BITS Pilani
Pilani | Dubai | Goa | Hyderabad

DSECL ZG 522: Big Data Systems

Session 3 - Big Data Analytics and Systems

Dr. Anindya Neogi

Associate Professor

anindya.neogi@pilani.bits-pilani.ac.in

Topics for today

- Analytics
  - Definitions
  - Maturity model
  - Types
- Big Data Analytics
  - Characterization
  - Adoption challenges
  - Requirements
  - Technology challenges
  - Popular technologies - Hadoop, Spark, Cloud, …
  - Case study - Nasdaq
- Characteristics of Big Data Systems
  - Failures - Reliability and Availability
  - Consistency

Will help you to pitch a Big Data project proposal

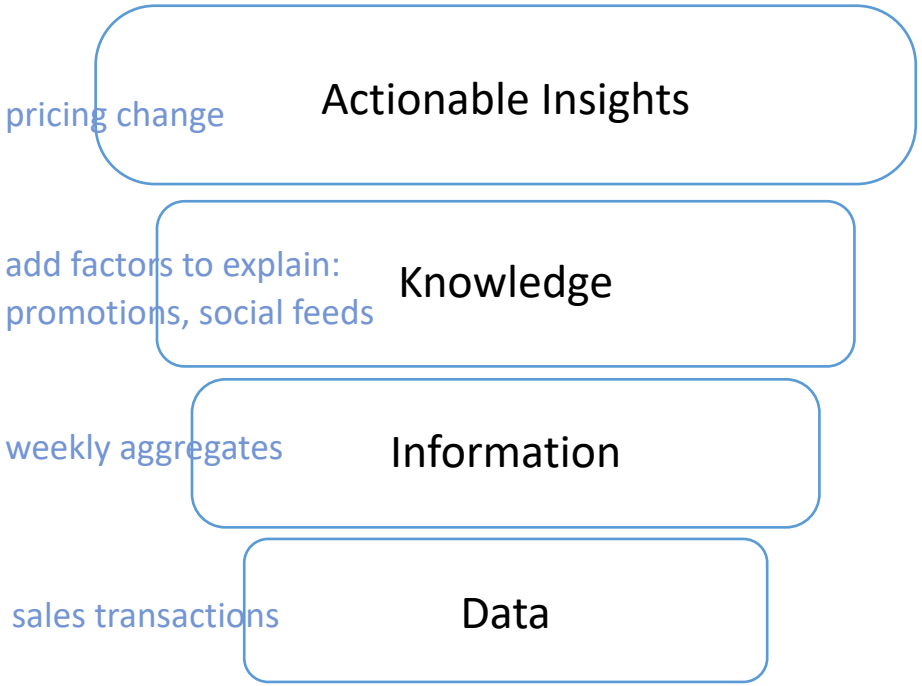
Will help you to build the system

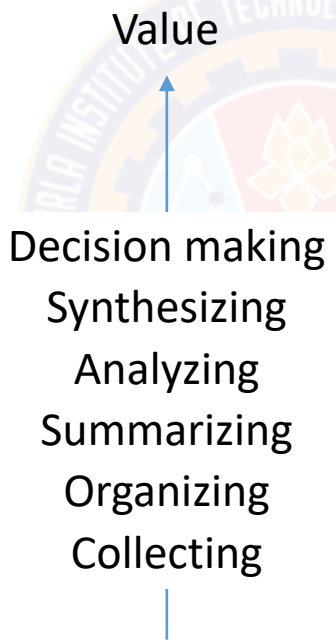
Analytics - Definitions

- Extensive use of data, statistical and quantitative analysis, explanatory and predictive models, and fact based management to drive decisions and actions
- Purpose
  - ✓ To unearth hidden patterns
    - ✓ *2 / 100 stores had no sales for a promotion item because it was not in the right shelf*
  - ✓ To decipher unknown correlations
    - ✓ *The famous "Beer and diapers" story*
  - ✓ Understand the rationale behind trends
    - ✓ *What do users like about a popular product that has growing sales*
  - ✓ Mine useful business information
    - ✓ *Popular items during specific holiday sales*
- Helps in
  - ✓ Effective marketing
  - ✓ Better customer service and satisfaction
  - ✓ Improved operational efficiency
  - ✓ Competitive advantage over rivals

Process of Analysis

Transformation of Data

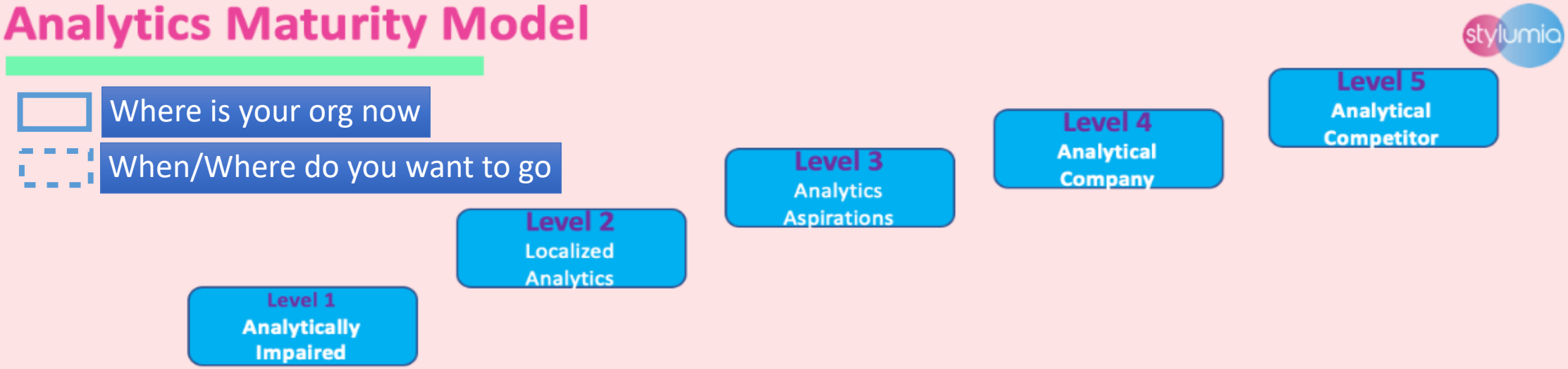




- Apply functions / transformations on data to get to the next level till it is actionable insight useful for the business
- Keep attaching more meta-data for context and make it meaningful

Analytics Maturity Model

Analytics Maturity Model

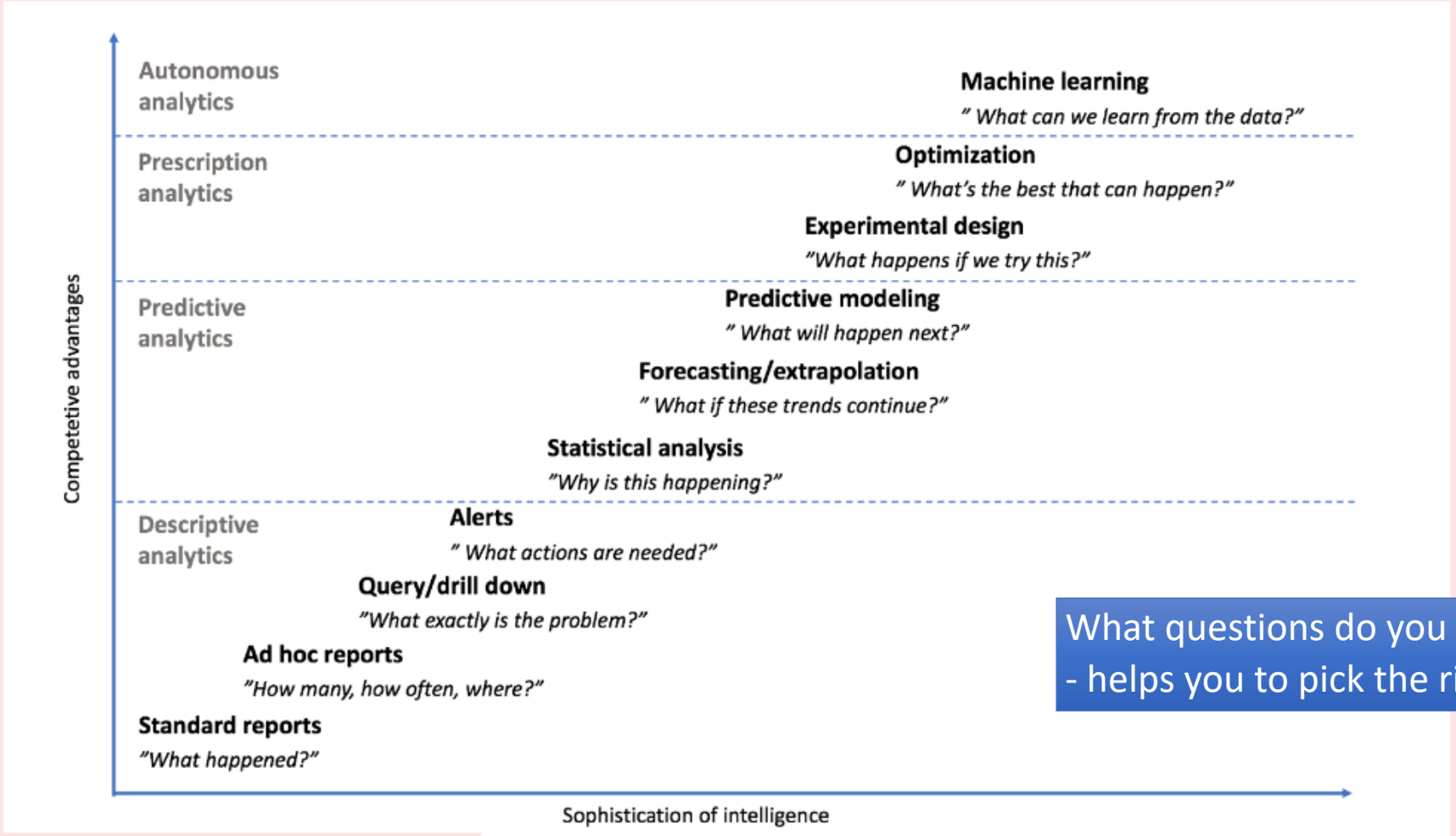


Data	Inconsistent, poor quality	Usable in functional silos	Beginning to create centralized repositories	Integrated data warehouse	Relentless search for new data & metrics
Enterprise	NA	Siloes of data, tech. and talent	Early stages of enterprise-wide approach	Key data, tech. and analysts are centralized	All key analytical resources centrally managed
Leadership	No awareness or interest	Only at functional or process level	Begins to recognize importance of analytics	Support for analytical component	Strong passion for analytics
Targets	NA	disconnected	Efforts aligned to small set of targets	Centered on few key domains	Supports overall strategic objectives
Analysts	Few skills in specific functions	Few isolated analysts	Influx of analysts in key target areas	Specialized analysts in central organization	Mix of analytics experts and amateurs

source: Competing on Analytics, Thomas Davenport

Analytics Maturity and Competitive Advantage

Analytics Maturity & Competitive Advantage



What questions do you want to answer - helps you to pick the right technology

source: Competing on Analytics, Thomas Davenport

Types of Analytics - Descriptive

- Provides ability to **alert, explore and report** using mostly internal and external data
- Business Intelligence (BI) or Performance reporting
- Provides access to historical and current data
- Reports on events, occurrences of the past
- Usually data from legacy systems, ERP, CRM used for analysis
- Based on relational databases and warehouse
- Structured and not very large data sets
- Sometimes also referred as Analytics 1.0
- Era : mid 1950s to 2009

- Questions asked
  - ✓ What happened?
  - ✓ Why did it happen? **(Diagnostic analysis)**

E.g. number of infections is significantly higher this month than last month and highly correlated with factor X

Types of Analytics - Predictive

- Uses **past data to predict** the future
- Uses quantitative techniques like segmentation, forecasting etc. but also makes use of descriptive analytics for data exploration
- Uses technologies like models and rule based systems
- Based on large data set gathered over period of time
- Externally sourced data also used
- Unstructured data may be included
- Hadoop clusters, SQL on Hadoop data etc. technologies used
- aka Analytics 2.0
- Era : from 2005 to 2012

- Key questions
  - ✓ What will happen?
  - ✓ Why it will happen?
  - ✓ When will it happen?

E.g. number of infections will reach X in month Y and likely cause will be Z

Types of Analytics - Prescriptive

- Uses data from past to make prophecies of future and at the same time make **recommendations** to leverage the situation to one's advantage
- Suggests optimal behaviors and actions
- Uses a variety of quantitative techniques like optimization and technologies like models, machine learning and recommendations engines
- Data is blend from Big data and legacy systems, ERP, CRM etc.
- In-memory analysis etc.
- Aka Analytics 3.0 = Descriptive + Predictive + Prescriptive
- post 2012
- Questions:
  - ✓ What will happen
  - ✓ When will it happen
  - ✓ Why will it happen
  - ✓ **What actions should be taken**

E.g. number of infections will reach X in month Y and likely cause will be Z. W is the best recommended action to keep the number in month Y below X/2.

