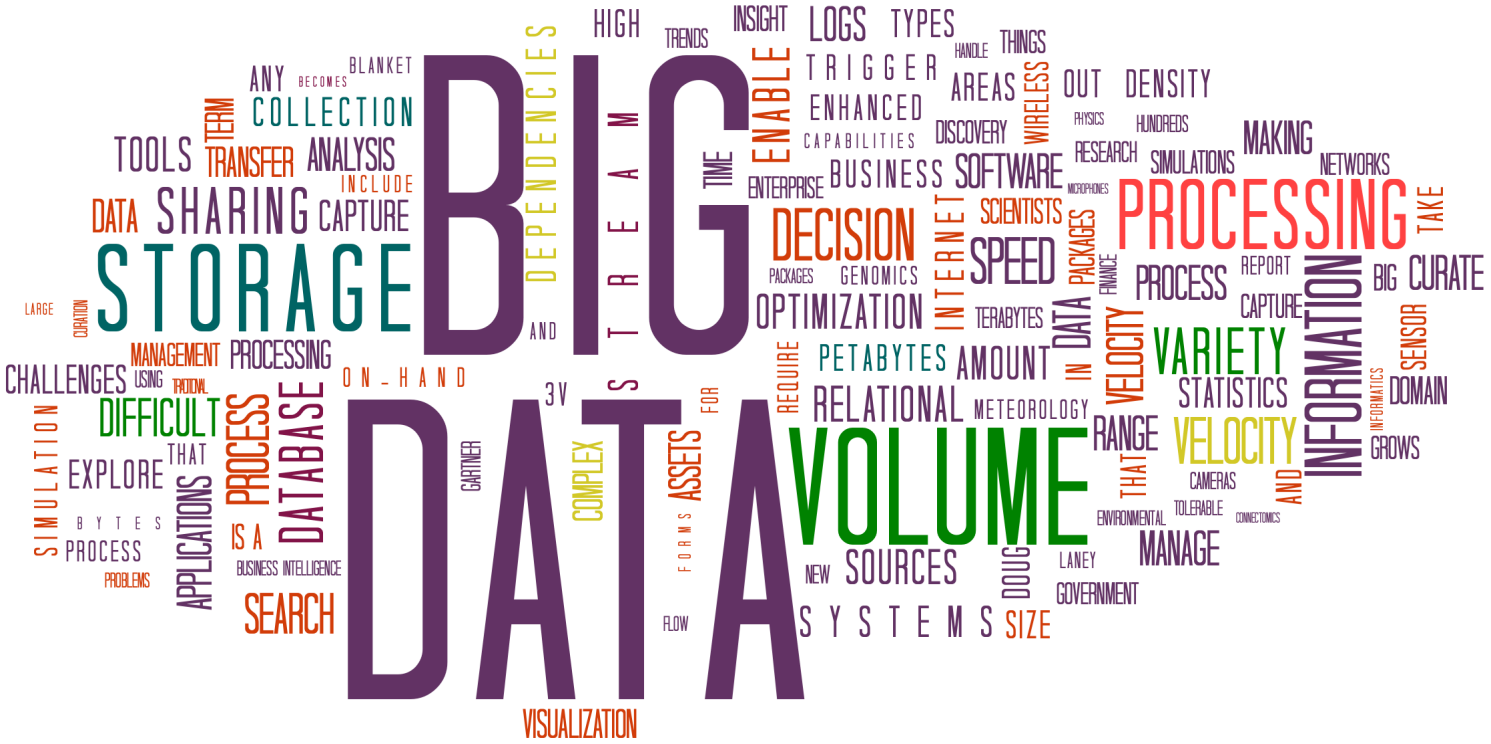


Introduction to Big Data

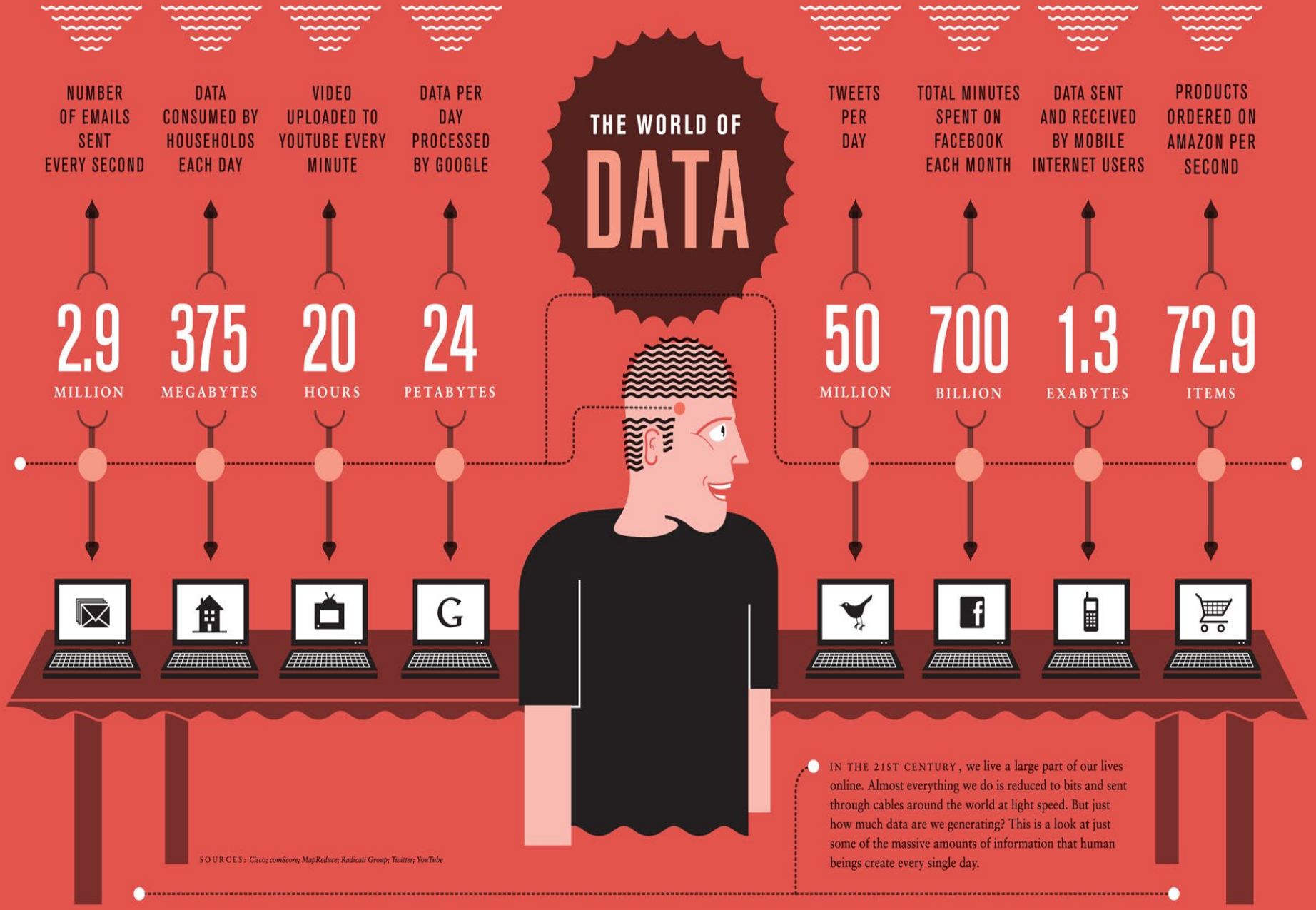
Dr. Putchong Uthayopas
Department of Computer Engineering,
Faculty of Engineering, Kasetsart University
Email: putchong@ku.th

Assoc. Prof. Dr. Thanachart Numnonda
Executive Director
IMC Institute

Adapted by Sorayut Glomglome
Department of Computer Engineering
King Mongkut's Institute of Technology Ladkrabang

PART I: INTRODUCTION TO BIG DATA

THE WORLD OF DATA



IN THE 21ST CENTURY, we live a large part of our lives online. Almost everything we do is reduced to bits and sent through cables around the world at light speed. But just how much data are we generating? This is a look at just some of the massive amounts of information that human beings create every single day.

SOURCES: *Cisco; comScore; MapReduce; Radicati Group; Twitter; YouTube*

A COLLABORATION BETWEEN GOOD AND OLIVER MUNDAY

IN PARTNERSHIP WITH IBM

1 Minutes in Your Life!

DATA NEVER SLEEPS

How Much Data Is Generated **Every Minute?**

Big data is not just some abstract concept used to inspire and mystify the IT crowd; it is the result of an avalanche of digital activity pulsating through cables and airwaves across the world. This data is being created every minute of the day through the most innocuous of online activity that many of us barely even notice. But with every website browsed, status shared, or photo uploaded, we leave digital trails that continually grow the hulking mass of big data. Below, we explore how much data is generated in one minute on the Internet.

