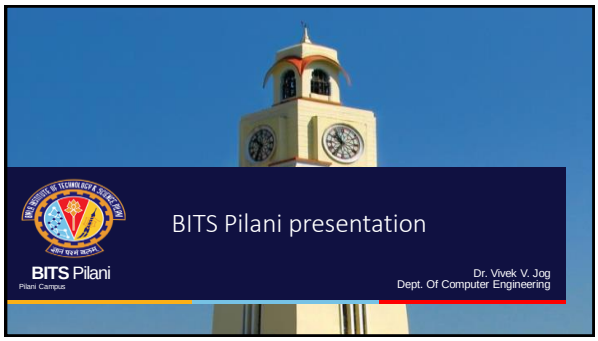
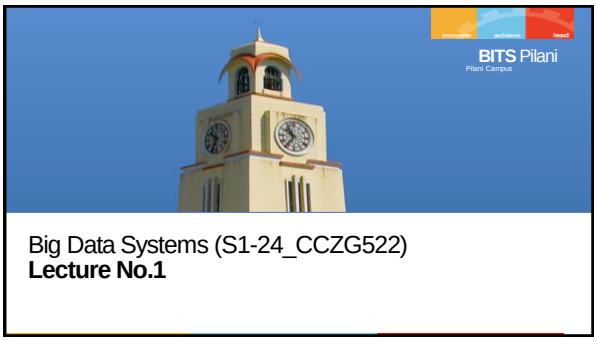
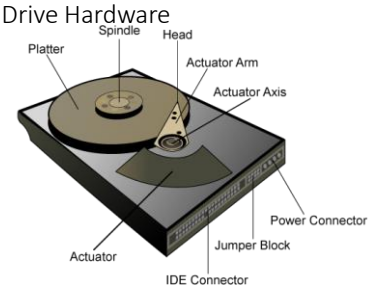




Storage Devices

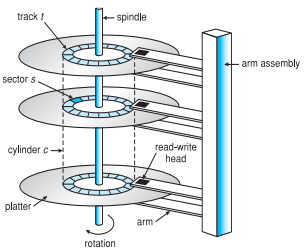
- Hard Drives
- RAID
- SSD

Hard Drive Hardware



4

A Multi-Platter Disk



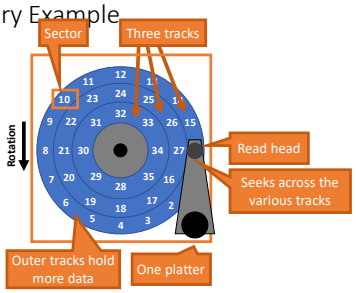
5

Addressing and Geometry

- Externally, hard drives expose a large number of sectors (blocks)
  - Typically 512 or 4096 bytes
  - Individual sector writes are atomic
  - Multiple sectors writes may be interrupted (torn write)
- Drive geometry
  - Sectors arranged into tracks
  - A cylinder is a particular track on multiple platters
  - Tracks arranged in concentric circles on platters
  - A disk may have multiple, double-sided platters
- Drive motor spins the platters at a constant rate
  - Measured in revolutions per minute (RPM)

6

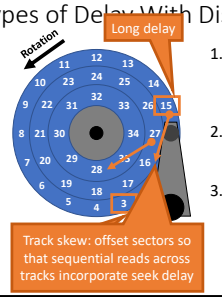
Geometry Example



Common Disk Interfaces

- ST-506 → ATA → IDE → SATA
  - Ancient standard
  - Commands (read/write) and addresses in cylinder/head/sector format placed in device registers
  - Recent versions support Logical Block Addresses (LBA)
- SCSI (Small Computer Systems Interface)
  - Packet based, like TCP/IP
  - Device translates LBA to internal format (e.g. c/h/s)
  - Transport independent
    - USB drives, CD/DVD/Bluray, Firewire
    - iSCSI is SCSI over TCP/IP and Ethernet

Types of Delay With Disks



Three types of delay

1. Rotational Delay
   - Time to rotate the desired sector to the read head
   - Related to RPM
2. Seek delay
   - Time to move the read head to a different track
3. Transfer time
   - Time to read or write bytes

How To Calculate Transfer Time

Seagate

	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15000	7200
Avg. Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s