Alternative categorisation

- Basic Analytics
  - ✓ Slicing and dicing of data to help with basic insights
  - ✓ Reporting on historical data, basic visualizations etc.
- Operationalized Analytics
  - ✓ Analytics integrated in business processes
- Advanced Analytics
  - ✓ Forecasting the future by predictive modelling
- Monetized Analytics
  - ✓ Used for direct business revenue

Topics for today

- Analytics
  - Definitions
  - Maturity model
  - Types
- Big Data Analytics
  - Characterization
  - Adoption challenges
  - Requirements
  - Technology challenges
  - Popular technologies
  - Case study - Nasdaq
- Characteristics of Big Data Systems
  - Failures - Reliability and Availability
  - Consistency

Big Data Analytics

Working with datasets with huge volume, variety and velocity beyond storage and processing capability of RDBMS

Better , Faster decision in real time

Richer, faster insights into customers, partners and business

Uses Principle Of Locality to move code near to Data

Big Data Analytics

Competitive Advantage

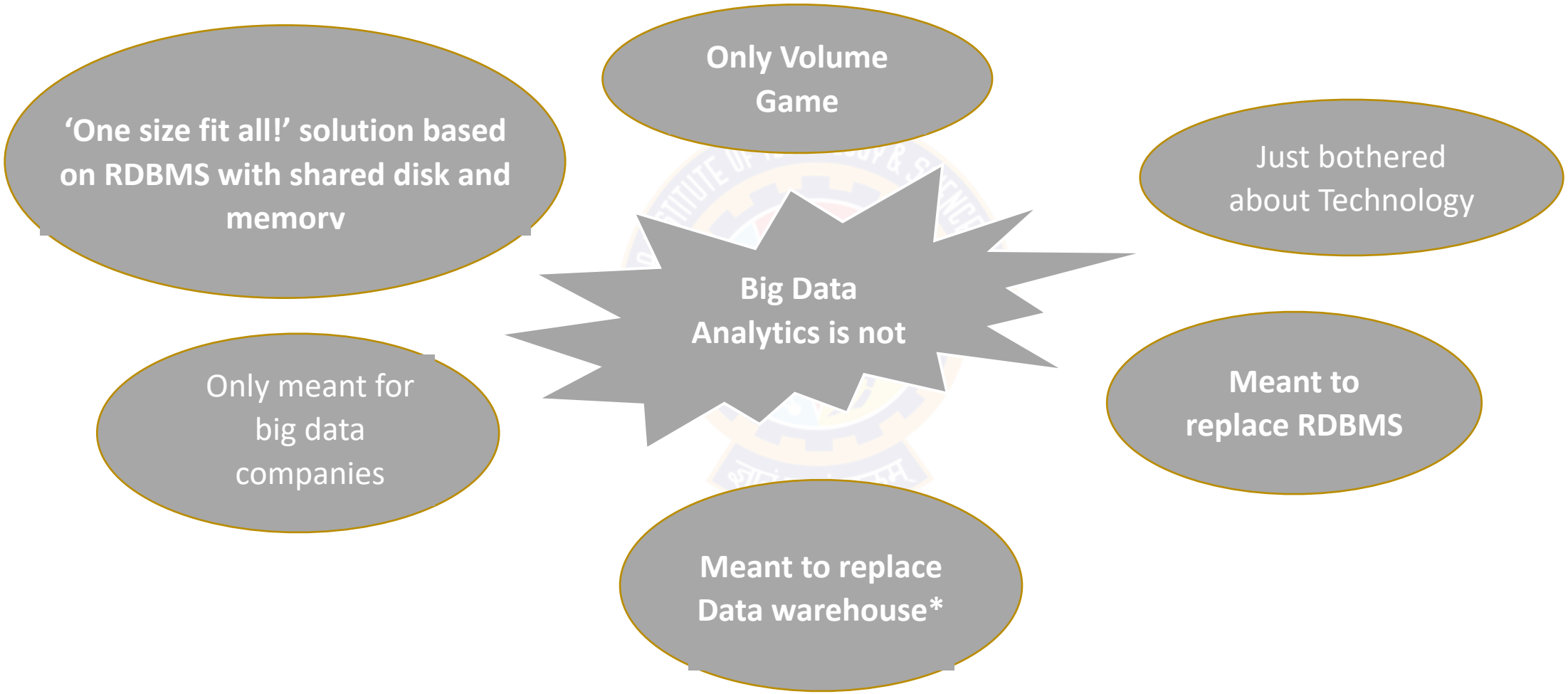
IT's collaboration with business users and Data Scientists

Technology enabled Analytics

Support for both batch and stream processing of data

What makes you think about this differently ?

What Big Data Analytics is not

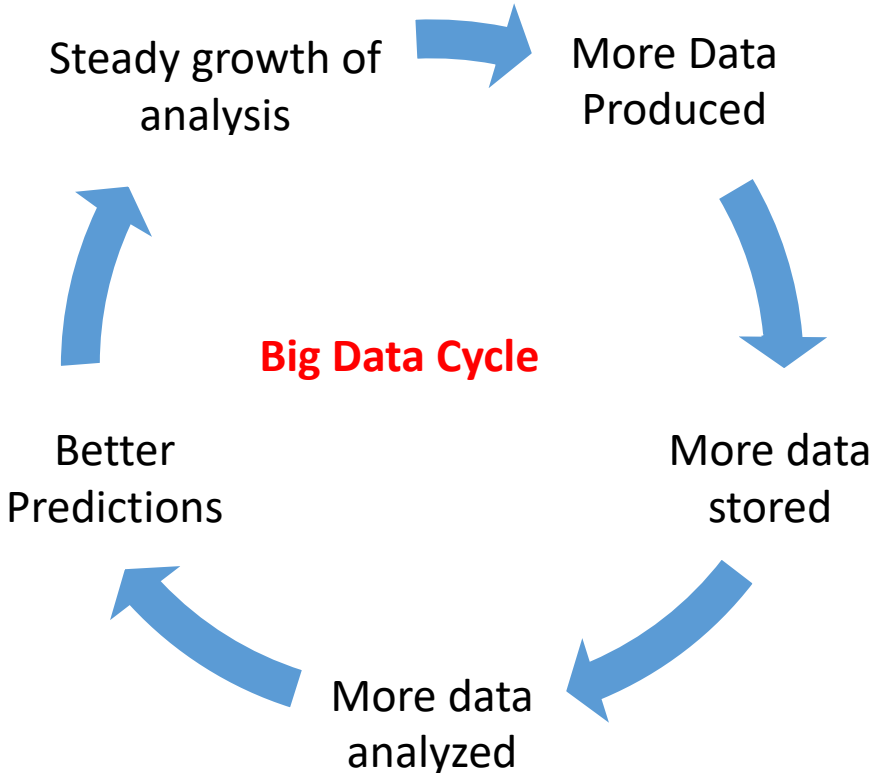


* refer Nasdaq case study

Things to know to avoid friction in Big Data projects

Why the sudden hype ?

- **Data is growing** at 40% compound annual rate
  - ✓ 45 ZB in 2020
  - ✓ In 2010, 1.2 trillion Gigabytes data generated
  - ✓ In 2012, reached to 2.4 trillion Gigabytes
  - ✓ Volume of world wide data expected to double every 1.2 years
  - ✓ Every day 2.5 quintillion bytes of data is created
  - ✓ 90% of today's data is generated in last few years only!
  - ✓ Walmart processes one million customer transaction per hour
  - ✓ 500 million "tweets" are posted by users every day
  - ✓ 2.7 billion "likes" and comments by Facebooks users per day
- **Cost of storage** has hugely dropped
- Large number of **user friendly analytics tools** available for data processing



Adoption Challenges in Organizations

- Obtaining **executive sponsorship for investments** in big data and its related activities
- Getting **business units to share data** / information across organizational silos
- Finding right **skills** (Business Analysts/Data Scientists and Data Engineers) that can manage large amount of variety of data and create insights from it
- Determining **approach to scale** rapidly and elastically , address storage and processing of large volume, velocity and variety of Big data
- Deciding whether to use **structured or unstructured**, **internal or external** data to make business decisions
- Choosing **optimal way to report findings** and analysis of big data
- Determining **what to do with the insights** created from big data

Requirements of Big Data analytics

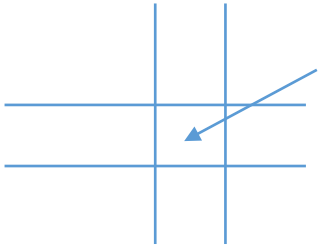
- Cheap abundant storage
- Processing options
  - batch / streaming,
  - disk based / memory based
- Open source platforms, e.g. Hadoop, Spark
- Parallel and distributed systems with high throughput rather than low latency
- Cloud or other flexible resource allocation arrangements
  - Flexibility to setup and tear down infrastructure for quick projects across various teams (e.g. Nasdaq AWS EMR case study discussed later)

Technology challenges (1)

- **Scale**
  - ✓ Need is to have storage that can best withstand large volume, velocity and variety of data
  - ✓ Scale vertically / horizontally ?
  - ✓ RDBMS / NoSQL ?
  - ✓ How does compute scale with storage - coupled or de-coupled, i.e. good idea to put common nodes for compute and storage (refer Nasdaq case)

compute + data on same node: locality helps, but does compute scale with storage ?

- **Security** *
  - ✓ Most of recent NoSQL big data platforms have poor security mechanisms, e.g. challenges:
    - ✓ Fine grain control in semi-structured data, esp with columnar storage
    - ✓ Options for inconsistent data complicate matters
    - ✓ Larger attack surface across distributed nodes
    - ✓ Often encryption is turned off for performance
  - ✓ Lack of authorization techniques while safeguarding big data
  - ✓ May contain PII data (personally identifiable info)



Easier in RDBMS to control at row / column / cell level with always consistent values in a tightly coupled system

* Ref: NoSQL security

Technology challenges (2)

- **Schema**
  - ✓ Need is to have dynamic schema, static / fixed schemas don't fit
- **Continuous availability**
  - ✓ Needs 24 * 7 * 365 support as data is continuously getting generated and needs to be processed
  - ✓ Almost all RDBMS, NoSQL big data platforms has some sort of downtime
    - ✓ Memory cleanup, replica rebalancing, indexing, …
    - ✓ Most of the large scale NoSQL systems also need weekly maintenance

Technology challenges (3)

- **Consistency**
  - ✓ Should one go for strict consistency or eventual consistency? Is this like social media comments or application needs consistent reads ?
- **Partition Tolerant**
  - ✓ When a system get's partitioned by hardware / software failures. How to build partition tolerant systems ? When faults happen is **consistent** data **available** ?
  - ✓ We will discuss options in CAP Theorem
- **Data quality**
  - ✓ How to maintain data quality – data accuracy, completeness, timeliness etc.?
  - ✓ Do we have appropriate metadata in place esp with semi/un-structured data ?

Popular technologies

- How to manage voluminous, varied, scattered and high velocity data ?
  - ✓ Think beyond an RDBMS depending on use case but not necessarily to replace it

- Some popular technologies
  - ✓ **Distributed and parallel processing** (covered in session 2)
  - ✓ **Hadoop** (more details in session 6-9)
    - ✓ File based large scale parallel data processing tasks
  - ✓ **In-memory computing** (more details in session 13-16)
    - ✓ Usage of main memory (RAM) helps to manage data processing tasks faster
  - ✓ **Big Data Cloud** (more details in session 11)
    - ✓ Helps to save cost and better management of resources using a services model

Introduction to Hadoop

What problems does Hadoop solve

Storage of huge amount of data

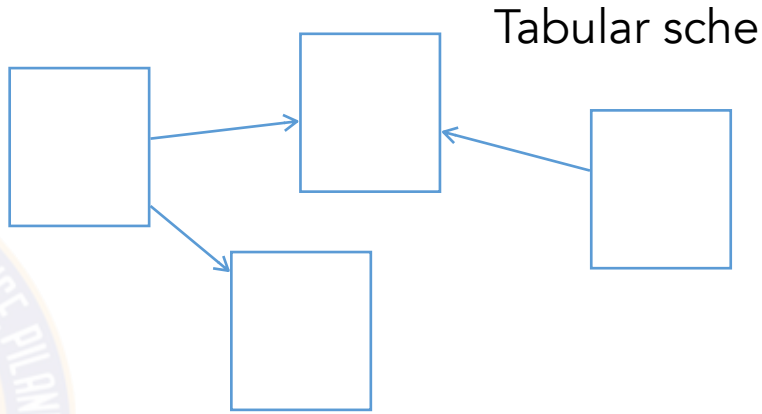
- ✓ Problems
  - ✓ Multiple partitions of data for parallel access but **more systems means more failures**
  - ✓ Multiple nodes can make the system **expensive**
  - ✓ **Arbitrary** data - binary, structured …
- ✓ Solution
  - ✓ Replication Factor (**RF**) for failures : Number of data copies of a given data item / data block stored across the network
  - ✓ Uses **commodity** heterogenous hardware
  - ✓ **Multiple** file formats