EVERY MINUTE of the DAY

- **THE MOBILE WEB** RECEIVES **217** NEW USERS.
- **YOUTUBE** USERS UPLOAD **48** HOURS OF NEW VIDEO.
- **EMAIL USERS** SEND **204,166,667** MESSAGES.
- **GOOGLE** RECEIVES **OVER 2,000,000** SEARCH QUERIES.
- **FACEBOOK USERS** SHARE **684,478** PIECES OF CONTENT.
- **WORDPRESS** USERS PUBLISH **347** NEW BLOG POSTS.
- **571** NEW WEBSITES ARE CREATED.
- CONSUMERS **SPEND $272,070** ON WEB SHOPPING.
- FOURSQUARE USERS **PERFORM 2,083** CHECK-INS.
- **TWITTER USERS** SEND OVER **100,000** TWEETS.
- **FLICKR** USERS ADD **3,125** NEW PHOTOS.
- **INSTAGRAM USERS** SHARE **3,600** NEW PHOTOS.
- **TUMBLR BLOG** OWNERS PUBLISH **27,778** NEW POSTS.
- **BRANDS & ORGANIZATIONS** ON FACEBOOK **RECEIVE 34,722** "LIKES."
- **APPLE** RECEIVES ABOUT **47,000 APP** DOWNLOADS.

WITH NO SIGNS OF SLOWING, THE DATA KEEPS GROWING

These are just some of the more common ways that Internet users add to the big data pool. In truth, depending on the niche of business you're in, there are virtually countless other sources of relevant data to pay attention to. Consider the following:

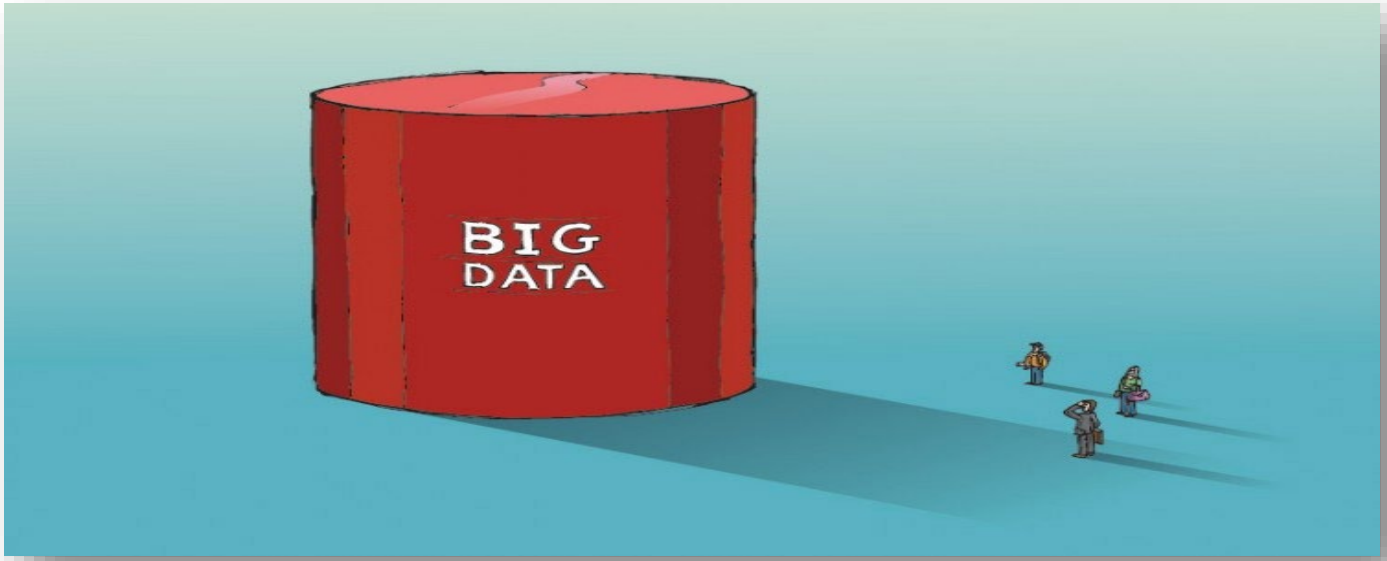
The global Internet population grew 6.59 percent from 2010 to 2011 and now represents **2.1 BILLION PEOPLE.**

These users are real, and they are out there leaving data trails everywhere they go. The team at Domo can help you make sense of this seemingly insurmountable heap of data, with solutions that help executives and managers bring all of their critical information together in one intuitive interface, and then use that insight to transform the way they run their business. To learn more, visit www.domo.com.

DOMO

SOURCES: HTTP://NEWS.INVESTORS.COM/, ROYAL.PINGDOM.COM, BLOG.GROVO.COM, BLOG.HUBSPOT.COM, SIMPLYZESTY.COM, PCWORLD.COM, BIZTECHMAGAZINE.COM, DIGBY.COM

WHAT IS BIG DATA?



Big data is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making.

"Gartner Inc."

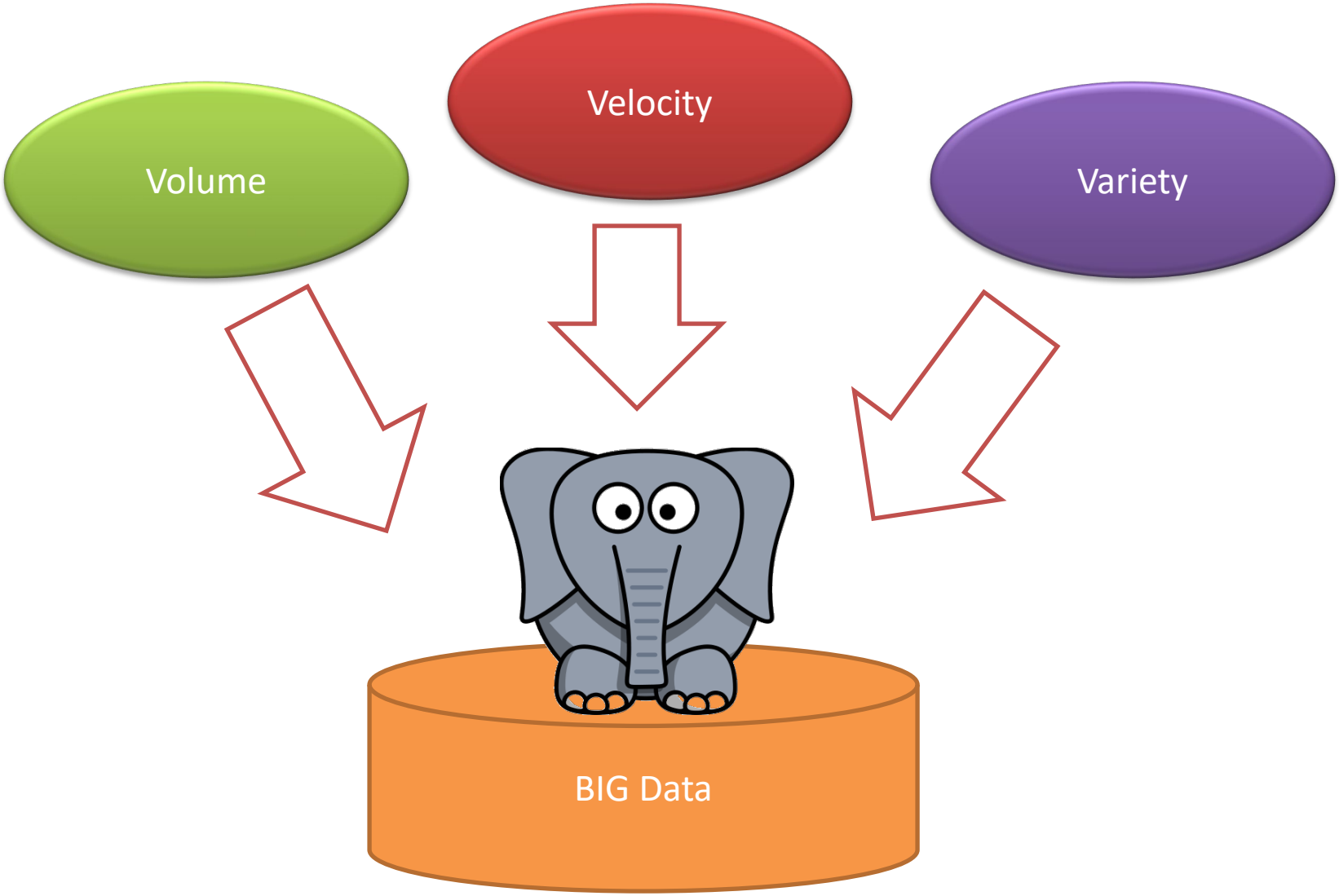
Why BigData?

The real value of big data is in the insights it produces when analyzed—discovered patterns, derived meaning, indicators for decisions, and ultimately the ability to respond to the world with greater intelligence.