Transfer time

$T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$

Assume we are transferring 4096 bytes

Cheetah

$T_{I/O}$ = 4 ms + 1 / (15000 RPM / 60 s/M / 1000 ms/s) / 2 + (4096 B / 125 MB/s * 1000 ms/s / 2^{20} MB/B)

$T_{I/O}$ = 4 ms + 2ms + 0.03125 ms ≈ 6 ms

Barracuda

$T_{I/O}$ = 9 ms + 1 / (7200 RPM / 60 s/M / 1000 ms/s) / 2 + (4096 B / 105 MB/s * 1000 ms/s / 2^{20} MB/B)

$T_{I/O}$ = 9 ms + 4.17 ms + 0.0372 ms ≈ 13.2 ms

Sequential vs. Random Access

Rate of I/O

$R_{I/O}$ = transfer_size / $T_{I/O}$

Access Type	Transfer Size		Cheetah 15K.5	Barracuda
Random	4096 B	$T_{I/O}$	6 ms	13.2 ms
		$R_{I/O}$	0.66 MB/s	0.31 MB/s

Random I/O results in very poor disk performance!

Caching

- Many disks incorporate caches (track buffer)
  - Small amount of RAM (8, 16, or 32 MB)
- Read caching
  - Reduces read delays due to seeking and rotation
- Write caching
  - Write back cache: drive reports that writes are complete after they have been cached
    - Possibly dangerous feature. Why?
  - Write through cache: drive reports that writes are complete after they have been written to disk
- Today, some disks include flash memory for persistent caching (hybrid drives)

Disk Scheduling

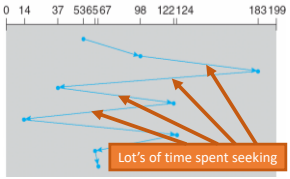
- Caching helps improve disk performance
- But it can't make up for poor random access times
- Key idea: if there are a queue of requests to the disk, they can be reordered to improve performance
  - First come, first serve (FCFC)
  - Shortest seek time first (SSTF)
  - SCAN, otherwise know as the elevator algorithm
  - C-SCAN, C-LOOK, etc.

13

FCFS Scheduling

- Most basic scheduler, serve requests in order

- Head starts at block 53
- Queue: 98, 183, 37, 122, 14, 124, 65, 67



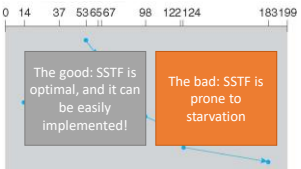
- Total movement: 640 cylinders

14

SSTF Scheduling

- Idea: minimize seek time by always selecting the block with the shortest seek time

- Head starts at block 53
- Queue: 98, 183, 37, 122, 14, 124, 65, 67



- Total movement: 236 cylinders

15

SCAN Example

- Head sweeps across the disk servicing requests in order

- Head starts at block 53
- Queue: 98, 183, 37, 122, 14, 124, 65, 67

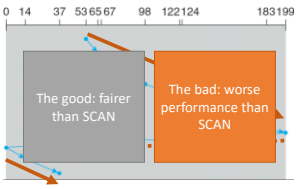
- Total movement: 236 cylinders

16

C-SCAN Example

- Like SCAN, but only service requests in one direction

- Head starts at block 53
- Queue: 98, 183, 37, 122, 14, 124, 65, 67



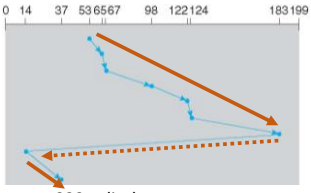
- Total movement: 382 cylinders

17

C-LOOK Example

- Peek at the upcoming addresses in the queue
- Head only goes as far as the last request

- Head starts at block 53
- Queue: 98, 183, 37, 122, 14, 124, 65, 67



- Total movement: 322 cylinders

18

Implementing Disk Scheduling

- We have talked about several scheduling problems that take place in the kernel
  - Process scheduling
  - Page swapping
- Where should disk scheduling be implemented?
  - OS scheduling
    - OS can implement SSTF or LOOK by ordering the queue by LBA
    - However, the OS cannot account for rotation delay
  - On-disk scheduling
    - Disk knows the exact position of the head and platters
    - Can implement more advanced schedulers (SPTF)
    - But, requires specialized hardware and drivers

19

Command Queuing

- Feature where a disk stores of queue of pending read/write requests
  - Called Native Command Queuing (NCQ) in SATA
- Disk may reorder items in the queue to improve performance
  - E.g. batch operations to close sectors/tracks
- Supported by SCSI and modern SATA drives
- Tagged command queuing: allows the host to place constraints on command re-ordering

20

- Hard Drives
- RAID
- SSD

21

Beyond Single Disks

- Hard drives are great devices
  - Relatively fast, persistent storage
- Shortcomings:
  - How to cope with disk failure?
    - Mechanical parts break over time
    - Sectors may become silently corrupted
  - Capacity is limited
    - Managing files across multiple physical devices is cumbersome
    - Can we make 10x 1 TB drives look like a 10 TB drive?