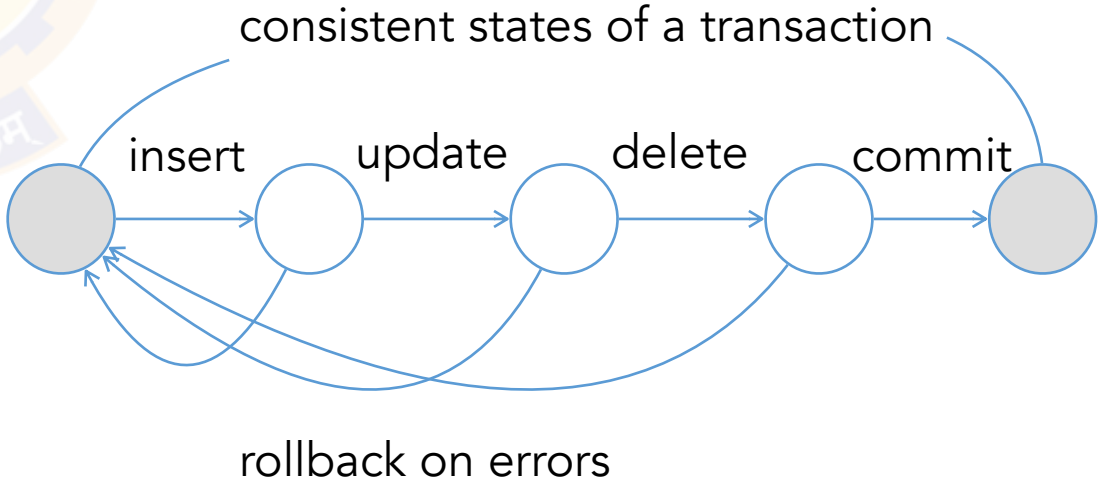
Processing the huge amount of data

- ✓ Problems
  - ✓ Data is **spread** across systems, how to process it in quick manner?
  - ✓ Challenge is to **integrate** data from different machines before processing
- ✓ Solution
  - ✓ **MapReduce** programming model to process huge amount of data with high throughput
  - ✓ Compute is **close** to storage for handling large data sets

What's different from a Distributed Database

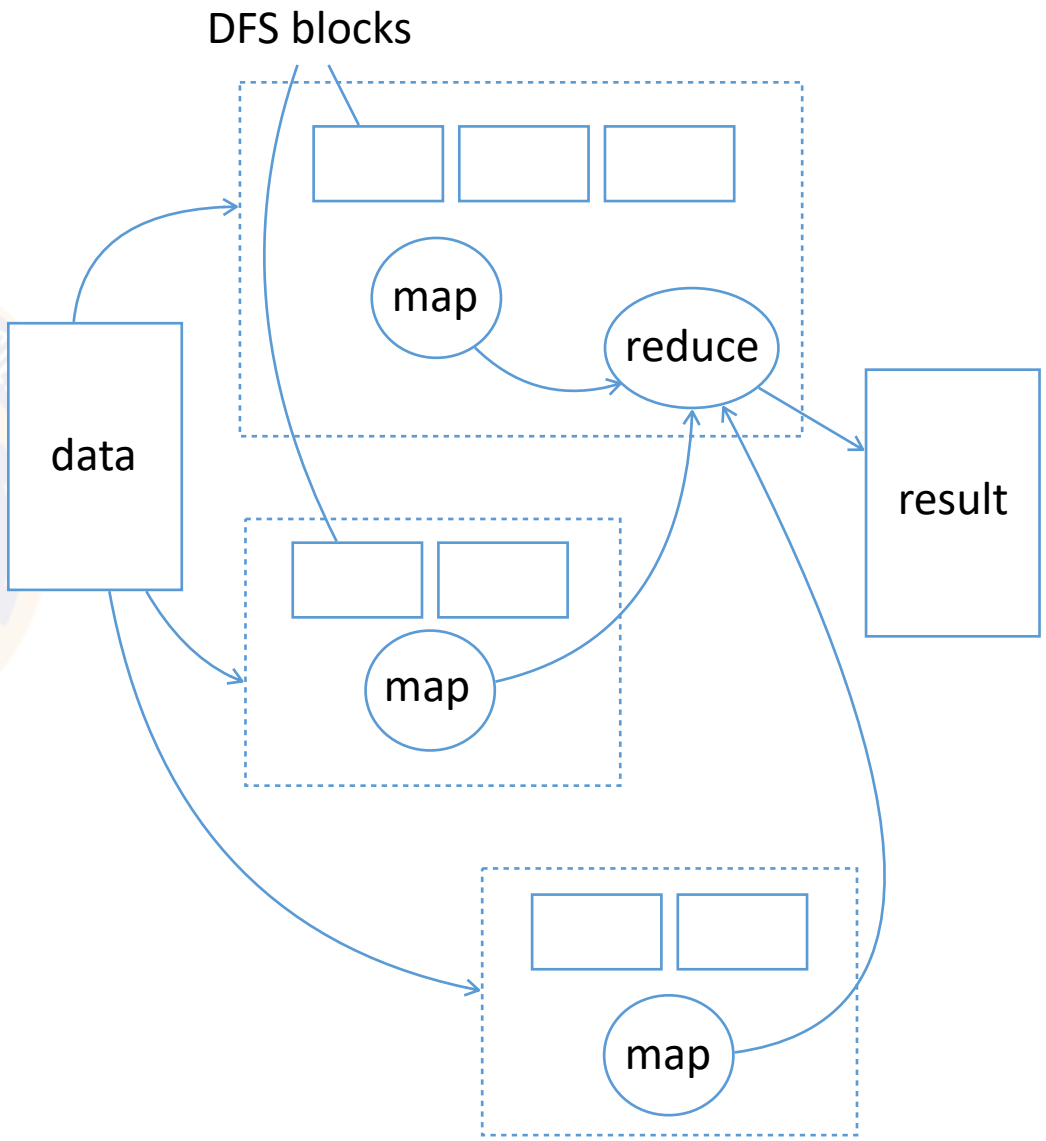
- Distributed Databases
  - Data model
    - Tables and relations
    - Schema is predefined (during write)
    - Supports partitioning
    - Fast indexed reads
    - Read and write many times
  - Compute model
    - Generate notations of a transaction
    - ACID properties
    - Allow distributed transactions
    - OLTP workloads



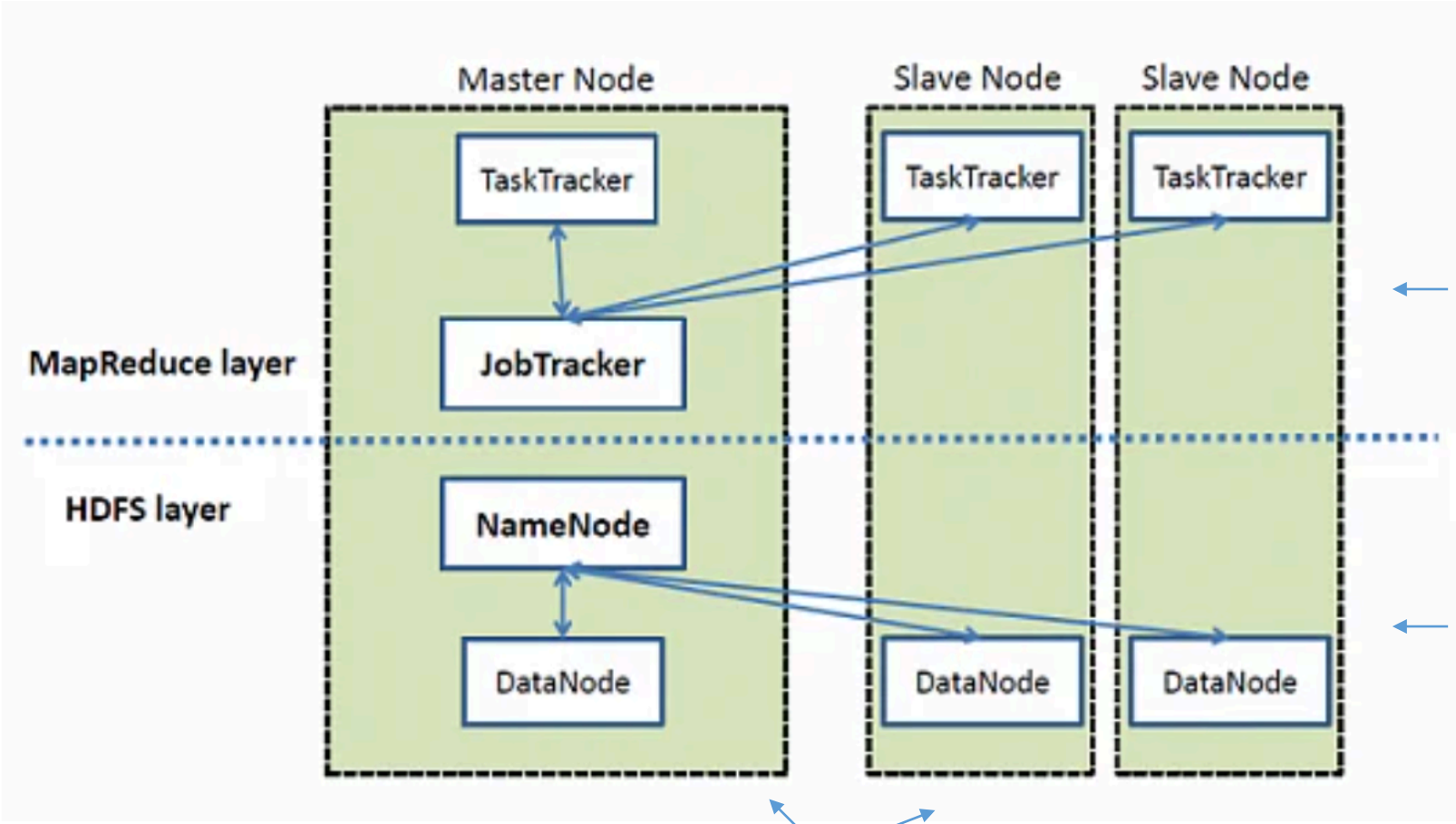


Hadoop in contrast …

- Data model
  - Flat files supporting multiple formats, including binary
  - No pre-defined schema (i.e. during write)
  - Divides files automatically into blocks
  - Handles large files
  - Optimized for write
  - Write once and read many times workload
  - Meant for scan workloads with high throughput
- Compute model
  - Generate notations of a job divided into tasks
  - MapReduce compute model
  - Every task is a map or a reduce
  - High latency analytics, data discovery workloads



Hadoop high level architecture



Tasks consisting of Map and Reduce jobs run on the DataNodes.
They are managed by JobTracker and TaskTrackers.

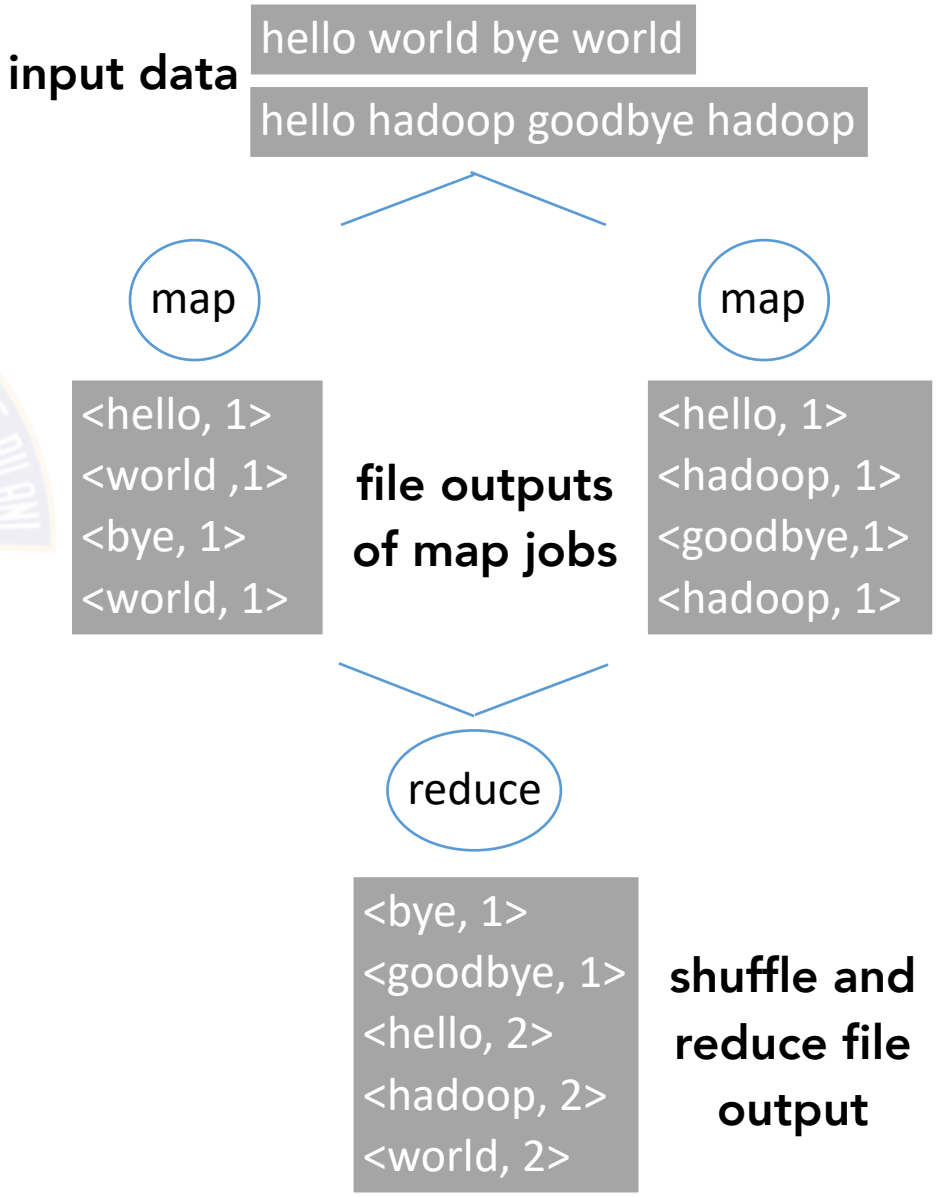
Data is partitioned into files on DataNodes.
NameNode is the file system management node.

