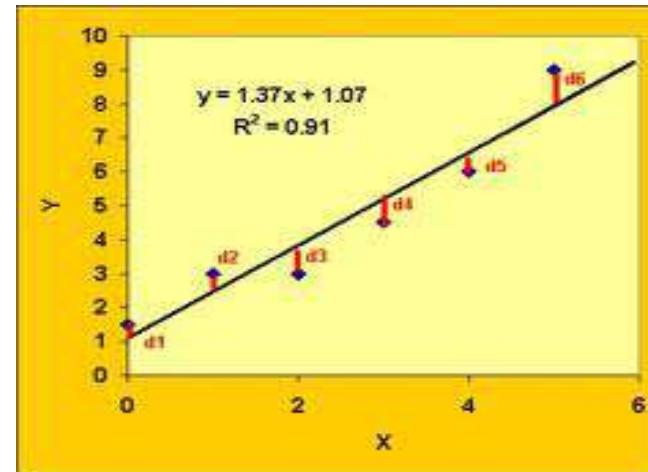
Descriptive  
analytics

What is happening



# Diagnostic Analytics

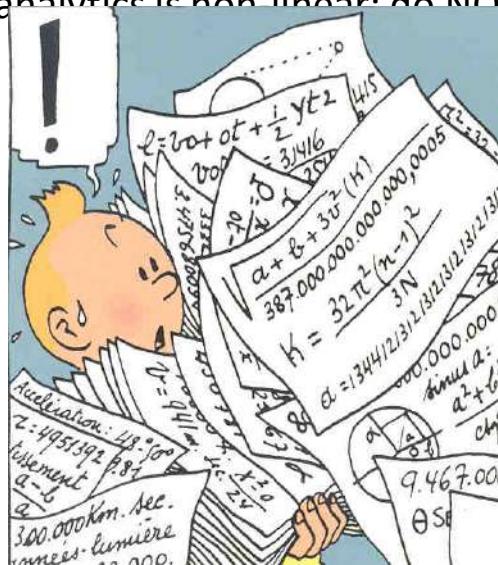
- Process:
  - Begin with descriptive analytics
  - Extract patterns from large data quantities via data mining
  - Correlate data types for explanation of near-term behavior – past and present
  - Estimate linear/non-linear behavior not easily identifiable through other approaches.
- Example: by classifying past insurance claims, estimate the number of future claims to flag for investigation with a high probability of being fraudulent.



# Predictive Analytics

- Process:

- Begin with descriptive AND diagnostic analytics
- Choose the right data based on domain knowledge and relationships among variables
- Choose the right techniques to yield insight into possible outcomes
- Determine the likelihood of possible outcomes given initial boundary conditions
- Remember! Data driven analytics is non-linear; do NOT treat like an engineering project



# Predictive Analytics



Source: SAS

## Predictive Analytics Process Cycle

# Common Predictive Analytics Methods

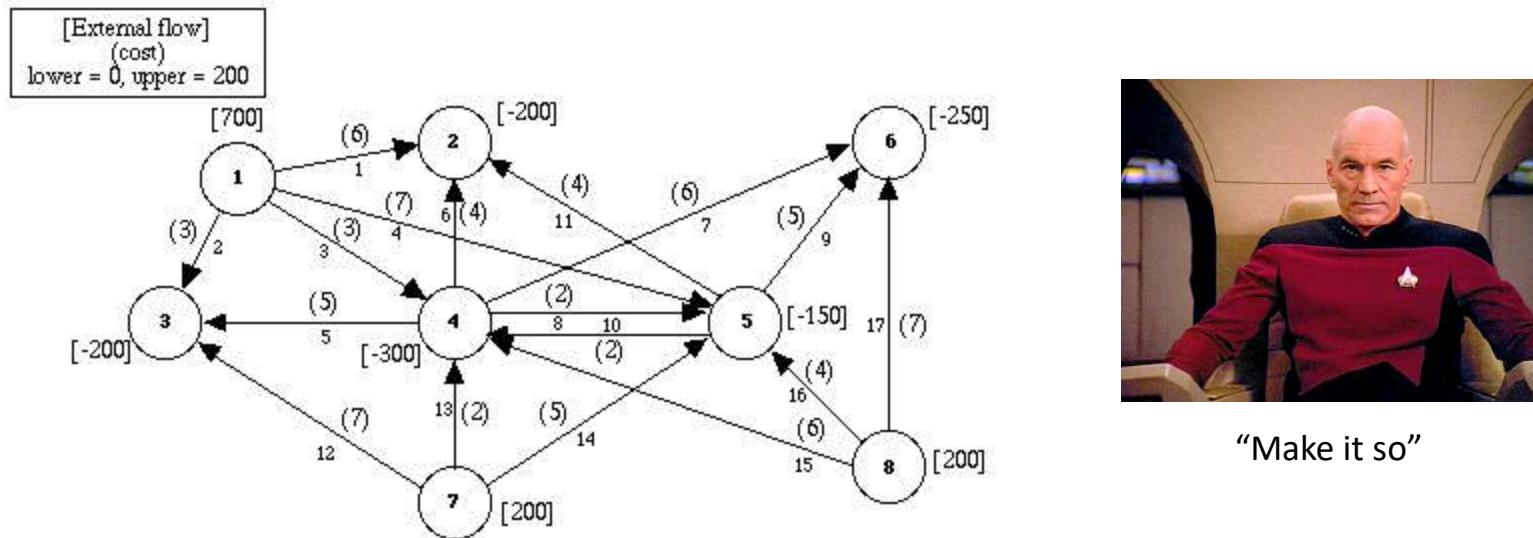
- **Regression:**
  - Predicting output variable using its cause-effect relationship with input variables. OLS Regression, GLM, Random forests, ANN etc.
- **Classification:**
  - Predicting the item class. Decision Tree, Logistic Regression, ANN, SVM, Naïve Bayes classifier etc.
- **Time Series Forecasting:**
  - Predicting future time events given past history. AR, MA, ARIMA, Triple Exponential Smoothing, Holt-Winters etc.

# Common Predictive Analytics Methods

- Association rule mining:
  - Mining items occurring together. Apriori Algorithm.
- Clustering:
  - Finding natural groups or clusters in the data. K-means, Hierarchical, Spectral, Density based EM algorithm Clustering etc.
- Text mining:
  - Model and structure the information content of textual sources. Sentiment Analysis, NLP

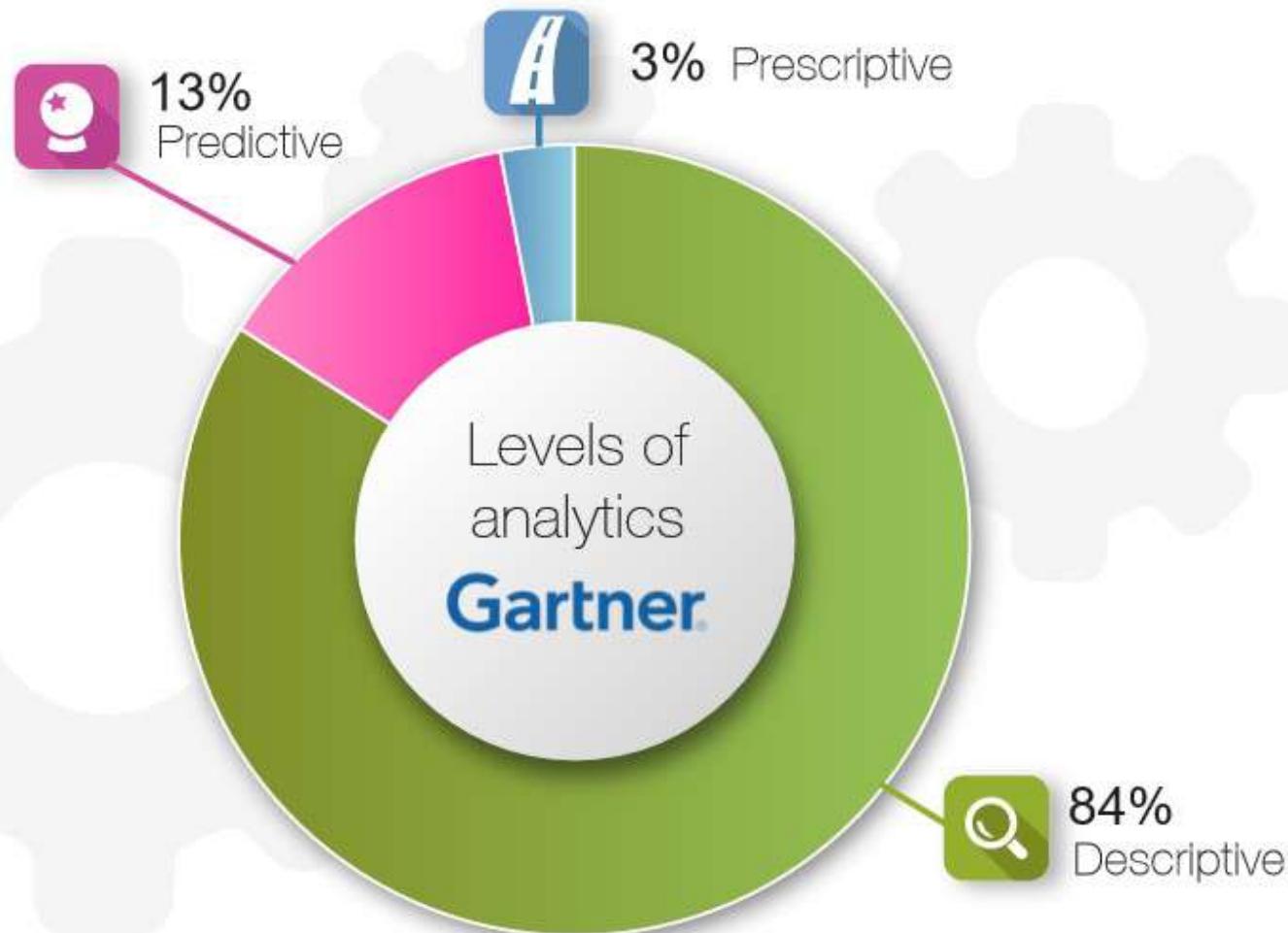
# Prescriptive Analytics

- **Process:**
  - Begin w/ predictive analytics
  - Determine what should occur and how to make it so
  - Determine the mitigating factors that lead to desirable/undesirable outcomes
  - “What-if” analysis w/ local or global optimization
  - Ex: Find the best set of prices and advertising frequency to maximize revenue
  - Ex: And, the right set of business moves to make to achieve that goal



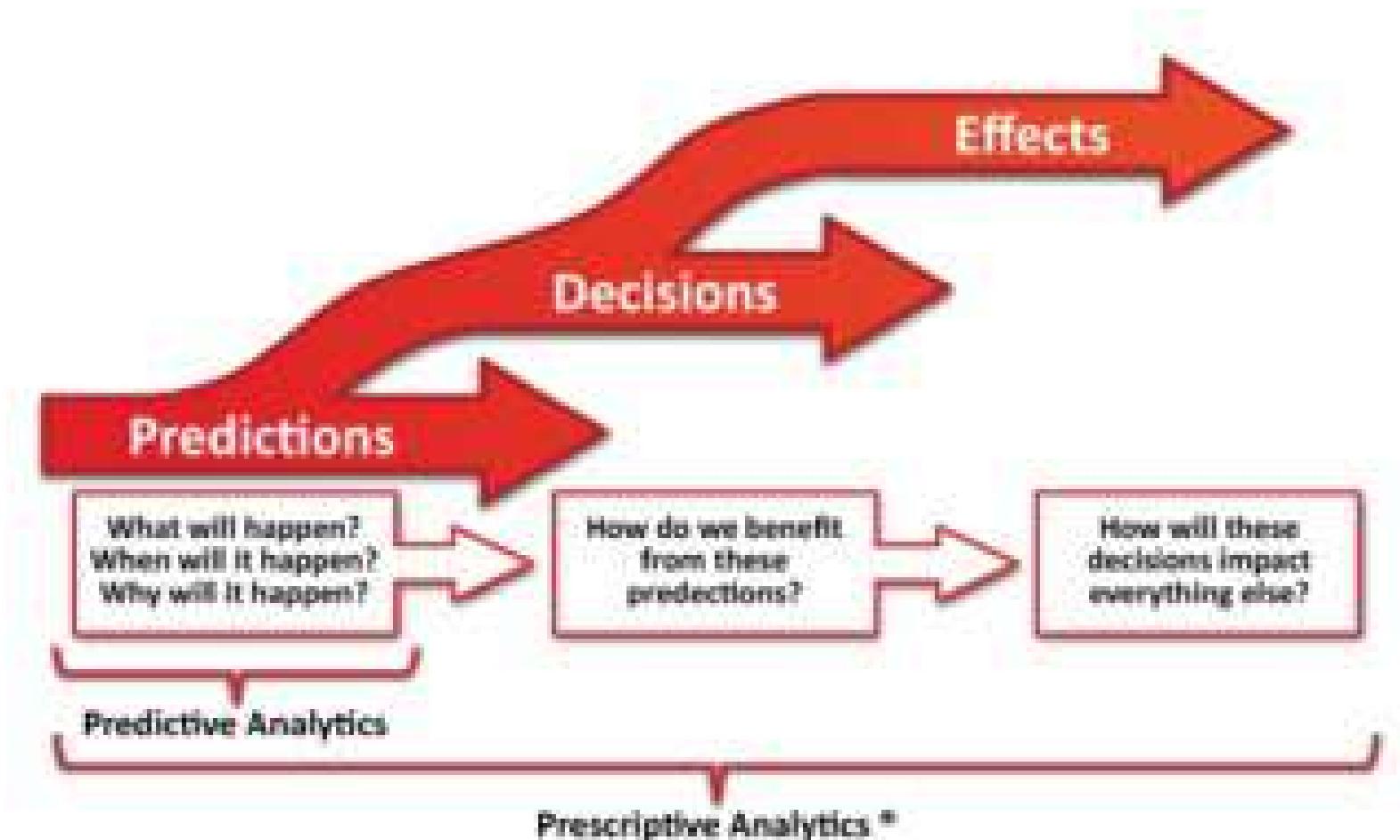
“Make it so”

# Analytics Status

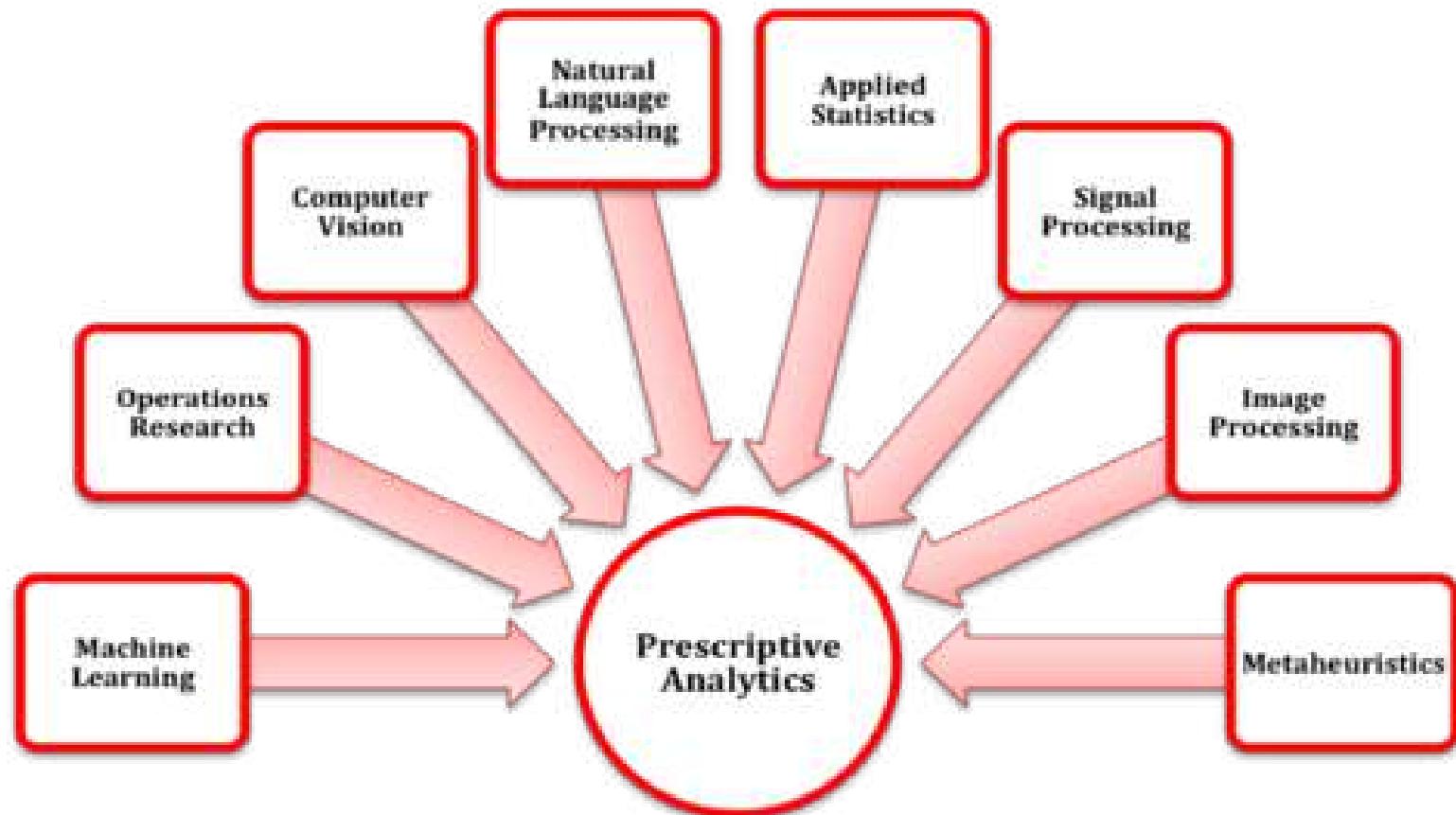


# Prescriptive Analytics

- Blend of descriptive and [predictive](#) analytics



# Prescriptive Analytics



- scientific disciplines that comprise Prescriptive Analytics

# Advanced Analytics

*Copyright (except where referenced) 2014-2016*

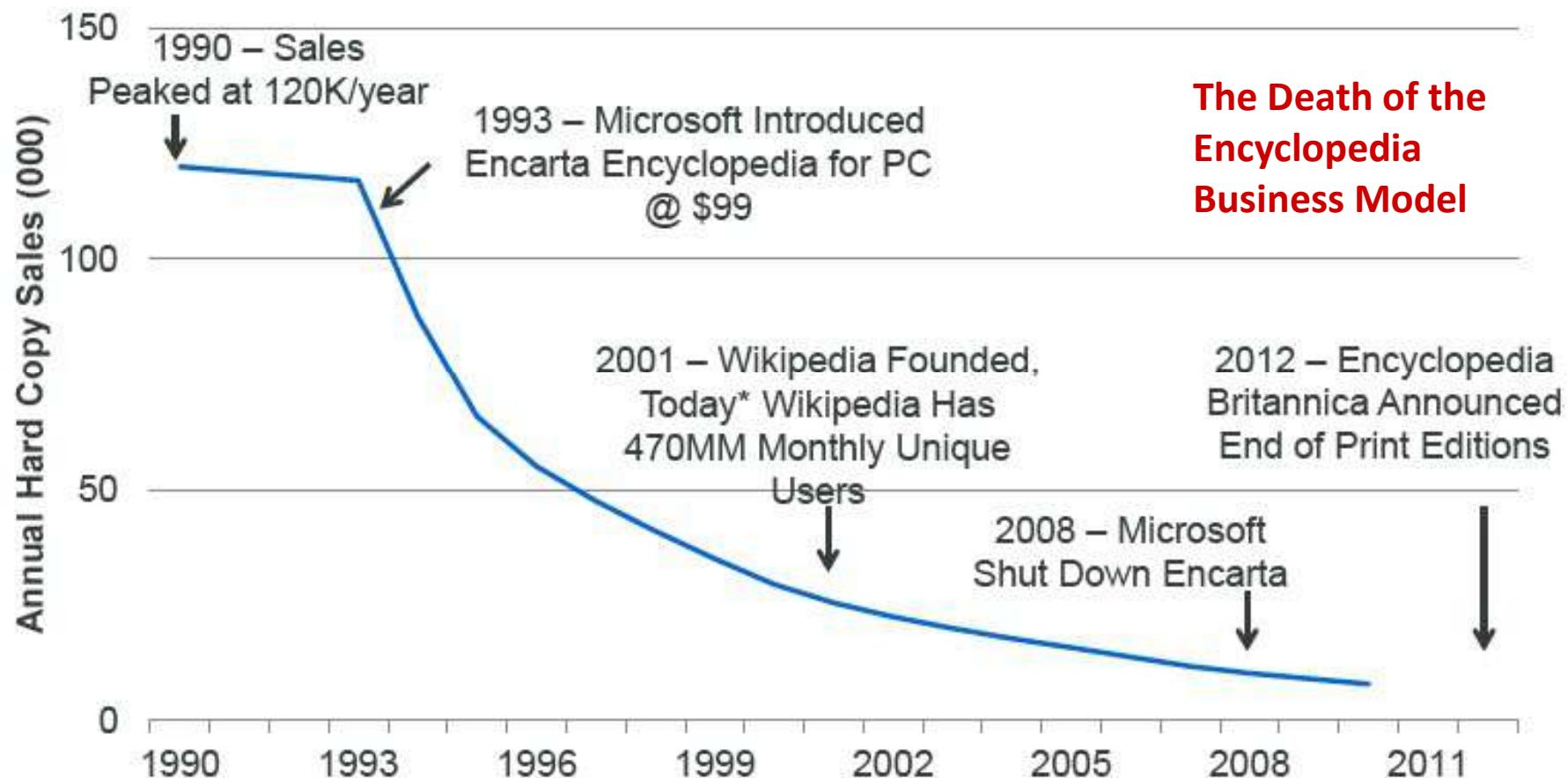
*Stephen H. Kaisler, Frank Armour, Alberto Espinosa and William H. Money*  
BDA-16

# Crowdsourcing: A New Analytic?

- Tasks normally performed by employees are outsourced via an *open call* to a large, self-selected community
- Some examples
  - Netflix prize
  - InnoCentive: solve R&D challenges
  - DARPA Network Challenge
- Follows an AI blackboard model
- Distributed co-creation has become a mainstream technology
- Issue: What metrics do we use to assess the results?
- Issue: How robust are the results?
- And, more issues to be addressed ....

## Crowdsourcing: Wikipedia: An Example

### Encyclopedia Britannica Hard Copy Sales, 1990 – 2011



Ref: Mary Meeker. KPCB, presentation at All Things D.

# Moving Analytics To the Edge

- Traditional analysis: data is stored and then analyzed
  - Usually at some central location or a few distributed locations
  - The cost and time to move large amounts of data may render it obsolete or of little worth
- Moving analytics to the frontier of the domain:
  - (Near) real-time analysis and decisions are required
  - Streaming massive amounts of data is expensive, fraught with error: microsecond latency, millions of events
  - We just can't store it all
  - Perishability is a key factor
  - May only really need the synthesized, aggregate information, not the raw data
  - True data-driven analysis
- Example: Pushing analytics into cameras for images, full motion video analysis, motion correction, 3D perception, ...

# Edge Device Analytics

- Filtering and compression processes are often tied to the downstream analytical requirements
  - Means the filtering and compression algorithms must be dynamic
- How close to the edge can we push the filtering and compression algorithms?
  - What (additional) data do these algorithms need to be effective?
  - How do we measure the efficiency of these algorithms?
  - Does the in situ hardware have the computational capacity to support such algorithms?
  - How much data correction can we do at the edges?
- Challenges:
  - How can we summarize streaming data?
  - How fast can we determine changes in the incoming data?
  - How fast can we adapt to changes in the data stream?
  - How fast can we affect the environment based on what we see?

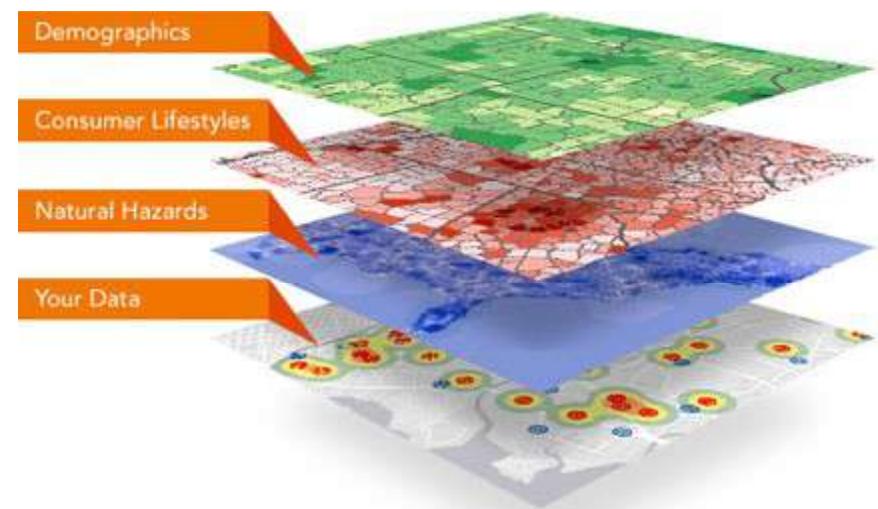
NOTE: Not the same as IBM Edge Analytics!!

# Streaming Analytics

- Streaming data: analytics must (often) occur in **real time**, as the data passes through the sensing/collecting device:
  - Allows you to identify and examine patterns of interest as the data is being created.
  - Can yield instant insight and immediate (re)action.
- **“Real-time is for robots”**
  - Joe Hellerstein, Professor of Computer Science at UC Berkeley.
  - “If you have people in the loop, it’s **not** real time. Most people take a **second or two** to react, and that’s plenty of time for a traditional transactional system to handle input and output.”

# Location Analytics

- What is it?
  - Augmenting mission-critical, enterprise business systems with complementary content, mapping, and geographic capabilities
  - Mapping & Visualization: use maps as the media to visualize data
  - Spatial analytics: merging GIS w/ other types of analytics
  - Find spatio-temporal patterns indicative of physical activities or social behavior
  - Data/information enrichment: add maps, imagery, demographics, consumer and lifestyle data, environment and weather, social media, etc.
- Ubiquity of GPS on cellphones, cars, wristwatches, laptops, tablets, etc.



Ref: <http://www.esri.com/software/location-analytics>

Ref: Kerr & Nelson, ESRI International User Conference, July 2012

# Location Analytics: After a Regime Change ...

- Immediate transition from “normal operations in an urban environment” to “government in place”
  - Need Civil Affairs support for managing city/region relief efforts
- No existing analysis, planning, and management tool to assist civil Government and Relief Officials
  - A lack of coherent view will lead to failure and, possibly, continuing deterioration of government, law and order, civil systems, etc...
- Why?
  - (nearly) complete destruction of governing regime
  - Disintegration of social/governmental institutions: water, education, health, law enforcement, financial, food
  - Society reorganized overnight to adapt to the new power structure, which may/may not include former government personnel
    - Ex: Sadr handed out food; created immediate constituency and had people talking about him
- Cannot win “hearts and minds” if you cannot “feed and house them”

# Web Analytics

- What is it?
  - Now: The study of the behavior of web users
  - Future: The study of one mechanism for how society makes decisions
  - Example: Behavior of Web Users
    - How many people clicked on Ebola (or related terms in the past 2 months)
    - Their location, their dwell time, the number of sites they examined, the difficulty or complexity of the material on the web site
    - What can this tell us about popular concern about Ebola?
    - Can it help decision makers to better present information and decisions
  - Commercially, it is the collection and analysis of data from a web site to determine which aspects of the website achieve the business objectives

# Web Analytics

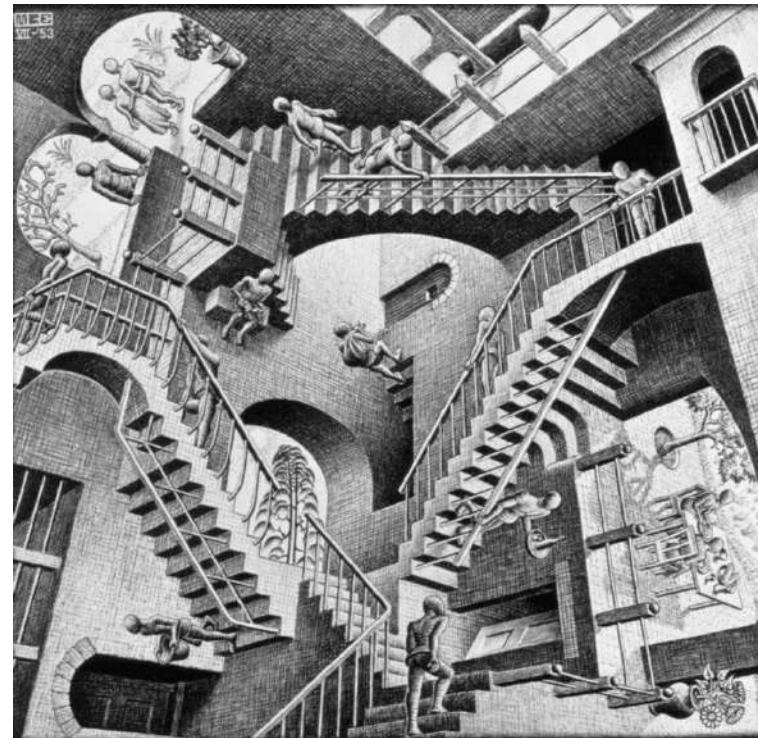
- Challenge: When people start getting more of their information from the Web than from radio/tv/papers:
  - How do we design web pages to influence the opinions of people?
    - The web is not a neutral medium
  - How do we measure the influence of a web page's contents on user opinions?
    - Does # visitors translate into a viable influence metric?
    - Does dwell time translate into influence?
    - Are opinions more or less influenced as the visitor clicks through a sequence of pages?
    - How many different web sites (each of one or more pages) does a user searching a particular topic click through?
  - How do we measure the difficulty and/or complexity of the material presented on a web page(s)?
    - What is the corollary to the Flesch-Kincaid Score for web pages?
  - How do we design “good” web pages to increase their Google Page Rank score?

# Visual Analytics

- *Visual analytics*: the science of analytical reasoning facilitated by interactive visual interfaces

Where are the  
walkers going?

Remember: The  
eye can be  
easily fooled  
and, thereby, fo  
ol the brain.



# Visual Analytics

- *Visual analytics*: an evolving discipline which is driving new ways of presenting data and information to the user.
- *Visual analytics*: “the science of analytical reasoning facilitated by interactive visual interfaces” (Thomas and Cook 2005)
- Visual analytics:
  - the formation of visual metaphors in combination with a human information discourse (interaction)
  - that enables detection of the expected and discovery of the unexpected within massive, dynamically changing information spaces. (Wong and Thomas 2004)
- Visual analytics provides the “last 12 inches” between the masses of information and the human mind that enables us to make decisions.

# Visual Analytics

- Why is it hard?
  - You can only see 2D because your screen is 2D
- To visualize k-dimensional data:
  - Divide the screen into multiple 2D regions and show pair-wise correlations across selected dimensions
  - Project k-dimensional data into 2D
- Projection Methods (Dimension Reduction):
  - PCA, MDS, LDA, LLE, many others ...
- Many others! Usually, try to preserve distances in 2D as they exist in k-D

# Visualization Analytics:

> ☼ < <b>C</b> continuum	Data Visualization Visual representations of quantitative data in schematic form (either with or without axes)												Strategy Visualization The systematic use of complementary visual representations in the analysis, development, formulation, communication, and implementation of strategies in organizations.												> ☼ < <b>G</b> graphic facilitation
> ☼ < <b>Tb</b> table	> ☼ < <b>Ca</b> cartesian coordinates	> ☼ < <b>R</b> radar chart	> ☼ < <b>Pa</b> parallel coordinates	> ☼ < <b>Hy</b> hyperbolic tree	> ☼ < <b>Cy</b> cycle diagram	> ☼ < <b>T</b> timeline	> ☼ < <b>Ve</b> venn diagram	< ☼ > <b>Mi</b> mindmap	< ☼ > <b>Sq</b> square of oppositions	> ☼ < <b>Cc</b> concentric circles	> ☼ < <b>Ar</b> argument slide	> ☼ < <b>Sw</b> swim lane diagram	> ☼ < <b>Gc</b> gantt chart	< ☼ > <b>Pm</b> perspectives diagram	> ☼ < <b>D</b> dilemma diagram	< ☼ > <b>Pr</b> parameter ruler	> ☼ < <b>Kn</b> knowledge map								
> ☼ < <b>Pi</b> pie chart	> ☼ < <b>L</b> line chart	> ☼ < <b>Ac</b> area chart	> ☼ < <b>Sc</b> scatterplot	> ☼ < <b>Sa</b> sankey diagram	> ☼ < <b>In</b> information lense	> ☼ < <b>E</b> entity relationship diagram	> ☼ < <b>Pt</b> petri net	> ☼ < <b>Fl</b> flow chart	< ☼ > <b>Cl</b> clustering	> ☼ < <b>Lc</b> layer chart	> ☼ < <b>Py</b> minto pyramid technique	> ☼ < <b>Ce</b> cause-effect chains	> ☼ < <b>Tl</b> toulmin map	> ☼ < <b>Dt</b> decision tree	> ☼ < <b>Cp</b> cpm critical path method	< ☼ > <b>Cf</b> concept fan	> ☼ < <b>Co</b> concept map	> ☼ < <b>Ic</b> iceberg	> ☼ < <b>Lm</b> learning map						
> ☼ < <b>Hi</b> histogram	> ☼ < <b>Tk</b> tukey box plot	> ☼ < <b>Sp</b> spectrogram	> ☼ < <b>Da</b> data map	> ☼ < <b>Tp</b> treemap	> ☼ < <b>Cn</b> cone tree	> ☼ < <b>Sy</b> system dyn./ simulation	> ☼ < <b>Df</b> data flow diagram	< ☼ > <b>Se</b> semantic network	> ☼ < <b>So</b> soft system modeling	< ☼ > <b>Sn</b> synergy map	< ☼ > <b>Fo</b> force field diagram	> ☼ < <b>Ib</b> ibis argumentation map	> ☼ < <b>Pr</b> process event chains	> ☼ < <b>Pe</b> pert chart	< ☼ > <b>Ev</b> evocative knowledge map	> ☼ < <b>V</b> Vee diagram	< ☼ > <b>Hh</b> heavens 'n' hell chart	> ☼ < <b>I</b> infomural							

## Process Visualization

Note: Depending on your location and connection speed it can take some time to load a pop-up picture.

version 1.5

© Ralph Lengler & Martin J. Eppler, [www.visual-literacy.org](http://www.visual-literacy.org)

## Structure Visualization

- Overview Detail
- Detail AND Overview
- < > Divergent thinking
- > < Convergent thinking

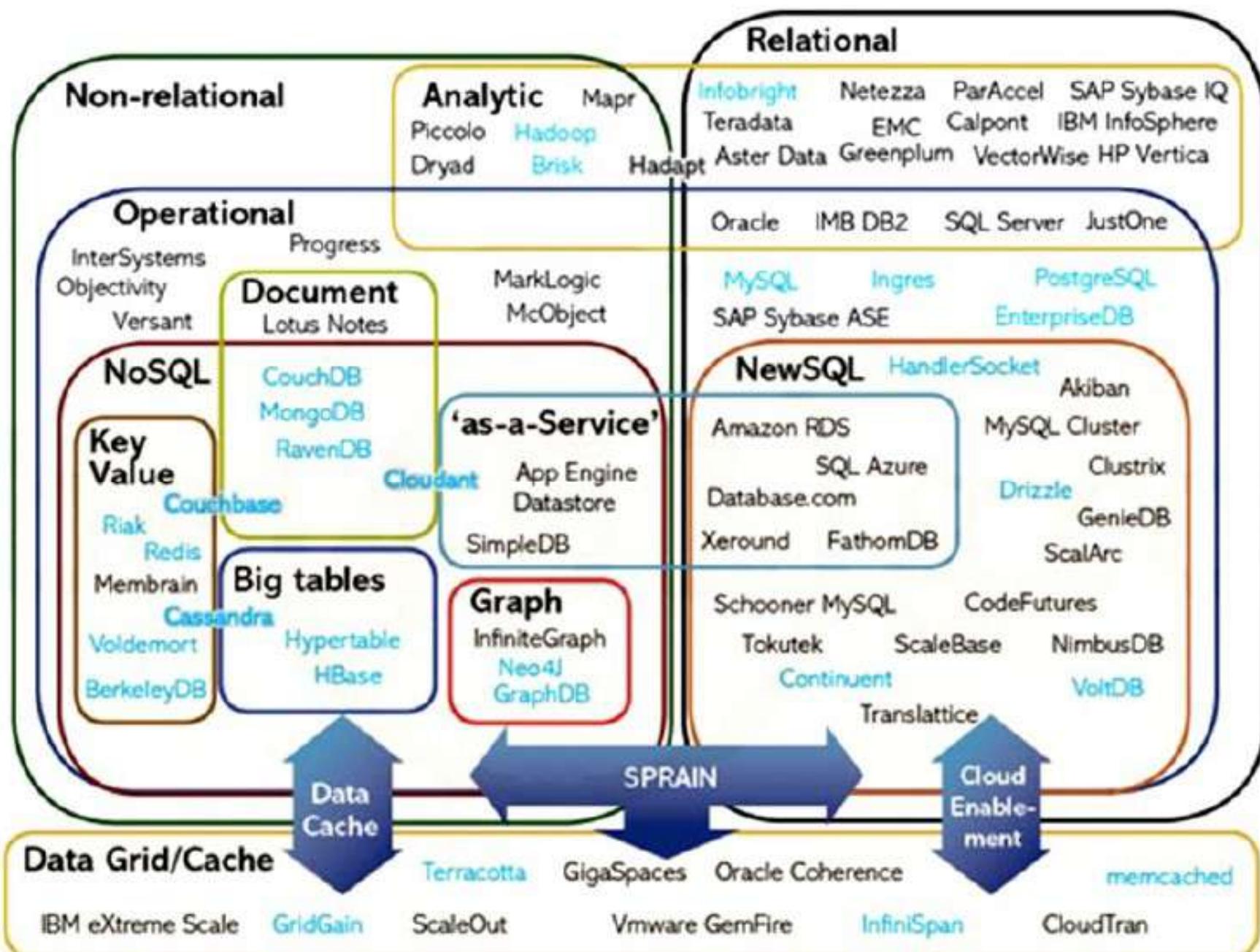
> ☼ < <b>Su</b> supply demand curve	> ☼ < <b>Pc</b> performance charting	> ☼ < <b>St</b> strategy map	> ☼ < <b>Oc</b> organisation chart	< ☼ > <b>Ho</b> house of quality	> ☼ < <b>Fd</b> feedback diagram	▫ <b>Ft</b> failure tree	> ☼ < <b>Mq</b> magic quadrant	> ☼ < <b>Ld</b> life-cycle diagram	> ☼ < <b>Po</b> porter's five forces	< ☼ > <b>S</b> s-cycle	> ☼ < <b>Sm</b> stakeholder map	& <b>Is</b> ishikawa diagram	> ☼ < <b>Tc</b> technology roadmap
& <b>Ed</b> edgeworth box	> ☼ < <b>Pf</b> portfolio diagram	& <b>Sg</b> strategic game board	> ☼ < <b>Mz</b> mintzberg's organigraph	< ☼ > <b>Z</b> zwicky's morphological box	< ☼ > <b>Ad</b> affinity diagram	▫ <b>De</b> decision discovery diagram	> ☼ < <b>Bm</b> bcg matrix	> ☼ < <b>Stc</b> strategy canvas	> ☼ < <b>Vc</b> value chain	< ☼ > <b>Hy</b> hype-cycle	< ☼ > <b>Sr</b> stakeholder rating map	> ☼ < <b>Ta</b> taps	< ☼ > <b>Sd</b> spray diagram

Copyright (except where referenced) 2014-2016

Stephen H. Kaisler, Frank Armour, Alberto Espinosa and William H. Money

# **Basic approaches to querying and exploring data using higher level tools built on top of a Hadoop Platform**

Adopted from slides by Perry Hoekstra, Jiaheng Lu, Avinash Lakshman, Prashant Malik, and Jimmy Lin



# History of the World, Part 1

- Relational Databases – mainstay of business
- Web-based applications caused spikes
  - Especially true for public-facing e-Commerce sites
- Developers begin to front RDBMS with memcache or integrate other caching mechanisms within the application (ie. Ehcache)

# Scaling Up

- Issues with scaling up when the dataset is just too big
- RDBMS were not designed to be distributed
- Began to look at multi-node database solutions
- Known as ‘scaling out’ or ‘horizontal scaling’
- Different approaches include:
  - Master-slave
  - Sharding

# Scaling RDBMS – Master/Slave

- Master-Slave
  - All writes are written to the master. All reads performed against the replicated slave databases
  - Critical reads may be incorrect as writes may not have been propagated down
  - Large data sets can pose problems as master needs to duplicate data to slaves

# Scaling RDBMS - Sharding

- Partition or sharding
  - Scales well for both reads and writes
  - Not transparent, application needs to be partition-aware
  - Can no longer have relationships/joins across partitions
  - Loss of referential integrity across shards

# Other ways to scale RDBMS

- Multi-Master replication
- INSERT only, not UPDATES/DELETES
- No JOINs, thereby reducing query time
  - This involves de-normalizing data
- In-memory databases

# What is NoSQL?

- Stands for **Not Only SQL**
- Class of non-relational data storage systems
- Usually do not require a fixed table schema nor do they use the concept of joins
- All NoSQL offerings relax one or more of the ACID properties (will talk about the CAP theorem)

# Why NoSQL?

- For data storage, an RDBMS cannot be the be-all/end-all
- Just as there are different programming languages, need to have other data storage tools in the toolbox
- A NoSQL solution is more acceptable to a client now than even a year ago
  - Think about proposing a Ruby/Rails or Groovy/Grails solution now versus a couple of years ago

# How did we get here?

- Explosion of social media sites (Facebook, Twitter) with large data needs
- Rise of cloud-based solutions such as Amazon S3 (simple storage solution)
- Just as moving to dynamically-typed languages (Ruby/Groovy), a shift to dynamically-typed data with frequent schema changes
- Open-source community

# Dynamo and BigTable

- Three major papers were the seeds of the NoSQL movement
  - BigTable (Google)
  - Dynamo (Amazon)
    - Gossip protocol (discovery and error detection)
    - Distributed key-value data store
    - Eventual consistency
  - CAP Theorem (discuss in a sec ..)

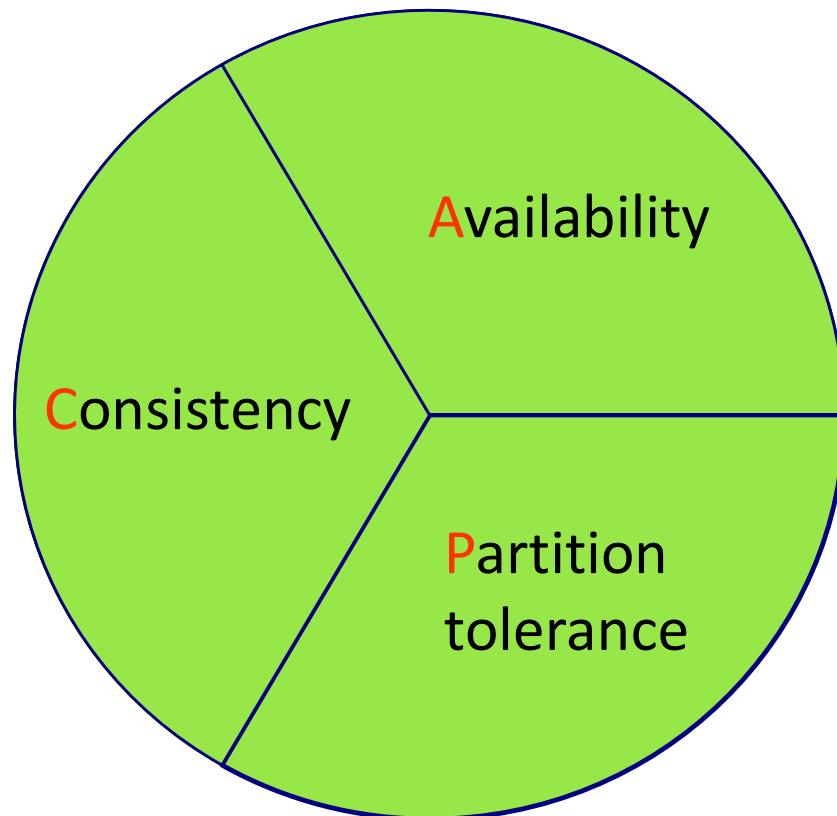
# The Perfect Storm

- Large datasets, acceptance of alternatives, and dynamically-typed data has come together in a perfect storm
- Not a backlash/rebellion against RDBMS
- SQL is a rich query language that cannot be rivaled by the current list of NoSQL offerings

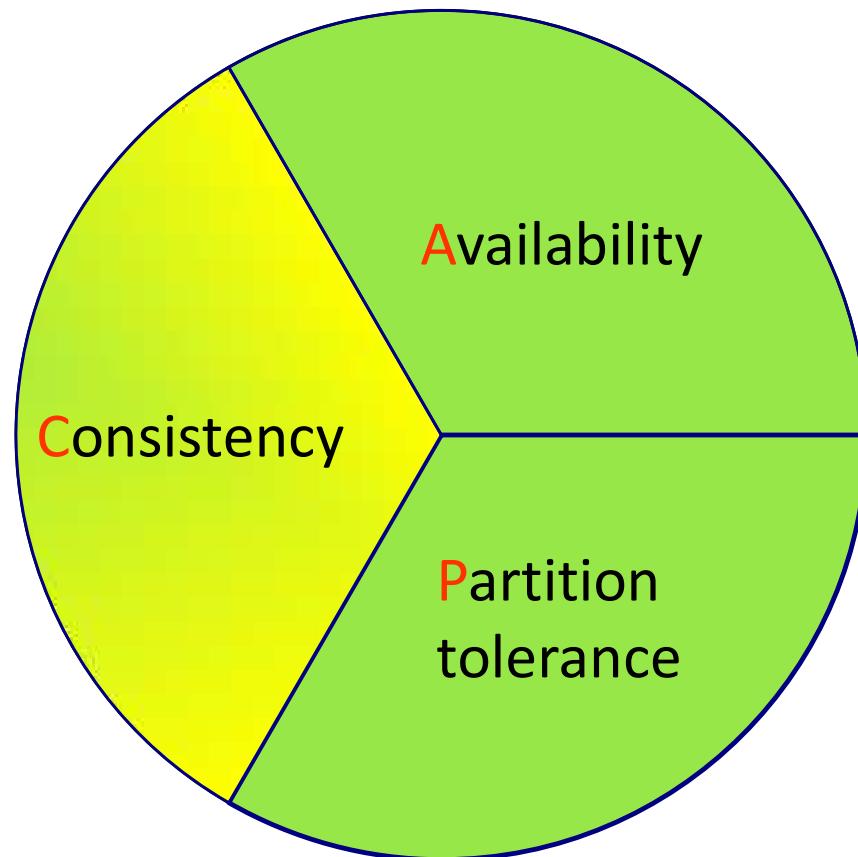
# CAP Theorem

- Three properties of a system: consistency, availability and partitions
- You can have at most two of these three properties for any shared-data system
- To scale out, you have to partition. That leaves either consistency or availability to choose from
  - In almost all cases, you would choose availability over consistency

# The CAP Theorem



# The CAP Theorem



Once a writer has written, all readers will see that write

# Consistency

- Two kinds of consistency:
  - strong consistency – ACID(Atomicity Consistency Isolation Durability)
  - weak consistency – BASE(Basically Available Soft-state Eventual consistency )

# ACID Transactions

- A DBMS is expected to support “*ACID transactions*,” processes that are:
  - *Atomic* : Either the whole process is done or none is.
  - *Consistent* : Database constraints are preserved.
  - *Isolated* : It appears to the user as if only one process executes at a time.
  - *Durable* : Effects of a process do not get lost if the system crashes.

# Atomicity

- A real-world event either happens or does not happen
  - Student either registers or does not register
- Similarly, the system must ensure that either the corresponding transaction runs to completion or, if not, it has no effect at all
  - Not true of ordinary programs. A crash could leave files partially updated on recovery

# Commit and Abort

- If the transaction successfully completes it is said to **commit**
  - The system is responsible for ensuring that all changes to the database have been saved
- If the transaction does not successfully complete, it is said to **abort**
  - The system is responsible for undoing, or rolling back, all changes the transaction has made

# Database Consistency

- Enterprise (Business) Rules limit the occurrence of certain real-world events
  - Student cannot register for a course if the current number of registrants equals the maximum allowed
- Correspondingly, allowable database states are restricted
  - $cur\_reg \leq max\_reg$
- These limitations are called (static) integrity constraints: assertions that must be satisfied by all database states (state invariants).

# Database Consistency

## (state invariants)

- Other static consistency requirements are related to the fact that the database might store the same information in different ways
  - $cur\_reg = |list\_of\_registered\_students|$
  - Such limitations are also expressed as integrity constraints
- Database is consistent if all static integrity constraints are satisfied

# Transaction Consistency

- A consistent database state does not necessarily model the actual state of the enterprise
  - A deposit transaction that increments the balance by the wrong amount maintains the integrity constraint  $balance \geq 0$ , but does not maintain the relation between the enterprise and database states
- A consistent transaction maintains database consistency and the correspondence between the database state and the enterprise state (implements its specification)
  - Specification of deposit transaction includes  
 $balance' = balance + amt\_deposit$   
,
  - ( $balance'$  is the next value of  $balance$ )

# Dynamic Integrity Constraints

(transition invariants)

- Some constraints restrict allowable state transitions
  - A transaction might transform the database from one consistent state to another, but the transition might not be permissible
  - Example: A letter grade in a course (A, B, C, D, F) cannot be changed to an incomplete (I)
- Dynamic constraints cannot be checked by examining the database state

# Transaction Consistency

- **Consistent transaction:** if DB is in **consistent state initially**, when the **transaction completes**:
  - **All static integrity constraints are satisfied** (but constraints might be violated in intermediate states)
    - Can be checked by examining snapshot of database
  - **New state satisfies specifications of transaction**
    - Cannot be checked from database snapshot
  - **No dynamic constraints have been violated**
    - Cannot be checked from database snapshot

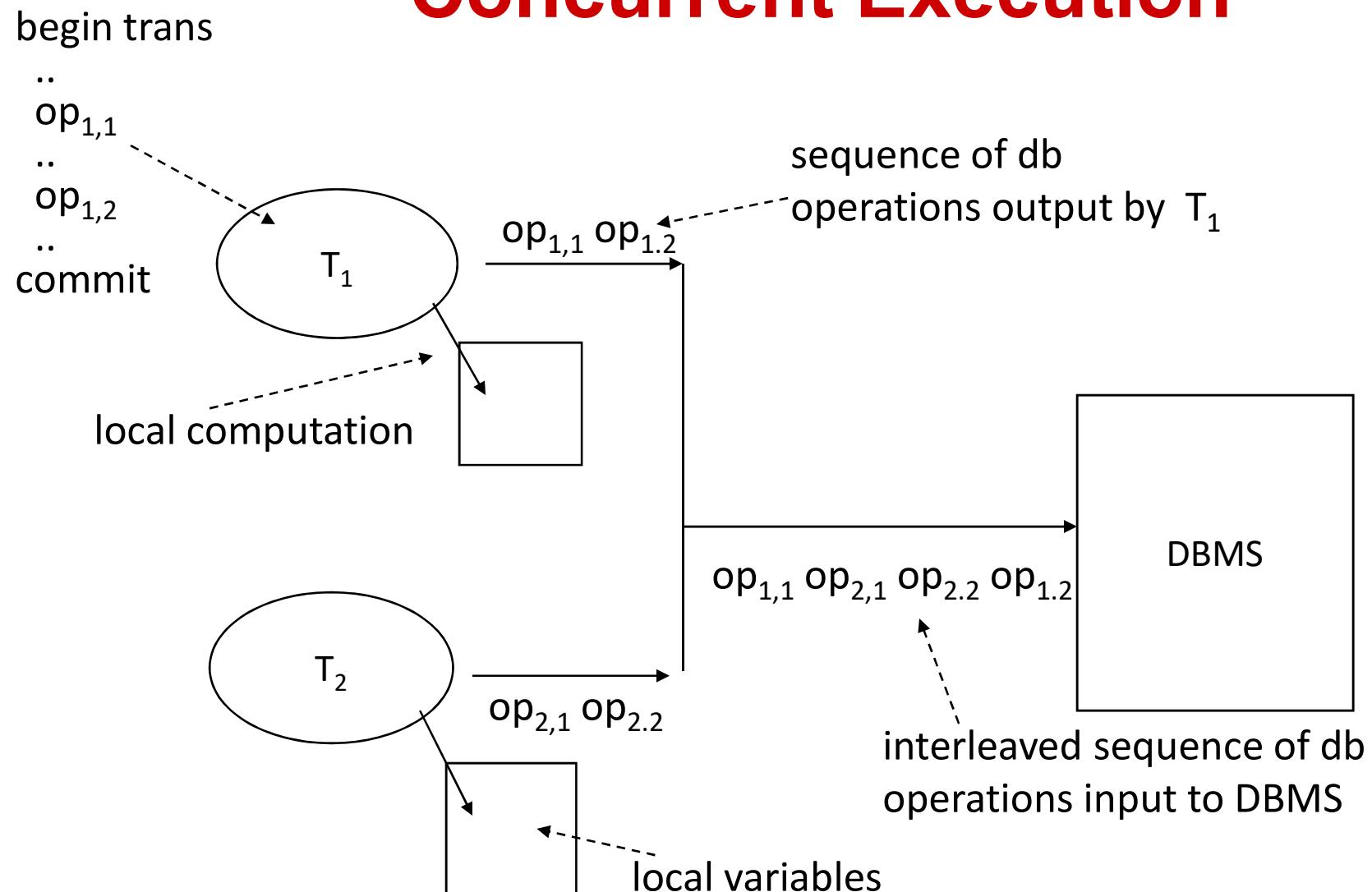
# Isolation

- **Serial Execution:** transactions execute in sequence
  - Each one starts after the previous one completes.
    - Execution of one transaction is not affected by the operations of another since they do not overlap in time
  - The execution of each transaction is isolated from all others.
- If the initial database state and all transactions are consistent, then the final database state will be consistent and will accurately reflect the real-world state, *but*
- Serial execution is inadequate from a performance perspective

# Isolation

- Concurrent execution offers performance benefits:
  - A computer system has multiple resources capable of executing independently (e.g., cpu's, I/O devices), but
  - A transaction typically uses only one resource at a time
  - Hence, only concurrently executing transactions can make effective use of the system
  - Concurrently executing transactions yield interleaved schedules

# Concurrent Execution



# Durability

- The system must ensure that once a transaction commits, its effect on the database state is not lost in spite of subsequent failures
  - Not true of ordinary programs. A media failure after a program successfully terminates could cause the file system to be restored to a state that preceded the program's execution

# Implementing Durability

- Database stored redundantly on mass storage devices to protect against media failure
- Architecture of mass storage devices affects type of media failures that can be tolerated
- Related to Availability: extent to which a (possibly distributed) system can provide service despite failure
  - Non-stop DBMS (mirrored disks)
  - Recovery based DBMS (log)

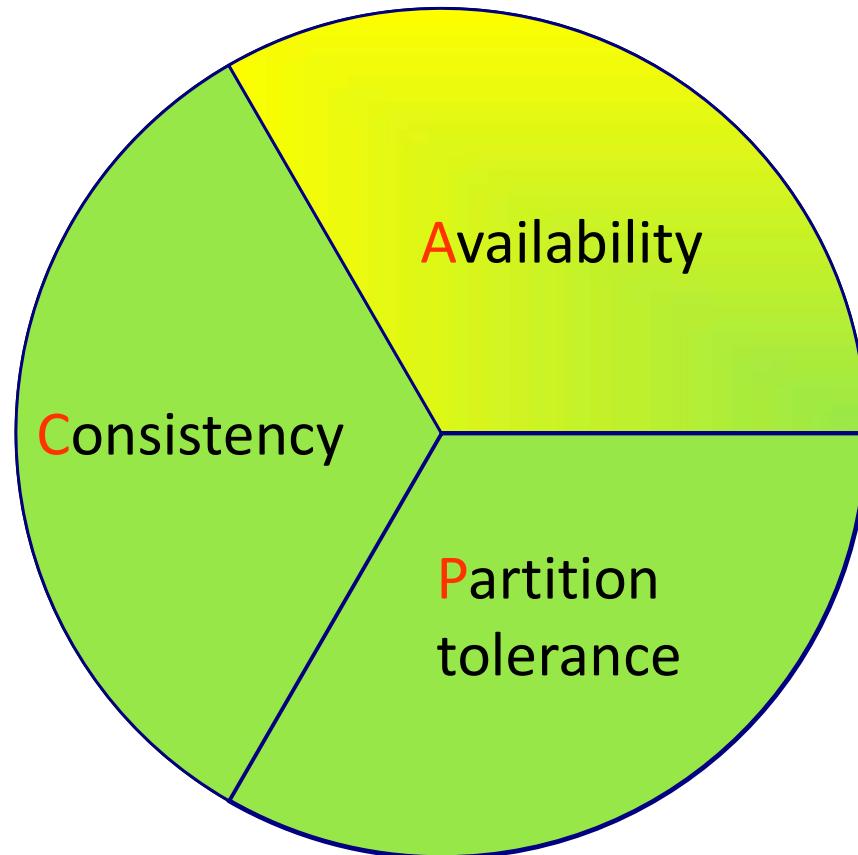
# Consistency Model

- A consistency model determines rules for visibility and apparent order of updates.
- For example:
  - Row X is replicated on nodes M and N
  - Client A writes row X to node N
  - Some period of time t elapses.
  - Client B reads row X from node M
  - Does client B see the write from client A?
  - Consistency is a continuum with tradeoffs
  - For NoSQL, the answer would be: maybe
  - CAP Theorem states: Strict Consistency can't be achieved at the same time as availability and partition-tolerance.

# Eventual Consistency

- When no updates occur for a long period of time, eventually all updates will propagate through the system and all the nodes will be consistent
- For a given accepted update and a given node, eventually either the update reaches the node or the node is removed from service
- Known as **BASE** (**B**asically **A**vailable, **S**oft state, **E**ventual consistency), as opposed to ACID

# The CAP Theorem

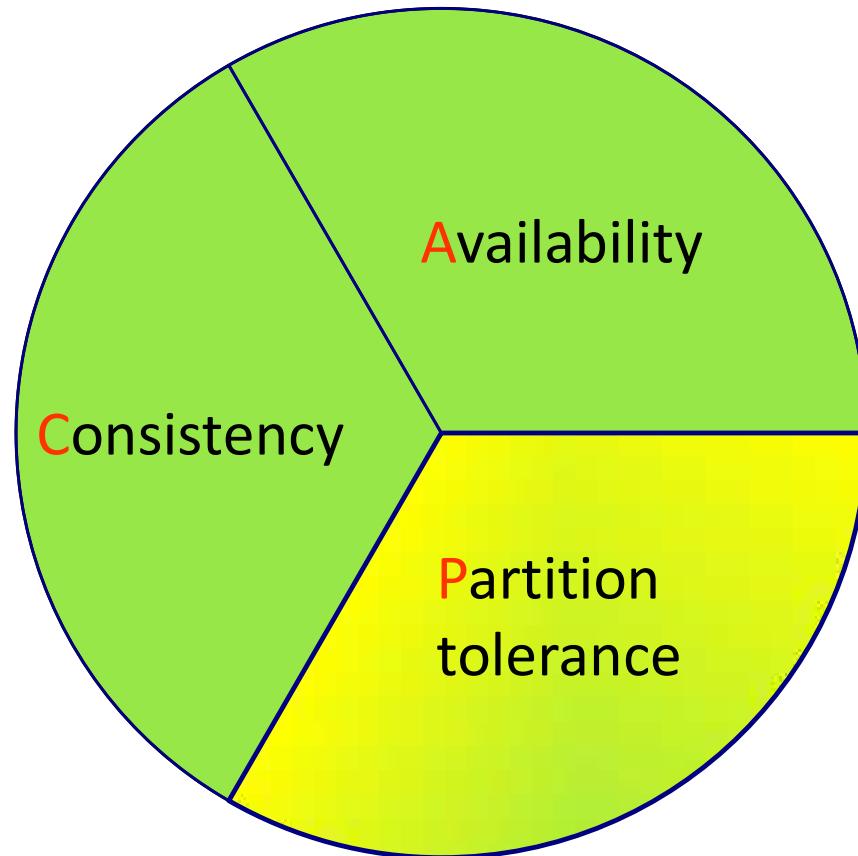


System is available during software and hardware upgrades and node failures.

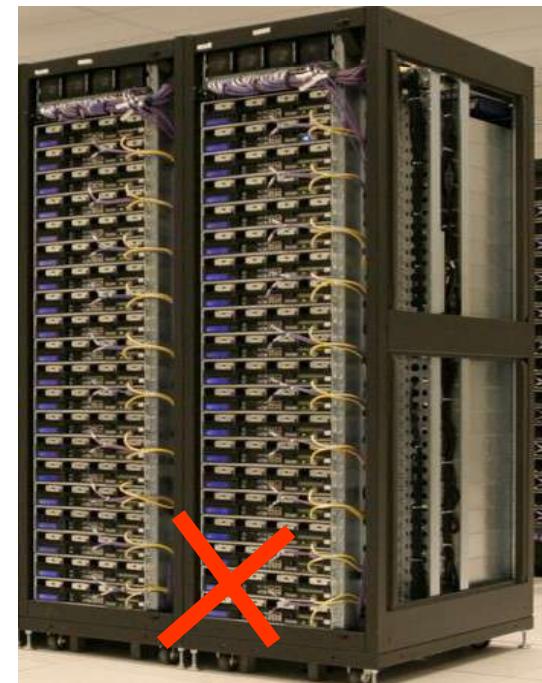
# Availability

- Traditionally, thought of as the server/process available five 9's (99.999 %).
- However, for large node system, at almost any point in time there's a good chance that a node is either down or there is a network disruption among the nodes.
  - Want a system that is resilient in the face of network disruption

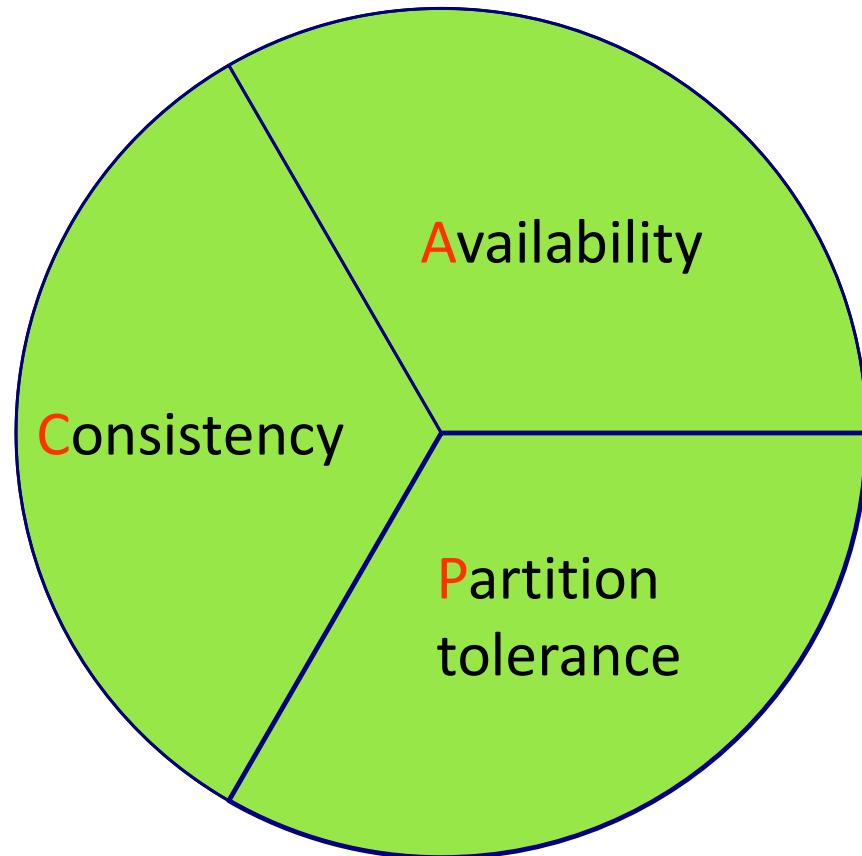
# The CAP Theorem



A system can continue to operate in the presence of a network partitions.



# The CAP Theorem



Theorem: You can have at most **two** of these properties for any shared-data system

# What kinds of NoSQL

- NoSQL solutions fall into two major areas:
  - Key/Value or ‘the big hash table’.
    - Amazon S3 (Dynamo)
    - Voldemort
    - Scalaris
    - Memcached (in-memory key/value store)
    - Redis
  - Schema-less which comes in multiple flavors, column-based, document-based or graph-based.
    - Cassandra (column-based)
    - CouchDB (document-based)
    - MongoDB (document-based)
    - Neo4J (graph-based)
    - HBase (column-based)

# Key/Value

## *Pros:*

- very fast
- very scalable
- simple model
- able to distribute horizontally

## *Cons:*

- many data structures (objects) can't be easily modeled as key value pairs

# Schema-Less

## *Pros:*

- Schema-less data model is richer than key/value pairs
- eventual consistency
- many are distributed
- still provide excellent performance and scalability

## *Cons:*

- typically no ACID transactions or joins

# Common Advantages

- Cheap, easy to implement (open source)
- Data are replicated to multiple nodes (therefore identical and fault-tolerant) and can be partitioned
  - Down nodes easily replaced
  - No single point of failure
- Easy to distribute
- Don't require a schema
- Can scale up and down
- Relax the data consistency requirement (CAP)

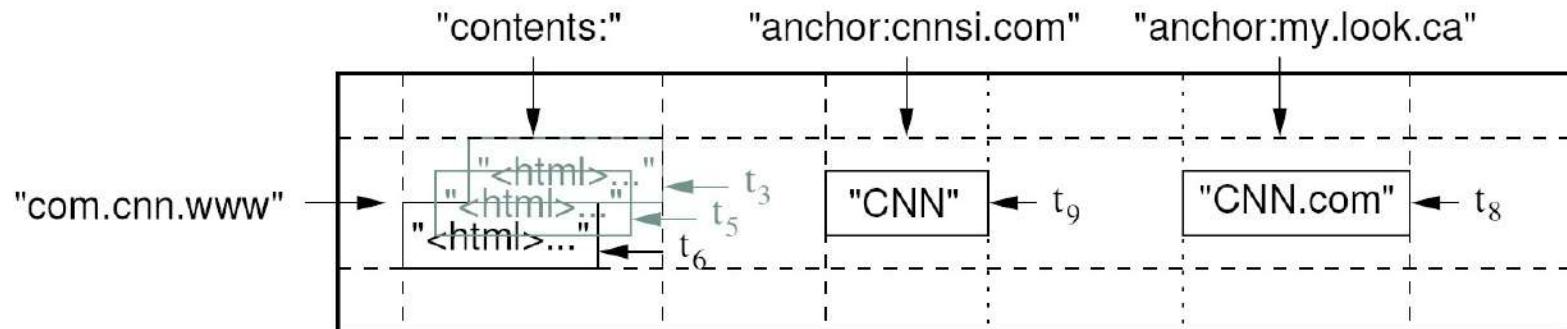
# What am I giving up?

- joins
- group by
- order by
- ACID transactions
- SQL as a sometimes frustrating but still powerful query language
- easy integration with other applications that support SQL

# **Big Table and Hbase (C+P)**

# Data Model

- A table in Bigtable is a sparse, distributed, persistent multidimensional sorted map
- Map indexed by a row key, column key, and a timestamp



- - Single row transactions only

# Rows and Columns

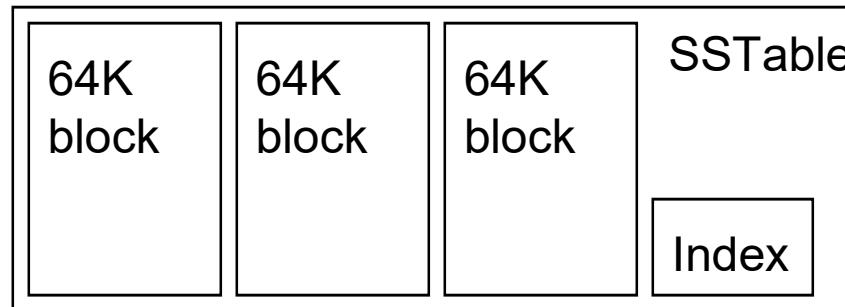
- Rows maintained in sorted lexicographic order
  - Applications can exploit this property for efficient row scans
  - Row ranges dynamically partitioned into tablets
- Columns grouped into column families
  - Column key = *family:qualifier*
  - Column families provide locality hints
  - Unbounded number of columns

# Bigtable Building Blocks

- GFS
- Chubby
- SSTable

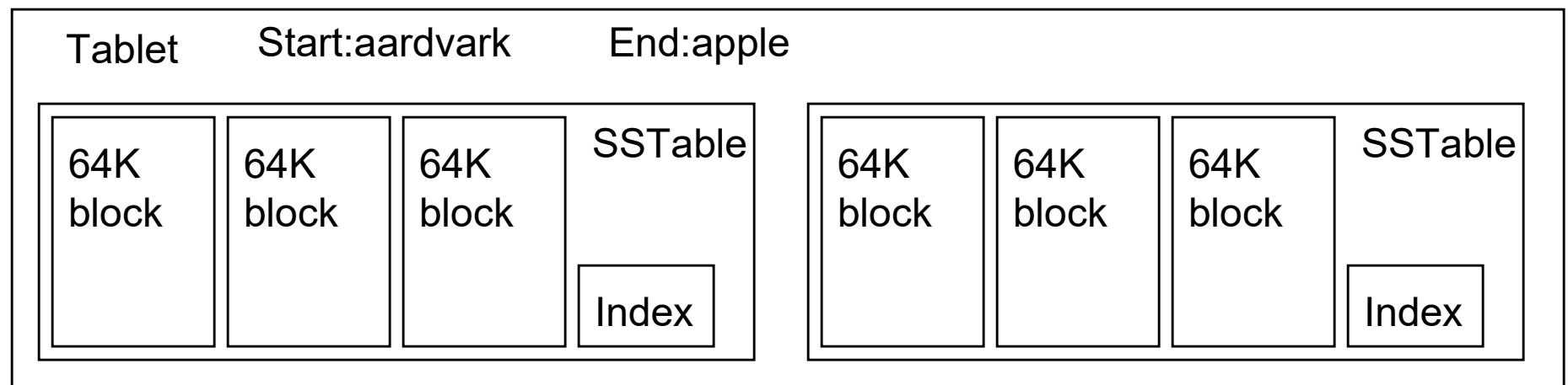
# SSTable

- Basic building block of Bigtable
- Persistent, ordered immutable map from keys to values
  - Stored in GFS
- Sequence of blocks on disk plus an index for block lookup
  - Can be completely mapped into memory
- Supported operations:
  - Look up value associated with key
  - Iterate key/value pairs within a key range



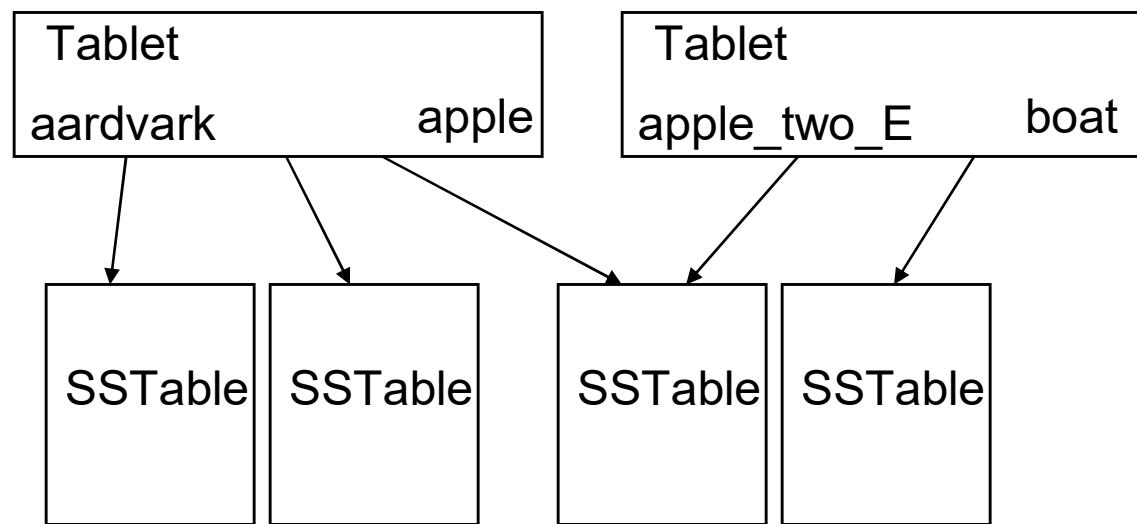
# Tablet

- Dynamically partitioned range of rows
- Built from multiple SSTables



# Table

- Multiple tablets make up the table
- SSTables can be shared



# Architecture

- Client library
- Single master server
- Tablet servers

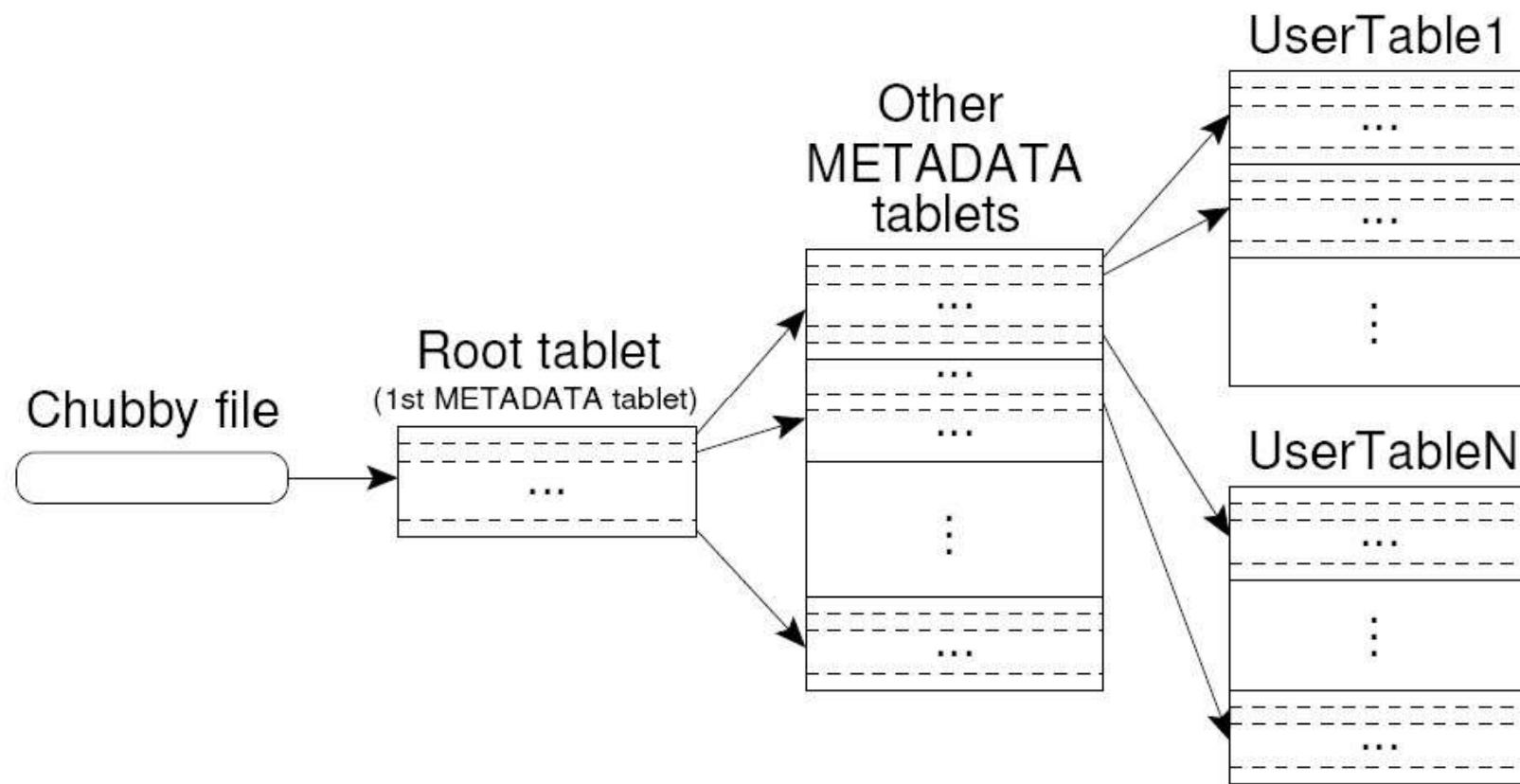
# Bigtable Master

- Assigns tablets to tablet servers
- Detects addition and expiration of tablet servers
- Balances tablet server load
- Handles garbage collection
- Handles schema changes

# Bigtable Tablet Servers

- Each tablet server manages a set of tablets
  - Typically between ten to a thousand tablets
  - Each 100-200 MB by default
- Handles read and write requests to the tablets
- Splits tablets that have grown too large

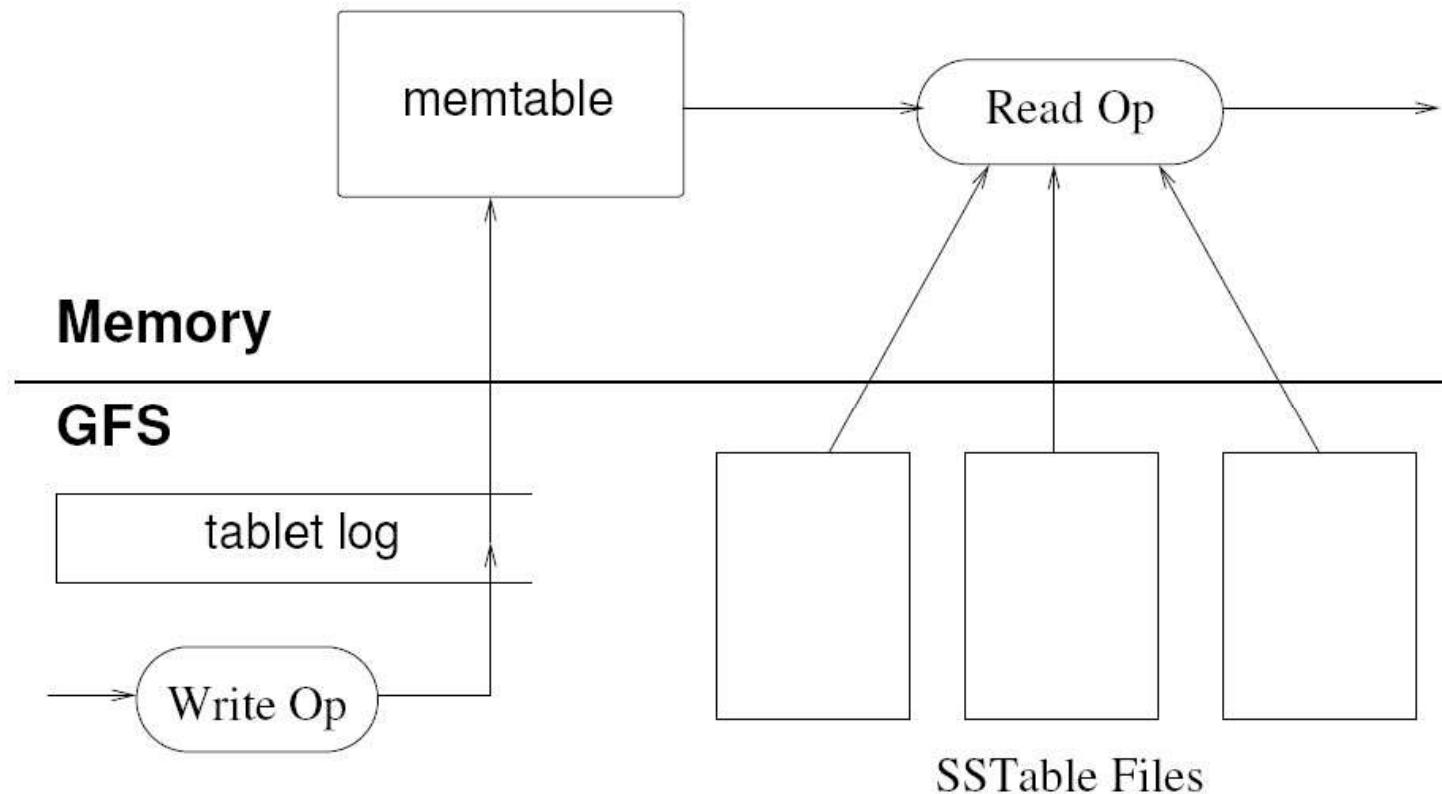
# Tablet Location



# Tablet Assignment

- Master keeps track of:
  - Set of live tablet servers
  - Assignment of tablets to tablet servers
  - Unassigned tablets
- Each tablet is assigned to one tablet server at a time
  - Tablet server maintains an exclusive lock on a file in Chubby
  - Master monitors tablet servers and handles assignment
- Changes to tablet structure
  - Table creation/deletion (master initiated)
  - Tablet merging (master initiated)
  - Tablet splitting (tablet server initiated)

# Tablet Serving



# Compactions

- Minor compaction
  - Converts the memtable into an SSTable
  - Reduces memory usage and log traffic on restart
- Merging compaction
  - Reads the contents of a few SSTables and the memtable, and writes out a new SSTable
  - Reduces number of SSTables
- Major compaction
  - Merging compaction that results in only one SSTable
  - No deletion records, only live data

# Bigtable Applications

- Data source and data sink for MapReduce
- Google's web crawl
- Google Earth
- Google Analytics

# Lessons Learned

- Fault tolerance is hard
- Don't add functionality before understanding its use
  - Single-row transactions appear to be sufficient
- Keep it simple!

HBase is an **open-source, distributed, column-oriented** database built on top of HDFS based on BigTable!

# HBase is ..

- A distributed data store that can scale horizontally to 1,000s of commodity servers and petabytes of indexed storage.
- Designed to operate on top of the Hadoop distributed file system (HDFS) or Kosmos File System (KFS, aka Cloudstore) for scalability, fault tolerance, and high availability.

# Benefits

- Distributed storage
- Table-like in data structure
  - multi-dimensional map
- High scalability
- High availability
- High performance

# Backdrop

- Started toward by Chad Walters and Jim
- 2006.11
  - Google releases paper on BigTable
- 2007.2
  - Initial HBase prototype created as Hadoop contrib.
- 2007.10
  - First useable HBase
- 2008.1
  - Hadoop become Apache top-level project and HBase becomes subproject
- 2008.10~
  - HBase 0.18, 0.19 released

# HBase Is Not ...

- Tables have one primary index, the *row key*.
- No join operators.
- Scans and queries can select a subset of available columns, perhaps by using a wildcard.
- There are three types of lookups:
  - Fast lookup using row key and optional timestamp.
  - Full table scan
  - Range scan from region start to end.

# HBase Is Not ...(2)

- Limited atomicity and transaction support.
  - HBase supports multiple batched mutations of single rows only.
  - Data is unstructured and untyped.
- No accessed or manipulated via SQL.
  - Programmatic access via Java, REST, or Thrift APIs.
  - Scripting via JRuby.

# Why Bigtable?

- Performance of RDBMS system is good for transaction processing but for very large scale analytic processing, the solutions are commercial, expensive, and specialized.
- Very large scale analytic processing
  - Big queries – typically range or table scans.
  - Big databases (100s of TB)

# Why Bigtable? (2)

- Map reduce on Bigtable with optionally Cascading on top to support some relational algebras may be a cost effective solution.
- Sharding is not a solution to scale open source RDBMS platforms
  - Application specific
  - Labor intensive (re)partitionaing

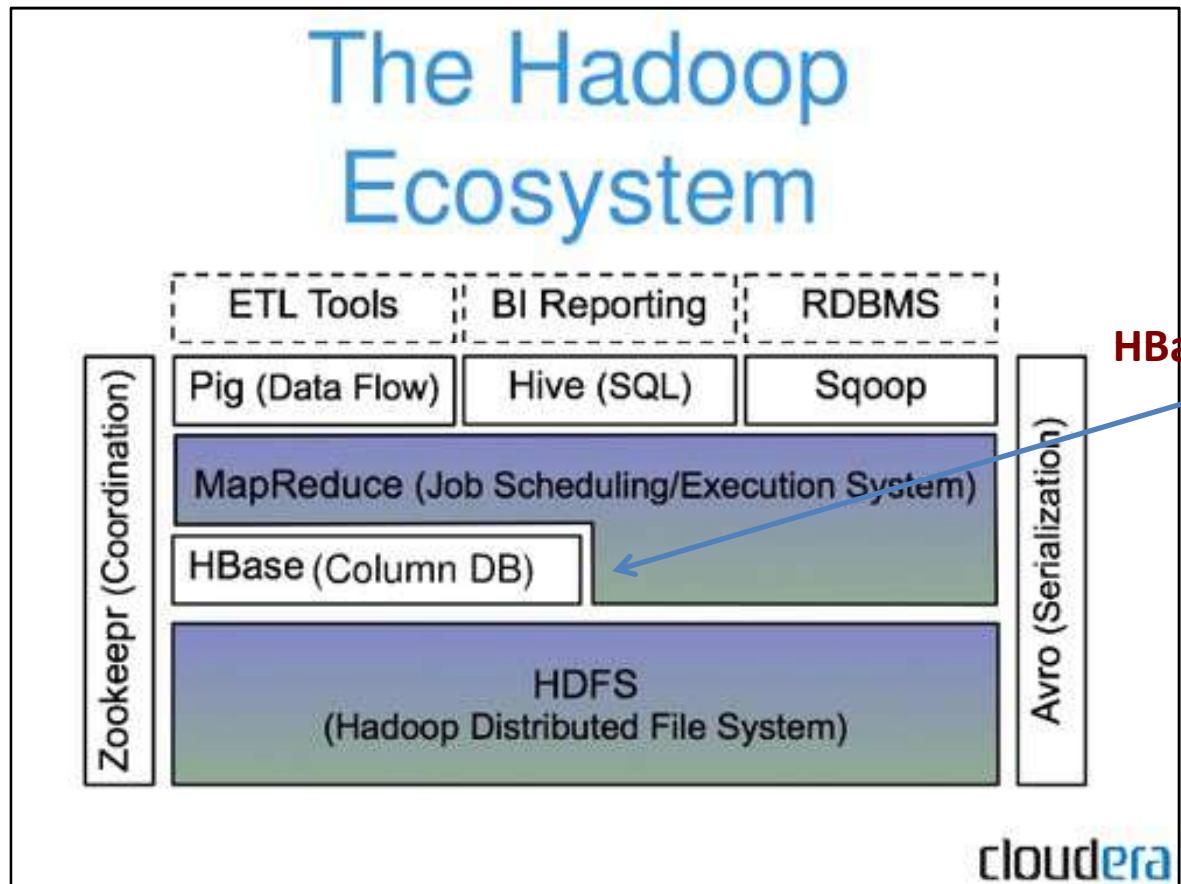
# Why HBase ?

- HBase is a Bigtable clone.
- It is open source
- It has a good community and promise for the future
- It is developed on top of and has good integration for the Hadoop platform, if you are using Hadoop already.
- It has a Cascading connector.