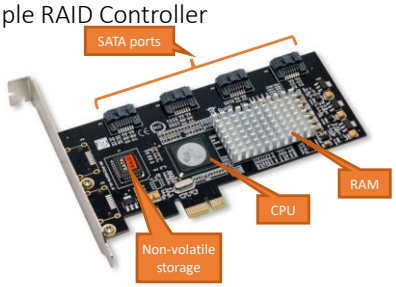
22

Redundant Array of Inexpensive Disks

- RAID: use multiple disks to create the illusion of a large, faster, more reliable disk
- Externally, RAID looks like a single disk
  - i.e. RAID is transparent
  - Data blocks are read/written as usual
  - No need for software to explicitly manage multiple disks or perform error checking/recovery
- Internally, RAID is a complex computer system
  - Disks managed by a dedicated CPU + software
  - RAM and non-volatile memory
  - Many different configuration options (RAID levels)

23

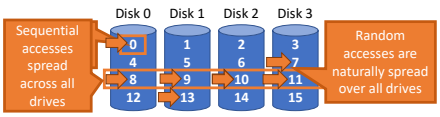
Example RAID Controller



24

RAID 0: Striping

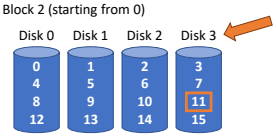
- Key idea: present an array of disks as a single large disk
- Maximize parallelism by striping data cross all *N* disks



25

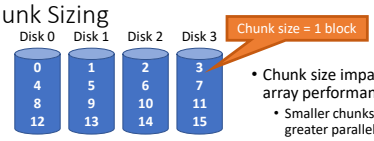
Addressing Blocks

- How do you access specific data blocks?
  - Disk = logical_block_number % number_of_disks
  - Offset = logical_block_number / number_of_disks
- Example: read block 11
  - 11 % 4 = Disk 3
  - 11 / 4 = Physical Block 2 (starting from 0)

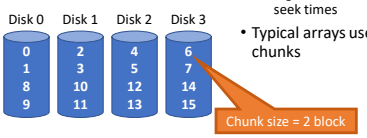


26

Chunk Sizing



- Chunk size impacts array performance
  - Smaller chunks → greater parallelism
  - Big chunks → reduced seek times
- Typical arrays use 64KB chunks



27

Measuring RAID Performance (1)

- As usual, we focus on sequential and random workloads
- Assume disks in the array have **sequential** access time *S*
  - 10 MB transfer
  - *S = transfer_size / time_to_access*
  - 10 MB / (7 ms + 3 ms + 10 MB / 50 MB/s) = 47.62 MB/s

Average seek time	7 ms
Average rotational delay	3 ms
Transfer rate	50 MB/s

28

Measuring RAID Performance (2)

- As usual, we focus on sequential and random workloads
- Assume disks in the array have **random** access time *R*
  - 10 KB transfer
  - *R = transfer_size / time_to_access*
  - 10 KB / (7 ms + 3 ms + 10 KB / 50 MB/s) = 0.98 MB/s

Average seek time	7 ms
Average rotational delay	3 ms
Transfer rate	50 MB/s

29

Analysis of RAID 0

- Capacity: *N*
  - All space on all drives can be filled with data
- Reliability: 0
  - If any drive fails, data is permanently lost
- Sequential read and write: *N * S*
  - Full parallelization across drives
- Random read and write: *N * R*
  - Full parallelization across all drives

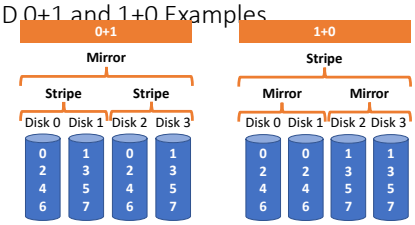
30

RAID 1: Mirroring

- RAID 0 offers high performance, but zero error recovery
- Key idea: make two copies of all data