Hadoop cluster of DataNodes that also run compute Tasks having one Master Node for management and control at compute task and file system layers.

Example - Word count

```
public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
```

```
public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
```



Advantages of using Hadoop

- Low cost – open source and low cost commodity storage
- Computing power – many nodes can be used for computation
- Scalability – simple to add nodes in system for parallel processing and storage
- Storage Flexibility – can store unstructured data easily
- Inherent data protection – protects against hardware failures

Hadoop ecosystem

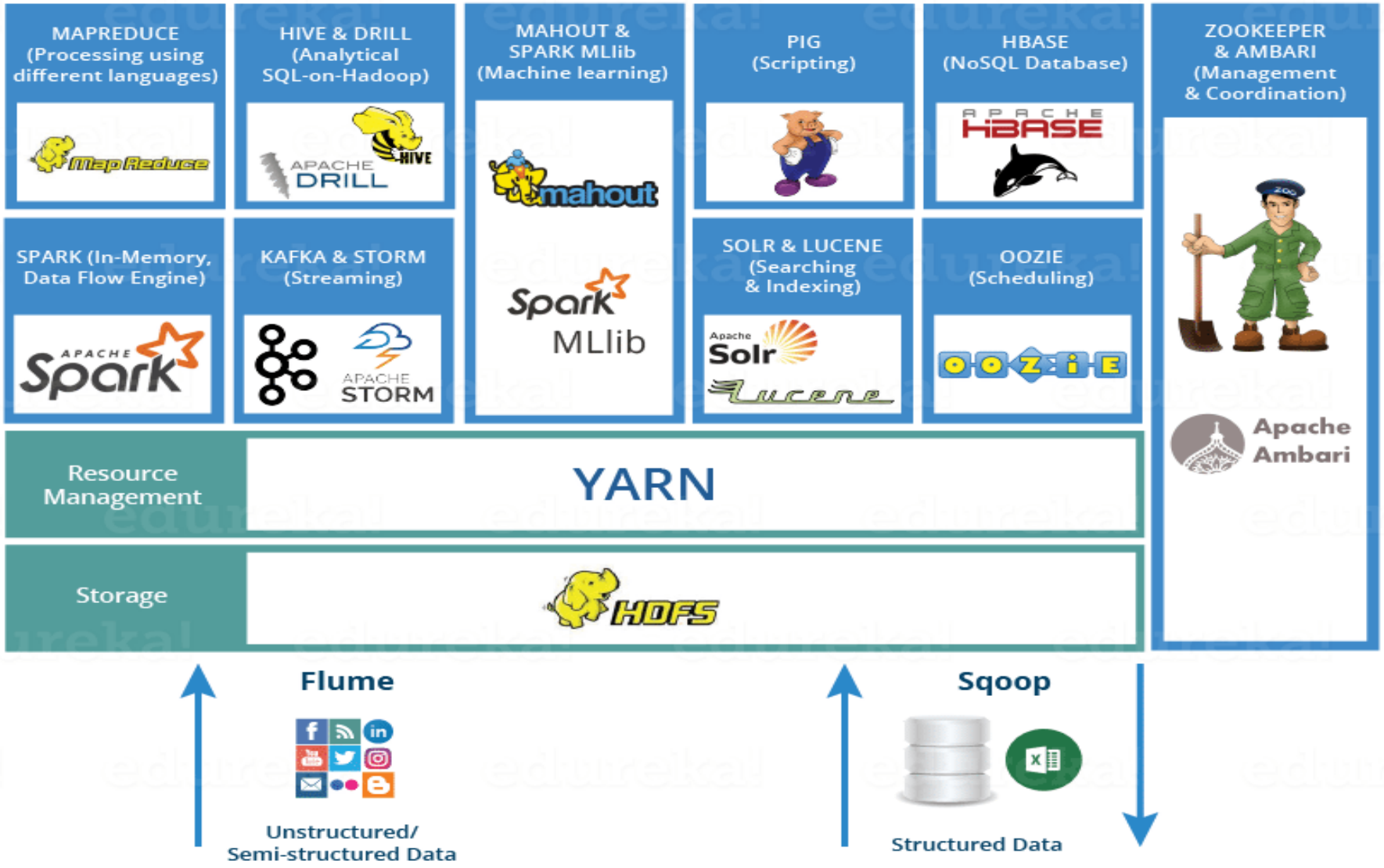


Image Source : [Edureka]

Introduction to in-memory computing

Issues with MapReduce on Hadoop

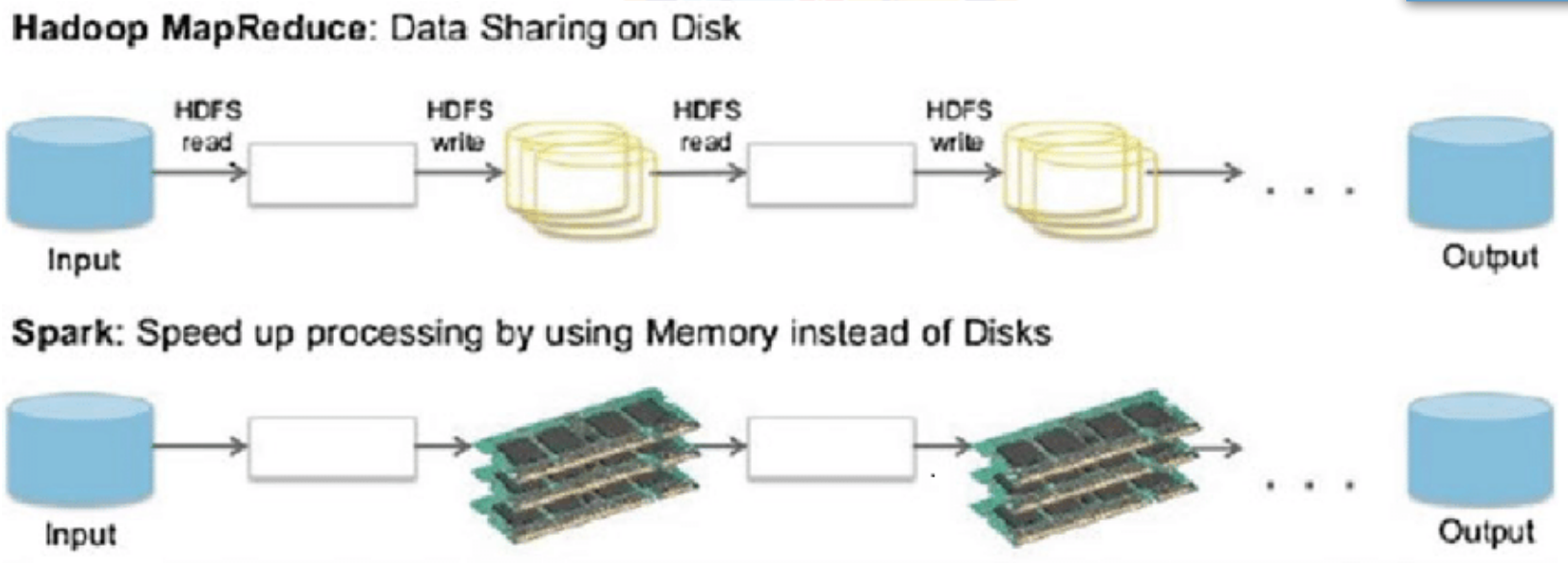
- ✓ Revolutionized big data processing, enabling users to store and process huge amounts of data at very low costs.
- ✓ An ideal platform to implement **complex batch applications** as diverse as
  - sifting through system logs
  - running ETL
  - computing web indexes
  - recommendation systems etc.
- ✓ Its reliance on **persistent storage** to provide fault tolerance and its **one-pass computation** model make **MapReduce a poor fit for**
  - low-latency applications
  - iterative computations, such as machine learning and graph algorithms
    - There are extensions for iterative MapReduce that we study later

Adapted from : https://databricks.com/blog/2013/11/21/putting-spark-to-use.html

In-Memory computing

- In-memory computing
  - ✓ means using a type of middleware software that allows one to store data in RAM, across a cluster of computers, and process it in parallel
- For example,
  - ✓ Operational datasets typically stored in a centralized database which you can now store in **"connected" RAM across multiple computers**.
  - ✓ RAM is roughly 5,000 times faster than traditional spinning disk.
  - ✓ Native support for parallel processing makes it faster

Note:
Could be batch or streaming



Fast and easy big data processing with Spark

- At its core, Spark provides a general programming model that enables developers to write application by composing arbitrary operators, such as
  - ✓ mappers
  - ✓ reducers
  - ✓ joins
  - ✓ group-bys
  - ✓ filters
- This composition makes it easy to express a **wide array of computations**, including iterative machine learning, streaming, complex queries, and batch.
- Spark keeps track of the data that each of the operators produces, and enables applications to **reliably store this data in memory** using RDDs*.
  - ✓ This is the key to Spark's performance, as it allows applications to avoid costly disk accesses.

** RDD: Resilient Distributed Dataset*

Example - Word count

```
sparkContext.textFile("hdfs://...")
  .flatMap(line => line.split(" "))
  .map(word => (word, 1))
  .reduceByKey(_ + _)
  .saveAsTextFile("hdfs://...")
```

convert a file into lines

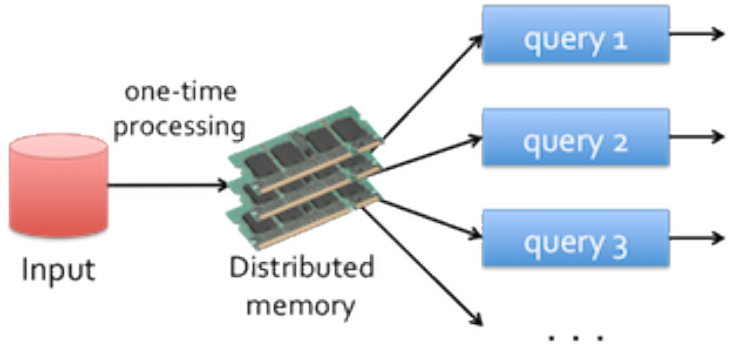
map: transform each word into <k,v> pair

reduce: sum up values for each key

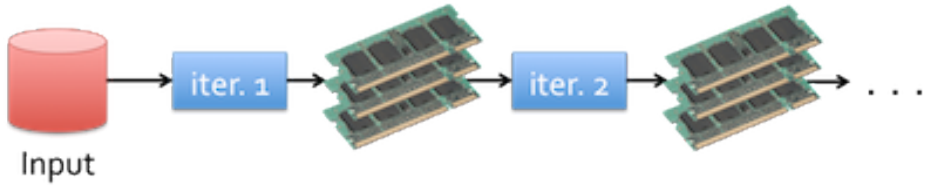
- Can read data from many sources, including HDFS
- Map / reduce output is written to memory instead of files
- Memory content can be written out to files etc.
- A rich set of primitives on top of MapReduce model to make it easier to program

Ideal Apache Spark applications

- Low-latency computations
  - ✓ by caching the working dataset in memory and then performing computations at memory speeds
- Efficient iterative algorithm
  - ✓ by having subsequent iterations share data through memory, or repeatedly accessing the same dataset



(a) Low-latency computations (queries)



(b) Iterative computations

Introduction to Cloud Computing

Definition

The US National Institute of Standards (NIST) defines cloud computing as follows:

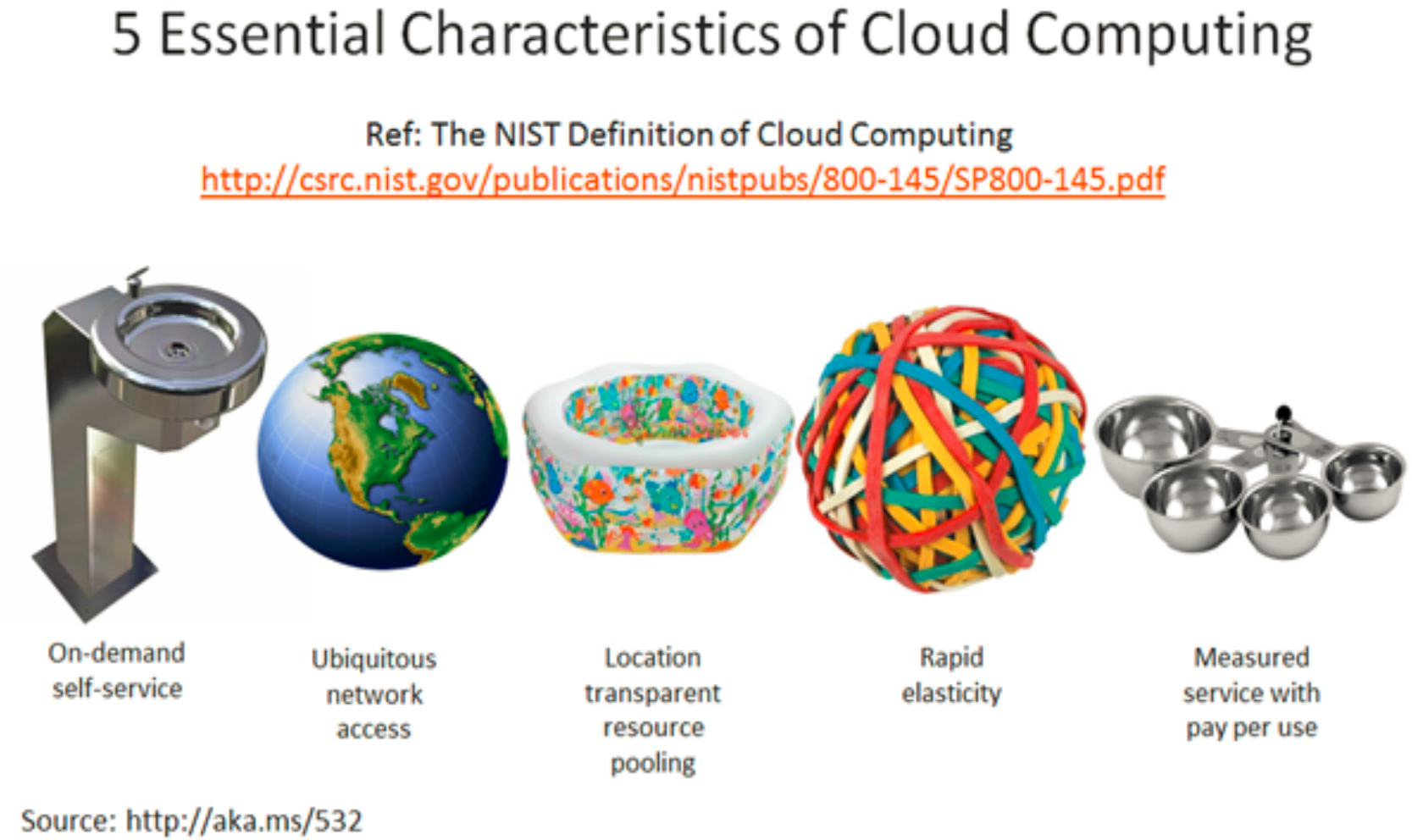
Cloud computing is a model for enabling **ubiquitous**, **convenient**, **on-demand** network access to a **shared** pool of **configurable** computing **resources** (e.g., networks, servers, storage, applications, and services) that can be **rapidly** provisioned and released with **minimal** management effort or service provider interaction.

NIST's 3-4-5 rule of Cloud Computing

- 3 cloud service models or service types for any cloud platform
- 4 deployment models
- 5 essential characteristics of cloud computing infrastructure

5 Characteristics of Cloud Computing

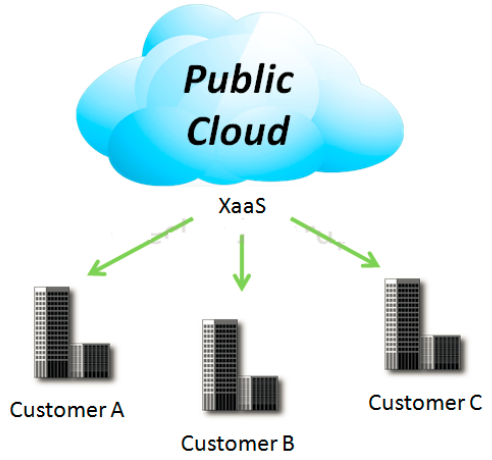
- On demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service



4 Deployment models

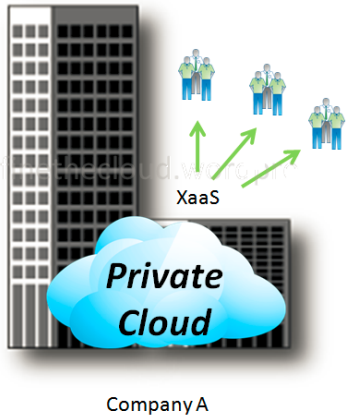
Public Cloud

Mega-scale cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services. E.g. AWS, Azure, Google, …



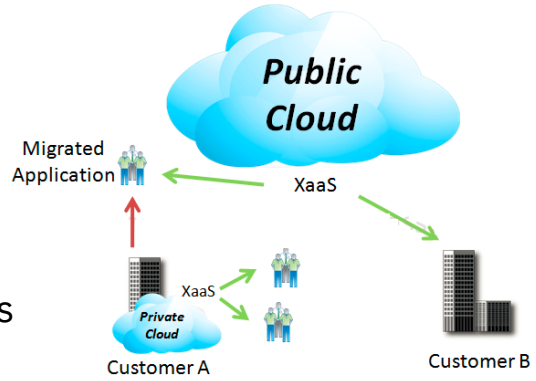
Private Cloud

The cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on premise or off premise.



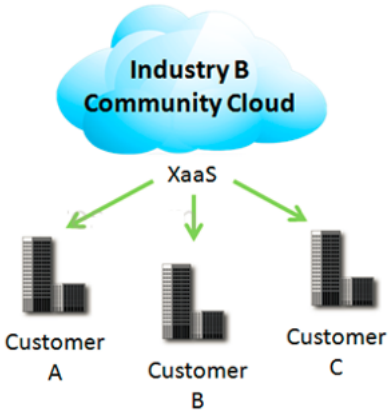
Hybrid Cloud

The cloud infrastructure is a composition of two or more clouds (private or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability, cross domain security etc.

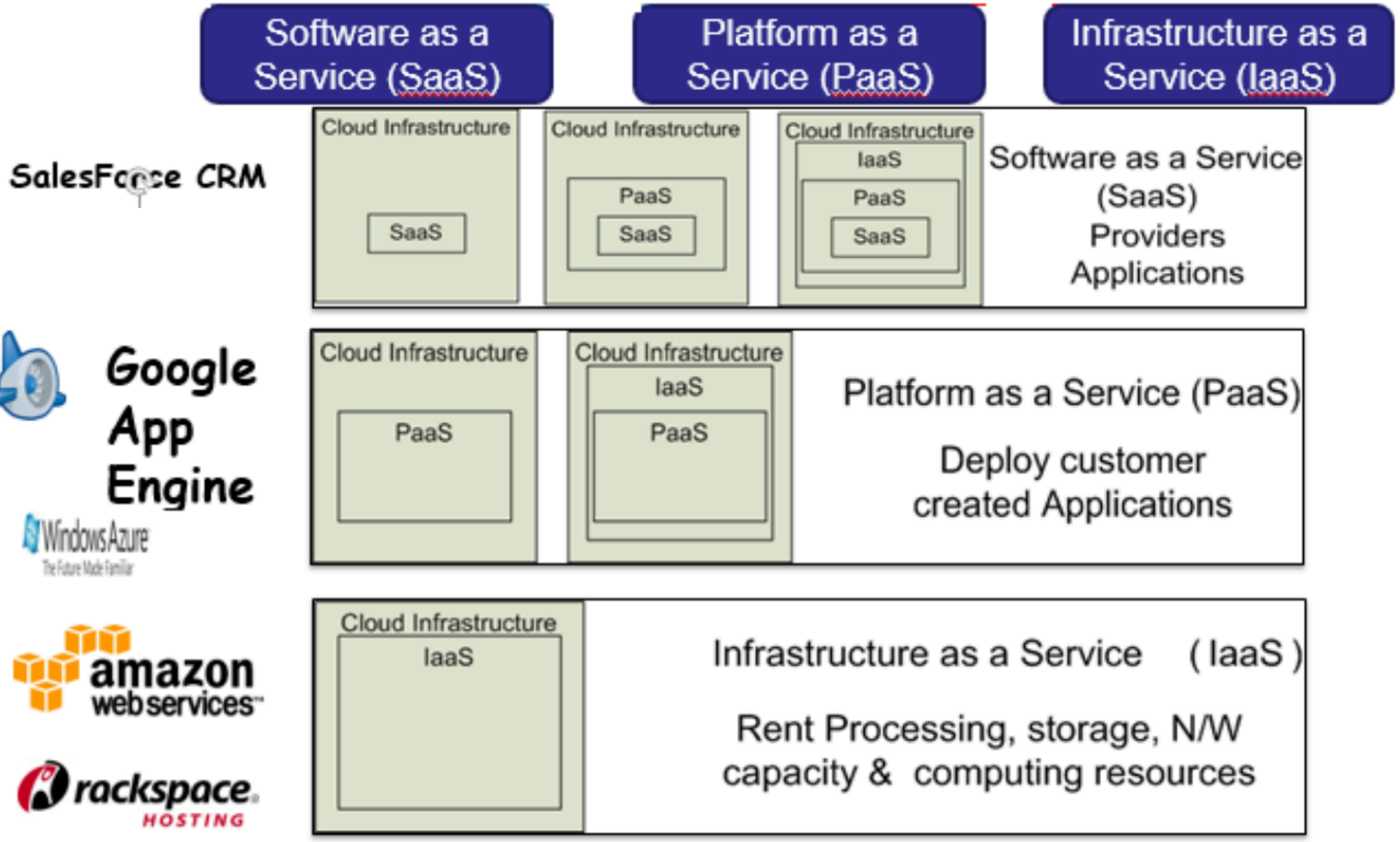


Community Cloud

An 'infrastructure shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party and may exist on premise or off premise' according to NIST.



3 Cloud Service Models



Cloud services for Big Data

- Cloud follows same model as Big Data, both requiring distributed clusters of computing devices.
  - ✓ Cloud Computing considered as ideal platform for handling big data