- Improve product and service
- Increase customer satisfaction/behavior
- Improve operation efficiency
- Understand emerging market trends

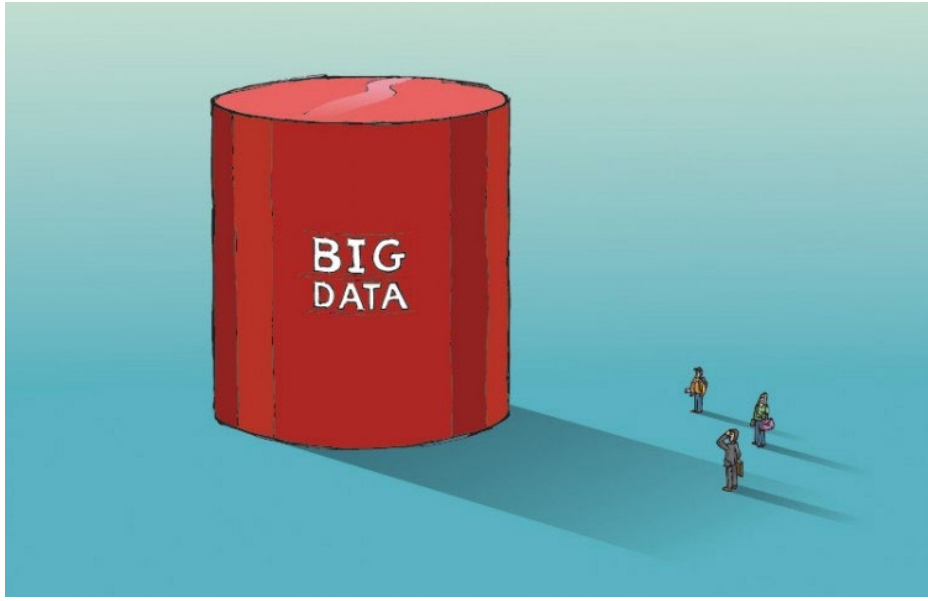
http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/big-data-cloud-technologies-brief.pdf)

Property of Big Data



Volume

- Big data must be huge
  - Beyond the capability of a single computer server to process it
  - Possible to store the data but difficult to process it

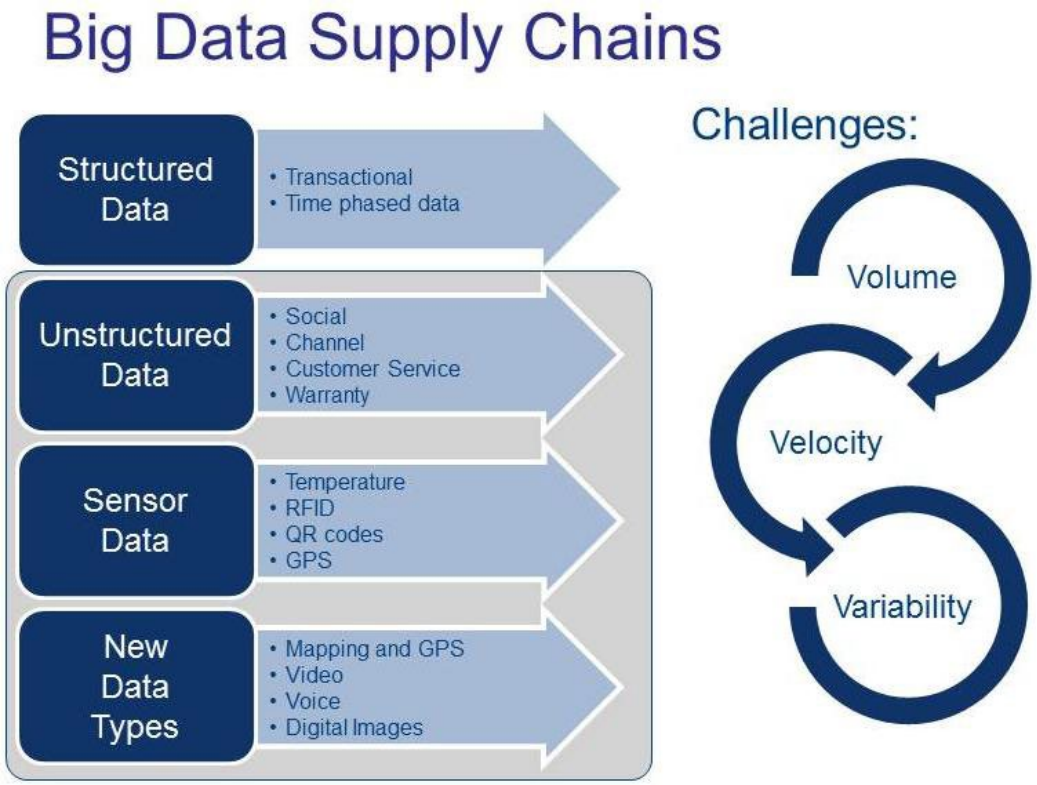


Velocity

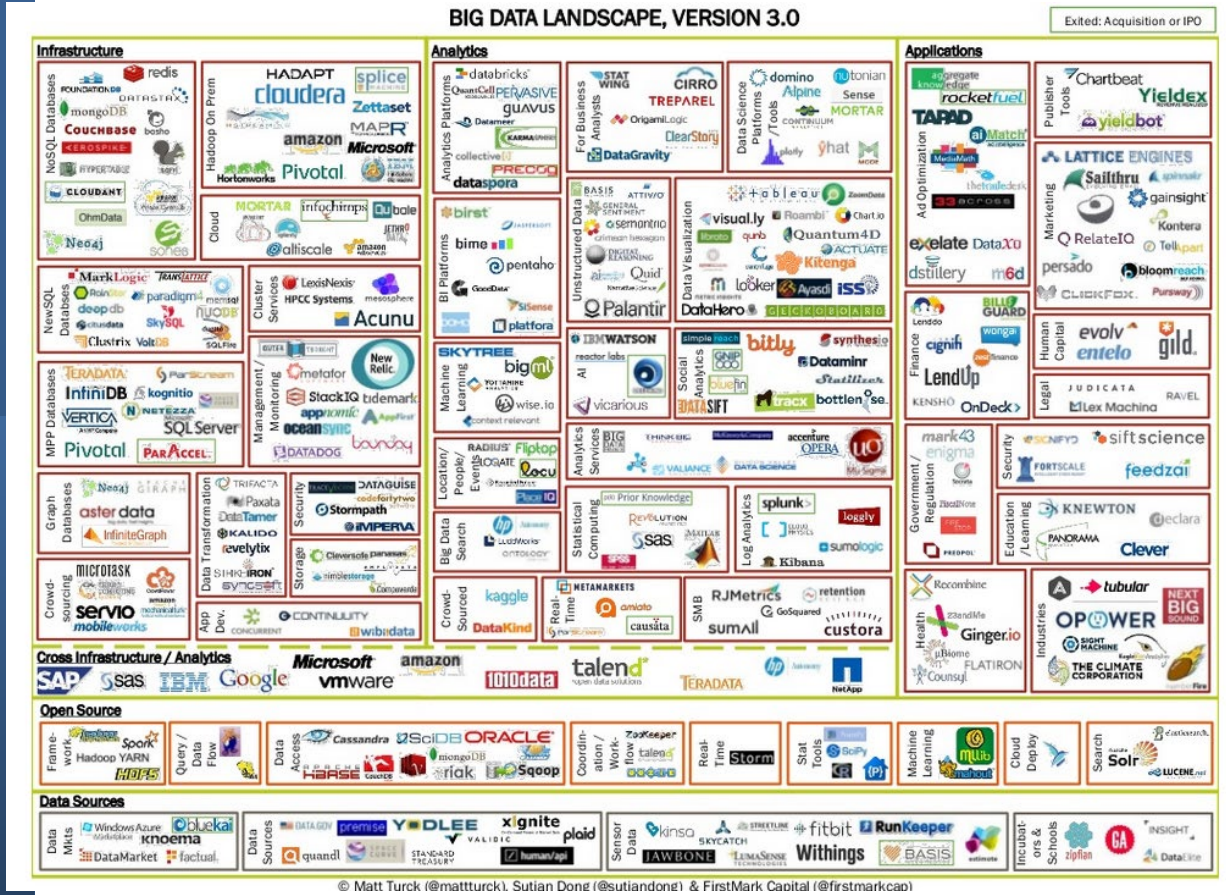
- Big data accumulate at a very fast speed
  - Stock market data
  - Internet access log
  - Social media data
    - Twitter , facebook, IG
- We need to
  - Extract meaning as fast and as much as we can before throwing away the data

Variety

- Data come with variety
  - Traditional data base
  - Documents
  - Web page
  - Social media data
  - Image
  - Video/Audio
  - Location