31

RAID 0+1 and 1+0 Examples

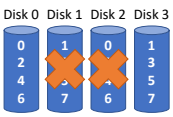


- Combines striping and mirroring
- Superseded by RAID 4, 5, and 6

32

Analysis of RAID 1 (1)

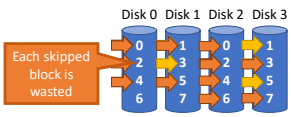
- Capacity: *N / 2*
  - Two copies of all data, thus half capacity
- Reliability: 1 drive can fail, sometime more
  - If you are lucky, *N / 2* drives can fail without data loss



33

Analysis of RAID 1 (2)

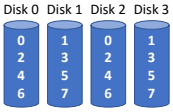
- Sequential write: *(N / 2) * S*
  - Two copies of all data, thus half throughput
- Sequential read: *(N / 2) * S*
  - Half of the read blocks are wasted, thus halving throughput



34

Analysis of RAID 1 (3)

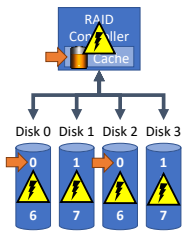
- Random read: *N * R*
  - Best case scenario for RAID 1
  - Reads can parallelize across all disks
- Random write: *(N / 2) * R*
  - Two copies of all data, thus half throughput



35

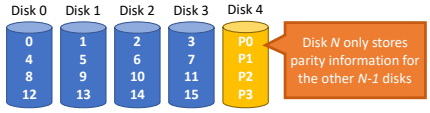
The Consistent Update Problem

- Mirrored writes should be atomic
  - All copies are written, or none are written
- However, this is difficult to guarantee
  - Example: power failure
- Many RAID controllers include a write-ahead log
  - Battery backed, non-volatile storage of pending writes



36

RAID 4: Parity Drive



Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	0	1	1	0 ^ 0 ^ 1 ^ 1 = 0
0	1	0	0	0 ^ 1 ^ 0 ^ 0 = 1
1	1	1	1	1 ^ 1 ^ 1 ^ 1 = 0
0	1	1	1	0 ^ 1 ^ 1 ^ 1 = 1

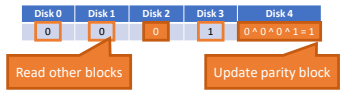
Parity calculated using XOR

37

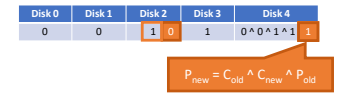
Updating Parity on Write

- How is parity updated when blocks are written?

1. Additive parity

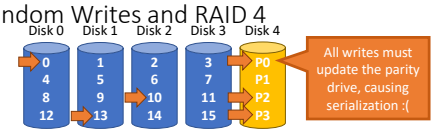


2. Subtractive parity



38

Random Writes and RAID 4



- Random writes in RAID 4
  1. Read the target block and the parity block
  2. Use subtraction to calculate the new parity block
  3. Write the target block and the parity block
- RAID 4 has terrible write performance
  - Bottlenecked by the parity drive

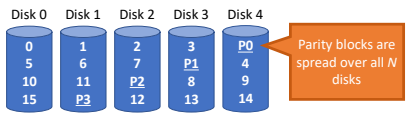
39

Analysis of RAID 4

- Capacity: *N – 1*
  - Space on the parity drive is lost
- Reliability: 1 drive can fail
- Sequential Read and write: *(N – 1) * S*
  - Parallelization across all non-parity blocks
- Random Read: *(N – 1) * R*
  - Reads parallelize over all but the parity drive
- Random Write: *R / 2*
  - Writes serialize due to the parity drive
  - Each write requires 1 read and 1 write of the parity drive, thus *R / 2*

40

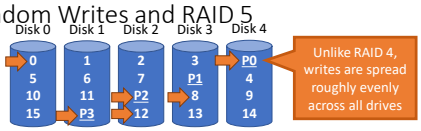
RAID 5: Rotating Parity



Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	0	1	1	0 ^ 0 ^ 1 ^ 1 = 0
1	0	0	0 ^ 1 ^ 0 ^ 0 = 1	0
1	1	1 ^ 1 ^ 1 ^ 1 = 0	1	1
1	0 ^ 1 ^ 1 ^ 1 = 1	0	1	1