# HBase benefits than RDBMS

- *No real indexes*
- *Automatic partitioning*
- *Scale linearly and automatically with new nodes*
- *Commodity hardware*
- *Fault tolerance*
- *Batch processing*

# HBase: Part of Hadoop's Ecosystem



HBase is built on top of HDFS  
↓  
HBase files are internally stored in HDFS

# HBase vs. HDFS

- Both are distributed systems that scale to hundreds or thousands of nodes
- HDFS is good for batch processing (scans over big files)
  - Not good for record lookup
  - Not good for incremental addition of small batches
  - Not good for updates

# HBase vs. HDFS (Cont'd)

- ***HBase*** is designed to efficiently address the above points
  - Fast record lookup
  - Support for record-level insertion
  - Support for updates (not in place)
- HBase updates are done by creating new versions of values

# HBase vs. HDFS (Cont'd)

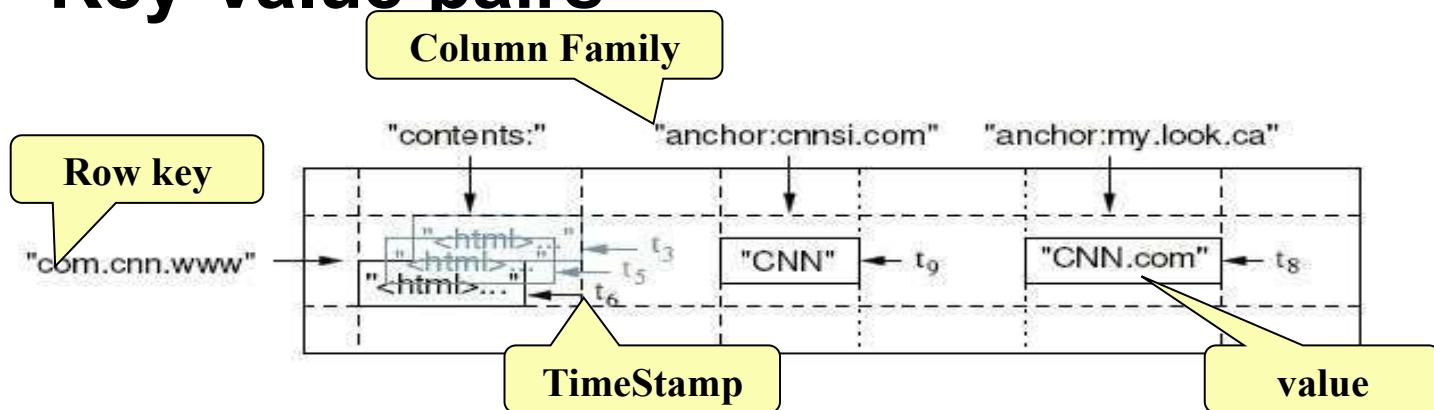
	<b>Plain HDFS/MR</b>	<b>HBase</b>
<b>Write pattern</b>	<b>Append-only</b>	<b>Random write, bulk incremental</b>
<b>Read pattern</b>	<b>Full table scan, partition table scan</b>	<b>Random read, small range scan, or table scan</b>
<b>Hive (SQL) performance</b>	<b>Very good</b>	<b>4-5x slower</b>
<b>Structured storage</b>	<b>Do-it-yourself / TSV / SequenceFile / Avro / ?</b>	<b>Sparse column-family data model</b>
<b>Max data size</b>	<b>30+ PB</b>	<b>~1PB</b>

**If application has neither random reads or writes → Stick to HDFS**

# **HBase Data Model**

# HBase Data Model

- HBase is based on Google's Bigtable model
  - Key-Value pairs



# HBase Logical View

Row key	Data
cutting	info: { 'height': '9ft', 'state': 'CA' } roles: { 'ASF': 'Director', 'Hadoop': 'Founder' }
tipcon	info: { 'height': '5ft7', 'state': 'CA' } roles: { 'Hadoop': 'Committer'@ts=2010, 'Hadoop': 'PMC'@ts=2011, 'Hive': 'Contributor' }

Implicit PRIMARY KEY in RDBMS terms

Data is all byte[] in HBase

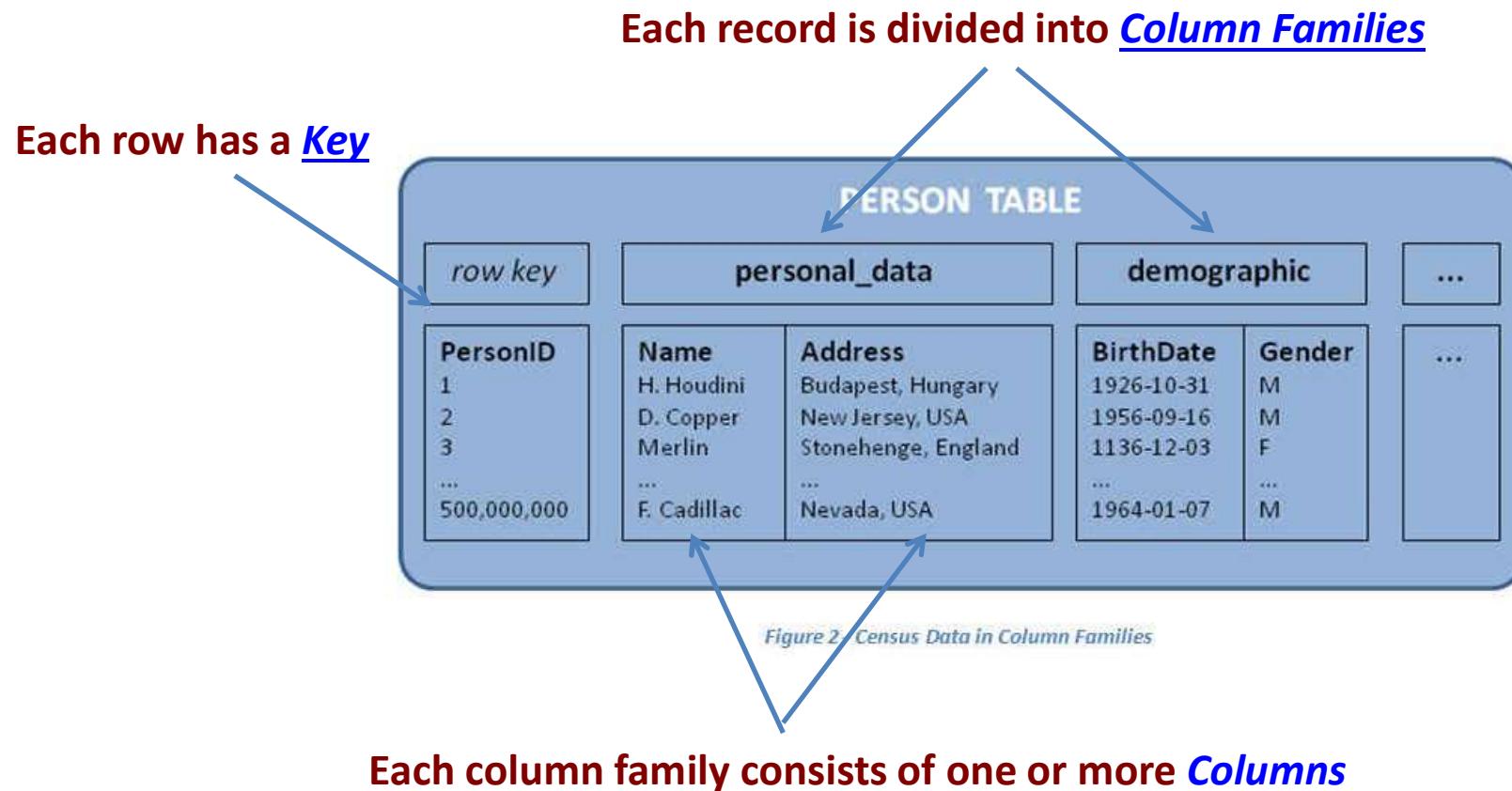
Different types of data separated into different "column families"

Different rows may have different sets of columns(table is sparse)

Useful for \*-To-Many mappings

A single cell might have different values at different timestamps

# HBase: Keys and Column Families



- **Key**
  - Byte array
  - Serves as the primary key for the table
  - Indexed for fast lookup
- **Column Family**
  - Has a name (string)
  - Contains one or more related columns
- **Column**
  - Belongs to one column family
  - Included inside the row
    - *familyName:columnName*

**Column family named “Contents”**

**Column family named “anchor”**

Row key	Time Stamp	Column “content s:”	Column “anchor:”
“com.apac he.ww w”	t12	“<html> ...”	
	t11	“<html> ...”	<b>Column named “apache.com”</b>
	t10		“anchor:apache .com” “APACH E”
	t15		“anchor:cnnsi.co m” “CNN”
	t13		“anchor:my.look. ca” “CNN.co m”
	t6	“<html> ...”	
	t5	“<html> ...”	
	t3	“<html> ...”	

- **Version Number**
  - Unique within each key
  - By default → System's timestamp
  - Data type is Long
- **Value (Cell)**
  - Byte array

Version number for each row

Row key	Time Stamp	Column “content s:”	Column “anchor:”
“com.apac he.ww w”	t12	“<html> ...”	value
	t11	“<html> ...”	
	t10		“APACH E”
	t15		“CNN”
	t13		“CNN.co m”
	t6	“<html> ...”	
	t5	“<html> ...”	
	t3	“<html> ...”	

# Notes on Data Model

- HBase schema consists of several *Tables*
  - Each table consists of a set of *Column Families*
    - Columns are not part of the schema
  - HBase has *Dynamic Columns*
    - Because column names are encoded inside the cells
    - Different cells can have different columns

“Roles” column family  
has different columns in  
different cells

Row key	Data
cutting	info: { 'height': '9ft', 'state': 'CA' } roles: { 'ASF': 'Director', 'Hadoop': 'Founder' }
tipcon	info: { 'height': '5ft7', 'state': 'CA' } roles: { 'Hadoop': 'Committer' @ts=2010, 'Hadoop': 'PMC' @ts=2011, 'Hive': 'Contributor' }

# Notes on Data Model (Cont'd)

- The **version number** can be user-supplied
  - Even does not have to be inserted in increasing order
  - Version number are unique within each key
- Table can be very sparse
  - Many cells are empty
- **Keys** are indexed as the primary key

Has two columns  
[cnnsi.com & my.look.ca]



Row Key	Time Stamp	ColumnFamily contents	ColumnFamily anchor
"com.cnn.www"	t9		anchor:cnnsi.com = "CNN"
"com.cnn.www"	t8		anchor:my.look.ca = "CNN.com"
"com.cnn.www"	t6	contents:html = "<html>..."	
"com.cnn.www"	t5	contents:html = "<html>..."	
"com.cnn.www"	t3	contents:html = "<html>..."	

# **HBase Physical Model**

# HBase Physical Model

- Each column family is stored in a separate file (called ***HTables***)
- Key & Version numbers are replicated with each column family
- Empty cells are not stored

HBase maintains a multi-level index on values:  
*<key, column family, column name, timestamp>*

**Table 5.3. ColumnFamily contents**

Row Key	Time Stamp	ColumnFamily "contents:"
"com.cnn.www"	t6	contents:html = "<html>..."
"com.cnn.www"	t5	contents:html = "<html>..."
"com.cnn.www"	t3	contents:html = "<html>..."

**Table 5.2. ColumnFamily anchor**

Row Key	Time Stamp	Column Family anchor
"com.cnn.www"	t9	anchor:cnnsi.com = "CNN"
"com.cnn.www"	t8	anchor:my.look.ca = "CNN.com"

# Example

Row key	Data
cutting	info: { 'height': '9ft', 'state': 'CA' } roles: { 'ASF': 'Director', 'Hadoop': 'Founder' }
tipcon	info: { 'height': '5ft7', 'state': 'CA' } roles: { 'Hadoop': 'Committer' @ts=2010, 'Hadoop': 'PMC' @ts=2011, 'Hive': 'Contributor' }

## **info Column Family**

Row key	Column key	Timestamp	Cell value
cutting	info:height	1273516197868	9ft
cutting	info:state	1043871824184	CA
tipcon	info:height	1273878447049	5ft7
tipcon	info:state	1273616297446	CA

## roles Column Family

Row key	Column key	Timestamp	Cell value
cutting	roles:ASF	1273871823022	Director
cutting	roles:Hadoop	1183746289103	Founder
tlipcon	roles:Hadoop	1300062064923	PMC
tlipcon	roles:Hadoop	1293388212294	Committer
tlipcon	roles:Hive	1273616297446	Contributor

Sorted  
on disk by  
Row key, Col  
key,  
descending  
timestamp

Milliseconds since unix epoch

cloudera

# Column Families

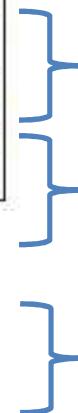
- Different sets of columns may have different properties and access patterns
- Configurable by column family:
  - Compression (none, gzip, LZO)
  - Version retention policies
  - Cache priority
- CFs stored separately on disk: access one without wasting IO on the other.

# HBase Regions

- Each HTable (column family) is partitioned horizontally into *regions*
  - Regions are counterpart to HDFS blocks

**Table 5.3. ColumnFamily contents**

Row Key	Time Stamp	ColumnFamily "contents:"
"com.cnn.www"	t6	contents:html = "<html>..."
"com.cnn.www"	t5	contents:html = "<html>..."
"com.cnn.www"	t3	contents:html = "<html>..."

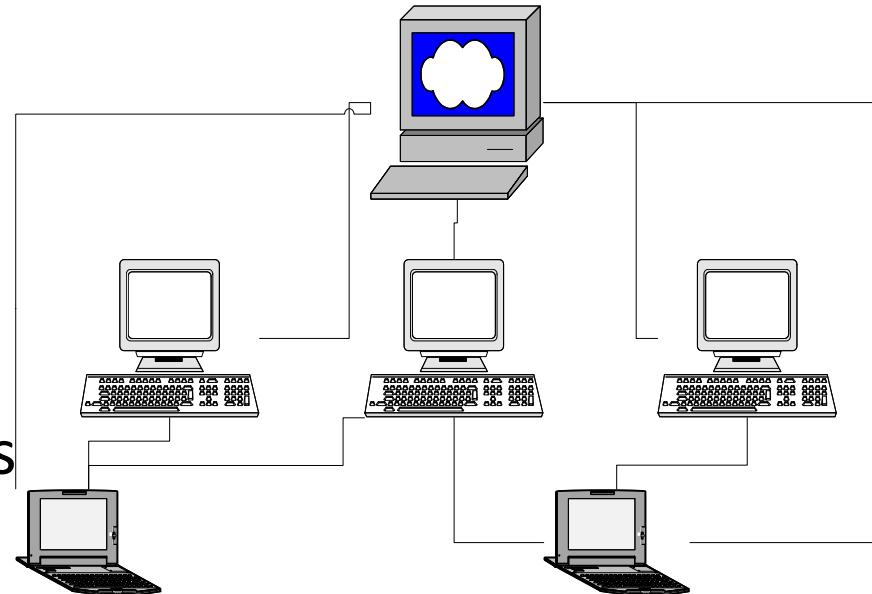


*Each will be one region*

# **HBase Architecture**

# Three Major Components

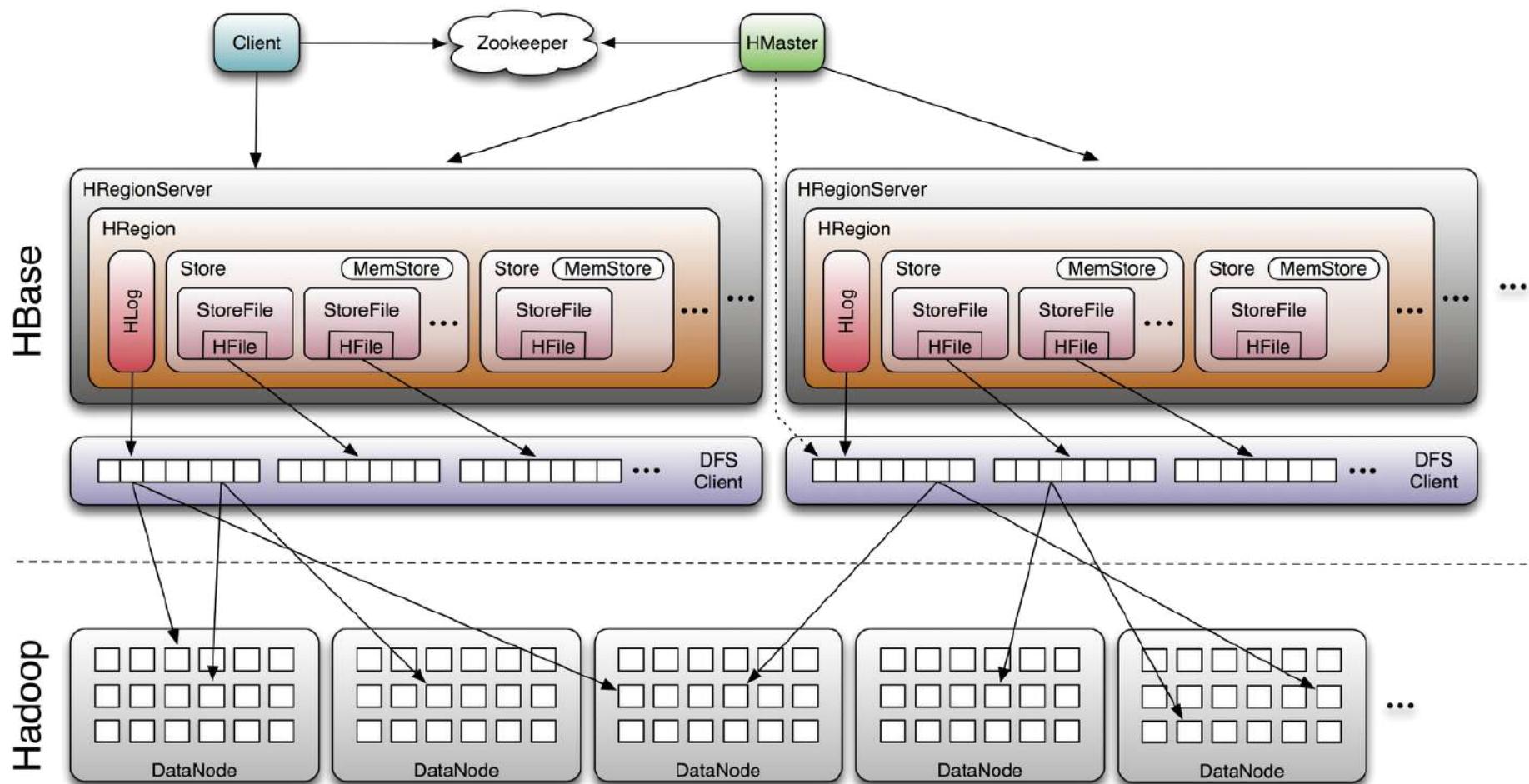
- The HBaseMaster
  - One master
- The HRegionServer
  - Many region servers
- The HBase client



# HBase Components

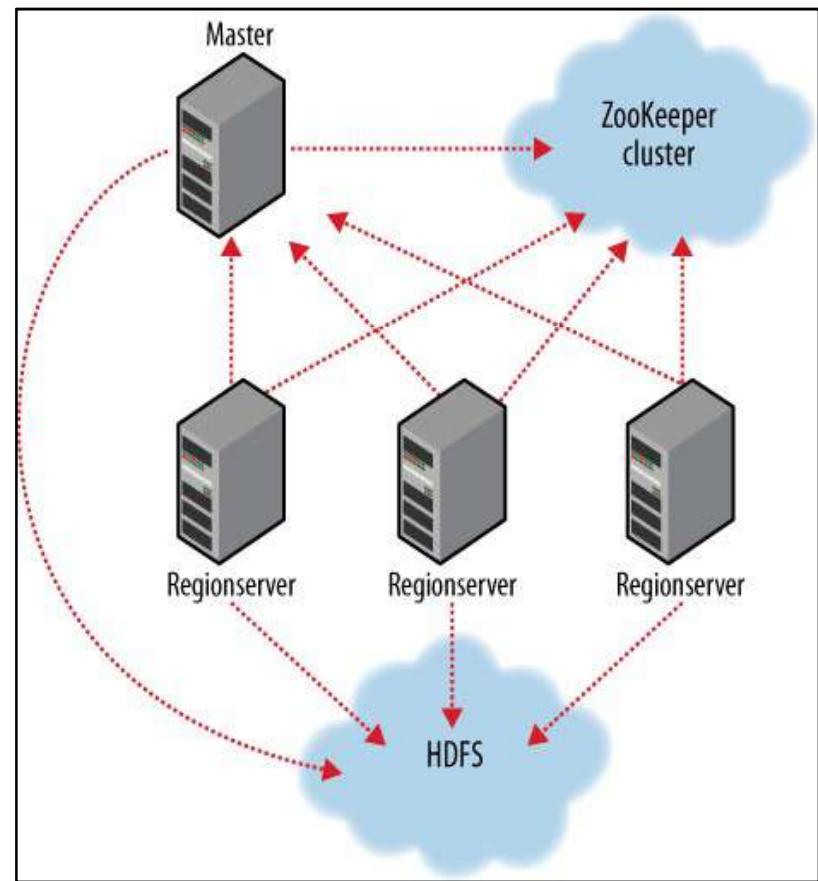
- **Region**
  - A subset of a table's rows, like horizontal range partitioning
  - Automatically done
- **RegionServer (many slaves)**
  - Manages data regions
  - Serves data for reads and writes (*using a log*)
- **Master**
  - Responsible for coordinating the slaves
  - Assigns regions, detects failures
  - Admin functions

# Big Picture



# ZooKeeper

- HBase depends on ZooKeeper
- By default HBase manages the ZooKeeper instance
  - E.g., starts and stops ZooKeeper
- HMaster and HRegionServers register themselves with ZooKeeper



# Creating a Table

```
HBaseAdmin admin= new HBaseAdmin(config);
HColumnDescriptor []column;
column= new HColumnDescriptor[2];
column[0]=new
    HColumnDescriptor("columnFamily1:");
column[1]=new
    HColumnDescriptor("columnFamily2:");
HTableDescriptor desc= new HTableDescriptor(Bytes.toBytes("MyTable"));
desc.addFamily(column[0]);
desc.addFamily(column[1]);
admin.createTable(desc);
```

# Operations On Regions: **Get()**

- Given a key → return corresponding record
- For each value return the highest version

```
Get get = new Get(Bytes.toBytes("row1"));
Result r = htable.get(get);
5.8.1.2. Default Get Example r.getValue(Bytes.toBytes("cf"), Bytes.toBytes("attr")); // returns current version of value
```

- Can control the number of versions you want

```
Get get = new Get(Bytes.toBytes("row1"));
get.setMaxVersions(3); // will return last 3 versions of row
Result r = htable.get(get);
byte[] b = r.getValue(Bytes.toBytes("cf"), Bytes.toBytes("attr")); // returns current version of value
List<KeyValue> kv = r.getColumn(Bytes.toBytes("cf"), Bytes.toBytes("attr")); // returns all versions of
```

# Operations On Regions: **Scan()**

```
HTable htable = ...      // instantiate HTable

Scan scan = new Scan();
scan.addColumn(Bytes.toBytes("cf"),Bytes.toBytes("attr"));
scan.setStartRow( Bytes.toBytes("row"));           // start key is inclusive
scan.setStopRow( Bytes.toBytes("row" + (char)0)); // stop key is exclusive
ResultScanner rs = htable.getScanner(scan);
try {
  for (Result r = rs.next(); r != null; r = rs.next()) {
    // process result...
  } finally {
    rs.close(); // always close the ResultScanner!
  }
```

# Get()

Select value from table where  
key= ‘com.apache.www’ AND  
label= ‘anchor:apache.com’

Row key	Time Stamp	Column “anchor:”	
“com.apache.www”	t12		
	t11		
	t10	“anchor:apache.com”	“APACHE”
	t9	“anchor:cnnsi.com”	“CNN”
	t8	“anchor:my.look.ca”	“CNN.com”
	t6		
	t5		
	t3		

# Scan()

Select value from table  
where  
anchor= ‘cnnsi.com’

Row key	Time Stamp	Column “anchor:”	
“com.apache.www”	t12		
	t11		
	t10	“anchor:apache.com”	“APACHE”
“com.cnn.www”	t9	“anchor:cnnsi.com”	“CNN”
	t8	“anchor:my.look.ca”	“CNN.com”
	t6		
	t5		
	t3		

# Operations On Regions: Put()

- Insert a new record (with a new key), Or
- Insert a record for an existing key

Implicit version number  
(timestamp)

```
Put put = new Put(Bytes.toBytes(row));
put.add(Bytes.toBytes("cf"), Bytes.toBytes("attr1"), Bytes.toBytes( data));
htable.put(put);
```

Explicit version number

```
Put put = new Put( Bytes.toBytes(row));
long explicitTimeInMs = 555; // just an example
put.add(Bytes.toBytes("cf"), Bytes.toBytes("attr1"), explicitTimeInMs, Bytes.toBytes(data));
htable.put(put);
```

# Operations On Regions: **Delete()**

- Marking table cells as deleted
- **Multiple levels**
  - Can mark an entire column family as deleted
  - Can make all column families of a given row as deleted

- All operations are logged by the RegionServers
- The log is flushed periodically

# HBase: Joins

- HBase does not support joins
- Can be done in the application layer
  - Using scan() and get() operations

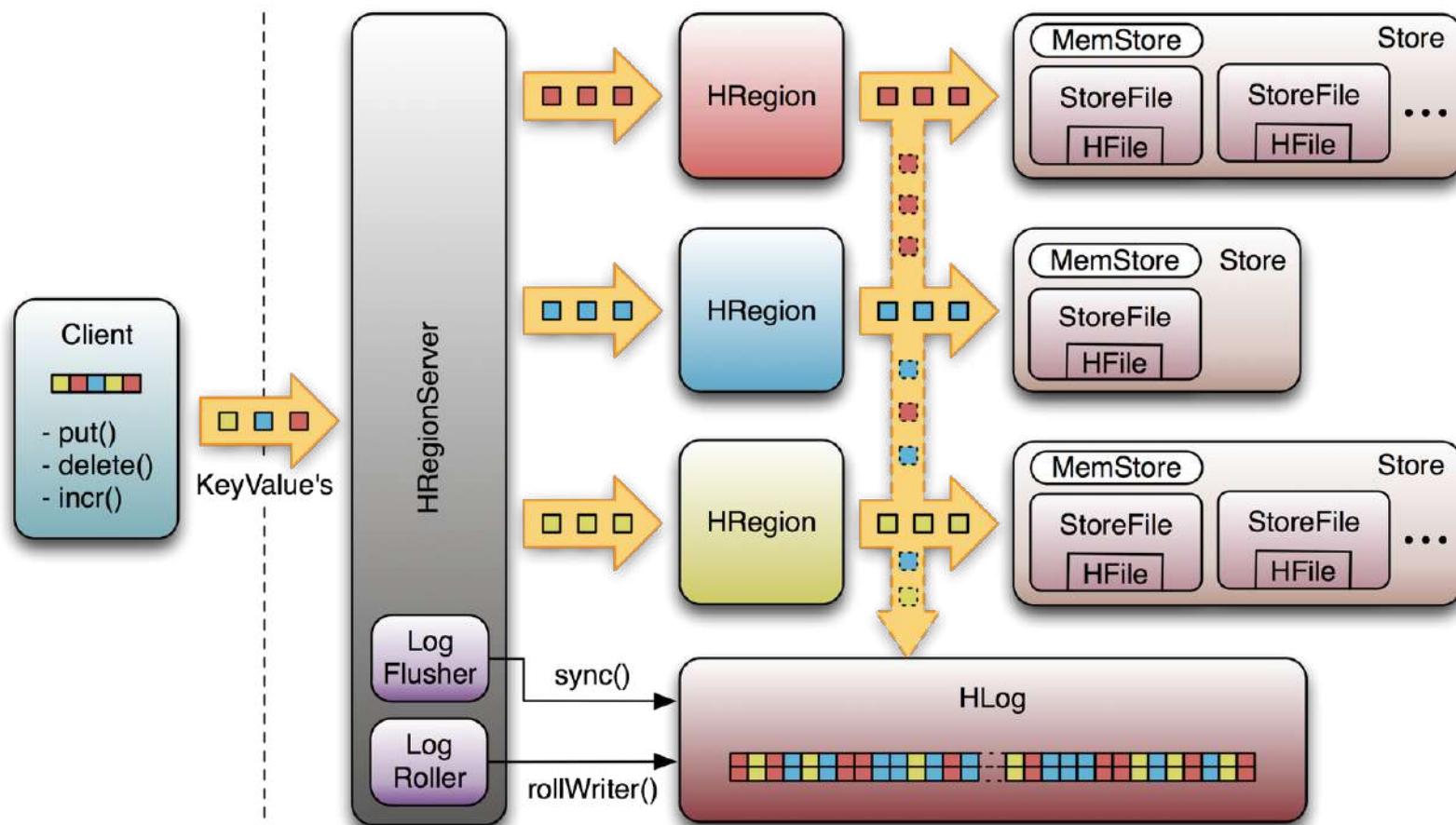
# Altering a Table

```
Configuration config = HBaseConfiguration.create();
HBaseAdmin admin = new HBaseAdmin(conf);
String table = "myTable";
admin.disableTable(table); ← Disable the table before changing the schema

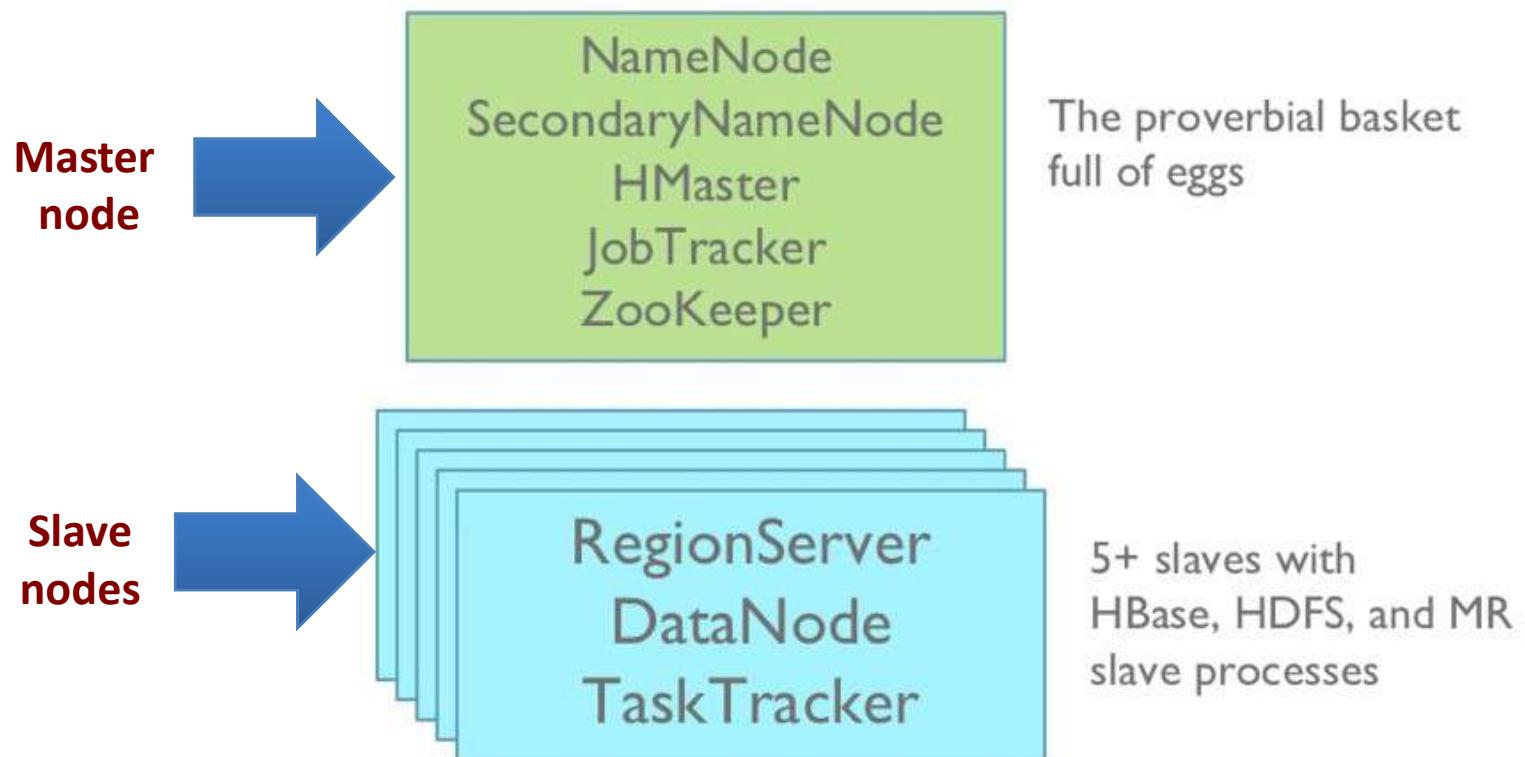
HColumnDescriptor cf1 = ...;
admin.addColumn(table, cf1);      // adding new ColumnFamily
HColumnDescriptor cf2 = ...;
admin.modifyColumn(table, cf2);   // modifying existing ColumnFamily

admin.enableTable(table); Schema Creation
```

# Logging Operations



# HBase Deployment



# HBase vs. HDFS

	<b>Plain HDFS/MR</b>	<b>HBase</b>
<b>Write pattern</b>	<b>Append-only</b>	<b>Random write, bulk incremental</b>
<b>Read pattern</b>	<b>Full table scan, partition table scan</b>	<b>Random read, small range scan, or table scan</b>
<b>Hive (SQL) performance</b>	<b>Very good</b>	<b>4-5x slower</b>
<b>Structured storage</b>	<b>Do-it-yourself / TSV / SequenceFile / Avro / ?</b>	<b>Sparse column-family data model</b>
<b>Max data size</b>	<b>30+ PB</b>	<b>~1PB</b>

# HBase vs. RDBMS

	RDBMS	HBase
Data layout	Row-oriented	Column-family-oriented
Transactions	Multi-row ACID	Single row only
Query language	SQL	get/put/scan/etc *
Security	Authentication/Authorization	Work in progress
Indexes	On arbitrary columns	Row-key only
Max data size	TBs	~1PB
Read/write throughput limits	1000s queries/second	Millions of queries/second

# When to use HBase

- You need random write, random read, or both (*but not neither*)
- You need to do many thousands of operations per second on multiple TB of data
- Your access patterns are well-known and simple

# **Hive and Pig**

# Need for High-Level Languages

- Hadoop is great for large-data processing!
  - But writing Java programs for everything is verbose and slow
  - Not everyone wants to (or can) write Java code
- Solution: develop higher-level data processing languages
  - Hive: HQL is like SQL
  - Pig: Pig Latin is a bit like Perl

# Hive and Pig

- Hive: data warehousing application in Hadoop
  - Query language is HQL, variant of SQL
  - Tables stored on HDFS as flat files
  - Developed by Facebook, now open source
- Pig: large-scale data processing system
  - Scripts are written in Pig Latin, a dataflow language
  - Developed by Yahoo!, now open source
  - Roughly 1/3 of all Yahoo! internal jobs
- Common idea:
  - Provide higher-level language to facilitate large-data processing
  - Higher-level language “compiles down” to Hadoop jobs



# Hive: Background

- Started at Facebook
- Data was collected by nightly cron jobs into Oracle DB
- “ETL” via hand-coded python
- Grew from 10s of GBs (2006) to 1 TB/day new data (2007), now 10x that

# Hive Components

- Shell: allows interactive queries
- Driver: session handles, fetch, execute
- Compiler: parse, plan, optimize
- Execution engine: DAG of stages (MR, HDFS, metadata)
- Metastore: schema, location in HDFS, SerDe

# Data Model

- Tables
  - Typed columns (int, float, string, boolean)
  - Also, list: map (for JSON-like data)
- Partitions
  - For example, range-partition tables by date
- Buckets
  - Hash partitions within ranges (useful for sampling, join optimization)

# Metastore

- Database: namespace containing a set of tables
- Holds table definitions (column types, physical layout)
- Holds partitioning information
- Can be stored in Derby, MySQL, and many other relational databases

# Physical Layout

- Warehouse directory in HDFS
  - E.g., /user/hive/warehouse
- Tables stored in subdirectories of warehouse
  - Partitions form subdirectories of tables
- Actual data stored in flat files
  - Control char-delimited text, or SequenceFiles
  - With custom SerDe, can use arbitrary format

# Hive: Example

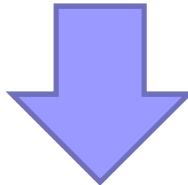
- Hive looks similar to an SQL database
- Relational join on two tables:
  - Table of word counts from Shakespeare collection
  - Table of word counts from the bible

```
SELECT s.word, s.freq, k.freq FROM shakespeare s
JOIN bible k ON (s.word = k.word) WHERE s.freq >= 1 AND k.freq >= 1
ORDER BY s.freq DESC LIMIT 10;
```

the	25848	62394
I	23031	8854
and	19671	38985
to	18038	13526
of	16700	34654
a	14170	8057
you	12702	2720
my	11297	4135
in	10797	12445
is	8882	6884

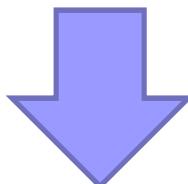
# Hive: Behind the Scenes

```
SELECT s.word, s.freq, k.freq FROM shakespeare s  
JOIN bible k ON (s.word = k.word) WHERE s.freq >= 1 AND k.freq >= 1  
ORDER BY s.freq DESC LIMIT 10;
```



(Abstract Syntax Tree)

```
(TOK_QUERY (TOK_FROM (TOK_JOIN (TOK_TABREF shakespeare s) (TOK_TABREF bible k) (= (. (TOK_TABLE_OR_COL s)  
word) (. (TOK_TABLE_OR_COL k) word)))) (TOK_INSERT (TOK_DESTINATION (TOK_DIR TOK_TMP_FILE)) (TOK_SELECT  
(TOK_SELEXPR (. (TOK_TABLE_OR_COL s) word)) (TOK_SELEXPR (. (TOK_TABLE_OR_COL s) freq)) (TOK_SELEXPR (.  
(TOK_TABLE_OR_COL k) freq))) (TOK_WHERE (AND (>= (. (TOK_TABLE_OR_COL s) freq) 1) (>= (. (TOK_TABLE_OR_COL k)  
freq) 1))) (TOK_ORDERBY (TOK_TABSORTCOLNAMEDESC (. (TOK_TABLE_OR_COL s) freq))) (TOK_LIMIT 10)))
```



(one or more of MapReduce jobs)

# Hive: Behind the Scenes

## STAGE DEPENDENCIES:

Stage-1 is a root stage  
Stage-2 depends on stages: Stage-1  
Stage-0 is a root stage

## STAGE PLANS:

Stage: Stage-1  
Map Reduce  
Alias -> Map Operator Tree:  
s  
TableScan  
alias: s  
Filter Operator  
predicate:  
expr: (freq >= 1)  
type: boolean  
Reduce Output Operator  
key expressions:  
expr: word  
type: string  
sort order: +  
Map-reduce partition columns:  
expr: word  
type: string  
tag: 0  
value expressions:  
expr: freq  
type: int  
expr: word  
type: string  
  
k  
TableScan  
alias: k  
Filter Operator  
predicate:  
expr: (freq >= 1)  
type: boolean  
Reduce Output Operator  
key expressions:  
expr: word  
type: string  
sort order: +  
Map-reduce partition columns:  
expr: word  
type: string  
tag: 1  
value expressions:  
expr: freq  
type: int

## Stage: Stage-2

Map Reduce  
Alias -> Map Operator Tree:  
hdfs://localhost:8022/tmp/hive-training/364214370/10002  
Reduce Output Operator  
key expressions:  
expr: \_col1  
type: int  
sort order: -  
tag: -1  
value expressions:  
expr: \_col0  
type: string  
expr: \_col1  
type: int  
expr: \_col2  
type: int  
Reduce Operator Tree:  
Extract  
Limit  
File Output Operator  
compressed: false  
GlobalTableId: 0  
table:  
input format: org.apache.hadoop.mapred.TextInputFormat  
output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat

## Stage: Stage-0

Fetch Operator  
limit: 10

## Stage: Stage-0

Select Operator  
expressions:  
expr: \_col1  
type: string  
expr: \_col0  
type: int  
expr: \_col2  
type: int  
outputColumnNames: \_col0, \_col1, \_col2  
File Output Operator  
compressed: false  
GlobalTableId: 0  
table:  
input format: org.apache.hadoop.mapred.SequenceFileInputFormat  
output format: org.apache.hadoop.hive.SequenceFileOutputFormat

# Example Data Analysis Task

**Find users who tend to visit “good” pages.**

**Visits**

user	url	time
Amy	www.cnn.com	8:00
Amy	www.crap.com	8:05
Amy	www.myblog.com	10:00
Amy	www.flickr.com	10:05
Fred	cnn.com/index.htm	12:00

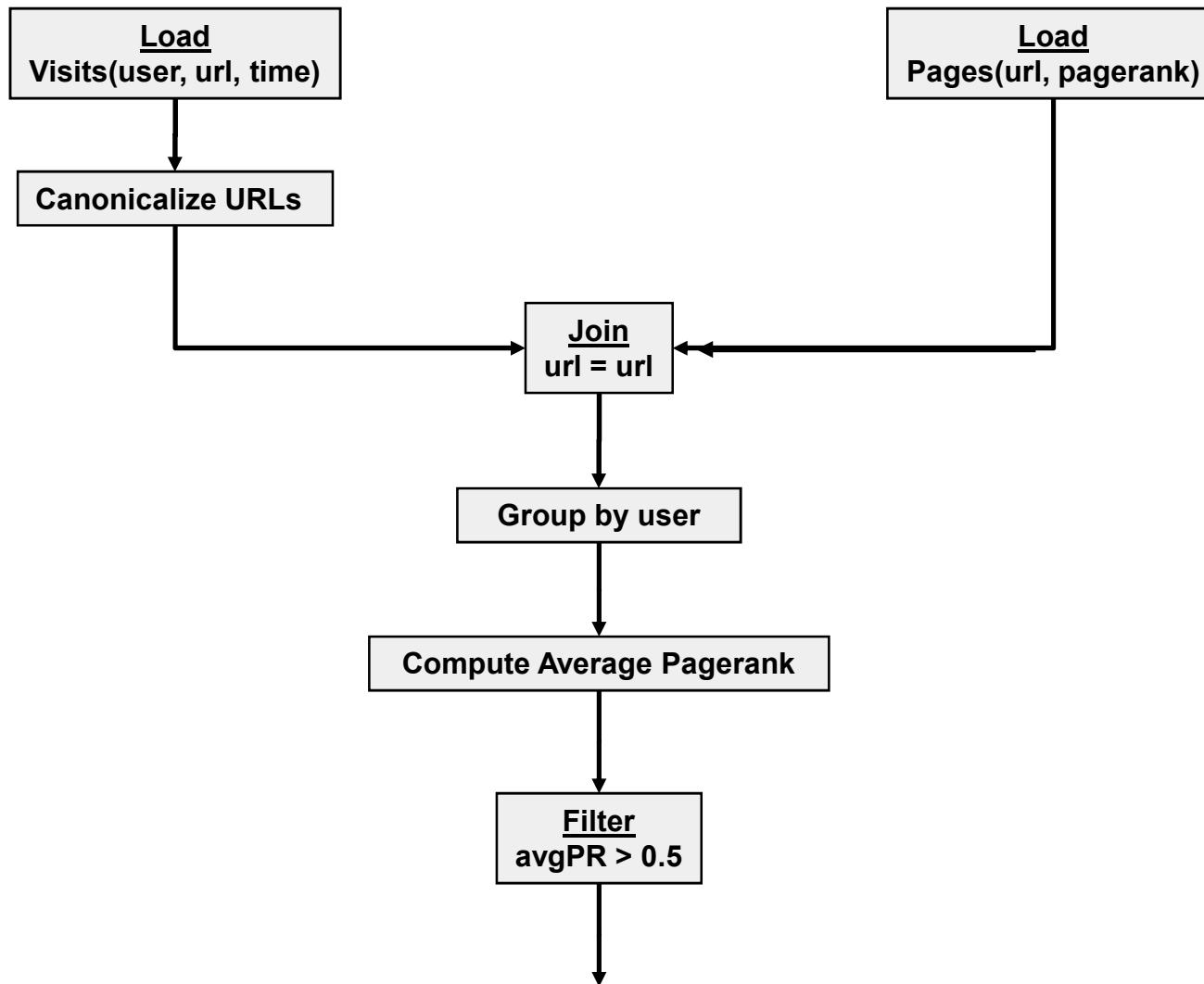
:

**Pages**

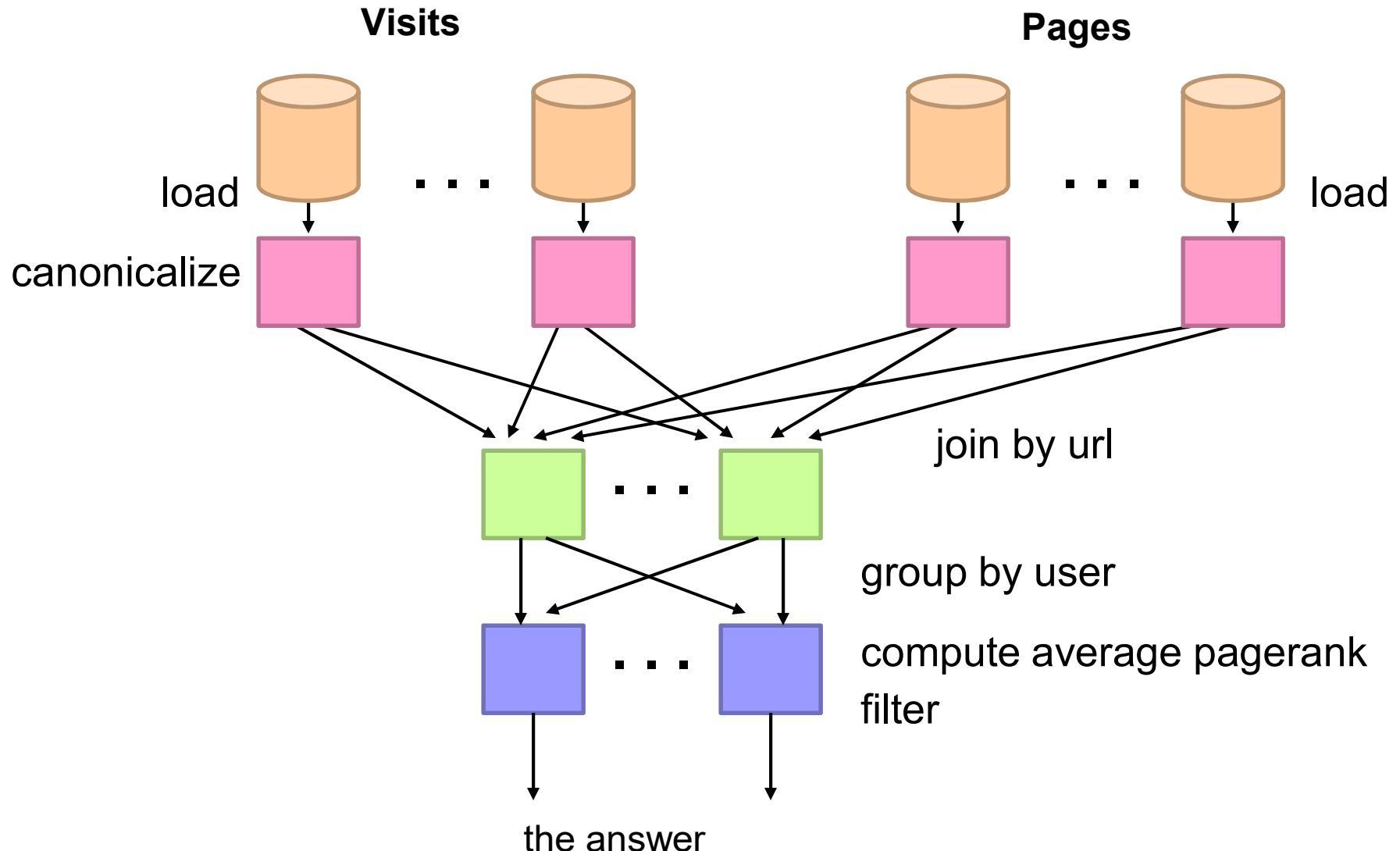
url	pagerank
www.cnn.com	0.9
www.flickr.com	0.9
www.myblog.com	0.7
www.crap.com	0.2

:

# Conceptual Dataflow



# System-Level Dataflow



# MapReduce Code

# Pig Latin Script

```
Visits = load '/data/visits' as (user, url, time);
```

```
Visits = foreach Visits generate user, Canonicalize(url), time;
```

```
Pages = load '/data/pages' as (url, pagerank);
```

```
VP = join Visits by url, Pages by url;
```

```
UserVisits = group VP by user;
```

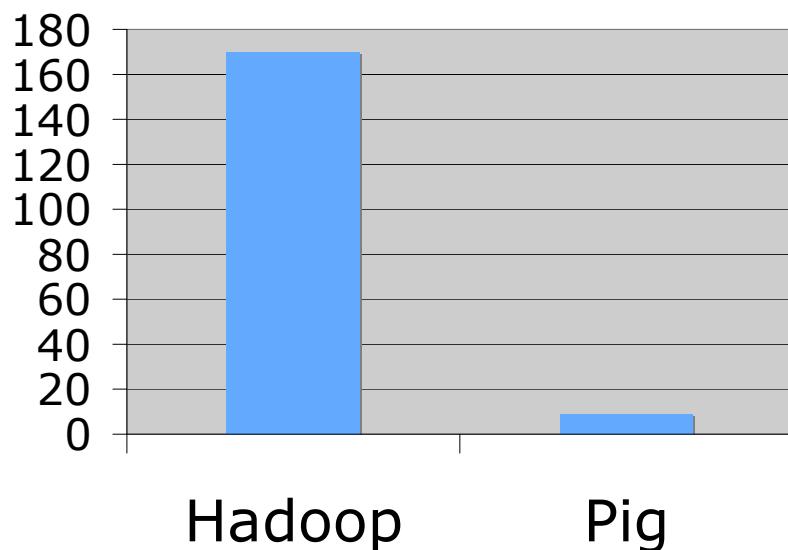
```
UserPageranks = foreach UserVisits generate user, AVG(VP.pagerank) as avgpr;
```

```
GoodUsers = filter UserPageranks by avgpr > '0.5';
```

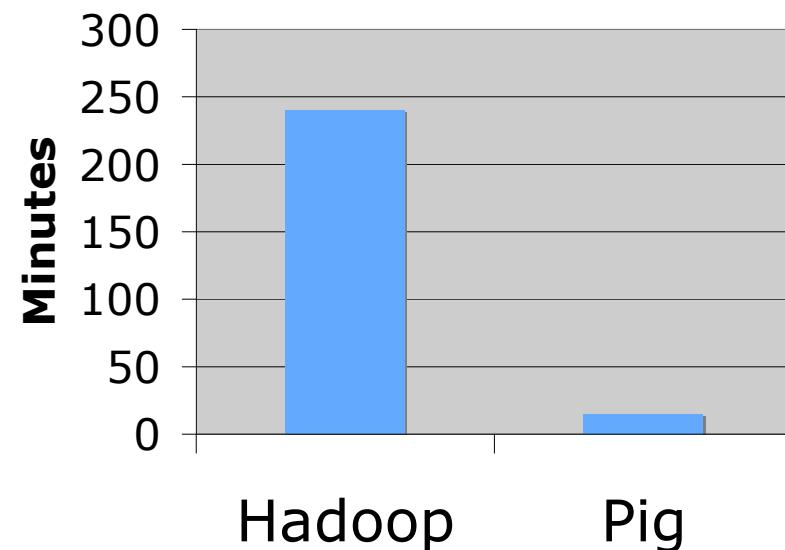
```
store GoodUsers into '/data/good_users';
```

# Java vs. Pig Latin

1/20 the lines of code



1/16 the development time



Performance on par with raw Hadoop!

# **Pig takes care of...**

- Schema and type checking
- Translating into efficient physical dataflow
  - (i.e., sequence of one or more MapReduce jobs)
- Exploiting data reduction opportunities
  - (e.g., early partial aggregation via a combiner)
- Executing the system-level dataflow
  - (i.e., running the MapReduce jobs)
- Tracking progress, errors, etc.

# **Hive + HBase?**

# Integration

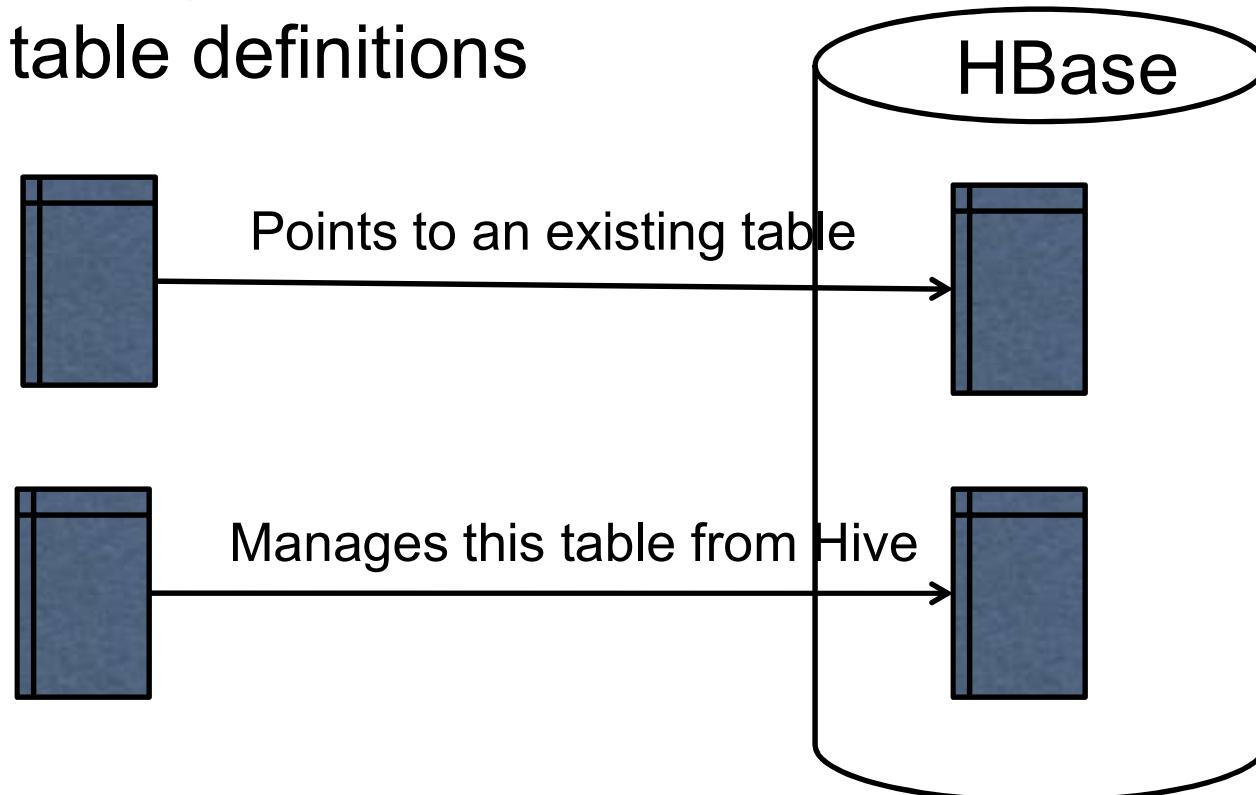
- Reasons to use Hive on HBase:
  - A lot of data sitting in HBase due to its usage in a real-time environment, but never used for analysis
  - Give access to data in HBase usually only queried through MapReduce to people that don't code (business analysts)
  - When needing a more flexible storage solution, so that rows can be updated live by either a Hive job or an application and can be seen immediately to the other
- Reasons not to do it:
  - Run SQL queries on HBase to answer live user requests (it's still a MR job)
  - Hoping to see interoperability with other SQL analytics systems

# Integration

- How it works:

- Hive can use tables that already exist in HBase or manage its own ones, but they still all reside in the same HBase instance

## Hive table definitions

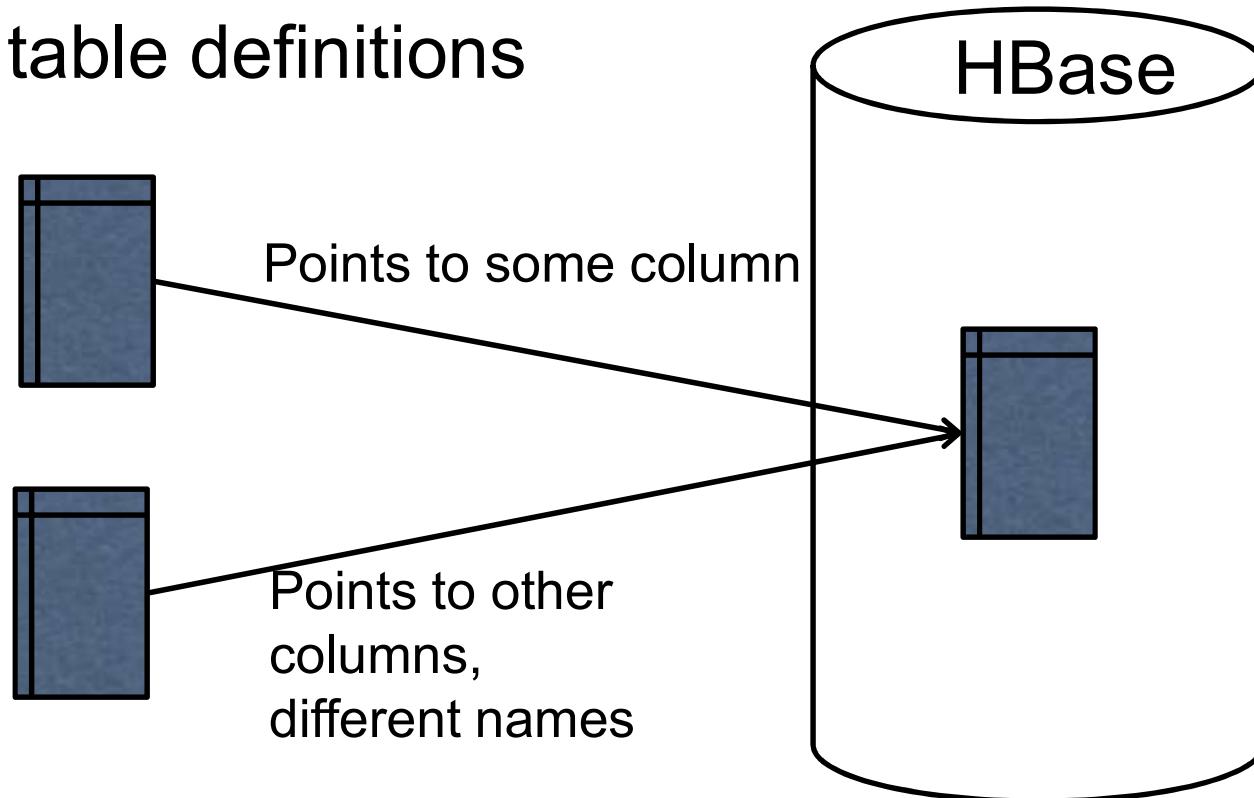


# Integration

- How it works:

- When using an already existing table, defined as EXTERNAL, you can create multiple Hive tables that point to it

## Hive table definitions



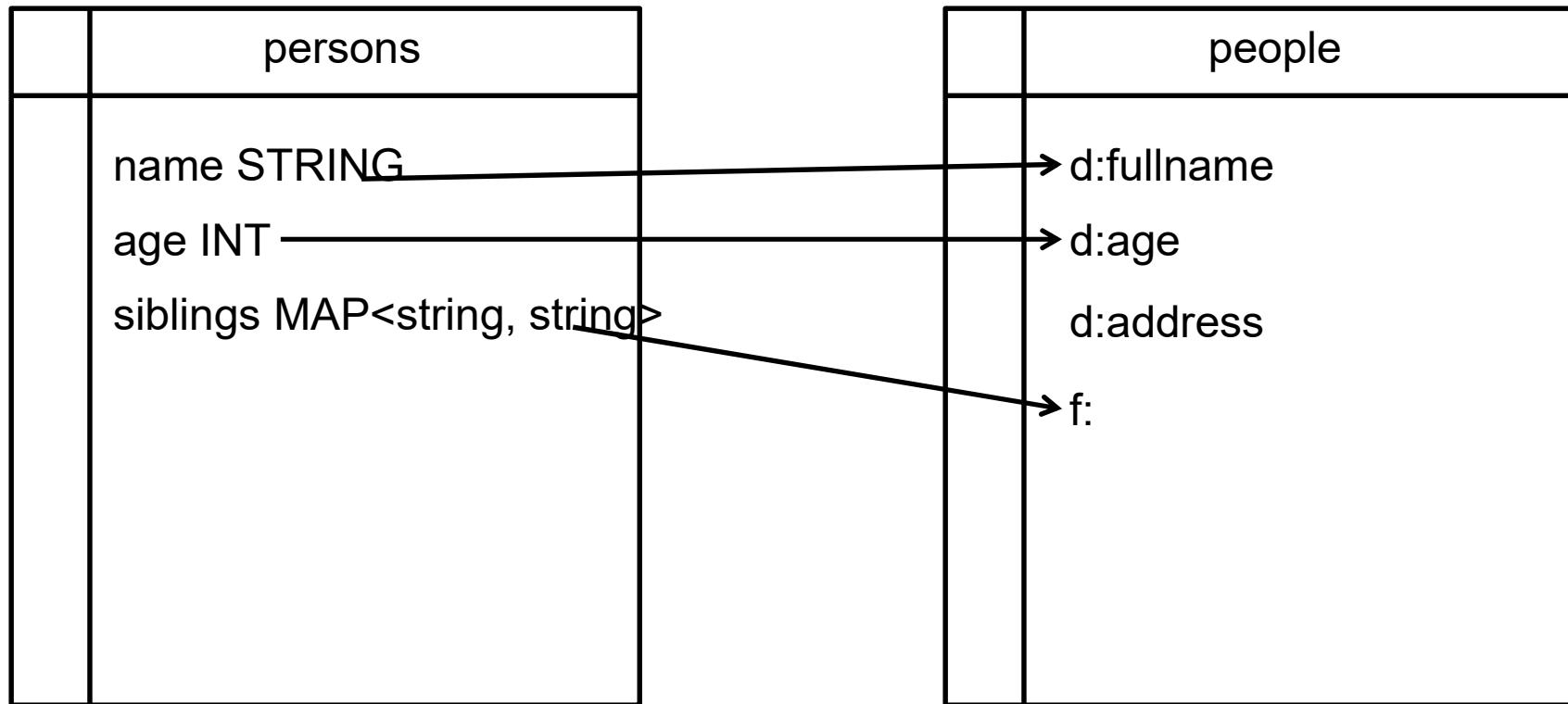
# Integration

- How it works:

- Columns are mapped however you want, changing names and giving types

Hive table definition

HBase table



# Integration

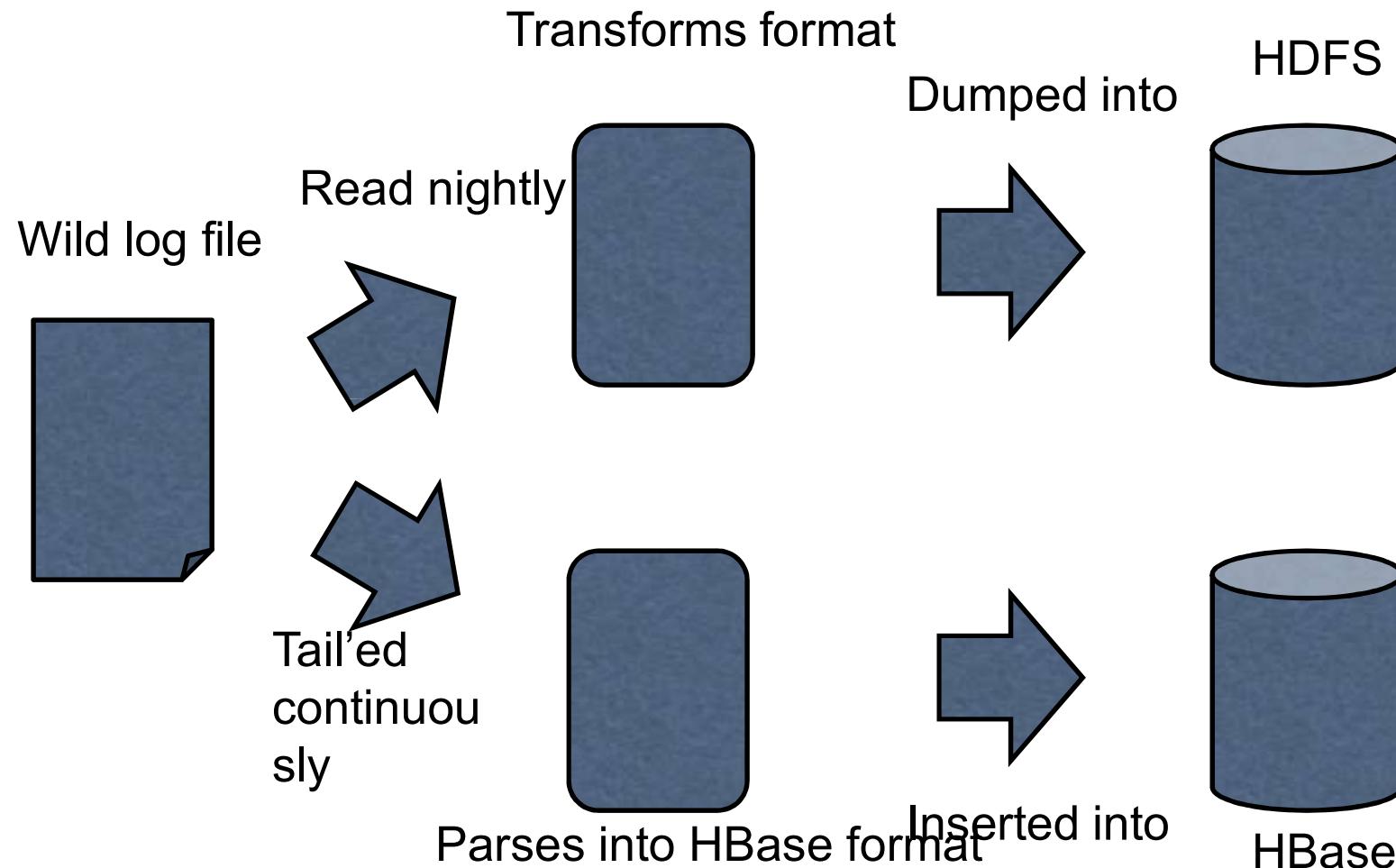
- Drawbacks (that can be fixed with brain juice):
  - Binary keys and values (like integers represented on 4 bytes) aren't supported since Hive prefers string representations, HIVE-1634
  - Compound row keys aren't supported, there's no way of using multiple parts of a key as different "fields"
  - This means that concatenated binary row keys are completely unusable, which is what people often use for HBase
  - Filters are done at Hive level instead of being pushed to the region servers
  - Partitions aren't supported

# Data Flows

- Data is being generated all over the place:
  - Apache logs
  - Application logs
  - MySQL clusters
  - HBase clusters

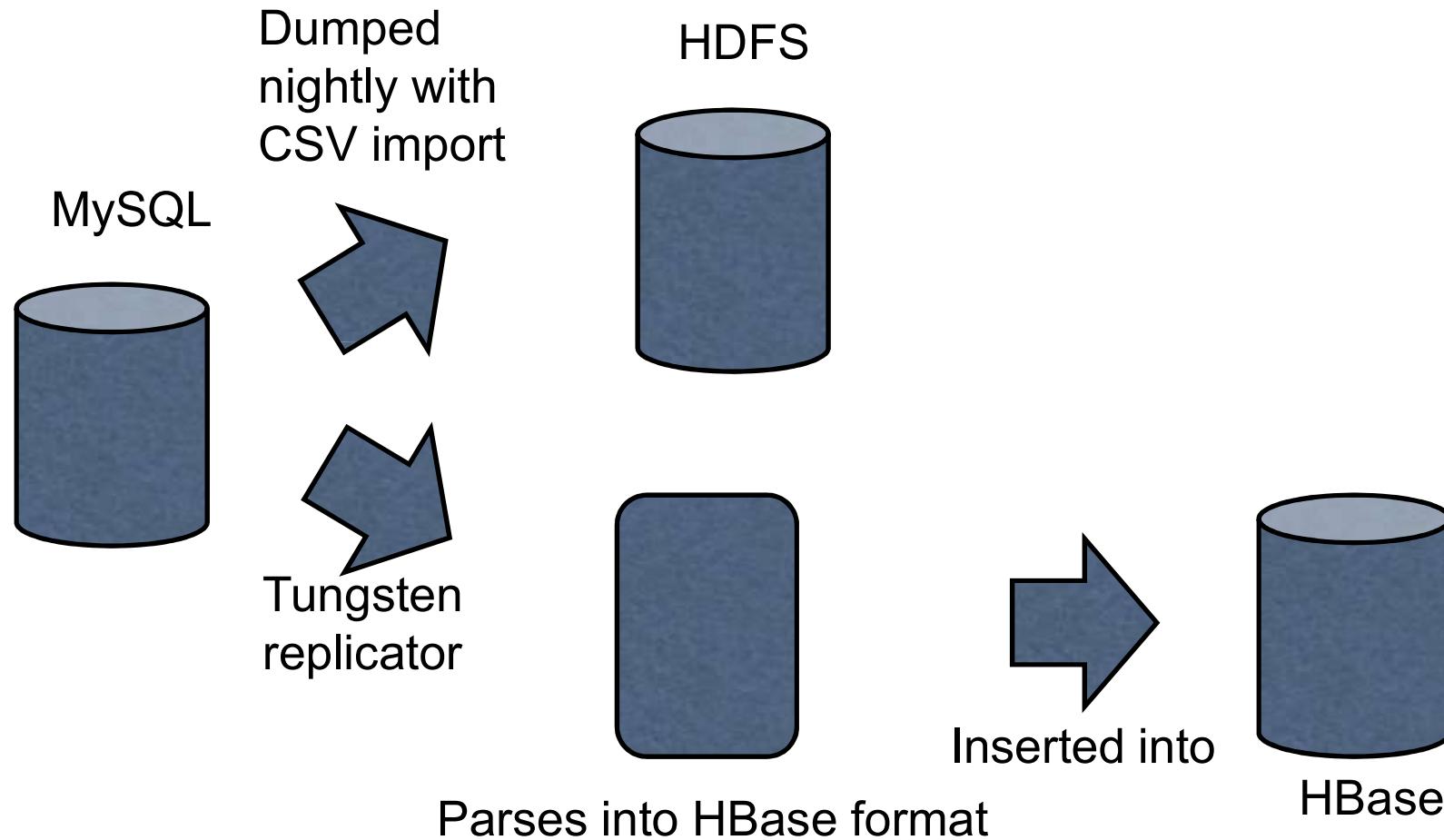
# Data Flows

- Moving application log files



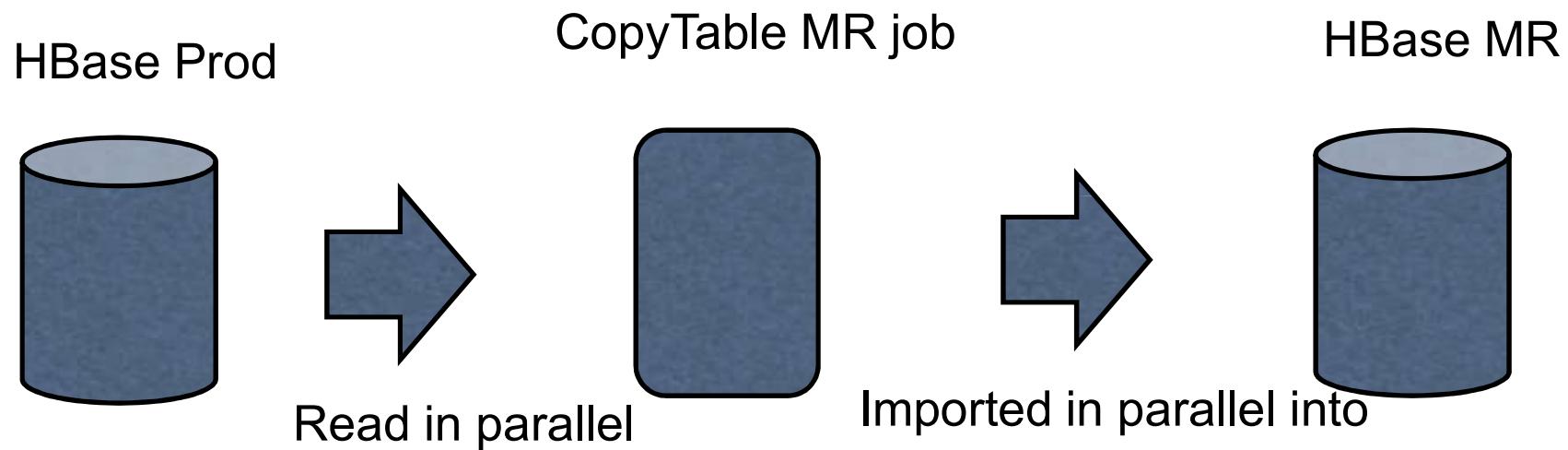
# Data Flows

- Moving MySQL data



# Data Flows

- Moving HBase data



# Use Cases

- Front-end engineers
  - They need some statistics regarding their latest product
- Research engineers
  - Ad-hoc queries on user data to validate some assumptions
  - Generating statistics about recommendation quality
- Business analysts
  - Statistics on growth and activity
  - Effectiveness of advertiser campaigns
  - Users' behavior VS past activities to determine, for example, why certain groups react better to email communications
  - Ad-hoc queries on stumbling behaviors of slices of the user base

# Use Cases

- Using a simple table in HBase:

```
CREATE EXTERNAL TABLE blocked_users(  
    userid INT,  
    blockee INT,  
    blocker INT,  
    created BIGINT)  
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'  
WITH SERDEPROPERTIES ("hbase.columns.mapping" =  
    ":key,f:blockee,f:blocker,f:created")  
TBLPROPERTIES("hbase.table.name" = "m2h_repl-userdb.stumble.blocked_users");
```

HBase is a special case here, it has a unique row key map with :key  
Not all the columns in the table need to be mapped

# Use Cases

- Using a complicated table in HBase:

```
CREATE EXTERNAL TABLE ratings_hbase(  
    userid INT,  
    created BIGINT,  
    urlid INT,  
    rating INT,  
    topic INT,  
    modified BIGINT)  
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'  
WITH SERDEPROPERTIES ("hbase.columns.mapping" =  
    ":key#b@0,:key#b@1,:key#b@2,default:rating#b,default:topic#b,default:modified#b")  
TBLPROPERTIES("hbase.table.name" = "ratings_by_userid");
```

#b means binary, @ means position in composite key (SU-specific hack)

# Graph Databases

# NEO4J (Graphbase)

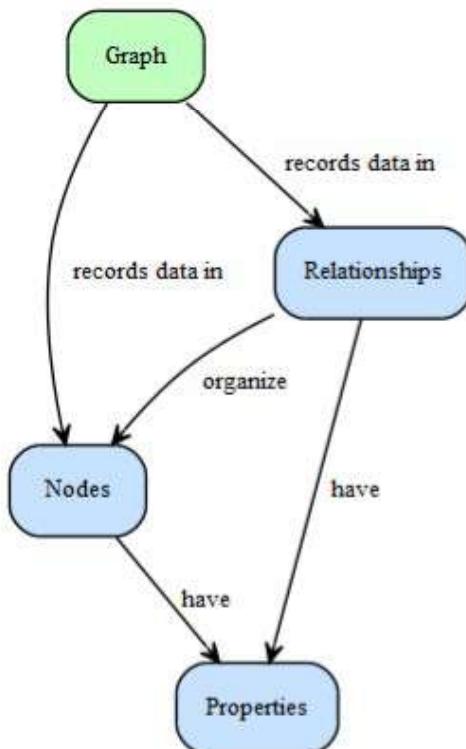
- A graph is a collection nodes (things) and edges (relationships) that connect pairs of nodes.
- Attach properties (key-value pairs) on nodes and relationships
- Relationships connect two nodes and both nodes and relationships can hold an arbitrary amount of key-value pairs.
- A graph database can be thought of as a key-value store, with full support for relationships.
- <http://neo4j.org/>

# NEO4J

## 2.1.1. A Graph contains Nodes and Relationships

"A Graph —records data in→ Nodes —which have→ Properties"

The simplest possible graph is a single Node, a record that has named values referred to as Properties. A Node could start with a single Property and grow to a few million, though that can get a little awkward. At some point it makes sense to distribute the data into multiple nodes, organized with explicit Relationships.



# NEO4J

## 2.1.2. Relationships organize the Graph

"Nodes → are organized by → Relationships → which also have → Properties"

Relationships organize Nodes into arbitrary structures, allowing a Graph to resemble a List, a Tree, a Map, or a compound Entity - any of which can be combined into yet more complex, richly inter-connected structures.

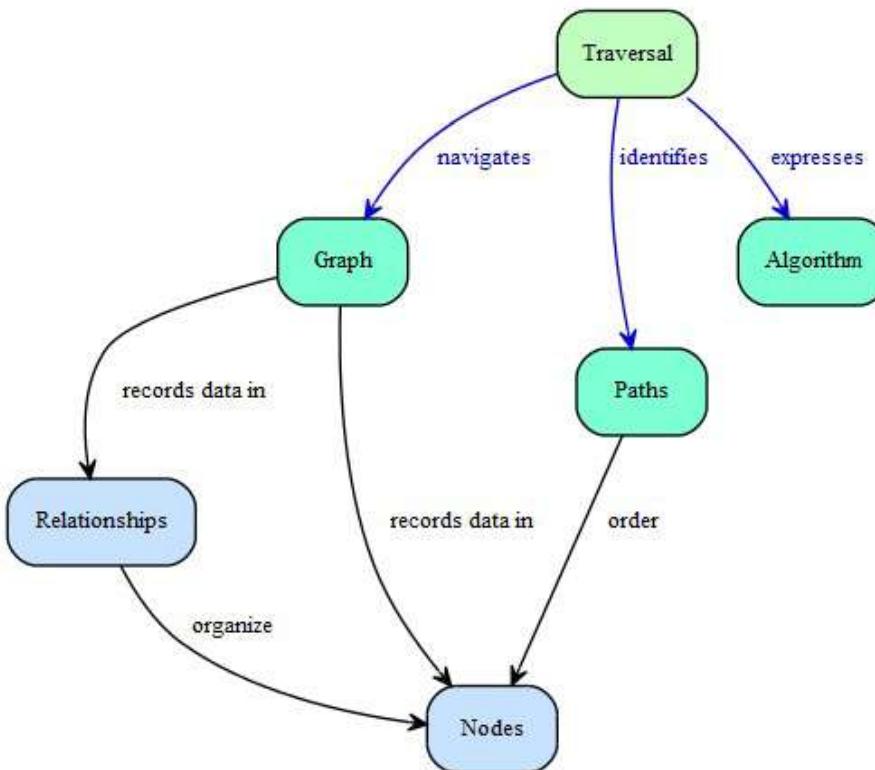


# NEO4J

## 2.1.3. Query a Graph with a Traversal

"A Traversal —navigates→ a Graph; it —identifies→ Paths —which order→ Nodes"

A Traversal is how you query a Graph, navigating from starting Nodes to related Nodes according to an algorithm, finding answers to questions like "what music do my friends like that I don't yet own," or "if this power supply goes down, what web services are affected?"

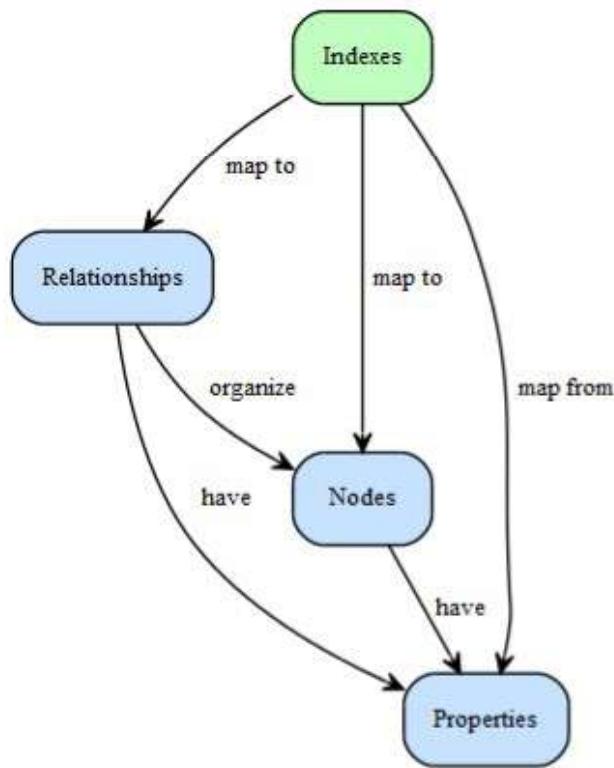


# NEO4J

## 2.1.4. Indexes look-up Nodes or Relationships

"An Index —maps from→ Properties —to either→ Nodes or Relationships"

Often, you want to find a specific Node or Relationship according to a Property it has. Rather than traversing the entire graph, use an Index to perform a look-up, for questions like "find the Account for username master-of-graphs."

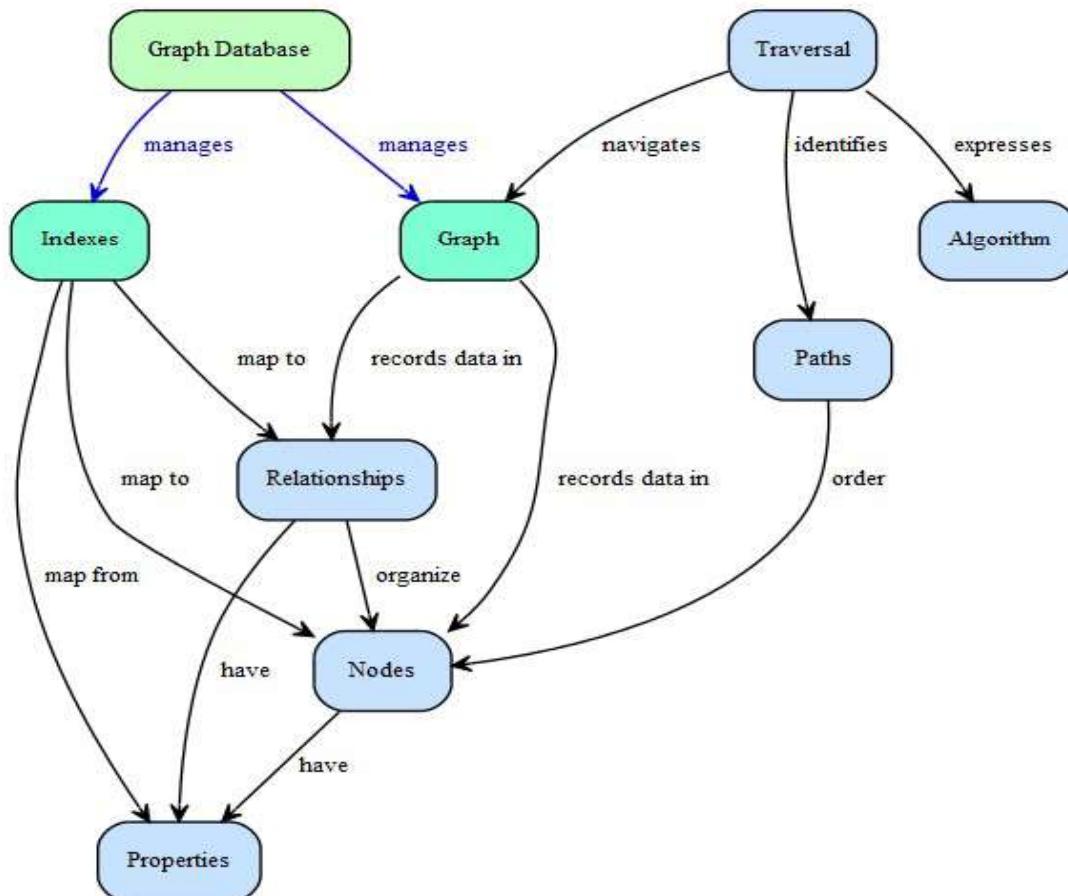


# NEO4J

## 2.1.5. Neo4j is a Graph Database

"A Graph Database —manages a→ Graph and —also manages related→ Indexes"

Neo4j is a commercially supported open-source graph database. It was designed and built from the ground-up to be a reliable database, optimized for graph structures instead of tables. Working with Neo4j, your application gets all the expressiveness of a graph, with all the dependability you expect out of a database.



# NEO4J

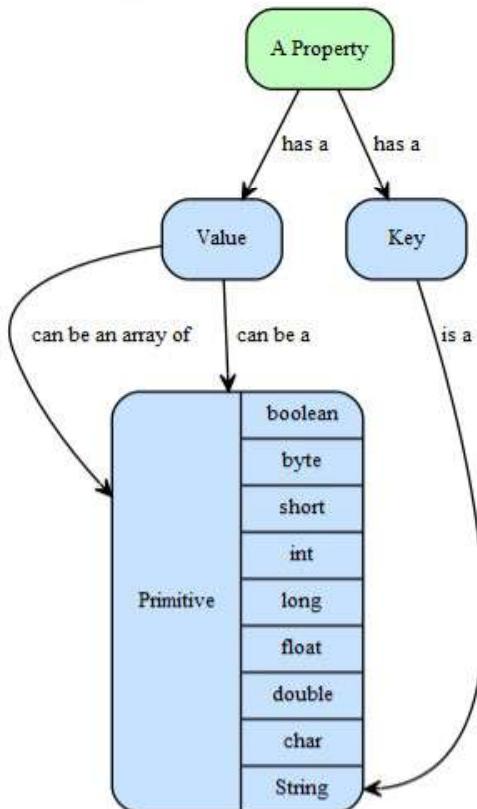
## Properties

Properties are key-value pairs where the key is a string. Property values can be either a primitive or an array of one primitive type. For example `string`, `int` and `int[]` values are valid for properties.



### Note

`null` is not a valid property value. Nulls can instead be modeled by the absence of a key.



# NEO4J Features

- Dual license: open source and commercial
- Well suited for many web use cases such as tagging, metadata annotations, social networks, wikis and other network-shaped or hierarchical data sets
- Intuitive graph-oriented model for data representation. Instead of static and rigid tables, rows and columns, you work with a flexible graph network consisting of nodes, relationships and properties.
- Neo4j offers performance improvements on the order of 1000x or more compared to relational DBs.
- A disk-based, native storage manager completely optimized for storing graph structures for maximum performance and scalability
- Massive scalability. Neo4j can handle graphs of several billion nodes/relationships/properties on a single machine and can be sharded to scale out across multiple machines
- Fully transactional like a real database
- Neo4j traverses depths of 1000 levels and beyond at millisecond speed.  
(many orders of magnitude faster than relational systems)

# Spark SQL: Relational Data Processing in Spark

Michael Armbrust, Reynold Xin, Cheng Lian, Yin Huai,  
Davies Liu, Joseph K. Bradley, Xiangrui Meng, Tomer  
Kaftan, Michael J. Franklin, Ali Ghodsi, Matei Zaharia

SIGMOD 2015 – Melbourne, Australia

Presented by Doris Xin

Based on slides provided by M. Armbrust



# Challenges and Solutions

## Challenges

- Perform ETL to and from various (semi- or unstructured) data sources
- Perform advanced analytics (e.g. machine learning, graph processing) that are hard to express in relational systems.

## Solutions

- A *DataFrame* API that can perform relational operations on both external data sources and Spark's built-in RDDs.
- A highly extensible optimizer, *Catalyst*, that uses features of Scala to add composable rule, control code gen., and define extensions.