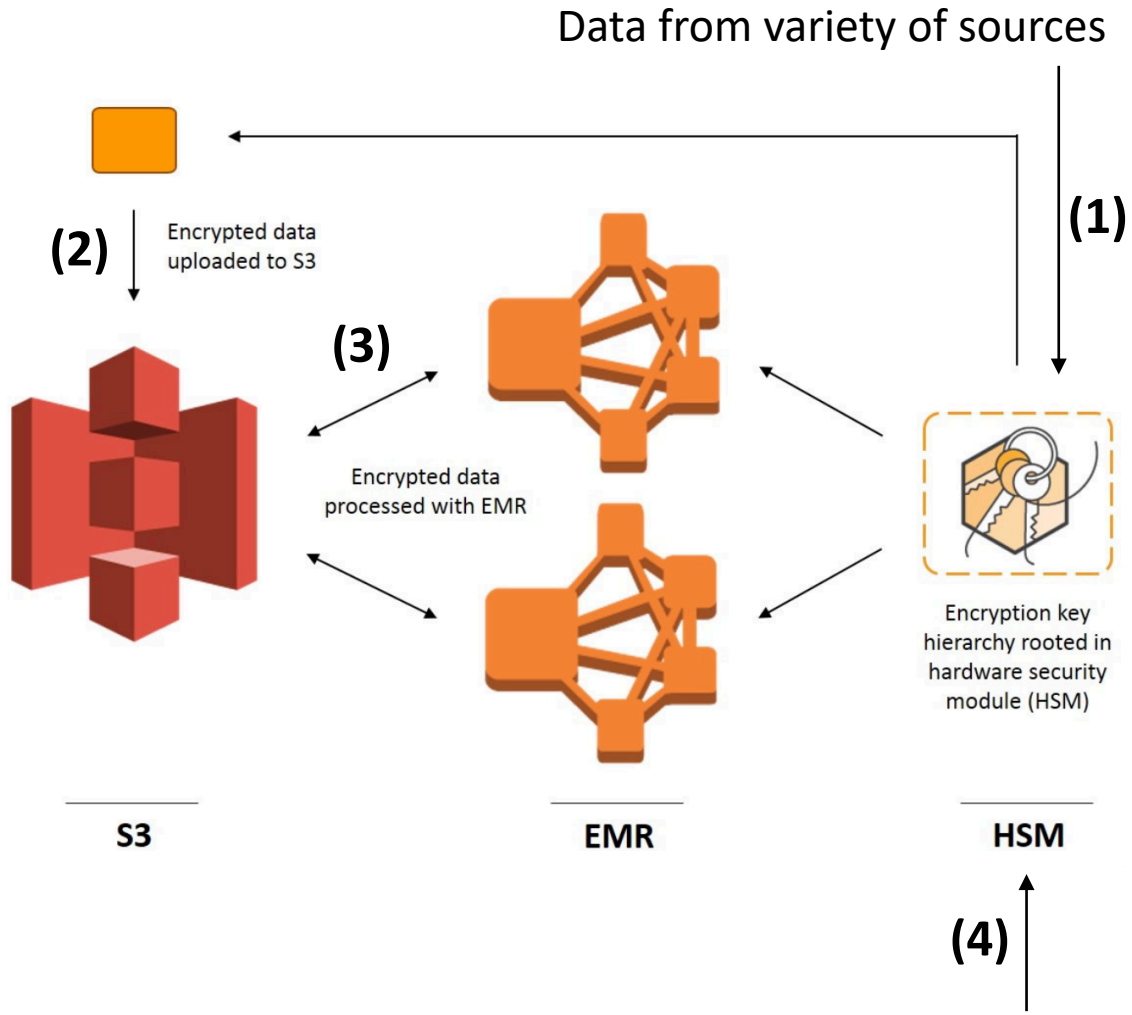
- IaaS:
  - ✓ Can provide huge storage and computational power requirements for Big Data through limitless storage and computing ability of cloud computing, e.g. AWS S3, EC2
- PaaS:
  - ✓ Vendors offers platforms ready with Hadoop and MapReduce (AWS EMR).
  - ✓ Saves hassles of installations and managements of these environments
- SaaS:
  - ✓ Great help to organizations which requires specialized software's for big data like for social media analytics, feedback monitoring etc.
  - ✓ SaaS vendors provides out of the box solution for such common use cases

Cloud providers in Big Data market

- Amazon (Amazon Web Services AWS)
  - ✓ EC2
  - ✓ Elastic MapReduce
  - ✓ DynamoDB
  - ✓ Amazon S3
  - ✓ High Performance Computing
  - ✓ Redshift
- Google (Google Cloud Platform GCP)
  - ✓ Google Compute Engine
  - ✓ Google BigQuery
  - ✓ Google Prediction API
- Windows Azure
  - ✓ Azure PaaS cloud based on Windows and SQL
  - ✓ Windows Azure HD Insight

Case study: Nasdaq

- Operates financial exchanges
- AWS Redshift is the data warehouse with 5.5B rows/day with peak 14B/ day in Oct 2014.
- A new DWH environment is setup using EMR and S3
  - give more teams access to gigantic data sets
  - cost efficiency
- Hadoop enables large and more varied historical data access at lower cost
- Why EMR and S3 ?
  - On demand set up of Hadoop clusters on Cloud to run short term projects.
  - Existing tech to move around TBs of data.
  - Storage on Cloud in S3 to scale to very large data sets.
- Why not HDFS and use S3 ?
  - Decouple data and compute because data size is disproportionately higher than compute frequency - so can't add Hadoop nodes just for data esp with replication factor
  - Similar to Netflix. EMRFS layer can enable compute nodes access data in S3



Nasdaq case study

Topics for today

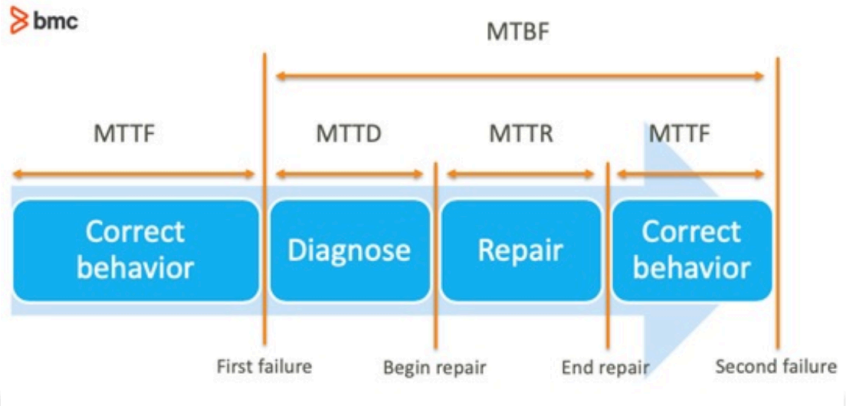
- Analytics
  - Definitions
  - Maturity model
  - Types
- Big Data Analytics
  - Characterization
  - Adoption challenges
  - Requirements
  - Technology challenges
  - Popular technologies
  - Case study - Nasdaq
- Characteristics of Big Data Systems
  - Failures - Reliability and Availability
  - Consistency

Distributed computing – living with failures

- Failures of nodes and links is a common concern in Distributed Systems
  - Essential to have fault tolerance aspect in design
- Fault tolerance is a measure of
  - How a distributed system functions in the presence of failures of system components
  - Tolerance of component faults is measured by 2 parameters
    - **Reliability** - An inverse indicator of failure rate
      - How soon a system will fail
    - **Availability** - An indicator of fraction of time a system is available for use
      - System is not available during failure

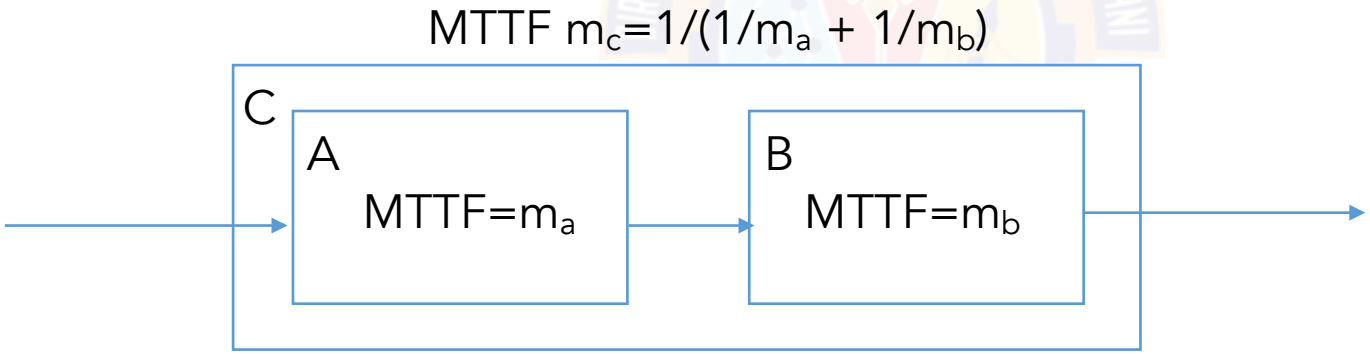
Metrics

- **MTTF** - Mean Time To Failure
  - MTTF = 1 / failure rate = Total #hours of operation / Total #units
  - MTTF is an averaged value. In reality failure rate changes over time because it may depend on age of component.
- **Failure rate** = 1 / MTTF (assuming average value over time)
- **MTTR** - Mean Time to Recovery / Repair
  - MTTR = Total #hours for maintenance / Total #repairs
- **MTTD** - Mean Time to Diagnose
- **MTBF** - Mean Time Between Failures
  - MTBF = MTTD + MTTR + MTTF



Reliability - serial assembly

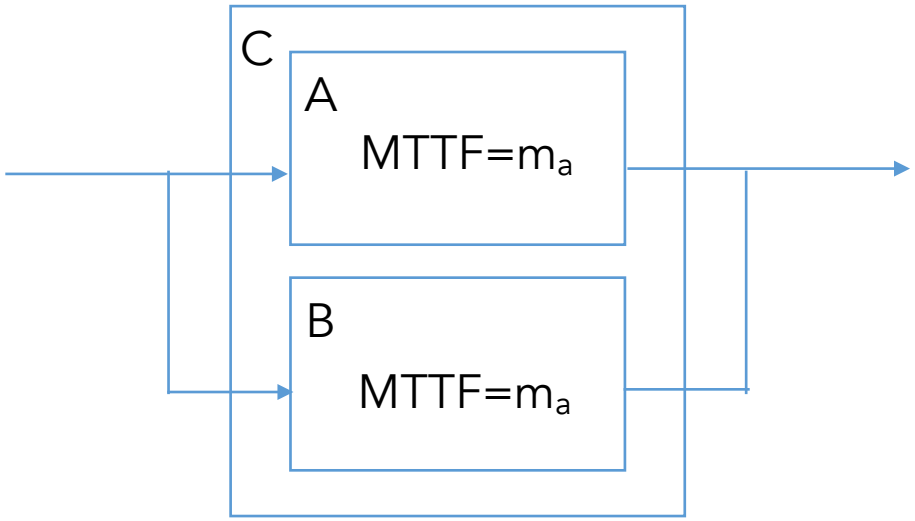
- MTTF of a system is a function of MTTF of components
- Serial assembly of components
  - Failure of any component results in system failure
  - Failure rate of C = Failure rate of A + Failure rate of B = $1/m_a + 1/m_b$



- **MTTF of system = 1 / SUM ($1/MTTF_i$)** for all components i
- **Failure rate of system = SUM($1/MTTF_i$)** for all components i

Reliability - parallel assembly

- In a parallel assembly, e.g. a cluster of nodes
  - MTTF of C = MTTF A + MTTF B because both A and B have to fail for C to fail
- **MTTF of system = SUM($MTTF_i$)** for all components i



MTTF $m_c = m_a + m_b$

Availability

- Availability = Time system is UP and accessible / Total time observed
- **Availability = MTTF / (MTTD* + MTTR + MTTF)**

  or

- Availability = MTTF / MTBF
- A system is highly available when
  - MTTF is high
  - MTTR is low

* Unless specified one can assume MTTD = 0

Example

- A node in a cluster fails every 100 hours while other parts never fail.
- On failure of the node the whole system needs to be shutdown, faulty node replaced and system. This takes 2 hours.
- The application needs to be restarted, which takes 2 hours.
- What is the availability of the cluster ?
- If downtime is $80k per hour, the what is the yearly cost ?

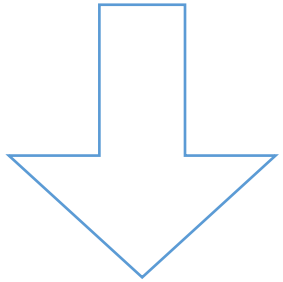
- Solution
  - MTTF = 100 hours
  - MTTR = 2 + 2 = 4 hours
  - Availability = 100/104 = 96.15%
  - Cost of downtime per year = 80000 x 3.85 * 365 * 24 / 100 = USD 27 million

https://www.brainkart.com/article/Fault-Tolerant-Cluster-Configurations_11320/

Fault tolerance configurations - standby options

High availability model	Secondary node behavior	Data protection	Failover time
Load-balanced	Both the primary node and the secondary node are active and they process system requests in parallel. aka active-active	Data replication is bidirectional and is performed based on the software capabilities.	Zero failover time
Hot standby	The software component is installed and available on both the primary node and the secondary node. The secondary system is up and running, but it does not process data until the primary node fails. aka active-passive	Data is replicated and both systems contain identical data. Data replication is performed based on the software capabilities.	A few seconds
Warm standby	The software component is installed and available on the secondary server, which is up and running. If a failure occurs on the primary node, the software components are started on the secondary node. This process is automated by using a cluster manager.	Data is regularly replicated to the secondary system or stored on a shared disk.	A few minutes
Cold standby	A secondary node acts as the backup for an identical primary system. The secondary node is installed and configured only when the primary node breaks down for the first time. Later, in the event of a primary node failure, the secondary node is powered on and the data is restored while the failed component is restarted.	Data from a primary system can be backed up on a storage system and restored on a secondary system when it is required.	A few hours