41

Random Writes and RAID 5



- Random writes in RAID 5
  1. Read the target block and the parity block
  2. Use subtraction to calculate the new parity block
  3. Write the target block and the parity block
- Thus, 4 total operations (2 reads, 2 writes)
  - Distributed across all drives

42

Analysis of Raid 5

- Capacity: *N – 1* [same as RAID 4]
- Reliability: 1 drive can fail [same as RAID 4]
- Sequential Read and write: *(N – 1) * S* [same]
  - Parallelization across all non-parity blocks
- Random Read: *N * R* [vs. *(N – 1) * R*]
  - Unlike RAID 4, reads parallelize over all drives
- Random Write: *N / 4 * R* [vs. *R / 2* for RAID 4]
  - Unlike RAID 4, writes parallelize over all drives
  - Each write requires 2 reads and 2 write, hence *N / 4*

43

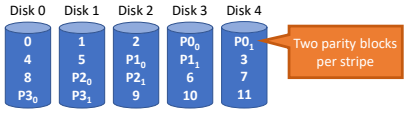
Comparison of RAID Levels

- *N* – number of drives
- *S* – sequential access speed
- *R* – random access speed
- *D* – latency to access a single disk

		RAID 0	RAID 1	RAID 4	RAID 5
	Capacity	***N***	*N / 2*	*N – 1*	*N – 1*
	Reliability	*0*	***1* (maybe *N / 2*)**	*1*	*1*
Throughput	Sequential Read	***N * S***	*(N / 2) * S*	*(N – 1) * S*	*(N – 1) * S*
Throughput	Sequential Write	***N * S***	*(N / 2) * S*	*(N – 1) * S*	*(N – 1) * S*
Throughput	Random Read	***N * R***	***N * R***	*(N – 1) * R*	***N * R***
Throughput	Random Write	***N * R***	*(N / 2) * R*	*R / 2*	*(N / 4) * R*
Latency	Read	***D***	***D***	***D***	***D***
Latency	Write	***D***	***D***	*2 * D*	*2 * D*

44

RAID 6



- Any two drives can fail
- *N – 2* usable capacity
- No overhead on read, significant overhead on write
- Typically implemented using Reed-Solomon codes

45

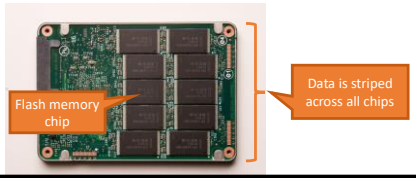
Beyond Spinning Disks

- Hard drives have been around since 1956
  - The cheapest way to store large amounts of data
  - Sizes are still increasing rapidly
- However, hard drives are typically the slowest component in most computers
  - CPU and RAM operate at GHz
  - PCI-X and Ethernet are GB/s
- Hard drives are not suitable for mobile devices
  - Fragile mechanical components can break
  - The disk motor is extremely power hungry

46

Solid State Drives

- NAND flash memory-based drives
  - High voltage is able to change the configuration of a floating-gate transistor
  - State of the transistor interpreted as binary data

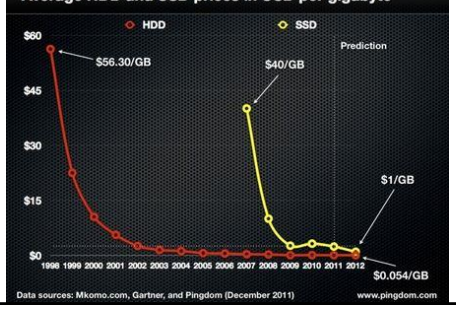


47

Advantages of SSDs

- More resilient against physical damage
  - No sensitive read head or moving parts
  - Immune to changes in temperature
- Greatly reduced power consumption
  - No mechanical, moving parts
- Much faster than hard drives
  - >500 MB/s vs ~200 MB/s for hard drives
  - No penalty for random access
    - Each flash cell can be addressed directly
    - No need to rotate or seek
  - Extremely high throughput
    - Although each flash chip is slow, they are RAIDed

48



Structured, Semi-structured and Unstructured Data

Structures Data

Structured data is data that has a standardized format for efficient access by software and humans alike. It is typically tabular with rows and columns that clearly define data attributes. Computers can effectively process structured data for insights due to its quantitative nature.