# What is Apache Spark?

Fast and general cluster computing system,  
interoperable with Hadoop, included in all major  
distros

Improves efficiency through:

- In-memory computing primitives
- General computation graphs

→ Up to 100× faster  
(2-10× on disk)

Improves usability through:

- Rich APIs in Scala, Java, Python
- Interactive shell

→ 2-5× less code

# Spark Model

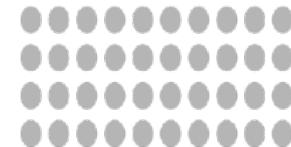
*Write programs in terms of transformations on distributed datasets*

## Resilient Distributed Datasets (RDDs)

- Collections of objects that can be stored in memory or disk across a cluster
- Parallel functional transformations (map, filter, ...)
- Automatically rebuilt on failure

# On-Disk Sort Record: Time to sort 100TB

2013 Record: 2100 machines  
Hadoop



72 minutes



2014  
Record:  
Spark

207  
machines

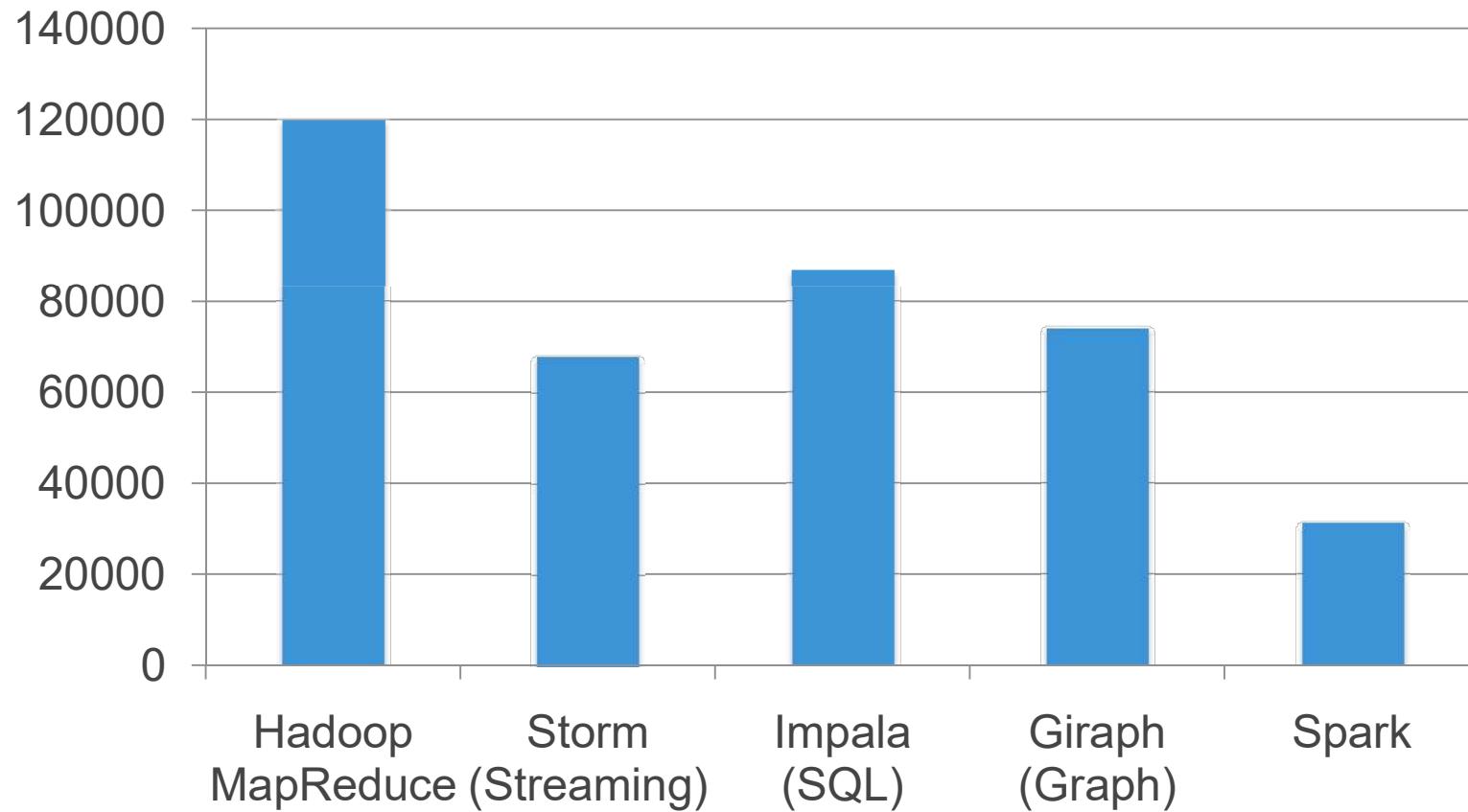


23 minutes



Also sorted 1PB in 4 hours

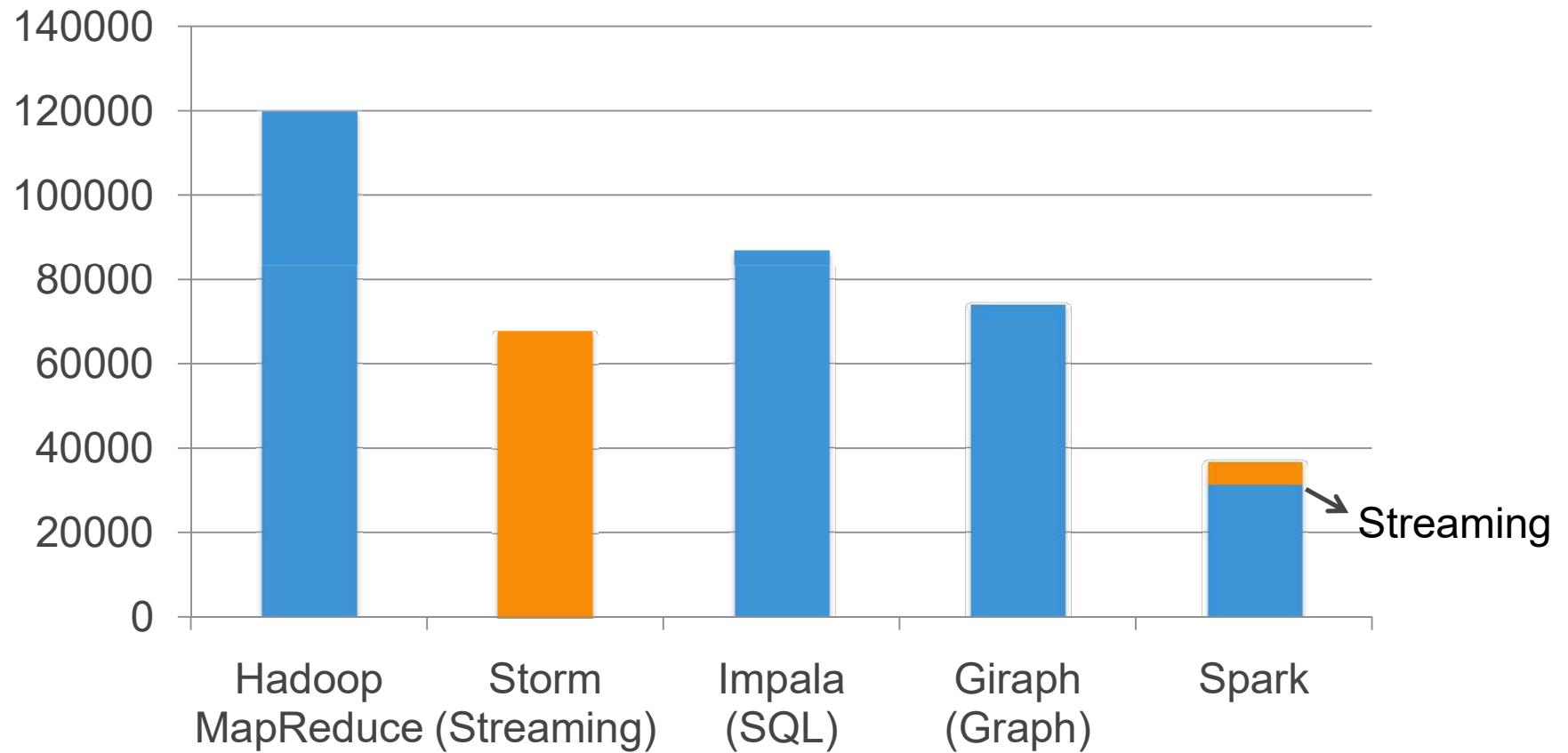
# Powerful Stack – Agile Development



non-test, non-example source lines



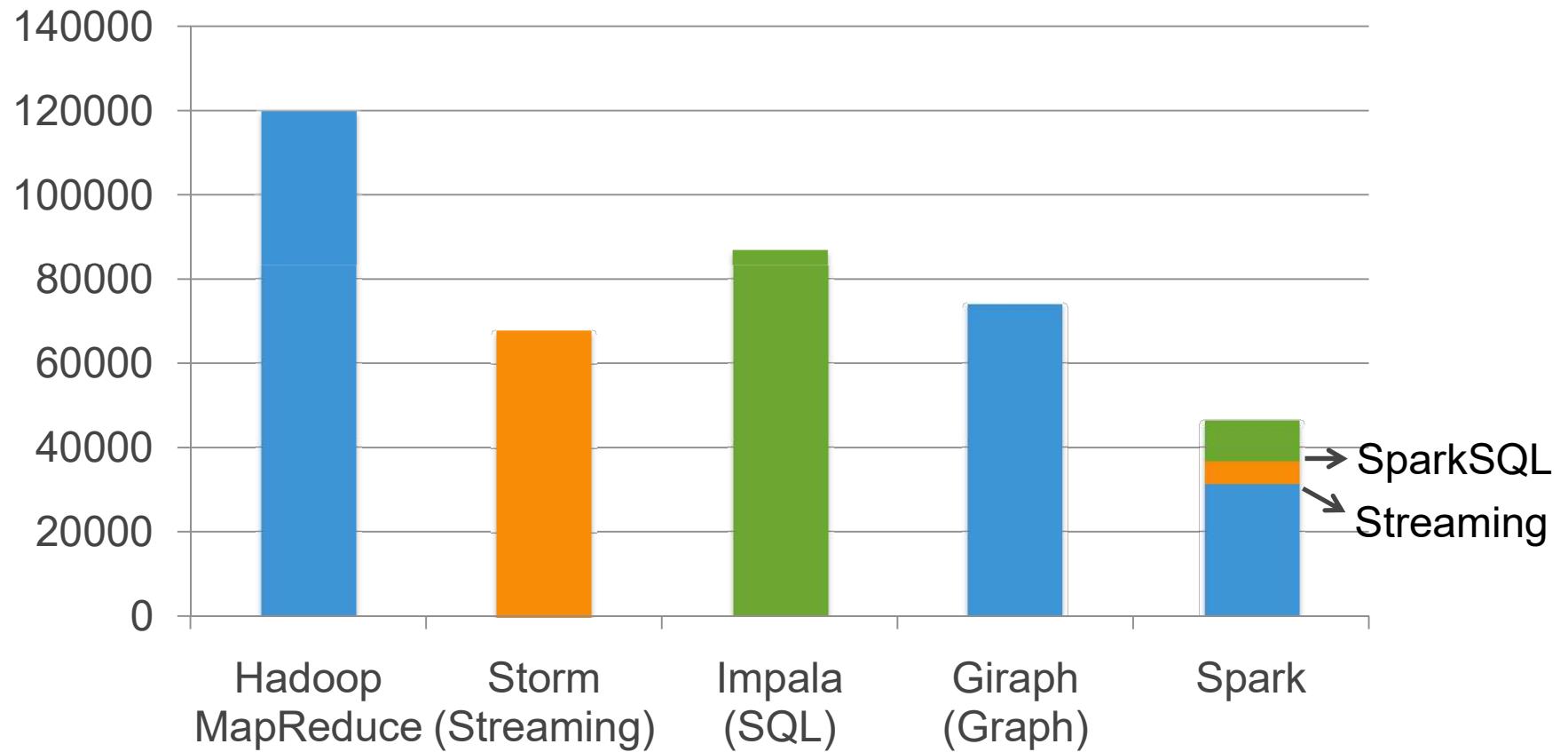
# Powerful Stack – Agile Development



non-test, non-example source lines



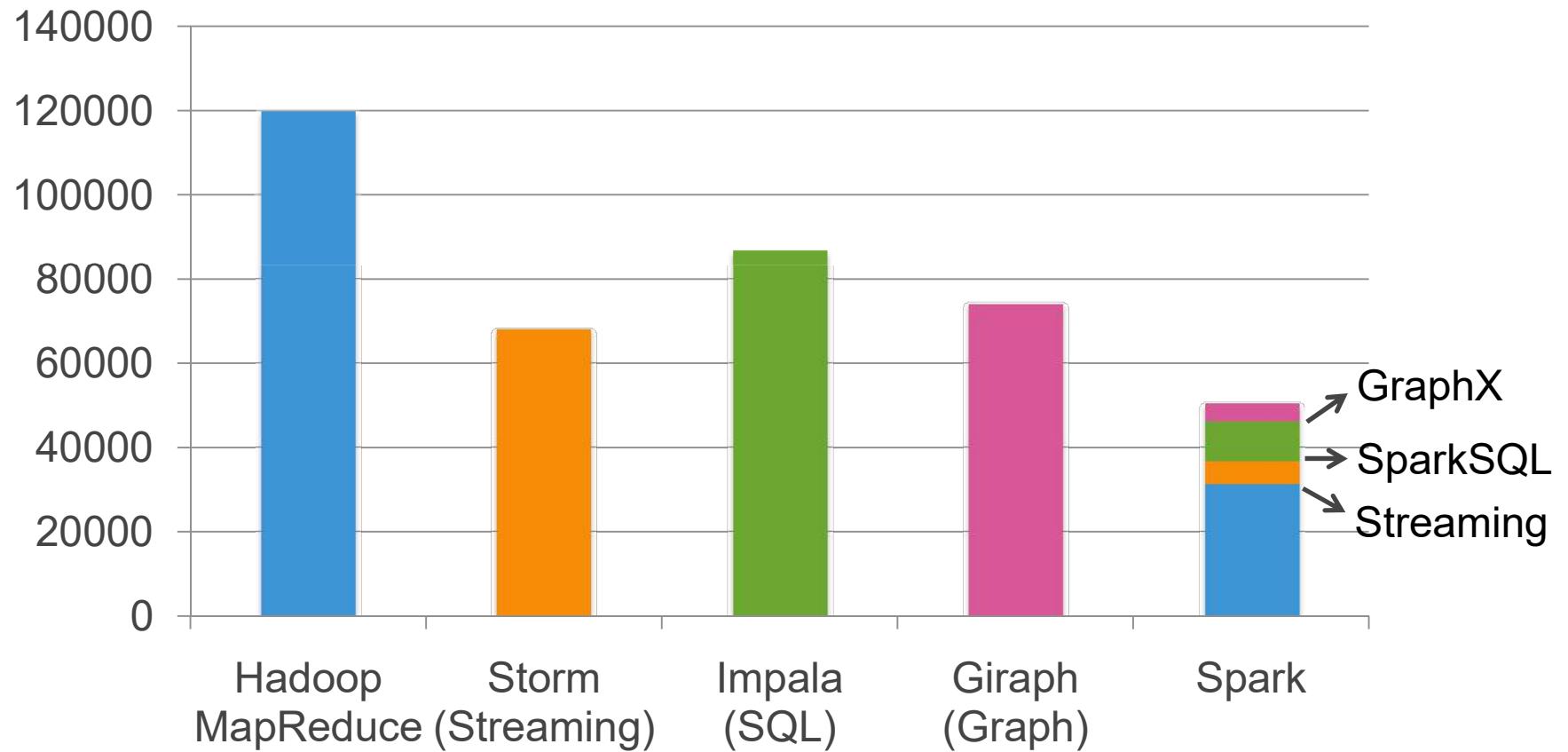
# Powerful Stack – Agile Development



non-test, non-example source lines

 databricks™

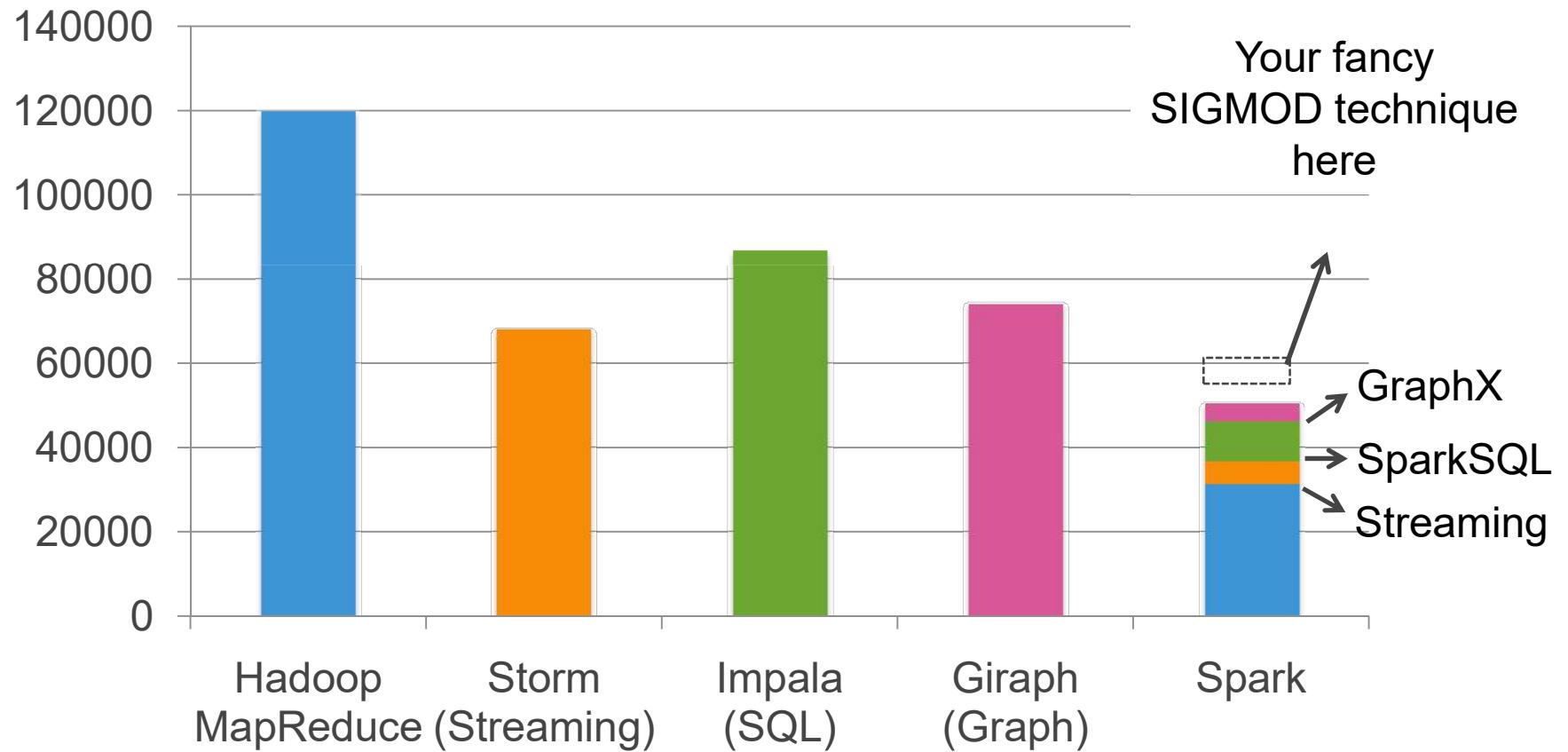
# Powerful Stack – Agile Development



non-test, non-example source lines



# Powerful Stack – Agile Development



non-test, non-example source lines

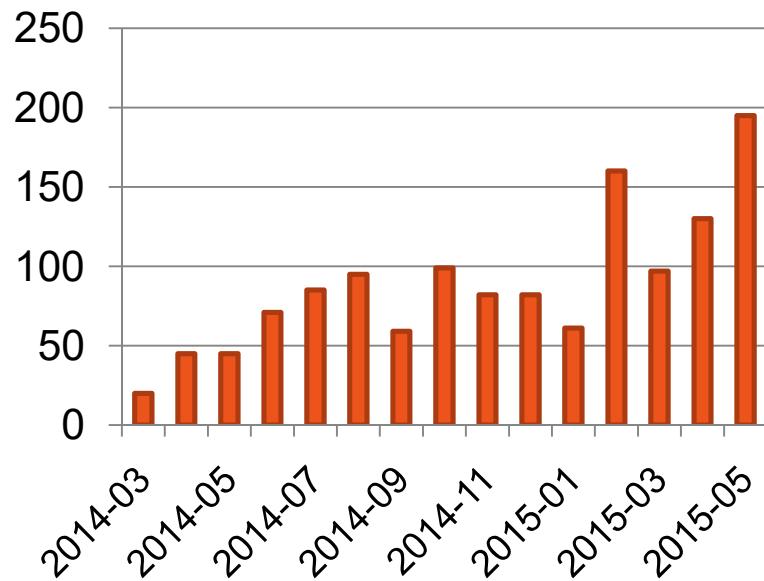
 databricks™

# About Spark SQL

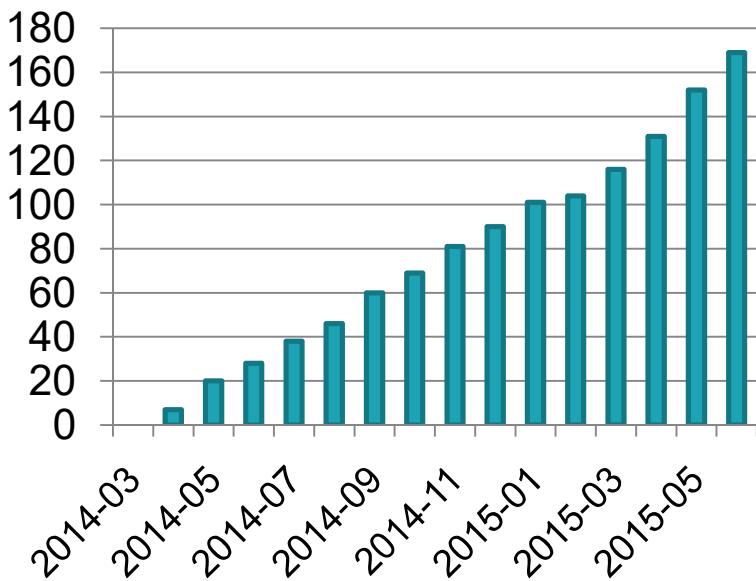
## Spark SQL

- Part of the core distribution since Spark 1.0 (April 2014)

# Of Commits Per Month



# of Contributors



# About SparkSQL

## Spark SQL

- Part of the core distribution since Spark 1.0 (April 2014)
- Runs SQL / HiveQL queries, optionally alongside or replacing existing Hive deployments



```
SELECT COUNT(*)  
FROM hiveTable  
WHERE hive_udf(data)
```

# Improvement upon Existing Art



Engine does not understand the structure of the data in RDDs or the semantics of user functions → limited optimization.



Can only be used to query external data in Hive catalog  
→ limited data sources  
  
Can only be invoked via SQL string from Spark → error prone  
  
Hive optimizer tailored for MapReduce → difficult to extend

# Programming Interface

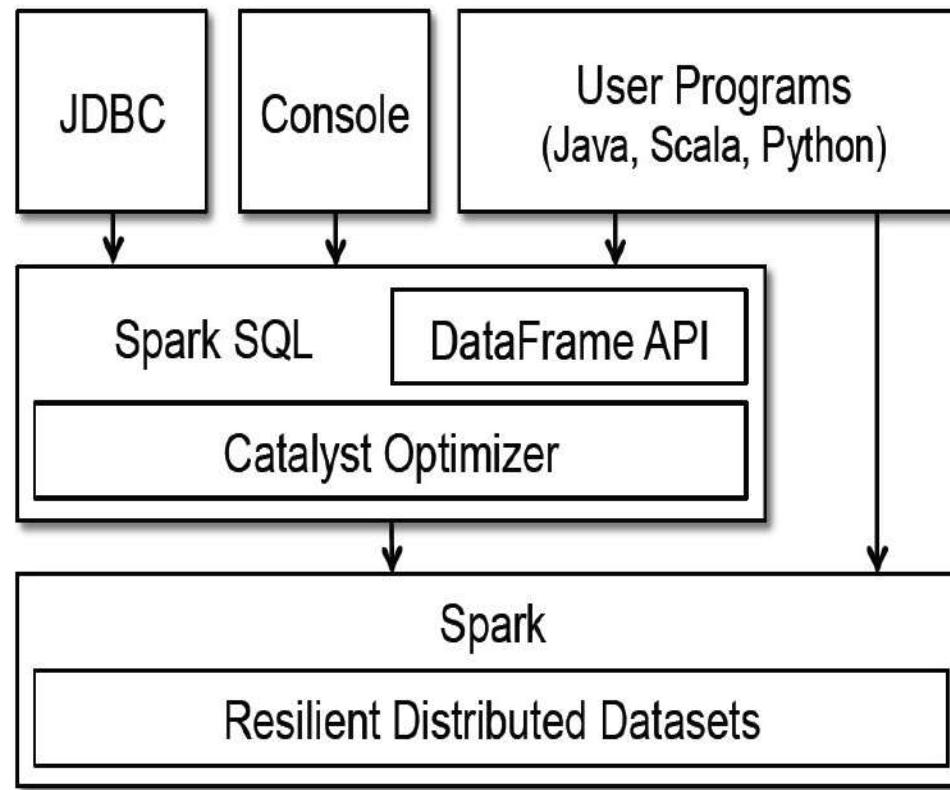


Figure 1: Interfaces to Spark SQL, and interaction with Spark.

```
ctx = new HiveContext()
users = ctx.table("users")
young = users.where(users("age") < 21)
println(young.count())
```

- A distributed collection of rows with the same schema (RDDs suffer from type erasure)
- Can be constructed from external data sources or RDDs into essentially an RDD of Row objects (SchemaRDDs as of Spark < 1.3)
- Supports relational operators (e.g. *where*, *groupby*) as well as Spark operations.
- Evaluated lazily → unmaterialized *logical* plan

# Data Model

- Nested data model
- Supports both primitive SQL types (boolean, integer, double, decimal, string, data, timestamp) and complex types (structs, arrays, maps, and unions); also user defined types.
- First class support for complex data types

# DataFrame Operations

- Relational operations (select, where, join, groupBy) via a DSL
- Operators take *expression* objects
- Operators build up an abstract syntax tree (AST), which is then optimized by *Catalyst*.

```
employees
    .join(dept, employees("deptId") === dept("id"))
    .where(employees("gender") === "female")           |
    .groupBy(dept("id"), dept("name"))
    .agg(count("name"))
```

• **Alternat  
traditior**

```
users.where(users("age") < 21)
    .registerTempTable("young")
ctx.sql("SELECT count(*), avg(age) FROM young")
```

# Advantages over Relational Query Languages

- Holistic optimization across functions composed in different languages.
- Control structures (e.g. *if, for*)
- Logical plan analyzed *eagerly* → identify code errors associated with data *schema* issues on the fly.

# Querying Native Datasets

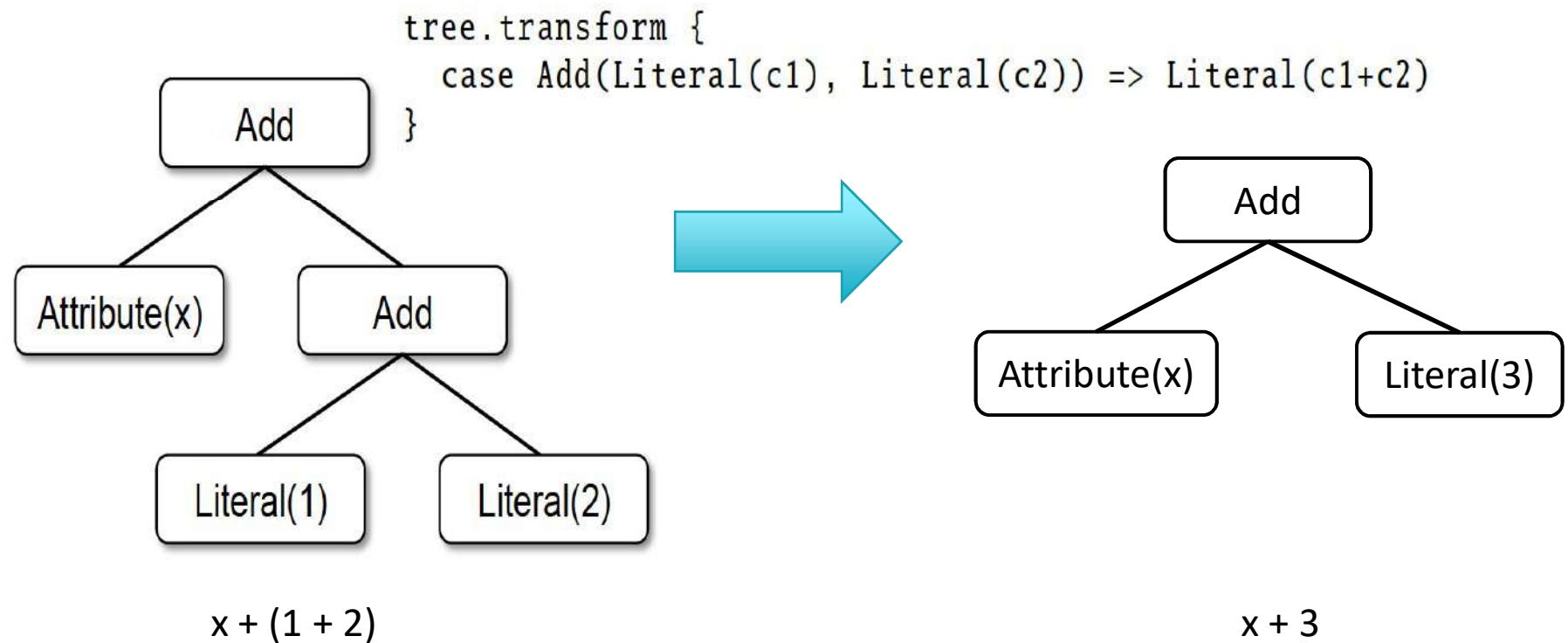
- Infer column names and types directly from data objects (via reflection in Java and Scala and data sampling in Python, which is dynamically typed)
  - Native case class `User(name: String, age: Int)` expensive data format transformation.
  - Benefits:
    - Run relational operations on existing Spark programs
    - Combine RDDs with external structured data
- Columnar storage  
with *hot* columns  
cached in memory

# User-Defined Functions (UDFs)

- Easy extension of limited operations supported.
- Allows inline registration of UDFs
  - Compare with Pig, which requires the UDF to be written in a Java package that's loaded into the Pig script.
- Can be defined on simple data types or entire tables.
- UDFs available to other interfaces after registration

```
val model: LogisticRegressionModel = ...  
  
ctx.udf.register("predict",  
  (x: Float, y: Float) => model.predict(Vector(x, y)))  
  
ctx.sql("SELECT predict(age, weight) FROM users")
```

# Catalyst



# Prior Work: Optimizer Generators

## Volcano / Cascades:

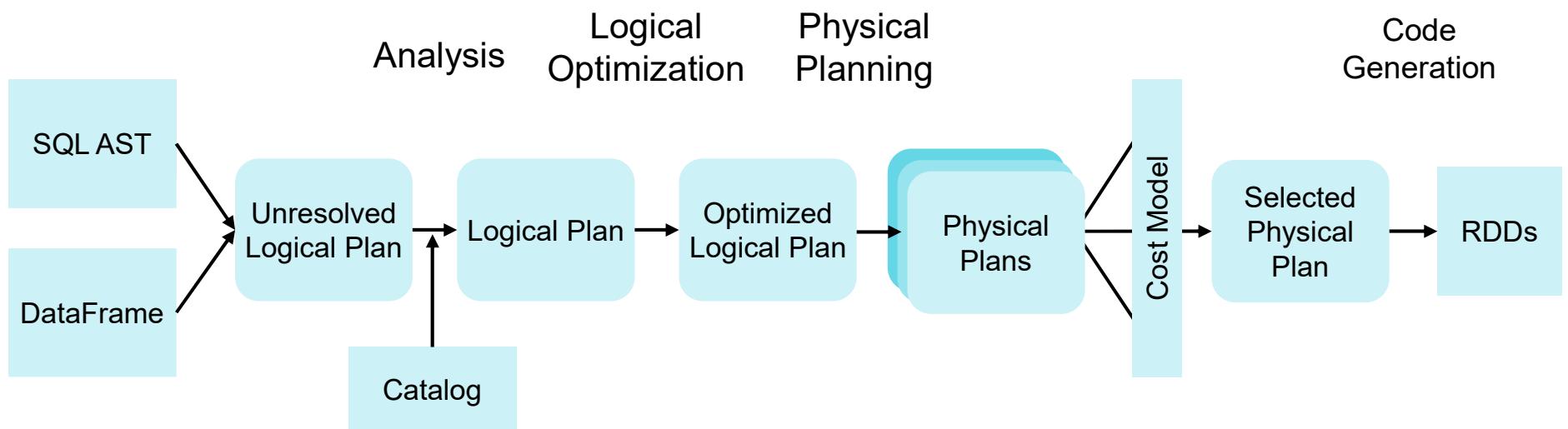
- Create a custom language for expressing rules that rewrite trees of relational operators.
- Build a compiler that generates executable code for these rules.

Cons: Developers need to learn this custom language. Language might not be powerful enough.

# Catalyst Rules

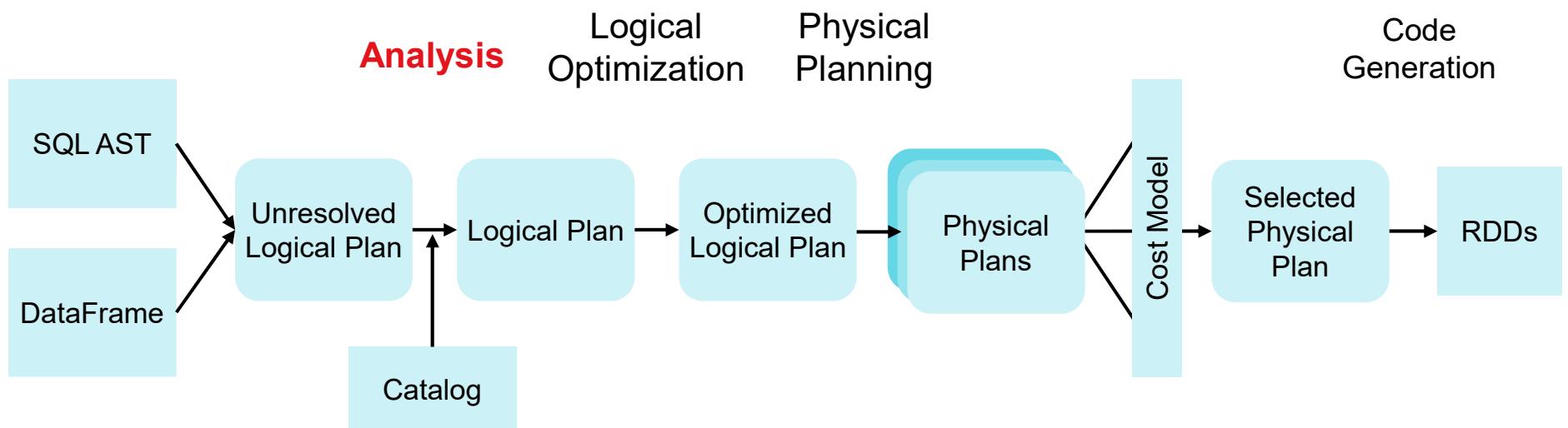
- *Pattern matching* functions that transform subtrees into specific structures.
  - *Partial function*—skip over subtrees that do not match → no need to modify existing rules when adding new types of operators.
- Multiple patterns in the same *transform* call.
- May take multiple *batches* to reach a *fixed point*.
- *transform* can contain arbitrary Scala code.

# Plan Optimization & Execution



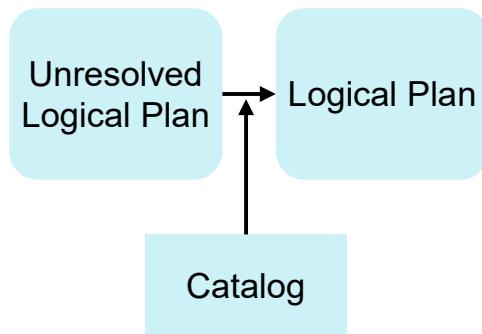
DataFrames and SQL share the same optimization/execution pipeline

# Plan Optimization & Execution



DataFrames and SQL share the same optimization/execution pipeline

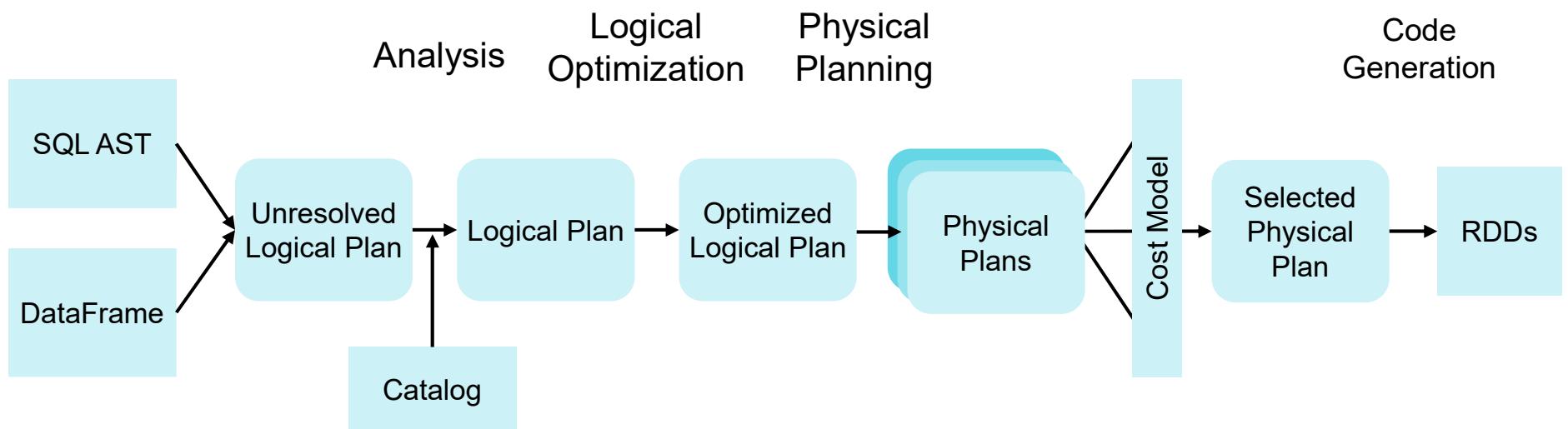
## Analysis



- An attribute is *unresolved* if its type is not known or it's not matched to an input table.
- To resolve attributes:
  - Look up relations by name from the catalog.
  - Map named attributes to the input provided given operator's children.
  - UID for references to the same value
  - Propagate and coerce types through expressions (e.g.  $1 + col$ )

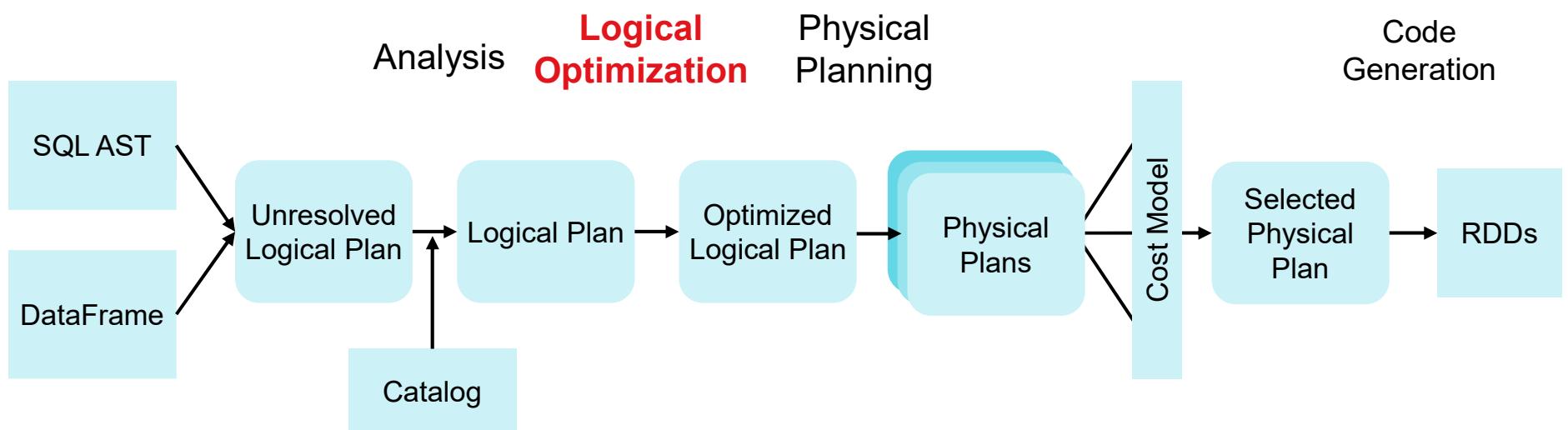
`SELECT col FROM sales`

# Plan Optimization & Execution



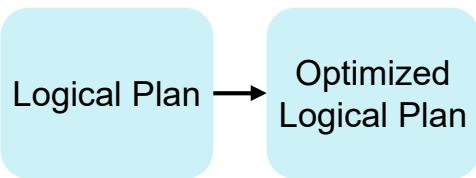
DataFrames and SQL share the same optimization/execution pipeline

# Plan Optimization & Execution



DataFrames and SQL share the same optimization/execution pipeline

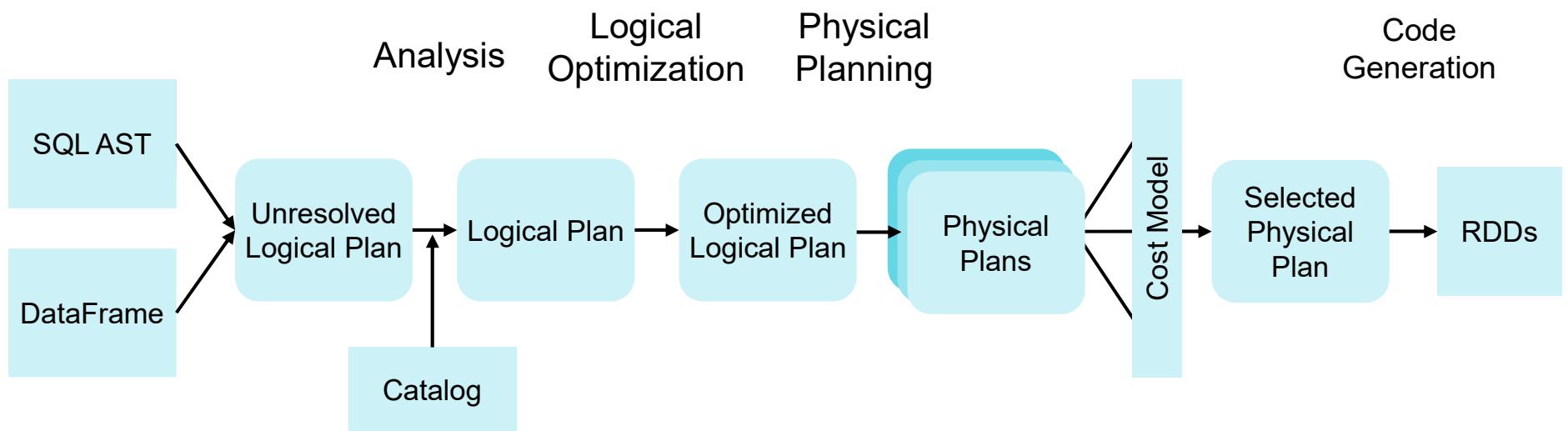
## Logical Optimization



- Applies standard rule-based optimization (constant folding, predicate-pushdown, projection pruning, null propagation, boolean expression simplification, etc)
- 800LOC

```
object DecimalAggregates extends Rule[LogicalPlan] {  
    /** Maximum number of decimal digits in a Long */  
    val MAX_LONG_DIGITS = 18  
  
    def apply(plan: LogicalPlan): LogicalPlan = {  
        plan transformAllExpressions {  
            case Sum(e @ DecimalType.Expression(prec, scale))  
                if prec + 10 <= MAX_LONG_DIGITS =>  
                    MakeDecimal(Sum(LongValue(e)), prec + 10, scale)  
            }  
    }  
}
```

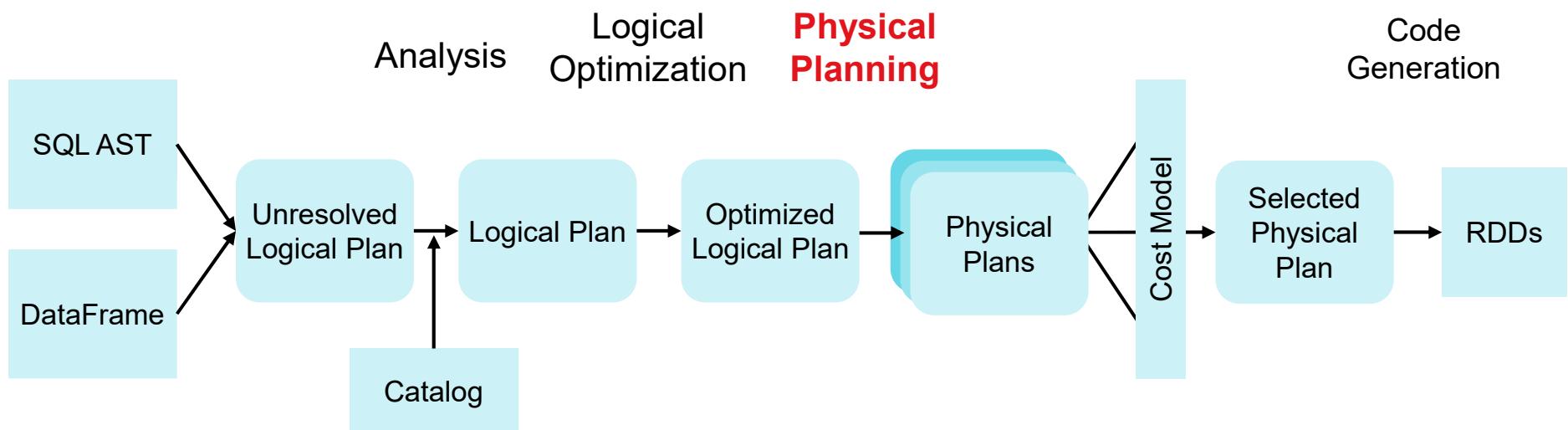
# Plan Optimization & Execution



DataFrames and SQL share the same optimization/execution pipeline

# Plan Optimization & Execution

e.g. Pipeline projections  
and filters into a single *map*



DataFrames and SQL share the same optimization/execution pipeline

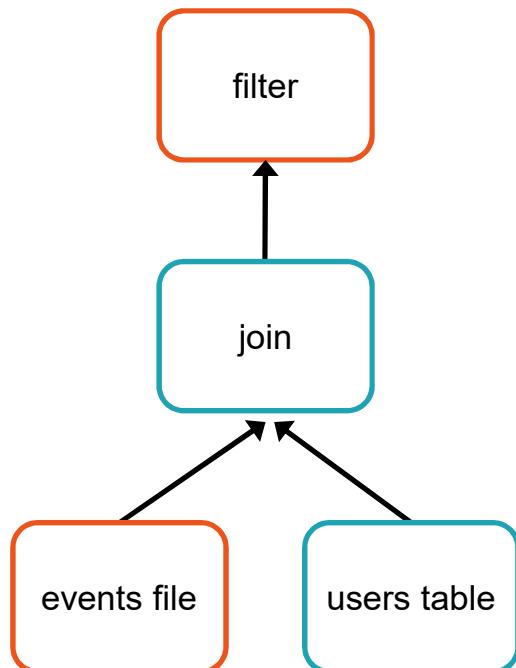
```

def add_demographics(events):
    u = sqlCtx.table("users")                                # Load partitioned Hive table
    events \
        .join(u, events.user_id == u.user_id) \                # Join on user_id
        .withColumn("city", zipToCity(df.zip))                 # Run udf to add city column

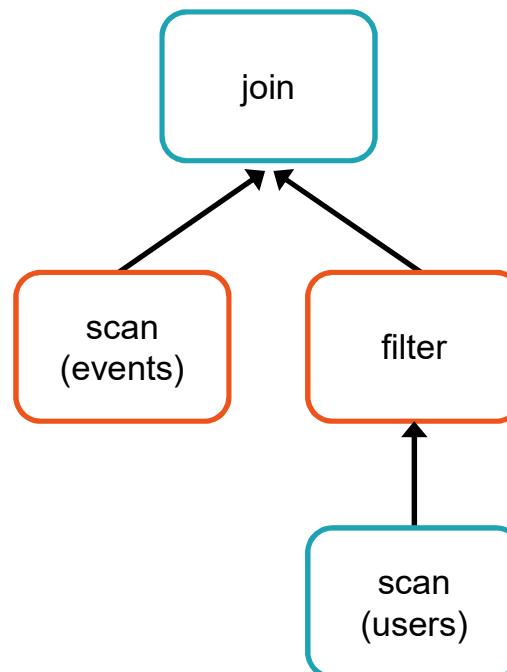
events = add_demographics(sqlCtx.load("/data/events", "parquet"))
training_data = events.where(events.city == "Melbourne").select(events.timestamp).collect()

```

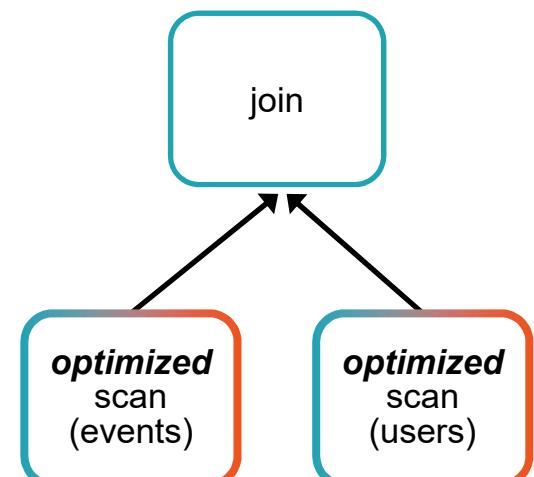
Logical Plan



Physical Plan

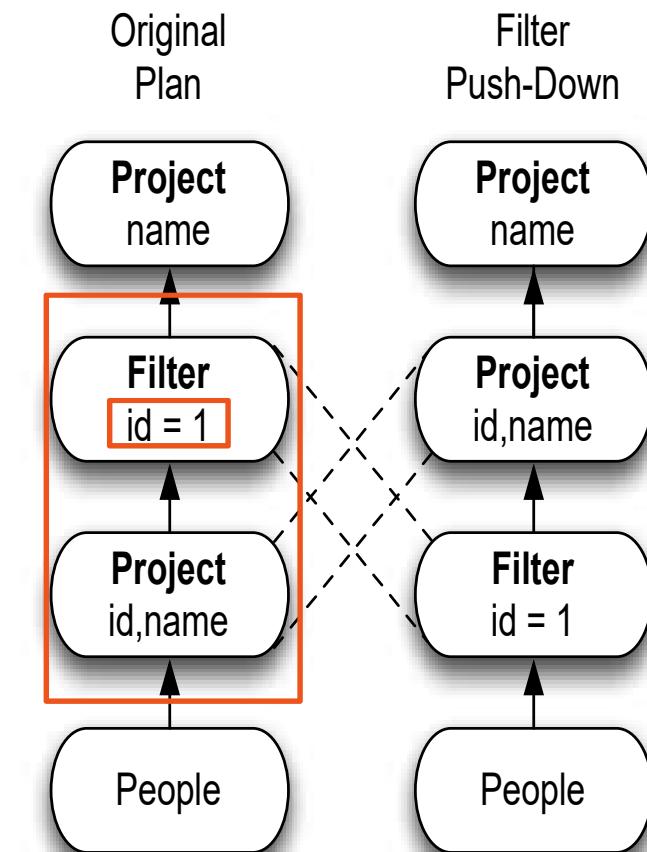


Physical Plan  
with Predicate Pushdown  
and Column Pruning

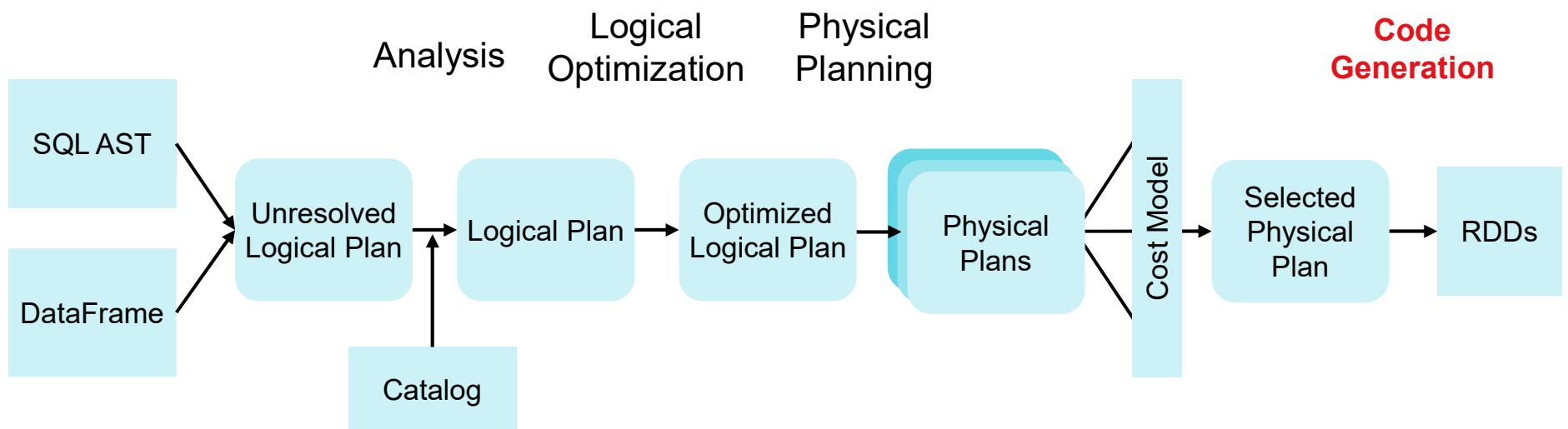


# An Example Catalyst Transformation

1. Find filters on top of projections.
2. Check that the filter can be evaluated without the result of the project.
3. If so, switch the operators.



# Plan Optimization & Execution



DataFrames and SQL share the same optimization/execution pipeline

# Code Generation

```
def compile(node: Node): AST = node match {  
  case Literal(value) => q"$value"  
  case Attribute(name) => q"row.get($name)"  
  case Add(left, right) =>  
    q"${compile(left)} + ${compile(right)}"  
}  
}
```

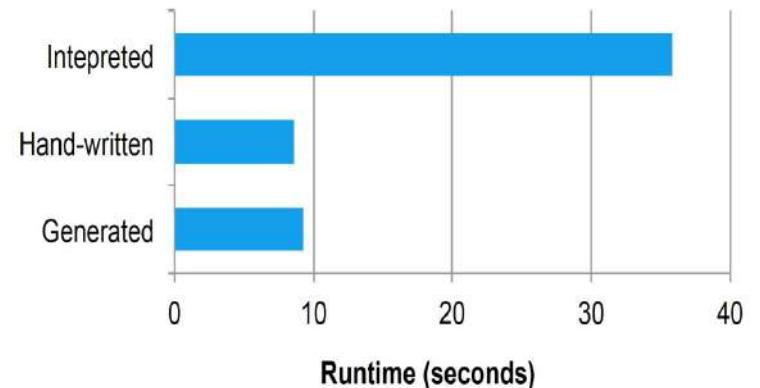


Figure 4: A comparision of the performance evaluating the expression  $x+x+x$ , where  $x$  is an integer, 1 billion times.

- Relies on Scala's *quasiquotes* to simplify code gen.
- Catalyst transforms a SQL tree into an abstract syntax tree (AST) for Scala code to eval expr and generate code
- 700LOC

# Extensions

## Data Sources

- must implement a *createRelation* function that takes a set of key-value params and returns a *BaseRelation* object.
- E.g. CSV, Avro, Parquet, JDBC

## User-Defined Types (UDTs)

- Map user-defined types to structures composed of Catalyst's built-in types.

```
class PointUDT extends UserDefinedType[Point] {  
    def dataType = StructType(Seq( // Our native structure  
        StructField("x", DoubleType),  
        StructField("y", DoubleType)  
    ))  
    def serialize(p: Point) = Row(p.x, p.y)  
    def deserialize(r: Row) =  
        Point(r.getDouble(0), r.getDouble(1))  
}
```

# Advanced Analytics Features

## Schema Inference for Semistructured Data

### JSON

- Automatically infers schema from a set of records, in one pass or sample
- A tree of STRUCT types, each of which may contain atoms, arrays, or other STRUCTs.
- Find the most appropriate type for a field based on all data observed in that column. Determine array element types in the same way.
- Merge schemata of single records in one *reduce* operation.
- Same trick for Python typing

```
{  
  "text": "This is a tweet about #Spark",  
  "tags": ["#Spark"],  
  "loc": {"lat": 45.1, "long": 90}  
}
```

```
{  
  "text": "This is another tweet",  
  "tags": [],  
  "loc": {"lat": 39, "long": 88.5}  
}
```

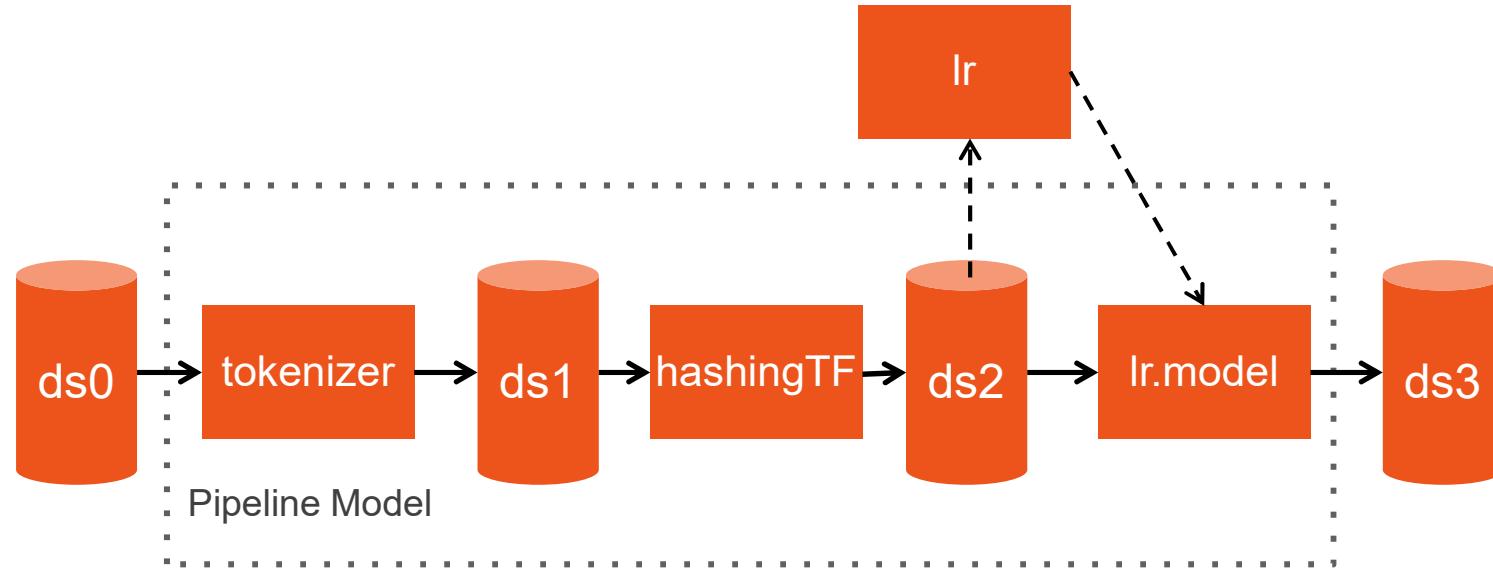
```
{  
  "text": "A #tweet without #location",  
  "tags": ["#tweet", "#location"]  
}
```

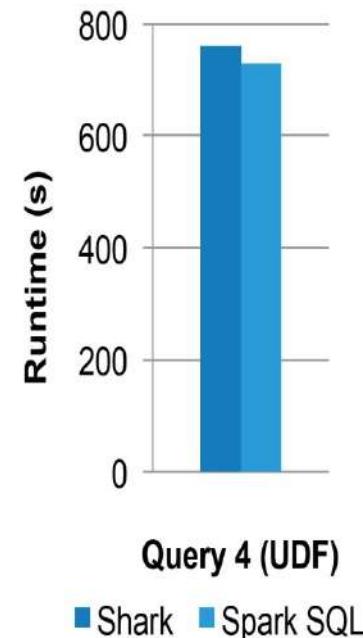
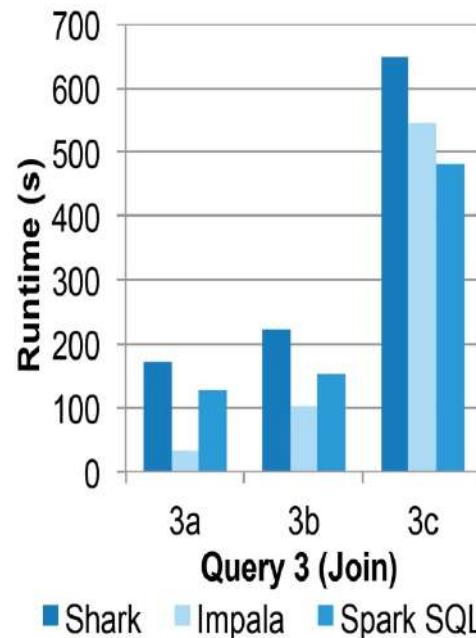
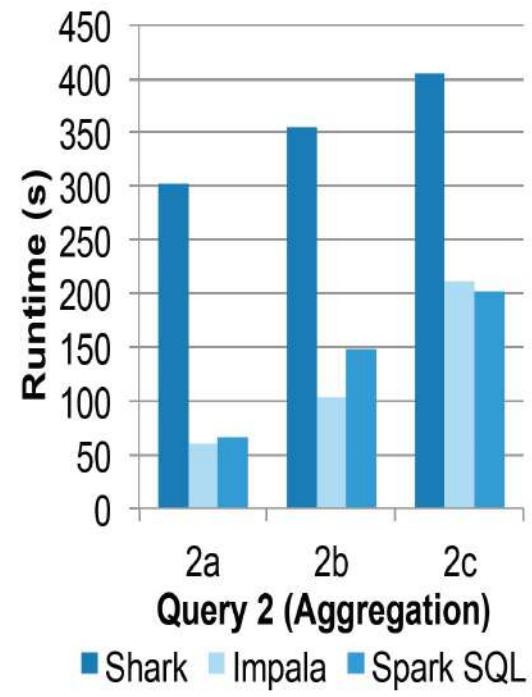
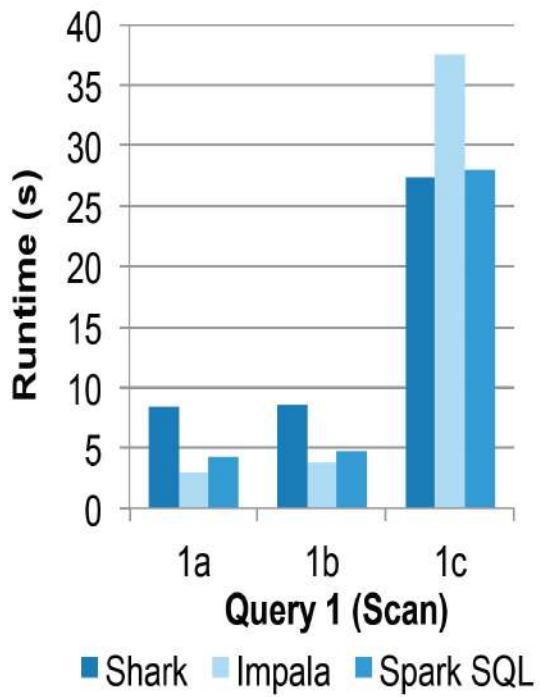
```
text STRING NOT NULL,  
tags ARRAY<STRING NOT NULL> NOT NULL,  
loc STRUCT<lat FLOAT NOT NULL, long FLOAT NOT NULL>
```

# Spark MLlib Pipelines

```
tokenizer = Tokenizer(inputCol="text", outputCol="words")
hashingTF = HashingTF(inputCol="words", outputCol="features")
lr = LogisticRegression(maxIter=10, regParam=0.01)
pipeline = Pipeline(stages=[tokenizer, hashingTF, lr])

df = sqlCtx.load("/path/to/data")
model = pipeline.fit(df)
```





110GB of data  
after columnar  
compression  
with Parquet

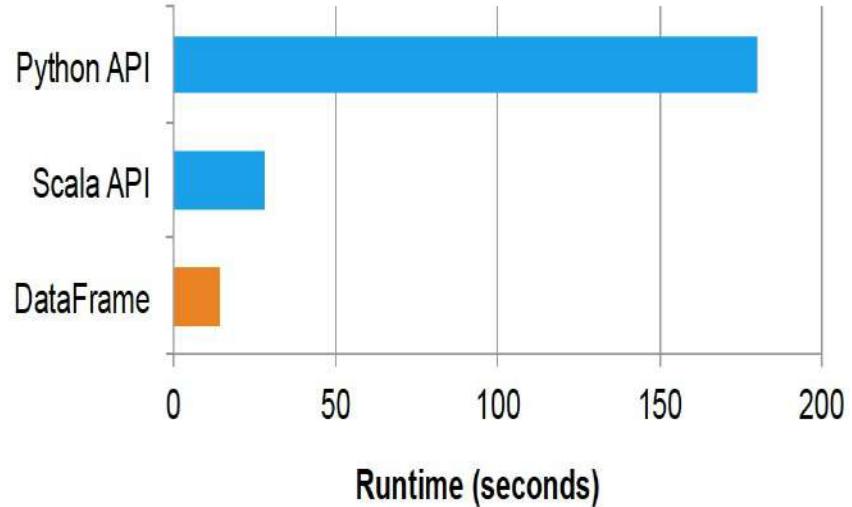


Figure 9: Performance of an aggregation written using the native Spark Python and Scala APIs versus the DataFrame API.

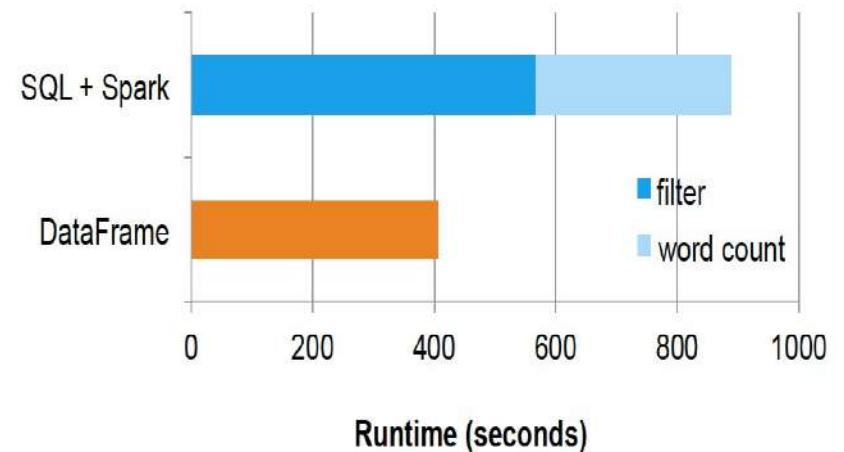
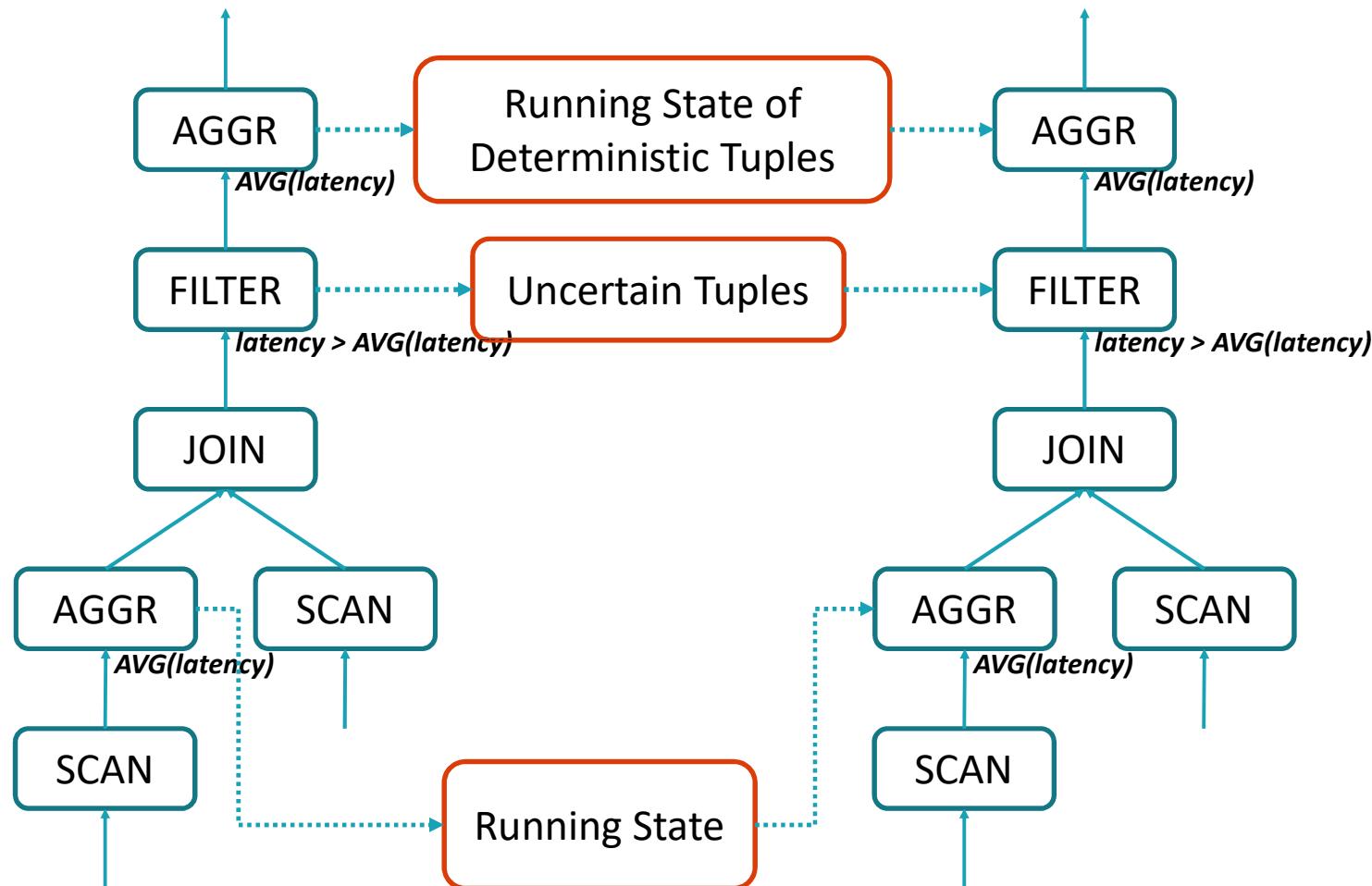
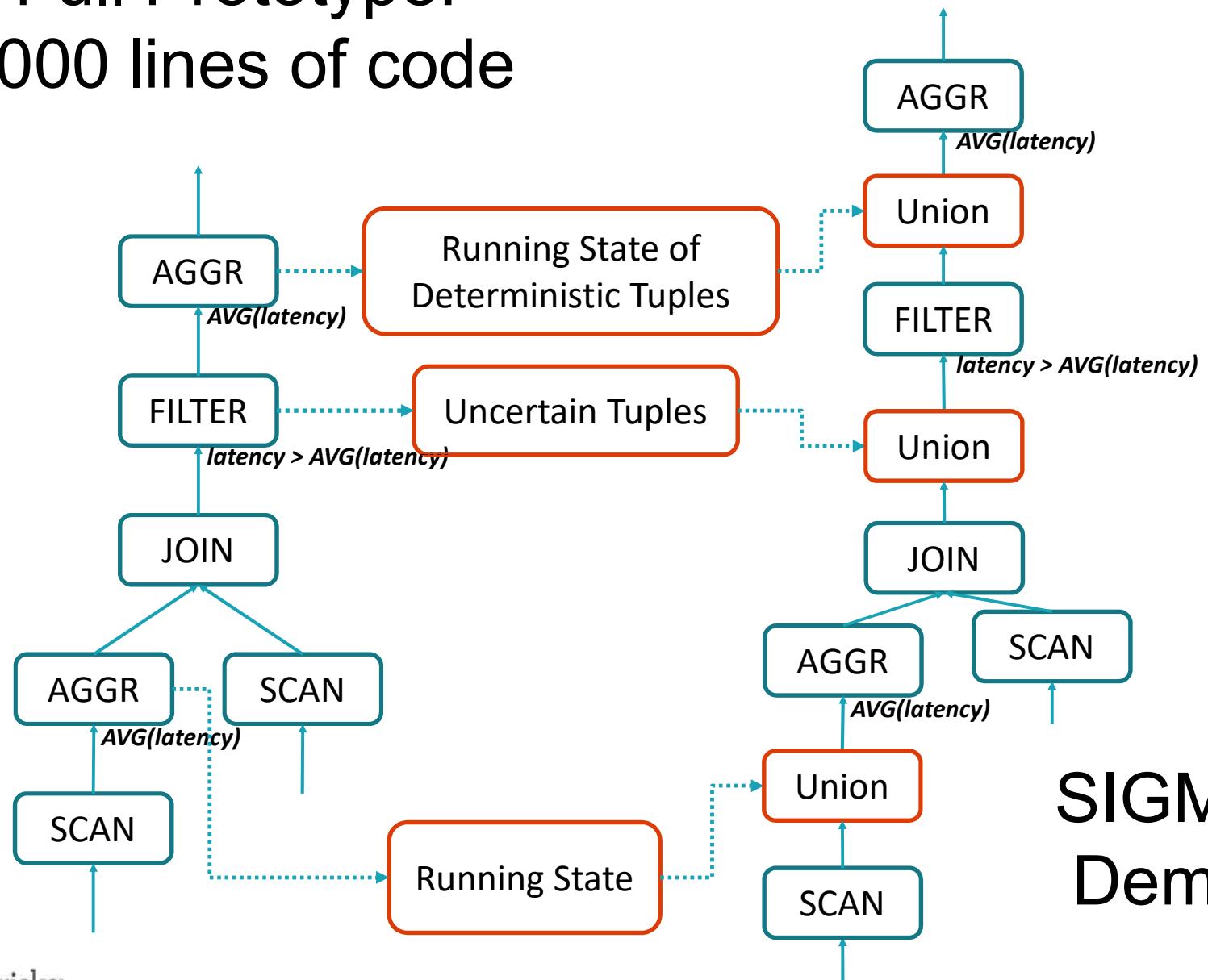


Figure 10: Performance of a two-stage pipeline written as a separate Spark SQL query and Spark job (above) and an integrated DataFrame job (below).

# Research Transformations: Generalized Online Aggregation



# Full Prototype: 3000 lines of code



SIGMOD  
Demo A

# Research Transformation: Genomics

Recognize range joins and use interval trees.

```
SELECT * FROM a JOIN b  
WHERE a.start < a.end  
    AND b.start < b.end  
    AND a.start < b.start  
    AND b.start < a.end
```



# Future Work: Project Tungsten

Overcome JVM limitations:

- **Memory Management and Binary Processing:** leveraging application semantics to manage memory explicitly and eliminate the overhead of JVM object model and garbage collection
- **Cache-aware computation:** algorithms and data structures to exploit memory hierarchy
- **Code generation:** using code generation to exploit modern compilers and CPUs



# Questions?







The not-so-secret truth...



is about more than SQL.

# **SparkSQL**: Declarative BigData Processing

Let Developers Create and Run Spark Programs Faster:

- Write less code
- Read less data
- Let the optimizer do the hard work

# DataFrame

*noun* – [dey-tuh-freym]

1. A distributed collection of rows organized into named columns.
2. An abstraction for selecting, filtering, aggregating and plotting structured data (*cf. R, Pandas*).
3. Archaic: Previously SchemaRDD (*cf. Spark < 1.3*).

# Write Less Code: Compute an Average



```
private IntWritable one =
    new IntWritable(1)
private IntWritable output =
    new IntWritable()
protected void map(
    LongWritable key,
    Text value,
    Context context) {
    String[] fields = value.split("\t")
    output.set(Integer.parseInt(fields[1]))
    context.write(one, output)
}

IntWritable one = new IntWritable(1)
DoubleWritable average = new DoubleWritable()

protected void reduce(
    IntWritable key,
    Iterable<IntWritable> values,
    Context context) {
    int sum = 0
    int count = 0
    for(IntWritable value : values) {
        sum += value.get()
        count++
    }
    average.set(sum / (double) count)
    context.write(key, average)
}
```



```
data = sc.textFile(...).split("\t")
data.map(lambda x: (x[0], [x[1], 1])) \
    .reduceByKey(lambda x, y: [x[0] + y[0], x[1] + y[1]]) \
    .map(lambda x: [x[0], x[1][0] / x[1][1]]) \
    .collect()
```

# Write Less Code: Compute an Average

## Using RDDs

```
data = sc.textFile(...).split("\t")
data.map(lambda x: (x[0], [int(x[1]), 1])) \
    .reduceByKey(lambda x, y: [x[0] + y[0], x[1] + y[1]]) \
    .map(lambda x: [x[0], x[1][0] / x[1][1]]) \
    .collect()
```

## Using SQL

```
SELECT name, avg(age)
FROM people
GROUP BY name
```

## Using Pig

```
P = load '/people' as (name, name);
G = group P by name;
R = foreach G generate ... AVG(G.age);
```

## Using DataFrames

```
sqlCtx.table("people") \
    .groupBy("name") \
    .agg("name", avg("age")) \
    .collect()
```

# Seamlessly Integrated: RDDs

Internally, DataFrame execution is done with Spark RDDs making interoperation with outside sources and custom algorithms easy.

## External Input

```
def buildScan(  
    requiredColumns: Array[String],  
    filters: Array[Filter]):  
  RDD[Row]
```

## Custom Processing

```
queryResult.rdd.mapPartitions { iter =>  
  ... Your code here ...  
}
```

# Extensible Input & Output

Spark's Data Source API allows optimizations like column pruning and filter pushdown into custom data sources.

## Built-In



## External



# Seamlessly Integrated

Embedding in a full programming language makes UDFs trivial and allows composition using functions.

```
zipToCity = udf(lambda city: <custom logic here>)
```

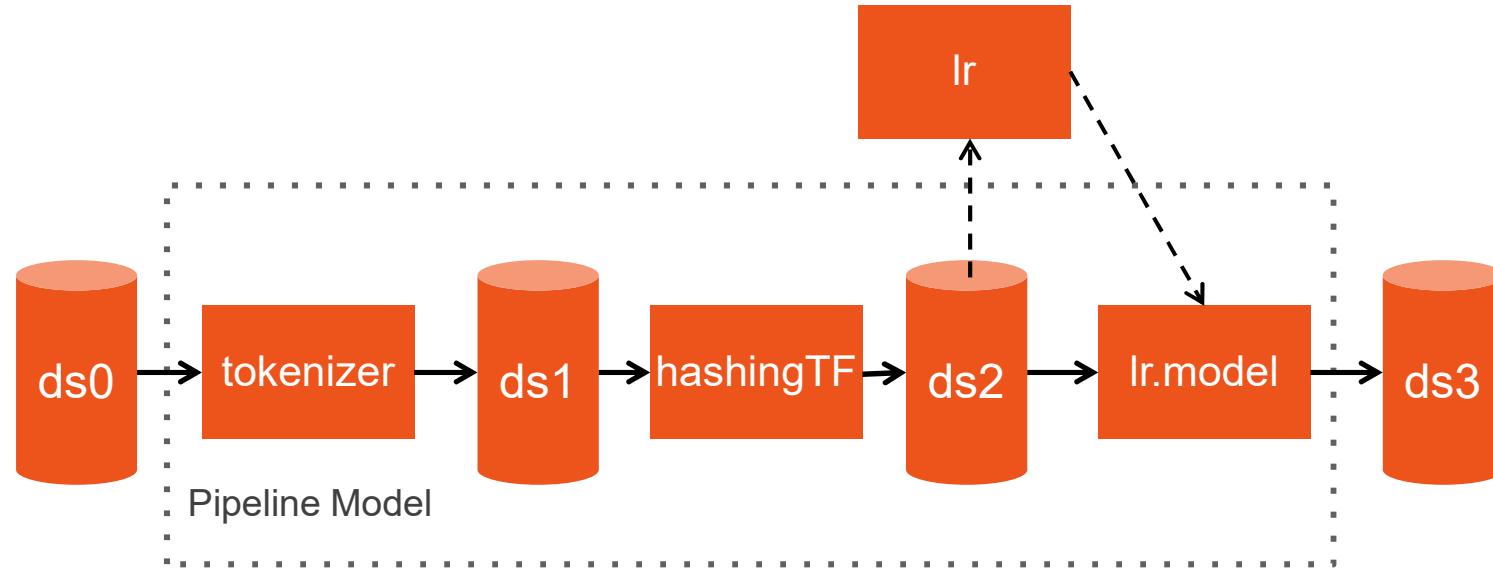
```
def add_demographics(events):
    u = sqlCtx.table("users")
    events \
        .join(u, events.user_id == u.user_id) \
        .withColumn("city", zipToCity(df.zip))
```

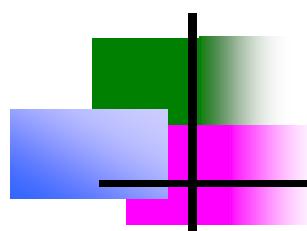
Takes and  
returns a  
DataFrame  
e

# Spark MLlib Pipelines

```
tokenizer = Tokenizer(inputCol="text", outputCol="words")
hashingTF = HashingTF(inputCol="words", outputCol="features")
lr = LogisticRegression(maxIter=10, regParam=0.01)
pipeline = Pipeline(stages=[tokenizer, hashingTF, lr])

df = sqlCtx.load("/path/to/data")
model = pipeline.fit(df)
```





---

# Chapter5

# Using Big Data for Analytics

NoSQL Database:

New Era of Databases for Big Data Analytics –  
Classification, Characteristics and Comparison

Basanta Joshi, PhD

Asst. Prof., Depart of Electronics and Computer Engineering

Program Coordinator, MSc in Information and Communication Engineering

Member, Laboratory for ICT Research and Development (LICT)

Member, Research Management Cell (RMC)

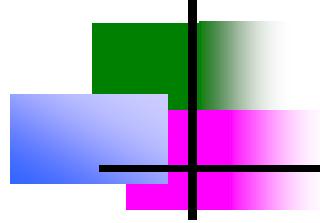
Institute of Engineering

basanta@ioe.edu.np

<http://www.basantajoshi.com.np>

<https://scholar.google.com/citations?user=iocLiGcAAAAJ>

[https://www.researchgate.net/profile/Basanta\\_Joshi2](https://www.researchgate.net/profile/Basanta_Joshi2)



# Keywords

---

- NoSQL Database
- Big Data
- NewSQL Database
  - ✓ Provide the same scalable performance maintaining the ACID
- Big Data Analytics
  - ✓ Examining big data to uncover information
  - ✓ Make informed business decision



ill

# Some History

## ❖ RDBMS

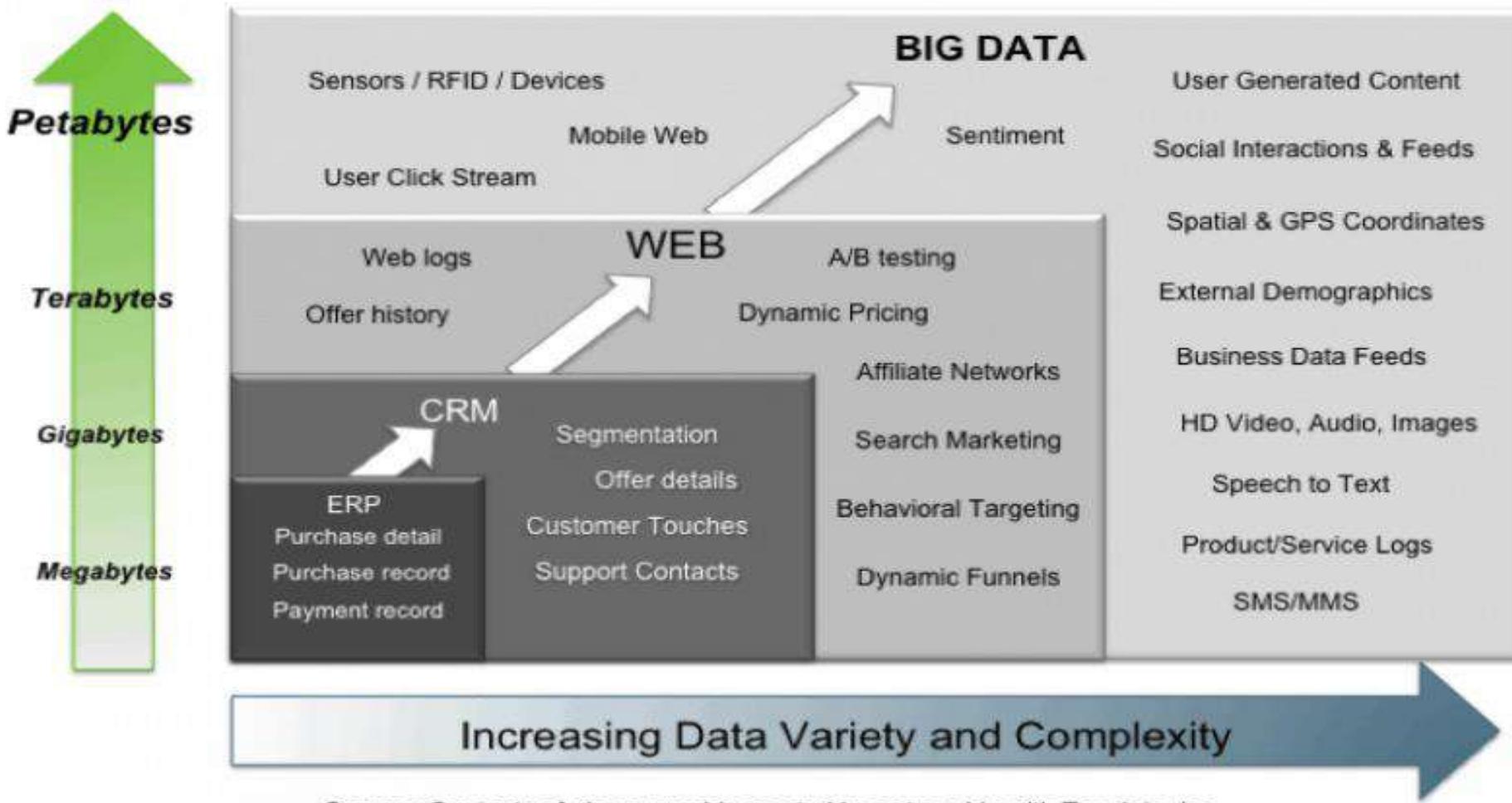
- ✓ Relational Database Management System
- ✓ Relational model of data
- ✓ Centralized database

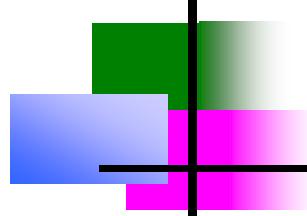


## ❖ Trends

- ✓ Exponential growth of volume of data
- ✓ Increasing interdependency and complexity

# Big Data = Transactions + Interactions + Observations

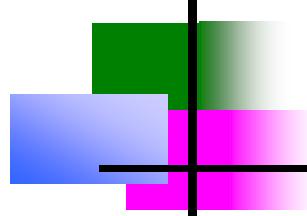




# Issues with RDBMS

---

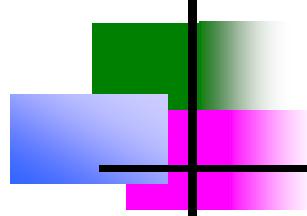
- ❖ Scalability
  - ✓ To scale relational database it has to be distributed on to multiple servers.
  - ✓ Handling tables across different servers is difficult .
- ❖ Not all data is relational
- ❖ Limits to scaling up (vertical scaling)
- ✓ Costly



# NoSQL System

---

- Distributed architecture
- Large scale data storage using non-relational databases
- Massively-parallel data processing across a large number of commodity servers
- Use non-SQL language (can use api's that translate SQL to non-SQL)
- Support exploratory and predictive analytics, ETL-style data transformation and non mission-critical OLTP



# NoSQL

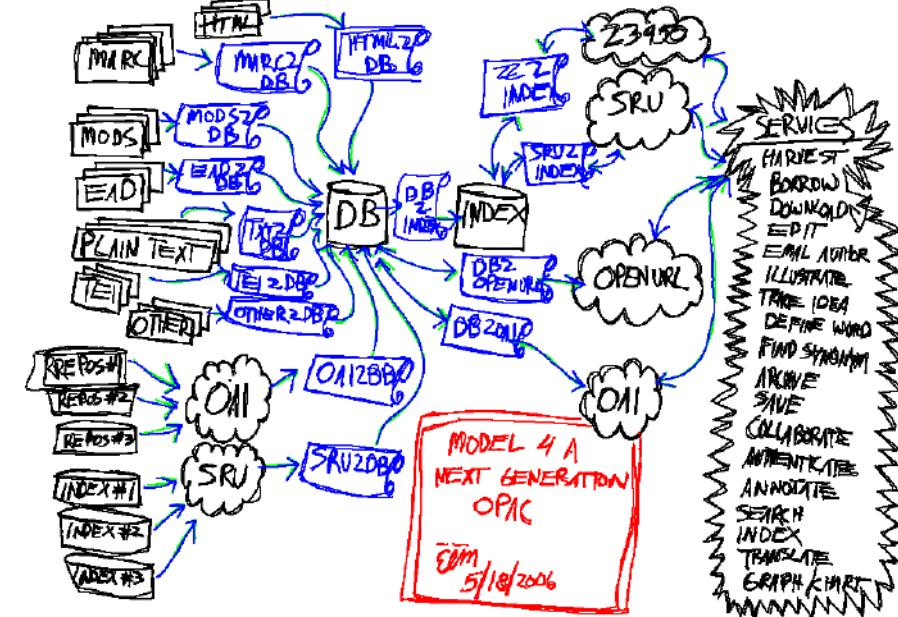
---

- ❖ Stand for “Not Only SQL”
- ❖ Non-relational data management systems
- ❖ Horizontally scalable

*“NoSQL systems are distributed, non-relational databases designed for large-scale data storage and for massively-parallel data processing across a large number of commodity servers.”*

# Background

- Relational Model dominated since 80s
  - MySQL, Oracle, MS-SQL Server
- Problems include deficits and modeling of data/constraints of horizontal scalability over several servers and big data
- MAJOR Trends
  - Exponential growth of volume of data generated by users, system,, sensors, Big distributed systems like Amazon and Google
  - Increasing independency and complexity of data accelerated by Internet, Web2.0, social networks



# The Era of Databases

## newSQL Systems

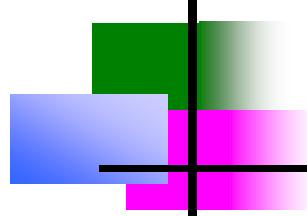
- Provide (Atomicity, Consistency, Isolation, Durability) ACID – compliant real-time OLTP
  - Eg. Manage long duration or inter-organization transactions
- Support conventional SQL-based OLAP (online analytical processing) in Big Data environments

## No-SQL Systems

- Break into conventional RDBMS
  - Column –oriented data storage
  - Distributed architectures
  - In-memory processing
  - Symmetric Multi-processing(SMP)
  - Massively parallel processing(MPP)

# Characteristics of NoSQL Databases

- ❖ Relaxed ACID (CAP Theorem) fewer guarantees
  - ❖ CAP Theorem 
  - ✓ Consistency
    - ❑ All copies have same value
  - ✓ Availability
    - ❑ Reads and writes always succeed
  - ✓ Partition-tolerance
    - ❑ System properties (consistency and/or availability hold even when network failures prevent some machines from communicating with others)



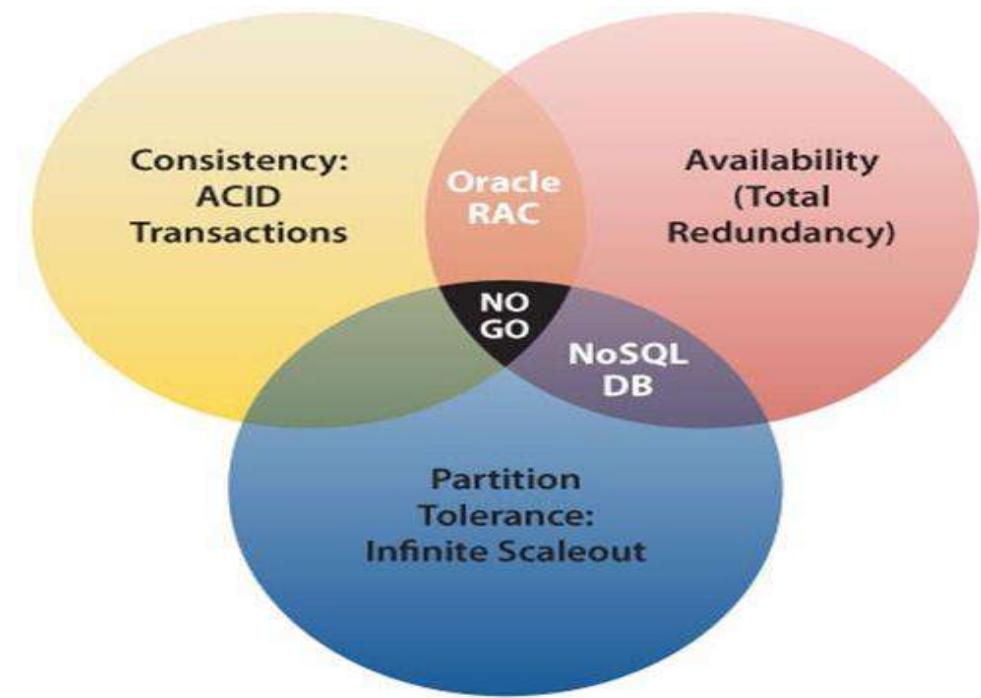
# Why Non-Relational Databases?