Decreasing cost
vs
Increasing MTTR

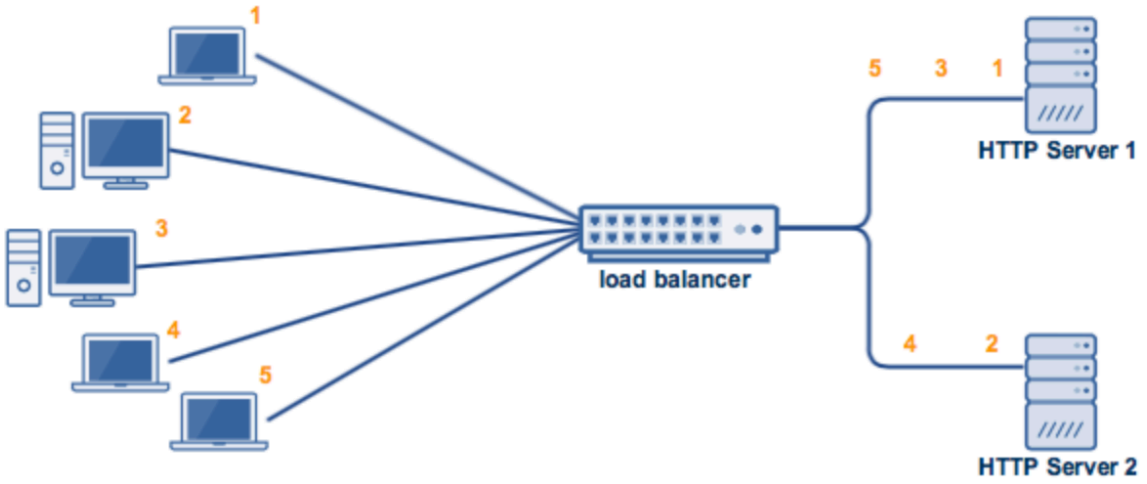


Assumption:
Same software bug or runtime fault will not recur in the standby

https://www.ibm.com/docs/en/cdfsp/7.6.0?topic=systems-high-availability-models

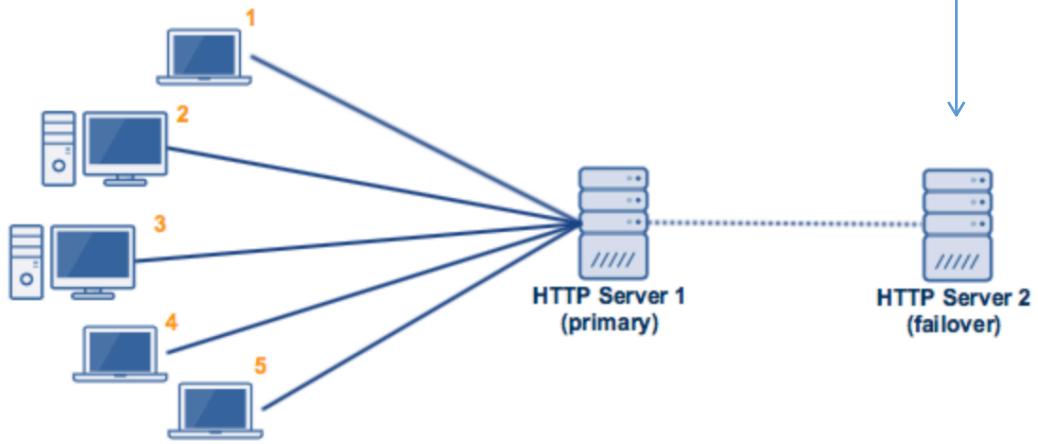
Fault tolerance configurations - standby options

Active-Active



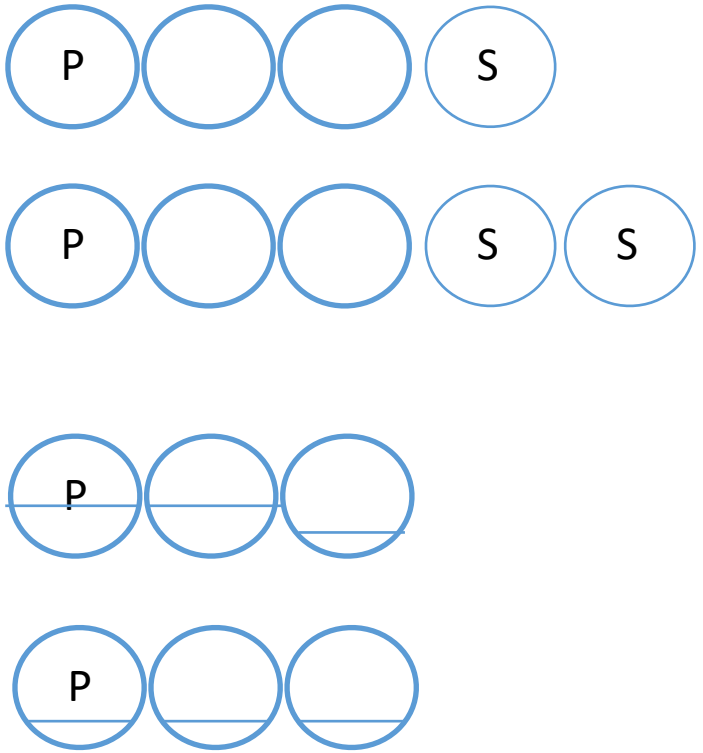
Warm and cold depends on how the passive / failover node is kept updated

Active-Passive



Fault tolerance configurations - cluster topologies

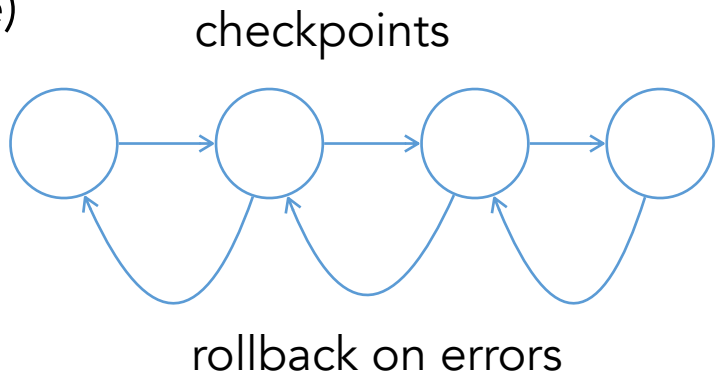
- N+1
  - One node is configured to take the role of the primary
- N+M
  - M standby nodes if multiple failures are anticipated especially when running multiple services in the cluster
- N to 1
  - The secondary failover node is a temporary replacement and once primary is restored, services must be reactivated on it
- N to N
  - When any node fails, the workload is redistributed to the remaining active nodes. So there is no special standby node.



https://www.ibm.com/docs/en/cdfsp/7.6.0?topic=systems-high-availability-models

Fault Tolerant Clusters – Recovery

- **Diagnosis**
  - Detection of failure and location of the failed component, e.g. using heartbeat messages between nodes
- **Backward recovery**
  - periodically do a checkpoint (save consistent state on stable storage)
  - on failure, isolate the failed component, **rollback** to last checkpoint and resume normal operation
  - Ease to implement, independent of application, but leads to wastage of execution time on rollback besides unused checkpointing work
- **Forward recovery**
  - In real-time systems or time-critical systems cannot rollback. So state is reconstructed on the fly from diagnosis data.
  - Application specific and may need additional hardware



Consistency

- Big Data systems write replicas of a shard / partition
- Any write needs to update all replicas
- Any reads can happen in between
- Consistency —
  - Do you allow a read of any replica in any thread to always read the latest value written in any thread ?
    - RDBMS / OLTP systems / Systems of Record
    - ACID (Atomicity, Consistency, Isolation, Durability)
  - Do you allow reads to return any value and eventually show the latest stable value
    - Some BigData systems / Systems of Engagement e.g. social network comments
    - BASE (Basic Availability, Soft state, Eventual consistency)

Ref: https://www.dummies.com/programming/big-data/applying-consistency-methods-in-nosql/

Consistency clarification

- C in ACID is different from C as in CAP Theorem (in next session) *
- **C in ACID of RDBMS based OLTP systems**
  - A broader concept at a data base level for a *transaction involving multiple data items*
  - Harder to achieve
- **C in CAP Theorem for most NoSQL systems**
  - Applies to *ordering of operations for a single data item* not a transaction involving multiple data items
  - So it is a *strict subset* of C in ACID
  - Typically support of ACID semantics for NoSQL systems defeats the purpose as it involves having a single transaction manager that becomes a bottleneck for large scale sharding to scale

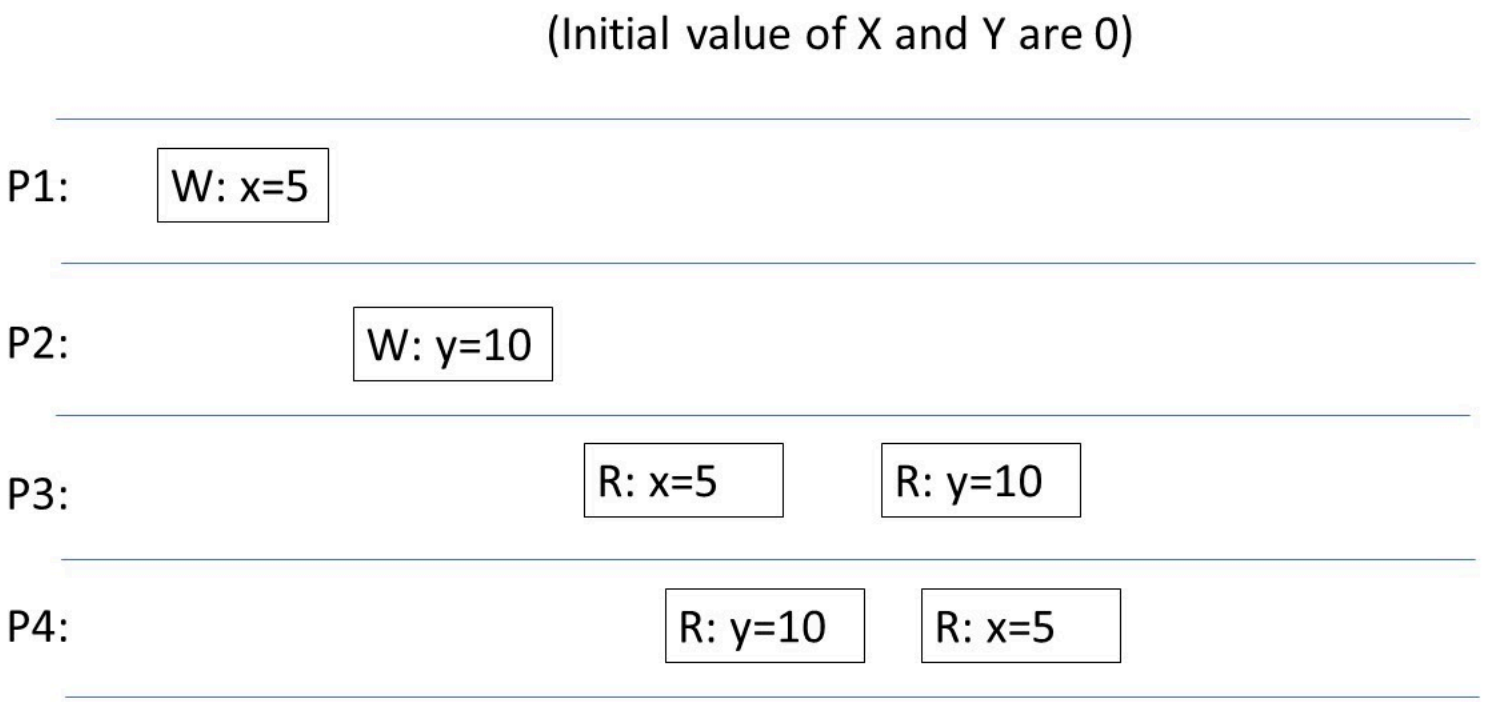
* https://hackingdistributed.com/2013/03/23/consistency-alphabet-soup/#:~:text=%22C%20as%20in%20ACID%22%20is,arbitrarily%20large%20groups%20of%20objects

Levels of Consistency

- Strict
  - Requires **real time line ordering of all writes** - assumes actual write time can be known. So "reads" read the latest data in real time across threads.
- Linearisable
  - Acknowledges that **write requests take time to write all copies** - so does not impose ordering within overlapping time periods of read / write
- Sequential
  - All **writes across threads for all data items are globally ordered**. All threads must see the same order. But does not need real-time ordering.
- Causal
  - Popular and useful model where only **causally connected writes and reads need to be ordered**. So if a write of a data item (Y) happened after a read of same or another data item (X) in a thread then all threads must observe write X before write to Y.
- Eventual (most relaxed as in BASE)
  - If there are no writes for "some time" then all **threads will eventually agree on a latest value** of the data item