Big Data Ecosystem

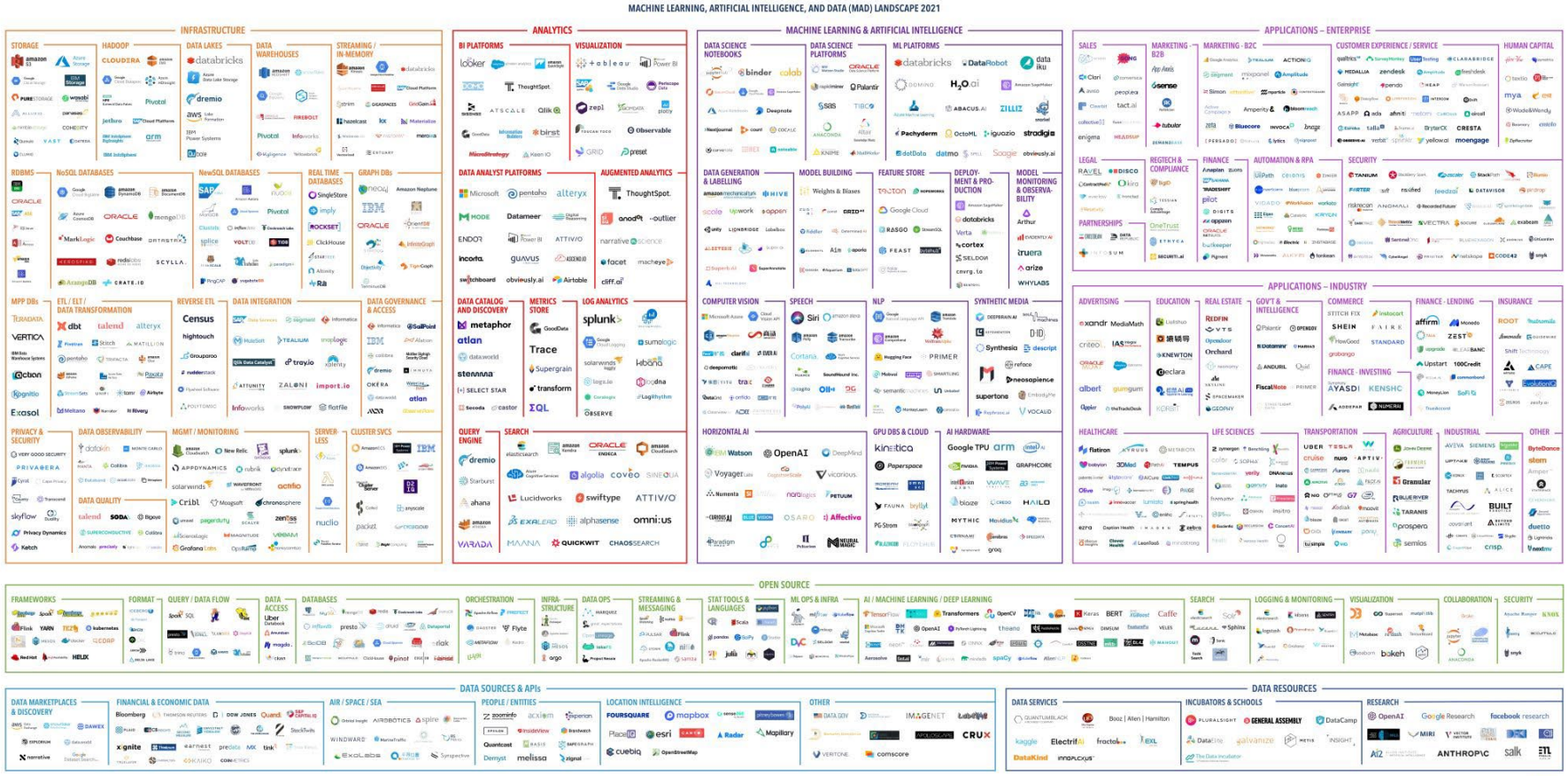


[Big data landscape v 3.0 - Matt Turck (FirstMark) (slideshare.net)]

[Big Data Landscape, v 3.0, analyzed - KDnuggets]

The 2021 Machine Learning, AI and Data (MAD) Landscape

[Red Hot: The 2021 Machine Learning, AI and Data (MAD) Landscape – Matt Turck]



Big Data Eco system- Infrastructure

- **Hadoop-**
  - technologies designed for the storing, processing and analysing of data by breaking up and distributing data into parts and analysing those parts concurrently, rather than tackling one monolithic block of data all in one go.
- **NoSQL**
  - Stands for Not Only SQL
  - involved in processing large volumes of multi-structured data. Most NoSQL databases are most adept at handling discrete data stored among multi-structured data.
- **Massively Parallel Processing (MPP) Databases**
  - MPP databases work by segmenting data across multiple nodes, and processing these segments of data in parallel, and uses SQL.

Reference: http://dataconomy.com/understanding-big-data-ecosystem/

Big Data Eco system- Analytics

- **Analytics Platforms**
  - Integrate and analyse data to uncover new insights, and help companies make better-informed decisions.
- **Visualization Platforms**
  - visualizing data; taking the raw data and presenting it in complex, multi-dimensional visual formats to illuminate the information
- **Business Intelligence (BI) Platforms**
  - analyze data from multiple sources to deliver services such as business intelligence reports, dashboards and visualizations
- **Machine Learning**
  - machine learning is data the algorithm 'learns from', and the output depends on the use case. One of the most famous examples is IBM's super computer Watson, which has 'learned' to scan vast amounts of information to find specific answers, and can comb through 200 million pages of structured and unstructured data in minutes.

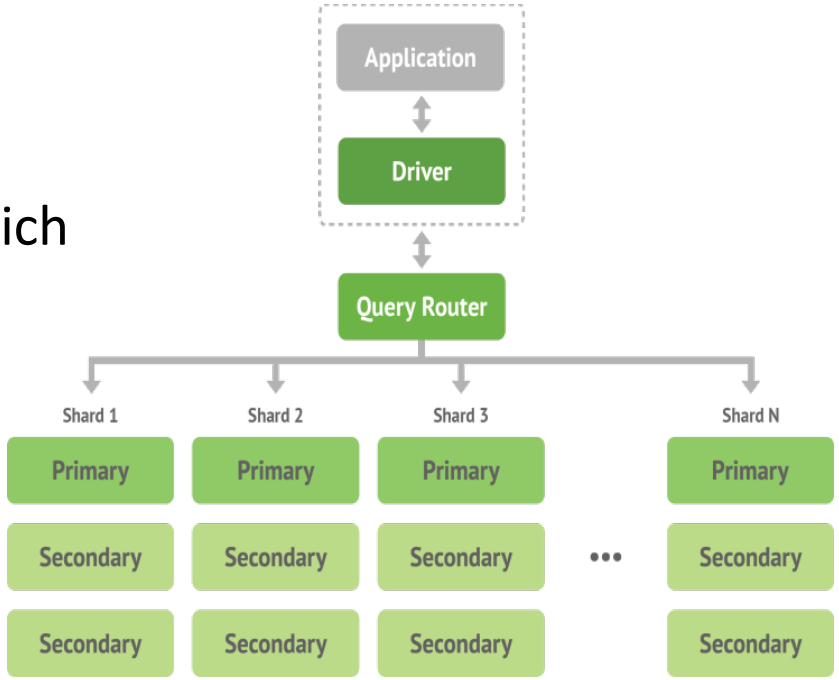
Reference: http://dataconomy.com/understanding-big-data-ecosystem/

NoSQL (Not Only SQL)

- A NoSQL (often interpreted as Not only SQL) database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.
  - being **non-relational, distributed, open-source** and **horizontally scalable**.
  - Used to handle a **huge amount of data**
  - The original intention has been **modern web-scale databases**.

Reference: http://nosql-database.org/

- MongoDB is a general purpose, open-source database.
- MongoDB features:
  - Document data model with dynamic schemas
  - Full, flexible index support and rich queries
  - Auto-Sharding for horizontal scalability
  - Built-in replication for high availability
  - Text search
  - Advanced security



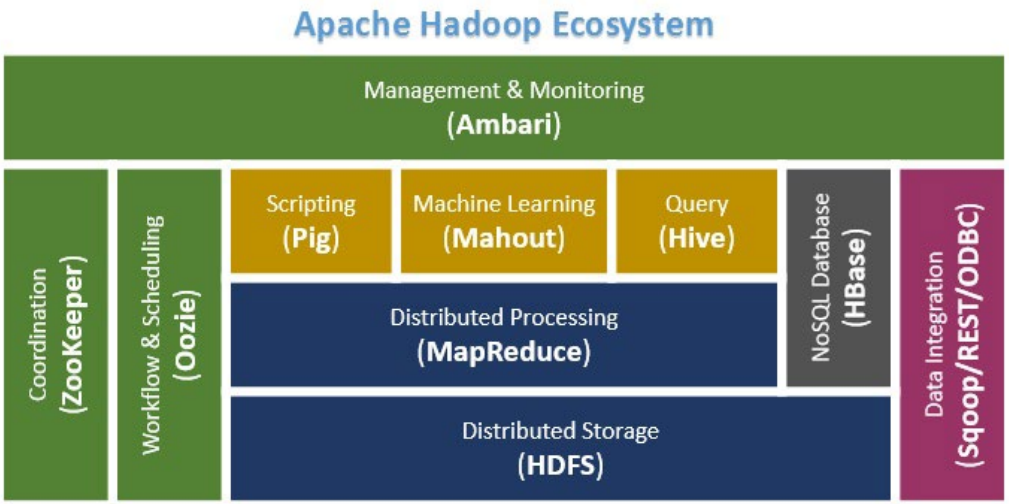
Hadoop is an open-source software framework written in Java for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware.

Hadoop was created by ***Doug Cutting*** and ***Mike Cafarella*** in 2005. Cutting, who was working at Yahoo! at the time, named it after his son's toy elephant.

- The base Apache Hadoop framework is composed of the following modules:
  - *Hadoop Common* – contains libraries and utilities needed by other Hadoop modules;
  - *Hadoop Distributed File System (HDFS)* – a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster;
  - *Hadoop YARN* – a resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users' applications;and
  - *Hadoop MapReduce* – a programming model for large scale data processing.