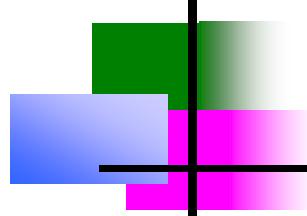
---

- They focus on analytical processing of large scale datasets
- Offer Increased Scalability over commodity hardware
- Business Intelligence, Big Data Analytics and social networking have computational and storage requirements over **peta-Bytes**
- Scalable with Data-Warehousing, Grid, Web2.0 and CloudApplications
- Exhibit the ability to store and index arbitrarily big data sets --- while enabling a large no. of concurrent user requests

# CAP Theorem

- Drop A or C of ACID
  - relaxing C makes replication easy, facilitates fault tolerance, speed up transactions
  - relaxing A reduces (or eliminates) need for distributed concurrency control.





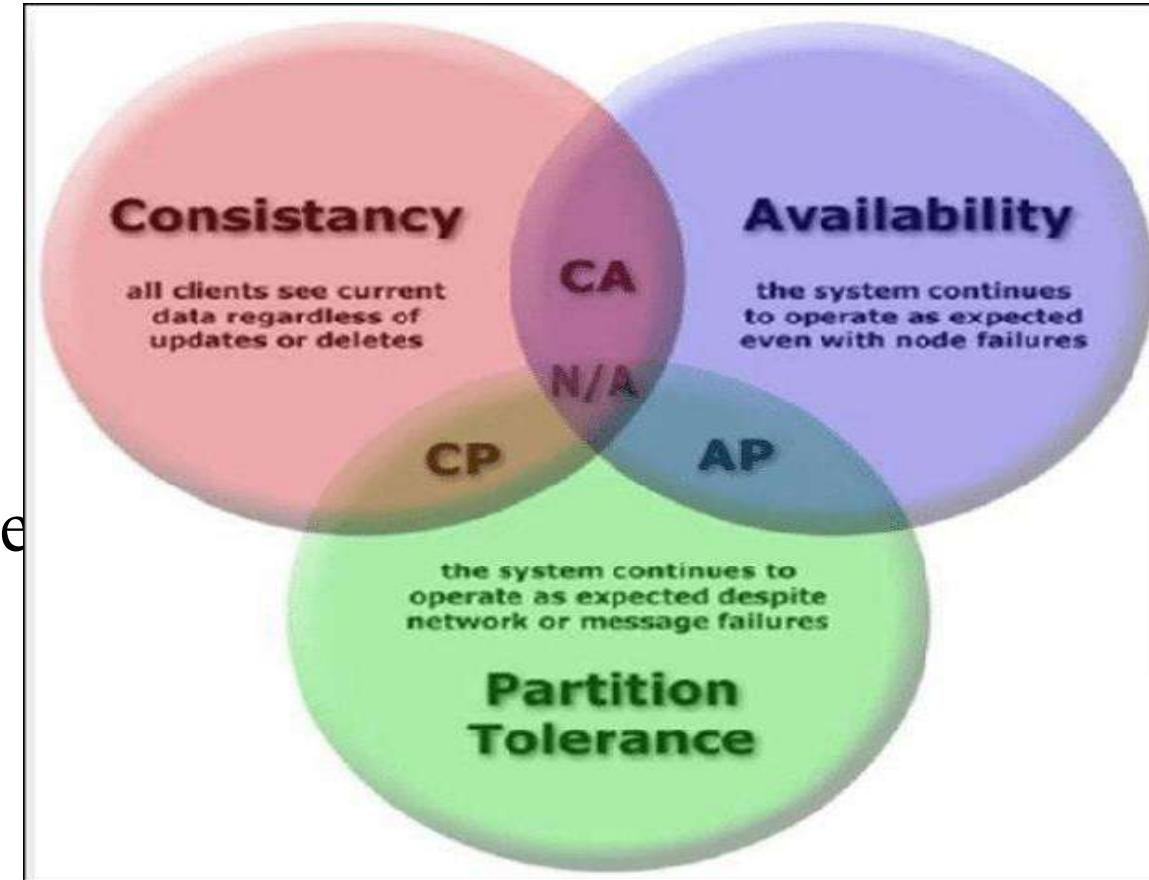
# CAP Theorem

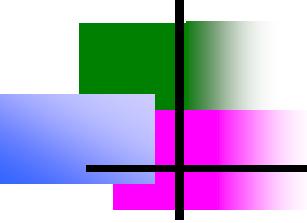
---

- Supposes three properties of a system
  - Consistency (all copies have same value)
  - Availability (system can run even if parts have failed)
  - Partitions (network can break into two or more parts, each with active systems that can not influence other parts)
- Eric Brewer’s CAP “Theorem” (1999) :  
*For any system sharing data it is impossible to guarantee simultaneously all of these three properties*

# CAP Theorem

- Pick 2!
  - ❖ To scale out, partition is needed
    - ✓ In almost all cases, availability and partition over consistency
- Resulted in BASE**
- “Basically Available, Soft-state, Eventually Consistent”

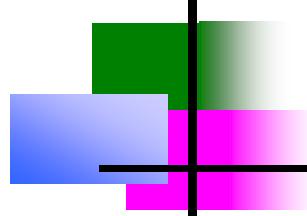




# CAP Theorem

---

- Traditional RDBMS implement **C** and **A**
- Very large systems will partition at some point
  - Node failure  
(Google reports MTTF=9 hours in its 1800' node cluster)
  - Network failure
  - Network delay



# CAP Theorem

---

- To deal with Big Data it is necessary to decide between **C** and **A**
- Most Web applications choose **A**  
(except in specific applications such as order processing)
- **P** depends on timeout settings
- **C** and **A** are more than just binary

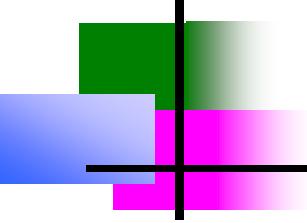


# BASE

# BASE

- Basically Available
- Soft state
- Eventual consistency
- BASE as opposed to ACID
  - Soft state: copies of a data item may be inconsistent
  - Eventually Consistent – copies becomes consistent at some later time if there are no more updates to that data item
  - Basically Available – possibilities of failures but not a failure of the whole system

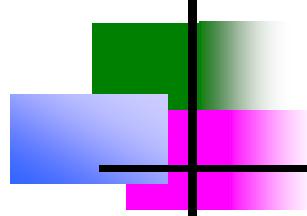




# BASE: Relaxing ACID properties

---

- BigData
  - ACID is hard to achieve, moreover, it is not always required, *e.g. for blogs, status updates, product listings, etc.*
- Availability
  - Traditionally, thought of as the server/process available 99.999% of time
  - For a large-scale node system, there is a high probability that a node is either down or that there is a network partitioning
- Partition tolerance
  - ensures that write and read operations are redirected to available replicas when segments of the network become disconnected

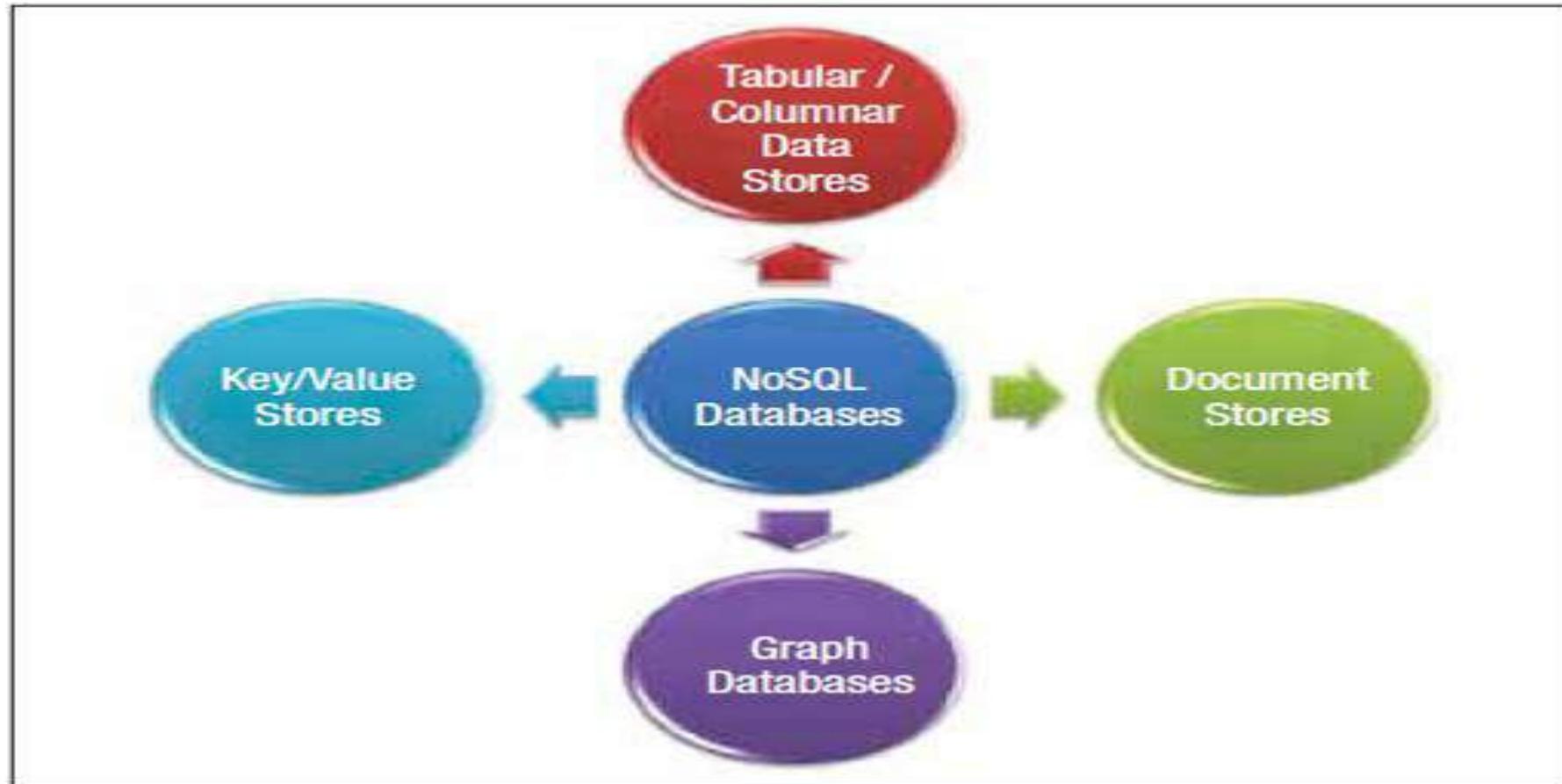


# BASE: Eventual Consistency

---

- When no updates occur for a long period of time, eventually all updates will propagate through the system and all the nodes will be consistent
- For a given accepted update and a given node, eventually either the update reaches the node or the node is removed from service

# Classification of NoSQL Databases





Redis



Neo4j  
the graph database

mongoDB

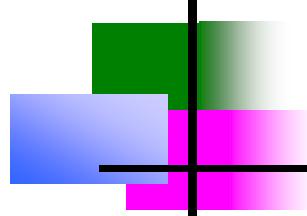


riak



A. Gorbenko

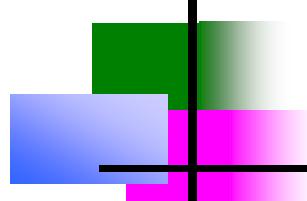
23



# NoSQL databases

---

- The name stands for **Not Only SQL**
  - 2009, Eric Evans (Rackspace)
- Common features:
  - non-relational
  - do not require a fixed table schema
  - horizontal scalable
  - very fast data access (reads and writes)
  - mostly open source



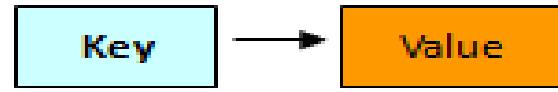
# NoSQL databases

---

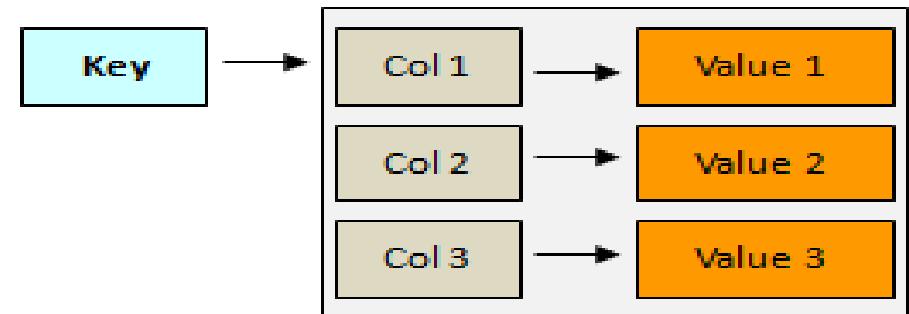
- More characteristics
  - the data structure (e.g. key-value, graph, or document) differs from the RDBMS
  - relax one or more of the ACID properties
  - choose A-P or C-P (see CAP theorem)
  - replication support
  - easy API (if SQL, then only its very restricted variant)
- Do not fully support relational features
  - no join operations (except within partitions),
  - no referential integrity constraints across partitions.

# Categories of NoSQL databases

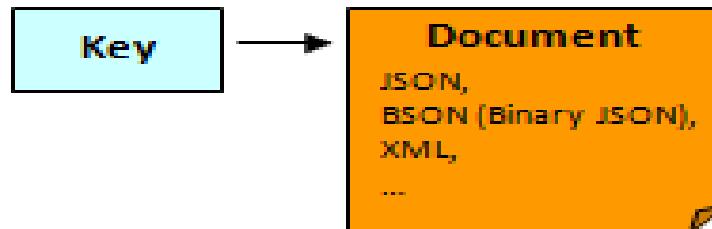
- **Key/Value**



- **Column-oriented**



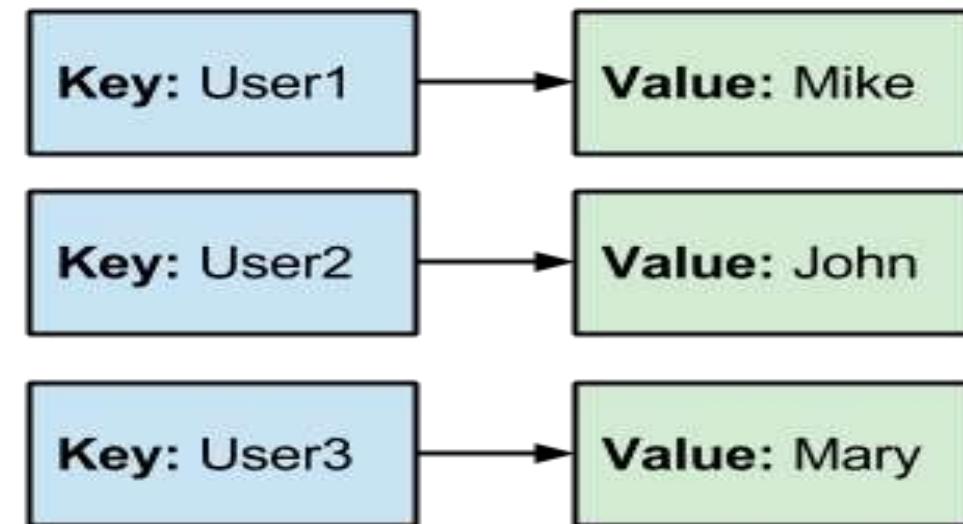
- **Document-oriented**



- Graph database (neo4j, InfoGrid)
- XML databases (myXMLDB, Tamino, Sedna)

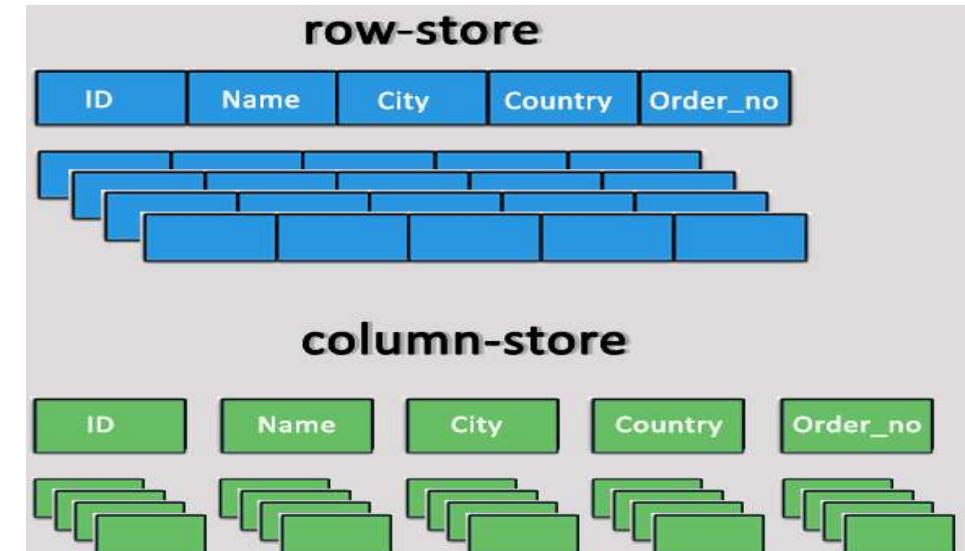
# NoSQL databases: Key/Value

- Works as a big cache stored on a remote server.
- Each value is associated with a key.
- The value is “opaque” (no specific structure, not indexed), though can be string, JSON, BLOB, etc.



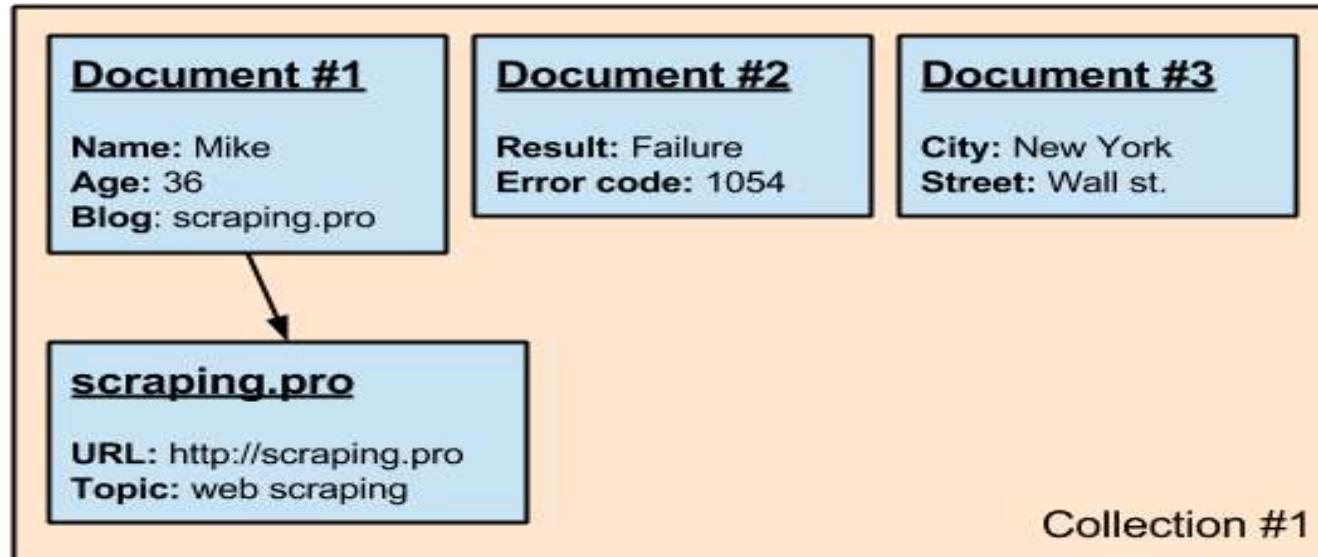
# NoSQL databases: Column-oriented

- the data is also associated with a “key” but it is organized by “columns”
- the columns can be grouped by “family”.
- There is no fixed column definition (no schema)
- the values for each column are physically stored sequentially into disk blocks



# NoSQL databases: Document

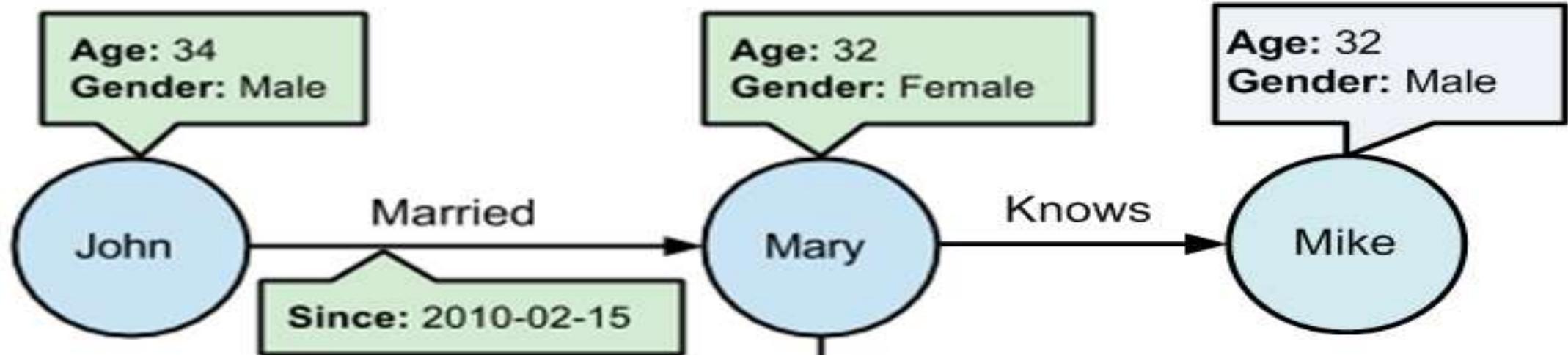
- each key is associated with a “document”
- Document is usually formatted in JSON or XML
- Each document has an arbitrary set of properties



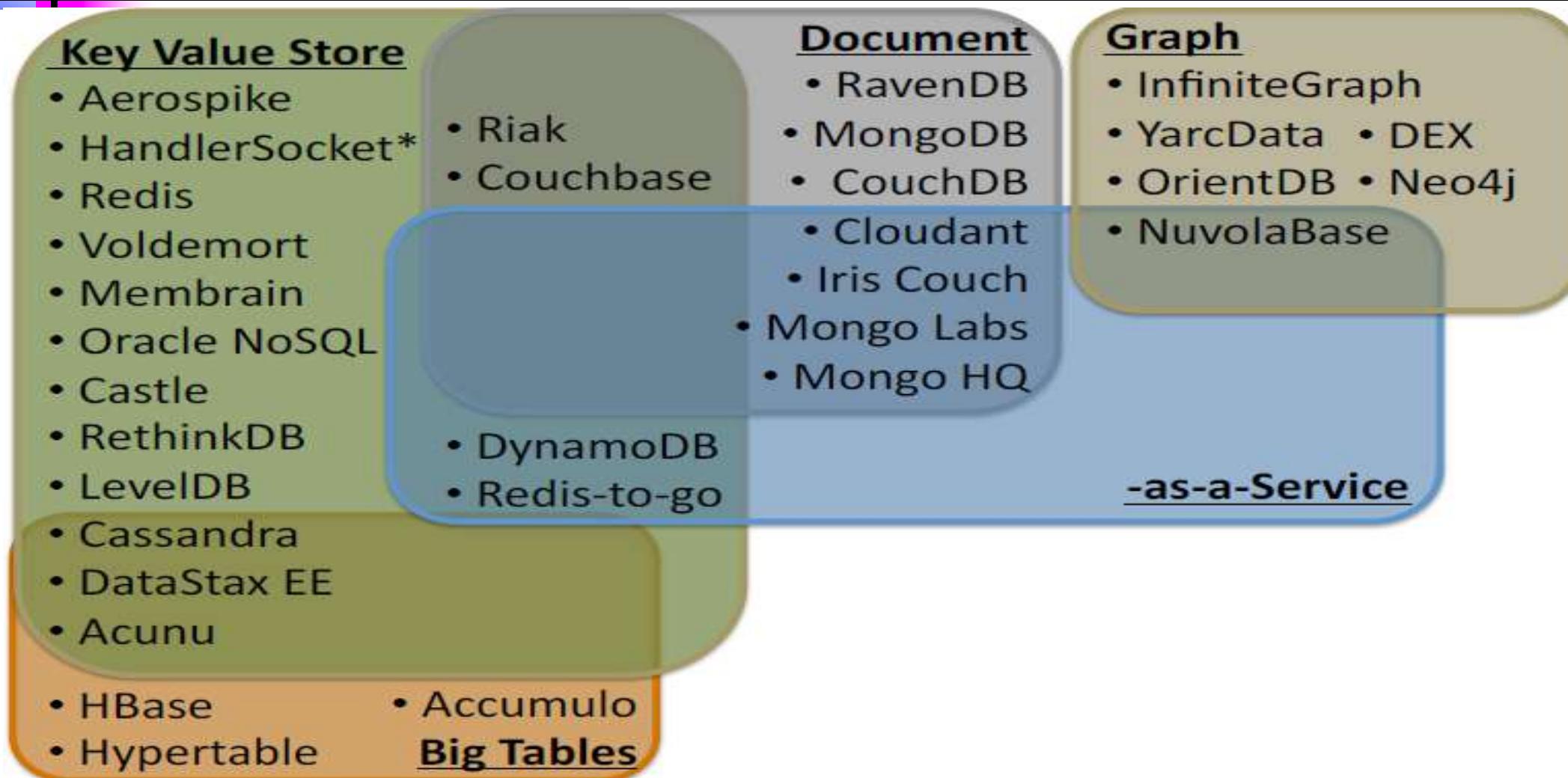
```
{Name: "Jack",
Address: "Graham str. 25, NE1 7RU,
Newcastle, UK"
Grandchildren: [Claire: "7", Barbara: "6",
Magda: "3", Kirsten: "1", Otis: "3", Richard:
"1"]
}
```

# NoSQL databases: Graph-oriented

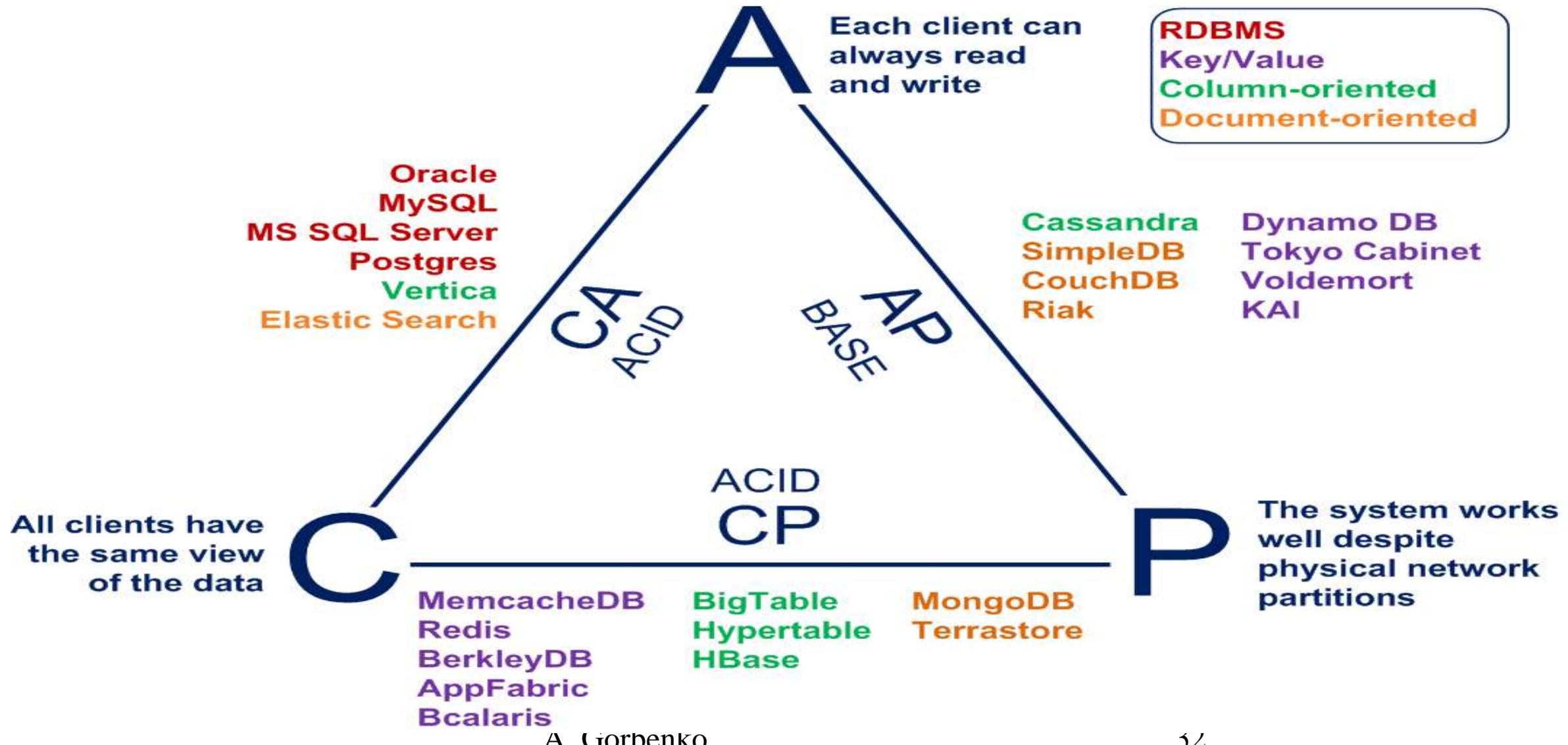
- Keep data in the forms of nodes, properties and edges
- Nodes stand for objects whose data we want to store
- Properties represent the features of those objects
- Edges show the relationships between those objects

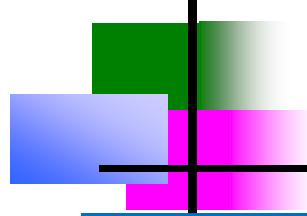


# NoSQL family



# NoSQL databases: CAP implication





# SQL vs NoSQL. Data definition

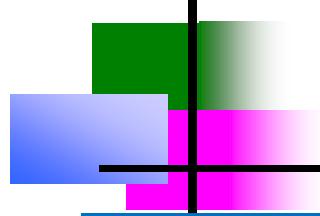
---

## SQL (RDBMS)

1. DB contains tables, table contains columns and rows, rows are made of column values. Rows within a table all have the same schema
2. Data model is well defined in advance. A schema is strongly typed, has constraints and relationships enforcing data integrity
3. The data model is normalized to remove data duplication. Normalization establishes table relationships that associate data between tables

## NoSQL

1. DB contains domains, domain contains rows (items), but rows contain variable set of attributes and can have different schema
2. Rows (items) are identified by keys. New attributes can be added into the row.
3. Attributes usually are textual or of a simple type
4. No relationships are explicitly defined between domains



# SQL vs NoSQL. Data access

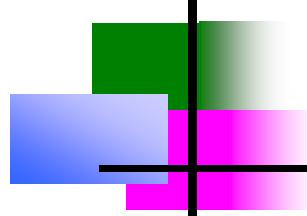
---

## SQL (RDBMS)

1. Data is created, updated, deleted and retrieved using SQL
2. SQL queries can access data from multiple tables (table joins)
3. SQL queries include functions for aggregation and complex filtering
4. DB contains means of supporting data integrity and embedding logic close to data (triggers, stored procedures)
5. Object-Relational Mapping is needed

## NoSQL

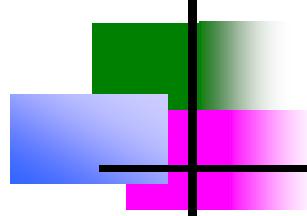
1. Data is created, updated, deleted and retrieved using API calls usually only by key
2. Tables joins are hardly supported
3. Only basic filter predicates ( $=, !=, >, <$ ) can often be applied
4. All application and data integrity logic is contained in the application code



# Typical NoSQL API

---

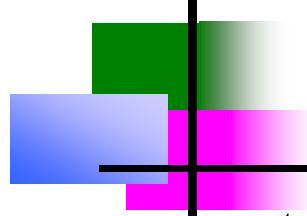
- Basic API access:
  - get (key) -- Extract the value given a key
  - put (key, value) -- Create or update the value given its key
  - delete (key) -- Remove the key and its associated value
  - execute (key, operation, parameters) -- Invoke an operation to the value (given its key) which is a special data structure (e.g. List, Set, Map .... etc).



# NewSQL

---

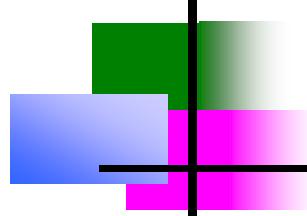
- NewSQL is a class of modern RDBMS (from April 2011) that seek to:
  - provide the same scalable performance of NoSQL systems
  - still maintain the ACID guarantees of a traditional RDBMS.



# NewSQL

---

- next generation of highly scalable and elastic RDBMS: NewSQL databases
  - still provide ACID guarantees,
  - still use SQL
  - designed to scale out horizontally on shared nothing machines,
  - employ a lock-free concurrency control scheme to avoid user shut down,
  - provide higher performance than available from the traditional systems.
- Examples: MySQL Cluster (most mature solution), VoltDB, Clustrix, ScalArc,  
...



# NoSQL vs NewSQL

---

## NoSQL

- New breed of non-relational database products
- Rejection of fixed table schema and join operations
- Designed to meet scalability requirements of distributed architectures
- And/or schema-less data management requirements

## NewSQL

- New breed of relational database products
- Retain SQL and ACID
- Designed to meet scalability requirements of distributed architectures
- Or improve performance so horizontal scalability is no longer a necessity

# NewSQL family

## -as-a-Service

- StormDB
- Xeround
- Tokutek

## Storage engines

- Datomic

- Akiban

- GenieDB

- ScaleDB

- MySQL Cluster

- Zimory Scale

- MemSQL

- Drizzle

- VoltDB

- JustOneDB

- ParElastic

- Continuent

- Galera

## New databases

- NuoDB

- SQLFire

- Translattice

- Clustrix

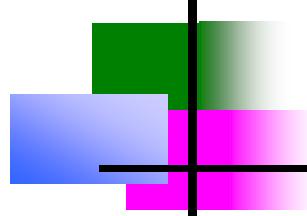
- SchoonerSQL

- ScaleBase

- ScaleArc

- CodeFutures

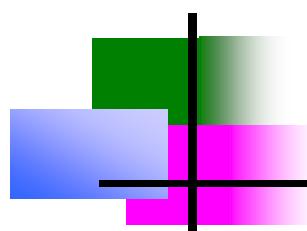
## Clustering/sharding



# NoSQL Database Uses

---

- Large Scale data processing (Parallel Processing over Distributed Processing)
- Embedded IR (basic machine to machine information lookup and retrieval)
- Exploratory analytics on semi-structured data (expert Level)
- Large volume data storage (un-, semi-, small-packet structured)



# NoSQL Databases -- Classification

---

- Key-Value Stores
- Column-oriented databases
- Wide-column stores
- Graph Databases

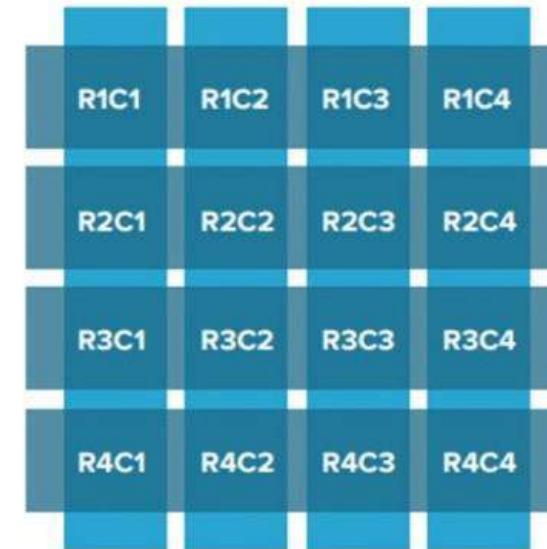
# Key Value Stores

- Stores alpha-numeric identifiers (keys)
- Associated values in hash tables
  - Simple text strings
  - Complex lists
  - Sets
- Data Search – Keys, not values, limited to exact matches
  - Primary USE:
    - Manage user profiles/sessions/retrieve product names
  - Eg. Dynamo of Amazon, Voldemort(LinkedIn), Redis, BerkeleyDB, Riak

Car	
Key	Attributes
1	Make: Nissan Model: Pathfinder Color: Green Year: 2003
2	Make: Nissan Model: Pathfinder Color: Blue Color: Green Year: 2005 Transmission: Auto

# Document Databases

- Value column contains semi-structured data
  - Esp. attribute name/value pairs
- Keys and values are fully searchable in document databases
- Primary USE:
  - Store and manage Big Data-size collections of literal documents – email, xml, email, de-normalized database entity such as product/customer
  - Storing parse data
  - Eg. CouchDB (JSON), MongoDB(BSON)



**Relational data model**

Highly-structured table organization with rigidly-defined data formats and record structure.

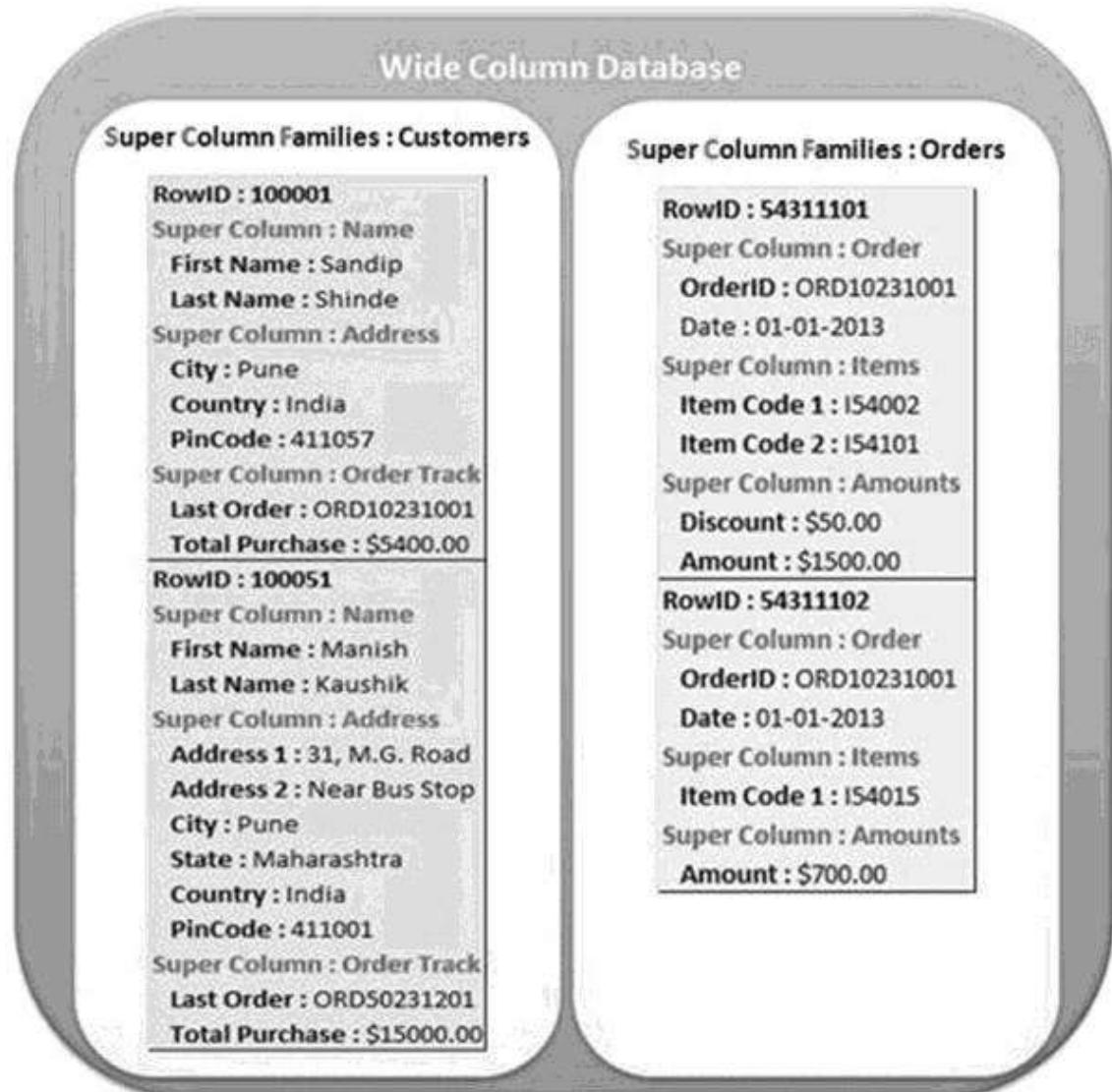


**Document data model**

Collection of complex documents with arbitrary, nested data formats and varying "record" format.

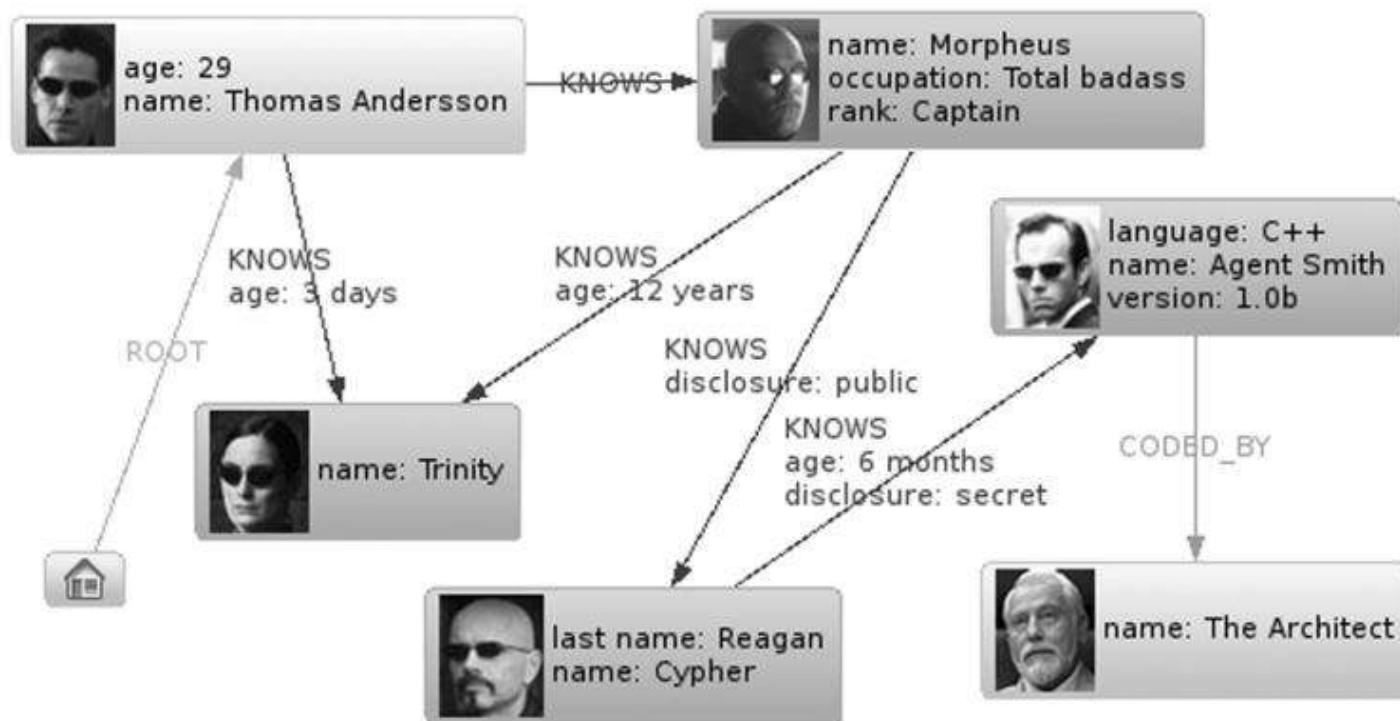
# Wide-Column Stores

- Similar to Doc DB in that one key can accommodate multiple attributes
- Patterned after --- Google's Big Table Data Storage (Google Search Engine)
- GFS filesystem, MapReduce parallel processing framework, Hadoop File System, Hbase
- Primary Uses:
  - Distributed Data Storage
  - Large-scale, batch-oriented data processing
  - Exploratory and Predictive Analytics
- It uses MapReduce – Batch Processing Method
  - Recently upgraded process is **Caffeine** -- search



# Graph Databases

- Replace Relational tables with
  - Structured Relational graphs of interconnected key-value pairings
- Resemble Object Oriented DBs.
  - Nodes – Conceptual Objects
  - Edges (Node Relationships)
  - Properties -- Object attributes (K-V pairs)
- Prime Uses:
  - Human-Friendly DB
  - Interesting Relationship Representation
    - Social Networks
    - Recommendation System
    - Forensic Investigations (Pattern Recognition)
  - Traversing data || Not querying!!
- Eg. InfoGrid, Neo4j, SonesGraphDB, AllegroGraph, Infinite Graph



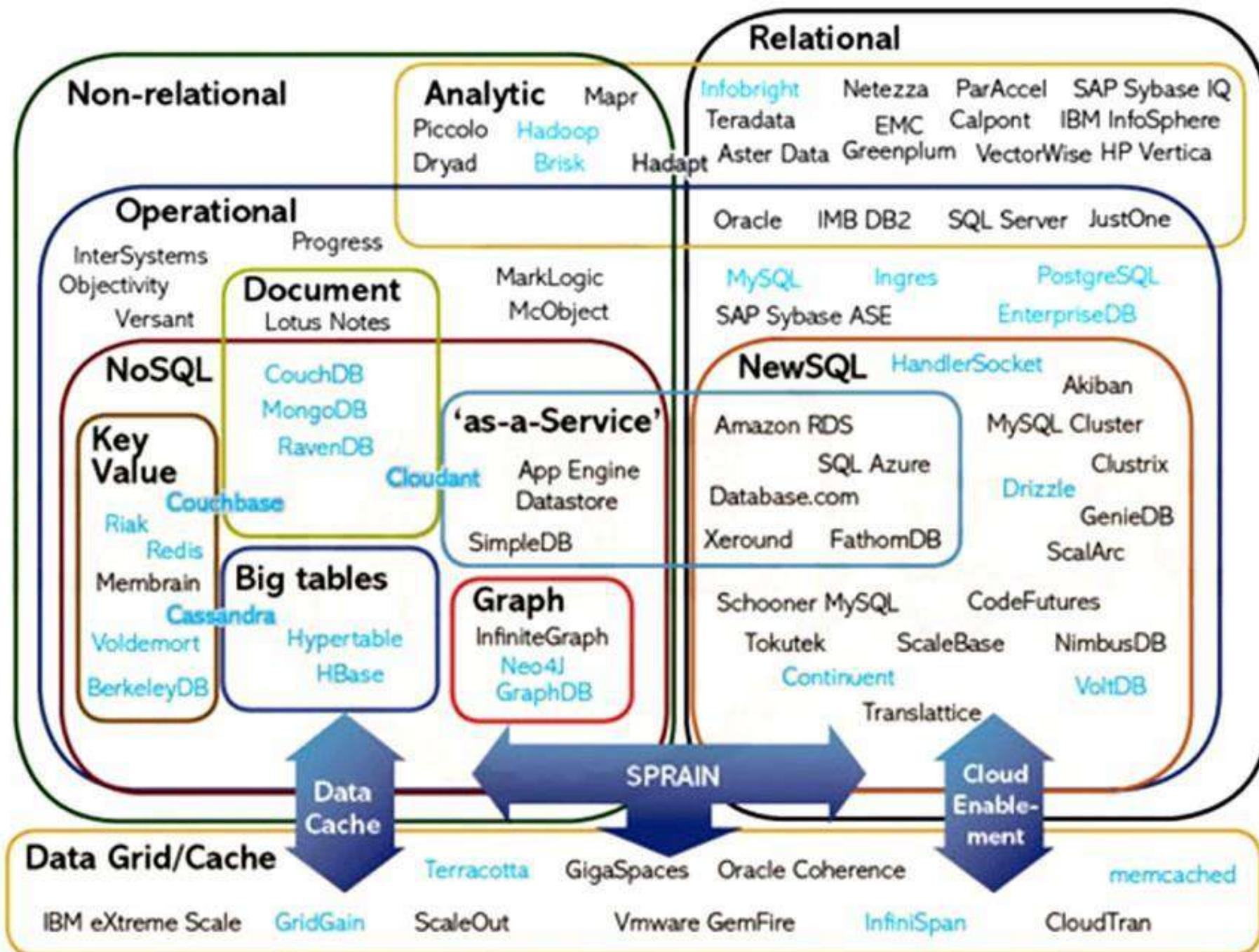
# Comparison of NoSQL Database

## Comparison of NoSQL models \*

Model	Performance	Scalability	Flexibility	Complexity	Functionality
Key-value	high	high	high	none	variable (none)
Document	high	variable (high)	high	low	variable (low)
Column	high	high	moderate	low	minimal
Graph	variable	variable	high	high	graph theory
Relational	variable	variable	low	moderate	relational algebra

\* Summary of a presentation by Ben Scofield: <https://www.slideshare.net/bscofield/nosql-codemash-2010>

# Comparision – Design, Integrity, Indexing, Distribution, System



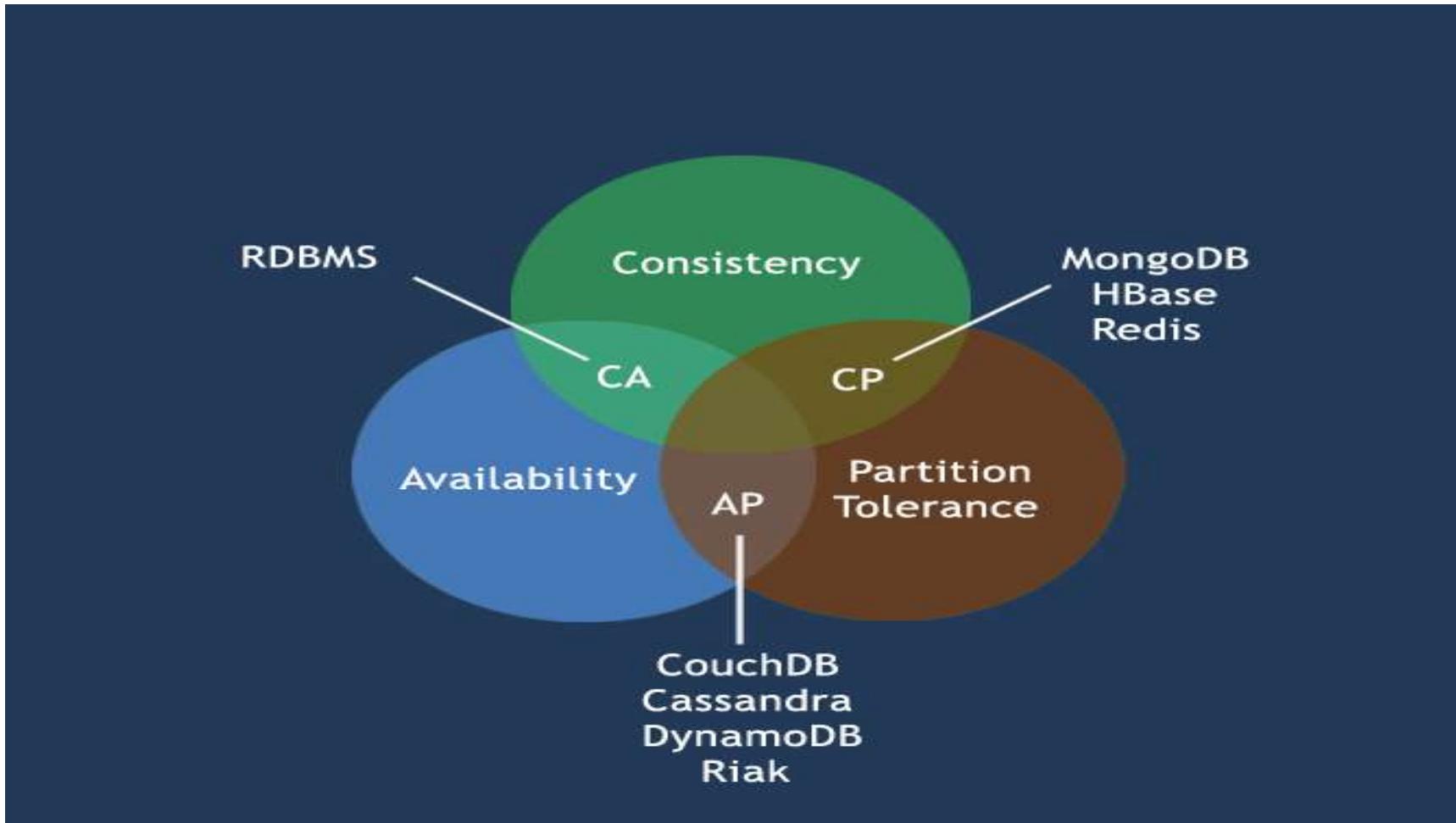
# Comparison of NoSQL Database

## Comparison of NoSQL models \*

Model	Performance	Scalability	Flexibility	Complexity	Functionality
Key-value	high	high	high	none	variable (none)
Document	high	variable (high)	high	low	variable (low)
Column	high	high	moderate	low	minimal
Graph	variable	variable	high	high	graph theory
Relational	variable	variable	low	moderate	relational algebra

\* Summary of a presentation by Ben Scofield: <https://www.slideshare.net/bscofield/nosql-codemash-2010>

# Comparison of NoSQL Database



# Comparison of NoSQL Database

Database		NoSQL							
Features	Design & Features	Document Stored		Wide-Column Stored		Key-Value Stored		Graph Database	
Integrity	BASE	 mongoDB	 CouchDB	 Amazon DynamoDB	 Google Bigtable	 redis	 riak	 Neo4j the graph database	 FlockDB
Indexing	MVCC	 ubuntu one	 eBay	ASID	 f digg	 Google	 GitHub	 jQuery	 Twitter
Secondary Index	Yes	Yes	Yes	Yes	Yes	-	Yes	-	Yes
Distribution	Master-Slave Replication	Master-Slave Replication	-	Master-Slave Replication	Master-Slave Replication	Master-Slave Replication	Master-Slave Replication	-	-
System	C++	Erlang, C++, C, Python	JAVA	JAVA	JAVA	C C++	Erlang	Erlang	JAVA

# Comparison of NoSQL Database

Attributes		NoSQL Databases								
Database model		Document-Stored		Wide-Column Stored				Key-Value Stored		Graph-oriented
Design & Features	Features	MongoDB	CouchDB	DynamoDB	HBase	Cassandra	Accumulo	Redis	Riak	Neo4j
	Data storage	Volatile memory File System	Volatile memory File System	SSD	HDFS		Hadoop	Volatile memory File System	Bitcask LevelDB Volatile memory	File System Volatile memory
	Query language	Volatile memory File System	JavaScript Memcached-protocol	API calls	API calls REST XML Thrift	API calls CQL Thrift		API calls	HTTP JavaScript REST Erlang	API calls REST SparQL Cypher Tinkerpop Gremlin
	Protocol	Custom, binary (BSON)	HTTP, REST	-	HTTP/REST Thrift	Thrift & custom binary CQL3	Thrift	Telnet-like	HTTP, REST	HTTP/REST Embedding in Java
	Conditional entry updates	Yes	Yes	Yes	Yes	No	Yes	No	No	
	MapReduce	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No
	Unicode	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	TTL for Entries	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	
	Compression	Yes	Yes	-	Yes	Yes	Yes	Yes	Yes	

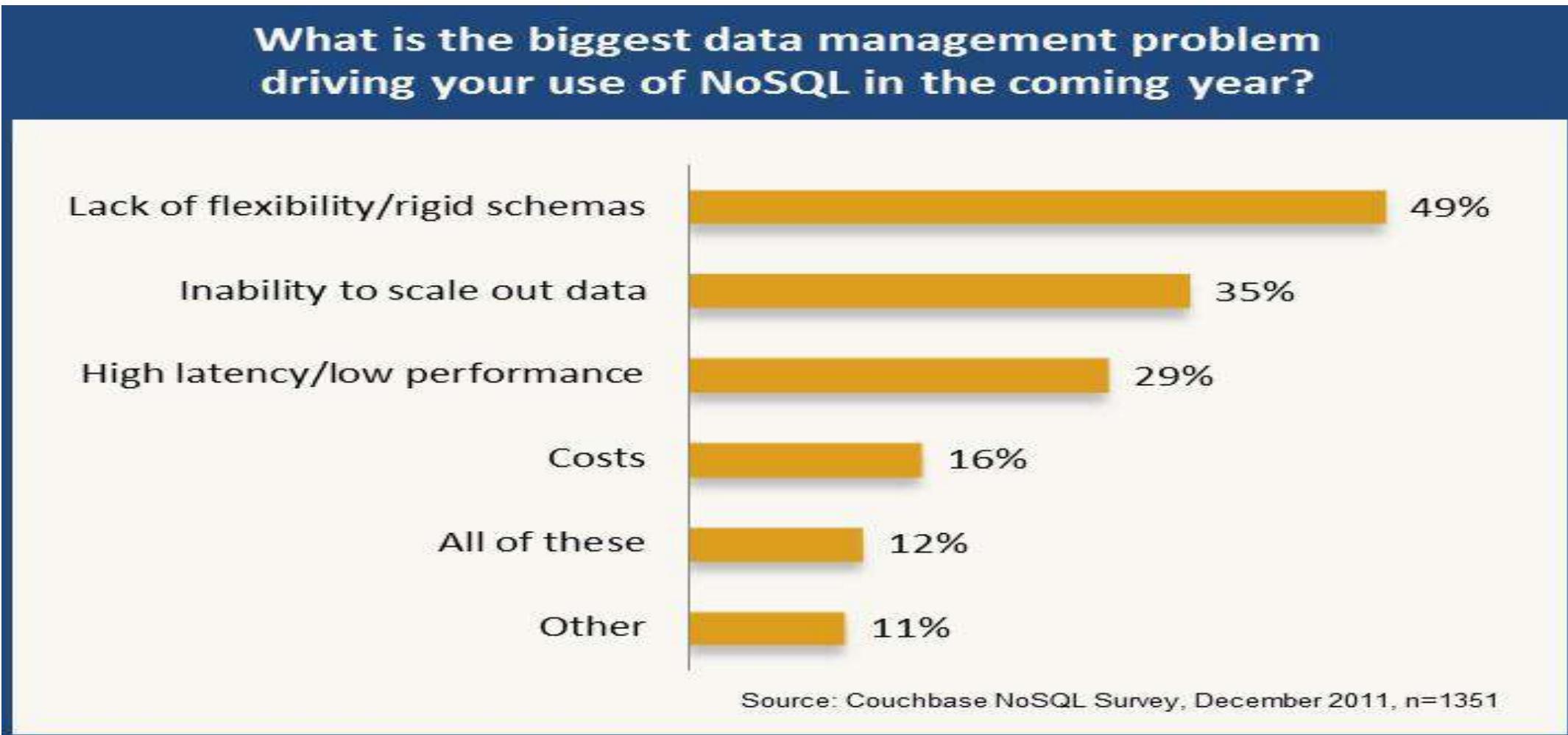
# Comparison of NoSQL Database

Attributes		NoSQL Databases								
Database model		Document-Stored		Wide-Column Stored				Key-Value Stored		Graph-oriented
	Features	MongoDB	CouchDB	DynamoDB	HBase	Cassandra	Accumulo	Redis	Riak	Neo4j
Integrity	Integrity model	BASE	MVCC	ASID	Log Replication	BASE	MVCC	-	BASE	ASID
	Atomicity	Conditional	Yes	Yes	Yes	Yes	Conditional	Yes	No	Yes
	Consistency	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
	Isolation	No	Yes	Yes	No	No	-	Yes	Yes	Yes
	Durability (data storage)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	-	Yes
	Transactions	No	No	No	Yes	No	Yes	Yes	No	Yes
	Referential integrity	No	No	No	No	No	No	Yes	No	Yes
	Revision control	No	Yes	Yes	Yes	No	Yes	No	Yes	No
Indexing	Secondary Indexes	Yes	Yes	No	Yes	Yes	Yes	-	Yes	-
	Composite keys	Yes	Yes	Yes	Yes	Yes	Yes	-	Yes	-
	Full text search	No	No	No	No	No	Yes	No	Yes	Yes
	Geospatial Indexes	Yes	No	No	No	No	Yes	-	-	Yes
	Graph support	No	No	No	No	No	Yes	No	Yes	Yes

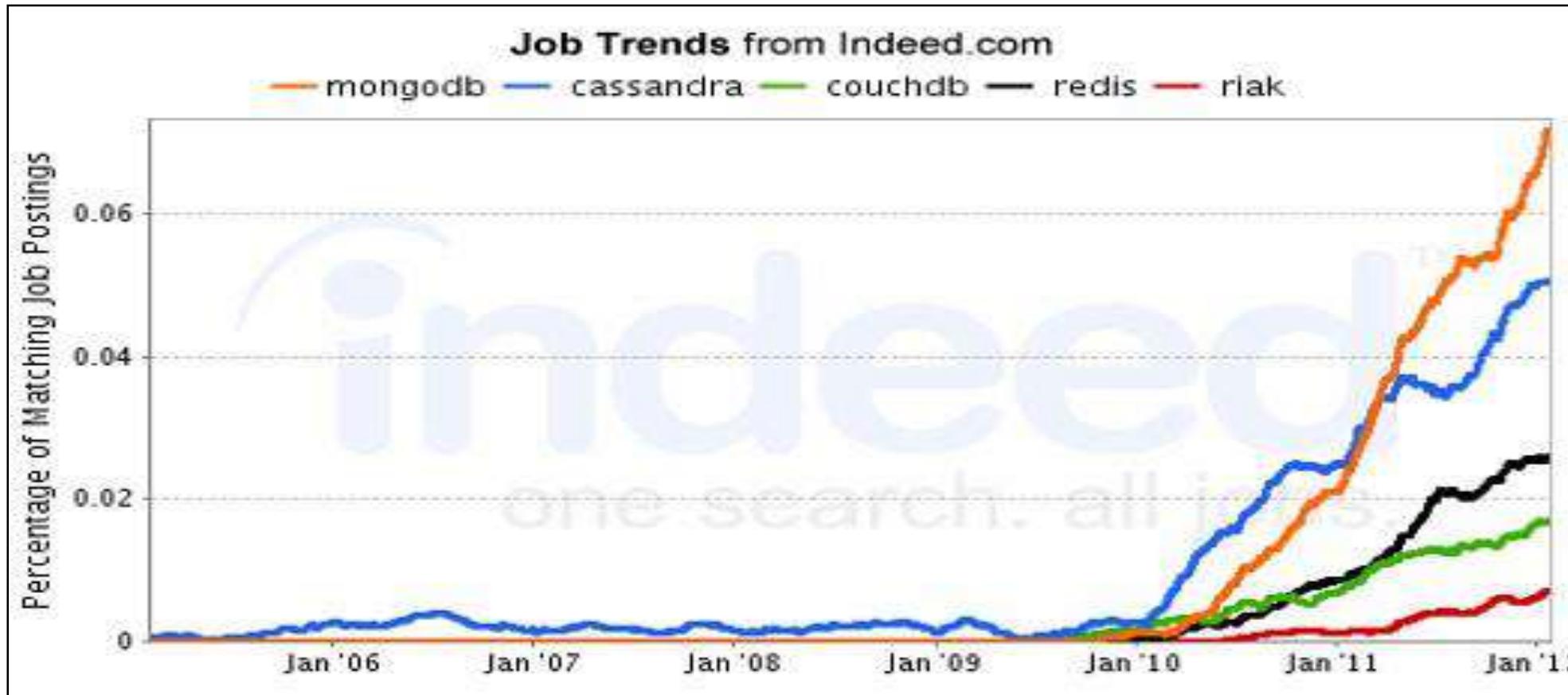
# Comparison of NoSQL Database

Attributes		NoSQL Databases								
Database model		Document-Stored		Wide-Column Stored				Key-Value Stored		Graph-oriented
Features	MongoDB	CouchDB	DynamoDB	HBase	Cassandra	Accumulo	Redis	Riak	Neo4j	
Distribution	Horizontal scalable	Yes	Yes	Yes	Yes	Yes		Yes		No
	Replication	Yes	Yes	Yes	Yes	Yes		Yes		Yes
	Replication mode	Master-Slave-Replica Replication	Master-Slave Replicatio n	-	Master-Slave Replicati on	Master-Slave Replicatio n	-	Master-Slave Replicati on	Multi-master replicati on	-
	Sharding	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
	Shared nothing architecture	Yes	Yes	Yes	Yes	Yes	-	-	Yes	-
	Value size max.	16MB	20MB	64KB	2TB	2GB	1EB	-	64MB	
System	Operating system	Cross-platform	Ubuntu Red Hat Windows Mac OS X	Cross-platform	Cross-platform	Cross-platform	NIX 32 entries Operating system	Linux *NIX Mac OS X Windows	Cross-platform	Cross-platform
	Programming language	C++	Erlang C++ C Python	Java	Java	Java	Java	C C++	Erlang	Java

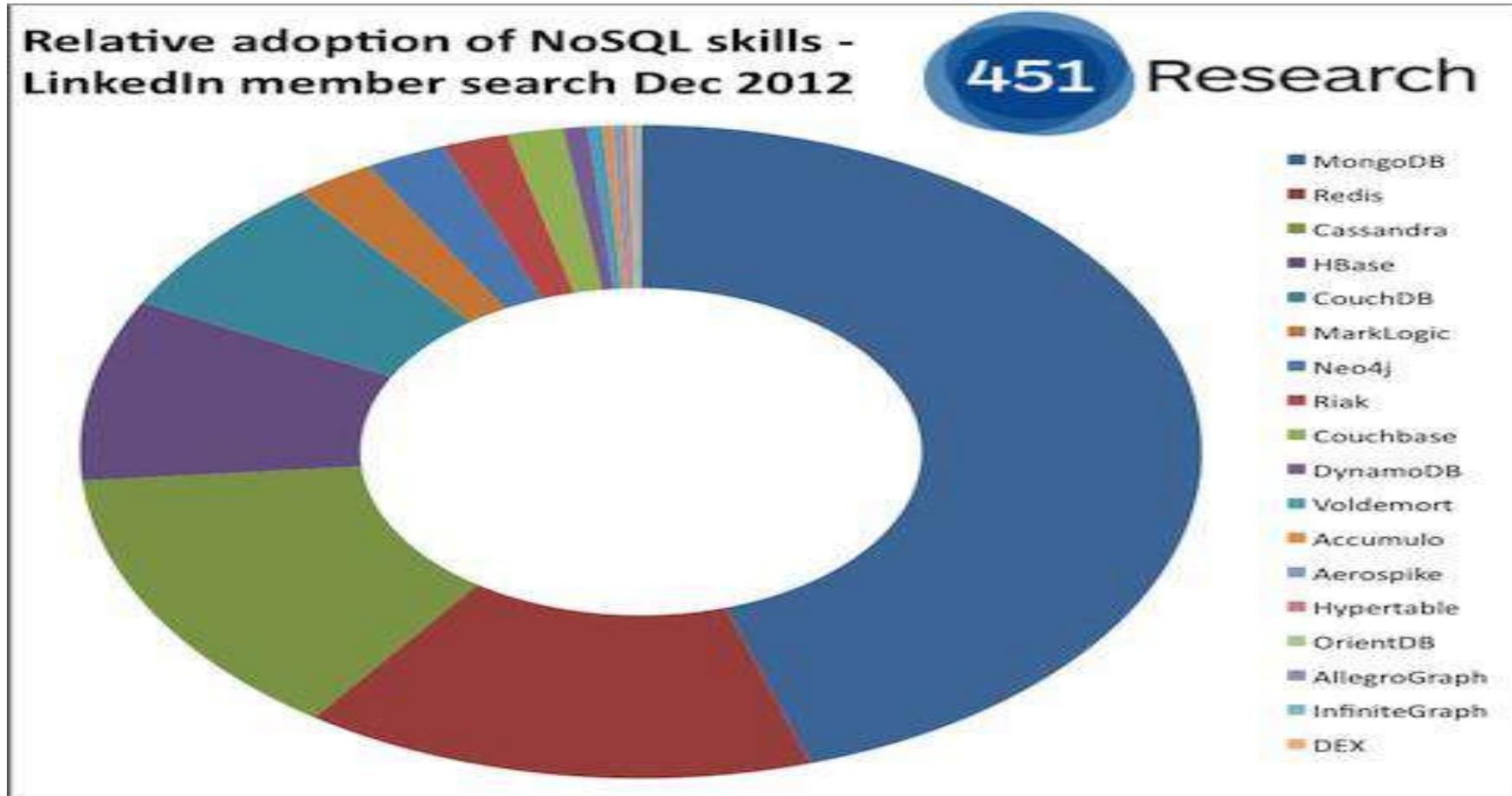
# Adoption of NoSQL Database



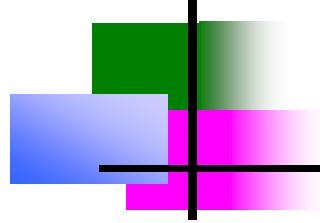
# Job trends of five NoSQL Databases (source: Indeed.com)



# NoSQL LinkedIn Skills Index – December 2012



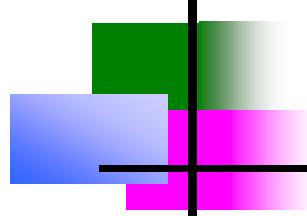
source: <http://blogs.the451group.com>



# Summary

---

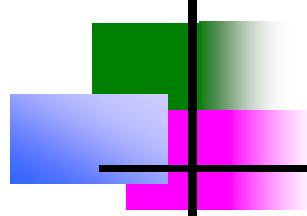
- ❖ Computational and storage requirements of applications led to the development of horizontally scalable, distributed non-relational No-SQL databases.
- ❖ Primary Uses
  - ✓ Large-scale data processing
  - ✓ Large volume data storage
  - ✓ Embedded IR (basic machine-to-machine information look-up & retrieval)
- ❖ CAP Theorem
- ❖ Flexibility, scalability



# References

---

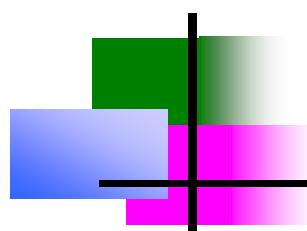
- [https://www.researchgate.net/publication/243963821\\_NoSQL\\_Database\\_New\\_Era\\_of\\_Databases\\_for\\_Big\\_data\\_Analytics\\_-Classification\\_Characteristics\\_and\\_Comparison](https://www.researchgate.net/publication/243963821_NoSQL_Database_New_Era_of_Databases_for_Big_data_Analytics_-Classification_Characteristics_and_Comparison)
- <https://www.slideshare.net/mayureesrikulwong/nosql-database-classification-characteristics-and>
- <https://www.linkedin.com/pulse/20141021201313-156372715-vertical-scaling-vs-horizontal-scaling-big-data/>



# Conclusion

---

- Computational and Storage Requirement unhandled by sql-like centralized DBs
  - Big Data Analytics, Business Intelligence, Social-Networking (peta-byte datasets)
- NoSQL – horizontally scalable, distributed non-relational DBs
- Motivational understanding of various types of NoSQL DBs.



# Chapter5

# Using Big Data for Analytics

## NoSQL

Basanta Joshi, PhD

Asst. Prof., Depart of Electronics and Computer Engineering

Deputy Director, MSc in Information and Communication Engineering

Member, Laboratory for ICT Research and Development (LICT)

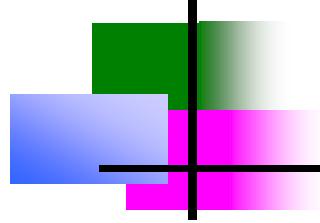
Institute of Engineering

basanta@ioe.edu.np

<http://www.basantajoshi.com.np>

<https://scholar.google.com/citations?user=iocLiGcAAAAJ>

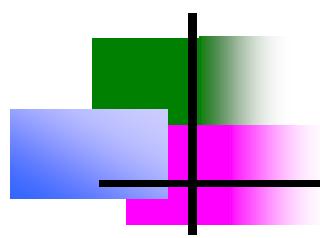
[https://www.researchgate.net/profile/Basanta\\_Joshi2](https://www.researchgate.net/profile/Basanta_Joshi2)



# CONTENTS

---

- Structured and unstructured data
- Taxonomy of NoSQL implementation
- Hbase
- Cassandra
- MongoDB



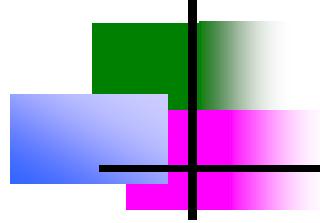
# STRUCTURED AND UNSTRUCTURED DATA

---

## Types of Data

- Structured
- Semi-structured
- Unstructured



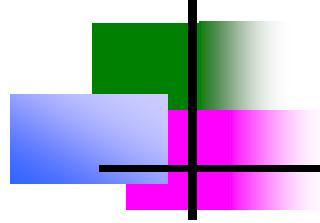


# STRUCTURED DATA

---

- “normal” RDBMS data
- Format is known and defined
- Example: Sales Order

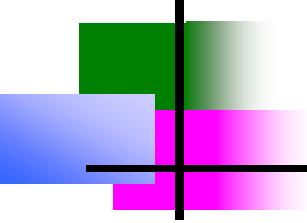
	Sales.SalesOrderHeader
	Columns
	SalesOrderID (PK, int, not null)
	RevisionNumber (tinyint, not null)
	OrderDate (datetime, not null)
	DueDate (datetime, not null)
	ShipDate (datetime, null)
	Status (tinyint, not null)
	OnlineOrderFlag (Flag(bit), not null)
	SalesOrderNumber (Computed, nvarchar(25), not null)
	PurchaseOrderNumber (OrderNumber(nvarchar(25)), null)
	AccountNumber (AccountNumber(nvarchar(15)), null)
	CustomerID (FK, int, not null)
	SalesPersonID (FK, int, null)
	TerritoryID (FK, int, null)
	BillToAddressID (FK, int, not null)
	ShipToAddressID (FK, int, not null)
	ShipMethodID (FK, int, not null)
	CreditCardID (FK, int, null)
	CreditCardApprovalCode (varchar(15), null)
	CurrencyRateID (FK, int, null)
	SubTotal (money, not null)
	TaxAmt (money, not null)
	Freight (money, not null)
	TotalDue (Computed, money, not null)
	Comment (nvarchar(128), null)
	rowguid (uniqueidentifier, not null)
	ModifiedDate (datetime, not null)



# SEMI-STRUCTURED DATA

---

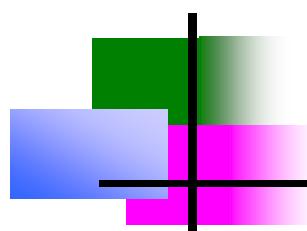
- some structure, but it is fluid
- changes in structure should not break code
- example: XML



# SEMI STRUCTURED DATA

---

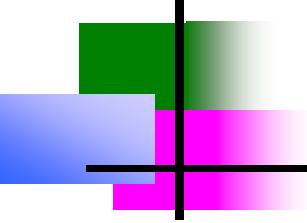
```
<SalesOrder DueDate="20120201">  
  <OrderID>12</OrderID>  
  <Customer>John Doe</Customer>  
  <OrderDate>2012/01/15</OrderDate>  
  Items  
    <Item>  
      <Product>Widget</Product>  
      <Quantity>12</Quantity>  
    </Item>  
    <Item>  
      <Product>Whatchamacallit</Product>  
      <Quantity>2</Quantity>  
    </Item>  
  Items  
</SalesOrder>
```



# SEMI STRUCTURED DATA

---

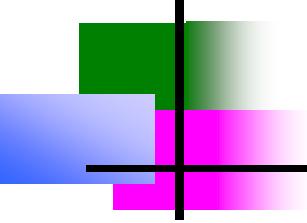
```
<SalesOrder DueDate="20120201">  
  <OrderID>12</OrderID>  
  <Customer>John Doe</Customer>  
  <OrderDate>2012/01/15</OrderDate>  
  <Items>  
    <Item>  
      <Product>Widget</Product>  
      <Quantity>12</Quantity>  
    </Item>  
    <Item>  
      <Product>Whatchamacallit</Product>  
      <Quantity>2</Quantity>  
    </Item>  
  </Items>  
</SalesOrder>
```



# SEMI STRUCTURED DATA

---

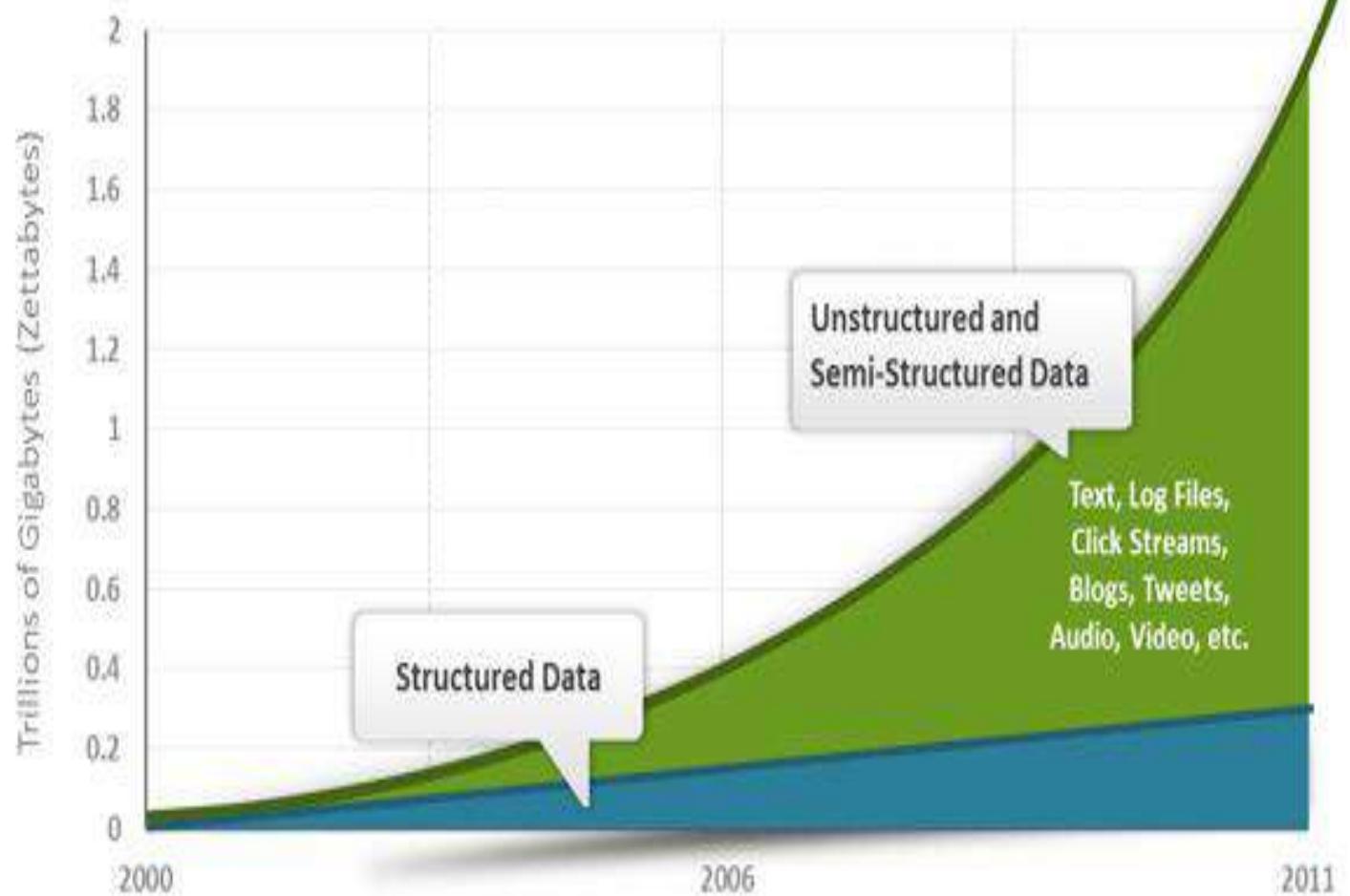
```
<SalesOrder DueDate="20120201">
  <OrderID>12</OrderID>
  <Customer>John Doe</Customer>
  <OrderDate>2012/01/15</OrderDate>
  <Items>
    <Item>
      <Product>Widget</Product>
      <Quantity>12</Quantity>
    </Item>
    <Item>
      <Product>Whatchamacallit</Product>
      <Quantity>2</Quantity>
    </Item>
  </Items>
</SalesOrder>
```



# UNSTRUCTURED DATA

---

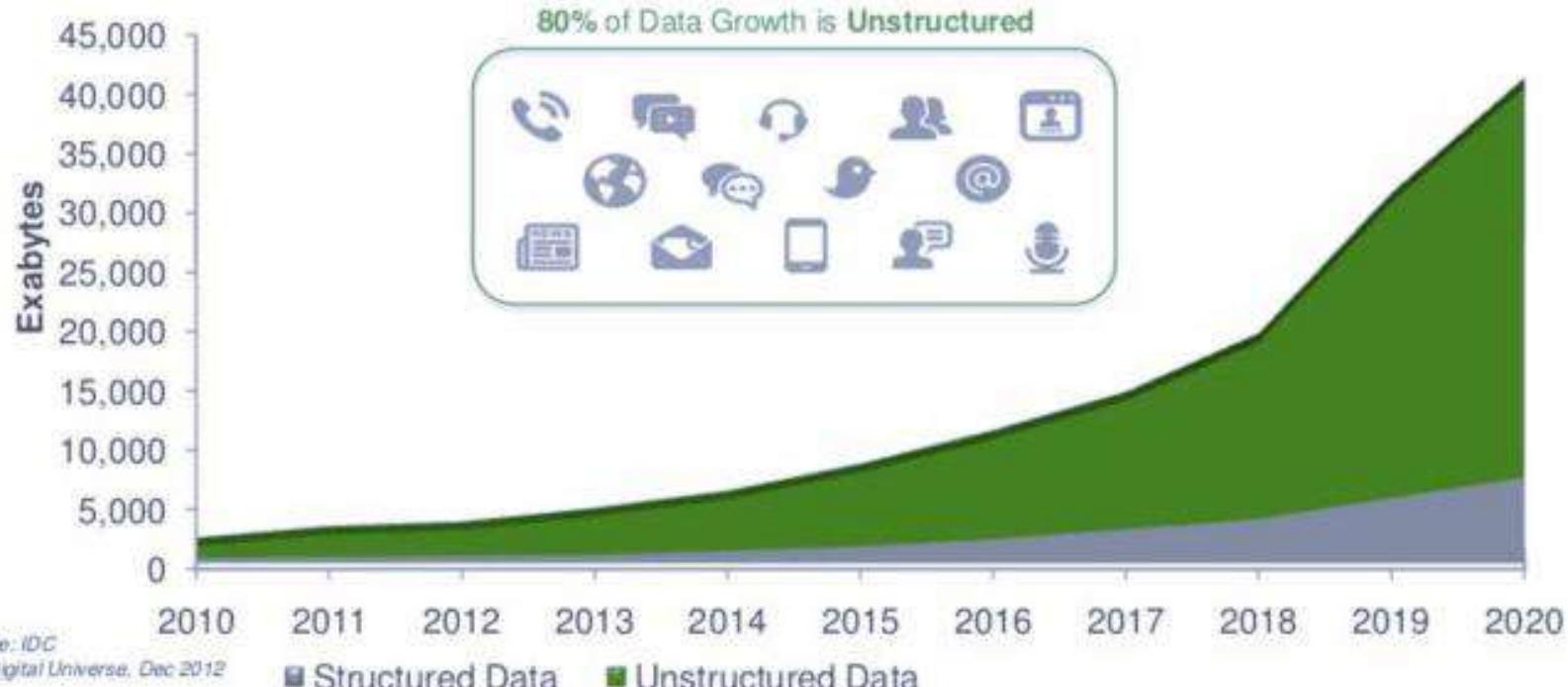
- structure is merely encoding.
- meta data may be in the structure
- examples:
  - Audio files
  - Word Documents
  - PDF
  - Movies

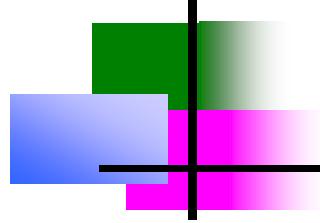


## MASSIVE GROWTH IN UNSTRUCTURED CONTENT



Worldwide Corporate Data Growth

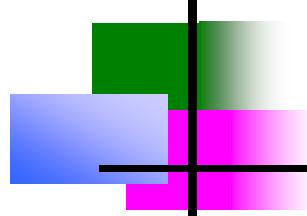




# NOSQL

---

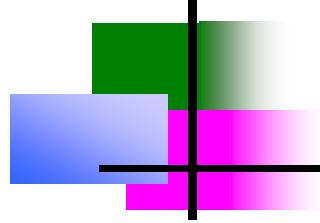
- Overview of NoSQL databases
- The need of NoSQL databases
- “Battle” between SQL and NoSQL databases
- CAP Theorem
- Challenges of NoSQL databases



# NOSQL: AN OVERVIEW OF NOSQL DATABASES

---

- the term means Not Only SQL
- It's not SQL and it's not relational. NoSQL is designed for distributed data stores for very large scale data needs.
- In a NoSQL database, there is no fixed schema and no joins. Scaling out refers to spreading the load over many commodity systems. This is the component of NoSQL that makes it an inexpensive solution for large datasets.