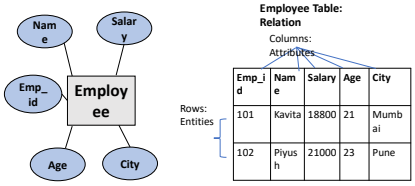
Examples

Here are examples of structured data systems:
1)Excel files
2)SQL databases
3)Point-of-sale data
4)Web form results
5)Search engine optimization (SEO) tags

Relational Database

- A Relational Database is a collection of data items which are organized in the form of tables of information, which can be easily accessed.
- This concept was introduced by E.F.Codd a researcher at IBM in 1970 .
- In Relational Databases the data is stored using rows and columns in the form of a table.

Name	Age	City	Salary
Santosh	29	Mumbai	34500
Rupali	30	Pune	30000
Kalpana	45	Delhi	50000
Monali	28	Surat	35000
Hemant	41	Mumbai	55000



Relation, Entities and Attributes.

1. Relation : Table which contains rows and columns of related data.
2. Entities (Rows) : They are items about which some relevant information is stored in the database.
3. Attributes (Columns) : They are the qualities of an entity that are stored as information.

- A relationship is a connection between the data stored in one relational database table and another.

Stock_id	Supp_id	Stock_name	clstock
1301	001	Soap	234
1102	002	Shampoo	153
1203	003	Toothpaste	105

Stock Table

Supp_id	Supp_name	Phone_no
001	Hiten & Co.	9914326571
002	Manu & Co.	8114456572
003	Raj & Co.	9942156992

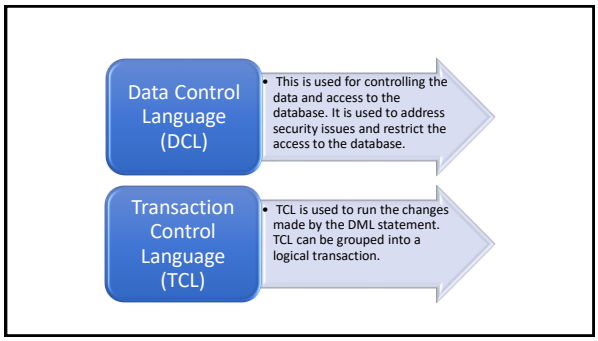
Supplier Table

- A join is a connection between two tables where the data from them is merged together based on a field (column) that is common to these tables, creating a new virtual table.
- To create a join it is necessary that the tables have a relationship.

Stock_id	Supp_id	Stock_name	Supp_name	Phone_no.
1301	001	Soap	Hiten & Co.	9914326571
1102	002	Shampoo	Manu & Co.	8114456572
1203	003	Toothpaste	Raj & Co.	9942156992

Database Languages

- **Data Definition Language (DDL)**
  - This language is used by the designer and programmers of the database to indicate the content and structure of the database.
- **Data Manipulation Language (DML)**
  - This language is used primarily for data manipulation and processing. It involves retrieving the data, arranging the data deleting the data and displaying the data etc.



- RDBMS applications store data in a tabular form.
- **Attributes** describe the characteristics or properties of an **entity** in a database table (**Relation**).
- A **JOIN** is used to combine rows from two or more tables, based on a related column between them.
- DDL is used for specifying the database schema.
- DML is used for accessing and manipulating data in a database.
- DCL is used for granting and revoking user access on a database.
- The changes in the database that we made using DML commands are either performed or rollbacked using TCL.

Challenges with Structured data

Limited usage

The predefined structure is a benefit but can also be a challenge. Structured data can only be utilized for its intended purpose. For example, booking data can give you information about booking system finances and booking popularity. But it can't reveal which marketing campaigns were more effective in bringing in more bookings without further modification. You'll have to add marketing campaign relational data to your bookings if you want the additional insights.

Inflexibility

It can be costly and resource-intensive to change the schema of structured data as circumstances change and new relationships or requirements emerge.

Semi-Structured Data

What is semi-structured data?

Semi-structured data is a type of digital information that does not adhere to the rigid structure of traditional databases or spreadsheets, yet it contains some level of organization and can be processed by computers.

It falls between fully structured data, which fits neatly into tables with predefined schemas, and unstructured data, which lacks a specific format or organization.