Hadoop and its Eco System

- Hadoop is not a piece of software but an ecosystem for Big Data processing
- Many tools has been build and share many of the Hadoop components especially the HDFS



Hadoop Eco System

- Pig (http://pig.apache.org)
  - High-level language for data analysis
- Hbase (http://hbase.apache.org) (very large tables -- billions of rows X millions of columns)
  - Table storage for semi-structured data
- Zookeeper (https://zookeeper.apache.org)
  - Coordinating distributed applications
- Hive (https://hive.apache.org)
  - SQL-like Query language and Metastore
- Mahout (http://www.tutorialspoint.com/mahout/)
  - Machine learning

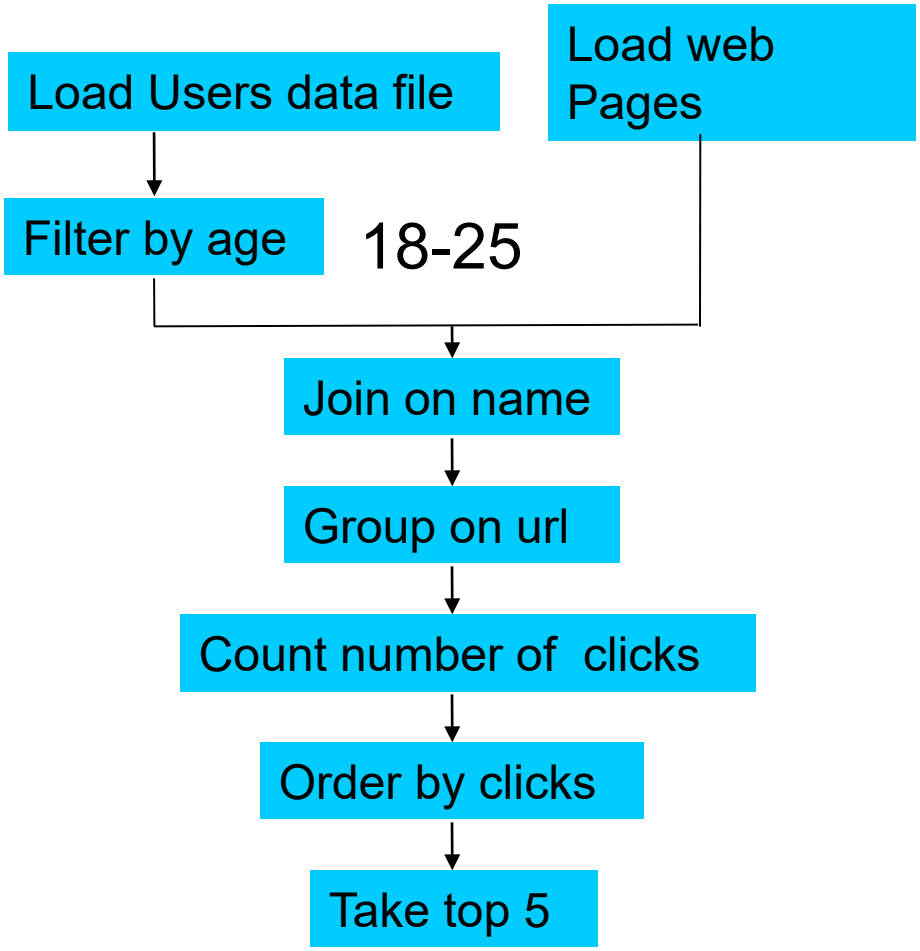
Apache Pig

- Apache Pig is a software framework which offers a run-time environment for execution of MapReduce jobs on a Hadoop Cluster via a high-level scripting language called Pig Latin. The following are a few highlights of this project:
  - Pig is an abstraction (high level programming language) on top of a Hadoop cluster.
  - Pig Latin queries/commands are compiled into one or more MapReduce jobs and then executed on a Hadoop cluster.
  - Just like a real pig can eat almost anything, Apache Pig can operate on almost any kind of data.
  - Hadoop offers a shell called Grunt Shell for executing Pig commands.
  - DUMP and STORE are two of the most common commands in Pig. DUMP displays the results to screen and STORE stores the results to HDFS.
  - Pig offers various built-in operators, functions and other constructs for performing many common operations.

Source: http://www.kalyanhadooptraining.com/2014/07/big-data-basics-part-6-related-apache.html

An Example Problem

- Input: user data in a file, website data in another
- Requirement: find the top 5 most visited pages by users aged 18-25



User A visited Page X

In MapReduce

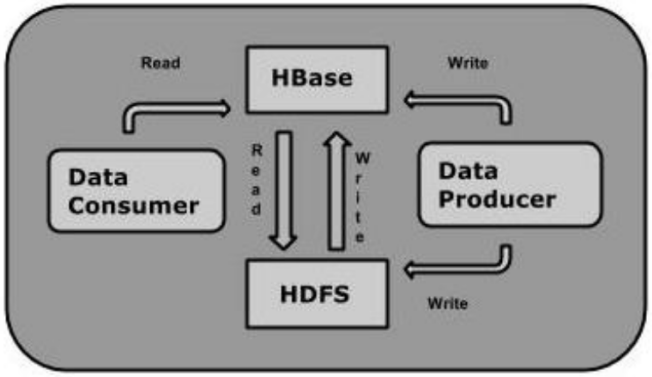
```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.KeyValueTextInputFormat;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.RecordReader;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.mapred.SequenceFileInputFormat;
import org.apache.hadoop.mapred.SequenceFileOutputFormat;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.jobcontrol.Job;
import org.apache.hadoop.mapred.jobcontrol.JobControl;
import org.apache.hadoop.mapred.lib.IdentityMapper;

public class MRExample {
    public static class LoadPages extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, Text> {

        public void map(LongWritable k, Text val,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // Pull the key out
            String line = val.toString();
            int firstComma = line.indexOf(',');
            String key = line.substring(0, firstComma);
            String value = line.substring(firstComma + 1);
            Text outKey = new Text(key);
            // Prepend an index to the value so we know which file
            // it came from.
            Text outVal = new Text("1" + value);
            oc.collect(outKey, outVal);
        }
    }
    public static class LoadAndFilterUsers extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, Text> {

        public void map(LongWritable k, Text val,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // Pull the key out
            String line = val.toString();
            int firstComma = line.indexOf(',');
            String value = line.substring(firstComma + 1);
            int age = Integer.parseInt(value);
            if (age < 18 || age > 25) return;
            String key = line.substring(0, firstComma);
            Text outKey = new Text(key);
            // Prepend an index to the value so we know which file
            // it came from.
            Text outVal = new Text("2" + value);
            oc.collect(outKey, outVal);
        }
    }
    public static class Join extends MapReduceBase
        implements Reducer<Text, Text, Text, Text> {

        public void reduce(Text key,
                Iterator<Text> iter,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // For each value, figure out which file it's from and
store it
            // accordingly.
            List<String> first = new ArrayList<String>();
            List<String> second = new ArrayList<String>();

            while (iter.hasNext()) {
                Text t = iter.next();
                String value = t.toString();
                if (value.charAt(0) == '1')
first.add(value.substring(1));
                else second.add(value.substring(1));
                reporter.setStatus("OK");
            }

            // Do the cross product and collect the values
            for (String s1 : first) {
                for (String s2 : second) {
                    String outval = key + "," + s1 + "," + s2;
                    oc.collect(null, new Text(outval));
                    reporter.setStatus("OK");
                }
            }
        }
    }
    public static class LoadJoined extends MapReduceBase
        implements Mapper<Text, Text, Text, LongWritable> {

        public void map(
                Text k,
                Text val,
                OutputCollector<Text, LongWritable> oc,
                Reporter reporter) throws IOException {
            // Find the url
            String line = val.toString();
            int firstComma = line.indexOf(',');
            int secondComma = line.indexOf(',', firstComma);
            String key = line.substring(firstComma, secondComma);
            // drop the rest of the record, I don't need it anymore,
            // just pass a 1 for the combiner/reducer to sum instead.
            Text outKey = new Text(key);
            oc.collect(outKey, new LongWritable(1L));
        }
    }
    public static class ReduceUrls extends MapReduceBase
        implements Reducer<Text, LongWritable, WritableComparable,
Writable> {

        public void reduce(
                Text key,
                Iterator<LongWritable> iter,
                OutputCollector<WritableComparable, Writable> oc,
                Reporter reporter) throws IOException {
            // Add up all the values we see

            long sum = 0;
            while (iter.hasNext()) {
                sum += iter.next().get();
                reporter.setStatus("OK");
            }

            oc.collect(key, new LongWritable(sum));
        }
    }
    public static class LoadClicks extends MapReduceBase
        implements Mapper<WritableComparable, Writable, LongWritable,
Text> {

        public void map(
                WritableComparable key,
                Writable val,
                OutputCollector<LongWritable, Text> oc,
                Reporter reporter) throws IOException {
            oc.collect((LongWritable)val, (Text)key);
        }
    }
    public static class LimitClicks extends MapReduceBase
        implements Reducer<LongWritable, Text, LongWritable, Text> {

        int count = 0;
        public void reduce(
            LongWritable key,
            Iterator<Text> iter,
            OutputCollector<LongWritable, Text> oc,
            Reporter reporter) throws IOException {

            // Only output the first 100 records
            while (count < 100 && iter.hasNext()) {
                oc.collect(key, iter.next());
                count++;
            }
        }
    }
    public static void main(String[] args) throws IOException {
        JobConf lp = new JobConf(MRExample.class);
        lp.setJobName("Load Pages");
        lp.setInputFormat(TextInputFormat.class);
        lp.setOutputKeyClass(Text.class);
        lp.setOutputValueClass(Text.class);
        lp.setMapperClass(LoadPages.class);
        FileInputFormat.addInputPath(lp, new
Path("/user/gates/pages"));
        FileOutputFormat.setOutputPath(lp,
                new Path("/user/gates/tmp/indexed_pages"));
        lp.setNumReduceTasks(0);
        Job loadPages = new Job(lp);

        JobConf lfu = new JobConf(MRExample.class);
        lfu.setJobName("Load and Filter Users");
        lfu.setInputFormat(TextInputFormat.class);
        lfu.setOutputKeyClass(Text.class);
        lfu.setOutputValueClass(Text.class);
        lfu.setMapperClass(LoadAndFilterUsers.class);
        FileInputFormat.addInputPath(lfu, new
Path("/user/gates/users"));
        FileOutputFormat.setOutputPath(lfu,
                new Path("/user/gates/tmp/filtered_users"));
        lfu.setNumReduceTasks(0);
        Job loadUsers = new Job(lfu);

        JobConf join = new JobConf(MRExample.class);
        join.setJobName("Join Users and Pages");
        join.setInputFormat(KeyValueTextInputFormat.class);
        join.setOutputKeyClass(Text.class);
        join.setOutputValueClass(Text.class);
        join.setMapperClass(IdentityMapper.class);
        join.setReducerClass(Join.class);
        FileInputFormat.addInputPath(join, new
Path("/user/gates/tmp/indexed_pages"));
        FileInputFormat.addInputPath(join, new
Path("/user/gates/tmp/filtered_users"));
        FileOutputFormat.setOutputPath(join, new
Path("/user/gates/tmp/joined"));
        join.setNumReduceTasks(50);
        Job joinJob = new Job(join);
        joinJob.addDependingJob(loadPages);
        joinJob.addDependingJob(loadUsers);

        JobConf group = new JobConf(MRExample.class);
        group.setJobName("Group URLs");
        group.setInputFormat(KeyValueTextInputFormat.class);
        group.setOutputKeyClass(Text.class);
        group.setOutputValueClass(LongWritable.class);
        group.setOutputFormat(SequenceFileOutputFormat.class);
        group.setMapperClass(LoadJoined.class);
        group.setCombinerClass(ReduceUrls.class);
        group.setReducerClass(ReduceUrls.class);
        FileInputFormat.addInputPath(group, new
Path("/user/gates/tmp/joined"));
        FileOutputFormat.setOutputPath(group, new
Path("/user/gates/tmp/grouped"));
        group.setNumReduceTasks(50);
        Job groupJob = new Job(group);
        groupJob.addDependingJob(joinJob);

        JobConf top100 = new JobConf(MRExample.class);
        top100.setJobName("Top 100 sites");
        top100.setInputFormat(SequenceFileInputFormat.class);
        top100.setOutputKeyClass(LongWritable.class);
        top100.setOutputValueClass(Text.class);
        top100.setOutputFormat(SequenceFileOutputFormat.class);
        top100.setMapperClass(LoadClicks.class);
        top100.setCombinerClass(LimitClicks.class);
        top100.setReducerClass(LimitClicks.class);
        FileInputFormat.addInputPath(top100, new
Path("/user/gates/tmp/grouped"));
        FileOutputFormat.setOutputPath(top100, new
Path("/user/gates/top100sitesforusers18to25"));
        top100.setNumReduceTasks(1);
        Job limit = new Job(top100);
        limit.addDependingJob(groupJob);

        JobControl jc = new JobControl("Find top 100 sites for users
18 to 25");
        jc.addJob(loadPages);
        jc.addJob(loadUsers);
        jc.addJob(joinJob);
        jc.addJob(groupJob);
        jc.addJob(limit);
        jc.run();
    }
}
```