# QUERYING NOSQL

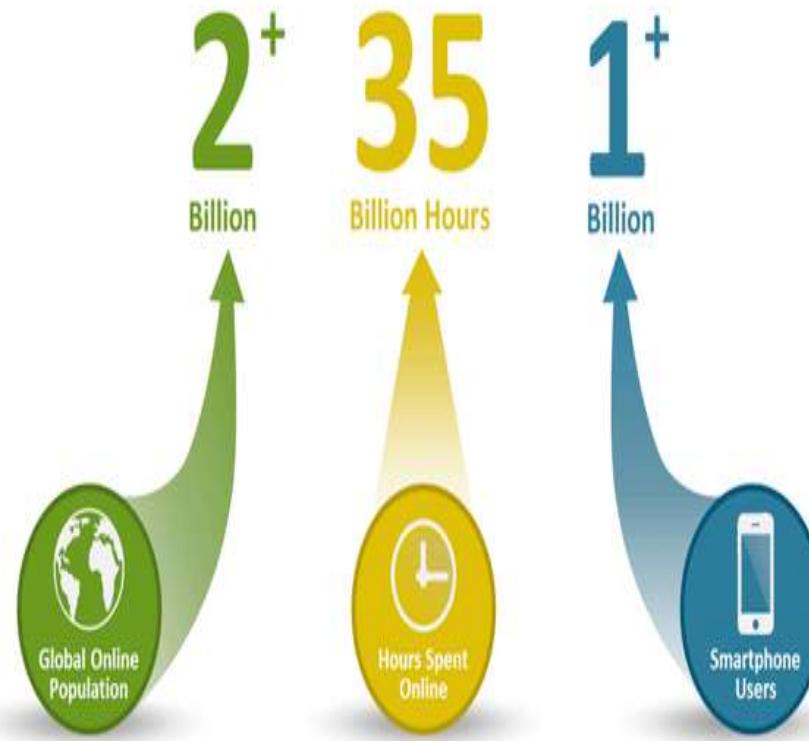
---

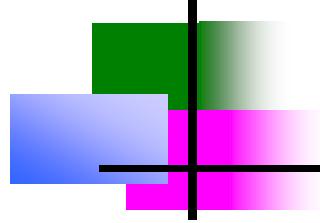
- The question of how to query a NoSQL database is what most developers are interested in. After all, data stored in a huge database doesn't do anyone any good if you can't retrieve and show it to end users or web services. NoSQL databases do not provide a high level declarative query language like SQL. Instead, querying these databases is data-model specific.
- Many of the NoSQL platforms allow for RESTful interfaces to the data. Others offer query APIs. There are a couple of query tools that have been developed that attempt to query multiple NoSQL databases. These tools typically work across a single NoSQL category.
- One example is [SPARQL](#). SPARQL is a declarative query specification designed for graph databases.
- Here is an example of a SPARQL query that retrieves the URL of a particular blogger (courtesy of [IBM](#)):

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?url
FROM <bloggers.rdf>
WHERE {
  ?contributor foaf:name "Jon Foobar" .
  ?contributor foaf:weblog ?url .
}
```

# WHY NOSQL?

- Three trends disrupting the database status quo—
  - Big Data,
  - Big Users, and
  - Cloud Computing
- ***Big Users*** : Not that long ago, 1,000 daily users of an application was a lot and 10,000 was an extreme case.
- Today, with the growth in global Internet use, the increased number of hours users spend online, and the growing popularity of smartphones and tablets, it's not uncommon for apps to have millions of users a day.
- 





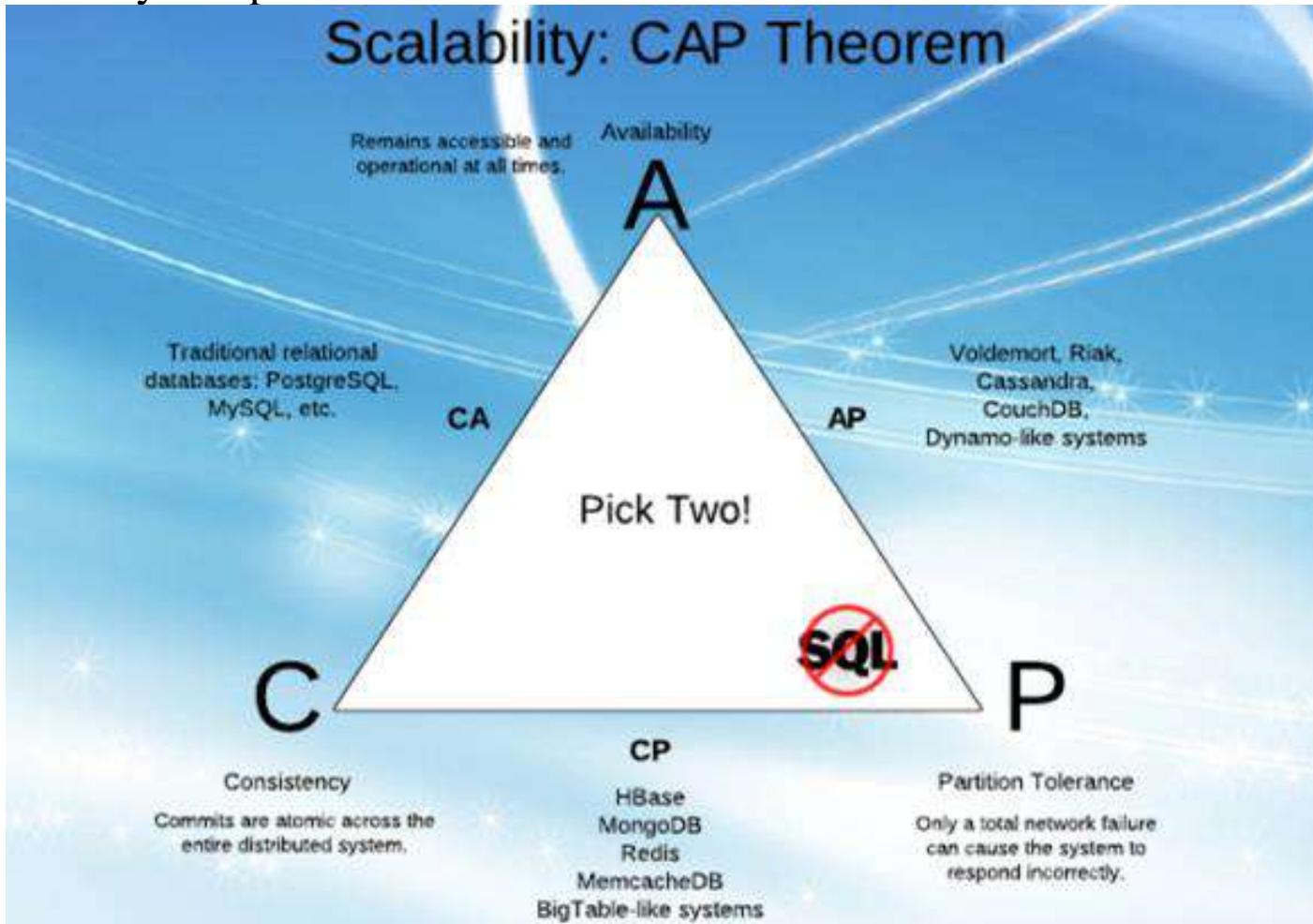
# WHY NOSQL?

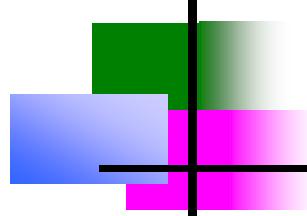
---

- ***Big Data*** :Data is becoming easier to capture and access through third parties such as Facebook, D&B, and others. Personal user information, geo location data, social graphs, user-generated content, machine logging data, and sensor-generated data are just a few examples of the ever-expanding array of data being captured.
- ***Cloud Computing***: Today, most new applications (both consumer and business) use a three-tier Internet architecture, run in a public or private cloud, and support large numbers of users.

# WHAT IS CAP?

- The CAP Theorem states that it is impossible for any shared-data system to *guarantee* simultaneously all of the following three properties: consistency, availability and partition tolerance.

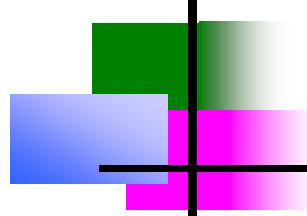




# WHAT IS CAP?

---

- Consistency in CAP is not the same as consistency in ACID (that would be too easy). According to CAP, consistency in a database means that whenever data is written, everyone who reads from the database will always see the latest version of the data. A database without strong consistency means that when the data is written, not everyone who reads from the database will see the new data right away; this is usually called eventual-consistency or weak consistency.
- Availability in a database according to CAP means you always can expect the database to be there and respond whenever you query it for information. High availability usually is accomplished through large numbers of physical servers acting as a single database through sharing (splitting the data between various database nodes) and replication (storing multiple copies of each piece of data on different nodes).
- Partition tolerance in a database means that the database still can be read from and written to when parts of it are completely inaccessible. Situations that would cause this include things like when the network link between a significant number of database nodes is interrupted. Partition tolerance can be achieved through some sort of mechanism whereby writes destined for unreachable nodes are sent to nodes that are still accessible. Then, when the failed nodes come back, they receive the writes they missed



# NOSQL: TAXONOMY OF IMPLEMENTATION

---

The current NoSQL world fits into 4 basic categories.

- **Key-values Stores**
- **Column Family Stores** were created to store and process very large amounts of data distributed over many machines. There are still keys but they point to multiple columns. In the case of [BigTable](#) (Google's Column Family NoSQL model), rows are identified by a row key with the data sorted and stored by this key. The columns are arranged by column family. E.g. **Cassandra**
- **Document Databases** were inspired by [Lotus Notes](#) and are similar to key-value stores. The model is basically versioned documents that are collections of other key-value collections. The semi-structured documents are stored in formats like [JSON](#) e.g. [MongoDB](#)
- **Graph Databases** are built with nodes, relationships between notes and the properties of nodes. Instead of tables of rows and columns and the rigid structure of SQL, a flexible graph model is used which can scale across many machines.

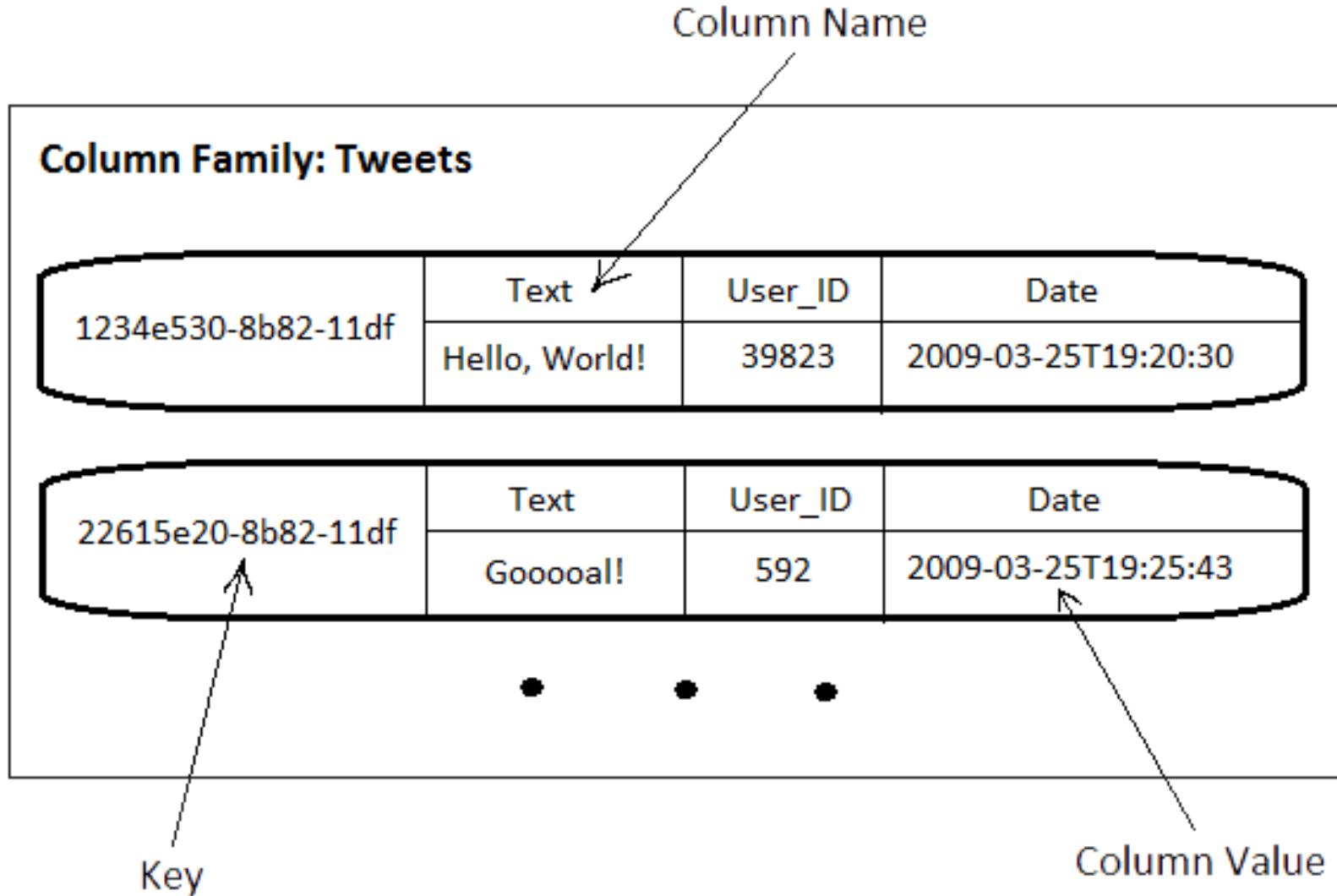
# KEY-VALUES STORES

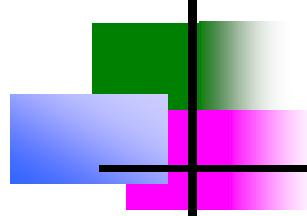
---

- The Key-Value database is a very simple structure based on Amazon's Dynamo DB. Data is indexed and queried based on it's key. Key-value stores provide consistent hashing so they can scale incrementally as your data scales. They communicate node structure through a gossip-based membership protocol to keep all the nodes synchronized. If you are looking to scale very large sets of low complexity data, key-value stores are the best option.
- **Examples**
  - [Riak](#)
  - [Voldemort](#)
  - Tokyo Cabinet

key	value
firstName	Bugs
lastName	Bunny
location	Earth

# COLUMN FAMILY STORES

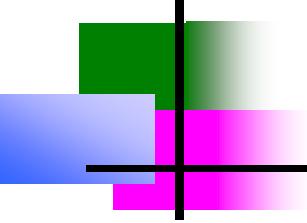




# COLUMN FAMILY STORES

---

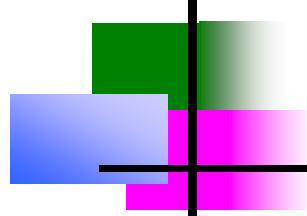
- These data stores are based on Google's BigTable implementation.
- may look similar to relational databases on the surface but under the hood a lot has changed.
- A column family database can have different columns on each row so it is not relational and doesn't have what qualifies in an RDBMS as a table.
- The only key concepts in a column family database are columns, column families and super columns.
- Column families define how the data is structured on disk. A column by itself is just a key-value pair that exists in a column family.
- A super column is like a catalogue or a collection of other columns except for other super columns.
- Column family databases are still extremely scalable but less-so than key-value stores. However, they work better with more complex data sets.
- **Examples of Column family databases are:**
  - [Apache HBase](#)
  - [Hypertable](#)
  - [Cassandra](#)



# COLUMN FAMILY STORES

---

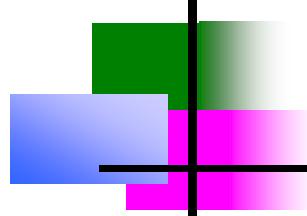
- Equivalent to “relational table” – “set of rows” identified by key
- Concept starts with columns
- Data is organized in the columns
- Columns are stored contiguously
- Columns tend to have similar data
- A row can have as many columns as needed
- Columns are essentially keys, that can let you lookup values in the rows
- Columns can be added any time, NULL don’t exist
- Unused columns in a row does not occupy storage
- Write Data to Append Only file, an extremely efficient and makes write are significantly faster then write
- Built in control about replication and distribution
- HBase :From CAP Theorem
  - Partition Tolerance
  - Consistency
- Cassandra :From CAP Theorem
  - Partition Tolerance
  - Availability Note: Tradeoff between availability and consistency is tunable



# DOCUMENT DATABASES

---

- A document database is not a new idea. It was used to power one of the more prominent communication platforms of the 90's and still in service today, Lotus Notes now called Lotus Domino. APIs for document DBs use Restful web services and JSON for message structure making them easy to move data in and out.
- A document database has a fairly simple data model based on collections of key-value pairs.
- A typical record in a document database would look like this:
- ```
{ "Subject": "I like Plankton"
  "Author": "Rusty"
  "PostedDate": "5/23/2006"
  "Tags": ["plankton", "baseball", "decisions"]
  "Body": "I decided today that I don't like baseball. I like plankton." }
```
- Document databases improve on handling more complex structures but are slightly less scalable than column family databases.
- **Examples of document databases are:**
  - [CouchDB](#)
  - [MongoDB](#)

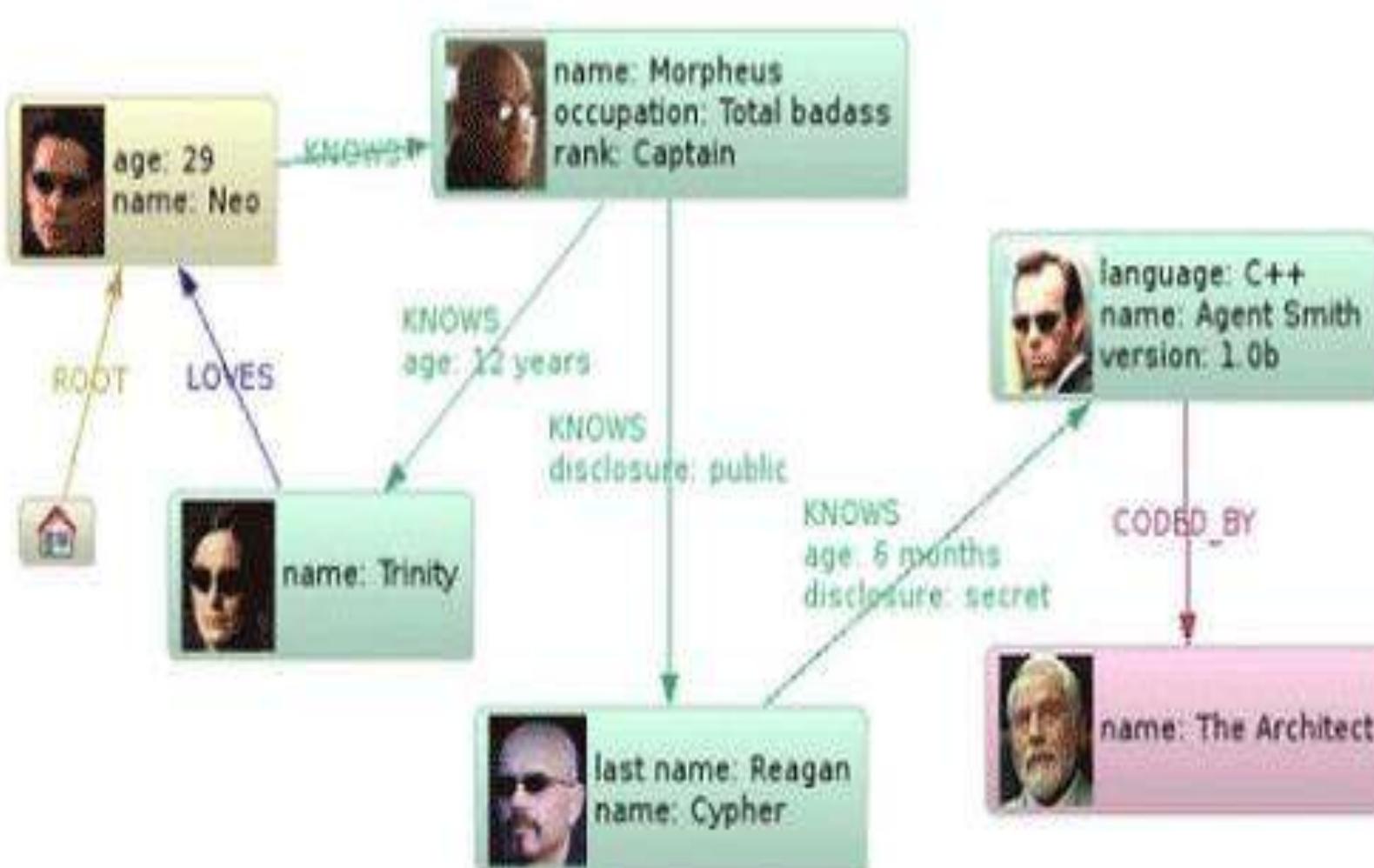


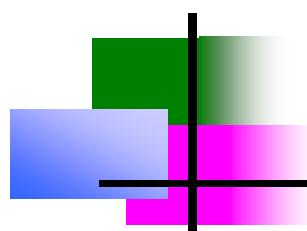
# DOCUMENT DATABASES

---

- Documents are key-value pair
- document can also be stored in JSON format
- Because of JSON document considered as object
- JSON documents are used as Key-Value pairs
- Document can have any set of keys
- Any key can associate with any arbitrarily complex value, that is itself a JSON document
- Documents are added with different sets of keys
  - Missing keys
  - Extra keys
  - Add keys in future when in need
  - Application must know that certain key present
  - Queries are made on Keys
  - Index are set to keys to make search efficient

# GRAPH DATABASES

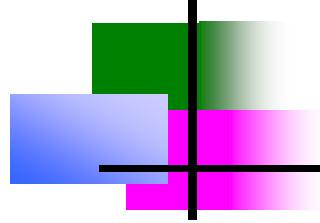




# GRAPH DATABASES

---

- Graph databases take document databases to the extreme by introducing the concept of type relationships between documents or nodes. The most common example is the relationship between people on a social network such as Facebook. The idea is inspired by the graph theory work by Leonhard Euler, the 18th century mathematician. Key/Value stores used key-value pairs as their modeling units. Column Family databases use the tuple with attributes to model the data store. A graph database is a big dense network structure.
- While it could take an RDBMS hours to sift through a huge linked list of people, a graph database uses sophisticated shortest path algorithms to make data queries more efficient. Although slower than its other NoSQL counterparts, a graph database can have the most complex structure of them all and still traverse billions of nodes and relationships with light speed.
- **Examples of Graph Databases are:**
  - [AllegroGraph](#)
  - [Sones](#)
  - [Neo4j](#)



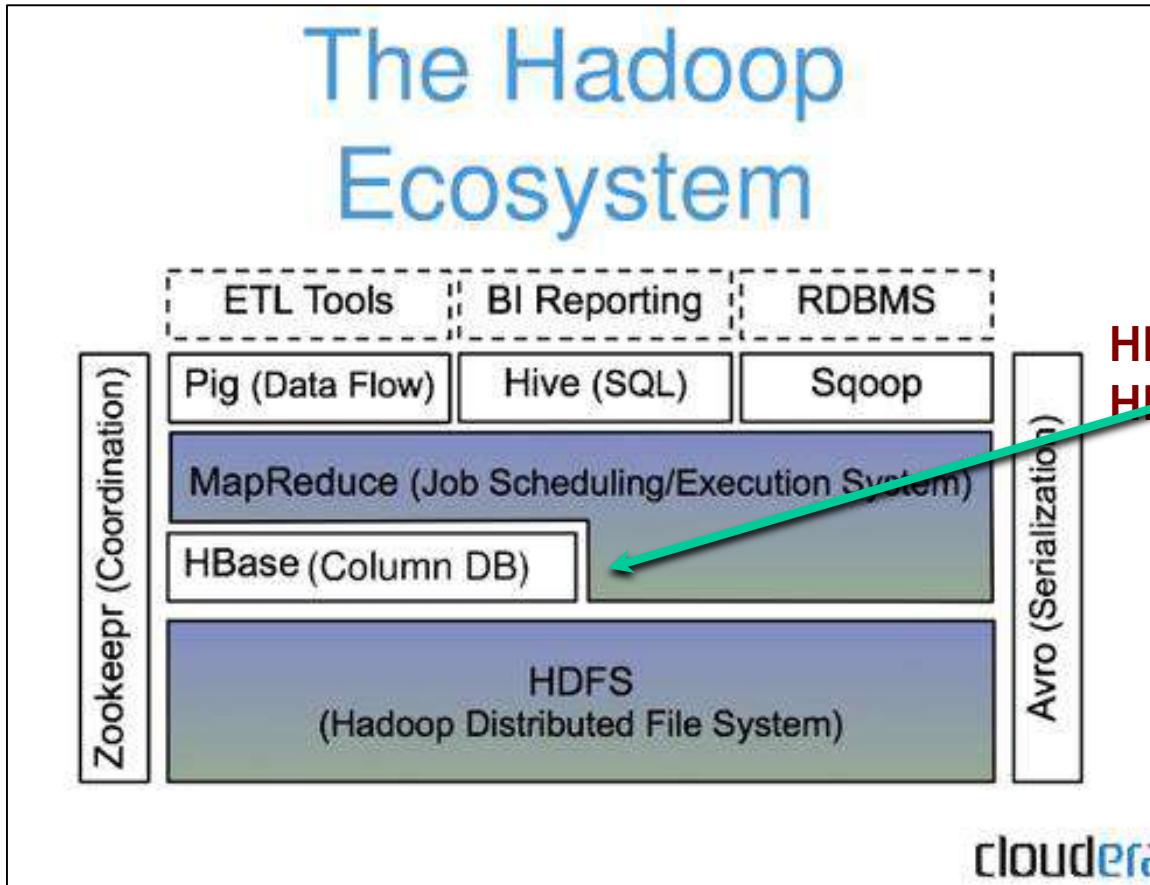
# HBASE: OVERVIEW

---

- HBase is a distributed column-oriented data store built on top of HDFS
- HBase is an Apache open source project whose goal is to provide storage for the Hadoop Distributed Computing
- Data is logically organized into tables, rows and columns

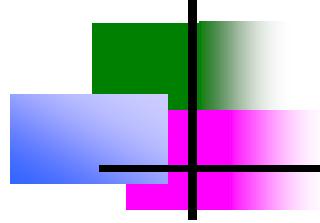


# HBASE: PART OF HADOOP'S ECOSYSTEM



HBase is built on top of  
HDFS

HBase files are  
internally  
stored in HDFS



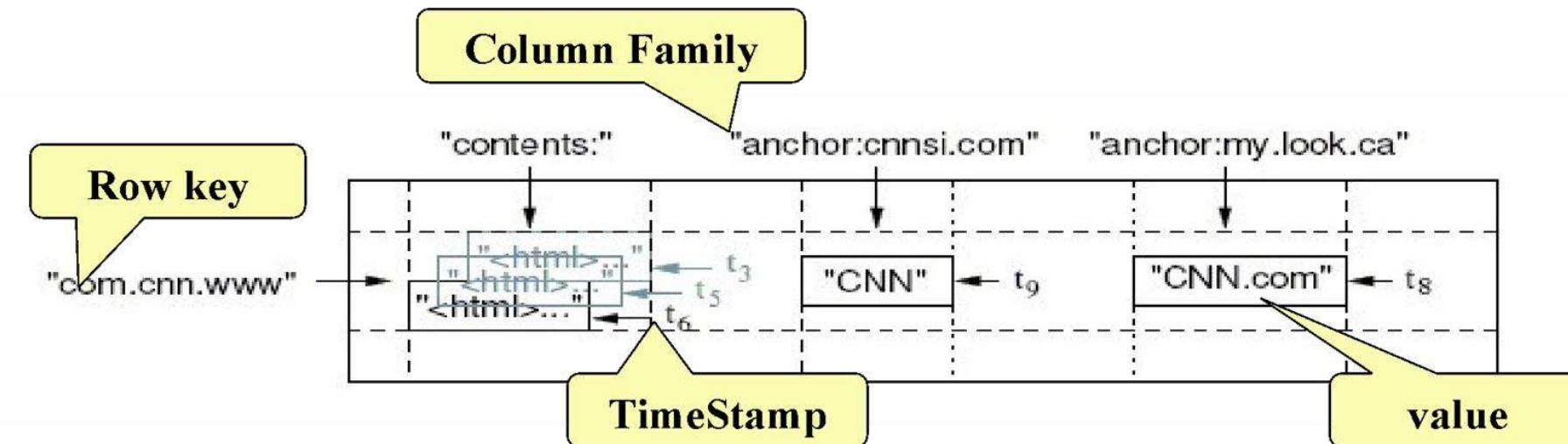
## HBASE VS. HDFS (CONT'D)

---

- *HBase* is designed to efficiently address
  - Fast record lookup
  - Support for record-level insertion
  - Support for updates (not in place)
- HBase updates are done by creating new versions of values

# HBASE DATA MODEL

- HBase is based on Google's Bigtable model
  - Key-Value pairs



# HBASE LOGICAL VIEW

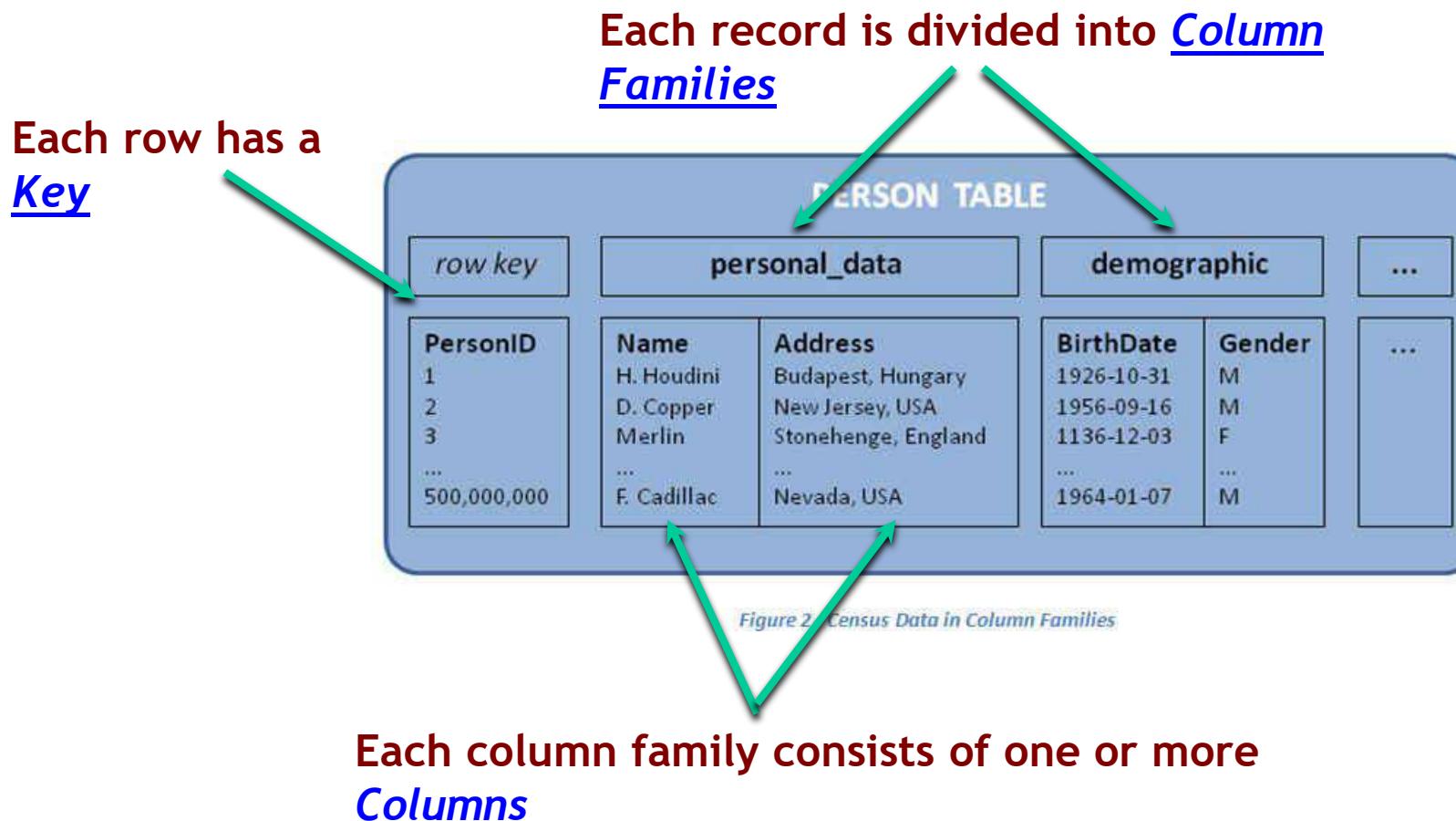
| Implicit PRIMARY KEY in RDBMS terms |                                                                                                                                               | Data is all byte[ ] in HBase                                      |
|-------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------|
| Row key                             | Data                                                                                                                                          |                                                                   |
| cutting                             | info: { 'height': '9ft', 'state': 'CA' }<br>roles: { 'ASF': 'Director', 'Hadoop': 'Founder' }                                                 |                                                                   |
| tipcon                              | info: { 'height': '5ft7', 'state': 'CA' }<br>roles: { 'Hadoop': 'Committer' @ts=2010,<br>'Hadoop': 'PMC' @ts=2011,<br>'Hive': 'Contributor' } | A single cell might have different values at different timestamps |

Different types of data separated into different “column families”

Different rows may have different sets of columns (table is sparse)

Useful for \*-To-Many mappings

# HBASE: KEYS AND COLUMN FAMILIES



# HBASE: KEYS AND COLUMN FAMILIES

- **Key**
  - Byte array
  - Serves as the primary key for the table
  - Indexed far fast lookup
- **Column Family**
  - Has a name (string)
  - Contains one or more related columns
- **Column**
  - Belongs to one column family
  - Included inside the row
    - *familyName:columnName*

| Row key            | Time Stamp | Column “content s.” | Column “anchor:”               |
|--------------------|------------|---------------------|--------------------------------|
| “com.apac he.ww w” | t12        | “<html><br>...”     |                                |
|                    | t11        | “<html><br>...”     | Column named “apache.com”      |
|                    | t10        |                     | “anchor:apache.com” “APACHE”   |
|                    | t15        |                     | “anchor:cnnsi.co m” “CNN”      |
|                    | t13        |                     | “anchor:my.look.ca” “CNN.co m” |
|                    | t6         | “<html><br>...”     |                                |
|                    | t5         | “<html><br>...”     |                                |
|                    | t3         | “<html><br>...”     |                                |

# HBASE: KEYS AND COLUMN FAMILIES

- **Version Number**
  - Unique within each key
  - By default System's timestamp
  - Data type is Long
- **Value (Cell)**
  - Byte array

Version number for each row

| Row key          | Time Stamp | Column "content s:" | Column "anchor:"              |
|------------------|------------|---------------------|-------------------------------|
| "com.apache.www" | t12        | "<html> ... "       | value                         |
|                  | t11        | "<html> ... "       |                               |
|                  | t10        |                     | "anchor:apache.com" "APACHE"  |
|                  | t15        |                     | "anchor:cnnsi.com" "CNN"      |
|                  | t13        |                     | "anchor:my.look.ca" "CNN.com" |
|                  | t6         | "<html> ... "       |                               |
|                  | t5         | "<html> ... "       |                               |
|                  | t3         | "<html> ... "       |                               |

# HBASE DATA MODEL

- HBase schema consists of several **Tables**
  - Each table consists of a set of **Column Families**
    - Columns are not part of the schema
  - HBase has **Dynamic Columns**
    - Because column names are encoded inside the cells
    - Different cells

“Roles” column family has different columns in different cells

| S | Row key | Data                                                                                                                                          |
|---|---------|-----------------------------------------------------------------------------------------------------------------------------------------------|
|   | cutting | info: { 'height': '9ft', 'state': 'CA' }<br>roles: { 'ASF': 'Director', 'Hadoop': 'Founder' }                                                 |
|   | tipcon  | info: { 'height': '5ft7', 'state': 'CA' }<br>roles: { 'Hadoop': 'Committer' @ts=2010,<br>'Hadoop': 'PMC' @ts=2011,<br>'Hive': 'Contributor' } |

# HBASE DATA MODEL

- The ***version number*** can be user-supplied
  - Even does not have to be inserted in increasing order
  - Version number are unique within each key
- Table can be very sparse
  - Many cells are empty
- ***Keys*** are indexed as the primary key

Has two columns  
[cnnsi.com &  
my.look.ca]

| Row Key       | Time Stamp | ColumnFamily contents       | ColumnFamily anchor           |
|---------------|------------|-----------------------------|-------------------------------|
| "com.cnn.www" | t9         |                             | anchor:cnnsi.com = "CNN"      |
| "com.cnn.www" | t8         |                             | anchor:my.look.ca = "CNN.com" |
| "com.cnn.www" | t6         | contents:html = "<html>..." |                               |
| "com.cnn.www" | t5         | contents:html = "<html>..." |                               |
| "com.cnn.www" | t3         | contents:html = "<html>..." |                               |

# HBASE PHYSICAL MODEL

- Each column family is stored in a separate file (called ***HTables***)
- Key & Version numbers are replicated with each column family
- Empty cells are not stored

HBase maintains a multi-level index on values:  
*<key, column family, column name, timestamp>*

**Table 5.3. ColumnFamily contents**

| Row Key       | Time Stamp | ColumnFamily "contents:"    |
|---------------|------------|-----------------------------|
| "com.cnn.www" | t6         | contents:html = "<html>..." |
| "com.cnn.www" | t5         | contents:html = "<html>..." |
| "com.cnn.www" | t3         | contents:html = "<html>..." |

**Table 5.2. ColumnFamily anchor**

| Row Key       | Time Stamp | Column Family anchor          |
|---------------|------------|-------------------------------|
| "com.cnn.www" | t9         | anchor:cnnsi.com = "CNN"      |
| "com.cnn.www" | t8         | anchor:my.look.ca = "CNN.com" |

# EXAMPLE

| Row key | Data                                                                                                                                          |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| cutting | info: { 'height': '9ft', 'state': 'CA' }<br>roles: { 'ASF': 'Director', 'Hadoop': 'Founder' }                                                 |
| tipcon  | info: { 'height': '5ft7', 'state': 'CA' }<br>roles: { 'Hadoop': 'Committer' @ts=2010,<br>'Hadoop': 'PMC' @ts=2011,<br>'Hive': 'Contributor' } |

## **info Column Family**

| Row key | Column key  | Timestamp     | Cell value |
|---------|-------------|---------------|------------|
| cutting | info:height | 1273516197868 | 9ft        |
| cutting | info:state  | 1043871824184 | CA         |
| tlipcon | info:height | 1273878447049 | 5ft7       |
| tlipcon | info:state  | 1273616297446 | CA         |

## **roles Column Family**

| Row key | Column key   | Timestamp     | Cell value  |
|---------|--------------|---------------|-------------|
| cutting | roles:ASF    | 1273871823022 | Director    |
| cutting | roles:Hadoop | 1183746289103 | Founder     |
| tlipcon | roles:Hadoop | 1300062064923 | PMC         |
| tlipcon | roles:Hadoop | 1293388212294 | Committer   |
| tlipcon | roles:Hive   | 1273616297446 | Contributor |

Sorted  
on disk by  
Row key, Col  
key,  
descending  
timestamp

Milliseconds since unix epoch

cloudera

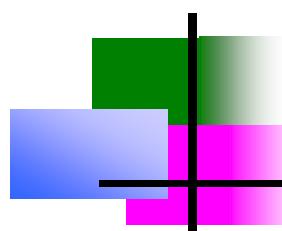
# HBASE REGIONS

- Each HTable (column family) is partitioned horizontally into *regions*
  - Regions are counterpart to HDFS blocks

**Table 5.3. ColumnFamily contents**

| Row Key       | Time Stamp | ColumnFamily "contents:"    |
|---------------|------------|-----------------------------|
| "com.cnn.www" | t6         | contents:html = "<html>..." |
| "com.cnn.www" | t5         | contents:html = "<html>..." |
| "com.cnn.www" | t3         | contents:html = "<html>..." |
|               |            |                             |
|               |            |                             |
|               |            |                             |

*Each will be one  
region*

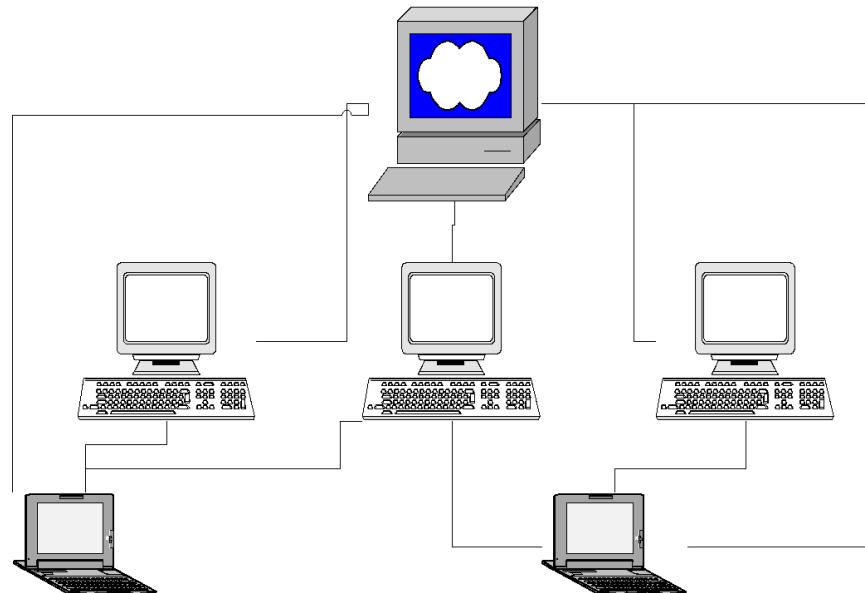


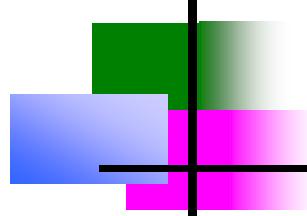
# HBASE ARCHITECTURE

# THREE MAJOR COMPONENTS

---

- The HBaseMaster
  - One master
- The HRegionServer
  - Many region servers
- The HBase client



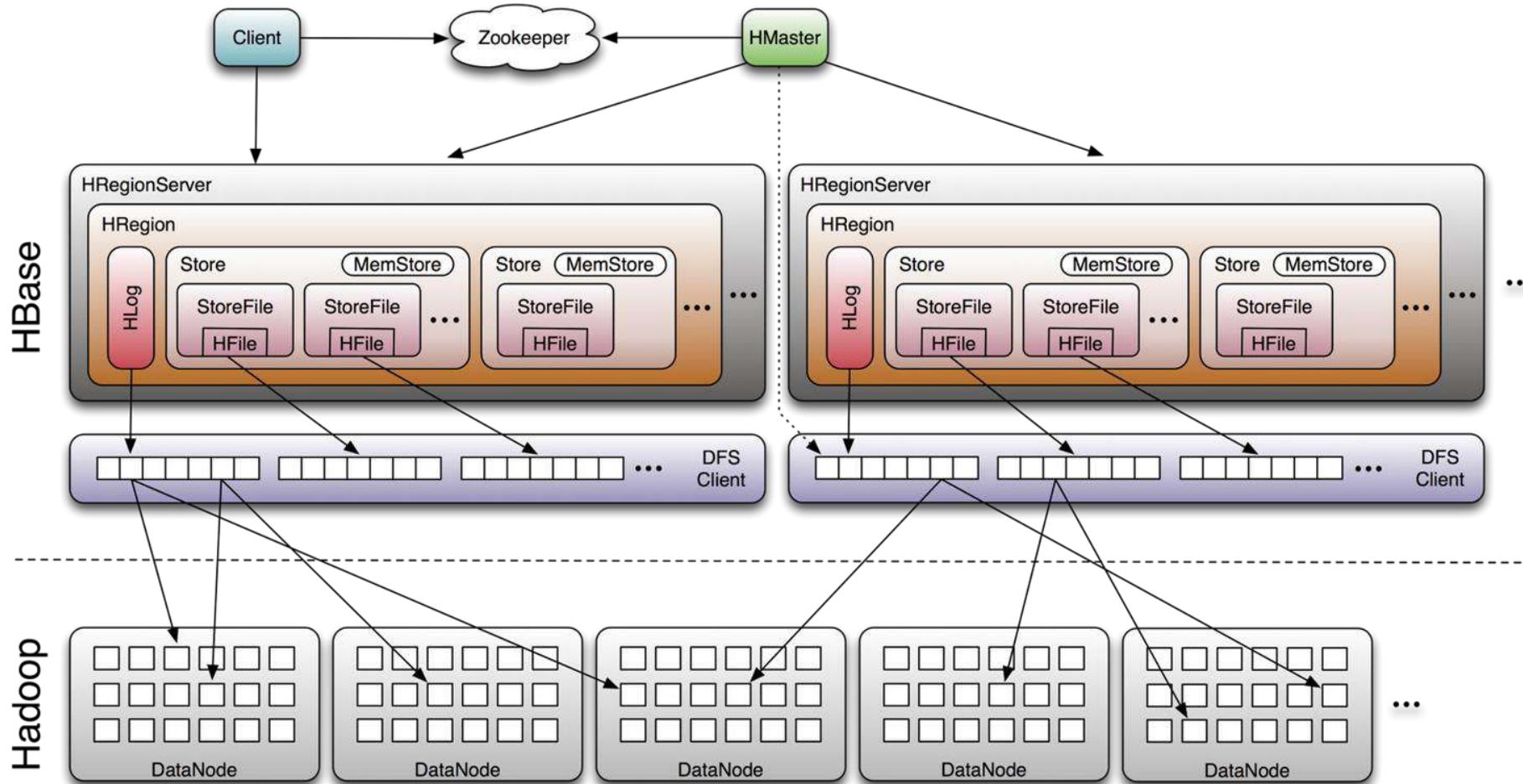


# HBASE COMPONENTS

---

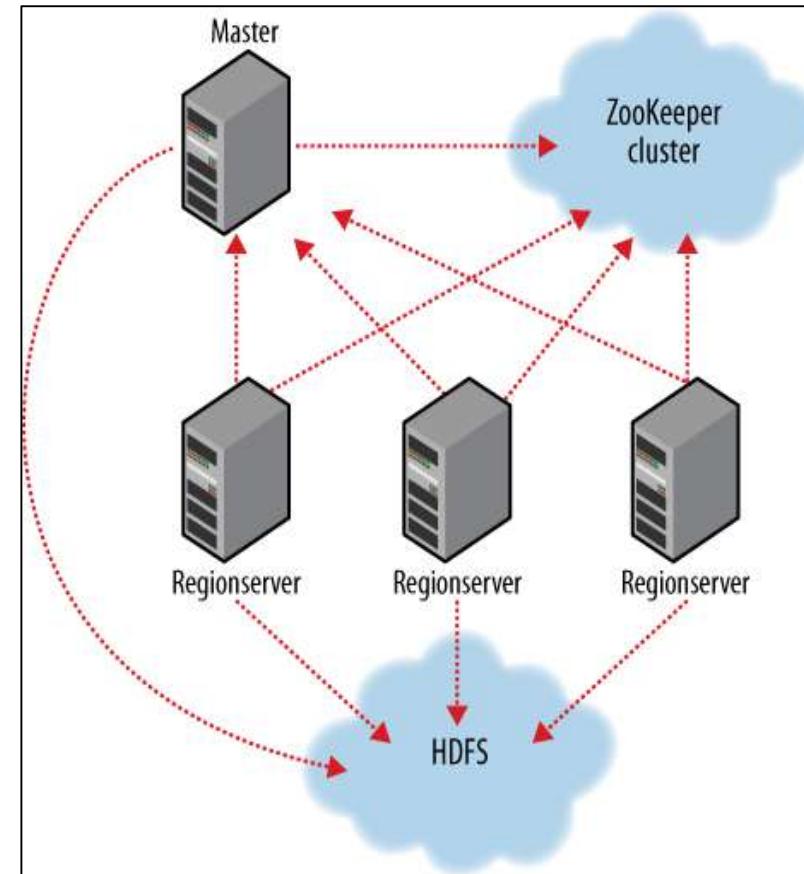
- **Region**
  - A subset of a table's rows, like horizontal range partitioning
  - Automatically done
- **RegionServer (many slaves)**
  - Manages data regions
  - Serves data for reads and writes (*using a log*)
- **Master**
  - Responsible for coordinating the slaves
  - Assigns regions, detects failures
  - Admin functions

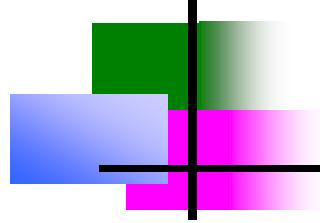
# BIG PICTURE



# ZOOKEEPER

- HBase depends on ZooKeeper
- By default HBase manages the ZooKeeper instance
  - E.g., starts and stops ZooKeeper
- HMaster and HRegionServers register themselves with ZooKeeper





# CREATING A TABLE

---

```
HBaseAdmin admin= new HBaseAdmin(config);
HColumnDescriptor []column;
column= new HColumnDescriptor[2];
column[0]=new HColumnDescriptor("columnFamily1:");
column[1]=new HColumnDescriptor("columnFamily2:");
HTableDescriptor desc= new
    HTableDescriptor(Bytes.toBytes("MyTable"));
desc.addFamily(column[0]);
desc.addFamily(column[1]);
admin.createTable(desc);
```

# OPERATIONS ON REGIONS: GET()

- Given a key □ return corresponding record
- For each value return the highest version

```
Get get = new Get(Bytes.toBytes("row1"));
Result r = htable.get(get);
5.8.1.2. Default Get Example r.getValue(Bytes.toBytes("cf"), Bytes.toBytes("attr")); // returns current version of value
```

- Can control the number of versions you want

```
Get get = new Get(Bytes.toBytes("row1"));
get.setMaxVersions(3); // will return last 3 versions of row
Result r = htable.get(get);
byte[] b = r.getValue(Bytes.toBytes("cf"), Bytes.toBytes("attr")); // returns current version of value
List<KeyValue> kv = r.getColumn(Bytes.toBytes("cf"), Bytes.toBytes("attr")); // returns all versions of
```

# OPERATIONS ON REGIONS: **SCAN()**

```
HTable htable = ...      // instantiate HTable

Scan scan = new Scan();
scan.addColumn(Bytes.toBytes("cf"),Bytes.toBytes("attr"));
scan.setStartRow( Bytes.toBytes("row"));                      // start key is inclusive
scan.setStopRow( Bytes.toBytes("row" + (char)0)); // stop key is exclusive
ResultScanner rs = htable.getScanner(scan);
try {
    for (Result r = rs.next(); r != null; r = rs.next()) {
        // process result...
    } finally {
        rs.close(); // always close the ResultScanner!
    }
}
```

# GET()

Select value from table where  
key='com.apache.www' AND  
label='anchor:apache.com'

| Row key          | Time Stamp | Column "anchor:"    |                 |
|------------------|------------|---------------------|-----------------|
| "com.apache.www" | t12        |                     |                 |
|                  | t11        |                     |                 |
|                  | t10        | "anchor:apache.com" | <b>"APACHE"</b> |
|                  | t9         | "anchor:cnnsi.com"  | "CNN"           |
|                  | t8         | "anchor:my.look.ca" | "CNN.com"       |
|                  | t6         |                     |                 |
|                  | t5         |                     |                 |
|                  | t3         |                     |                 |

# SCAN()

Select value from table  
where  
anchor='cnnsi.com'

| Row key          | Time Stamp | Column "anchor:"    |           |
|------------------|------------|---------------------|-----------|
| "com.apache.www" | t12        |                     |           |
|                  | t11        |                     |           |
|                  | t10        | "anchor:apache.com" | "APACHE"  |
| "com.cnn.www"    | t9         | "anchor:cnnsi.com"  | "CNN"     |
|                  | t8         | "anchor:my.look.ca" | "CNN.com" |
|                  | t6         |                     |           |
|                  | t5         |                     |           |
|                  | t3         |                     |           |

# OPERATIONS ON REGIONS: PUT()

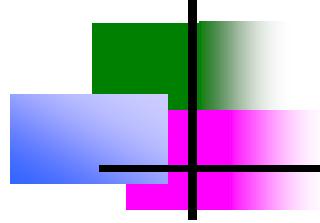
- Insert a new record (with a new key), Or
- Insert a record for an existing key

```
Put put = new Put(Bytes.toBytes(row));
put.add(Bytes.toBytes("cf"), Bytes.toBytes("attr1"), Bytes.toBytes( data));
htable.put(put);
```

Implicit version  
number  
(timestamp)

```
Put put = new Put( Bytes.toBytes(row));
long explicitTimeInMs = 555; // just an example
put.add(Bytes.toBytes("cf"), Bytes.toBytes("attr1"), explicitTimeInMs, Bytes.toBytes(data));
htable.put(put);
```

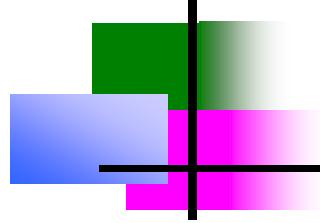
Explicit version  
number



# OPERATIONS ON REGIONS: **DELETE()**

- Marking table cells as deleted
- **Multiple levels**
  - Can mark an entire column family as deleted
  - Can make all column families of a given row as deleted

- All operations are logged by the RegionServers
- The log is flushed periodically



# HBASE: JOINS

---

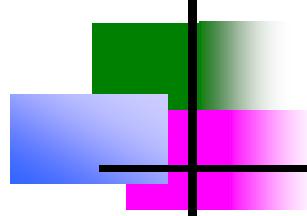
- HBase does not support joins
- Can be done in the application layer
  - Using scan() and get() operations

# ALTERING A TABLE

```
Configuration config = HBaseConfiguration.create();
HBaseAdmin admin = new HBaseAdmin(config);
String table = "myTable";
admin.disableTable(table);           ← Disable the table before changing the schema

HColumnDescriptor cf1 = ....;
admin.addColumn(table, cf1);        // adding new ColumnFamily
HColumnDescriptor cf2 = ....;
admin.modifyColumn(table, cf2);     // modifying existing ColumnFamily

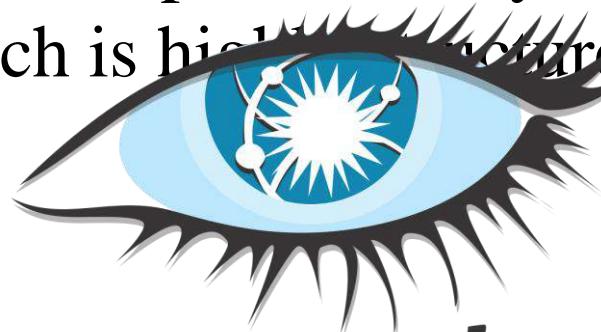
admin.enableTable(table);          6.1 Schema Creation
```



# CASSANDRA: DATA MODEL

---

A table in Cassandra is a distributed multi dimensional map indexed by a key. The value is an object which is highly structured.

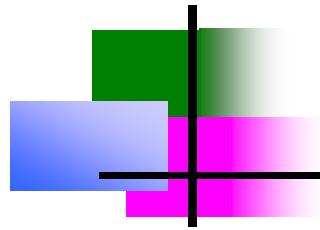


*cassandra*

**Cassandra exposes two kinds of columns families**

Simple column families

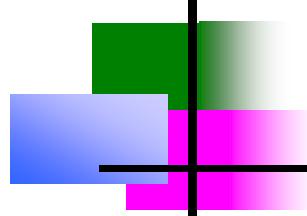
Super column families



# STRUCTURE

---

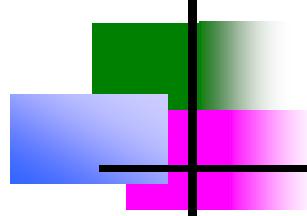
- keyspace
  - settings (eg, partitioner)
  - column family
    - settings (eg, comparator, type [Std])
  - column
  - name
- value
- clock



# COLUMN FAMILY

---

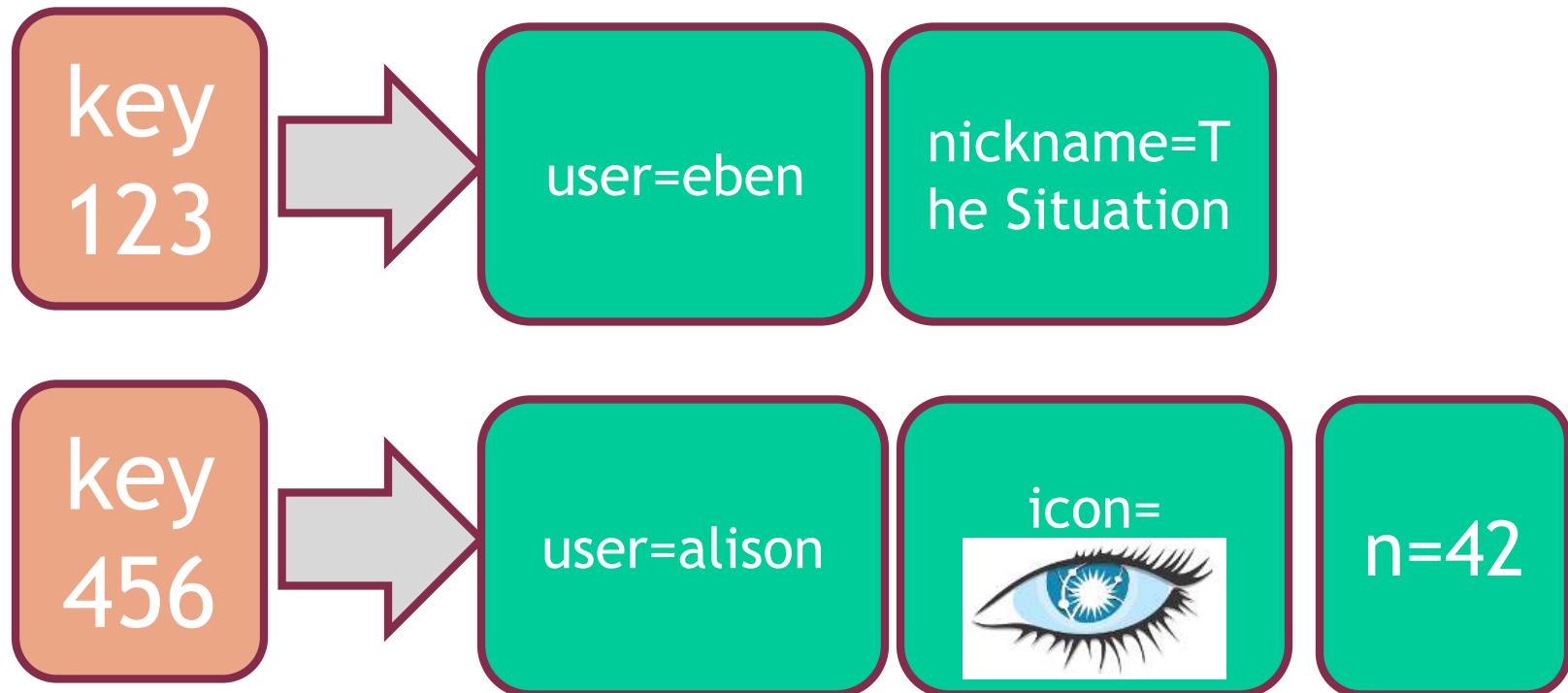
- group records of *similar* kind
- not *same* kind, because CFs are **sparse tables**
- ex:
  - User
  - Address
  - Tweet
  - PointOfInterest
  - HotelRoom

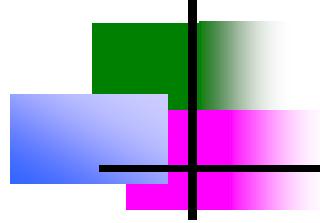


# CASSANDRA

---

- THINK OF CASSANDRA AS ROW-ORIENTED
- each row is uniquely identifiable by key
- rows group columns and super columns  
**COLUMN FAMILY**

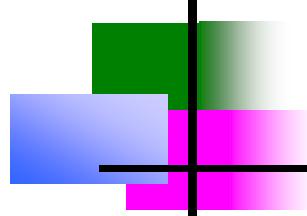




# JSON-LIKE NOTATION

---

User {  
  123 : { email: alison@foo.com,  
     },  
  456 : { email: eben@bar.com,  
    location: The Danger Zone }  
}



## 0.6 EXAMPLE

---

```
$cassandra -f
```

```
$bin/cassandra-cli
```

```
cassandra> connect localhost/9160
```

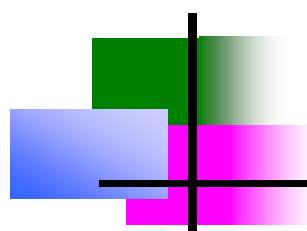
```
cassandra> set Keyspace1.Standard1['eben']['age']='29'
```

```
cassandra> set
```

```
    Keyspace1.Standard1['eben']['email']='e@e.com'
```

```
cassandra> get Keyspace1.Standard1['eben']['age']
```

```
=> (column=6e616d65, value=39,  
     timestamp=1282170655390000)
```



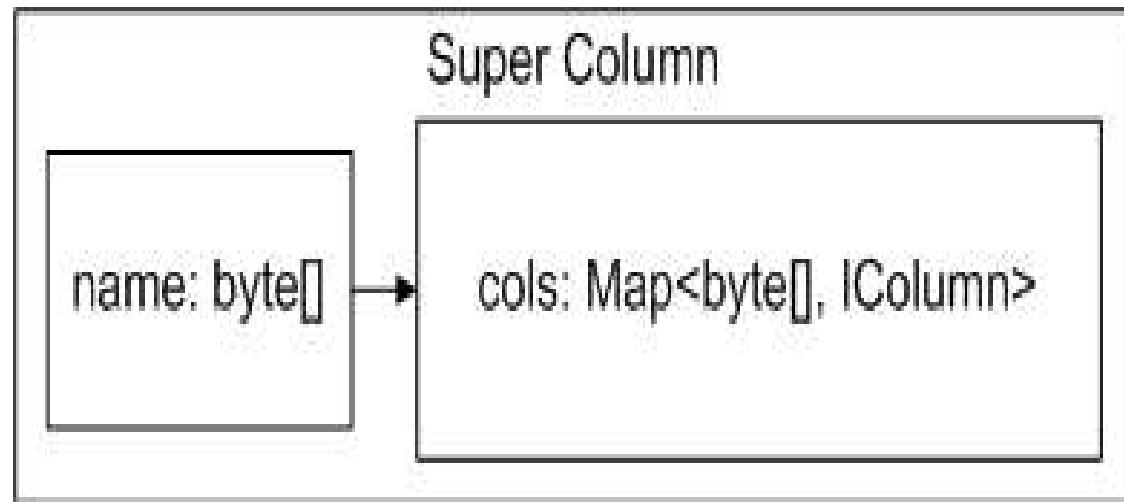
# A COLUMN HAS 3 PARTS

---

1. name
  - byte[]
  - determines sort order
  - used in queries
  - indexed
2. value
  - byte[]
  - *you don't query on column values*
3. timestamp
  - long (clock)
  - last write wins conflict resolution

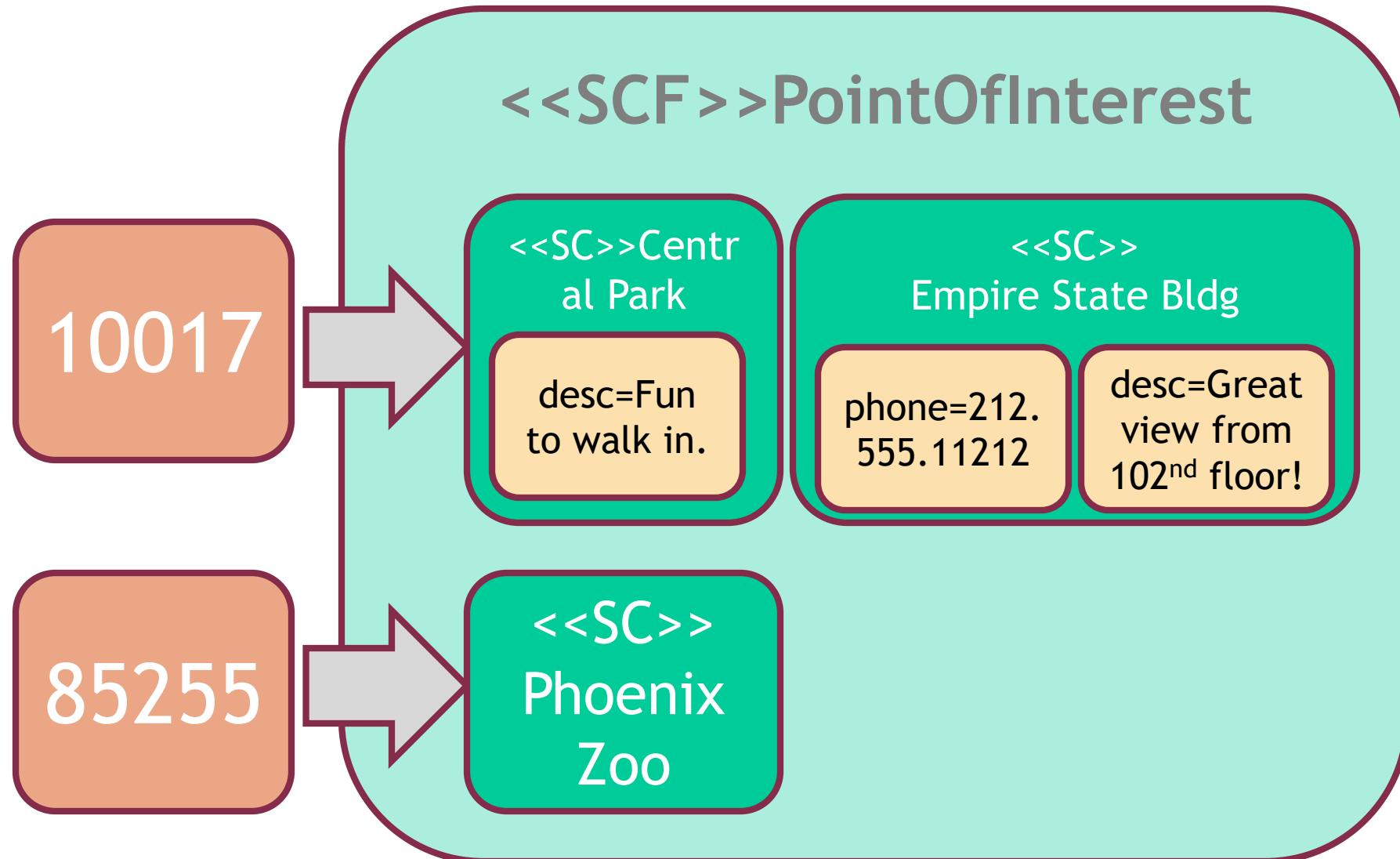
# SUPER COLUMN

- super columns group columns under a common name



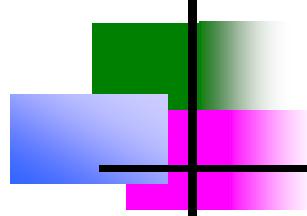
- sub-column names in a SCF are *not* indexed
  - top level columns (SCF Name) are *always* indexed
- often used for **denormalizing** data from standard CFs

# SUPER COLUMN FAMILY



# SUPER COLUMN FAMILY

```
super column family  
PointOfInterest :  
key: 85255 {  
    Phoenix Zoo { phone: 480-555-5555, desc: They have animals here.  
    },  
    Spring Training { phone: 623-333-3333, desc: Fun for baseball fans. },  
}, //end phx  
key  
super column  
key: 10019 {  
    Central Park { desc: Walk around. It's pretty. } ,  
    Empire State Building { phone: 212-777-7777,  
                           desc: Great view from 102nd floor. }  
} //end nyc  
}
```



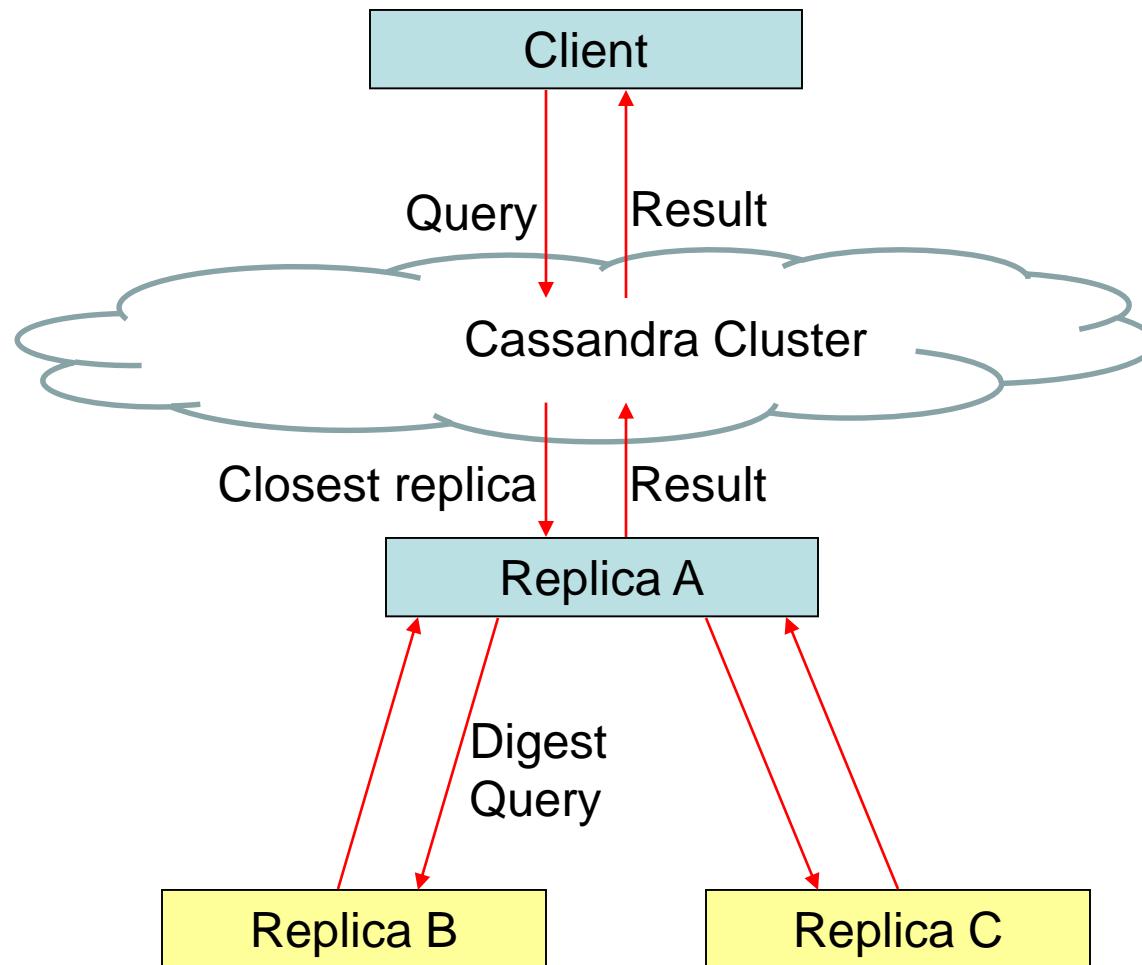
# API

---

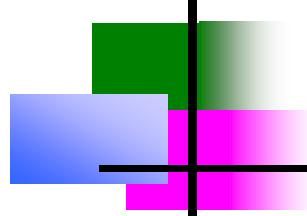
The Cassandra API consists of the following three simple methods.

- insert(table; key; rowMutation)
- get(table; key; columnName)
- delete(table; key; columnName)

# ARCHITECTURE



- Built from the start to solve the scaling problem
- Consistency, Availability, Partitioning
  - (can't have it all)
- Configurable to fit requirements
- *MongoDB is an open source, document-oriented database designed with both scalability and developer agility in mind.*
- *Instead of storing data in tables and rows like as relational database, in MongoDB store JSON-like documents with dynamic schemas(schema-free, schemaless).*

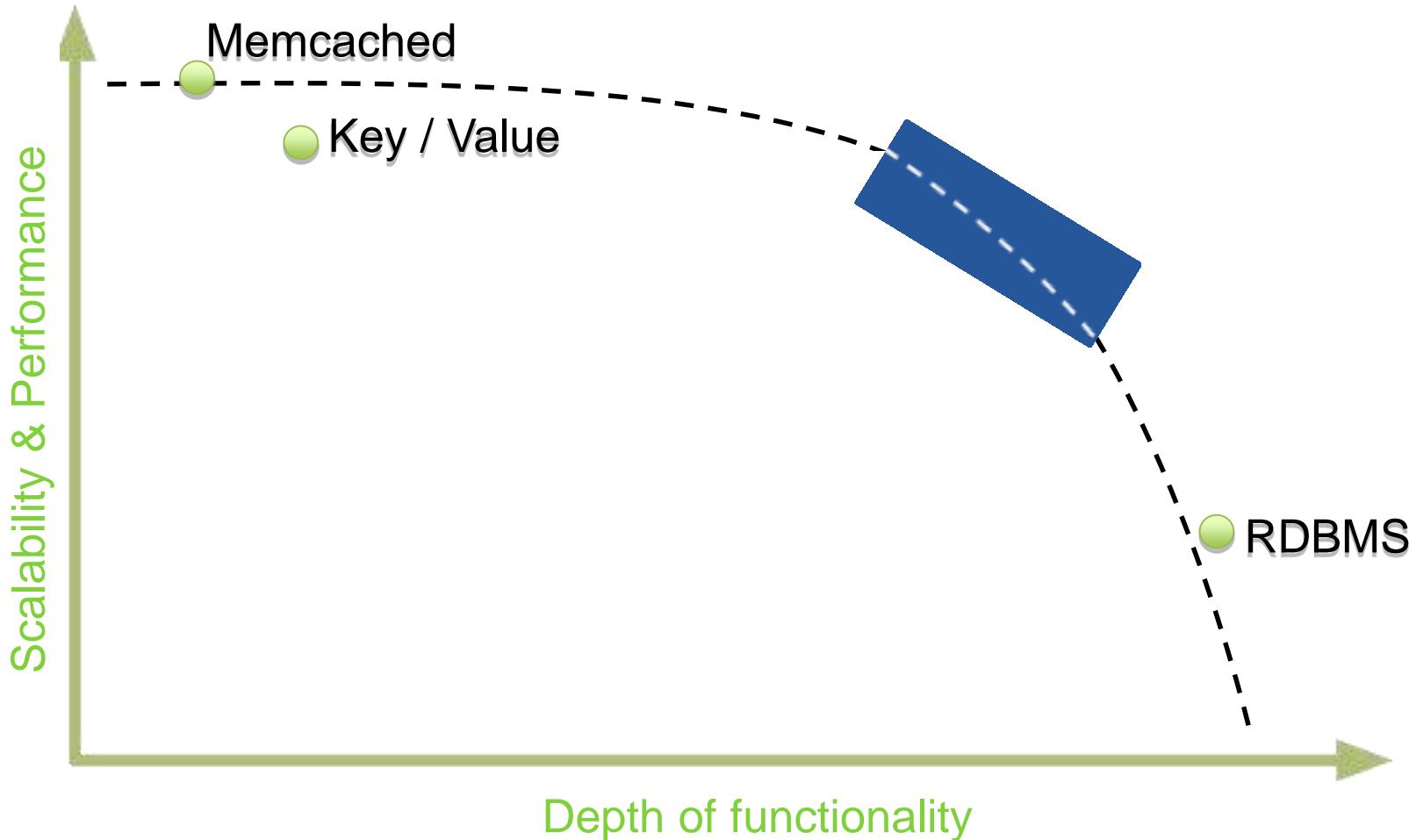


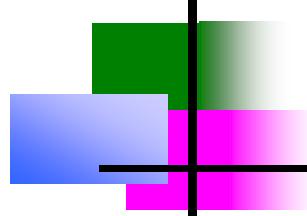
# MangoDB (Introduction)

---

- **Written in:** C++
- **Main point:** Retains some friendly properties of SQL
- **License:** AGPL (Drivers: Apache)
- Master/slave replication (auto failover with replica sets)
- Sharding built-in
- Queries are javascript expressions
- Run arbitrary javascript functions server-side
- Better update-in-place than CouchDB
- Uses memory mapped files for data storage
- An empty database takes up 192Mb
- GridFS to store big data + metadata (not actually an FS)

# AS SIMPLE AS POSSIBLE, BUT NO SIMPLER





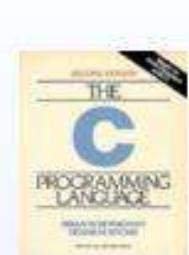
# Data Model

---

- **Data model:** Using BSON (binary JSON), developers can easily map to modern object-oriented languages without a complicated ORM layer.
- BSON is a binary format in which zero or more key/value pairs are stored as a single entity.
- lightweight, traversable, efficient.

```
{"hello": "world"}           → "\x16\x00\x00\x00\x02hello\x00\x06\x00\x00\x00world\x00\x00"  
  
{"BSON": ["awesome", 5.05, 1986]} → "1\x00\x00\x00\x04BSON\x00&\x00\x00\x00\x020\x00\x08\x00\x00\x00\x00awesome\x00\x011\x00333333\x14@\x102\x00\xc2\x07\x00\x00\x00\x00"
```

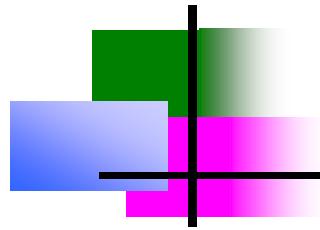
# Supported Languages



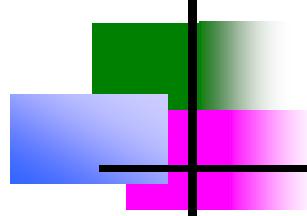
=GO



<http://www.mongodb.org/display/DOCS/Drivers>



# REPRESENTING & QUERYING DATA

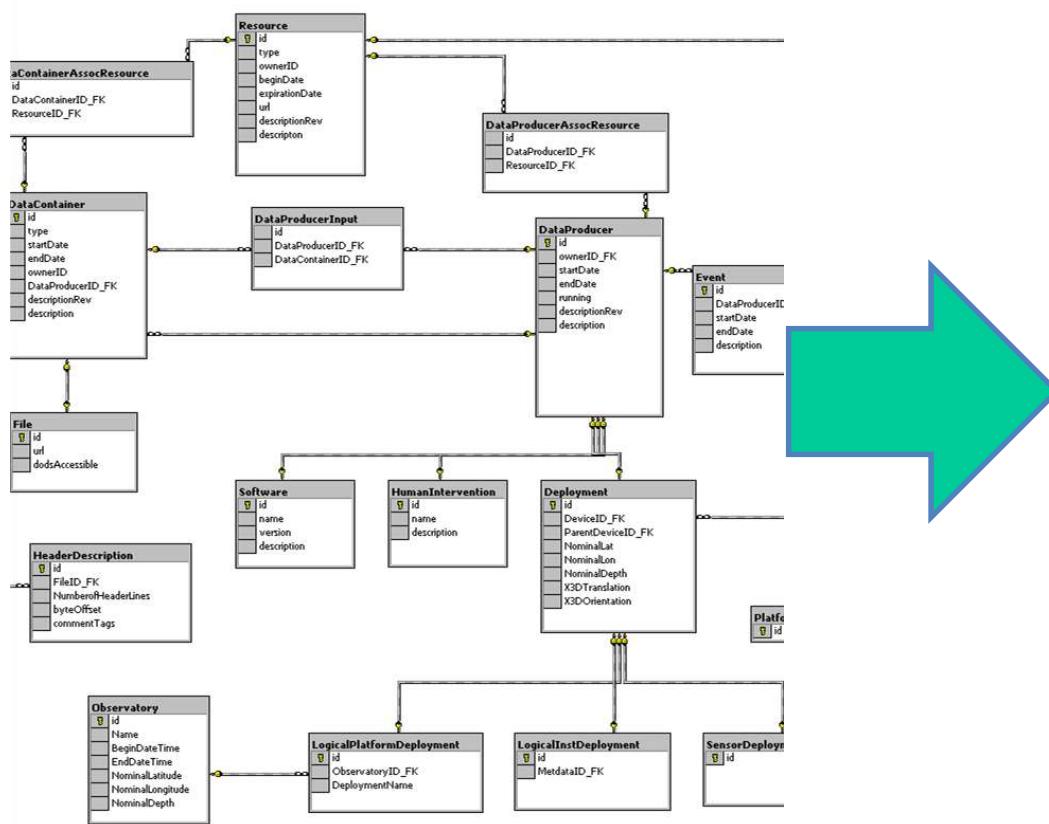


# TERMINOLOGY

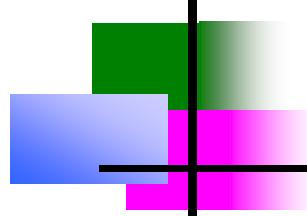
---

| RDBMS  | MongoDB             |
|--------|---------------------|
| Table  | Collection          |
| Row(s) | JSON Document       |
| Index  | Index               |
| Join   | Embedding & Linking |
|        |                     |
|        |                     |

# TABLES TO COLLECTIONS OF JSON DOCUMENTS



```
{  
  title: 'MongoDB',  
  contributors:  
  [  
    { name: 'Eliot Horowitz',  
      email: 'eliot@10gen.com' },  
    { name: 'Dwight Merriman',  
      email: 'dwight@10gen.com' }  
  ],  
  model:  
  {  
    relational: false,  
    awesome: true  
  }  
}
```



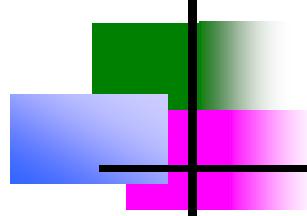
# DOCUMENTS

---

- Collections contain documents
- Documents can contain other documents  
and/or
- Documents can reference other documents

## Flexible Schema

- Flexible/powerful ability to relate data



# DOCUMENTS

---

```
var p = { author: "roger",
          date: new Date(),
          title: "Spirited Away",
          avgRating: 9.834,
          tags: ["Tezuka", "Manga"]}
```

```
> db.posts.save(p)
```



# LINKED VS EMBEDDED DOCUMENTS

The diagram illustrates the difference between EMBEDDED and LINKED document structures in MongoDB.

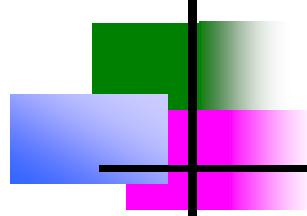
**EMBEDDED:**

```
{  
  "_id": "61829c46914595254881d99c",  
  "text": "This is the text of the post document",  
  "comments": [  
    {  
      "text": "This is the text of the first comment",  
      "author": "author"  
    },  
    {  
      "text": "This is the text of the second comment",  
      "author": "author2"  
    }  
  ]  
}
```

**LINKED:**

```
{  
  "_id": "61829c46914595254881d99c",  
  "text": "This is the text of the post document"  
}  
  
[  
  {  
    "_id": "61829c46914595254881d99c",  
    "text": "This is the text of the first comment",  
    "author": "author",  
    "postId": "61829c46914595254881d99c"  
  },  
  {  
    "_id": "61829c46914595254881d99c",  
    "text": "This is the text of the second comment",  
    "author": "author2",  
    "postId": "61829c46914595254881d99c"  
  }  
]
```

<https://medium.com/geekculture/mongodb-relationships-embedding-vs-linking-pro-and-cons-5f7583e655ab>



# QUERYING

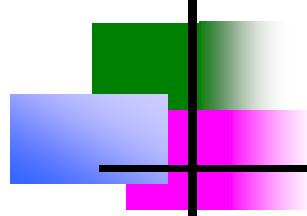
---

```
>db.posts.find()
```

```
{ _id : ObjectId("4c4ba5c0672c685e5e8aabf3"),  
  author : "roger",  
  date : "Sat Jul 24 2010 19:47:11 GMT-0700 (PDT)",  
  text : "Spirited Away",  
  tags : [ "Tezuka", "Manga" ] }
```

Note:

- `_id` is unique, but can be anything you'd like



# QUERY OPERATORS

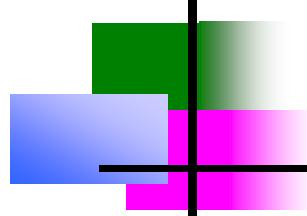
---

- Conditional Operators
  - \$all, \$exists, \$mod, \$ne, \$in, \$nin, \$nor, \$or, \$size, \$type
  - \$lt, \$lte, \$gt, \$gte

```
// find posts with any tags  
> db.posts.find( {tags: {$exists: true}} )
```

```
// find posts matching a regular expression  
> db.posts.find( {author: /^rog*/i} )
```

```
// count posts by author  
> db.posts.find( {author: 'roger'} ).count()
```



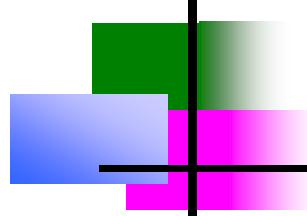
# ATOMIC OPERATIONS

---

- \$set, \$unset, \$inc, \$push, \$pushAll, \$pull, \$pullAll, \$bit

```
> comment = { author: "fred",
    date: new Date(),
    text: "Best Movie Ever"}
```

```
> db.posts.update( { _id: "..." },
    $push: { comments: comment } );
```



# INDEXES

---

Create index on any Field in Document

```
>db.posts.ensureIndex({author: 1})
```

// Index nested documents

```
> db.posts.ensureIndex( "comments.author":1 )  
> db.posts.find({ 'comments.author':'Fred' })
```

// Index on tags (array values)