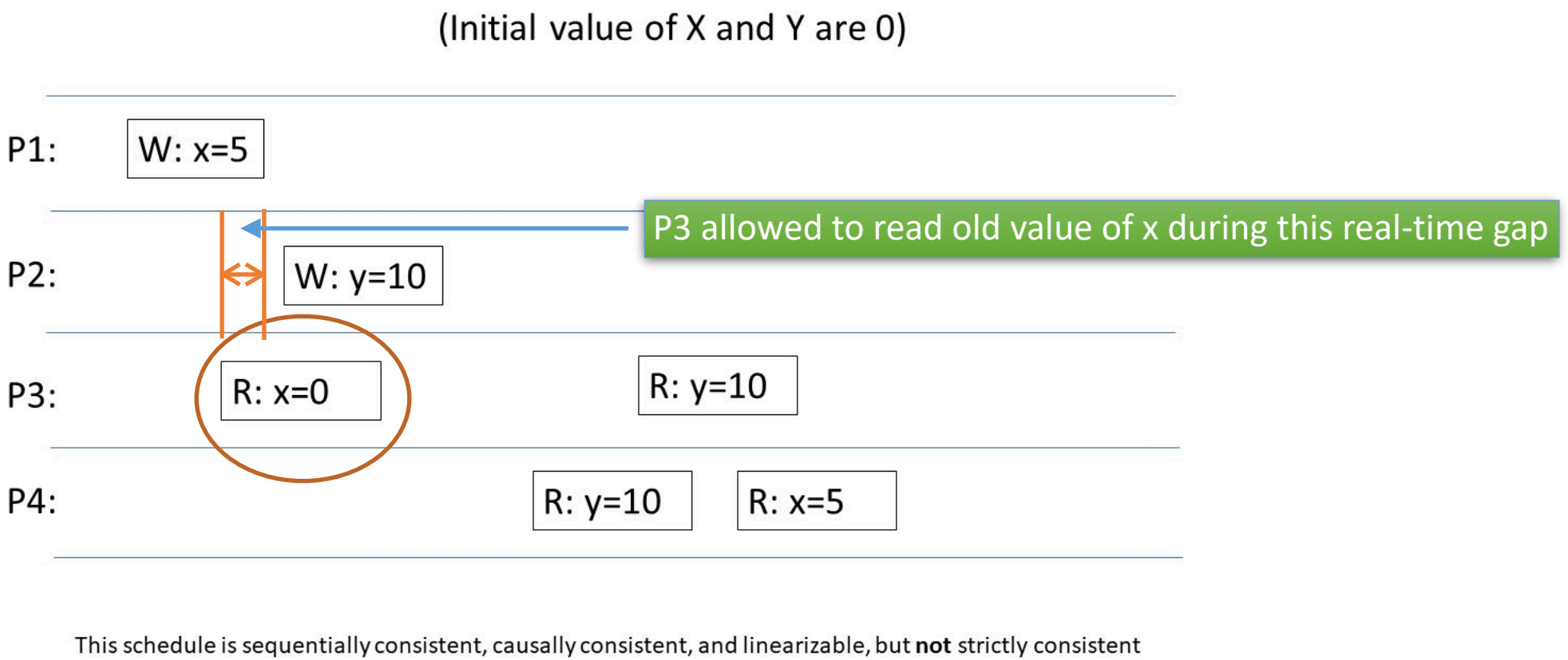
ref: http://dbmsmusings.blogspot.com/2019/07/overview-of-consistency-levels-in.html

Example: Strictly consistent



This schedule is sequentially consistent, causally consistent, linearizable and strictly consistent

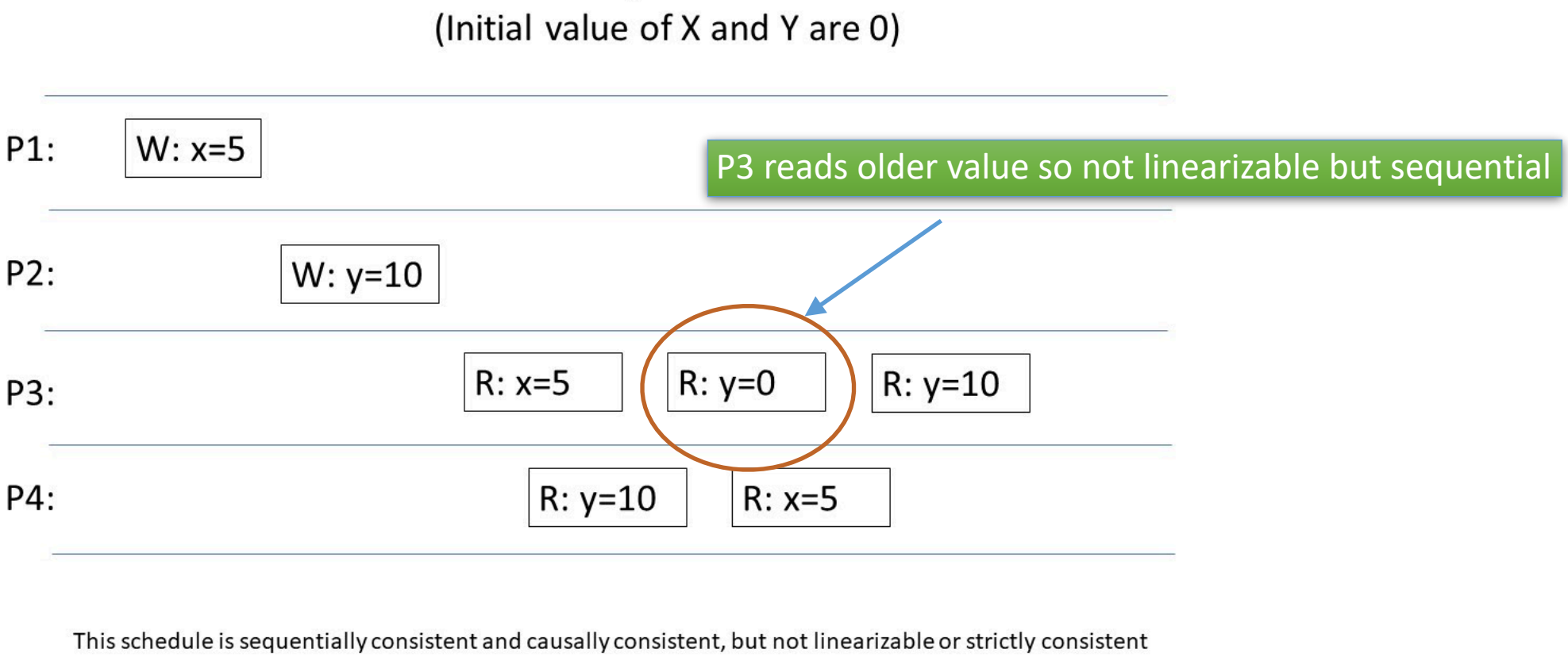
Example: Linearizable



This schedule is sequentially consistent, causally consistent, and linearizable, but **not** strictly consistent

Linearizability takes into account the overlapping time when P3 could not read the latest write of x.

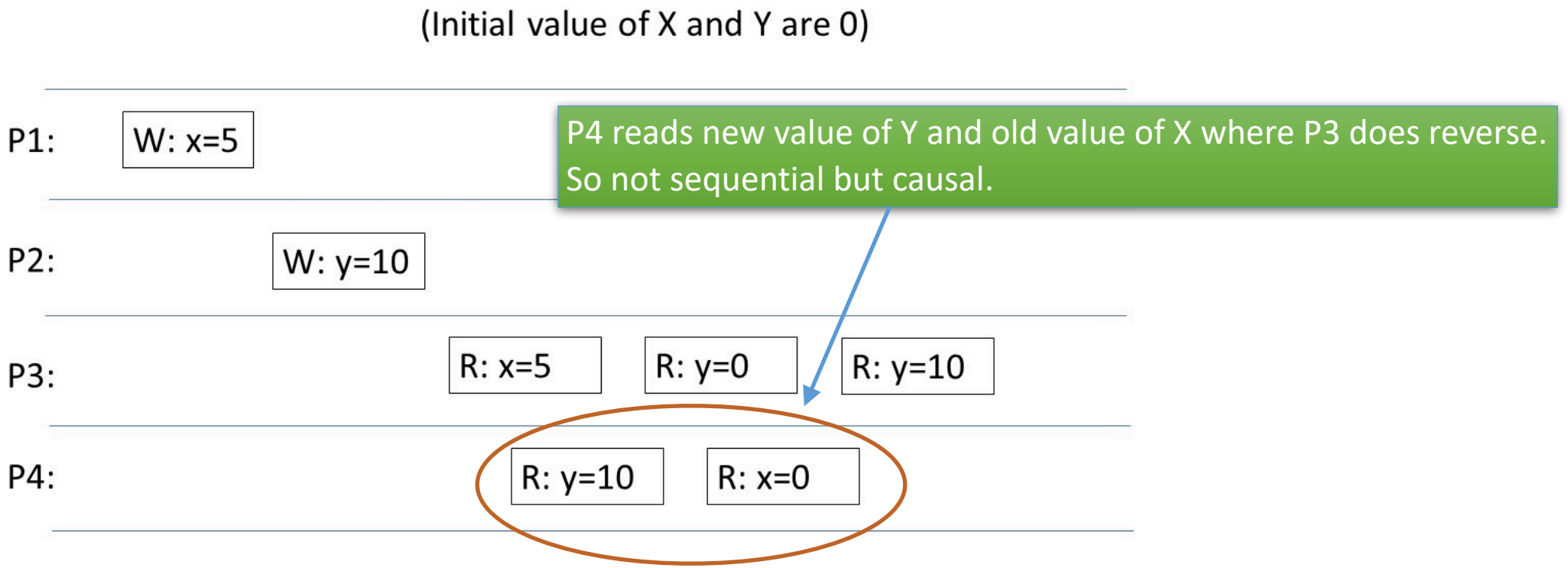
Example: Sequentially consistent



This schedule is sequentially consistent and causally consistent, but not linearizable or strictly consistent

All writes across threads are globally ordered but not in real-time. Here P3 reads older value of Y in real-time but P1 and P2 writes are deemed parallel in sequential order.

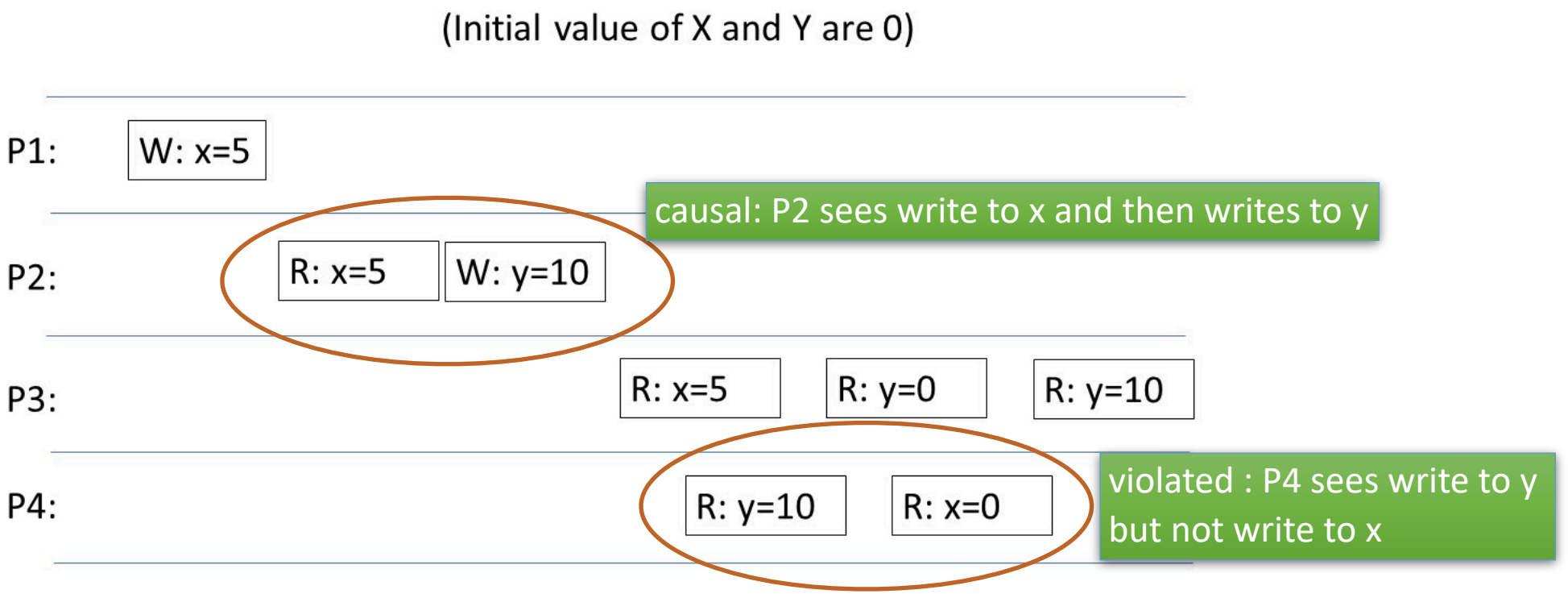
Example: Causally consistent



This schedule is causally consistent, but not linearizable or strictly consistent or even sequentially consistent

Cannot be sequentially consistent because P3 and P4 read X and Y values in different order.
But no causal order violated because x and y are not causally related.

Example: Eventually consistent



This schedule is **not** causally consistent, nor linearizable or strictly consistent or sequentially consistent

Summary

- Basic concepts and definitions of analytics and Big Data analytics
- Overview of some systems / technologies that support Big Data Analytics
  - Cloud, Hadoop/MapReduce, Spark/In-memory computing …
- Concepts of fault tolerance - availability and reliability
  - Calculating MTTR and availability of systems
  - HA cluster configurations
- Concepts of consistency models

Next Session:
CAP Theorem and Big Data Lifecycle