In Pig Latin

```
Users = load ‘users’ as (name, age);
Filtered = filter Users by age >= 18 and age <= 25;
Pages = load ‘pages’ as (user, url);
Joined = join Filtered by name, Pages by user;
Grouped = group Joined by url;
Summed = foreach Grouped generate group,
                 count(Joined) as clicks;
Sorted = order Summed by clicks desc;
Top5 = limit Sorted 5;
store Top5 into ‘top5sites’;
```

Apache HBase

- Apache HBase is a distributed, versioned, column-oriented, scalable and a big data store on top of Hadoop/HDFS. The following are a few highlights of this project:
  - Runs on top of Hadoop and HDFS in a distributed fashion.
  - Supports Billions of Rows and Millions of Columns.
  - Runs on a cluster of commodity hardware and scales linearly.
  - Offers consistent reads and writes.
  - Offers easy to use Java APIs for client access.



HBASE Table Structure

Rowid	Column Family			Column Family			Column Family			Column Family		
	clo1	col 2	col 3	col 1	col 2	col 3	col1	col2	col3	col1	col2	col3
1												
2												
3												

COLUMN FAMILIES

Row key	personal data		professional data	
empid	name	city	designation	salary
1	raju	hyderabad	manager	50,000
2	ravi	chennai	sr.engineer	30,000
3	rajesh	delhi	jr.engineer	25,000

Hive

- Developed at Facebook
- Used for majority of Facebook jobs
- "Relational database" built on Hadoop
  - Maintains list of table schemas
  - SQL-like query language (HiveQL)
  - Can call Hadoop Streaming scripts from HiveQL
  - Supports table partitioning, clustering, complex data types, some optimizations

Original Slides by Matei Zaharia UC Berkeley RAD Lab

What Is Hive?

- Hive is designed to enable easy data summarization, ad-hoc querying and analysis of large volumes of data.
  - Hive provides a simple query language called Hive QL, which is based on SQL and which enables users familiar with SQL to do ad-hoc querying, summarization and data analysis easily.
  - Hive QL also allows traditional map/reduce programmers to be able to plug in their custom mappers and reducers to do more sophisticated analysis that may not be supported by the built-in capabilities of the language.

https://cwiki.apache.org/confluence/display/Hive/Tutorial

Creating a Hive Table

```
CREATE TABLE page_views(viewTime INT, userid BIGINT,
                    page_url STRING, referrer_url
STRING,
                    ip STRING COMMENT 'User IP
address')
COMMENT 'This is the page view table'
PARTITIONED BY(dt STRING, country STRING)
STORED AS SEQUENCEFILE;
```

- Partitioning breaks table into separate files for each (dt, country) pair

  Ex: /hive/page_view/dt=2008-06-08,country=USA

  /hive/page_view/dt=2008-06-08,country=CA

A Simple Query

- Find all page views coming from xyz.com on March 31st:

```
SELECT page_views.*
FROM page_views
WHERE page_views.date >= '2008-03-01'
AND page_views.date <= '2008-03-31'
AND page_views.referrer_url like '%xyz.com';
```

- Hive only reads partition `2008-03-01,*` instead of scanning entire table

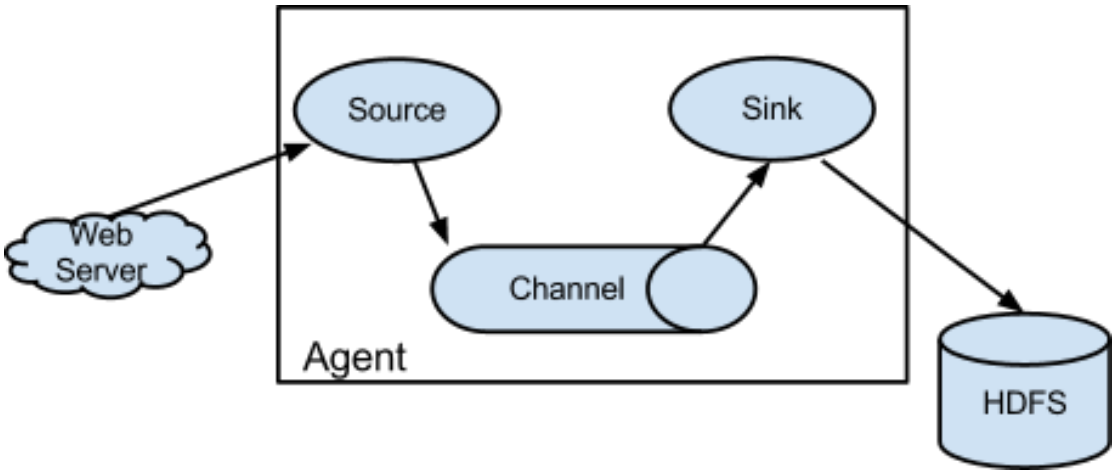
- Apache Mahout is a scalable machine learning and data mining library. The following are a few highlights of this project:
  - Mahout implements the machine learning and data mining algorithms using MapReduce.
  - Mahout has 4 major categories of algorithms: Collaborative Filtering, Classification, Clustering, and Dimensionality Reduction.
  - Mahout library contains two types of algorithms: Ones that can run in local mode and the others which can run in a distributed fashion.
- List of algorithm supported is here. http://mahout.apache.org/users/basics/algorithms.html

Who use Mahout?

- Foursquare uses Mahout for its recommendation engine.
- Twitter uses Mahout's LDA implementation for user interest modeling
- Yahoo! Mail uses Mahout's Frequent Pattern Set Mining. See slides
- Foursquare uses Mahout for its recommendation engine.

Apache Flume

- Apache Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of streaming data into the Hadoop Distributed File System (HDFS).
  - based on streaming data flows
  - tunable reliability mechanisms for failover and recovery.



Source: http://hortonworks.com/hadoop/flume/

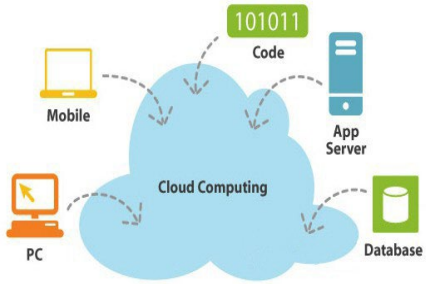
Flume Example Use Case

- Flume can be used to log manufacturing operations.
  - When one run of product comes off the line, it generates a log file about that run.
    - this occurs hundreds or thousands of times per day
  - Large volume log file data can stream through Flume
  - Years of production runs can be stored in HDFS and analyzed by a quality assurance engineer using Apache Hive.

"Your recent Amazon purchases, Tweet score and location history makes you 23.5% welcome here."