```
> db.posts.ensureIndex( tags: 1)  
> db.posts.find( { tags: 'Manga' } )
```

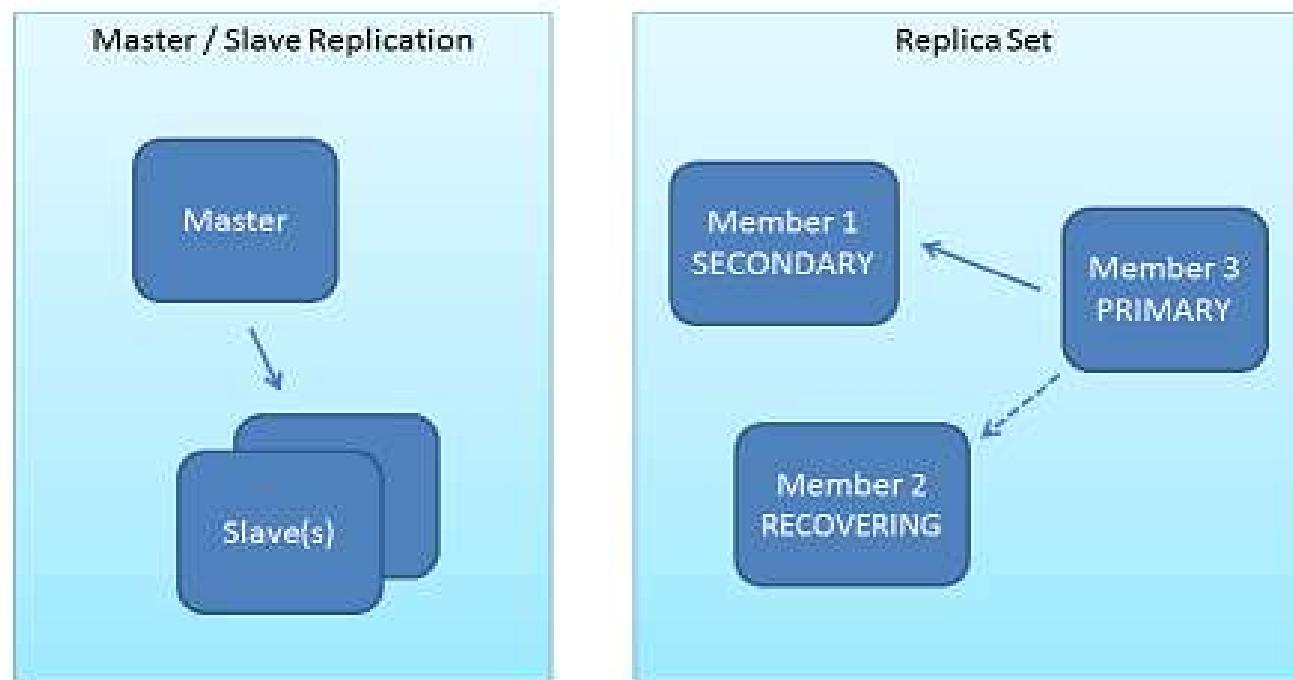
// geospatial index

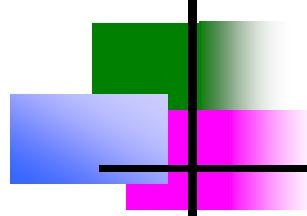
```
> db.posts.ensureIndex( { "author.location": "2d" } )  
> db.posts.find( "author.location" : { $near : [22,42] } )
```

# Architecture (MongoDB)

## 1. Replication

- Replica Sets and Master-Slave
- replica sets are a functional superset of master/slave.

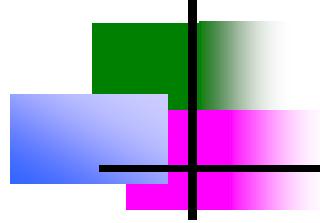




# Architecture (Write process)

---

- *All write operation go through primary, which applies the write operation.*
- *write operation than records the operations on primary's operation log “oplog”*
- *Secondary are continuously replicating the oplog and applying the operations to themselves in a asynchronous process.*



# Why replica sets

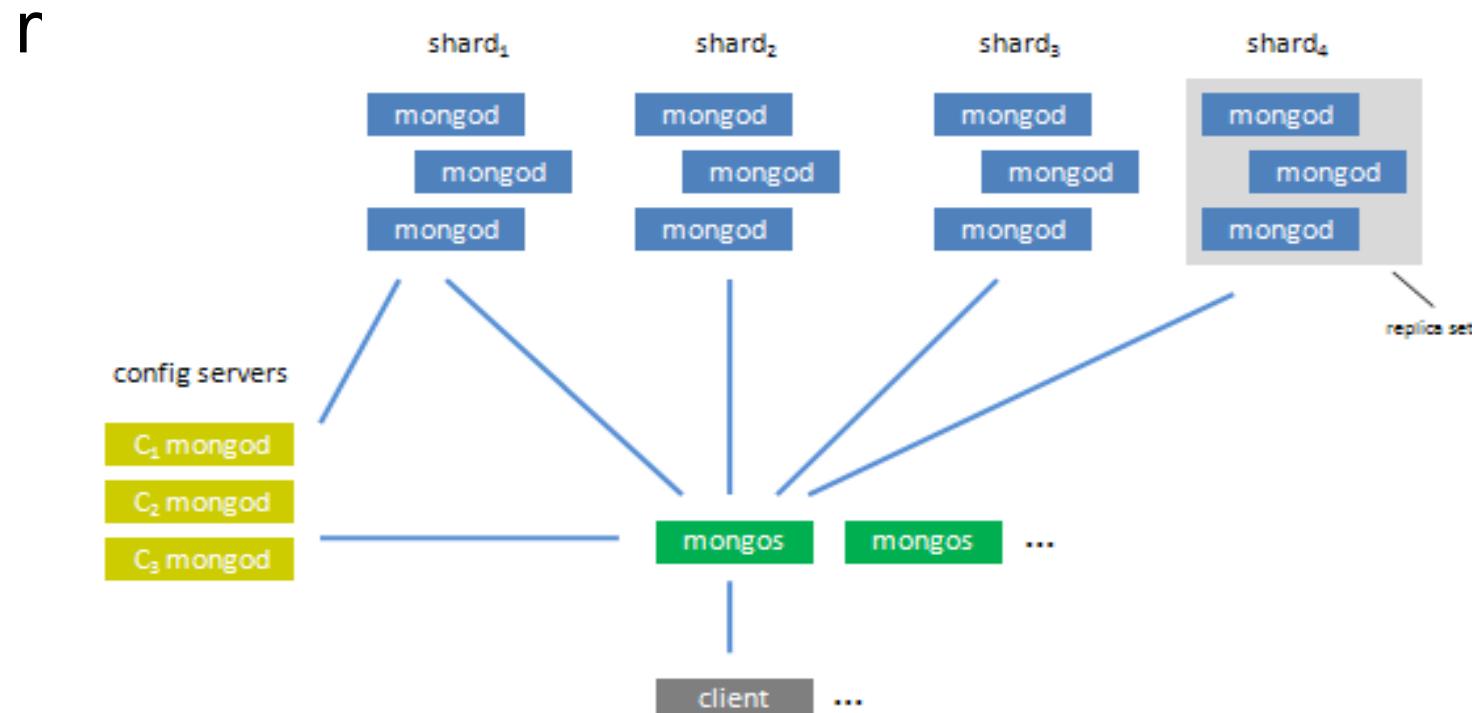
---

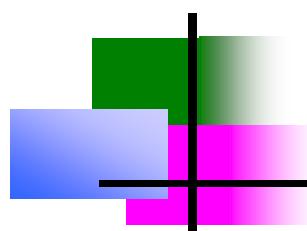
- Data Redundancy
- Automated Failover
- Read Scaling
- Maintenance
- Disaster Recovery(delayed secondary)

# Architecture

## 2. Sharding

Sharding is the partitioning of data among multiple machines in an order-preserving manner.

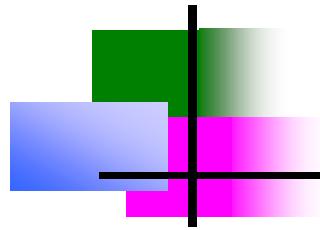




# AGGREGATION/BATCH DATA PROCESSING

---

- Map/Reduce can be used for batch data processing
  - Currently being used for totaling, averaging, etc
  - Map/Reduce is a big hammer
- Simple aggregate functions available
- Aggregation Framework: Simple, Fast
  - No Javascript Needed, runs natively on server
  - Filter or Select Only Matching Sub-documents or Arrays via new operators
- MongoDB Hadoop Connector
  - Useful for Hadoop Integration
  - Massive Batch Processing Jobs



# Chapter 6: Machine Learning with Big data

Basanta Joshi, PhD

Asst. Prof., Depart of Electronics and Computer Engineering

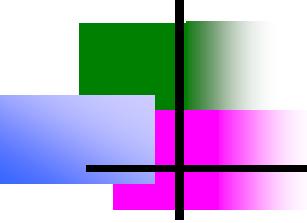
Program Coordinator, MSc in Information and Communication Engineering

Member, Laboratory for ICT Research and Development (LICT)

Institute of Engineering

basanta@ioe.edu.np

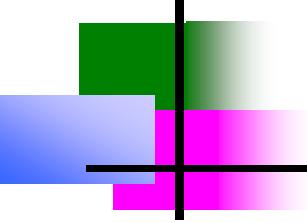
<http://www.basantajoshi.com.np>



# What is Learning?

---

- Learning is one of those everyday terms which is broadly and vaguely used in the English language
  - Learning is making useful changes in our minds
  - Learning is constructing or modifying representations of what is being experienced
  - Learning is the phenomenon of knowledge acquisition in the absence of explicit programming
- **Herbert Simon, 1983**
  - Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively next time.



# Implications

---

Learning involves 3 factors:

changes

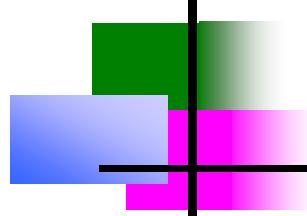
generalization

improvement

Learning changes the learner: for machine learning the problem is determining the nature of these changes and how to best represent them

Learning leads to generalization: performance must improve not only on the same task but on similar tasks

Learning leads to improvements: machine learning must address the possibility that changes may degrade performance and find ways to prevent it.



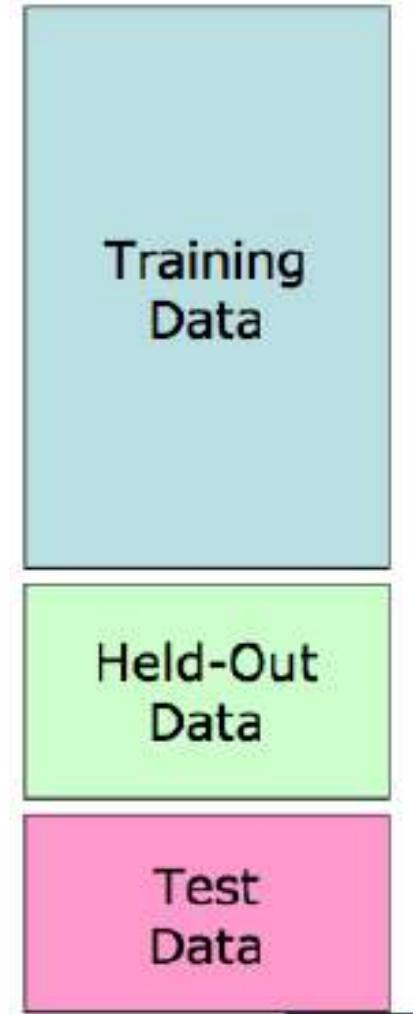
# Areas of Influence for Machine Learning

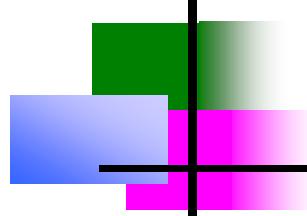
---

- *Statistics*: How best to use samples drawn from unknown probability distributions to help decide from which distribution some new sample is drawn?
- *Brain Models*: Non-linear elements with weighted inputs (Artificial Neural Networks) have been suggested as simple models of biological neurons.
- *Adaptive Control Theory*: How to deal with controlling a process having unknown parameters that must be estimated during operation?

# Important Concepts

- Data: labeled instances, e.g. emails marked spam/ham
  - Training set
  - Held out set
  - Test set
- Features: attribute-value pairs which characterize each  $x$
- Experimentation cycle
  - Learn parameters (e.g. model probabilities) on training set
  - (Tune hyperparameters on held-out set)
  - Compute accuracy of test set
  - Very important: never "peek" at the test set!
- Evaluation
  - Accuracy : fraction of instances predicted correctly
- Overfitting and generalization
  - Want a classifier which does well on *test* data
  - Overfitting: fitting the training data very closely, but not generalizing well

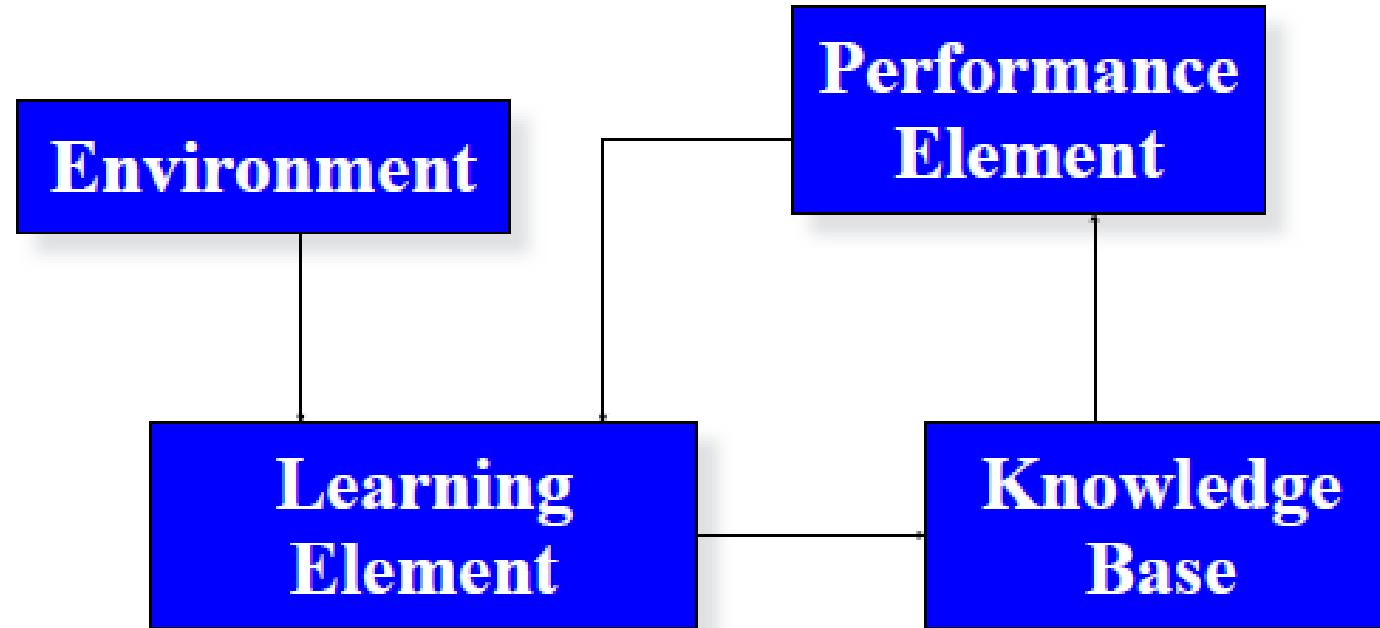


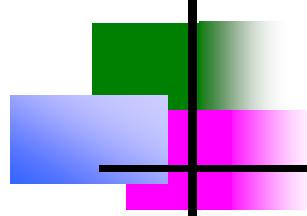


# Learning Framework

---

- There are four major components in a learning system:

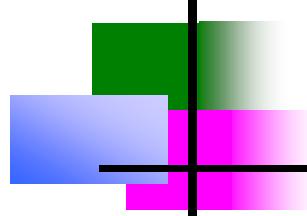




# The Environment

---

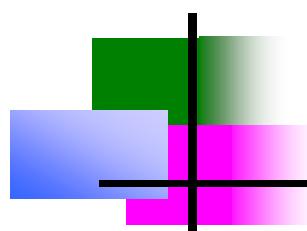
- The environment refers the nature and quality of information given to the learning element
- The nature of information depends on its level (the degree of generality wrt the performance element)
  - high level information is abstract, it deals with a broad class of problems
  - low level information is detailed, it deals with a single problem.
- The quality of information involves
  - noise free
  - reliable
  - ordered



# Learning Elements

---

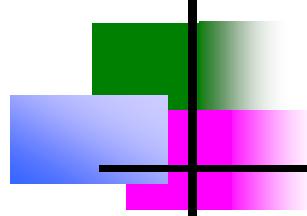
- **Four Learning situations**
- Rote Learning
  - environment provides information at the required level
- Learning by being told
  - information is too abstract, the learning element must hypothesize missing data
- Learning by example
  - information is too specific, the learning element must hypothesize more general rules
- Learning by analogy
  - information provided is relevant only to an analogous task, the learning element must discover the analogy



# The Knowledge Base

---

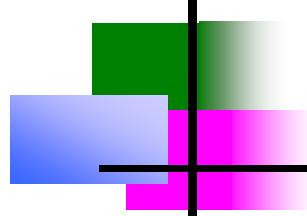
- Expressive
  - the representation contains the relevant knowledge in an easy to get to fashion
- Modifiable
  - it must be easy to change the data in the knowledge base
- Extendibility
  - the knowledge base must contain meta-knowledge (knowledge on how the data base is structured) so the system can change its structure



# The Performance Element

---

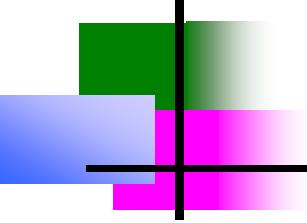
- Complexity
  - for learning, the simplest task is classification based on a single rule while the most complex task requires the application of multiple rules in sequence
- Feedback
  - the performance element must send information to the learning system to be used to evaluate the overall performance
- Transparency
  - the learning element should have access to all the internal actions of the performance element



# What is Machine Learning?

---

- Humans and animals: learning from experience.
- Machine learning: computational algorithms which enable machine to “**learn**” from experience (a dataset) to perform tasks without following **static program instructions**.
- Performance increases as number of samples increases.
- Machines performance better than humans in certain tasks e.g. cancer detection, character recognition



# Machine Learning and Artificial Intelligence

---

“Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data – such algorithms overcome following strictly static program instructions by making data-driven predictions or decisions, through building a model from sample inputs.”

by Arthur Samuel (pioneer in machine learning at IBM in 1959)

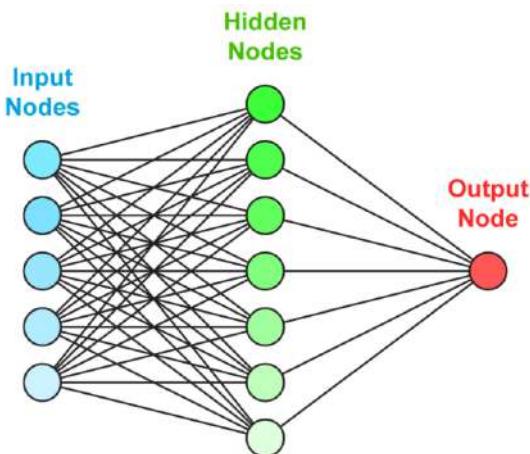
Reference: [https://en.wikipedia.org/wiki/Machine\\_learning](https://en.wikipedia.org/wiki/Machine_learning)



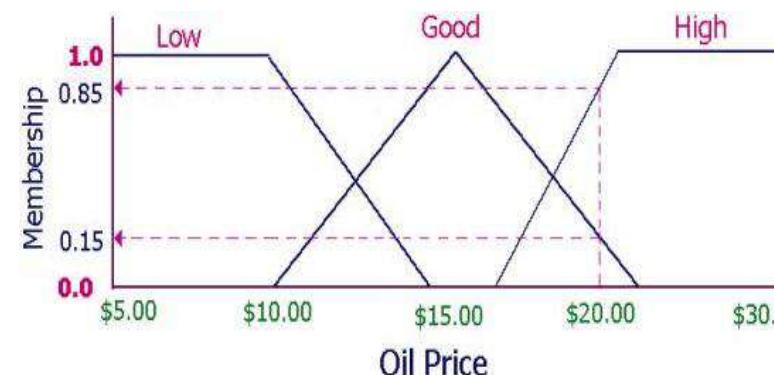
# Artificial Intelligence (A.I.)

Intelligence exhibited by machines

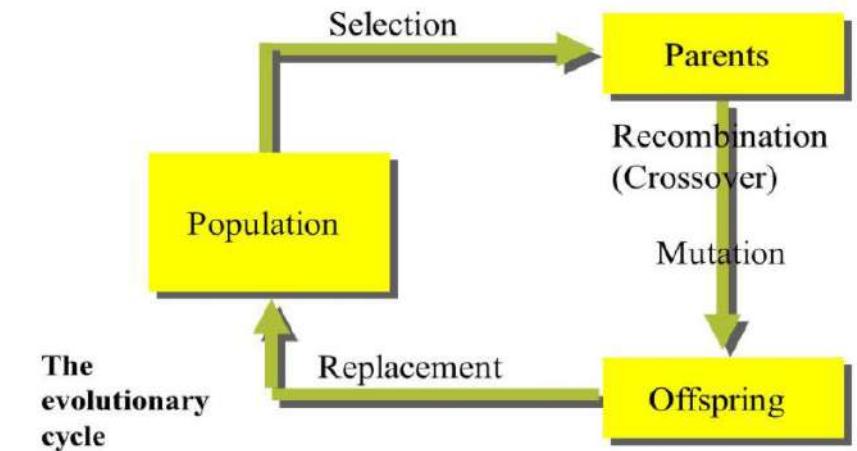
Machine Learning  
(e.g. neural networks)



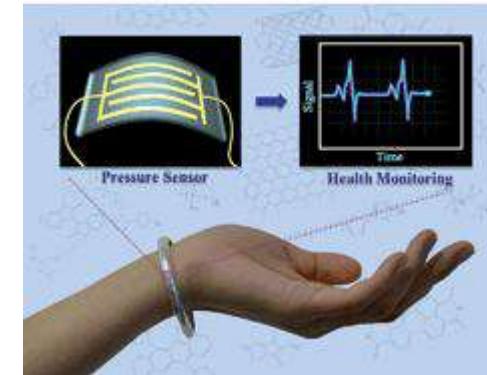
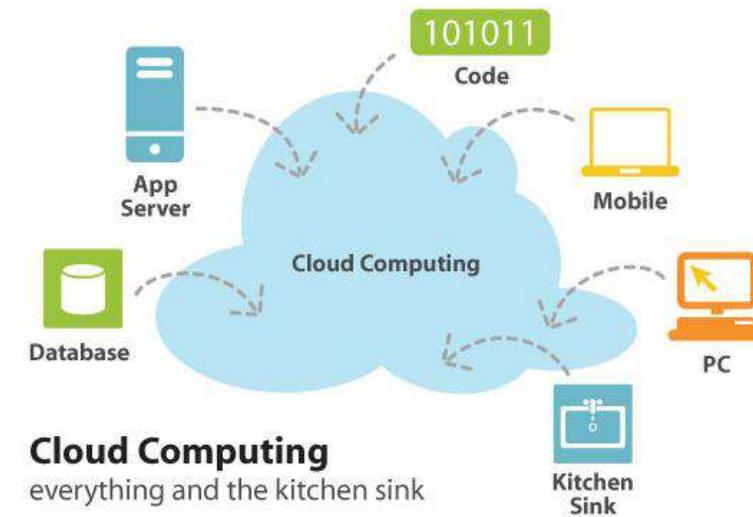
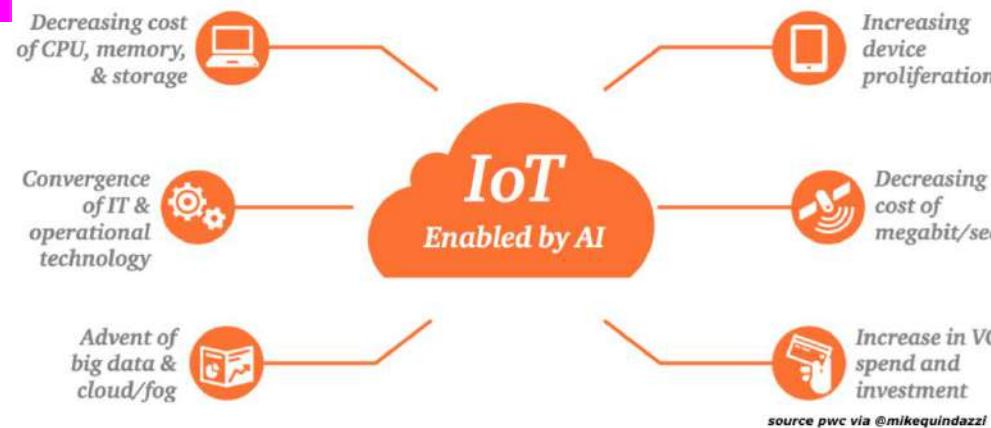
Fuzzy Logic  
(modelling human vague concepts)



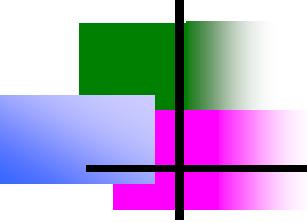
Evolutionary optimisation algorithms  
(near-optimal solutions)



# Applications of A.I.



Images source: Google

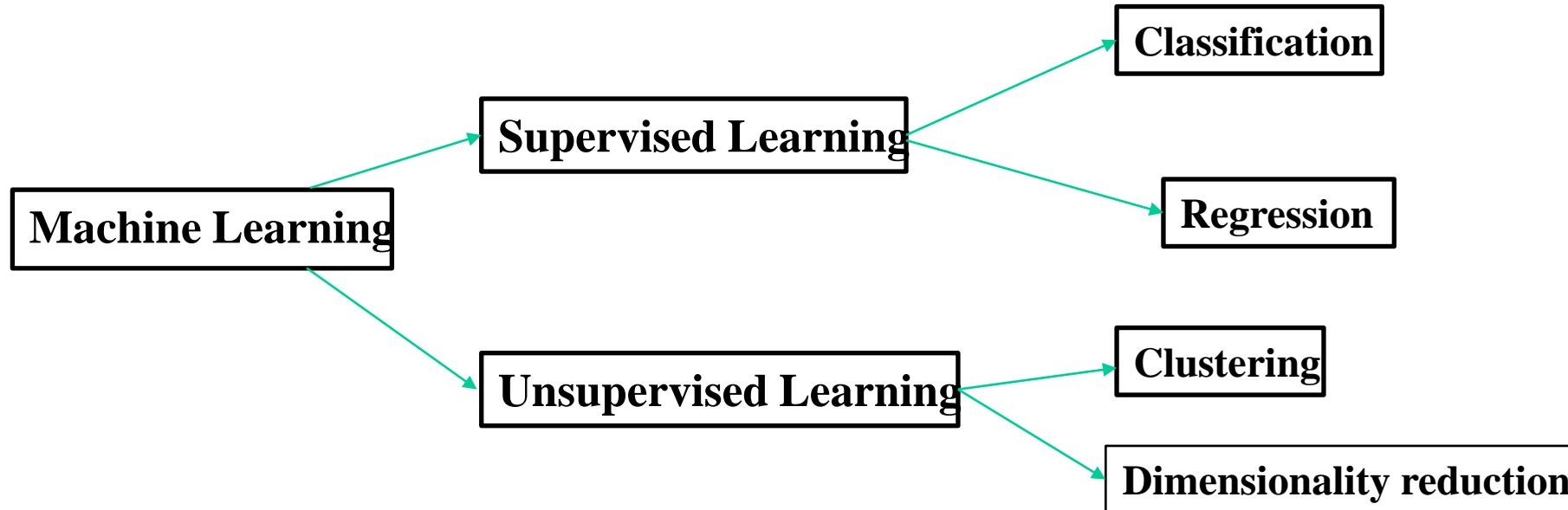


# When to use Machine Learning?

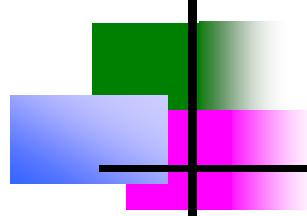
---

- Complex tasks involving **a large amount of data**, but **no formula (static rules)** of performing the **tasks** can be defined.
- Applications:
  - Sales forecasting
  - Face recognition (identification of users)
  - Speech recognition (customer services)
  - Stock market forecasting
  - Medical diagnosis e.g. cancer detection
  - Energy usage forecasting e.g. British Gas
  - Automatic recommendations on web

# Types of Machine Learning Problems



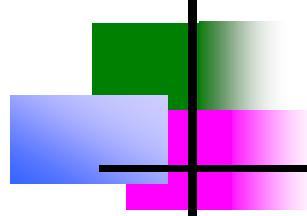
- **Supervised Learning:** develop predictive models from labelled data (i.e. data with classes or targets)
- **Unsupervised learning:** describe hidden structure of unlabelled data
  - Clustering: Group similar data into categories (clusters) based only on input data
  - Dimensionality reduction: Reduce input variables of a dataset to a smaller set of variables (structure of dataset)



# Supervised Learning

---

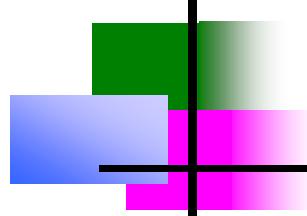
- Aim: build a model (e.g. neural network) to makes predictions by optimizing the parameters of the model
- The process of supervised learning:
  - **Phase 1 (Training):** Train a model using a training set (optimizing the parameters of the model)
  - **Phase 2 (Testing):** Evaluate the performance of the mode using a test set
- The training set: a dataset with known output (classes or targets).
- The test set: a dataset with known output and **no common data with the training set**
- Supervised learning problems:
  1. Classification
  2. Regression



# Classification

---

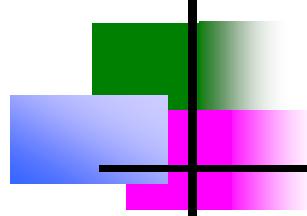
- Aim: Build a model to predict the classes or categories of input data.
- Examples of classification problems:
  - Customer credit rating: 3 classes (high, medium, low)
  - Energy consumption rating of a household
  - Email spam detection: 2 classes (spam and non-spam)
  - Tumour detection: 2 classes (has tumour and has no tumour)
  - Detection of different types of aeroplanes
  - Websites categorization and recommendation



# Regression

---

- Aim: Build a model to predict a continuous real value (any value in a range)
- Examples of regression problems:
  - Sales of products (millions of pounds)
  - House price (thousands of pounds)
  - Weather forecast (temperature degree, wind speed and direction)
  - Electric load forecasting (MegaWatts)



# Machine Learning algorithms

---

- Supervised Learning Algorithms:
  - **Neural networks**
  - Logistic regression
  - Support Vector Machines
  - Decision tree
  - Naïve Bayes
  - Bayesian networks
  - Nearest neighbours
  - ...
- Unsupervised Learning Algorithms:
  - Principal Components Analysis (PCA)
  - K-means clustering
  - Self-organizing map (SOM)
  - ...

# E6893 Big Data Analytics Lecture 4:

## *Big Data Analytics – Clustering and Classification*

Ching-Yung Lin, Ph.D.

Adjunct Professor, Dept. of Electrical Engineering and Computer Science

IBM Distinguished Researcher and Chief Scientist, Graph Computing



September 29th, 2016

# Review — Key Components of Mahout



## Collaborative Filtering

- User-Based Collaborative Filtering - [single machine](#)
- Item-Based Collaborative Filtering - [single machine / MapReduce](#)
- Matrix Factorization with Alternating Least Squares - [single machine / MapReduce](#)
- Matrix Factorization with Alternating Least Squares on Implicit Feedback- [single machine / MapReduce](#)
- Weighted Matrix Factorization, SVD++, Parallel SGD - [single machine](#)

## Classification

- Logistic Regression - trained via SGD - [single machine](#)
- Naive Bayes/ Complementary Naive Bayes - [MapReduce](#)
- Random Forest - [MapReduce](#)
- Hidden Markov Models - [single machine](#)
- Multilayer Perceptron - [single machine](#)

## Clustering

- Canopy Clustering - [single machine / MapReduce](#) (deprecated, will be removed once Streaming k-Means is stable enough)
- k-Means Clustering - [single machine / MapReduce](#)
- Fuzzy k-Means - [single machine / MapReduce](#)
- Streaming k-Means - [single machine / MapReduce](#)
- Spectral Clustering - [MapReduce](#)

# Machine Learning example: using SVM to recognize a Toyota Camry

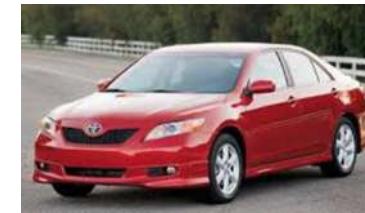
## Non-ML

- Rule 1. Symbol has something like bull's head
- Rule 2. Big black portion in front of car.
- Rule 3. ....????

## ML – Support Vector Machine

### *Feature Space*

### Positive SVs



### Negative SVs

# Machine Learning example: using SVM to recognize a Toyota Camry

## ML – Support Vector Machine



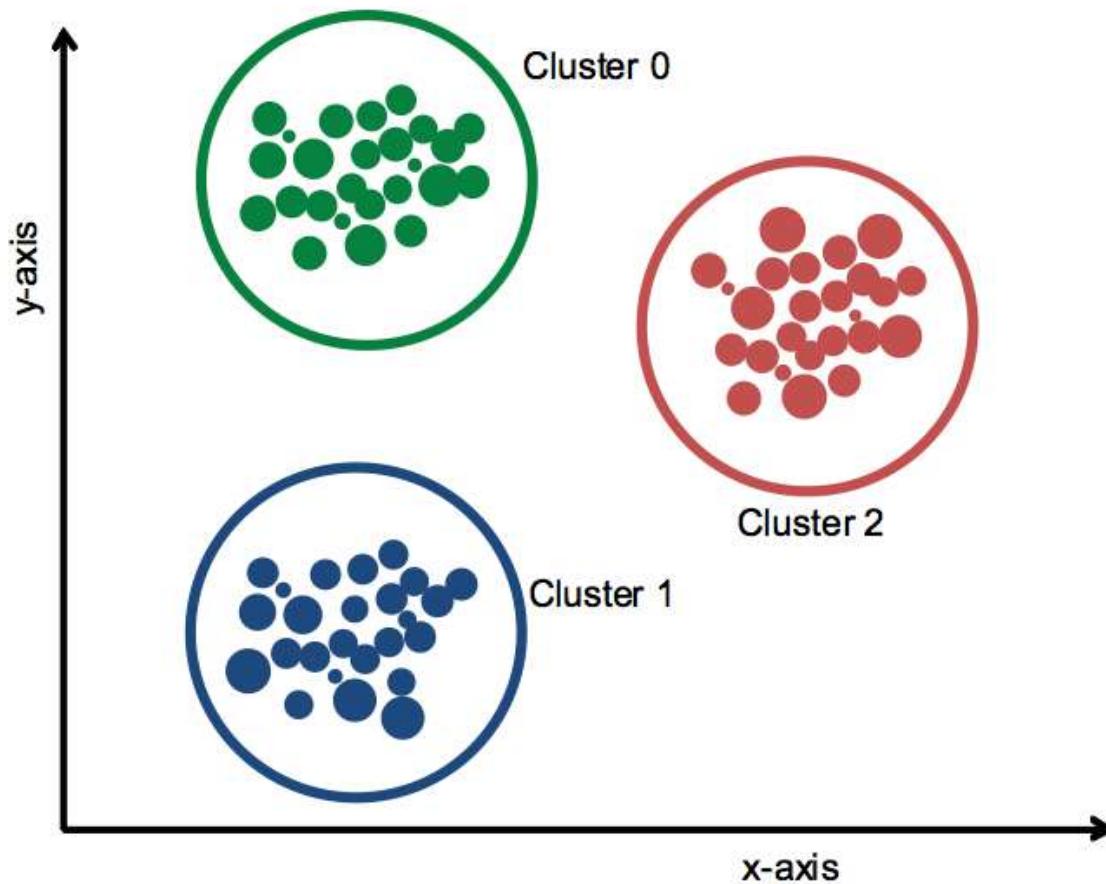
# Clustering

---

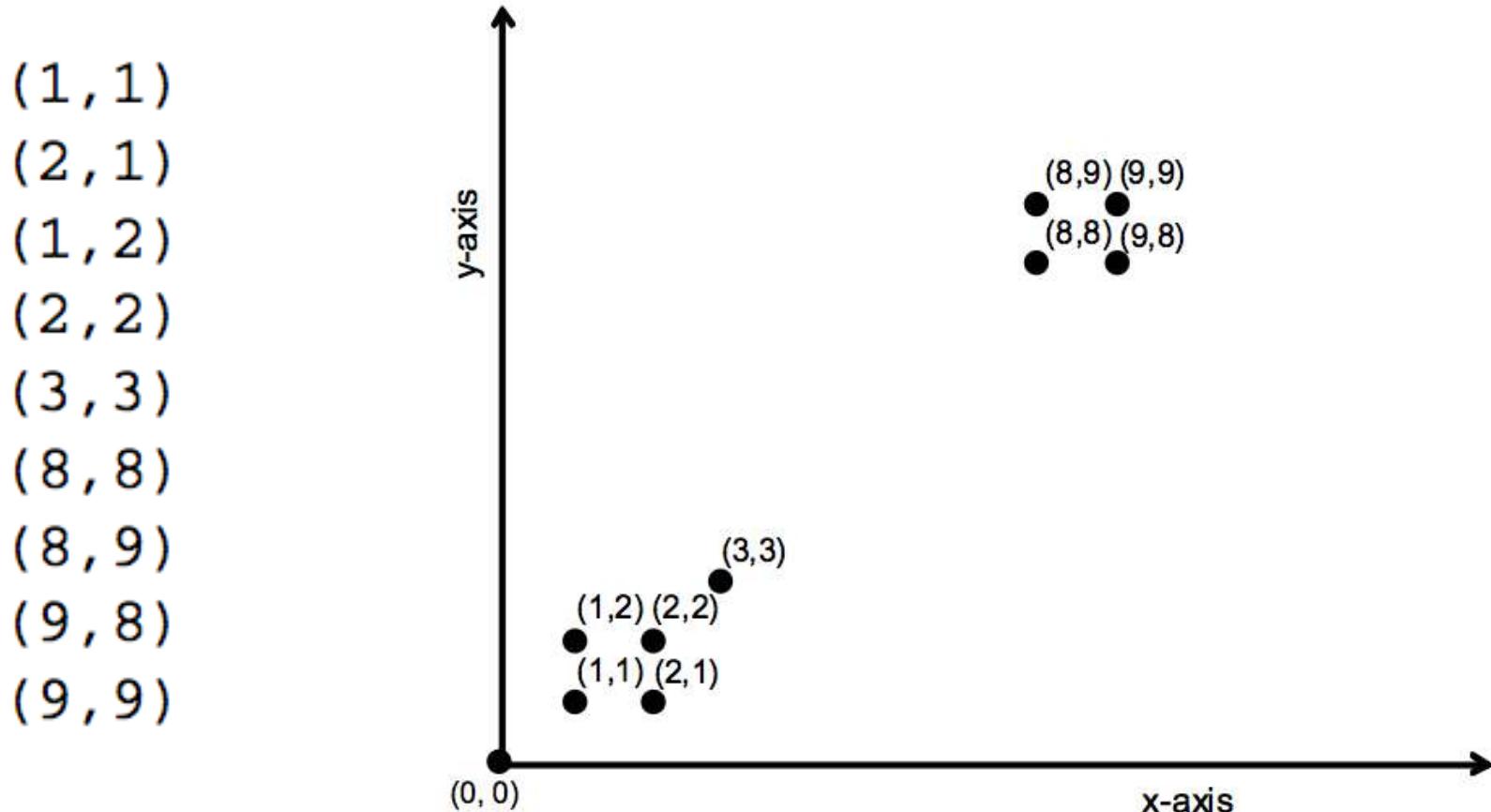
Clustering a collection involves three things:

- *An algorithm*—This is the method used to group the books together.
- *A notion of both similarity and dissimilarity*—In the previous discussion, we relied on your assessment of which books belonged in an existing stack and which should start a new one.
- *A stopping condition*—In the library example, this might be the point beyond which books can't be stacked anymore, or when the stacks are already quite dissimilar.

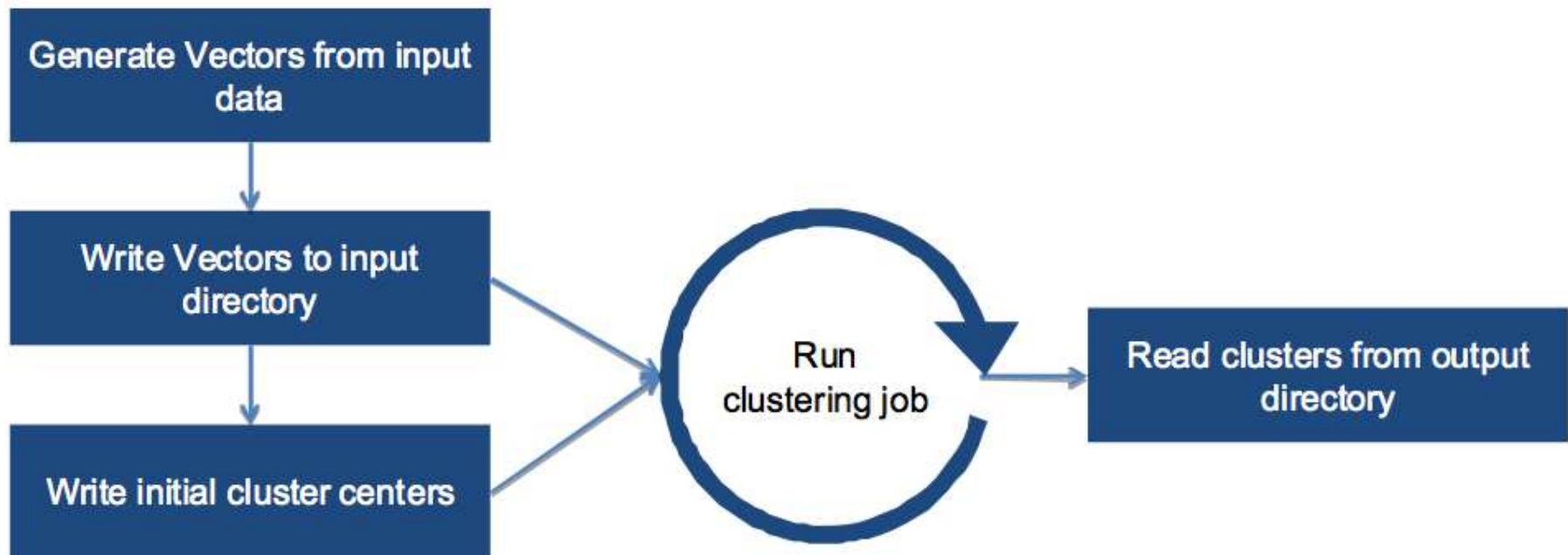
# Clustering – on feature plane



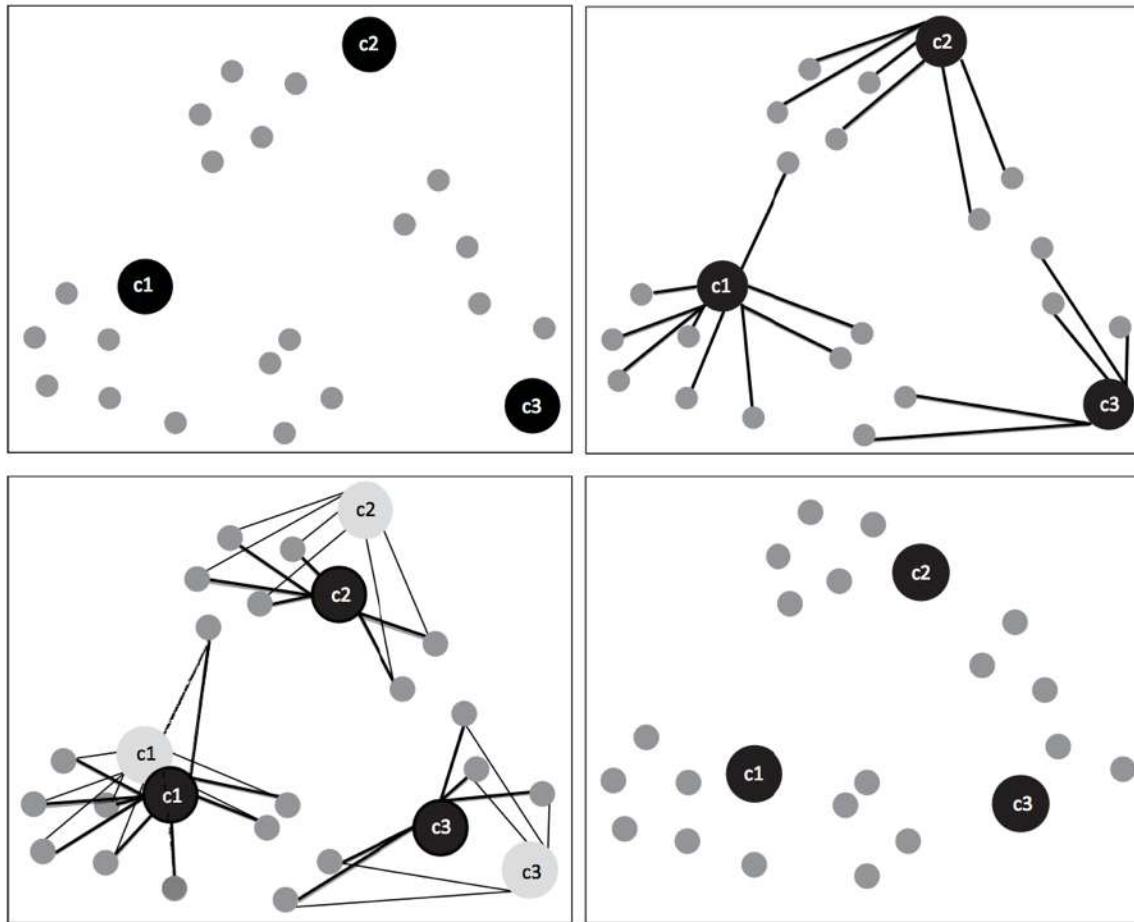
# Clustering example



# Steps on clustering

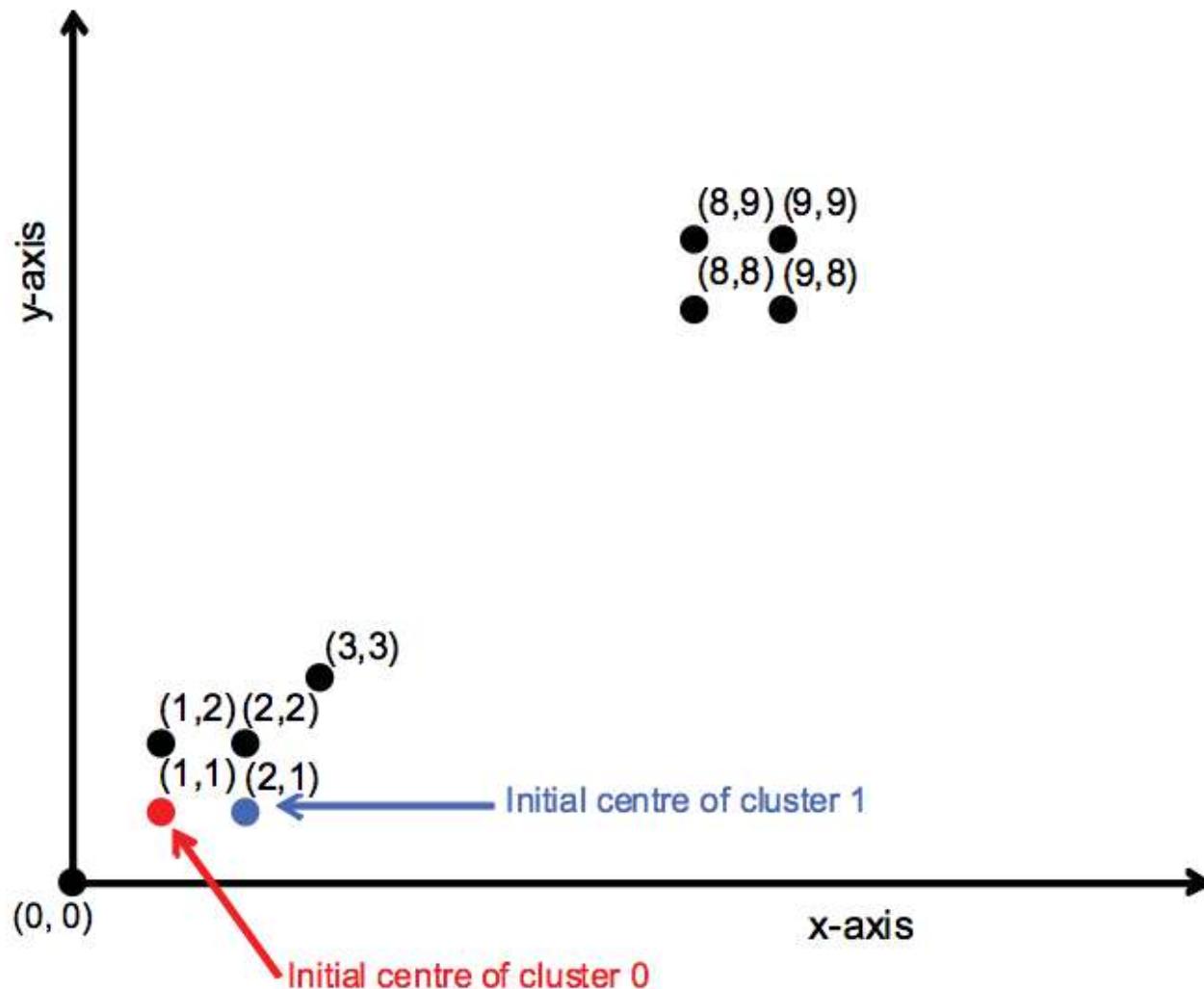


# K-mean clustering



**K-means clustering in action.** Starting with three random points as centroids (top left), the map stage (top right) assigns each point to the cluster nearest to it. In the reduce stage (bottom left), the associated points are averaged out to produce the new location of the centroid, leaving you with the final configuration (bottom right). After each iteration, the final configuration is fed back into the same loop until the centroids come to rest at their final positions.

# Making initial cluster centers



# Parameters to Mahout k-mean clustering algorithm

- The SequenceFile containing the input vectors.
- The SequenceFile containing the initial cluster centers. In this case, we seed two clusters, so there are two centers.
- The similarity measure to be used. We use EuclideanDistanceMeasure as the measure of similarity here, and we explore other similarity measures later in this chapter.
- The convergenceThreshold. If in a particular iteration the centers of the clusters don't change beyond this threshold, no further iterations are done.
- The number of iterations to be done.
- The Vector implementation used in the input files.

# HelloWorld clustering scenario

---

```

public static final double[][] points = { {1, 1}, {2, 1}, {1, 2},
                                         {2, 2}, {3, 3}, {8, 8},
                                         {9, 8}, {8, 9}, {9, 9}};

public static void writePointsToFile(List<Vector> points,
                                      String fileName,
                                      FileSystem fs,
                                      Configuration conf) throws IOException {
  Path path = new Path(fileName);
  SequenceFile.Writer writer = new SequenceFile.Writer(fs, conf,
    path, LongWritable.class, VectorWritable.class);
  long recNum = 0;
  VectorWritable vec = new VectorWritable();
  for (Vector point : points) {
    vec.set(point);
    writer.append(new LongWritable(recNum++), vec);
  }
  writer.close();
}

public static List<Vector> getPoints(double[][] raw) {
  List<Vector> points = new ArrayList<Vector>();
  for (int i = 0; i < raw.length; i++) {
    double[] fr = raw[i];
    Vector vec = new RandomAccessSparseVector(fr.length);
    vec.assign(fr);
    points.add(vec);
  }
  return points;
}

```

# HelloWorld Clustering scenario - II

```

public static void main(String args[]) throws Exception {
    int k = 2;

    List<Vector> vectors = getPoints(points);
    File testData = new File("testdata");
    if (!testData.exists()) {
        testData.mkdir();
    }
    testData = new File("testdata/points");
    if (!testData.exists()) {
        testData.mkdir();
    }

    Configuration conf = new Configuration();
    FileSystem fs = FileSystem.get(conf);
    writePointsToFile(vectors,
        "testdata/points/file1", fs, conf);           ← Write initial centers

    Path path = new Path("testdata/clusters/part-00000");
    SequenceFile.Writer writer
        = new SequenceFile.Writer(
            fs, conf,      path, Text.class, Cluster.class);

    for (int i = 0; i < k; i++) {
        Vector vec = vectors.get(i);
        Cluster cluster = new Cluster(
            vec, i, new EuclideanDistanceMeasure());
        writer.append(new Text(cluster.getIdentifer()), cluster);
    }
    writer.close();
}
  
```

Specify number of clusters to be formed

Create input directories for data

# HelloWorld Clustering scenario - III

```

KMeansDriver.run(conf, new Path("testdata/points"),
  new Path("testdata/clusters"),
  new Path("output"), new EuclideanDistanceMeasure(),
  0.001, 10, true, false);

SequenceFile.Reader reader
  = new SequenceFile.Reader(fs,
    new Path("output/" + Cluster.CLUSTERED_POINTS_DIR
      + "/part-m-00000"), conf);

IntWritable key = new IntWritable();
WeightedVectorWritable value = new WeightedVectorWritable();
while (reader.next(key, value)) {
  System.out.println(
    value.toString() + " belongs to cluster "
    + key.toString());
}
reader.close();
}

```

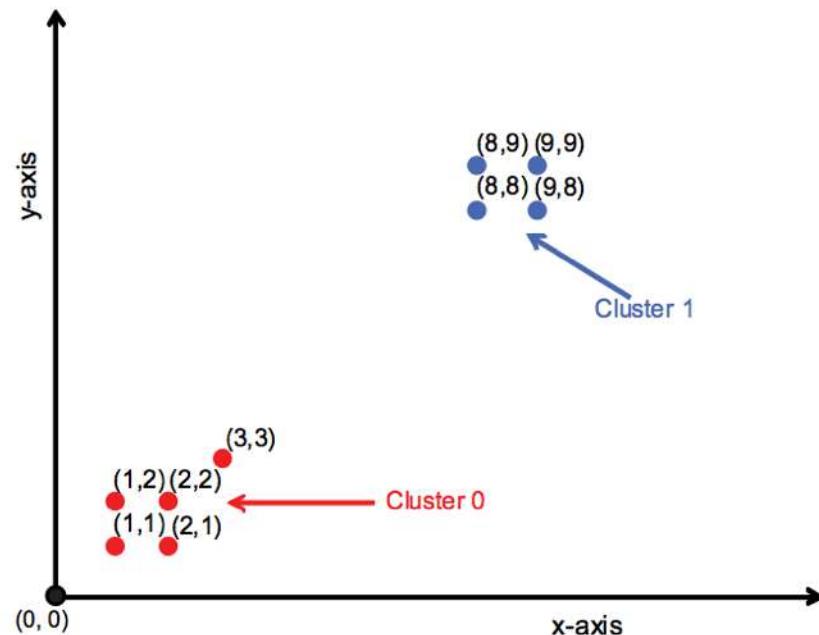
**Run k-means algorithm**

**Read output, print vector, cluster ID**

# HelloWorld clustering scenario result

```

1.0: [1.000, 1.000] belongs to cluster 0
1.0: [2.000, 1.000] belongs to cluster 0
1.0: [1.000, 2.000] belongs to cluster 0
1.0: [2.000, 2.000] belongs to cluster 0
1.0: [3.000, 3.000] belongs to cluster 0
1.0: [8.000, 8.000] belongs to cluster 1
1.0: [9.000, 8.000] belongs to cluster 1
1.0: [8.000, 9.000] belongs to cluster 1
1.0: [9.000, 9.000] belongs to cluster 1
  
```



## ***Euclidean distance measure***

$$d = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_n - b_n)^2}$$

## ***Squared Euclidean distance measure***

$$d = (a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_n - b_n)^2$$

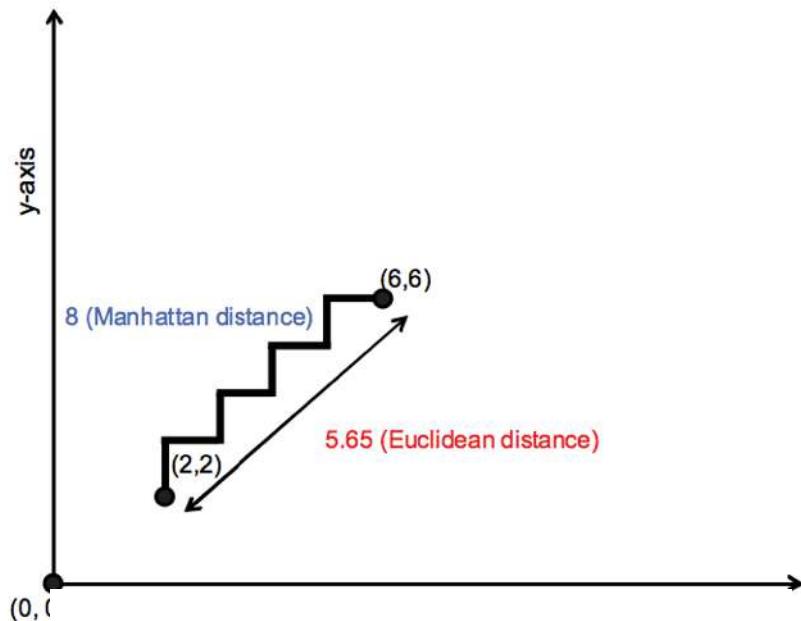
## ***Manhattan distance measure***

$$d = |a_1 - b_1| + |a_2 - b_2| + \dots + |a_n - b_n|$$

# Manhattan and Cosine distances

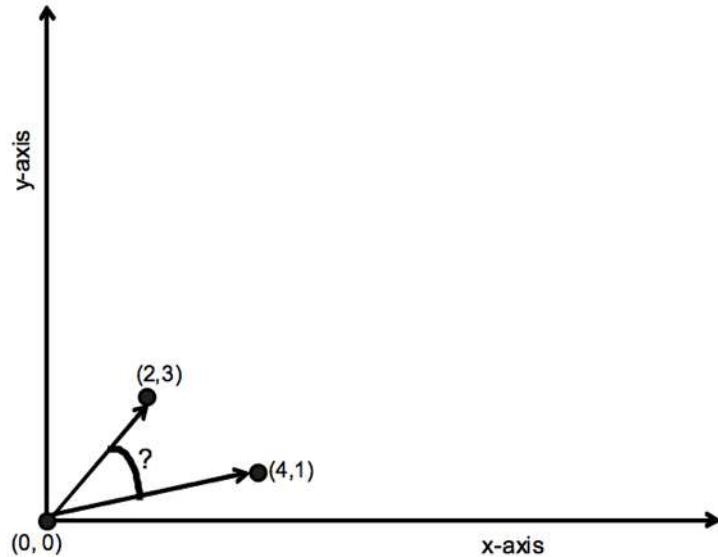
## **Manhattan distance measure**

$$d = |a_1 - b_1| + |a_2 - b_2| + \dots + |a_n - b_n|$$



## **Cosine distance measure**

$$d = 1 - \frac{(a_1 b_1 + a_2 b_2 + \dots + a_n b_n)}{(\sqrt{a_1^2 + a_2^2 + \dots + a_n^2}) \sqrt{(b_1^2 + b_2^2 + \dots + b_n^2)})}$$



# Tanimoto distance and weighted distance

## **Tanimoto distance measure**

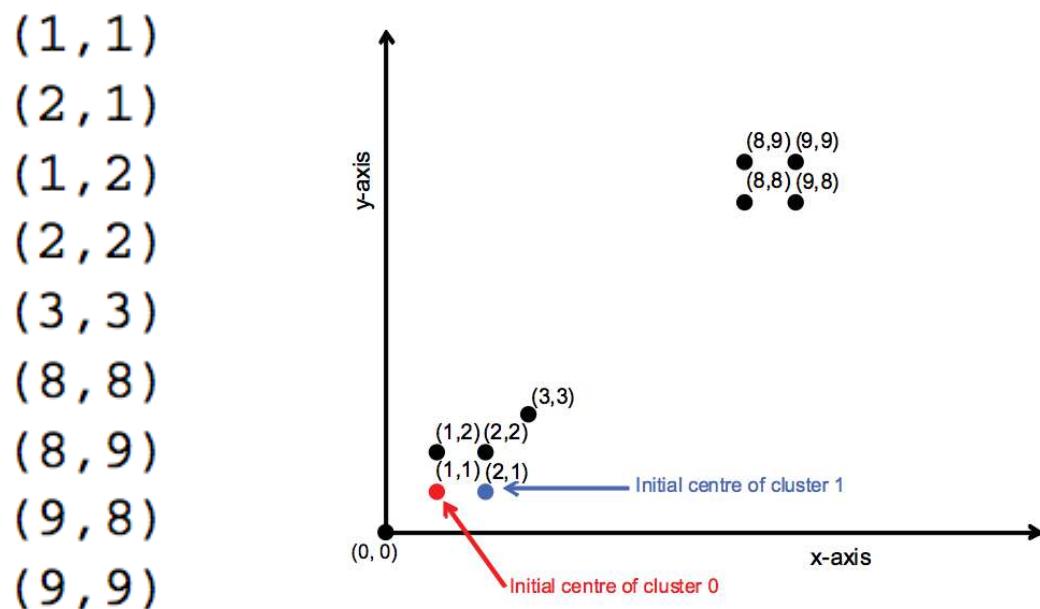
$$d = 1 - \frac{(a_1 b_1 + a_2 b_2 + \dots + a_n b_n)}{\sqrt{(a_1^2 + a_2^2 + \dots + a_n^2)} + \sqrt{(b_1^2 + b_2^2 + \dots + b_n^2)} - (a_1 b_1 + a_2 b_2 + \dots + a_n b_n)}$$

## **Weighted distance measure**

Mahout also provides a `WeightedDistanceMeasure` class, and implementations of Euclidean and Manhattan distance measures that use it. A weighted distance measure is an advanced feature in Mahout that allows you to give weights to different dimensions in order to either increase or decrease the effect of a dimension

# Results comparison

| Distance measure                | Number of iterations | Vectors <sup>a</sup> in cluster 0 | Vectors in cluster 1   |
|---------------------------------|----------------------|-----------------------------------|------------------------|
| EuclideanDistanceMeasure        | 3                    | 0, 1, 2, 3, 4                     | 5, 6, 7, 8             |
| SquaredEuclideanDistanceMeasure | 5                    | 0, 1, 2, 3, 4                     | 5, 6, 7, 8             |
| ManhattanDistanceMeasure        | 3                    | 0, 1, 2, 3, 4                     | 5, 6, 7, 8             |
| CosineDistanceMeasure           | 1                    | 1                                 | 0, 2, 3, 4, 5, 6, 7, 8 |
| TanimotoDistanceMeasure         | 3                    | 0, 1, 2, 3, 4                     | 5, 6, 7, 8             |

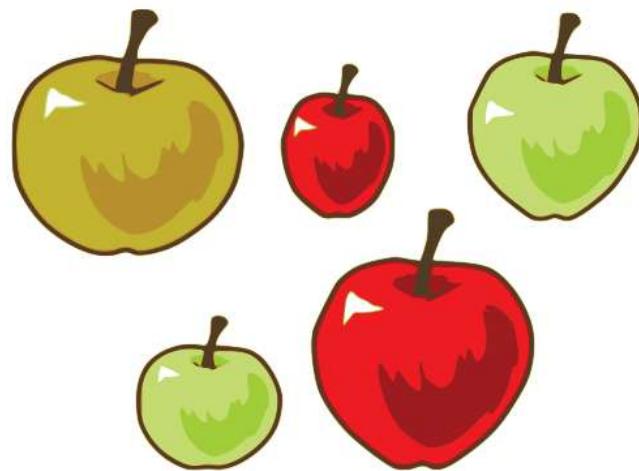


# Data preparation in Mahout — vectors

In Mahout, vectors are implemented as three different classes, each of which is optimized for different scenarios: `DenseVector`, `RandomAccessSparseVector`, and `SequentialAccessSparseVector`.

- `DenseVector` can be thought of as an array of doubles, whose size is the number of features in the data. Because all the entries in the array are preallocated regardless of whether the value is 0 or not, we call it *dense*.
- `RandomAccessSparseVector` is implemented as a `HashMap` between an integer and a double, where only nonzero valued features are allocated. Hence, they're called as `SparseVectors`.
- `SequentialAccessSparseVector` is implemented as two parallel arrays, one of integers and the other of doubles. Only nonzero valued entries are kept in it. Unlike the `RandomAccessSparseVector`, which is optimized for random access, this one is optimized for linear reading.

# vectorization example



0: weight

1: color

2: size

[0 => 100 gram, 1 => red, 2 => small]

| Apple                 | Weight (kg)<br>(0) | Color<br>(1) | Size<br>(2) | Vector         |
|-----------------------|--------------------|--------------|-------------|----------------|
| Small, round, green   | 0.11               | 510          | 1           | [0.11, 510, 1] |
| Large, oval, red      | 0.23               | 650          | 3           | [0.23, 650, 3] |
| Small, elongated, red | 0.09               | 630          | 1           | [0.09, 630, 1] |
| Large, round, yellow  | 0.25               | 590          | 3           | [0.25, 590, 3] |
| Medium, oval, green   | 0.18               | 520          | 2           | [0.18, 520, 2] |

# Mahout codes to create vectors of the apple example

```
public static void main(String args[]) throws Exception {  
    List<NamedVector> apples = new ArrayList<NamedVector>();  
  
    NamedVector apple;  
    apple = new NamedVector(  
        new DenseVector(new double[] {0.11, 510, 1}),  
        "Small round green apple");  
    apples.add(apple);  
    apple = new NamedVector(  
        new DenseVector(new double[] {0.23, 650, 3}),  
        "Large oval red apple");  
    apples.add(apple);  
    apple = new NamedVector(  
        new DenseVector(new double[] {0.09, 630, 1}),  
        "Small elongated red apple");  
    apples.add(apple);  
    apple = new NamedVector(  
        new DenseVector(new double[] {0.25, 590, 3}),  
        "Large round yellow apple");  
    apples.add(apple);  
    apple = new NamedVector(  
        new DenseVector(new double[] {0.18, 520, 2}),  
        "Medium oval green apple");  
}
```

**Associates a name  
with the vector**

```
Configuration conf = new Configuration();
FileSystem fs = FileSystem.get(conf);

Path path = new Path("appledata/apples");
SequenceFile.Writer writer = new SequenceFile.Writer(fs, conf,
    path, Text.class, VectorWritable.class);
VectorWritable vec = new VectorWritable();
for (NamedVector vector : apples) {
    vec.set(vector);
    writer.append(new Text(vector.getName()), vec);
}
writer.close();

SequenceFile.Reader reader = new SequenceFile.Reader(fs,
    new Path("appledata/apples"), conf);

Text key = new Text();
VectorWritable value = new VectorWritable();
while (reader.next(key, value)) {
    System.out.println(key.toString() + " "
        + value.get().asFormatString());
}
reader.close();
}
```

 **Serializes  
vector data**

 **Deserializes  
vector data**

## Vector Space Model: Term Frequency (TF)

For example, if the word *horse* is assigned to the 39,905<sup>th</sup> index of the vector, the word *horse* will correspond to the 39,905<sup>th</sup> dimension of document vectors. A document's vectorized form merely consists, then, of the number of times each word occurs in the document, and that value is stored in the vector along that word's dimension. The dimension of these document vectors can be very large.

Stop Words: *a, an, the, who, what, are, is, was, and so on.*

Stemming:

A stemmer for English, for example, should identify the **string** "cats" (and possibly "catlike", "catty" etc.) as based on the root "cat", and "stemmer", "stemming", "stemmed" as based on "stem". A stemming algorithm reduces the words "fishing", "fished", and "fisher" to the root word, "fish". On the other hand, "argue", "argued", "argues", "arguing", and "argus" reduce to the stem "argu" (illustrating the case where the stem is not itself a word or root) but "argument" and "arguments" reduce to the stem "argument".

# Most Popular Stemming algorithms

## Lookup algorithms

A simple stemmer looks up the inflected form in a [lookup table](#). The advantages of this approach is that it is simple, fast, and easily handles exceptions. The disadvantages are that all inflected forms must be explicitly listed in the table: new or unfamiliar words are not handled, even if they are perfectly regular (e.g. iPads ~ iPad), and the table may be large. For languages with simple morphology, like English, table sizes are modest, but

## Suffix-stripping algorithms

Suffix stripping algorithms do not rely on a lookup table that consists of inflected forms and root form relations. Instead, a typically smaller list of "rules" is stored which provides a path for the algorithm, given an input word form, to find its root form. Some examples of the rules include:

- if the word ends in 'ed', remove the 'ed'
- if the word ends in 'ing', remove the 'ing'
- if the word ends in 'ly', remove the 'ly'

The value of word is reduced more if it is used frequently across all the documents in the dataset.

To calculate the inverse document frequency, the document frequency (DF) for each word is first calculated. Document frequency is the number of documents the word occurs in. The number of times a word occurs in a document isn't counted in document frequency. Then, the inverse document frequency or  $IDF_i$  for a word,  $w_i$ , is

$$IDF_i = \frac{1}{DF_i}$$

$$W_i = TF_i \cdot IDF_i = TF_i \cdot \frac{N}{DF_i} \quad \text{or} \quad W_i = TF_i \cdot \log \frac{N}{DF_i}$$

It was the best of time. it was the worst of times.

==>  
bigram

It was  
was the  
the best  
best of  
of times  
times it  
it was  
was the  
the worst  
worst of  
of times

Mahout provides a log-likelihood test to reduce the dimensions of n-grams

## Examples — using a news corpus

Reuters-21578 dataset: 22 files, each one has 1000 documents except the last one.

<http://www.daviddlewis.com/resources/testcollections/reuters21578/>

Extraction code:

```
mvn -e -q exec:java  
-Dexec.mainClass="org.apache.lucene.benchmark.utils.ExtractReuters"  
-Dexec.args="reuters/ reuters-extracted/"
```

Using the extracted folder, run the SequenceFileFromDirectory class. You can use the launcher script from the Mahout root directory to do the same:

```
bin/mahout seqdirectory -c UTF-8  
-i examples/reuters-extracted/ -o reuters-seqfiles
```

This will write the Reuters articles in the SequenceFile format. Now the only step left is to convert this data to vectors. To do that, run the SparseVectorsFromSequenceFiles class using the Mahout launcher script:

```
bin/mahout seq2sparse -i reuters-seqfiles/ -o reuters-vectors -ow
```

# Mahout dictionary-based vectorizer

| Option                           | Flag   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                 | Default value                                        |
|----------------------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------|
| Overwrite<br>(bool)              | -ow    | If set, the output folder is overwritten. If not set, the output folder is created if the folder doesn't exist. If the output folder does exist, the job fails and an error is thrown. Default is unset.                                                                                                                                                                                                                                    | N/A                                                  |
| Lucene analyzer name<br>(String) | -a     | The class name of the analyzer to use.                                                                                                                                                                                                                                                                                                                                                                                                      | org.apache.lucene.analysis.standard.StandardAnalyzer |
| Chunk size<br>(int)              | -chunk | The chunk size in MB. For large document collections (sizes in GBs and TBs), you won't be able to load the entire dictionary into memory during vectorization, so you can split the dictionary into chunks of the specified size and perform the vectorization in multiple stages. It's recommended you keep this size to 80 percent of the Java heap size of the Hadoop child nodes to prevent the vectorizer from hitting the heap limit. | 100                                                  |
| Weighting<br>(String)            | -wt    | The weighting scheme to use: tf for term-frequency based weighting and tfidf for TF-IDF based weighting.                                                                                                                                                                                                                                                                                                                                    | tfidf                                                |
| Minimum support<br>(int)         | -s     | The minimum frequency of the term in the entire collection to be considered as a part of the dictionary file. Terms with lesser frequency are ignored.                                                                                                                                                                                                                                                                                      | 2                                                    |

| Option                                  | Flag | Description                                                                                                                                                                                                                                                      | Default value |
|-----------------------------------------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| Minimum document frequency (int)        | -md  | The minimum number of documents the term should occur in to be considered a part of the dictionary file. Any term with lesser frequency is ignored.                                                                                                              | 1             |
| Max document frequency percentage (int) | -x   | The maximum number of documents the term should occur in to be considered a part of the dictionary file. This is a mechanism to prune out high frequency terms (stop-words). Any word that occurs in more than the specified percentage of documents is ignored. | 99            |
| $N$ -gram size (int)                    | -ng  | The maximum size of $n$ -grams to be selected from the collection of documents.                                                                                                                                                                                  | 1             |

# Mahout dictionary-based vectorizer — III

| Option                                         | Flag | Description                                                                                                                                                                                                                                                                                                                                                                                                              | Default value |
|------------------------------------------------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| Minimum log-likelihood ratio (LLR) (float)     | -ml  | This flag works only when $n$ -gram size is greater than 1. Very significant $n$ -grams have large scores, such as 1000; less significant ones have lower scores. Although there's no specific method for choosing this value, the rule of thumb is that $n$ -grams with a LLR value less than 1.0 are irrelevant.                                                                                                       | 1.0           |
| Normalization (float)                          | -n   | The normalization value to use in the $L_p$ space. A detailed explanation of normalization is given in section 8.4. The default scheme is to not normalize the weights.                                                                                                                                                                                                                                                  | 0             |
| Number of reducers (int)                       | -nr  | The number of reducer tasks to execute in parallel. This flag is useful when running a dictionary vectorizer on a Hadoop cluster. Setting this to the maximum number of nodes in the cluster gives maximum performance. Setting this value higher than the number of cluster nodes leads to a slight decrease in performance. For more details, read the Hadoop documentation on setting the optimum number of reducers. | 1             |
| Create sequential access sparse vectors (bool) | -seq | If set, the output vectors are created as <code>SequentialAccessSparseVectors</code> . By default the dictionary vectorizer generates <code>RandomAccessSparseVectors</code> . The former gives higher performance on certain algorithms like k-means and SVD due to the sequential nature of vector operations. By default the flag is unset.                                                                           | N/A           |

# Outputs & Steps

---

```
$ ls reuters-vectors/  
df-count/  
dictionary.file-0  
frequency.file-0  
tfidf-vectors/  
tf-vectors/  
tokenized-documents/  
  
wordcount/
```

1. Tokenization using Lucene StandardAnalyzer
2. n-gram generation step
3. converts the tokenized documents into vectors using TF
4. count DF and then create TF-IDF

# A practical setting of flags

- -a—Use `org.apache.lucene.analysis.WhitespaceAnalyzer` to tokenize words based on the whitespace characters between them.
- -chunk—Use a chunk size of 200 MB. This value won't produce any effect on the Reuters data, because the dictionary sizes are usually in the 1 MB range.
- -wt—Use the tfidf weighting method.
- -s—Use a minimum support value of 5.
- -md—Use a minimum document frequency value of 3.
- -x—Use a maximum document frequency percentage of 90 percent to aggressively prune away high-frequency words.
- -ng—Use an *n*-gram size of 2 to generate both unigrams and bigrams.
- -ml—Use a minimum log-likelihood ratio (LLR) value of 50 to keep only very significant bigrams.
- -seq—Set the `SequentialAccessSparseVectors` flag.

Run the vectorizer using the preceding options in the Mahout launcher script:

```
bin/mahout seq2sparse -i reuters-seqfiles/ -o reuters-vectors-bigram -ow  
-a org.apache.lucene.analysis.WhitespaceAnalyzer  
-chunk 200 -wt tfidf -s 5 -md 3 -x 90 -ng 2 -ml 50 -seq
```

# normalization

---

Some documents may pop up showing they are similar to all the other documents because it is large. ==> Normalization can help.

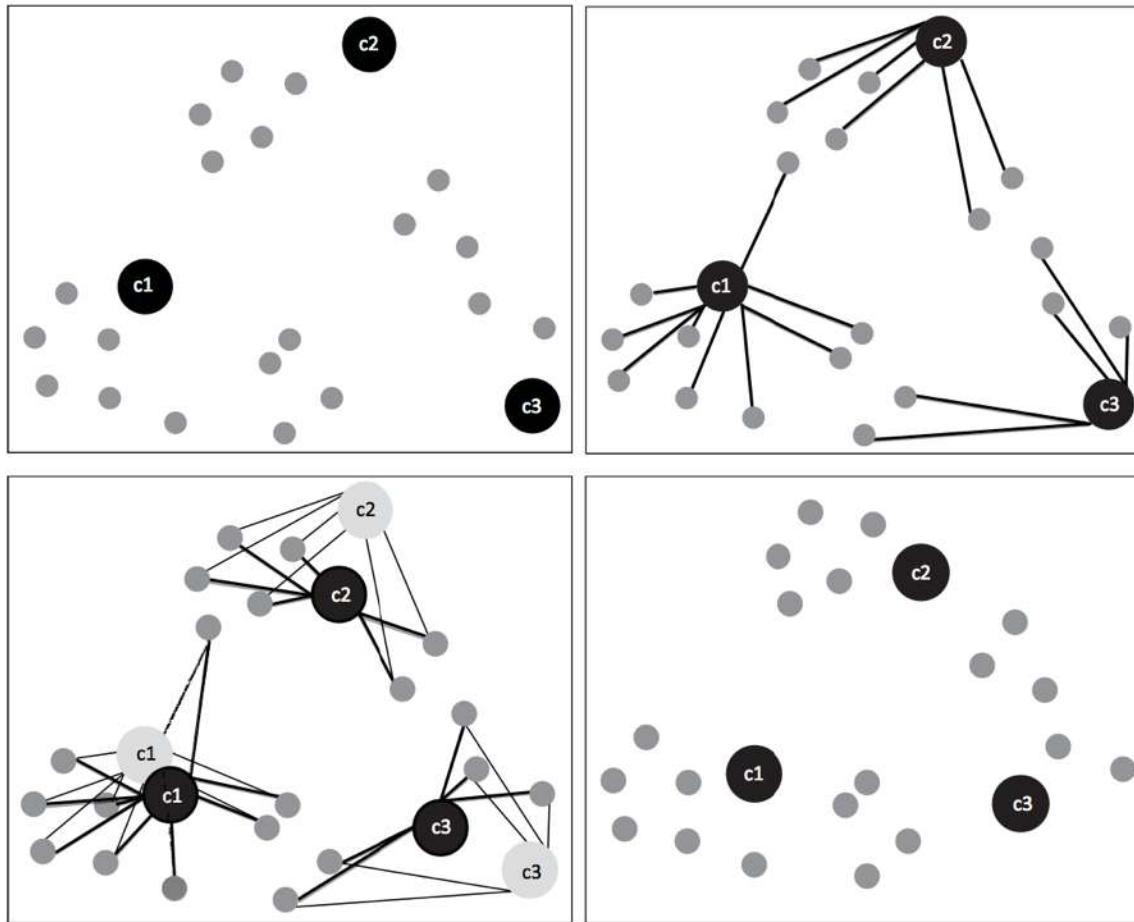
In Mahout, normalization uses what is known in statistics as a  $p$ -norm. For example, the  $p$ -norm of a 3-dimensional vector,  $[x, y, z]$ , is

$$\frac{x}{(|x|^p + |y|^p + |z|^p)^{1/p}}, \frac{y}{(|x|^p + |y|^p + |z|^p)^{1/p}}, \frac{z}{(|x|^p + |y|^p + |z|^p)^{1/p}}$$

# Clustering methods provided by Mahout

- K-means clustering
- Centroid generation using canopy clustering
- Fuzzy k-means clustering and Dirichlet clustering
- Topic modeling using latent Dirichlet allocation as a variant of clustering

# K-mean clustering



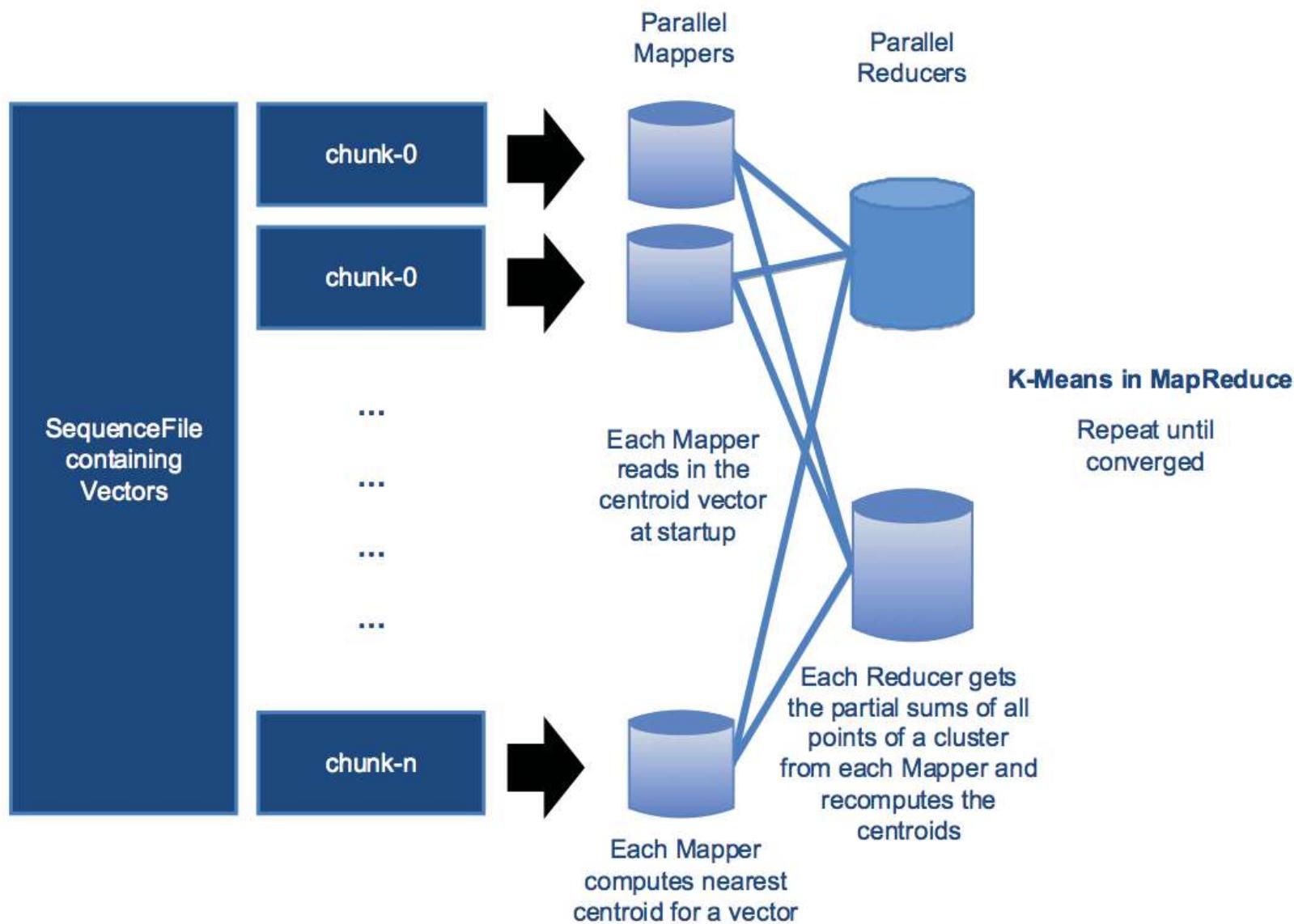
**K-means clustering in action.** Starting with three random points as centroids (top left), the map stage (top right) assigns each point to the cluster nearest to it. In the reduce stage (bottom left), the associated points are averaged out to produce the new location of the centroid, leaving you with the final configuration (bottom right). After each iteration, the final configuration is fed back into the same loop until the centroids come to rest at their final positions.

# Hadoop k-mean clustering jobs

In Mahout, the MapReduce version of the k-means algorithm is instantiated using the KMeansDriver class. The class has just a single entry point—the runJob method.

- The Hadoop configuration.
- The SequenceFile containing the input Vectors.
- The SequenceFile containing the initial Cluster centers.
- The similarity measure to be used. We'll use EuclideanDistanceMeasure as the measure of similarity and experiment with the others later.
- The convergenceThreshold. If in an iteration, the centroids don't move more than this distance, no further iterations are done and clustering stops.
- The number of iterations to be done. This is a hard limit; the clustering stops if this threshold is reached.

# K-mean clustering running as MapReduce job



# Hadoop k-mean clustering code

```
KmeansDriver.runJob(hadoopConf,  
    inputVectorFilesDirPath, clusterCenterFilesDirPath,  
    outputDir, new EuclideanDistanceMeasure(),  
    convergenceThreshold, numIterations, true, false);
```

Mahout reads and writes data using the Hadoop `FileSystem` class. This provides seamless access to both the local filesystem (via `java.io`) and distributed filesystems like HDFS and S3FS (using internal Hadoop classes). This way, the same code that works on the local system will also work on the Hadoop filesystem on the cluster, provided the paths to the Hadoop configuration files are correctly set in the environment variables. In Mahout, the `bin/mahout` shell script finds the Hadoop configuration files automatically from the `$HADOOP_CONF` environment variable.

```
$ bin/mahout kmeans -i reuters-vectors/tfidf-vectors/ \  
-c reuters-initial-clusters \  
-o reuters-kmeans-clusters \  
-dm org.apache.mahout.common.distance.SquaredEuclideanDistanceMeasure \  
-cd 1.0 -k 20 -x 20 -cl
```

# The output

The directory listing of the output folder looks something like this:

```
$ ls -l reuters-kmeans-clusters
drwxr-xr-x  4 user  5000  136 Feb 1 18:56 clusters-0
drwxr-xr-x  4 user  5000  136 Feb 1 18:56 clusters-1
drwxr-xr-x  4 user  5000  136 Feb 1 18:56 clusters-2
...
drwxr-xr-x  4 user  5000  136 Feb 1 18:59 clusteredPoints

$ bin/mahout clusterdump -dt sequencefile \
-d reuters-vectors/dictionary.file-* \
-s reuters-kmeans-clusters/clusters-19 -b 10 -n 10
```

Running ClusterDumper on the output folder corresponding to the last iteration produces output similar to the following:

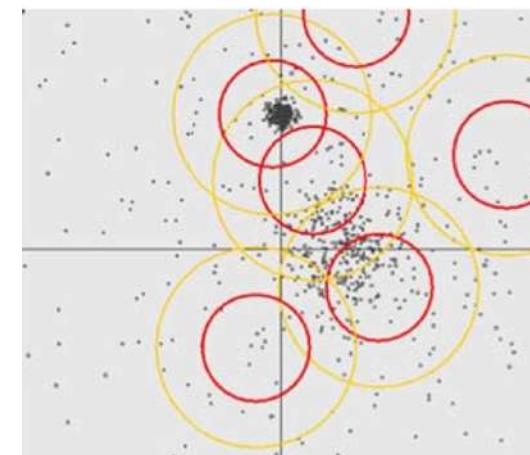
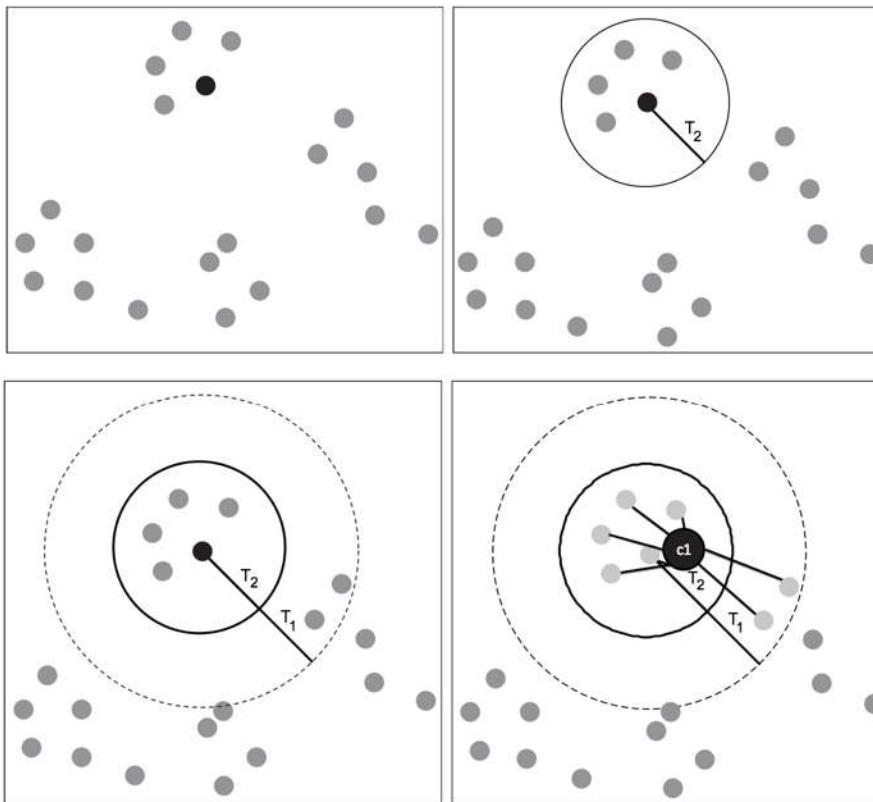
```
Id: 11736:
Top Terms: debt, banks, brazil, bank, billion, he, payments, billion
dlrs, interest, foreign

Id: 11235:
Top Terms: amorphous, magnetic, metals, allied signal, 19.39, corrosion,
allied, molecular, mode, electronic components
...

Id: 20073:
Top Terms: ibm, computers, computer, att, personal, pc, operating system,
intel, machines, dos
```

# Canopy clustering to estimate the number of clusters

Tell what size clusters to look for. The algorithm will find the number of clusters that have approximately that size. The algorithm uses two distance thresholds. This method prevents all points close to an already existing canopy from being the center of a new canopy.



**Canopy clustering:** if you start with a point (top left) and mark it as part of a canopy, all the points within distance  $T_2$  (top right) are removed from the data set and prevented from becoming new canopies. The points within the outer circle (bottom-right) are also put in the same canopy, but they're allowed to be part of other canopies. This assignment process is done in a single pass on a mapper. The reducer computes the average of the centroid (bottom right) and merges close canopies.

# Running canopy clustering

To run canopy generation over the Reuters data set, execute the canopy program using the Mahout launcher as follows:

```
$ bin/mahout canopy -i reuters-vectors/tfidf-vectors \
-o reuters-canopy-centroids \
-dm org.apache.mahout.common.distance.EuclideanDistanceMeasure \
-t1 1500 -t2 2000
```

Within a minute, CanopyDriver will generate the centroids in the output folder.  
Created less than 50 centroids.