It exhibits the following key characteristics:

1)Flexible schema
2)Human-readable
3)Metadata
4)Mix of data types
5)Hierarchy
6)Partial consistency
7)Scalability

1. Flexible schema

Semi-structured data does not adhere to a strict, predefined schema, allowing for variations in the structure and content of each data instance.

2. Human-readable

It is often human-readable, with elements like labels and tags, making it more accessible for both machines and humans.

3. Metadata

Semi-structured data typically contains metadata, such as tags, attributes, or keys, which provide context and organization to the data elements.

4. Mix of data types

This type of data can encompass a variety of data formats, including JSON, XML, HTML, and YAML, and may include text, images, or multimedia content.

5. Hierarchy

It often exhibits hierarchical relationships, enabling the representation of nested and related data elements.

6. Partial consistency

Semi-structured data allows for partial consistency, meaning that not all data instances need to have the same attributes or structure.

7. Scalability

It is well-suited for data generated from diverse sources like IoT devices, mobile apps, and web pages, making it scalable and adaptable to evolving data needs.

Here are some common examples of semi-structured data:

@ JSON (JavaScript Object Notation)
@ XML (eXtensible Markup Language)
@ CSV (Comma-Separated Values)
@ HTML (Hypertext Markup Language)
@ Log files
@ NoSQL databases

1. JSON (JavaScript Object Notation)

JSON is a widely used format for representing data in a hierarchical structure composed of key-value pairs. It is easy to read and write for both humans and machines. JSON is commonly used in web APIs, configuration files, and data interchange between applications.

2. XML (eXtensible Markup Language)

XML is a versatile format for encoding structured data using tags to define elements and attributes. It allows for creating custom document structures and is commonly found in web services, RSS feeds, and configuration files.

3. CSV (Comma-Separated Values)

CSV files store tabular data with values separated by commas or other delimiters. While they lack a formal schema, they are commonly used for data exchange between spreadsheets and databases, as well as in log files.

5. HTML (Hypertext Markup Language)

HTML is primarily used for structuring web pages, but it contains valuable data elements such as meta-tags, attributes, and text content. Web scraping techniques are often employed to extract data from HTML documents.

6. Log files

Log files generated by various systems contain semi-structured data, including timestamps, events, and metadata. They are essential for system monitoring, troubleshooting, and security analysis.

7. NoSQL databases

NoSQL databases, like MongoDB and Cassandra, store data in semi-structured formats, allowing flexibility in data modeling and schema design. These databases are popular for handling unstructured and rapidly changing data.

Example : JSON

```
{"widget": {
    "debug": "on",
    "window": {
        "title": "Sample Konfabulator Widget",
        "name": "main_window",
        "width": 500,
        "height": 500
    },
    "image": {
        "src": "Images/Sun.png",
        "name": "sun1",
        "hOffset": 250,
        "vOffset": 250,
        "alignment": "center"
    },
    "text": {
        "data": "Click Here",
        "size": 36,
        "style": "bold",
        "name": "text1",
        "hOffset": 250,
        "vOffset": 100,
        "alignment": "center",
        "onMouseUp": "sun1.opacity = (sun1.opacity / 100) * 90;"
    }
}}
```

Example: XML

The same text expressed as XML:

```
<widget>
    <debug>on</debug>
    <window title="Sample Konfabulator Widget">
        <name>main_window</name>
        <width>500</width>
        <height>300</height>
    </window>
    <image src="Images/Sun.png" name="sun1">
        <hOffset>250</hOffset>
        <vOffset>250</vOffset>
        <alignment>center</alignment>
    </image>
    <text data="Click Here" size="36" style="bold">
        <name>text1</name>
        <hOffset>250</hOffset>
        <vOffset>100</vOffset>
        <alignment>center</alignment>
        <onMouseUp>
            sun1.opacity = (sun1.opacity / 100) * 90;
        </onMouseUp>
    </text>
</widget>
```


Comparison

FEATURES	STRUCTURED	SEMI STRUCTURED	UNSTRUCTURED
Format Type	Relational Database	HTML, XML, JSON	Binary, Character
Version Management	Rows, columns, tuples	Not as common – graph is possible	Whole data
Implementation	SQL	Anonymous nodes	-
Robustness	Robust	Limited robustness	-
Storage Requirement	Less	Significant	Large
Applications	DBMS, RDF, ERP system, Data Warehouse, Apache Parquet, Financial Data, Relational Table	Server Logs, Sensor Output	No SQL, Video, Audio, Social Media, Online Forums, MRI, Ultrasound

Disadvantages of Semi Structured data

Difficult Analysis

While semi-structured data is more usable than unstructured data, it's less useful than structured data and it can be difficult to tag and index.