PART II : BIG DATA BENEFIT AND USE CASE

The buzzwords converge!



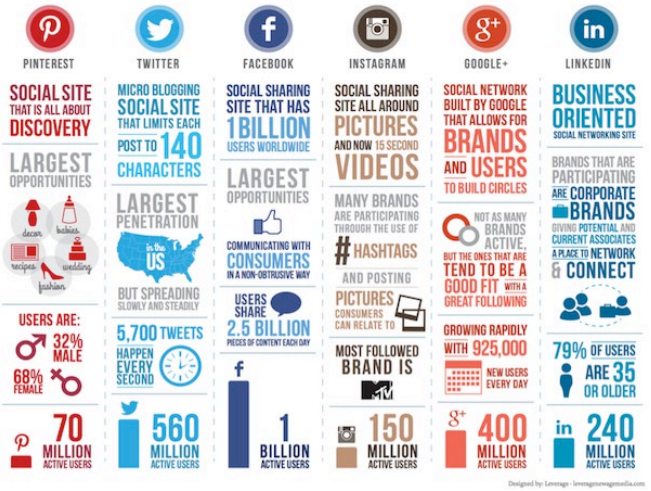
Cloud Computing

Internet of Things

Big Data

Social Media Analytics

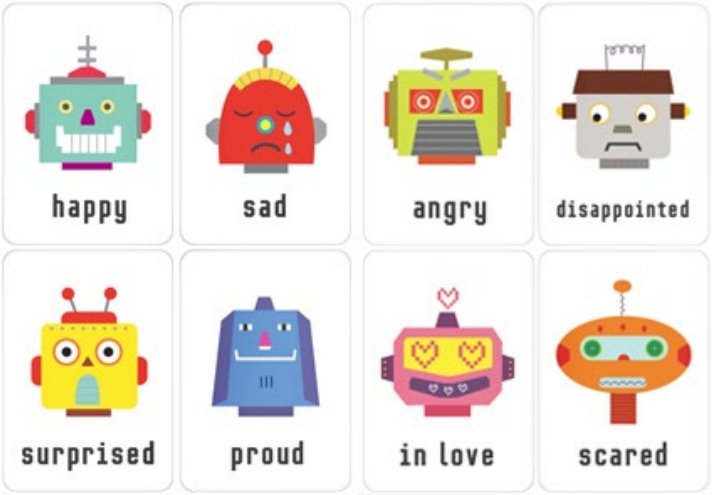
- **Social media analytics** is the practice of gathering data from blogs and **social media** websites and analyzing that data to make business decisions. The most common use of **social media analytics** is to mine customer sentiment in order to support marketing and customer service activities.



[What is social media analytics? - Definition from WhatIs.com]

Sentiment Analysis

- Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials.
- sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document.
    - judgment or evaluation (see appraisal theory)
    - affective state (that is to say, the emotional state of the author when writing)
    - intended emotional communication (that is to say, the emotional effect the author wishes to have on the reader).
- Application
    - Book Recommendation
    - Product review



Use Cases

Banking/Insurance: better and deeper understanding of risk to avoid credit crisis – compliance and regulation – risk management – fraud analysis – catastrophe modeling

Telecommunications: more reliable network where we can predict and prevent failure – customers churn prevention – CDRs analysis – customer retention

Media: more content that is lined up with your personal preferences – customer relationship management

Energy: smart meters - distribution load forecasting and scheduling

Life Science: better targeted medicines with fewer complications and side effects – medical data – data satellite analysis – weather forecasting – global warming

Retail: a personal experience with products and offers that are just what you need – Supply Chain management and analytics – dynamic prices optimization

Government: government services that are based on hard data, not just gut

Manufacturing: preventive maintenance and repair – sensors data to look at failure – supply chain management – products life management

Study of Human Society

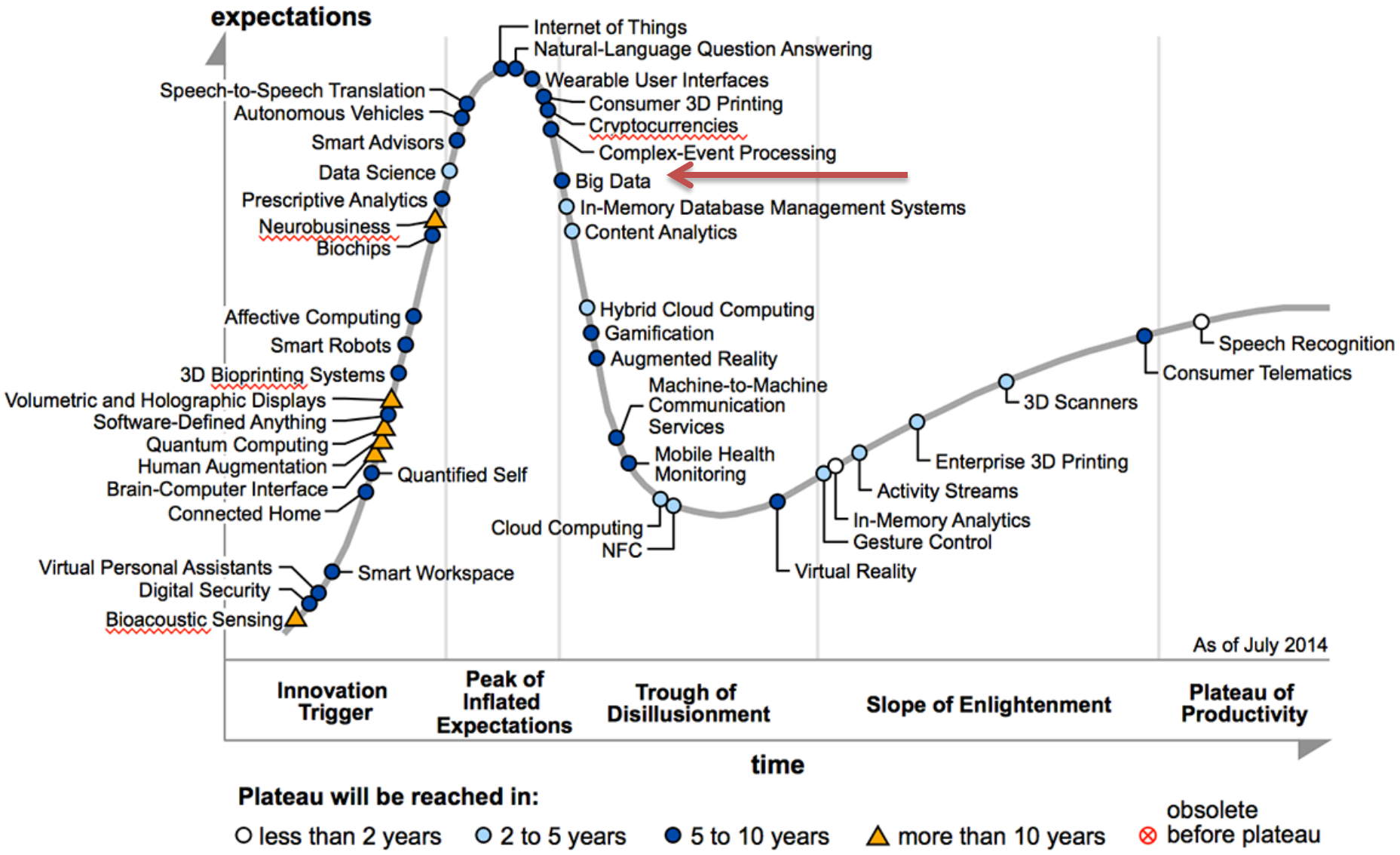
- Facebook, in collaboration with the University of Milan, conducted experiment that involved
  - the entire social network as of May 2011
  - more than 10 percent of the world's population.
- Analyzing the 69 billion friend connections among those 721 million people showed that
  - four intermediary friends are usually enough to introduce anyone to a random stranger.

How facebook handle Big Data?