```
$ bin/mahout kmeans -i reuters-vectors/tfidf-vectors \
-o reuters-kmeans-clusters \
-dm org.apache.mahout.common.distance.TanimotoDistanceMeasure \
-c reuters-canopy-centroids/clusters-0 -cd 0.1 -ow -x 20 -cl
```

After the clustering is done, use ClusterDumper to inspect the clusters. Some of them are listed here:

```
Id: 21523:name:
    Top Terms:
tones, wheat, grain, said, usda, corn, us, sugar, export, agriculture
Id: 21409:name:
    Top Terms:
stock, share, shares, shareholders, dividend, said, its, common, board,
    company
Id: 21155:name:
    Top Terms:
oil, effective, crude, raises, prices, barrel, price, cts, said, dtrs
Id: 19658:name:
    Top Terms:
drug, said, aids, inc, company, its, patent, test, products, food
Id: 21323:name:
    Top Terms:
7-apr-1987, 11, 10, 12, 07, 09, 15, 16, 02, 17
```

# News clustering code

```
public class NewsKMeansClustering {  
    public static void main(String args[]) throws Exception {  
        int minSupport = 2;  
        int minDf = 5;  
        int maxDFPercent = 95;  
        int maxNGramSize = 2;  
        int minLLRValue = 50;  
        int reduceTasks = 1;  
        int chunkSize = 200;  
        int norm = 2;  
        boolean sequentialAccessOutput = true;  
  
        String inputDir = "inputDir";  
  
        Configuration conf = new Configuration();  
        FileSystem fs = FileSystem.get(conf);  
  
        String outputDir = "newsClusters";  
        HadoopUtil.delete(new Path(outputDir));  
        Path tokenizedPath = new Path(outputDir,  
            DocumentProcessor.TOKENIZED_DOCUMENT_OUTPUT_FOLDER);  
        MyAnalyzer analyzer = new MyAnalyzer();
```

Custom  
Lucene  
Analyzer

## [Obama to Name 'Smart Grid' Projects](#)

Wall Street Journal - [Rebecca Smith](#) - 1 hour ago

The Obama administration is expected Tuesday to name 100 utility projects that will share \$3.4 billion in federal stimulus funding to speed deployment of advanced technology designed to cut energy use and make the electric-power grid ...

[Cobb firm wins "smart-grid" grant](#) Atlanta Journal Constitution

[Obama putting \\$3.4B toward a 'smart' power grid](#) The Associate  
Baltimore Sun - Bloomberg - New York Times - Reuters

[all 594 news articles »](#) [Email this story](#)

# News clustering code – II

```

DocumentProcessor.tokenizeDocuments(new Path(inputDir),
    analyzer.getClass().asSubclass(Analyzer.class),
    tokenizedPath, conf);                                ← Tokenize text

DictionaryVectorizer.createTermFrequencyVectors(tokenizedPath,
    new Path(outputDir), conf, minSupport, maxNGramSize, minLLRValue,
    2, true, reduceTasks,
    chunkSize, sequentialAccessOutput, false);

TFIDFConverter.processTfIdf(
    new Path(outputDir ,
    DictionaryVectorizer.DOCUMENT_VECTOR_OUTPUT_FOLDER),
    new Path(outputDir), conf, chunkSize, minDf,
    maxDFPercent, norm, true, sequentialAccessOutput, false,
    reduceTasks);

Path vectorsFolder = new Path(outputDir, "tfidf-vectors");
Path canopyCentroids = new Path(outputDir ,
                                  "canopy-centroids");

Path clusterOutput = new Path(outputDir , "clusters");

CanopyDriver.run(vectorsFolder, canopyCentroids,
    new EuclideanDistanceMeasure(), 250, 120,
    false, false);

KMeansDriver.run(conf, vectorsFolder,
    new Path(canopyCentroids, "clusters-0"),
    clusterOutput, new TanimotoDistanceMeasure(), 0.01,
    20, true, false);
  
```

← **Calculate TF-IDF vectors**

← **Run canopy centroid generation**

← **Run k-means algorithm**

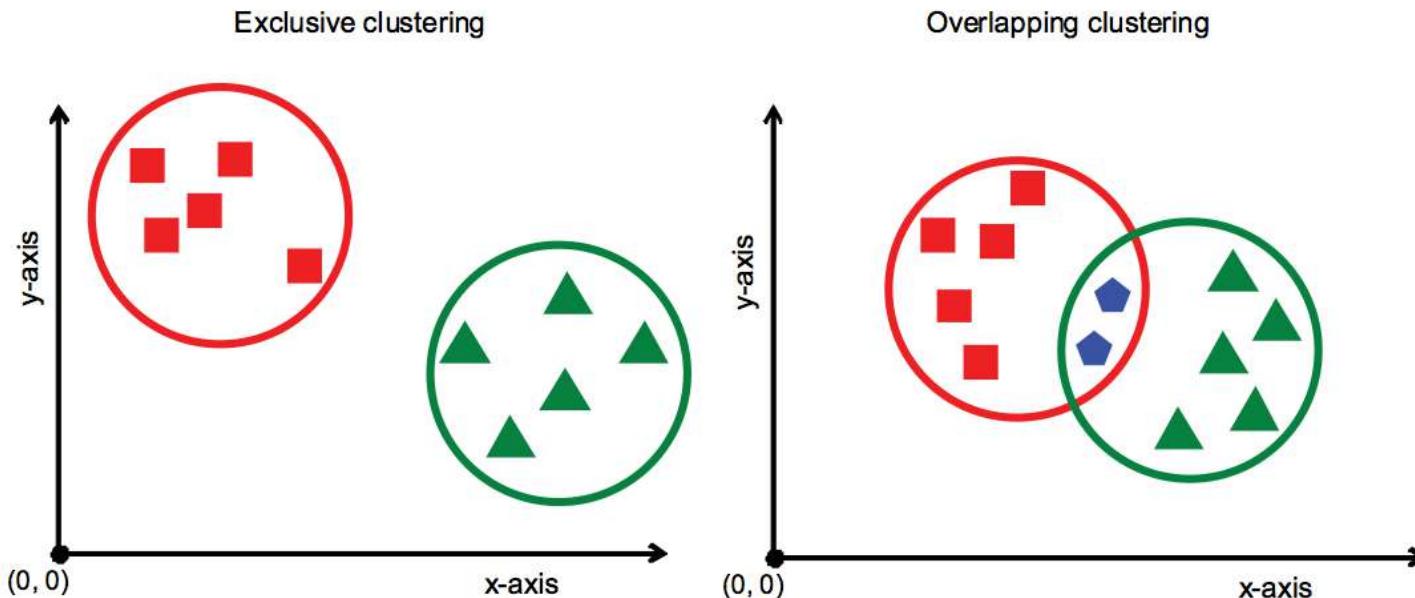
# News clustering code – III

```
SequenceFile.Reader reader = new SequenceFile.Reader(fs,
    new Path(clusterOutput
        + Cluster.CLUSTERED_POINTS_DIR + "/part-00000"), conf);

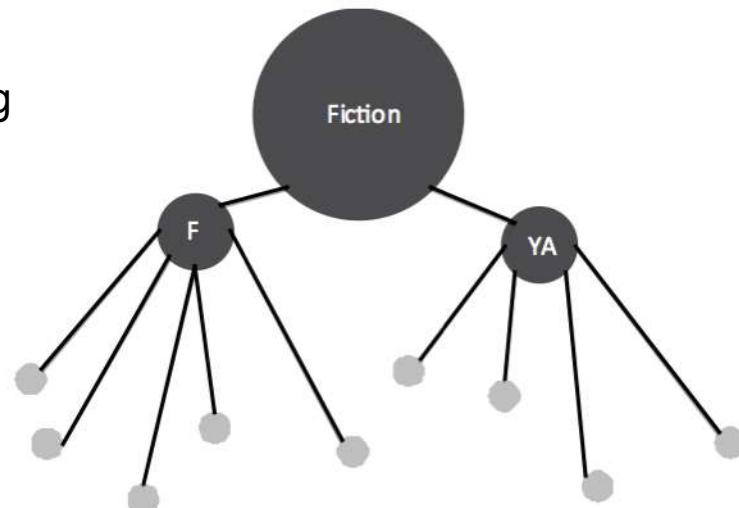
IntWritable key = new IntWritable();
WeightedVectorWritable value = new WeightedVectorWritable();
while (reader.next(key, value)) {
    System.out.println(key.toString() + " belongs to cluster "
        + value.toString());
}
reader.close();
}
```

Read Vector to  
Cluster mappings

# Other clustering algorithms



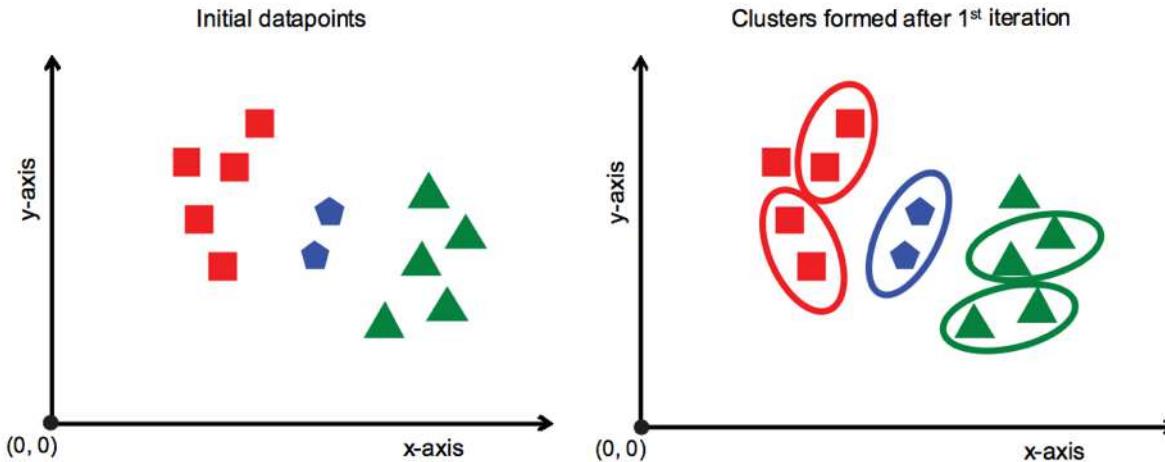
Hierarchical clustering



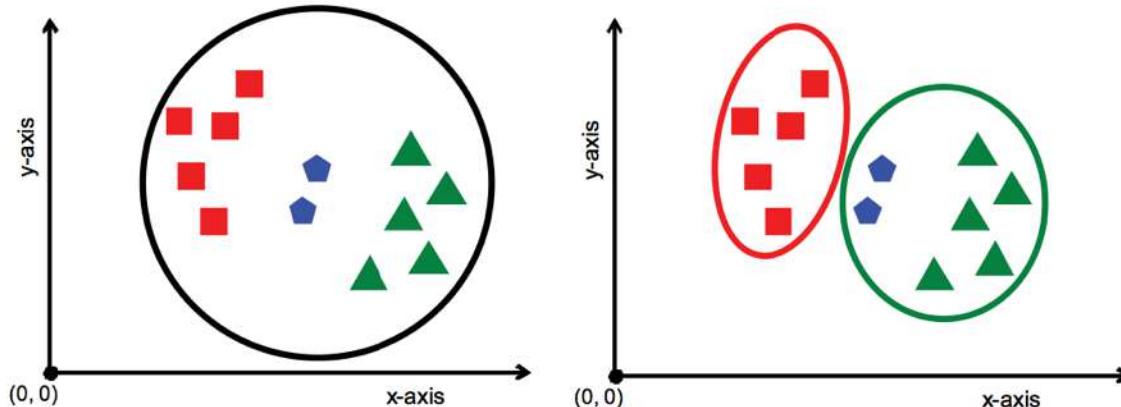
# Different clustering approaches

## FIXED NUMBER OF CENTERS

### BOTTOM-UP APPROACH: FROM POINTS TO CLUSTERS VIA GROUPING



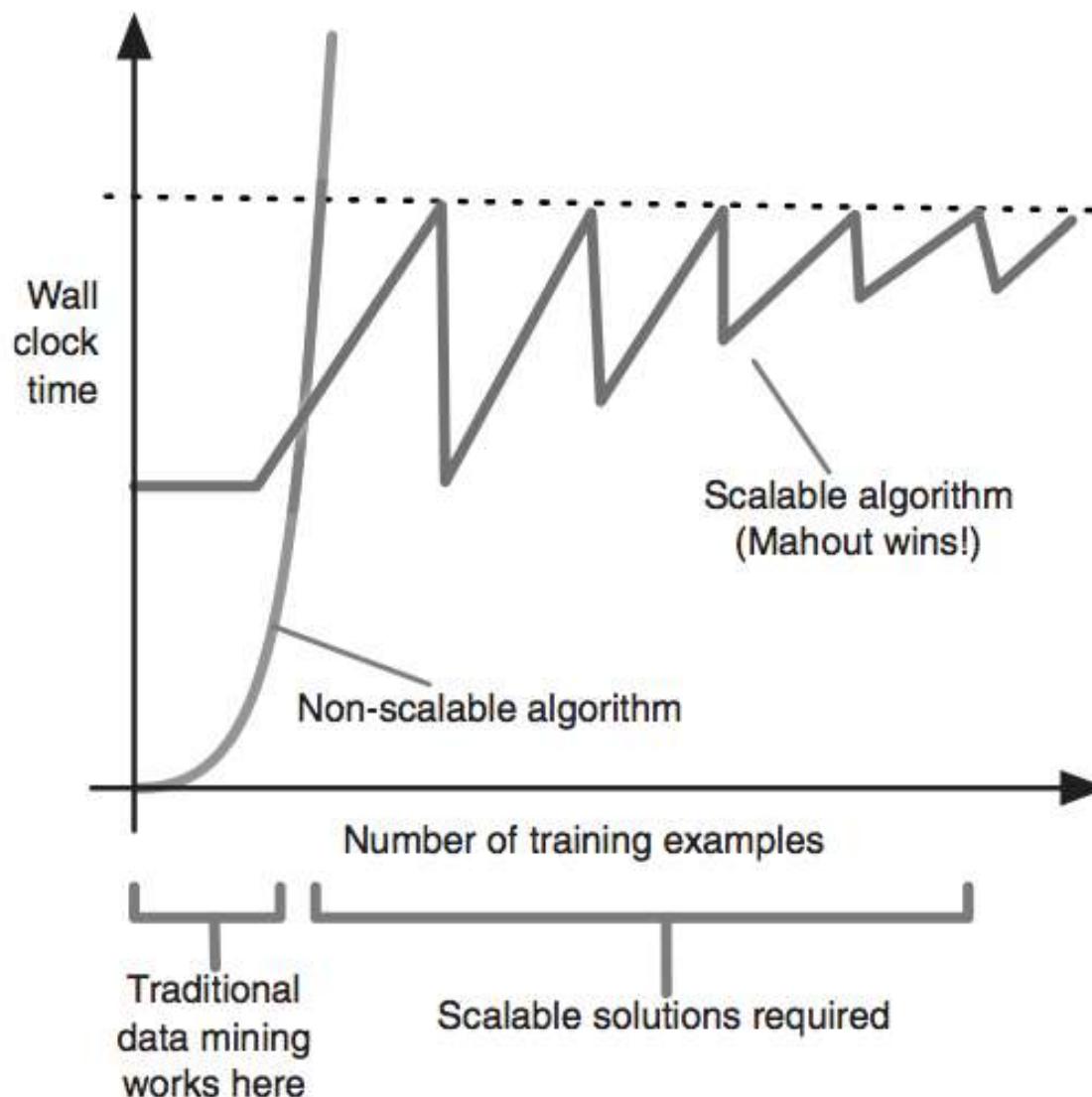
### TOP-DOWN APPROACH: SPLITTING THE GIANT CLUSTER



# When to use Mahout for classification?

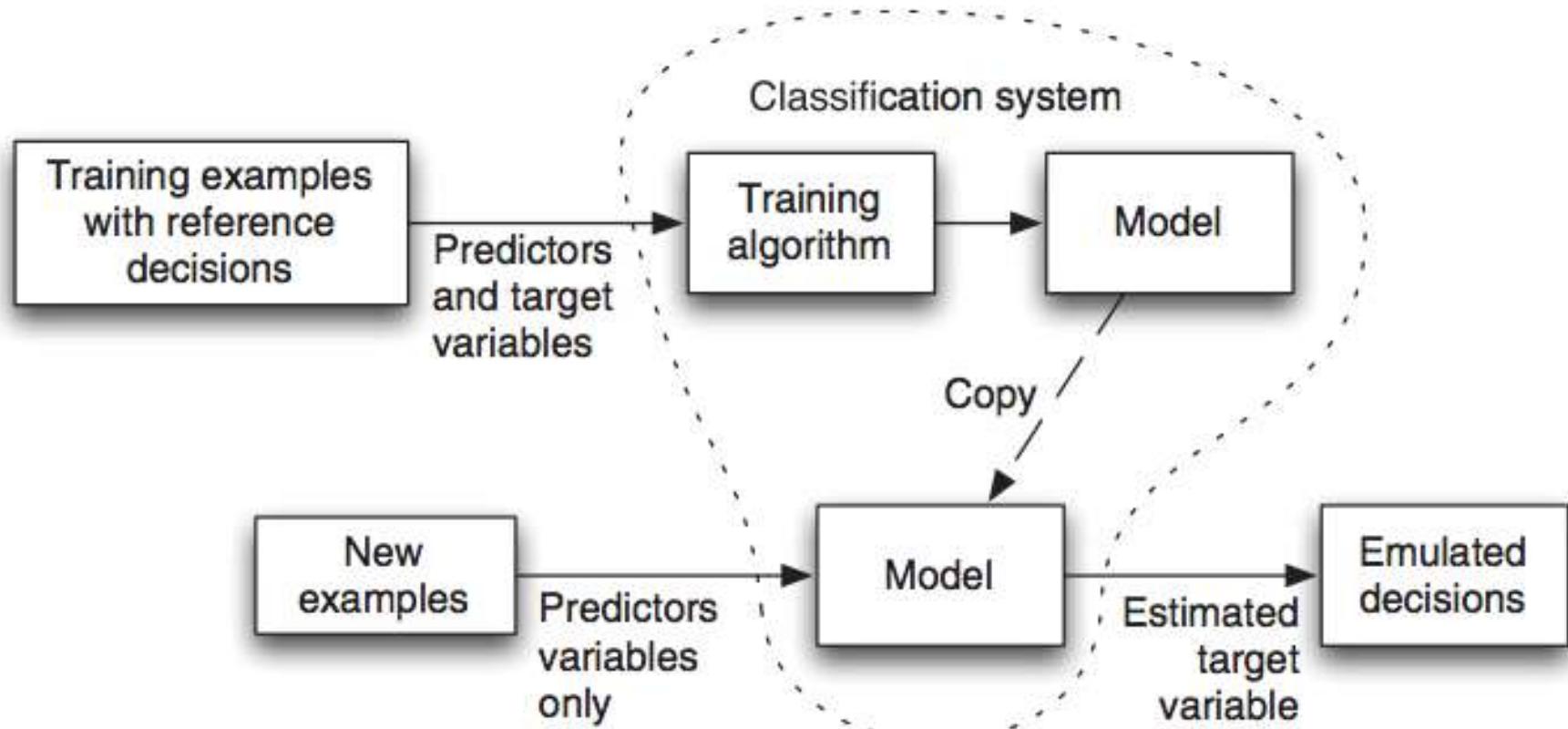
| System size In number of examples | Choice of classification approach                                                                                                      |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| < 100,000                         | Traditional, non-Mahout approaches should work very well. Mahout may even be slower for training.                                      |
| 100,000 to 1 million              | Mahout begins to be a good choice. The flexible API may make Mahout a preferred choice, even though there is no performance advantage. |
| 1 million to 10 million           | Mahout is an excellent choice in this range.                                                                                           |
| > 10 million                      | Mahout excels where others fail.                                                                                                       |

# The advantage of using Mahout for classification



**DEFINITION** Computer classification systems are a form of machine learning that use learning algorithms to provide a way for computers to make decisions based on experience and, in the process, emulate certain forms of human decision making.

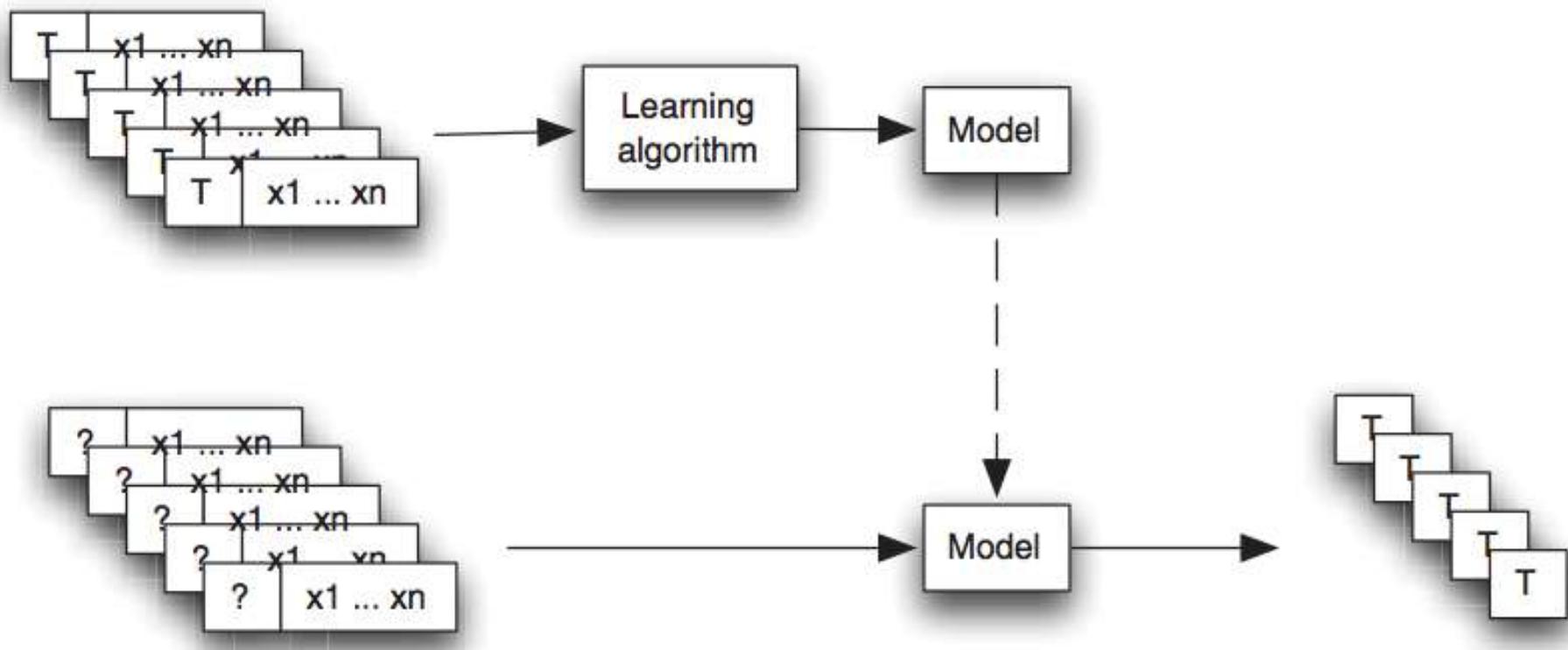
# How does a classification system work?



# Key terminology for classification

| Key Idea           | Description                                                                                                                                                            |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model              | A computer program that makes decisions; in classification, the output of the training algorithm is a model.                                                           |
| Training data      | A subset of training examples labeled with the value of the target variable and used as input to the learning algorithm to produce the model.                          |
| Test data          | A withheld portion of the training data with the value of the target variable hidden so that it can be used to evaluate the model.                                     |
| Training           | The learning process that uses training data to produce a model. That model can then compute estimates of the target variable given the predictor variables as inputs. |
| Training example   | An entity with features that will be used as input for learning algorithm.                                                                                             |
| Feature            | A known characteristic of a training or a new example; a feature is equivalent to a characteristic.                                                                    |
| Variable           | In this context, the value of a feature or a function of several features. This usage is somewhat different from the use of <i>variable</i> in a computer program.     |
| Record             | A container where an example is stored; such a record is composed of fields.                                                                                           |
| Field              | Part of a record that contains the value of a feature (a variable).                                                                                                    |
| Predictor variable | A feature selected for use as input to a classification model. Not all features need be used. Some features may be algorithmic combinations of other features.         |
| Target variable    | A feature that the classification model is attempting to estimate: the target variable is categorical, and its determination is the aim of the classification system.  |

# Input and Output of a classification model



# Four types of values for predictor variables

| Type of value | Description                                                                                                                                                                                                                                                                                      |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Continuous    | This is a floating-point value. This type of value might be a price, a weight, a time, or anything else that has a numerical magnitude and where this magnitude is the key property of the value.                                                                                                |
| Categorical   | A categorical value can have one of a set of prespecified values. Typically the set of categorical values is relatively small and may be as small as two, although the set can be quite large. Boolean values are generally treated as categorical values. Another example might be a vendor ID. |
| Word-like     | A word-like value is like a categorical value, but it has an open-ended set of possible values.                                                                                                                                                                                                  |
| Text-like     | A text-like value is a sequence of word-like values, all of the same kind. Text is the classic example of a text-like value, but a list of email addresses or URLs is also text-like.                                                                                                            |

# Sample data that illustrates all four value types

| Name             | Type                      | Value                           |
|------------------|---------------------------|---------------------------------|
| from-address     | Word-like                 | George <george@fumble-tech.com> |
| in-address-book? | Categorical (TRUE, FALSE) | TRUE                            |
| non-spam-words   | Text-like                 | Ted, Mahout, User, lunch        |
| spam-words       | Text-like                 | available                       |
| unknown-words    | Continuous                | 0                               |
| message-length   | Continuous                | 31                              |

# Supervised vs. Unsupervised Learning

---

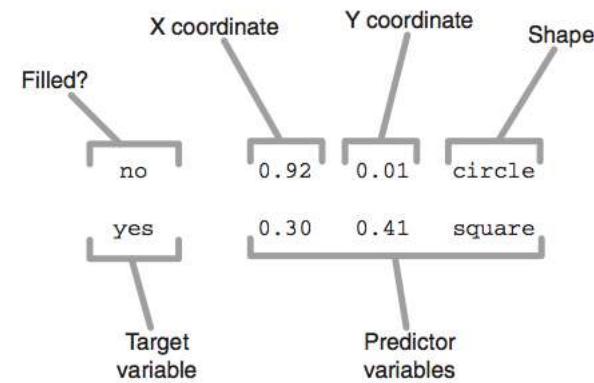
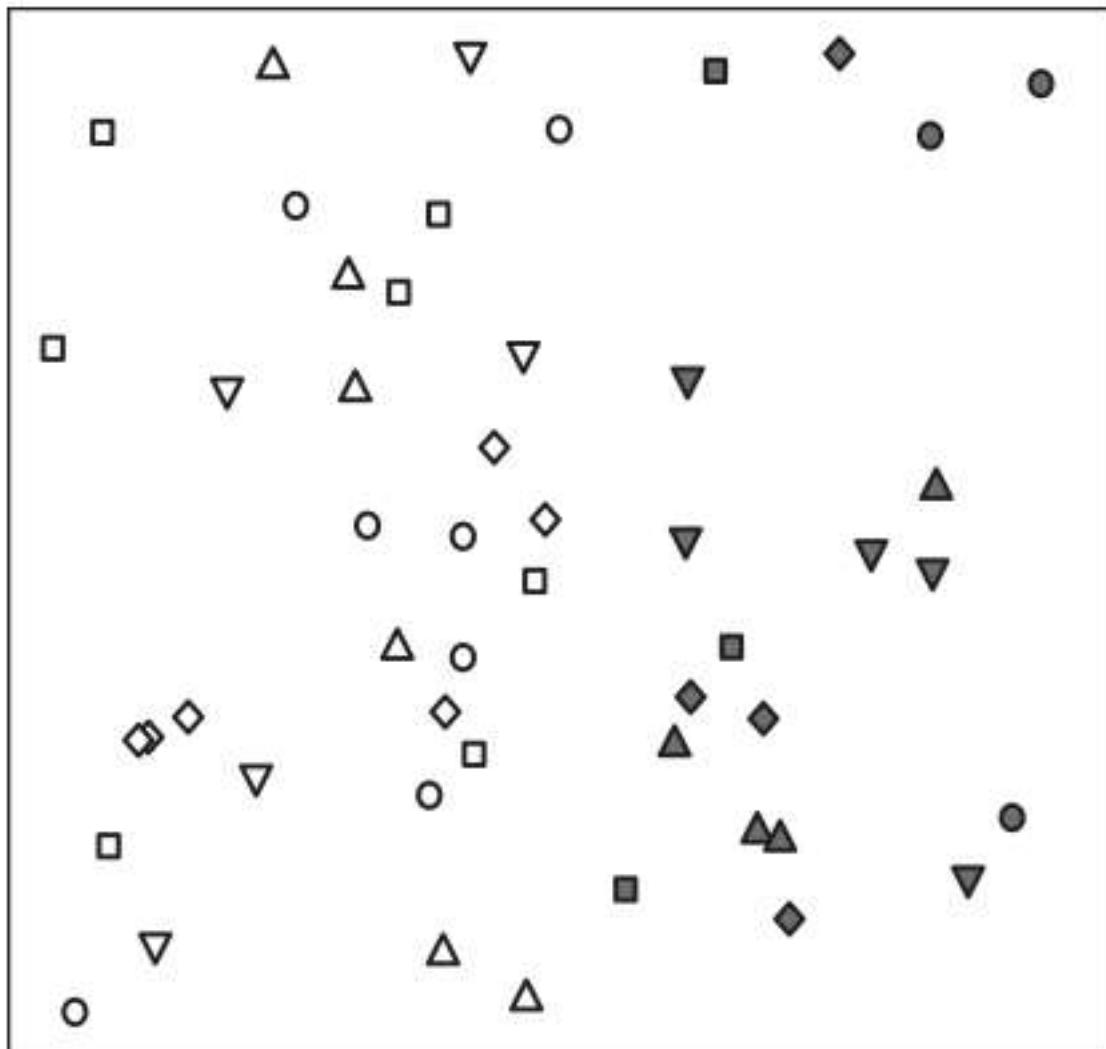
Classification algorithms are related to, but still quite different from, clustering algorithms such as the k-means algorithm described in previous chapters. Classification algorithms are a form of supervised learning, as opposed to unsupervised learning, which happens with clustering algorithms. A supervised learning algorithm is one that's given examples that contain the desired value of a target variable. Unsupervised algorithms aren't given the desired answer, but instead must find something plausible on their own.

Supervised and unsupervised learning algorithms can often be usefully combined. A clustering algorithm can be used to create features that can then be used by a learning algorithm, or the output of several classifiers can be used as features by a clustering algorithm. Moreover, clustering systems often build a model that can be used to categorize new data. This clustering system model works much like the model produced by a classification system. The difference lies in what data was used to produce the model. For classification, the training data includes the target variables; for clustering, the training data doesn't include target variables.

# Work flow in a typical classification project

| Stage                            | Step                                                                                                                                                                 |
|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. Training the model            | Define target variable.<br>Collect historical data.<br>Define predictor variables.<br>Select a learning algorithm.<br>Use the learning algorithm to train the model. |
| 2. Evaluating the model          | Run test data.<br>Adjust the input (use different predictor variables, different algorithms, or both).                                                               |
| 3. Using the model in production | Input new examples to estimate unknown target values.<br>Retrain the model as needed.                                                                                |

# Classification Example 1 — Color-Fill



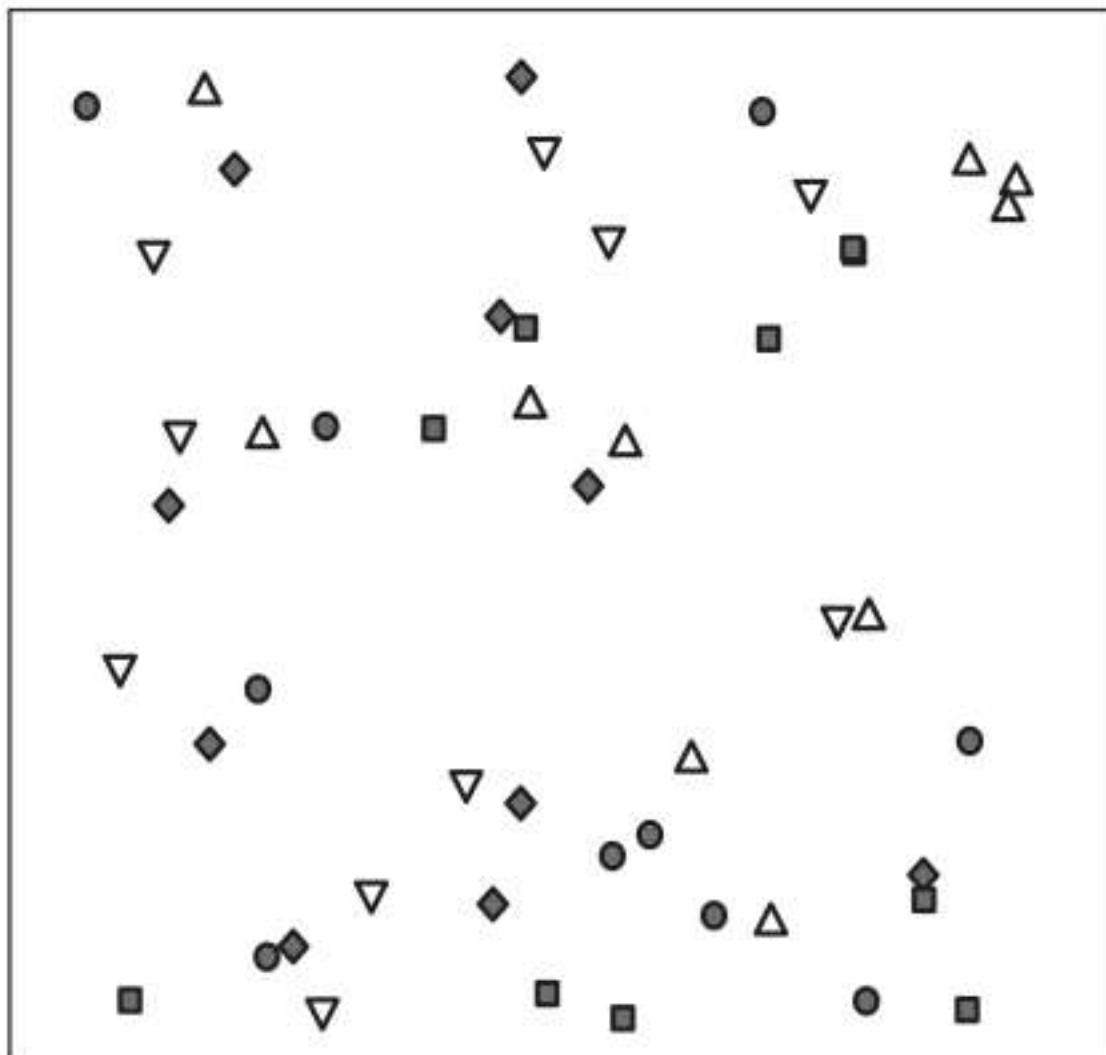
Position looks promising, especially the x-axis ==> predictor variable.  
Shape seems to be irrelevant. Target variable is “color-fill” label.

# Target leak

---

- A target leak is a bug that involves unintentionally providing data about the target variable in the section of the predictor variables.
- Don't confused with intentionally including the target variable in the record of a training example.
- Target leaks can seriously affect the accuracy of the classification system.

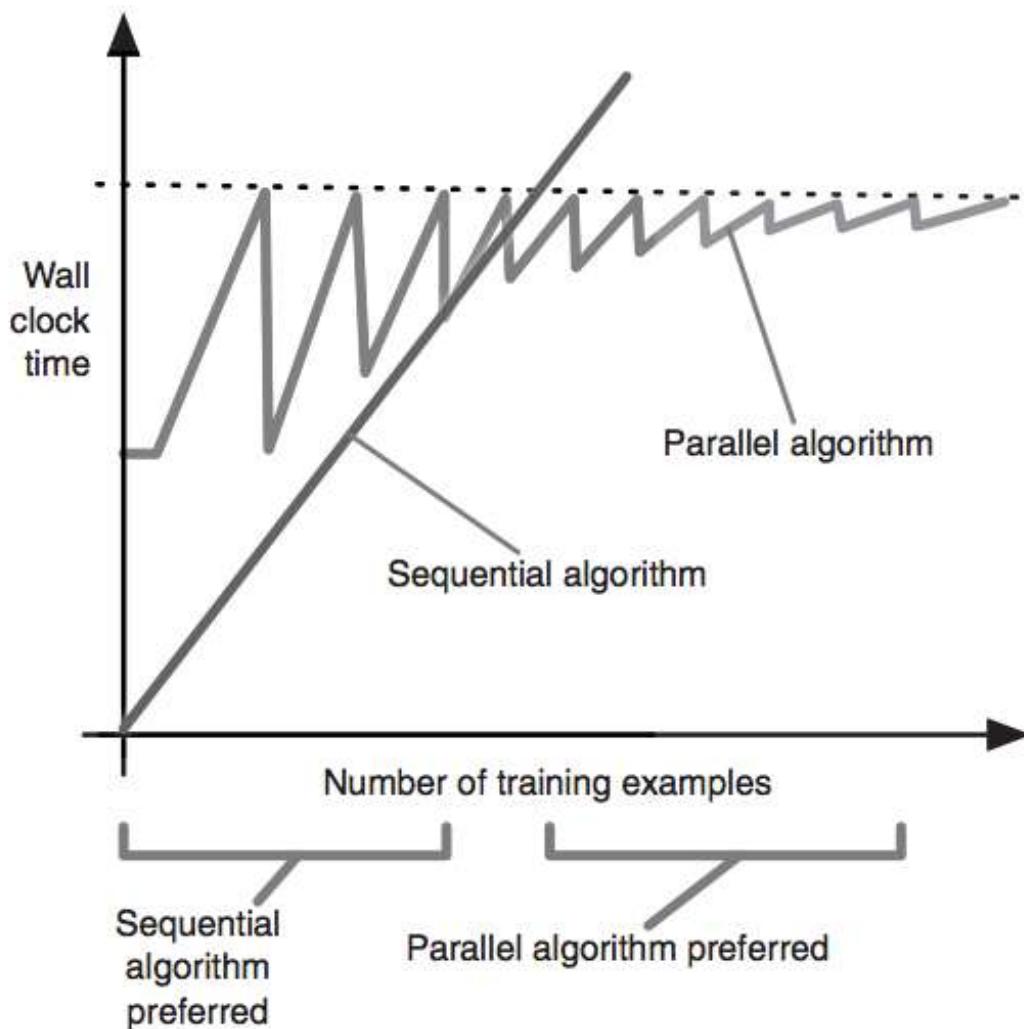
## Classification Example 2 — Color-Fill (another feature)



Mahout classification algorithms include:

- Naive Bayesian
- Complementary Naive Bayesian
- Stochastic Gradient Descent (SGD)
- Random Forest

# Comparing two types of Mahout Scalable algorithms

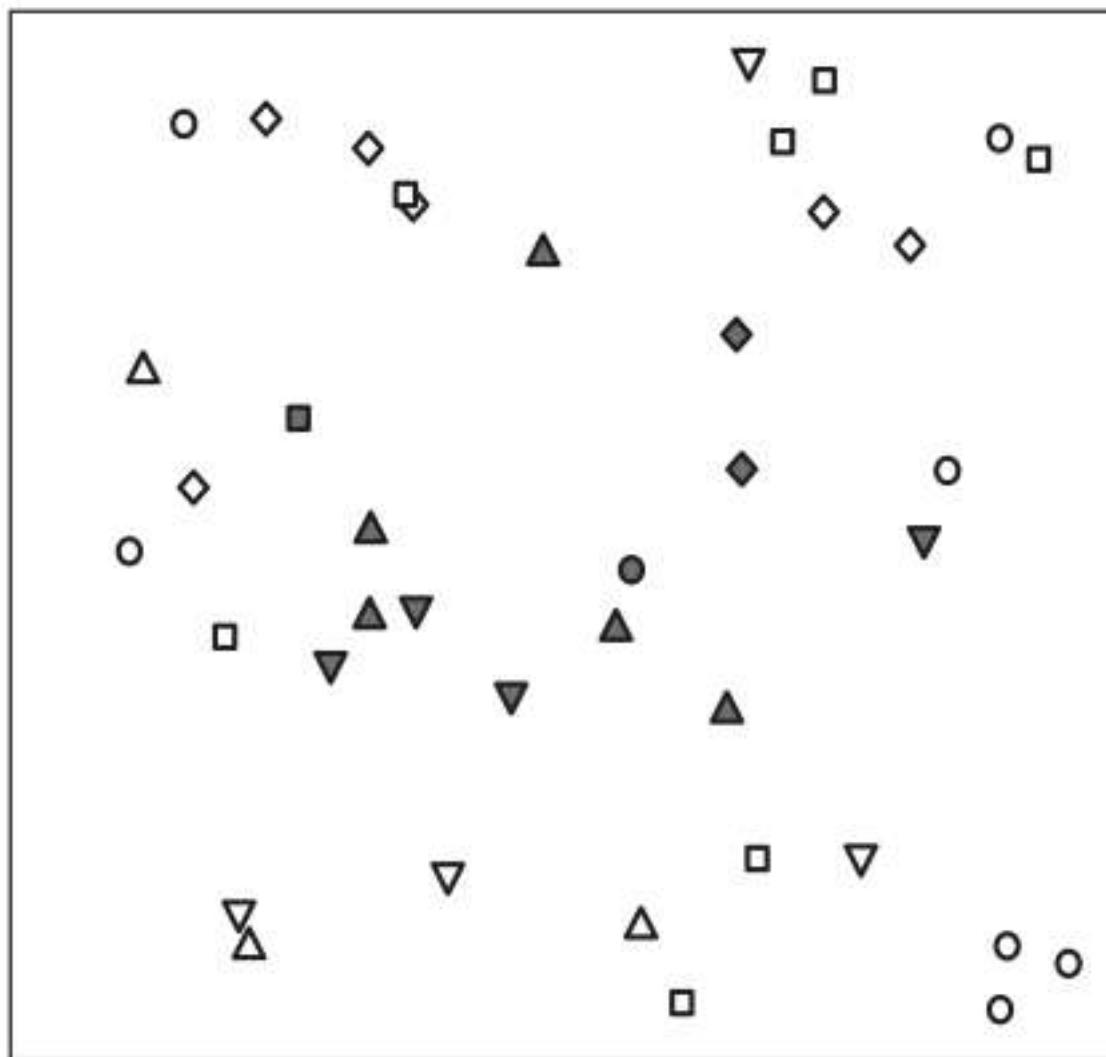


# Step-by-step simple classification example

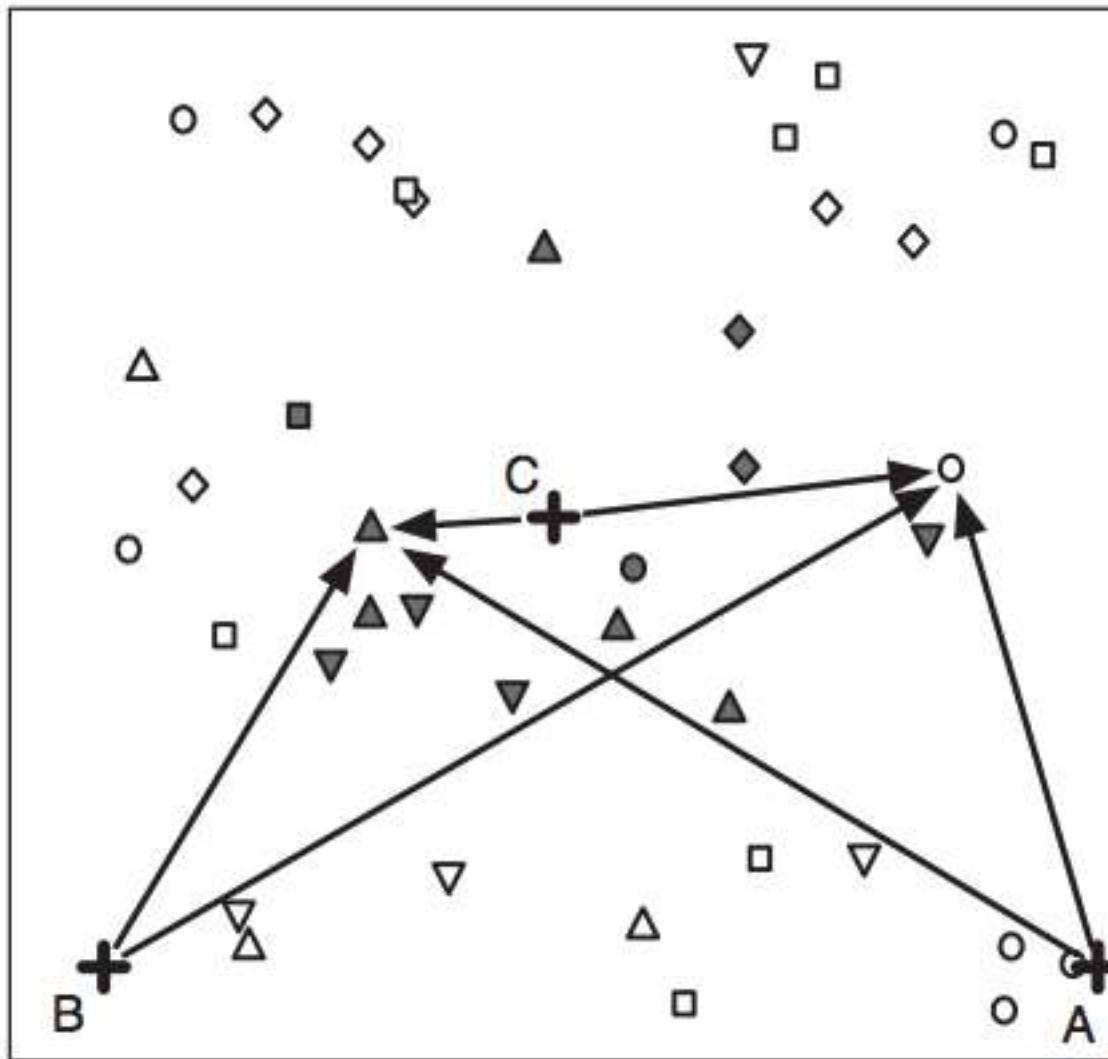
---

1. The data and the challenge
2. Training a model to find color-fill: preliminary thinking
3. Choosing a learning algorithm to train the model
4. Improving performance of the classifier

## Classification Example 3



# What may be a good predictor?



# Choose algorithm via Mahout

| Size of data set                                                           | Mahout algorithm                                                                                                          | Execution model                       | Characteristics                                                                                                                                                                                                                           |
|----------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|---------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Small to medium<br>(less than tens of millions of training examples)       | Stochastic gradient descent (SGD) family:<br>OnlineLogisticRegression,<br>CrossFoldLearner,<br>AdaptiveLogisticRegression | Sequential,<br>online,<br>incremental | Uses all types of predictor variables; sleek and efficient over the appropriate data range (up to millions of training examples)                                                                                                          |
|                                                                            | Support vector machine (SVM)                                                                                              | Sequential                            | Experimental still; sleek and efficient over the appropriate data range                                                                                                                                                                   |
| Medium to large<br>(millions to hundreds of millions of training examples) | Naive Bayes                                                                                                               | Parallel                              | Strongly prefers text-like data; medium to high overhead for training; effective and useful for data sets too large for SGD or SVM                                                                                                        |
|                                                                            | Complementary naive Bayes                                                                                                 | Parallel                              | Somewhat more expensive to train than naive Bayes; effective and useful for data sets too large for SGD, but has similar limitations to naive Bayes                                                                                       |
| Small to medium<br>(less than tens of millions of training examples)       | Random forests                                                                                                            | Parallel                              | Uses all types of predictor variables; high overhead for training; not widely used (yet); costly but offers complex and interesting classifications, handles nonlinear and conditional relationships in data better than other techniques |

# Stochastic Gradient Descent (SGD)

Both statistical estimation and machine learning consider the problem of minimizing an objective function that has the form of a sum:

$$Q(w) = \sum_{i=1}^n Q_i(w),$$

where the parameter  $w$  is to be estimated and where typically each summand function  $Q_i()$  is associated with the  $i$ -th observation in the data set (used for training).

- Choose an initial vector of parameters  $w$  and learning rate  $\alpha$ .
- Randomly shuffle examples in the training set.
- Repeat until an approximate minimum is obtained:
  - For  $i = 1, 2, \dots, n$ , do:
    - $w := w - \alpha \nabla Q_i(w)$ .

Let's suppose we want to fit a straight line  $y = w_1 + w_2x$  to a training set of two-dimensional points  $(x_1, y_1), \dots, (x_n, y_n)$  using least squares. The objective function to be minimized is:

$$Q(w) = \sum_{i=1}^n Q_i(w) = \sum_{i=1}^n (w_1 + w_2x_i - y_i)^2.$$

The last line in the above pseudocode for this specific problem will become:

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} := \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} - \alpha \left[ \begin{array}{c} \sum_{i=1}^n 2(w_1 + w_2x_i - y_i) \\ \sum_{i=1}^n 2x_i(w_1 + w_2x_i - y_i) \end{array} \right].$$

# Characteristic of SGD

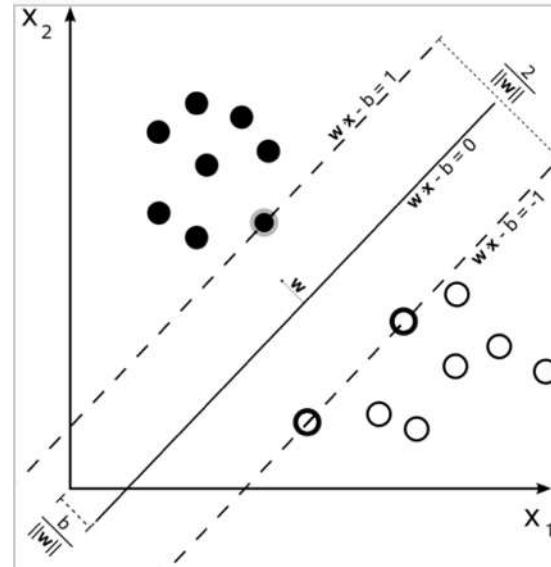
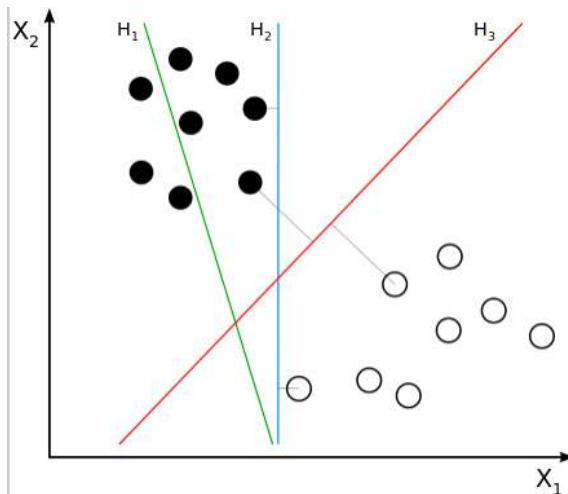
---

## THE SGD ALGORITHM

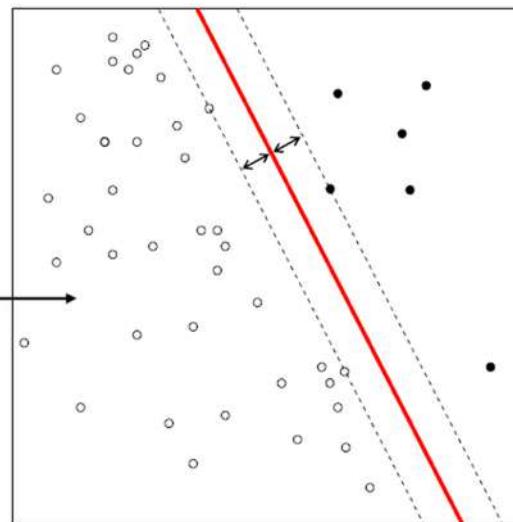
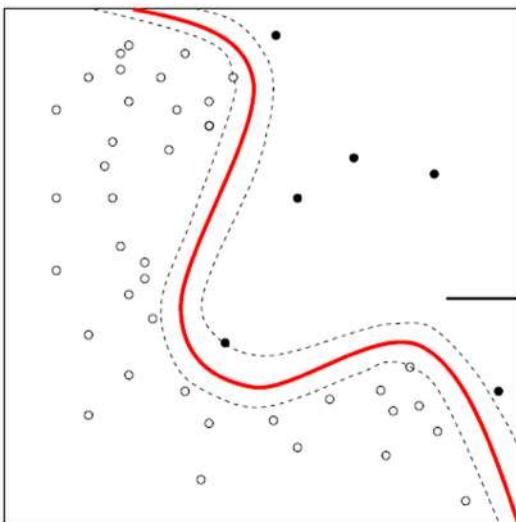
Stochastic gradient descent (SGD) is a widely used learning algorithm in which each training example is used to tweak the model slightly to give a more correct answer for that one example. This incremental approach is repeated over many training examples. With some special tricks to decide how much to nudge the model, the model accurately classifies new data after seeing only a modest number of examples. Although SGD algorithms are difficult to parallelize effectively, they're often so fast that for a wide variety of applications, parallel execution isn't necessary.

Importantly, because these algorithms do the same simple operation for each training example, they require a constant amount of memory. For this reason, each training example requires roughly the same amount of work. These properties make SGD-based algorithms scalable in the sense that twice as much data takes only twice as long to process.

# Support Vector Machine (SVM)



maximize boundary distances; remembering “support vectors”



nonlinear kernels

# Naive Bayes

Training set:

| sex    | height (feet) | weight (lbs) | foot size(inches) |
|--------|---------------|--------------|-------------------|
| male   | 6             | 180          | 12                |
| male   | 5.92 (5'11")  | 190          | 11                |
| male   | 5.58 (5'7")   | 170          | 12                |
| male   | 5.92 (5'11")  | 165          | 10                |
| female | 5             | 100          | 6                 |
| female | 5.5 (5'6")    | 150          | 8                 |
| female | 5.42 (5'5")   | 130          | 7                 |
| female | 5.75 (5'9")   | 150          | 9                 |

Classifier using Gaussian distribution assumptions:

| sex    | mean (height) | variance (height) | mean (weight) | variance (weight) | mean (foot size) | variance (foot size) |
|--------|---------------|-------------------|---------------|-------------------|------------------|----------------------|
| male   | 5.855         | 3.5033e-02        | 176.25        | 1.2292e+02        | 11.25            | 9.1667e-01           |
| female | 5.4175        | 9.7225e-02        | 132.5         | 5.5833e+02        | 7.5              | 1.6667e+00           |

Test Set:

| sex    | height (feet) | weight (lbs) | foot size(inches) |
|--------|---------------|--------------|-------------------|
| sample | 6             | 130          | 8                 |

$$posterior(male) = \frac{P(male) p(\text{height|male}) p(\text{weight|male}) p(\text{footsize|male})}{\text{evidence}}$$

$$\text{evidence} = P(\text{male}) p(\text{height|male}) p(\text{weight|male}) p(\text{footsize|male}) + P(\text{female}) p(\text{height|female}) p(\text{weight|female}) p(\text{footsize|female})$$

$$P(\text{male}) = 0.5$$

$$p(\text{height|male}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(6 - \mu)^2}{2\sigma^2}\right) \approx 1.5789$$

$$p(\text{weight|male}) = 5.9881 \cdot 10^{-6}$$

$$p(\text{foot size|male}) = 1.3112 \cdot 10^{-3}$$

$$\text{posterior numerator (male)} = \text{their product} = 6.1984 \cdot 10^{-9}$$

$$P(\text{female}) = 0.5$$

$$p(\text{height|female}) = 2.2346 \cdot 10^{-1}$$

$$p(\text{weight|female}) = 1.6789 \cdot 10^{-2}$$

$$p(\text{foot size|female}) = 2.8669 \cdot 10^{-1}$$

$$\text{posterior numerator (female)} = \text{their product} = 5.3778 \cdot 10^{-4}$$

==> female

# Random Forest

**Random forests** are an [ensemble learning](#) method for [classification](#) (and [regression](#)) that operate by constructing a multitude of [decision trees](#) at training time and outputting the class that is the [mode](#) of the classes output by individual trees.

The training algorithm for random forests applies the general technique of [bootstrap aggregating](#), or bagging, to tree learners. Given a training set  $X = x_1, \dots, x_n$  with responses  $Y = y_1$  through  $y_n$ , bagging repeatedly selects a [bootstrap sample](#) of the training set and fits trees to these samples:

For  $b = 1$  through  $B$ :

1. Sample, with replacement,  $n$  training examples from  $X, Y$ ; call these  $X_b, Y_b$ .
2. Train a decision or regression tree  $f_b$  on  $X_b, Y_b$ .

After training, predictions for unseen samples  $x'$  can be made by averaging the predictions from all the individual regression trees on  $x'$ :

$$\hat{f} = \frac{1}{B} \sum_{b=1}^B \hat{f}_b(x')$$

or by taking the majority vote in the case of decision trees.

Random forest uses a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features.

# Choosing a learning algorithm to train the model

- One low overhead classification method is the stochastic gradient descent (SGD) algorithm for logistic regression.
- This algorithm is sequential, but it's fast.

## START RUNNING MAHOUT

```
$ $MAHOUT_HOME/bin/mahout
An example program must be given as the first argument.
Valid program names are:
canopy: : Canopy clustering
cat : Print a file or resource as the logistic regression models would see
      it
...
runlogistic : Run a logistic regression model against CSV data
...
trainlogistic : Train a logistic regression using stochastic gradient
      descent
...
```

## CHECK MAHOUT'S BUILT-IN DATA

```
$ bin/mahout cat donut.csv
"x", "y", "shape", "color", "k", "k0", "xx", "xy", "yy", "a", "b", "c", "bias"
0.923307513352484, 0.0135197141207755, 21, 2, 4, 8, 0.852496764213146, ..., 1
0.711011884035543, 0.909141522599384, 22, 2, 3, 9, 0.505537899239772, ..., 1
...
0.67132937326096, 0.571220482233912, 23, 1, 5, 2, 0.450683127402953, ..., 1
0.548616112209857, 0.405350996181369, 24, 1, 5, 3, 0.300979638576258, ..., 1
0.677980388281867, 0.993355110753328, 25, 2, 3, 9, 0.459657406894831, ..., 1
$
```

## The donut.csv data file in Example 3

| Variable | Description                                          | Possible values                    |
|----------|------------------------------------------------------|------------------------------------|
| x        | The x coordinate of a point                          | Numerical from 0 to 1              |
| y        | The y coordinate of a point                          | Numerical from 0 to 1              |
| shape    | The shape of a point                                 | Shape code from 21 to 25           |
| color    | Whether the point is filled or not                   | 1=empty, 2=filled                  |
| k        | The k-means cluster ID derived using only x and y    | Integer cluster code from 1 to 10  |
| k0       | The k-means cluster ID derived using x, y, and color | Integer cluster code from 1 to 10  |
| xx       | The square of the x coordinate                       | Numerical from 0 to 1              |
| xy       | The product of the x and y coordinates               | Numerical from 0 to 1              |
| yy       | The square of the y coordinate                       | Numerical from 0 to 1              |
| a        | The distance from the point (0,0)                    | Numerical from 0 to $\sqrt{2}$     |
| b        | The distance from the point (1,0)                    | Numerical from 0 to $\sqrt{2}$     |
| c        | The distance from the point (0.5,0.5)                | Numerical from 0 to $(\sqrt{2})/2$ |
| bias     | A constant                                           | 1                                  |

# Build a model using Mahout

```
$ bin/mahout trainlogistic --input donut.csv \
    --output ./model \
    --target color --categories 2 \
    --predictors x y --types numeric \
    --features 20 --passes 100 --rate 50
...
color ~ -0.157*Intercept Term + -0.678*x + -0.416*y
Intercept Term -0.15655
    x -0.67841
    y -0.41587
...
```

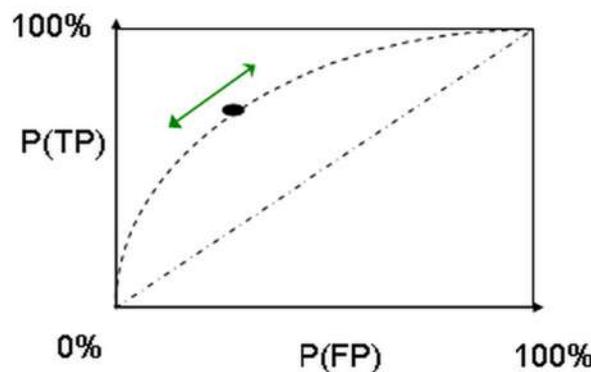
This command specifies that the input comes from the resource named `donut.csv`, that the resulting model is stored in the file `./model`, that the target variable is in the field named `color` and that it has two possible values. The command also specifies that the algorithm should use variables `x` and `y` as predictors, both with numerical types. The remaining options specify internal parameters for the learning algorithm.

# Trainlogistic program

| Option                     | What It does                                                                                                                                                                                                                                                      |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| --quiet                    | Produces less status and progress output.                                                                                                                                                                                                                         |
| --input <file-or-resource> | Uses the specified file or resource as input.                                                                                                                                                                                                                     |
| --output <file-for-model>  | Puts the model into the specified file.                                                                                                                                                                                                                           |
| --target <variable>        | Uses the specified variable as the target.                                                                                                                                                                                                                        |
| --categories <n>           | Specifies how many categories the target variable has.                                                                                                                                                                                                            |
| --predictors <v1> ... <vn> | Specifies the names of the predictor variables.                                                                                                                                                                                                                   |
| --types <t1> ... <tm>      | Gives a list of the types of the predictor variables. Each type should be one of numeric, word, or text. Types can be abbreviated to their first letter. If too few types are given, the last one is used again as necessary. Use word for categorical variables. |
| --passes                   | Specifies the number of times the input data should be re-examined during training. Small input files may need to be examined dozens of times. Very large input files probably don't even need to be completely examined.                                         |
| --lambda                   | Controls how much the algorithm tries to eliminate variables from the final model. A value of 0 indicates no effort is made. Typical values are on the order of 0.00001 or less.                                                                                  |
| --rate                     | Sets the initial learning rate. This can be large if you have lots of data or use lots of passes because it's decreased progressively as data is examined.                                                                                                        |
| --noBias                   | Eliminates the intercept term (a built-in constant predictor variable) from the model. Occasionally this is a good idea, but generally it isn't since the SGD learning algorithm can usually eliminate the intercept term if warranted.                           |
| --features                 | Sets the size of the internal feature vector to use in building the model. A larger value here can be helpful, especially with text-like input data.                                                                                                              |

# Evaluate the model

```
$ bin/mahout runlogistic --input donut.csv --model ./model \
    --auc --confusion
AUC = 0.57
confusion: [[27.0, 13.0], [0.0, 0.0]]
...
```



AUC (0 ~ 1):  
 1 — perfect  
 0 — perfectly wrong  
 0.5 — random

|                                          |                                         |
|------------------------------------------|-----------------------------------------|
| <b>True positive</b>                     | <b>False positive</b><br>(Type I error) |
| <b>False negative</b><br>(Type II error) | <b>True negative</b>                    |

confusion matrix

| Option          | What It does                                                          |
|-----------------|-----------------------------------------------------------------------|
| --quiet         | Produces less status and progress output.                             |
| --auc           | Prints AUC score for model versus input data after reading data.      |
| --scores        | Prints target variable value and scores for each input example.       |
| --confusion     | Prints confusion matrix for a particular threshold (see --threshold). |
| --input <input> | Reads data records from specified file or resource.                   |
| --model <model> | Reads model from specified file.                                      |

# Questions?

# MLlib: Scalable Machine Learning on Spark

Xiangrui Meng



Collaborators: Ameet Talwalkar, Evan Sparks, Virginia Smith, Xinghao Pan, Shivaram Venkataraman, Matei Zaharia, Rean Griffith, John Duchi, Joseph Gonzalez, Michael Franklin, Michael I. Jordan, Tim Kraska, etc.

# What is MLlib?

# What is MLlib?

MLlib is a Spark subproject providing machine learning primitives:

- initial contribution from AMPLab, UC Berkeley
- shipped with Spark since version 0.8
- 33 contributors

# What is MLlib?

## Algorithms:

- **classification:** logistic regression, linear support vector machine (SVM), naive Bayes
- **regression:** generalized linear regression (GLM)
- **collaborative filtering:** alternating least squares (ALS)
- **clustering:** k-means
- **decomposition:** singular value decomposition (SVD), principal component analysis (PCA)

# Why MLlib?

# scikit-learn?

## Algorithms:

- **classification:** SVM, nearest neighbors, random forest, ...
- **regression:** support vector regression (SVR), ridge regression, Lasso, logistic regression, ...
- **clustering:** k-means, spectral clustering, ...
- **decomposition:** PCA, non-negative matrix factorization (NMF), independent component analysis (ICA), ...

# Mahout?

## Algorithms:

- **classification:** logistic regression, naive Bayes, random forest, ...
- **collaborative filtering:** ALS, ...
- **clustering:** k-means, fuzzy k-means, ...
- **decomposition:** SVD, randomized SVD, ...

Mahout?

**LIBLINEAR?**

H<sub>2</sub>O?

Vowpal Wabbit?

**MATLAB?**

R?

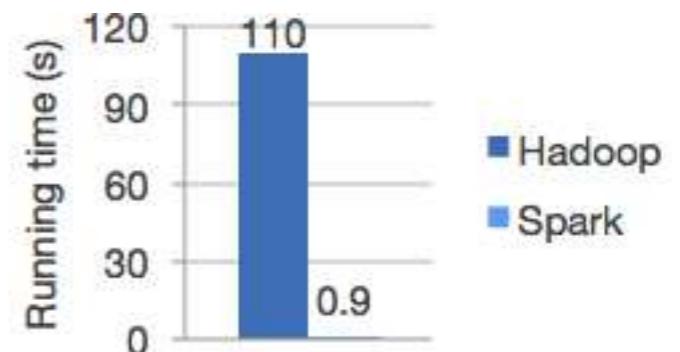
scikit-learn?

Weka?

# Why MLlib?

# Why MLlib?

- It is built on Apache Spark, a fast and general engine for large-scale data processing.
  - Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.
  - Write applications quickly in Java, Scala, or Python.



# Gradient descent

$$w \leftarrow w - \alpha \cdot \sum_{i=1}^n g(w; x_i, y_i)$$

```
val points = spark.textFile(...).map(parsePoint).cache()
var w = Vector.zeros(d)
for (i <- 1 to numIterations) {
    val gradient = points.map { p =>
        (1 / (1 + exp(-p.y * w.dot(p.x))) - 1) * p.y * p.x
    }.reduce(_ + _)
    w -= alpha * gradient
}
```

# k-means (scala)

```
// Load and parse the data.  
val data = sc.textFile("kmeans_data.txt")  
val parsedData = data.map(_.split(' ')).map(_.toDouble)).cache()  
  
// Cluster the data into two classes using KMeans.  
val clusters = KMeans.train(parsedData, 2, numIterations = 20)  
  
// Compute the sum of squared errors.  
val cost = clusters.computeCost(parsedData)  
println("Sum of squared errors = " + cost)
```

# k-means (python)

```
# Load and parse the data
data = sc.textFile("kmeans_data.txt")
parsedData = data.map(lambda line:
                      array([float(x) for x in line.split(' ')])).cache()

# Build the model (cluster the data)
clusters = KMeans.train(parsedData, 2, maxIterations = 10,
                        runs = 1, initialization_mode = "kmeans||")

# Evaluate clustering by computing the sum of squared errors
def error(point):
    center = clusters.centers[clusters.predict(point)]
    return sqrt(sum([x**2 for x in (point - center)]))

cost = parsedData.map(lambda point: error(point))
    .reduce(lambda x, y: x + y)
print("Sum of squared error = " + str(cost))
```

# Dimension reduction + k-means

```
// compute principal components
val points: RDD[Vector] = ...
val mat = RowRDDMatrix(points)
val pc = mat.computePrincipalComponents(20)

// project points to a low-dimensional space
val projected = mat.multiply(pc).rows

// train a k-means model on the projected data
val model = KMeans.train(projected, 10)
```

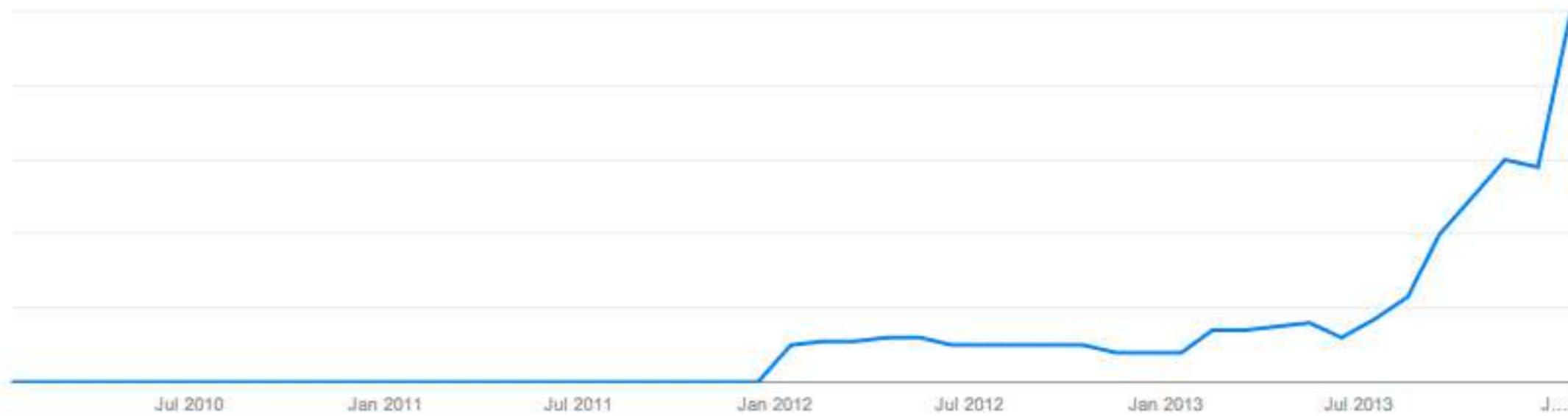
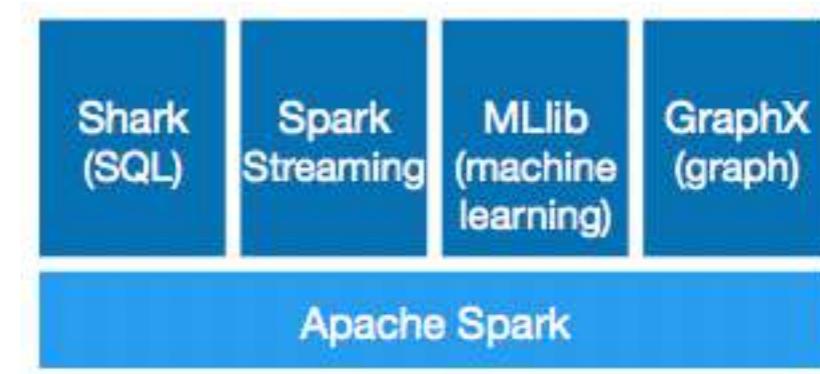
# Collaborative filtering

```
// Load and parse the data
val data = sc.textFile("mllib/data/als/test.data")
val ratings = data.map(_.split(',') match {
    case Array(user, item, rate) =>
        Rating(user.toInt, item.toInt, rate.toDouble)
})
// Build the recommendation model using ALS
val model = ALS.train(ratings, 1, 20, 0.01)

// Evaluate the model on rating data
val usersProducts = ratings.map { case Rating(user, product, rate) =>
    (user, product)
}
val predictions = model.predict(usersProducts)
```

# Why MLlib?

- It ships with Spark as a standard component.



## **Out for dinner?**

- Search for a restaurant and make a reservation.
- Start navigation.
- Food looks good? Take a photo and share.

# Why smartphone?

## Out for dinner?

- Search for a restaurant and make a reservation. (~~Yellow Pages?~~)
- Start navigation. (~~GPS?~~)
- Food looks good? Take a photo and share. (~~Camera?~~)

# Why MLlib?

A special-purpose device may be better at one aspect than a general-purpose device. But the cost of context switching is high:

- different languages or APIs
- different data formats
- different tuning tricks

# Spark SQL + MLlib

```
// Data can easily be extracted from existing sources,  
// such as Apache Hive.  
val trainingTable = sql(  
    "SELECT e.action,  
        u.age,  
        u.latitude,  
        u.longitude  
    FROM Users u  
    JOIN Events e  
    ON u.userId = e.userId")  
  
// Since `sql` returns an RDD, the results of the above  
// query can be easily used in MLlib.  
val training = trainingTable.map { row =>  
    val features = Vectors.dense(row(1), row(2), row(3))  
    LabeledPoint(row(0), features)  
}  
  
val model = SVMWithSGD.train(training)
```

# Streaming + MLlib

```
// collect tweets using streaming  
  
// train a k-means model  
val model: KMmeansModel = ...  
  
// apply model to filter tweets  
val tweets = TwitterUtils.createStream(ssc, Some(authorizations(0)))  
val statuses = tweets.map(_.getText)  
val filteredTweets =  
    statuses.filter(t => model.predict(featurize(t)) == clusterNumber)  
  
// print tweets within this particular cluster  
filteredTweets.print()
```

# GraphX + MLlib

```
// assemble link graph
val graph = Graph(pages, links)
val pageRank: RDD[(Long, Double)] = graph.staticPageRank(10).vertices

// load page labels (spam or not) and content features
val labelAndFeatures: RDD[(Long, (Double, Seq((Int, Double))))] = ...
val training: RDD[LabeledPoint] =
  labelAndFeatures.join(pageRank).map {
    case (id, ((label, features), pageRank)) =>
      LabeledPoint(label, Vectors.sparse(features ++ (1000, pageRank)))
  }

// train a spam detector using logistic regression
val model = LogisticRegressionWithSGD.train(training)
```

# Why MLlib?

- Spark is a general-purpose big data platform.
  - Runs in standalone mode, on YARN, EC2, and Mesos, also on Hadoop v1 with SIMR.
  - Reads from HDFS, S3, HBase, and any Hadoop data source.
- MLlib is a standard component of Spark providing machine learning primitives on top of Spark.

# Why MLlib?

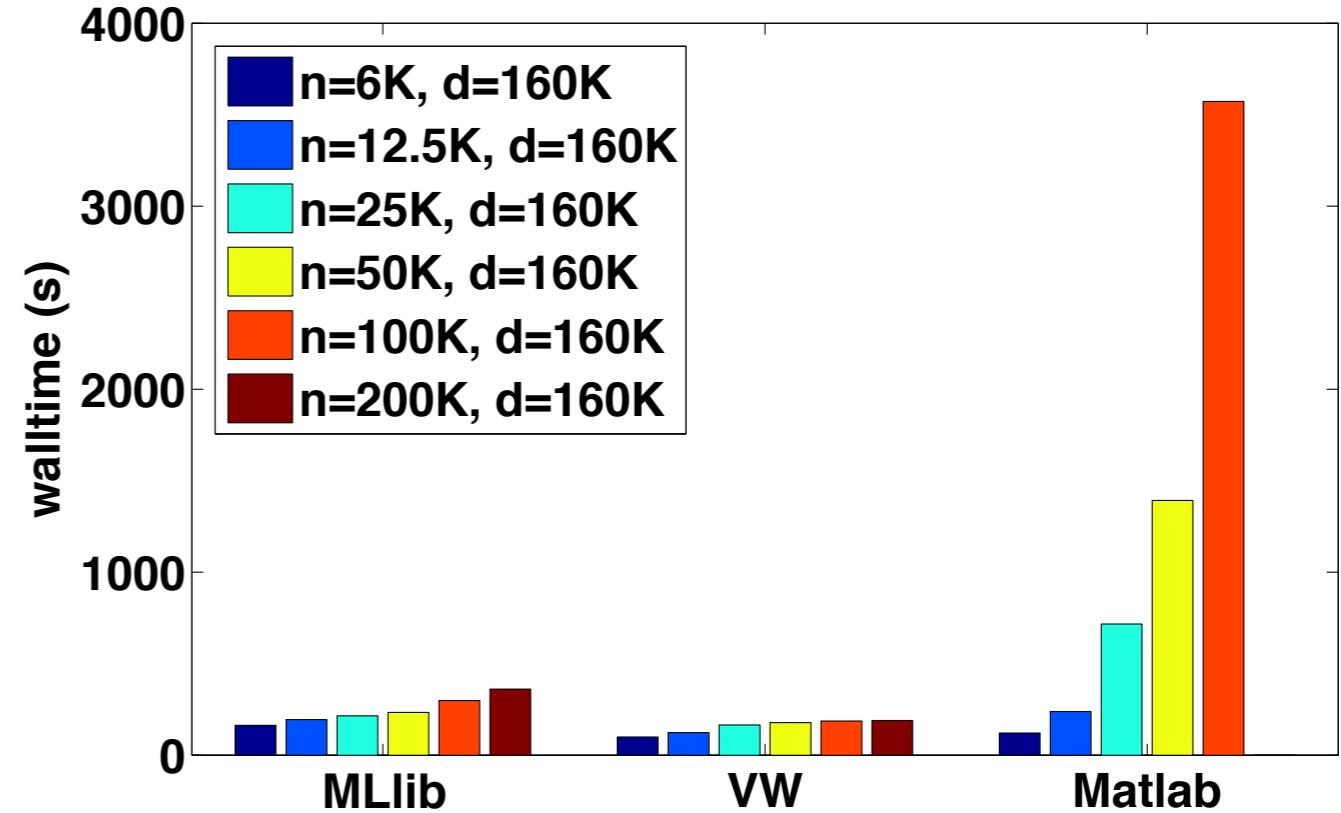
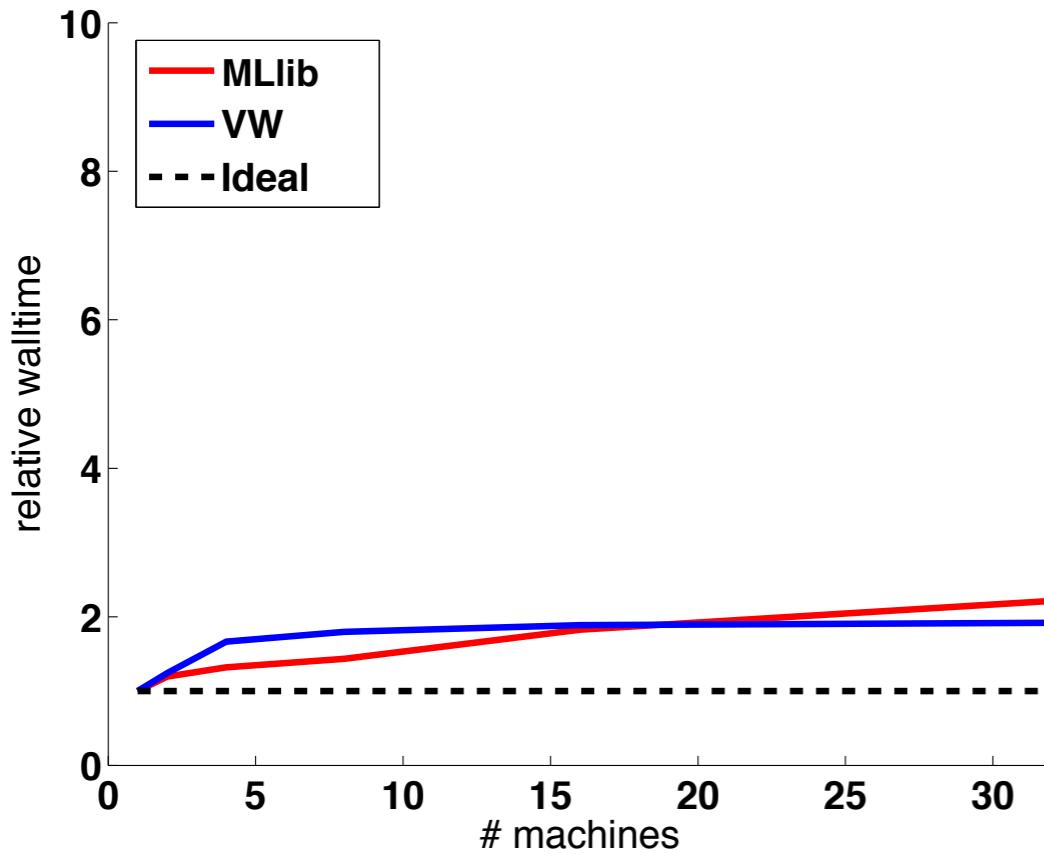
- Spark is a general-purpose big data platform.
  - Runs in standalone mode, on YARN, EC2, and Mesos, also on Hadoop v1 with SIMR.
  - Reads from HDFS, S3, HBase, and any Hadoop data source.
- MLlib is a standard component of Spark providing machine learning primitives on top of Spark.
- MLlib is also comparable to or even better than other libraries specialized in large-scale machine learning.

# Why MLlib?

- Scalability
- Performance
- User-friendly APIs
- Integration with Spark and its other components

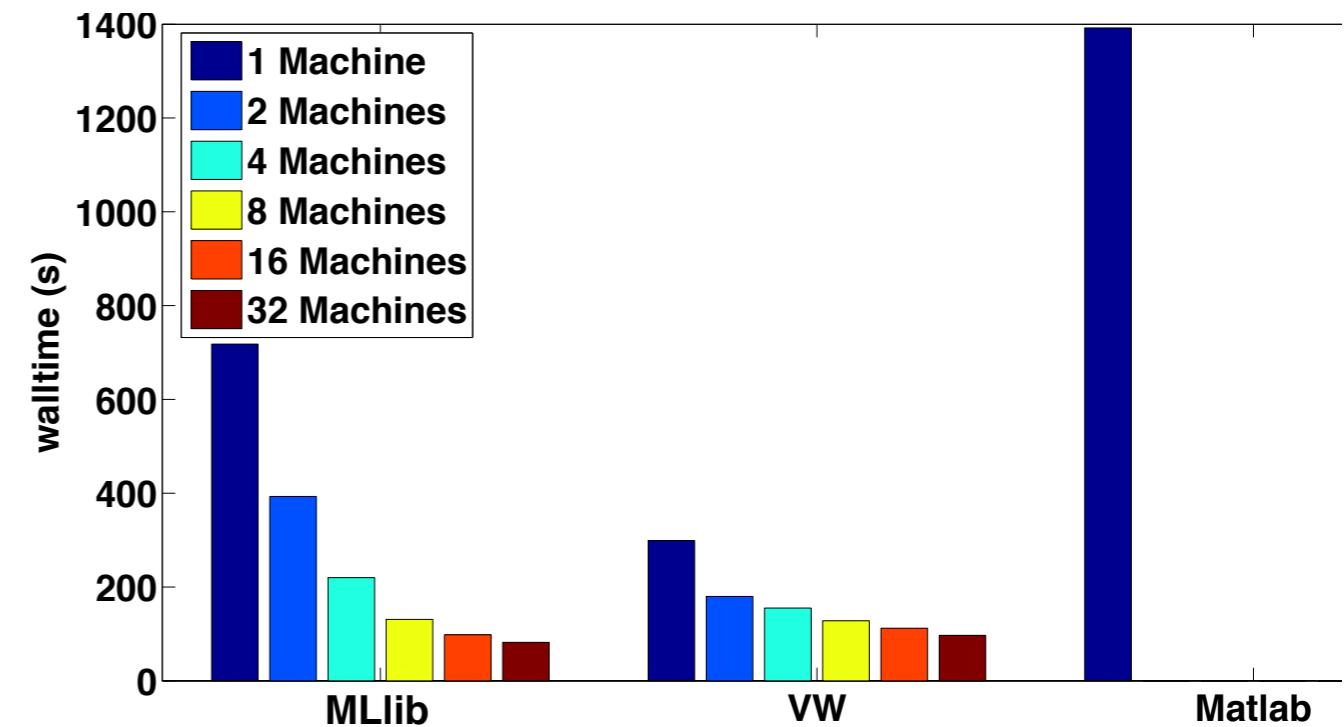
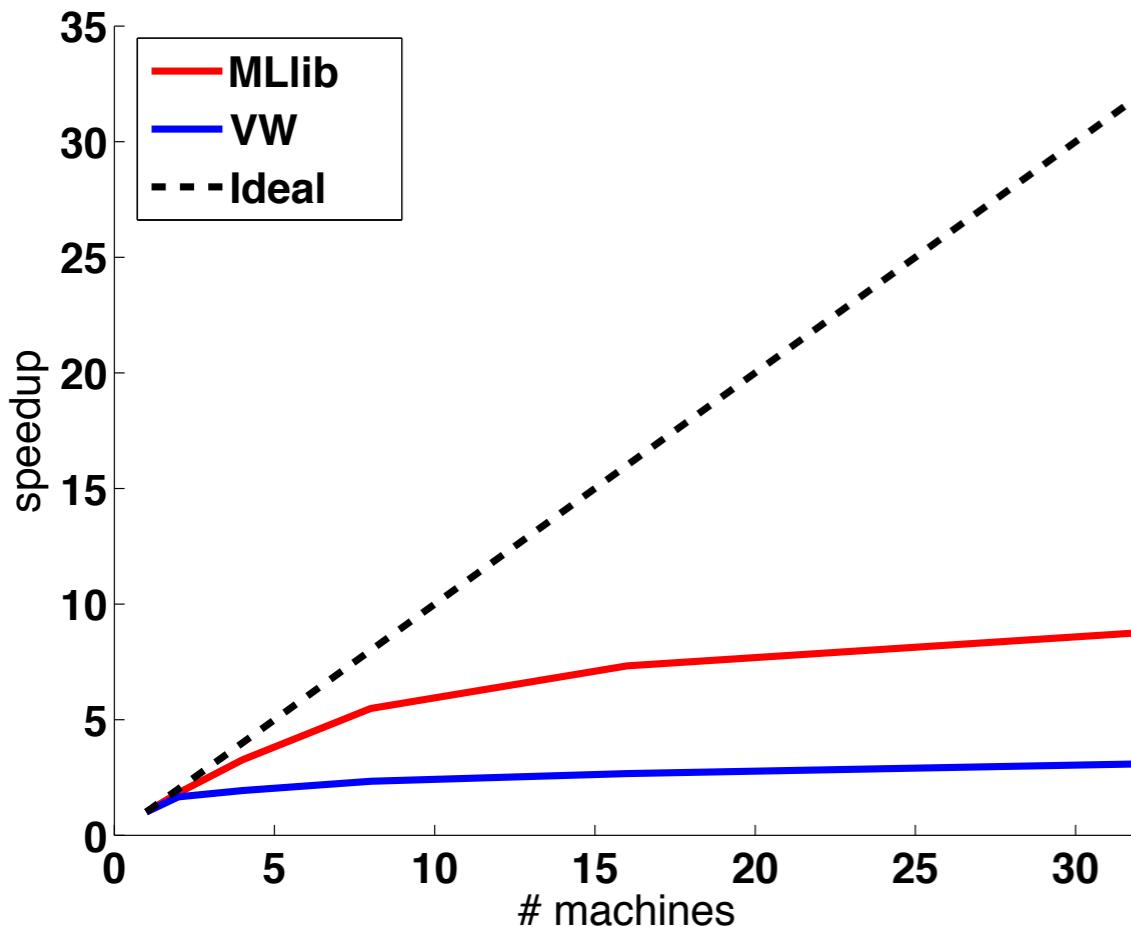
# Logistic regression

# Logistic regression - weak scaling



- Full dataset: 200K images, 160K dense features.
- Similar weak scaling.
- MLlib within a factor of 2 of VW's wall-clock time.

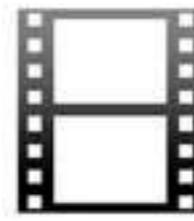
# Logistic regression - strong scaling



- Fixed Dataset: 50K images, 160K dense features.
- MLlib exhibits better scaling properties.
- MLlib is faster than VW with 16 and 32 machines.

# Collaborative filtering

# Collaborative filtering



|  |       |       |    |
|--|-------|-------|----|
|  | ★     | ★★★★★ | ?  |
|  | ★     | ★★★   | ★★ |
|  | ★★★★★ | ?     | ★  |
|  | ★     | ?     | ★★ |
|  | ?     | ★★★   | ★★ |
|  | ★★★★★ | ★★    | ?  |



- Recover a rating matrix from a subset of its entries.

last.fm

NETFLIX®

PANDORA®



Linked in

amazon.com®

# ALS - wall-clock time

| System   | Wall-clock time (seconds) |
|----------|---------------------------|
| MATLAB   | 15443                     |
| Mahout   | 4206                      |
| GraphLab | 291                       |
| MLlib    | 481                       |

- Dataset: scaled version of Netflix data (9X in size).
- Cluster: 9 machines.
- MLlib is an order of magnitude faster than Mahout.
- MLlib is within factor of 2 of GraphLab.

# Implementation of k-means

Initialization:

- random
- k-means++
- k-means||

# Implementation of k-means

Iterations:

- For each point, find its closest center.

$$l_i = \arg \min_j \|x_i - c_j\|_2^2$$

- Update cluster centers.

$$c_j = \frac{\sum_{i, l_i=j} x_j}{\sum_{i, l_i=j} 1}$$

# Implementation of k-means

The points are usually sparse, but the centers are most likely to be dense. Computing the distance takes  $O(d)$  time. So the time complexity is  $O(n d k)$  per iteration. We don't take any advantage of sparsity on the running time. However, we have

$$\|x - c\|_2^2 = \|x\|_2^2 + \|c\|_2^2 - 2\langle x, c \rangle$$

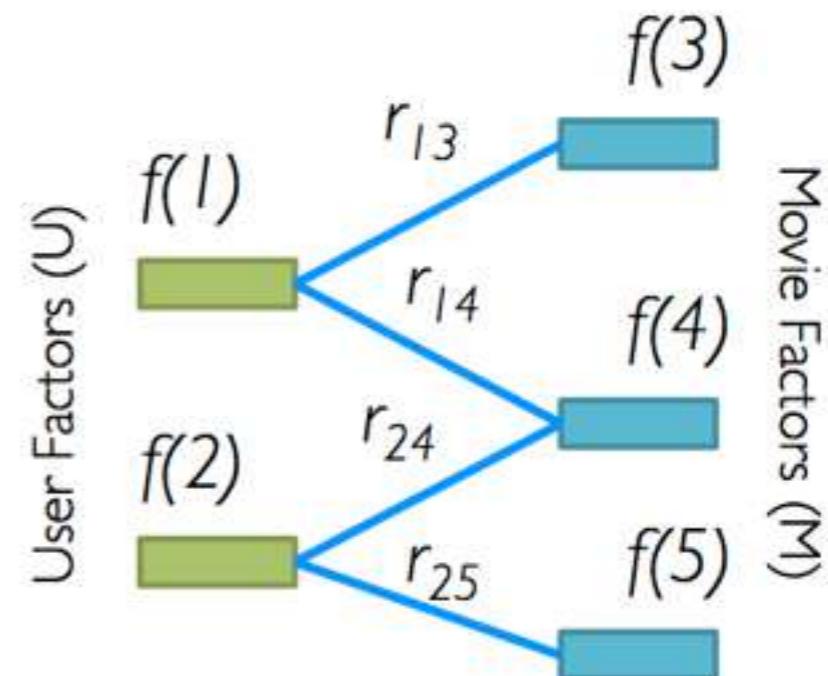
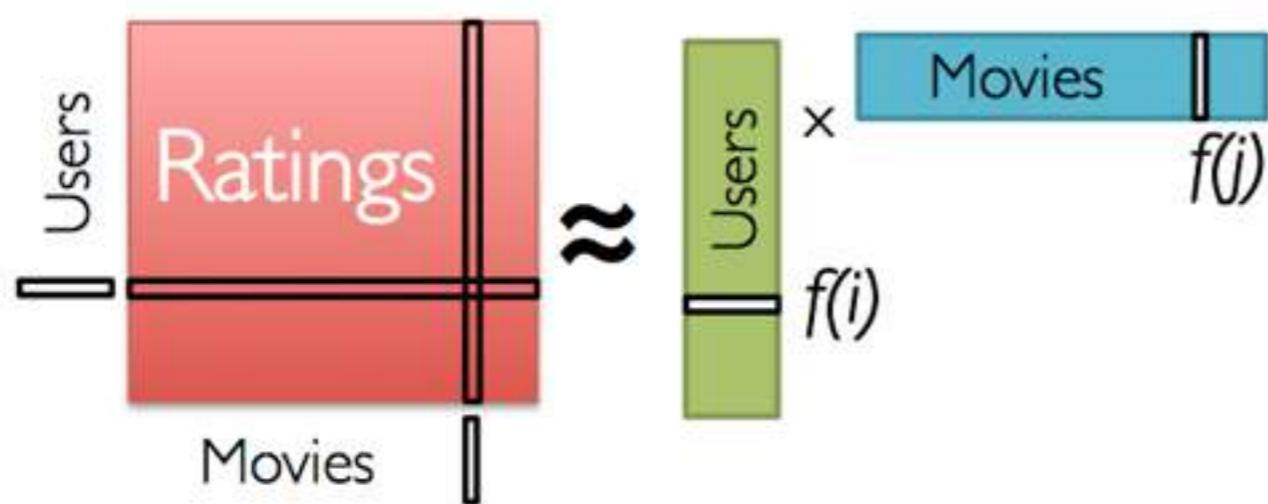
Computing the inner product only needs non-zero elements. So we can **cache** the norms of the points and of the centers, and then only need the inner products to obtain the distances. This reduce the running time to  $O(nnz k + d k)$  per iteration.

However, is it accurate?

# Implementation of ALS

- broadcast everything
- data parallel
- fully parallel

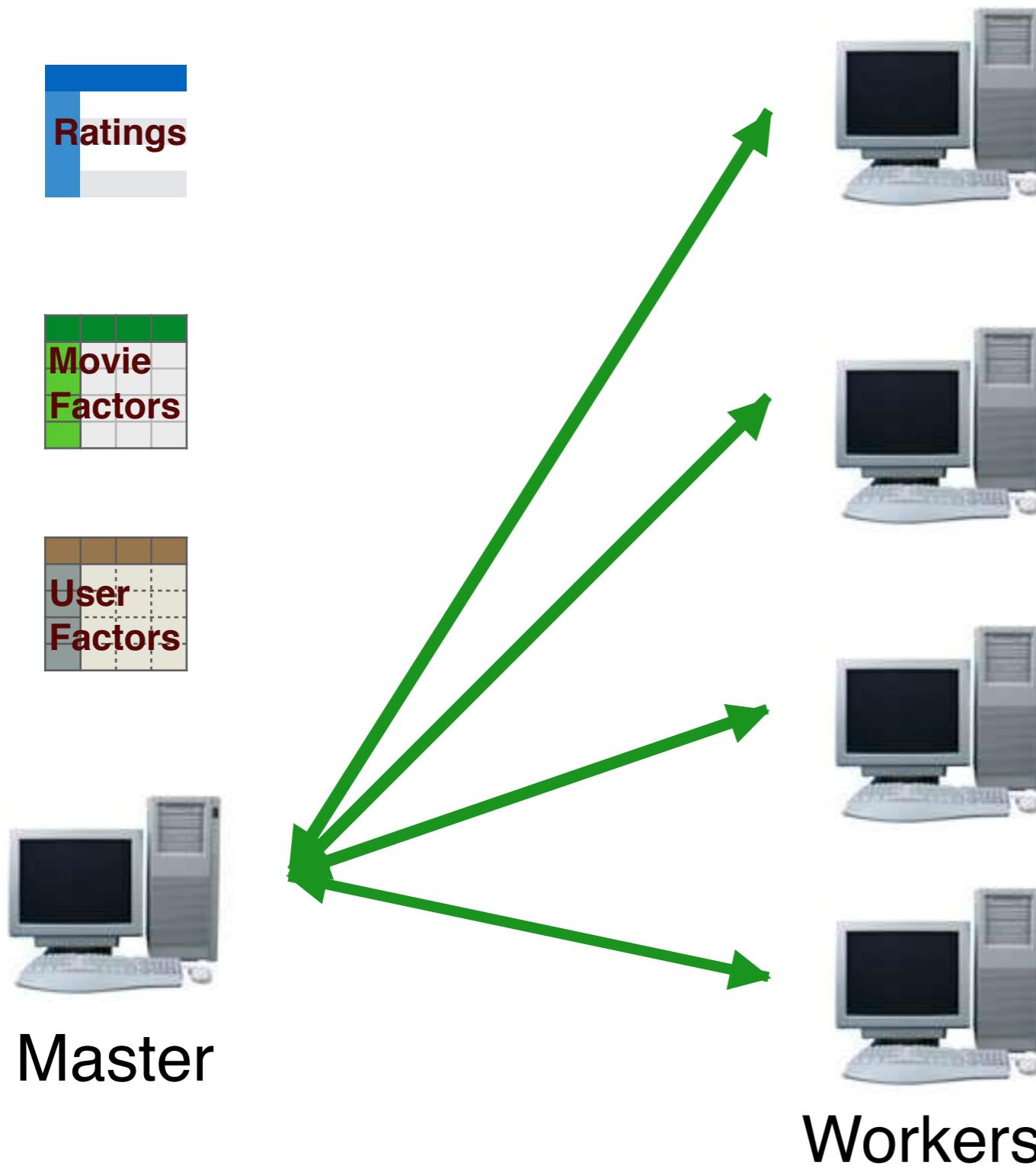
# Alternating least squares (ALS)



Iterate:

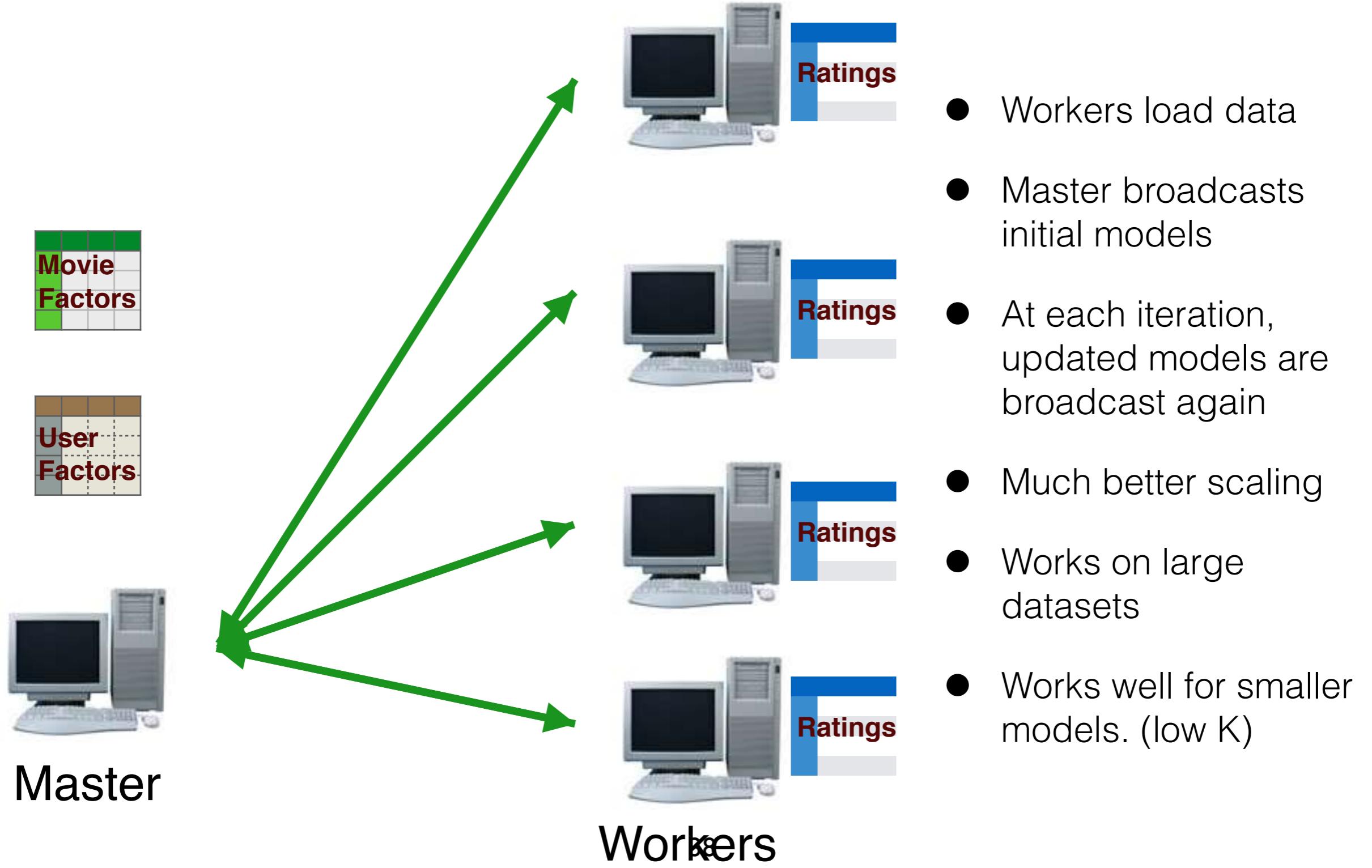
$$f[i] = \arg \min_{w \in \mathbb{R}^d} \sum_{j \in \text{Nbrs}(i)} (r_{ij} - w^T f[j])^2 + \lambda ||w||_2^2$$

# Broadcast everything

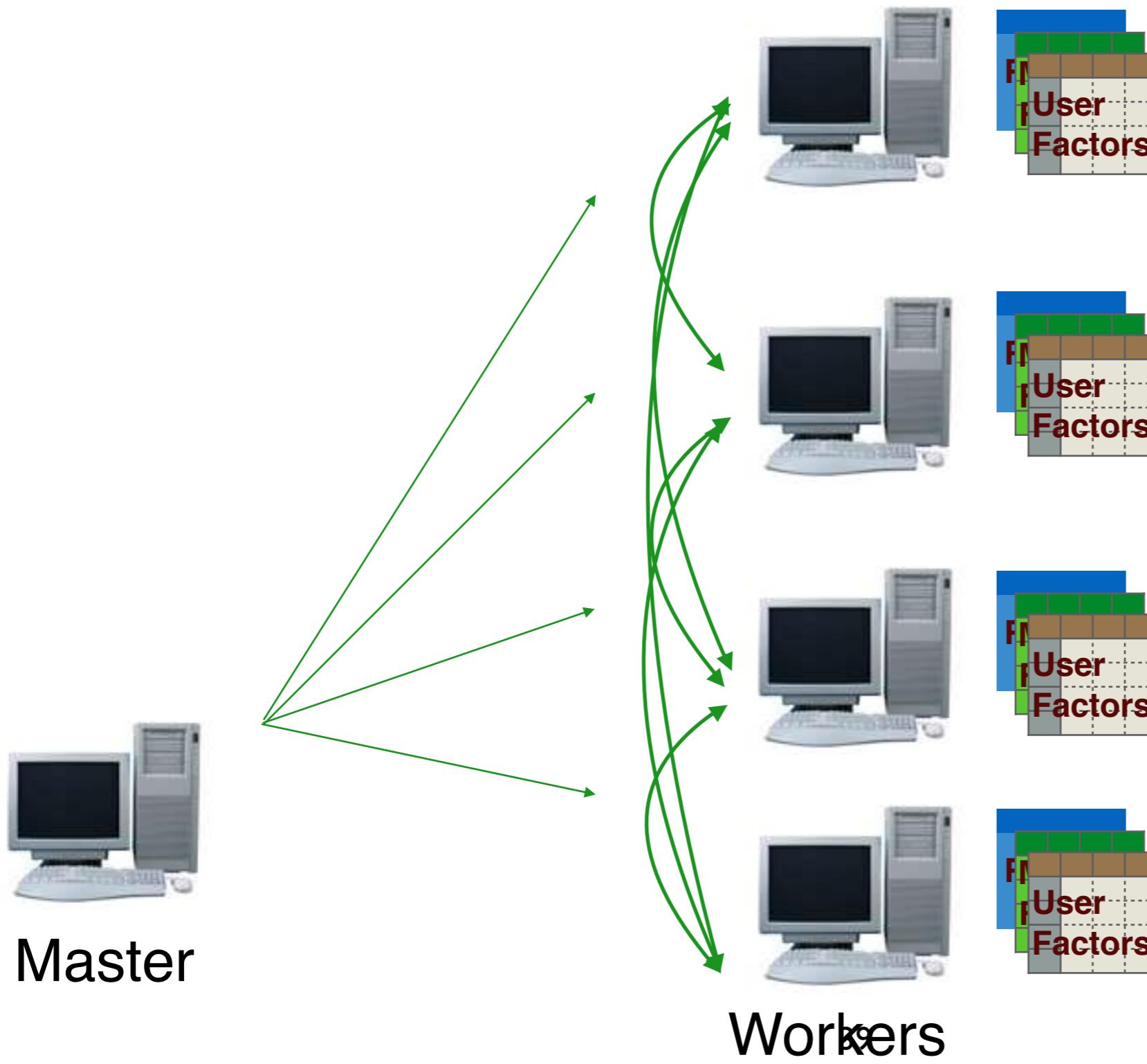


- Master loads (small) data file and initializes models.
- Master broadcasts data and initial models.
- At each iteration, updated models are broadcast again.
- Works OK for small data.
- Lots of communication overhead - doesn't scale well.

# Data parallel



# Fully parallel



- Workers load data
- Models are instantiated at workers.
- At each iteration, models are shared via join between workers.
- Much better scalability.
- Works on large datasets

# Implementation of ALS

- ~~broadcast everything~~
- ~~data parallel~~
- ~~fully parallel~~
- block-wise parallel
  - Users/products are partitioned into blocks and join is based on blocks instead of individual user/product.

# New features for v1.x

- Sparse data
- Classification and regression tree (CART)
- SVD and PCA
- L-BFGS
- Model evaluation
- Discretization

# Contributors

Ameet Talwalkar, Andrew Tulloch, Chen Chao, Nan Zhu, DB Tsai, Evan Sparks, Frank Dai, Ginger Smith, Henry Saputra, Holden Karau, Hossein Falaki, Jey Kottalam, Cheng Lian, Marek Kolodziej, Mark Hamstra, Martin Jaggi, Martin Weindel, Matei Zaharia, Nick Pentreath, Patrick Wendell, Prashant Sharma, Reynold Xin, Reza Zadeh, Sandy Ryza, Sean Owen, Shivararam Venkataraman, Tor Myklebust, Xiangrui Meng, Xinghao Pan, Xusen Yin, Jerry Shao, Ryan LeCompte

# Interested?

- Website: <http://spark.apache.org>
- Tutorials: <http://ampcamp.berkeley.edu>
- Spark Summit: <http://spark-summit.org>
- Github: <https://github.com/apache/spark>
- Mailing lists: [user@spark.apache.org](mailto:user@spark.apache.org)  
[dev@spark.apache.org](mailto:dev@spark.apache.org)



Your Connection to **ICT** Research

# ***Machine Learning With Spark***

***Ons Dridi***  
***R&D Engineer***

***13 Novembre 2015***

FEDER



LE FONDS EUROPÉEN DE DÉVELOPPEMENT RÉGIONAL  
ET LA WALLONIE INVESTISSENT DANS VOTRE AVENIR.

***Centre d'Excellence en Technologies de l'Information et  
de la Communication***

- An applied research centre in the field of ICT
- The knowledge developed by CETIC is made available to companies to help them integrate these technological breakthroughs into their products, processes and services, enabling them to innovate faster, save time and money and develop new markets.
- Three departments:
  - Software & System Engineering**
  - Software & Services Technologies**
  - Embedded & Communication Systems**



## Background:

- Engineering computer science degree
- Master degree in Computer Science and Mathematics applied to Finance and Insurance

## Mission:

- Big Data: data analysis

<https://www.cetic.be/Ons-Dridi>

- What is Spark ?

High level Architecture

How does it Work ?

RDD and Operations

Hadoop MapReduce

DAG (Directed Acyclic Graph)

Run Mode of Spark

Programming model

- Machine Learning With Spark

MLLib Library

Types of Machine Learning

ML Architecture

Comparison with other tools

- Other Applications



# What is Spark ?

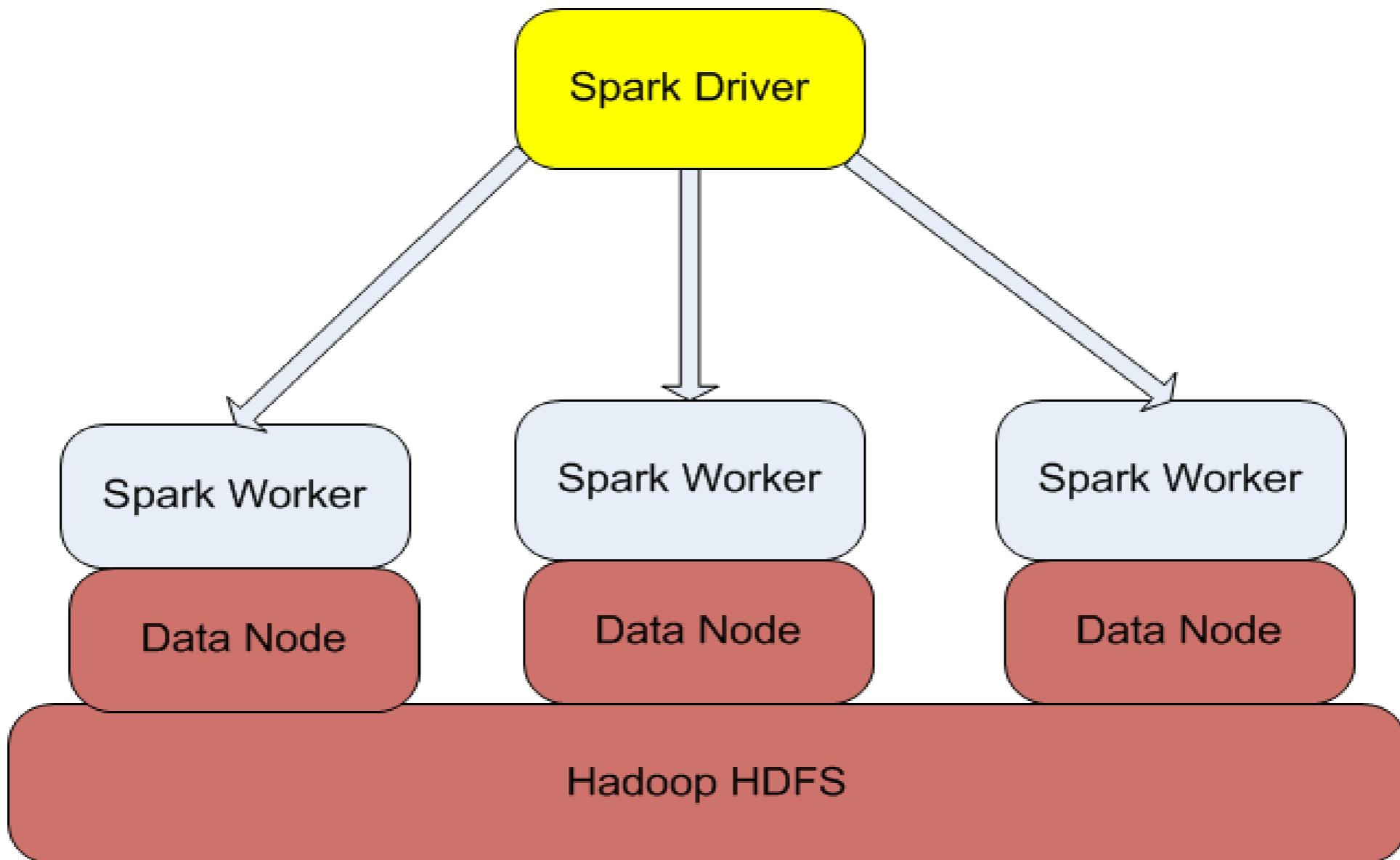
- ***Definition***

“Apache Spark is an open source big data processing framework built around speed, ease of use, and sophisticated analytics. It was originally developed in 2009 in UC Berkeley’s AMPLab, and open sourced in 2010 as an Apache project.”



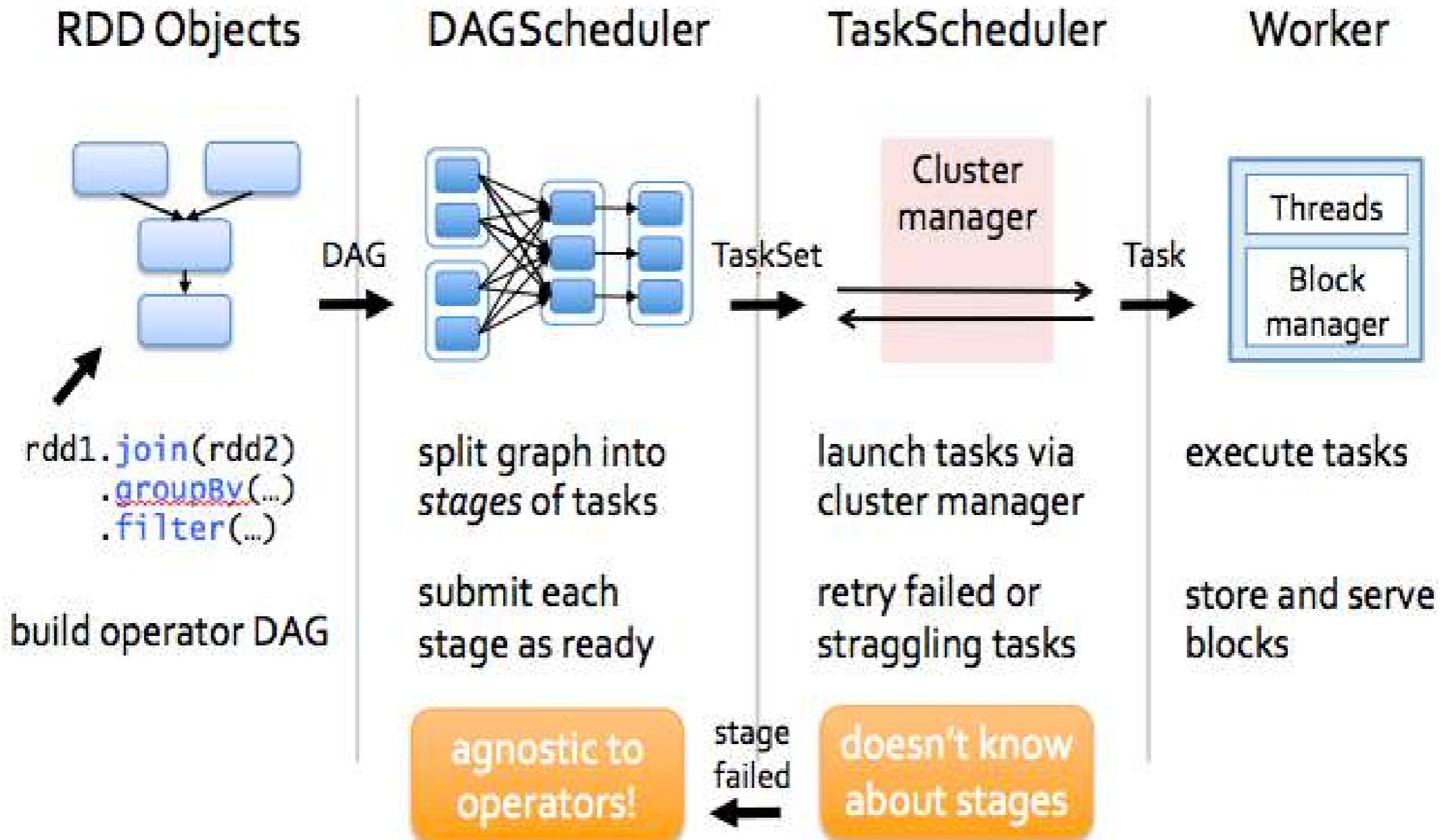
Source : <http://www.infoq.com/articles/apache-spark-introduction>

# High Level Architecture



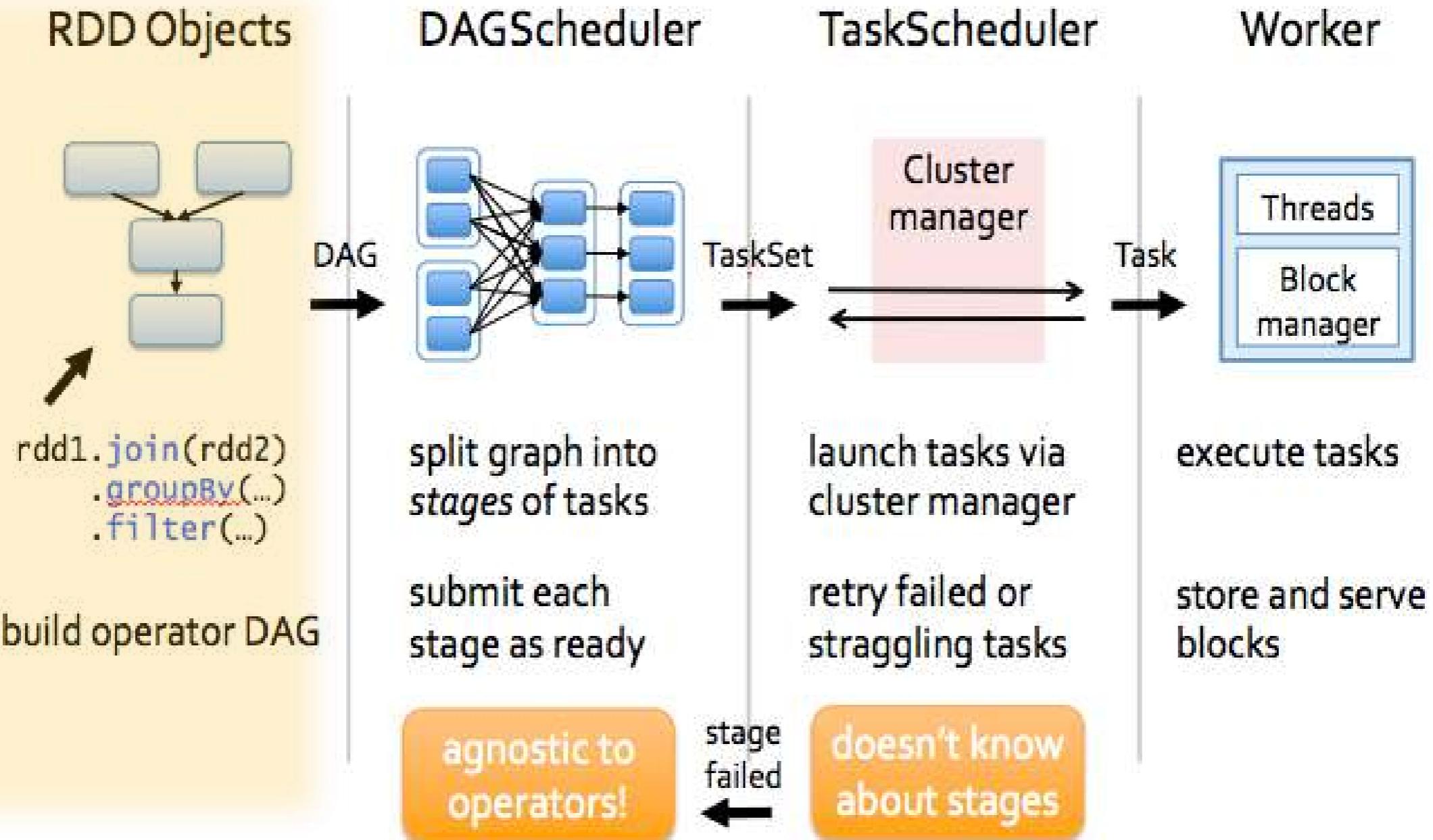
Source : <https://bighadoop.wordpress.com/2014/04/03/apache-spark-a-fast-big-data-analytics-engine/>

# How does it work ?



Source: <https://www.sigmoid.com/apache-spark-internals/>

# RDD and Operations



Source : <http://spark.apache.org/docs/0.6.2/api/core/spark/RDD.html>

- *Definition*

“A Resilient Distributed Dataset (RDD), the basic abstraction in Spark. Represents an immutable, partitioned collection of elements that can be operated on in parallel. This class contains the basic operations available on all RDD”

Source : <http://spark.apache.org/docs/0.6.2/api/core/spark/RDD.html>

- *Creating RDD:*

- *From existing collection*

```
val collection = List ("a", "b", "c", "d")
val rddFromCollection = sc.parallelize (collection)
```

- *From Hadoop-based input sources*

```
val rddFromTextFile= sc.textFile ("List ")
```

- **Operations:**

- **Transformations**

Apply functions to all the records, ex: map, filter, join

```
val sizeFromStringsRDD = rddFromFile.map (line => line.size)
```

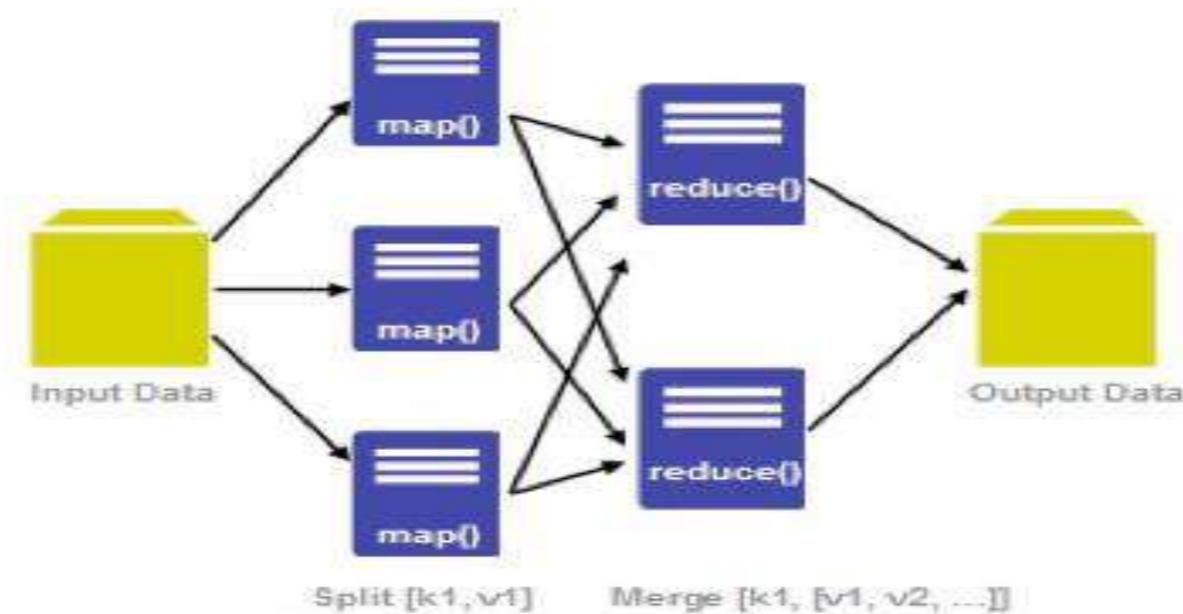
- **Actions**

Apply some computations and return the results to the driver program ,  
ex: reduce, count

```
sizeFromStringsRDD.count
```

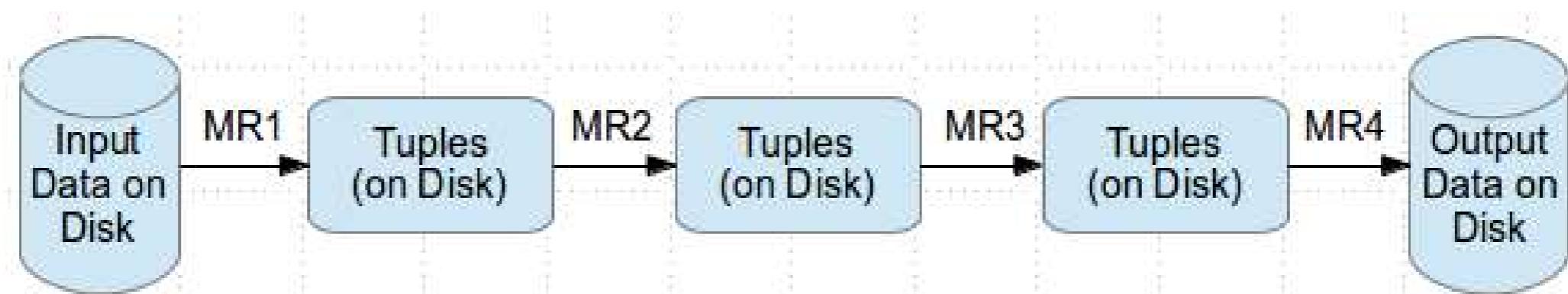
- *Definition*

“The term MapReduce actually refers to two separate and distinct tasks that Hadoop programs perform. The first is the map job ... and the second is the reduce.”

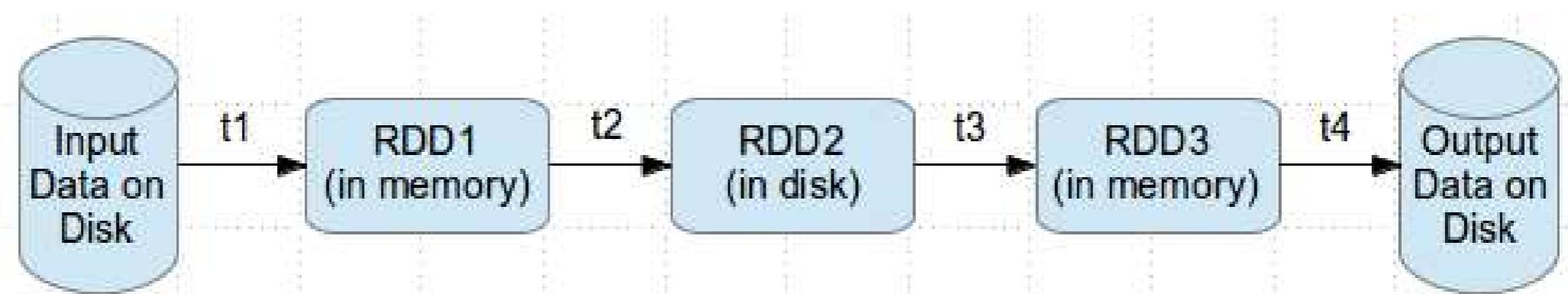


Source: <https://www-01.ibm.com/software/data/infosphere/hadoop/mapreduce/>

## *Case of MapReduce :*

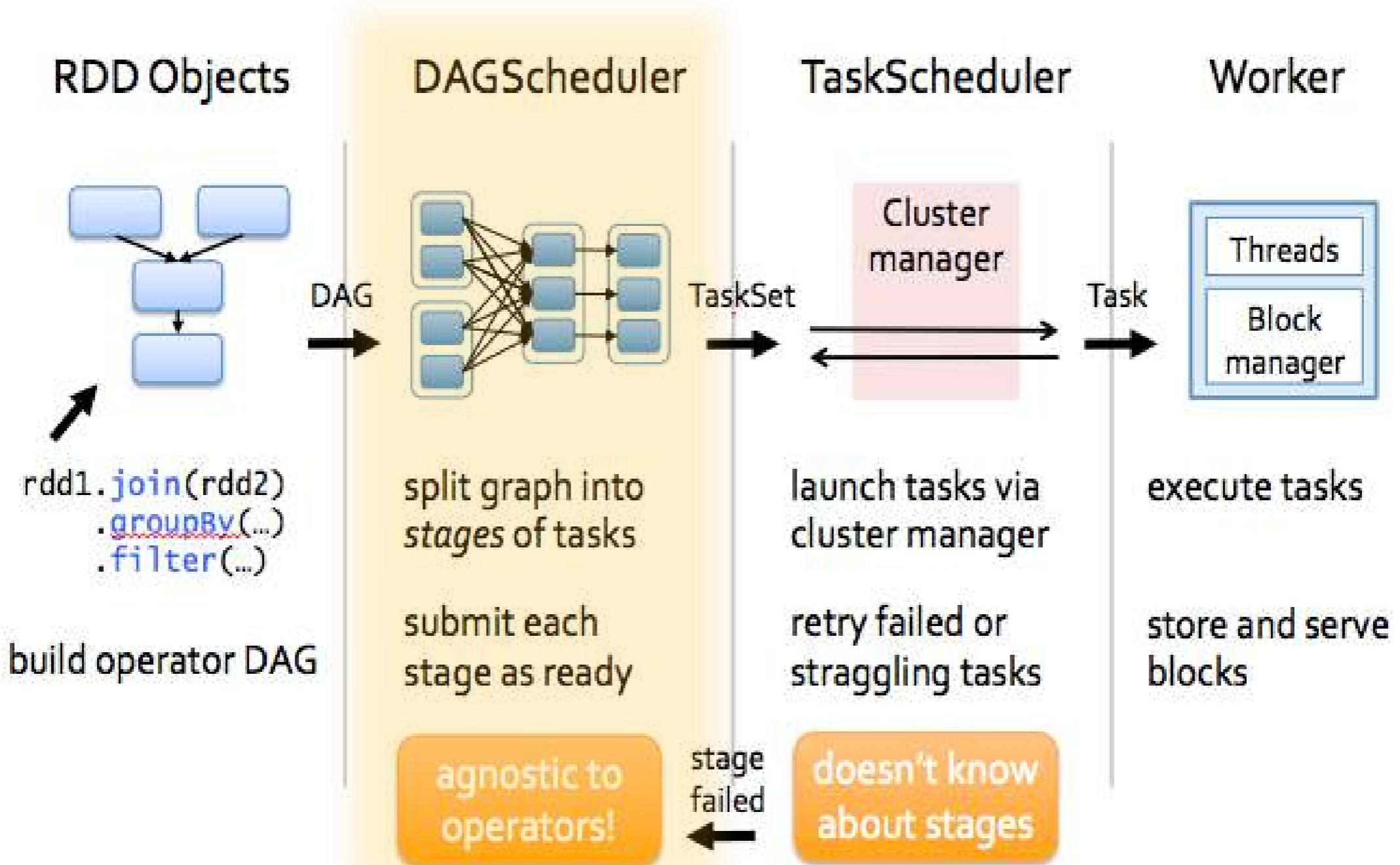


## *Case of RDD:*



Source: <http://www.thecloudavenue.com/>

# DAG (*Direct Acyclic Graph*)

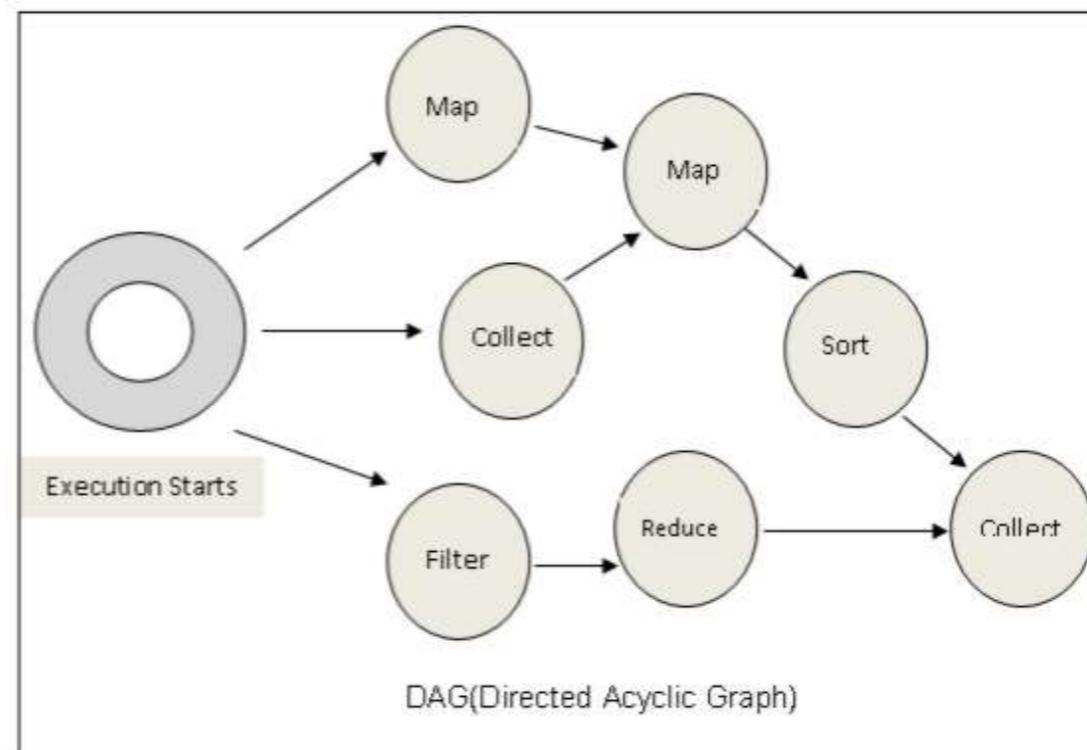


Source : <http://spark.apache.org/docs/0.6.2/api/core/spark/RDD.html>

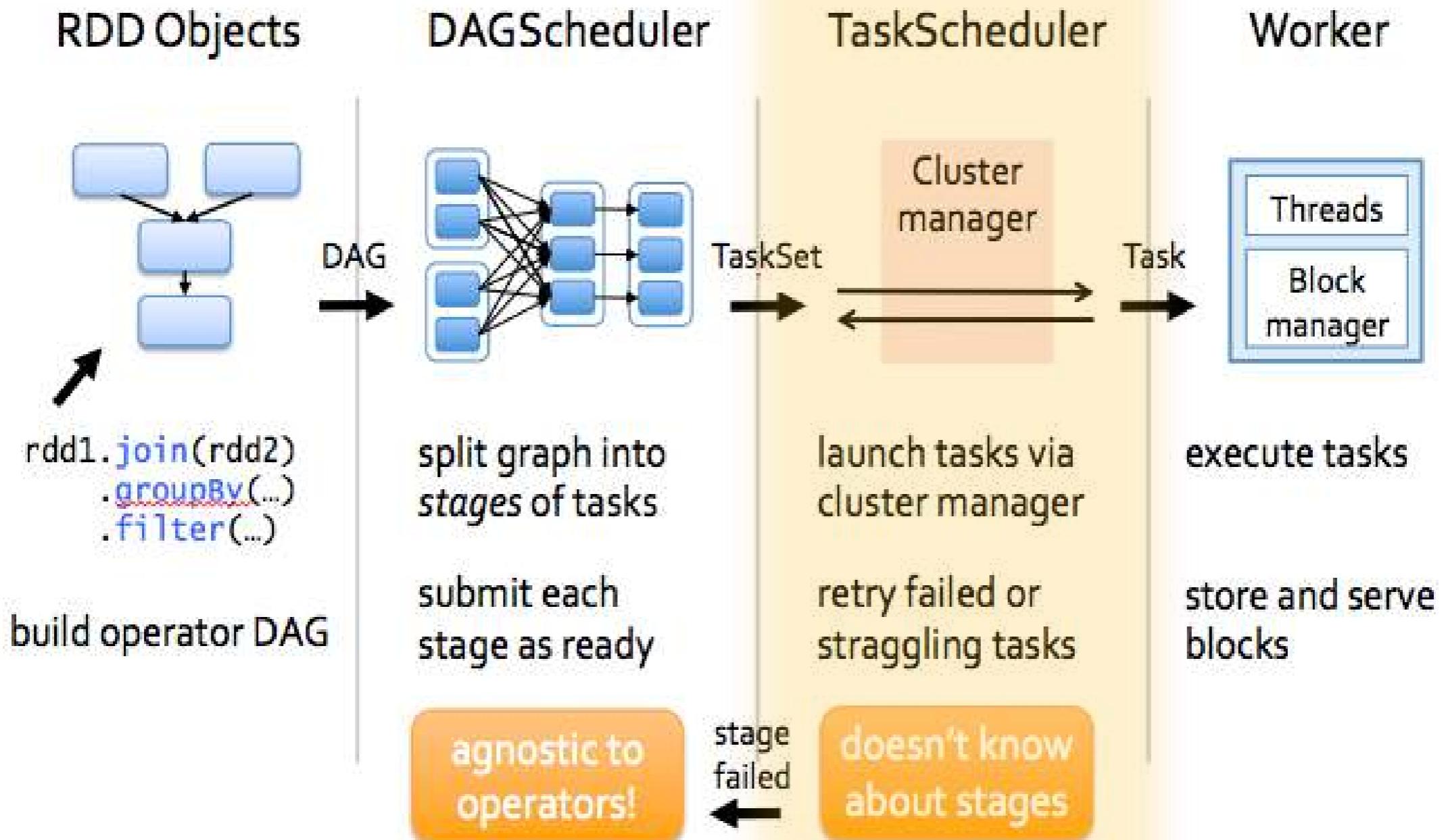
# DAG (*Directed Acyclic Graph*)

- ***Definition :***

“DAG stands for Directed Acyclic Graph, in the present context. It’s a DAG of operators. The DAG is optimized by rearranging and combining operators where possible”



# *Run mode of Spark*



Source : <http://spark.apache.org/docs/0.6.2/api/core/spark/RDD.html>

Spark runs in four modes:

- **Local mode:** run our application locally. This is really useful for debugging, we can step our code line by line with an IDE
- **Cluster Mode:**

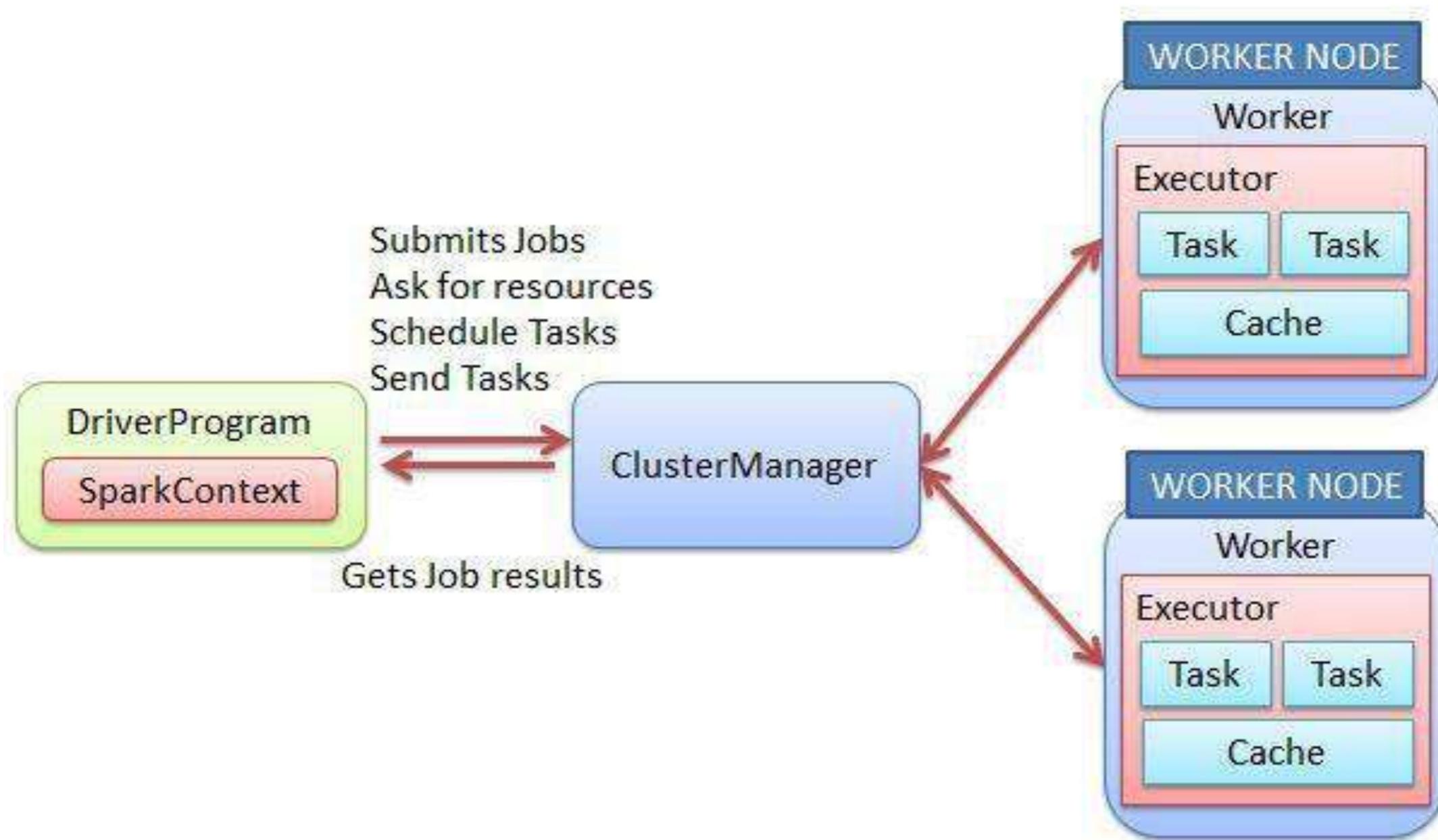
**Standalone mode:** we can easily deploy a standalone cluster with very few steps and configurations and then we can play around with it.

**Apache Mesos**

**Hadoop Yarn**

# Run mode of Spark

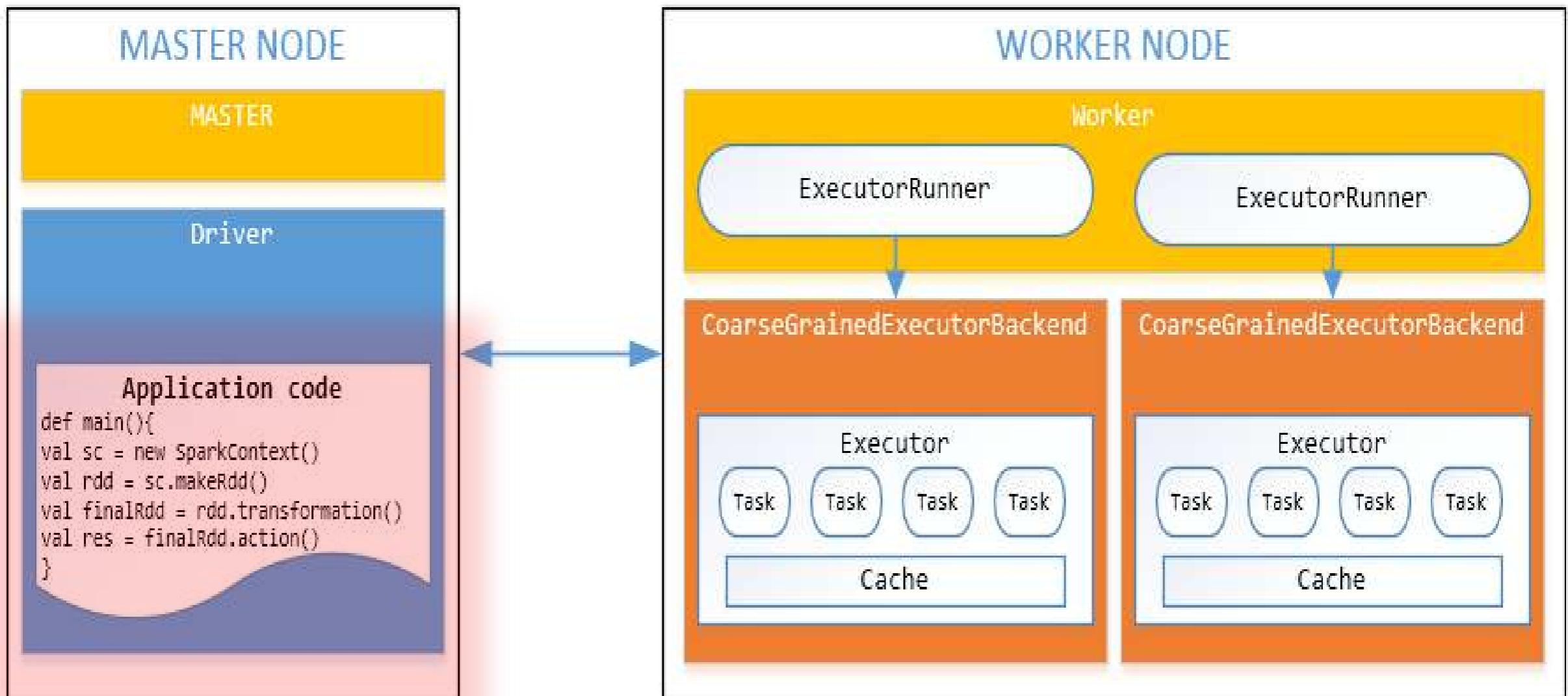
- **Cluster Mode:**



Source: <https://trongkhoanguyenblog.wordpress.com>

# Programming model

- **Master-Workers:**



Source: <https://trongkhoanguyenblog.wordpress.com>

- ***Spark Context and SparkConf:***

- The starting point of any spark program is *Spark Context*
- It's initialized with an instance of *SparkConf*
- Contains various methods to manipulate RDD

- ***Initialize SparkContext:***

```
val conf = new SparkConf()  
    .setAppName(" Test Spark ")  
    .setMaster(" local[4] ")  
val sc= new SparkContext(conf)
```

```
val sc = new SparkContext(" local[4] ", " Test Spark " )
```

- ***Definition :***

“Machine learning is a method of data analysis that automates analytical model building. Using algorithms that iteratively learn from data, machine learning allows computers to find hidden insights without being explicitly programmed where to look.”



Source: <http://www.sas.com/>

# Machine Learning With Spark

- *What problems does it solve ?*

- Marketing
- Human resources
- Risk management
- Health care
- Travel
- Education
- ...



# Machine Learning With Spark

- ***MLlib Library :***

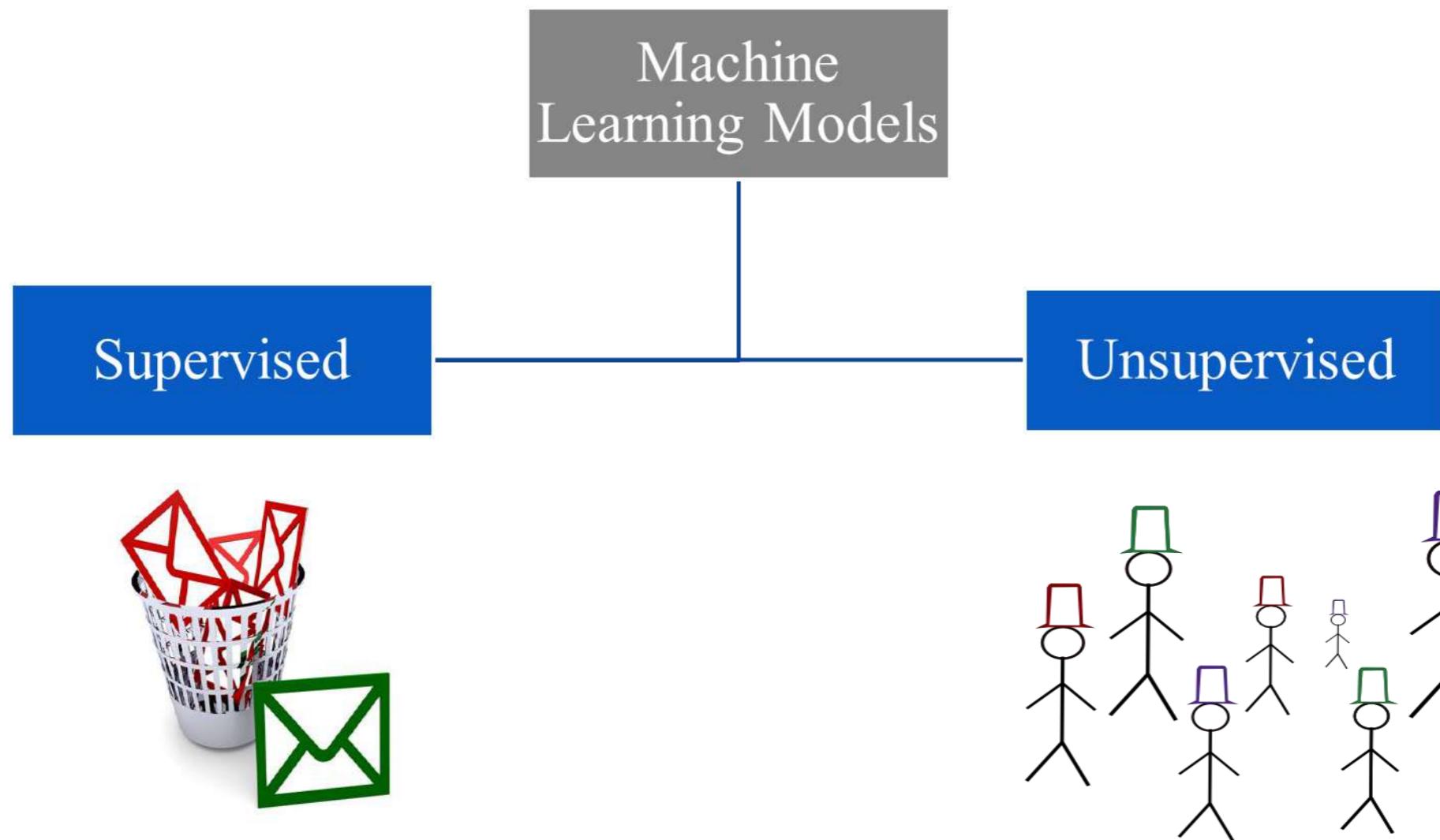
“MLlib is Spark’s scalable machine learning library consisting of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as underlying optimization Primitives”



Source: <https://spark.apache.org>

# Machine Learning With Spark

- *Types of Machine Learning system:*



- ***Supervised models:***

- Build a model that makes predictions
- The correct classes of the training data are known
- We can validate performance
- Two broad categories:

*Classification:* assign a class to an observation. e.g.: patient will have a heart attack or not.

*Regression:* predict a continuous measurement for an observation. e.g.: car prices

- Algorithms:

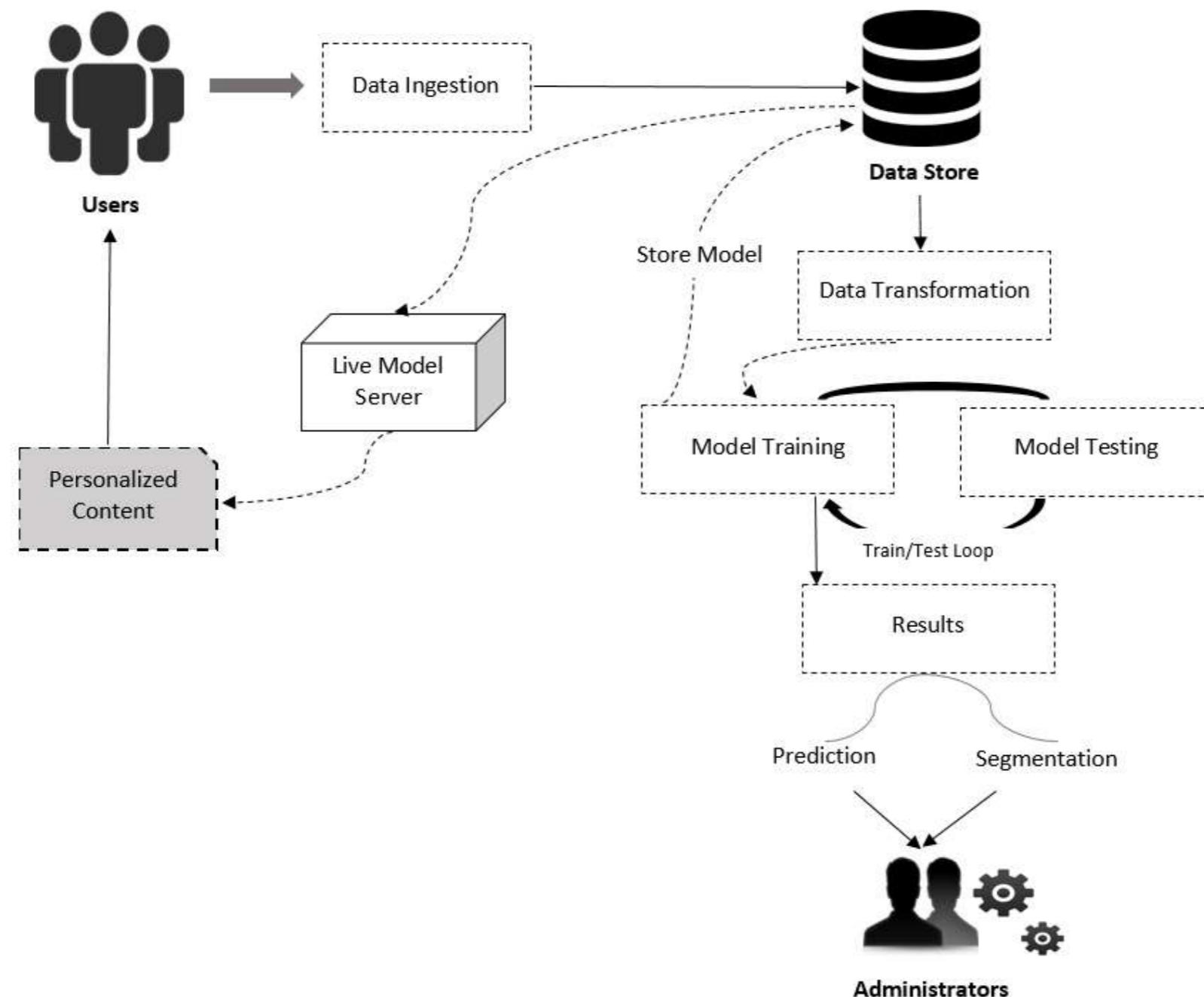
Regression, Decision Tree, Naive Bayes, SVM, ...

- ***Unsupervised models:***

- Exploratory data analysis to find hidden patterns or grouping in data
- The clusters are modeled using a measure of similarity
- Performance ?
- Algorithms:  
ACP, K-means Clustering , Hierarchical clustering ...

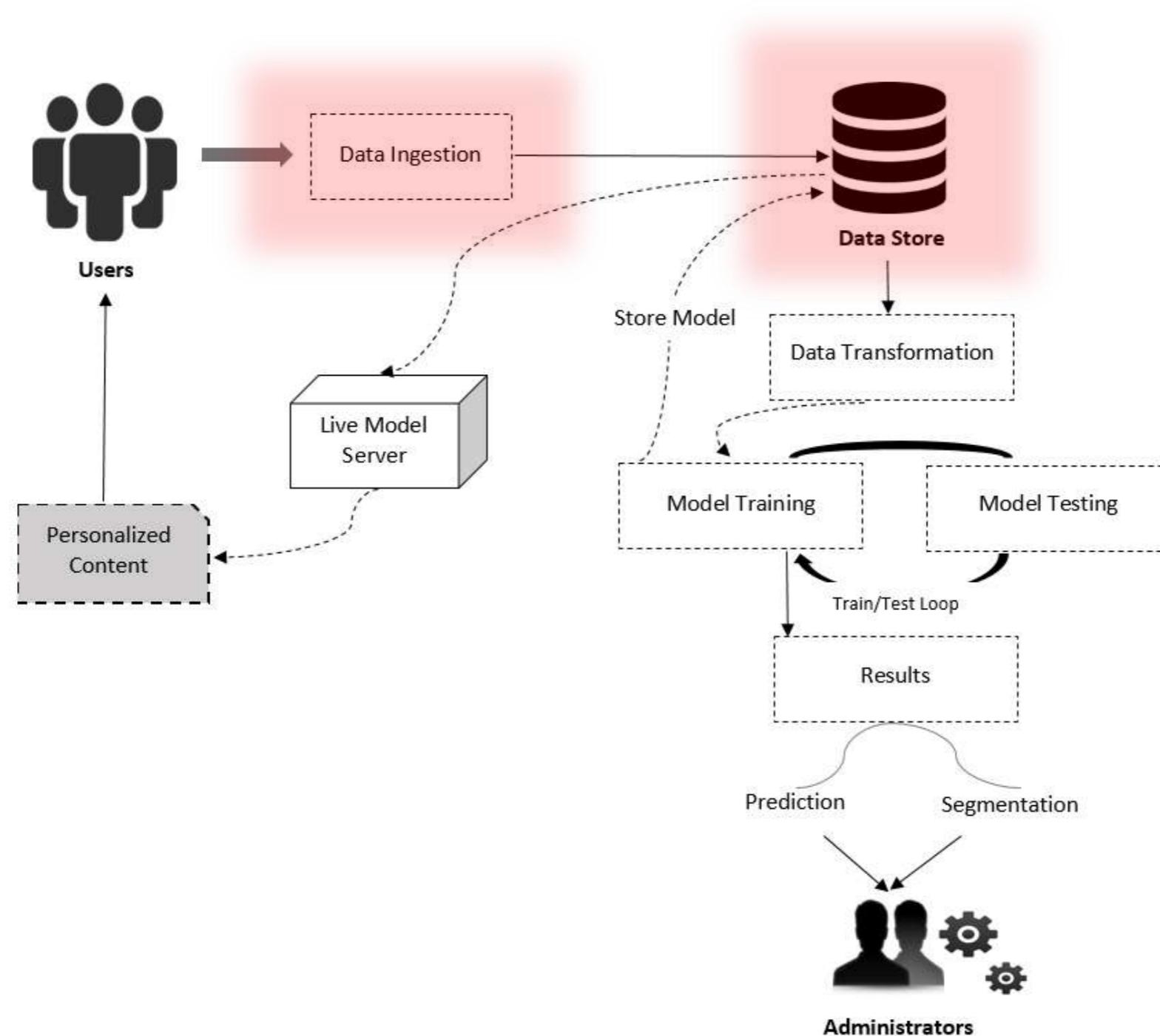
# Machine Learning With Spark

- **ML Architecture:**



# Machine Learning With Spark

- *Data Ingestion and storage:*



- ***Data Ingestion:***

- Browser, and mobile application event logs or accessing external web APIs

- ***Data Storage:***

- HDFS, Amazon S3, and other filesystems; SQL databases such as MySQL or PostgreSQL; distributed NoSQL data stores such as HBase, Cassandra, and DynamoDB, ...

- ***Useful datasets available publicly:***

- UCI Machine Learning Repository: <http://archive.ics.uci.edu/ml/>
- Amazon AWS public datasets: <http://aws.amazon.com/publicdatasets/>
- Kaggle: <http://www.kaggle.com/competitions>
- KDnuggets: <http://www.kdnuggets.com/datasets/index.html>

- ***Used dataset:***

- Training data from the Kaggle: train.tsv
- Classifying problem : web page is a ‘*Short-lived*’ or ‘*not*’

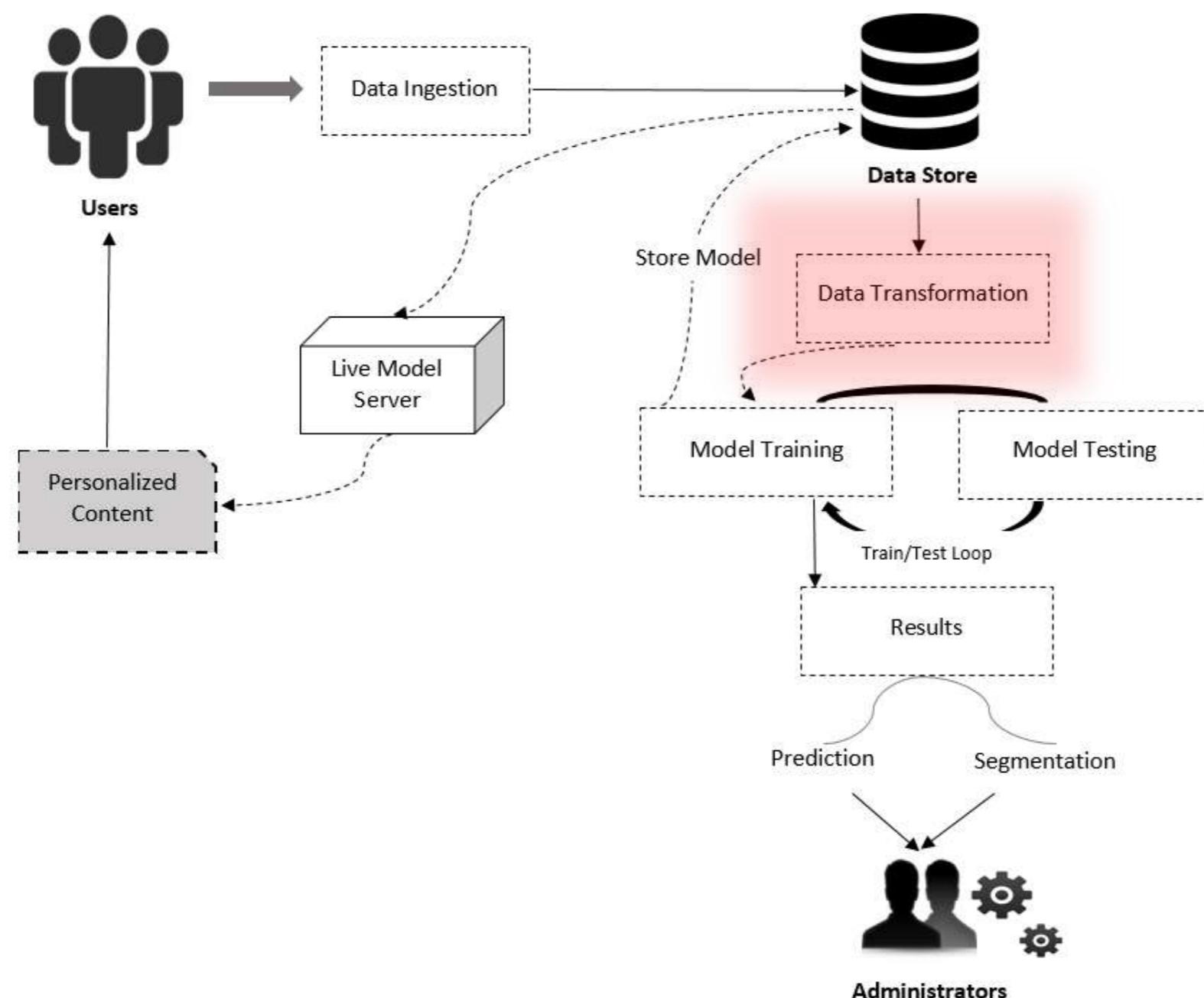
<http://www.kaggle.com/c/stumbleupon/data>

- Source code on Github

<https://github.com/onsDridi/SparkML.git>

# Machine Learning With Spark

- *Data transformation:*



- *Data transformation:*

- Filter out or remove records with bad or missing values
- Fill in bad or missing data
- Apply robust techniques to outliers
- Apply transformations to potential outliers
- Extract useful features



- ***Data transformation:***

- Remove the first line

```
sed 1d train.tsv > train_noheader.tsv
```

- Run Spark

```
>/bin/spark-shell --driver-memory 4g
```

- Create RDD

```
val rawData = sc.textFile("data/train_noheader.tsv")
val records = rawData.map(line => line.split("\t"))
records.first()
```

```
Array[String] = Array("http://www.bloomberg.com/news/2010-12-23/ibm-
predicts-holographic-calls-air-breathing-batteries-by-2015.html", "4042",
...")
```

- ***Data transformation:***
  - Apply some data cleaning

```
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.linalg.Vectors
    val data = records.map { r => val trimmed = r.map(_.replaceAll("\\\"", ""))
    val label = trimmed(r.size - 1).toInt
    val features = trimmed.slice(4, r.size - 1).map(d => if (d == "?") 0.0 else
      d.toDouble)
    LabeledPoint(label, Vectors.dense(features))}
```

- ***Data transformation:***

- Cache the data

```
[data.cache  
val numData = data.count]
```

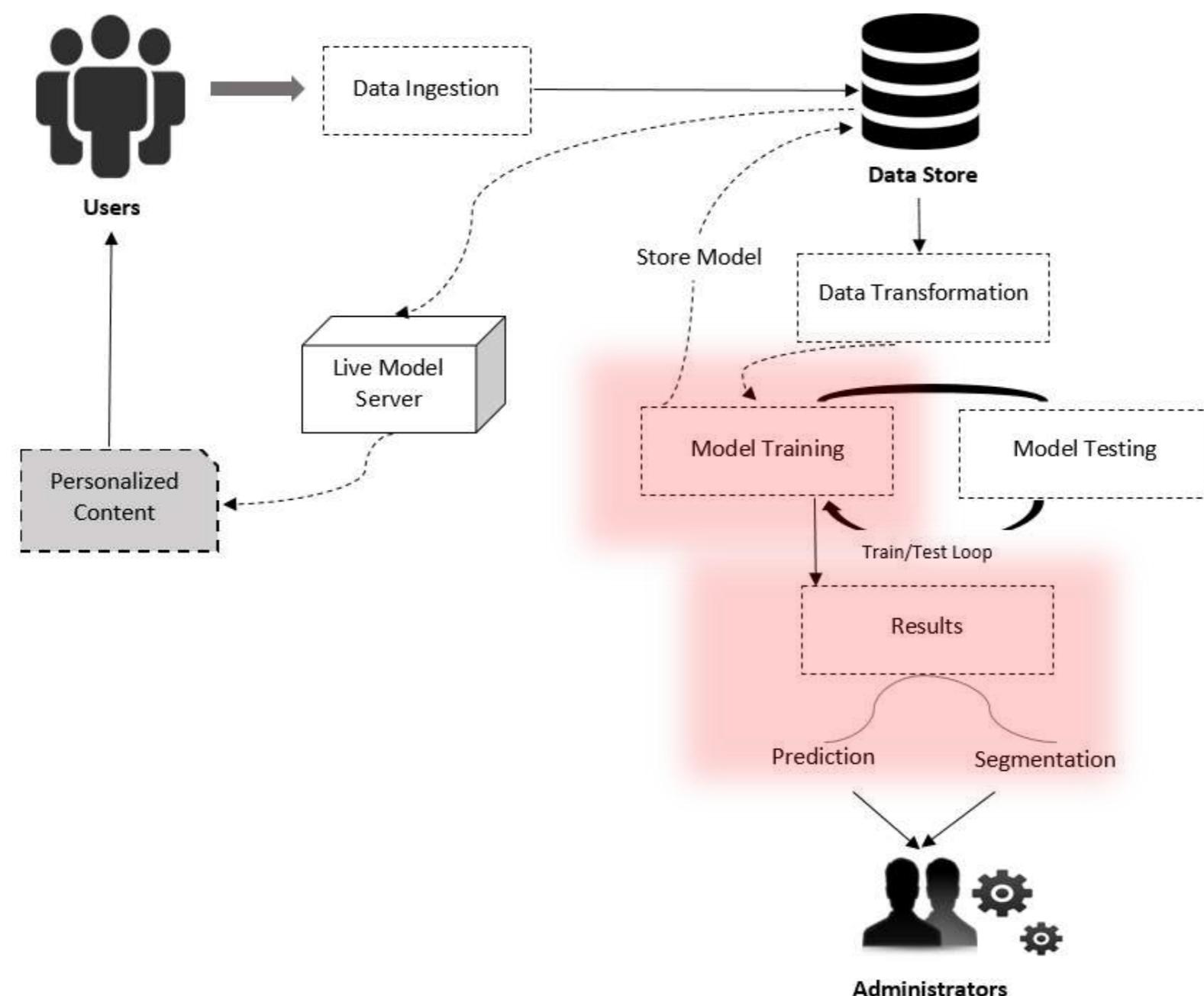
The value is: **7395**

- Replace the negative values by zero

```
val nbData = records.map { r =>  
    val trimmed = r.map(_.replaceAll("\\\"", ""))  
    val label = trimmed(r.size - 1).toInt  
    val features = trimmed.slice(4, r.size - 1).map(d => if (d == "?")  
        0.0 else d.toDouble).map(d => if (d < 0) 0.0 else d)  
    LabeledPoint(label, Vectors.dense(features))  
}
```

# Machine Learning With Spark

- *Training Classification Model:*



- ***Training Classification model:***

- Set up input parameters

```
import org.apache.spark.mllib.classification.LogisticRegressionWithSGD  
import org.apache.spark.mllib.classification.SVMWithSGD  
import org.apache.spark.mllib.classification.NaiveBayes  
import org.apache.spark.mllib.tree.DecisionTree  
import org.apache.spark.mllib.tree.configuration.Algo  
import org.apache.spark.mllib.tree.impurity.Entropy  
val numIterations = 10  
val maxTreeDepth = 5
```

- *Training Classification model:*

- Train logistic regression

```
val lrModel = LogisticRegressionWithSGD.train(data, numIterations)
```

```
14/12/06 13:41:47 INFO DAGScheduler: Job 81 finished: reduce at  
RDDFunctions.scala:112, took 0.011968 s  
  
14/12/06 13:41:47 INFO GradientDescent: GradientDescent.  
runMiniBatchSGD finished. Last 10 stochastic losses 0.6931471805599474,  
1196521.395699124, Infinity, 1861127.002201189, Infinity,  
2639638.049627607, Infinity, Infinity, Infinity, Infinity  
  
lrModel: org.apache.spark.mllib.classification.LogisticRegressionModel =  
(weights=[-0.11372778986947886,-0.511619752777837,
```

- *Training Classification model:*

- Train an SVM model

```
val svmModel = SVMWithSGD.train(data, numIterations)
```

```
14/12/06 13:43:08 INFO GradientDescent: GradientDescent.runMiniBatchSGD  
finished. Last 10 stochastic losses 1.0, 2398226.619666797,  
2196192.9647478117, 3057987.2024311484, 271452.9038284356,  
3158131.191895948, 1041799.350498323, 1507522.941537049,  
1754560.9909073508, 136866.76745605646
```

```
svmModel: org.apache.spark.mllib.classification.SVMMModel = (weigh  
ts=[-0.12218838697834929,-0.5275107581589767,
```

- *Training Classification model:*

- Train the naïve Bayes model

```
val nbModel = NaiveBayes.train(nbData)
```

```
14/12/06 13:44:48 INFO DAGScheduler: Job 95 finished: collect at  
NaiveBayes.scala:120, took 0.441273 s
```

```
nbModel: org.apache.spark.mllib.classification.NaiveBayesModel = org.  
apache.spark.mllib.classification.NaiveBayesModel@666ac612
```

- *Training Classification model:*

- Train the Decision tree model

```
val dtModel = DecisionTree.train(data, Algo.Classification, Entropy,  
maxTreeDepth)
```

```
14/12/06 13:46:03 INFO DAGScheduler: Job 104 finished: collectAsMap at  
DecisionTree.scala:653, took 0.031338 s  
  
...  
  
total: 0.343024  
  
findSplitsBins: 0.119499  
  
findBestSplits: 0.200352  
  
chooseSplits: 0.199705  
  
dtModel: org.apache.spark.mllib.tree.model.DecisionTreeModel =  
DecisionTreeModel classifier of depth 5 with 61 nodes
```

- **Generate Predictions**
- Use the logistic regression model: only the first feature

```
val dataPoint = data.first  
val prediction = lrModel.predict(dataPoint.features)
```

The value is:

**prediction: Double = 1.0**

```
val trueLabel = dataPoint.label
```

The value is:

**trueLabel = Double = 0.0**

This is wrong !



- ***Generate Predictions***
- Use the logistic regression model: RDD vector

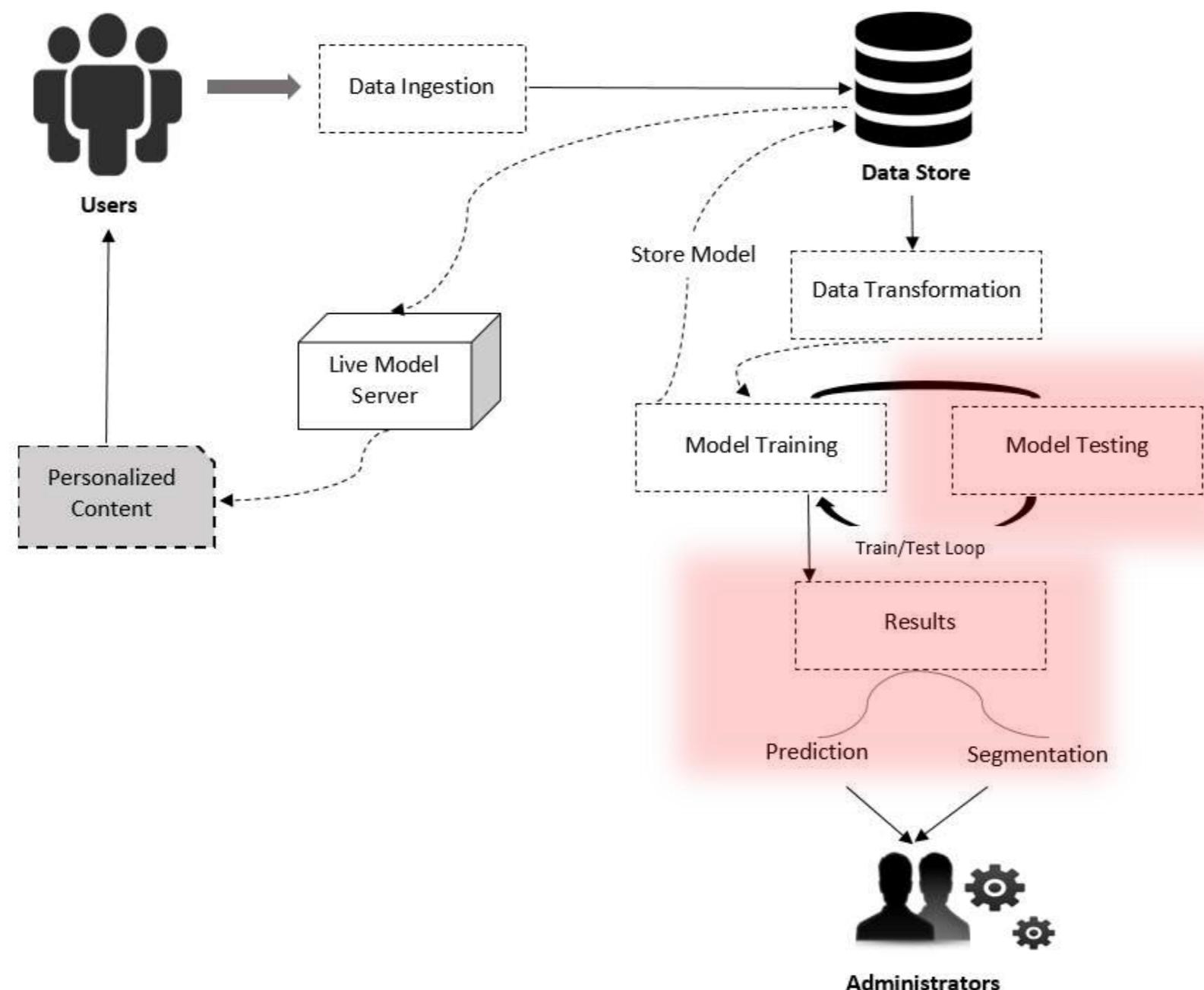
```
val predictions = lrModel.predict(data.map(lp => lp.features))
predictions.take(5)
```

The output is:

**Array[Double] = Array(1.0, 1.0, 1.0, 1.0, 1.0)**

# *Machine Learning With Spark*

# • *Evaluate Performance*



- **Evaluate Performance**
  - Accuracy and prediction error

```
val lrTotalCorrect = data.map { point => if (lrModel.predict(point.features) == point.label) 1 else 0 }.sum  
val lrAccuracy = lrTotalCorrect / data.count
```

The output is:

**lrAccuracy: Double = 0.5146720757268425**

**Not excellent !**

- **Improving model performance**
  - Feature standardization, Additional features, Correct form of data
  - Feature standardization

Matrix of vectors using the *RowMatrix* class

```
import org.apache.spark.mllib.linalg.distributed.RowMatrix  
val vectors = data.map(lp => lp.features)  
val matrix = new RowMatrix(vectors)  
val matrixSummary = matrix.computeColumnSummaryStatistics()
```

```
    println(matrixSummary.mean)
```

```
    println(matrixSummary.min)
```

```
    println(matrixSummary.variance)
```

- ***Improving model performance***
  - Feature standardization

*StandardScaler ou Normalizer*

```
import org.apache.spark.mllib.feature.StandardScaler  
val scaler = new StandardScaler(withMean = true, withStd =  
true).fit(vectors)  
val scaledData = data.map(lp =>  
LabeledPoint(lp.label,scaler.transform(lp.features)))
```

- ***Improving model performance***

- Feature standardization

```
val lrModelScaled = LogisticRegressionWithSGD.train(scaledData,  
numIterations)  
val lrTotalCorrectScaled = scaledData.map { point =>  
    if (lrModelScaled.predict(point.features) == point.label) 1 else  
    0 }.sum  
val lrAccuracyScaled = lrTotalCorrectScaled / numData
```

The output is:

**lrAccuracyScaled: Double = 62.0419%**

**It's better !**

- ***Text Mining***

Text processing is too complex for two reasons:

- Text and language have an inherent structure
- The effective dimensionality of text data is extremely large

Possibilities with MLLib:

- Term weighting schemes, Feature hashing , tokenization, Removing stop words , Excluding terms based on frequency, stemming , etc.

# *Comparison with other tools*

Spark MLLib and Apache Mahout:

- In case of Mahout it is Hadoop MapReduce and in case of MLlib it is RDD
- Algorithms with Spark MLLib are more developed

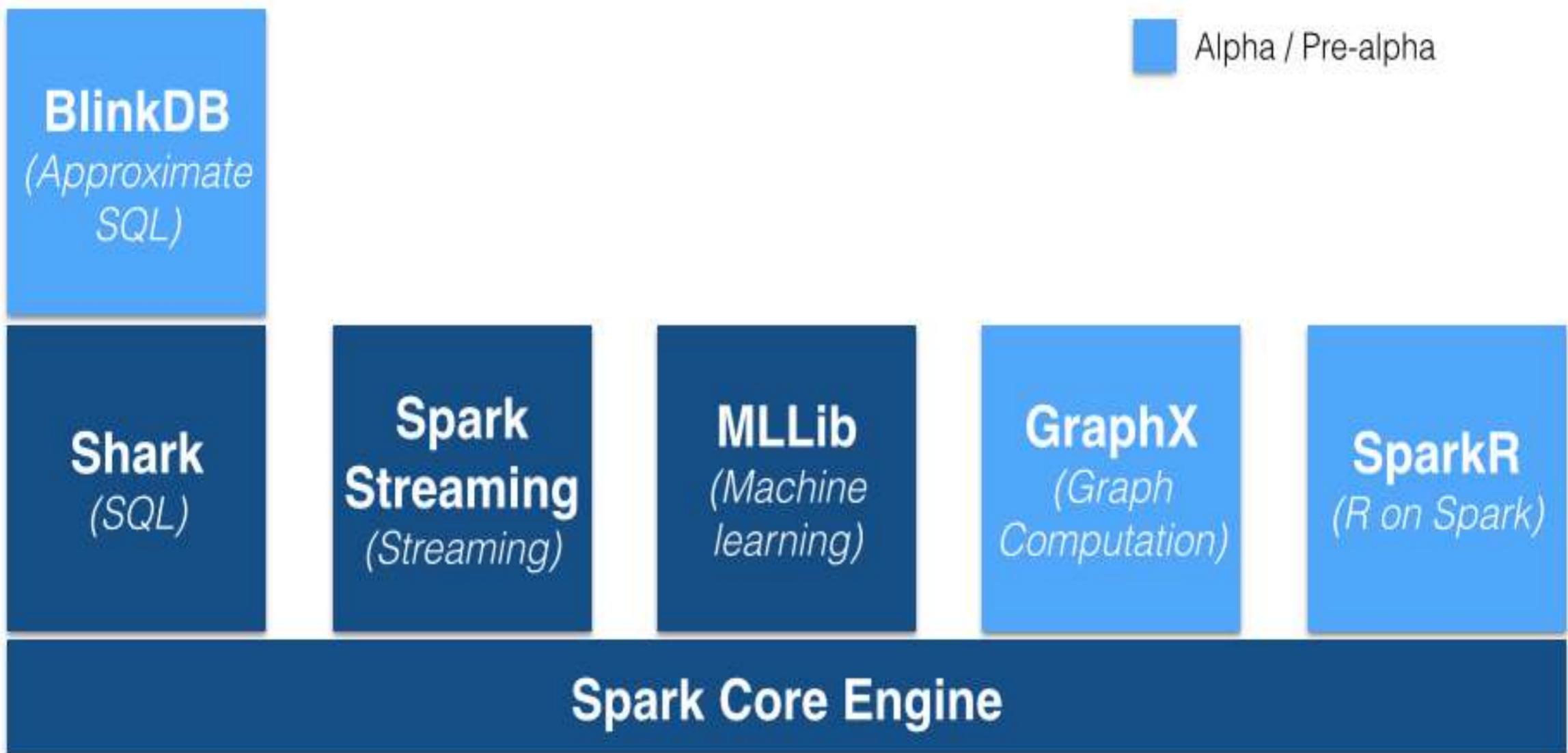


Spark MLLib and R project:



- Spark includes SparkR library
- Working with R is not so fast

# Other applications



Source: <http://www.jorditorres.org/spark-ecosystem/>

- *References*

- Machine Learning With Spark Nick Pentreath
- [spark.apache.org](http://spark.apache.org)
- <https://bighadoop.wordpress.com/2014/04/03/apache-spark-a-fast-big-data-analytics-engine/>
- <https://www.sigmoid.com/apache-spark-internals/>
- <http://www.jorditorres.org/spark-ecosystem/>
- <http://www.sas.com/>
- <https://trongkhoanguyenblog.wordpress.com>

Source: <http://www.jorditorres.org/spark-ecosystem/>

*Thank you*

## CETIC

Aéropôle de Charleroi-Gosselies  
Rue des Frères Wright, 29/3  
B-6041 Gosselies  
[info@cetic.be](mailto:info@cetic.be)

**[www.cetic.be](http://www.cetic.be)**