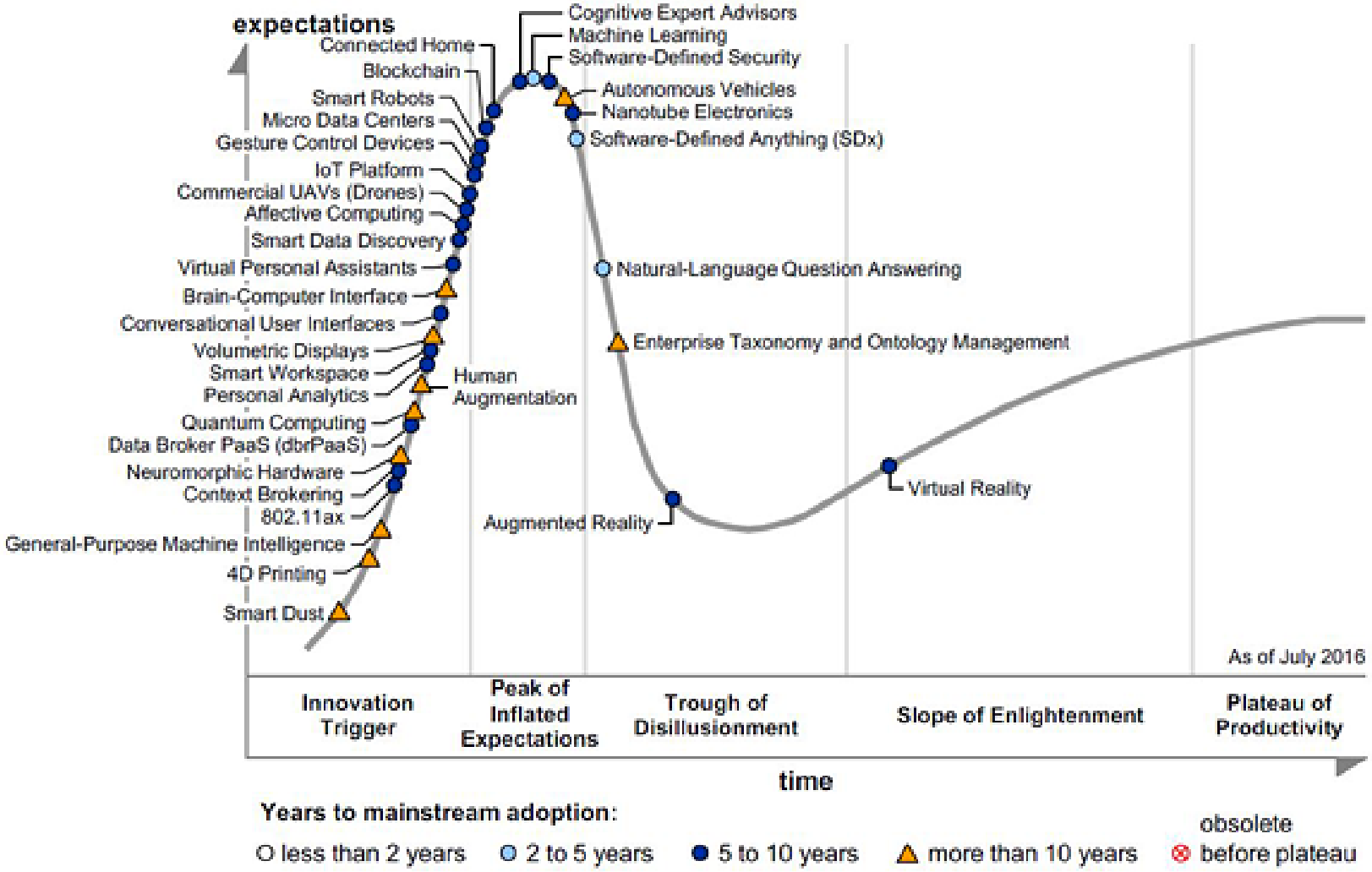
- Facebook built its data storage system using open-source software called Hadoop.
  - Hadoop spreading them across many machines inside a data center.
  - Use Hive, open-source that acts as a translation service, making it possible to query vast Hadoop data stores using relatively simple code.
- Much of Facebook's data resides in one Hadoop store more than 100 petabytes (a million gigabytes) in size, says Sameet Agarwal, a director of engineering at Facebook who works on data infrastructure, and the quantity is growing exponentially. "Over the last few years we have more than doubled in size every year,"

Malaysia BDA Pilot Projects

PART III: TRENDS



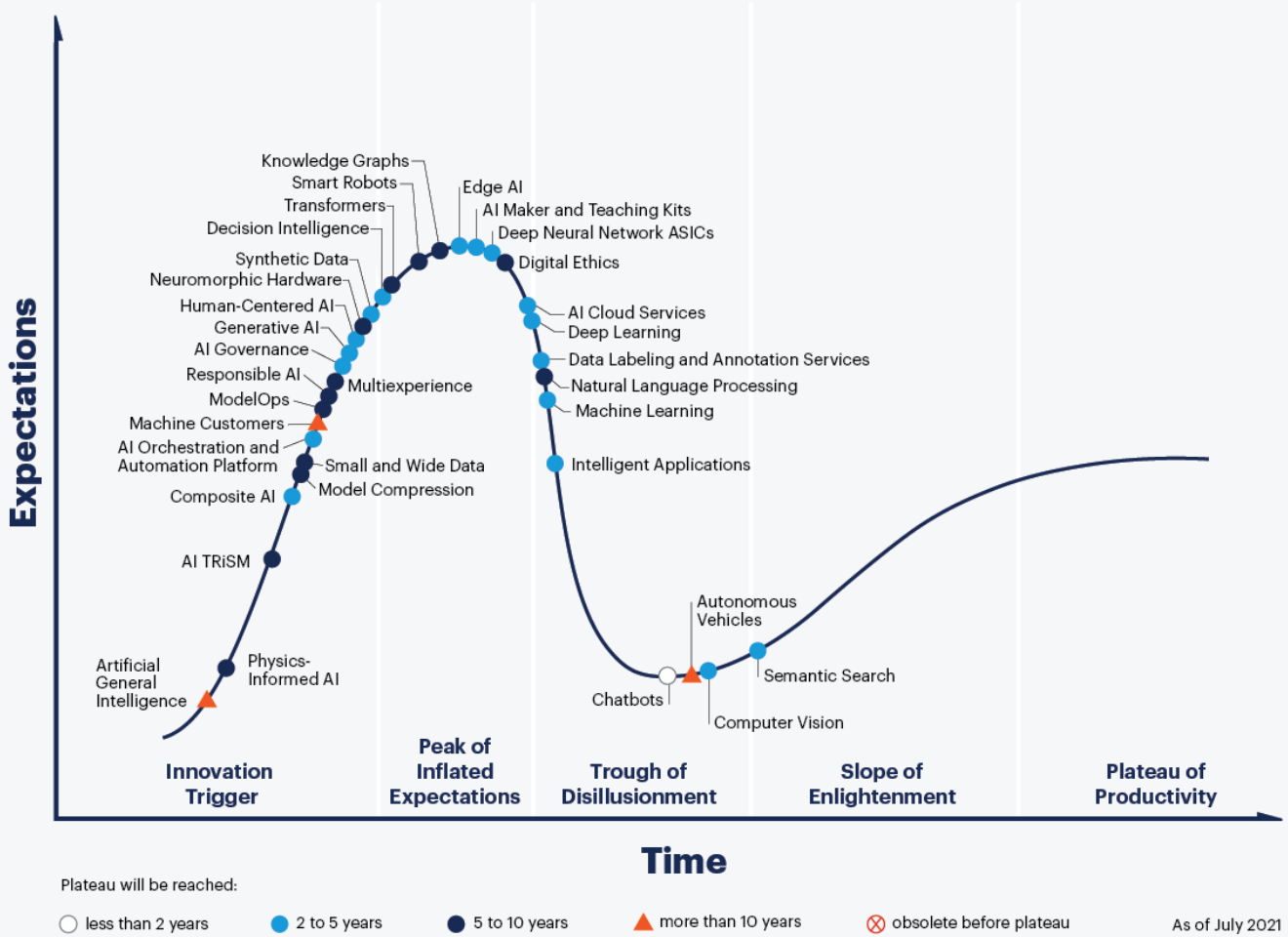
Gartner's Hype Curve 2014



Source: Gartner (July 2016)

Gartner's Hype Curve 2016

Hype Cycle for Artificial Intelligence, 2021



gartner.com

[The 4 Trends That Prevail on the Gartner Hype Cycle for AI, 2021]

Source: Gartner

Gartner

Amazon Web Services

- Amazon EC2
  - Computation Service using VM
- Amazon DynamoDB
  - Large scalable NoSQL databased
  - Fully distributed shared nothing architecture
- Amazon Elastic MapReduce (Amazon EMR)
  - Hadoop based analysis engine
  - Can be used to analyse big data without the need to build the infrastucture

http://aws.amazon.com/big-data/

Challenges

- Developing Big Data Application is not simple
  - New algorithm, new software development tools
- Proper policy about data security and ownership
- Lack of Data Scientists
  - Different from Software Developer

5 Career Paths in Big Data and Data Science

1. Data Engineer
2. Data Management Professional
3. Business Analyst
4. Machine Learning Researcher/Practitioner
5. Data-oriented Professional

http://www.kdnuggets.com/2017/02/5-career-paths-data-science-big-data-explained.html

Non Analytic Career Path

Data Engineer

- Designed and Implement (Infrastructure)

Data Management

- Managing data and the infrastructure
- DBA

Key technologies and skills to focus on:

•Apache Hadoop & its ecosystem
•Apache Spark & its ecosystem
•SQL & relational databases
•NoSQL databases

Analytic Career Path

Business Analyst

- Analysis and presentation of data
- Report, Dashboard
- Business Intelligence
- RDB, Big Data, NoSQL

Key technologies and skills to focus on:

•SQL & relational databases
•NoSQL databases
•Often requires commercial reporting and dashboard package know-how
•Reporting can often be *ad hoc* in nature, and mastery of tools for quickly adapting is key
•Data warehousing

Analytic Career Path

Machine Learning

- Predictive analytics
- not letting the data speak for itself in its current form

Key technologies and skills to focus on:

•Statistics!
•Algebra & calculus
•Programming skills: Python, C++, or some other general-purpose language
•Learning theory
•An understanding of machine learning algorithms (the more algorithms the better, and the deeper the understanding the better!)

Data Scientist / Data-oriented

The data management professional and data engineer were concerned with the infrastructure which houses the data. The business analytics professional is concerned with pulling facts from the data as it exists. The machine learning researcher and practitioner are concerned with advancing and employing the tools available to leverage data for predictive and correlative capabilities, with both roles being algorithm-based (either developing, or utilizing, or both). **The data-oriented professional is concerned primarily with the data, and the stories it can tell, regardless of what technologies or tools are needed to carry out that task.**

Key technologies and skills to focus on:

- Statistics!
- Programming languages: Python, R, SQL
- Data visualization
- Communication skills