Limited Tooling

Artificial intelligence and machine learning (AI/ML) tools have not yet had a significant impact on the semi-structured data space—as such, existing tool stacks and models offer limited capability to transform datasets into actionable insights.

Data Security

It can be easy to overlook sensitive information contained in less visible or hidden parts of semi-structured data, and complacency or mistakes can make organizations vulnerable to security risks.

Unstructured Data

In its simplest form, refers to any data that does not have predefined structure or organization. Unlike structured data, which is organized into neat rows and columns within a database, unstructured data is an unsorted and vast information collection. It can come in different forms, such as text documents, emails, images, videos, social media posts, sensor data, etc.

Examples of Unstructured Data

Examples of unstructured data are:

1)Business Documents.
2)Emails.
3)Social Media.
4)Customer Feedback.
5)Webpages.
6)Open Ended Survey Responses.
7)Images, Audio, and Video.

Unstructured Data types

Unstructured data can be broadly classified into two categories:

human-generated unstructured data which includes various forms of content people create, such as text documents, emails, social media posts, images, and videos; and

machine-generated unstructured data, on the other hand, which is generated by devices and sensors, including log files, GPS data, Internet of Things (IoT) output, and other telemetry information.

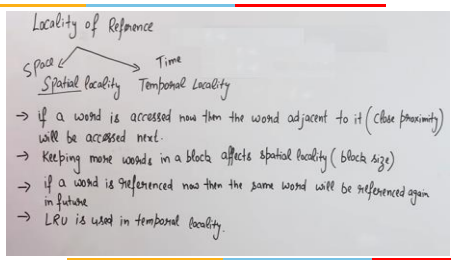
Tools Used To Manage Unstructured Data

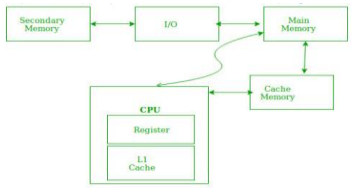
Technology category	Technology examples	Important features
APIs	Twitter API	Allows developers to access and collect public tweets, user profiles, and other data from the Twitter platform.
	Instagram API	Enables the extraction of data, such as user profiles, images, and comments
Data ingestion tools	Apache Nifi	Automates the movement and transformation of data between systems; provides a web-based interface to design, control, and monitor data flows
	Logstash	Server-side data processing pipeline; ingests data from multiple sources; sends data to various output destinations or file storage in real-time
Data lakes and NoSQL databases	Amazon S3	Scalable; low-latency access; easy integration with AWS services; virtually unlimited storage
	Google Cloud Storage	Multiple storage classes; automatic scaling; global edge-caching; easy-to-use API for data access
	MongoDB	JSON-like format; horizontal scalability; rich query language

Locality of reference

It refers to a phenomenon in which a computer program tends to access same set of memory locations for a particular time period. In other words, **Locality of Reference** refers to the tendency of the computer program to access instructions whose addresses are near one another. The property of locality of reference is mainly shown by loops and subroutine calls in a program.

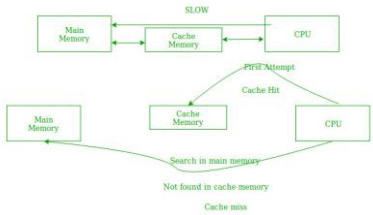
Conept





- In case of loops in program control processing unit repeatedly refers to the set of instructions that constitute the loop.
- In case of subroutine calls, everytime the set of instructions are fetched from memory.
- References to data items also get localized that means same data item is referenced again and again.

Flow



Cahe Hit & Miss

First, it will access the cache memory as it is near to it and provides very fast access. If the required data or instruction is found, it will be fetched. This situation is known as a cache hit.

if the required data or instruction is not found in the cache memory then this situation is known as a cache miss. Now the main memory will be searched for the required data or instruction that was being searched and if found will go through one of the two ways:

First way is that the CPU should fetch the required data or instruction and use it and that's it but what, when the same data or instruction is required again.CPU again has to access the same main memory location for it and we already know that main memory is the slowest to access.

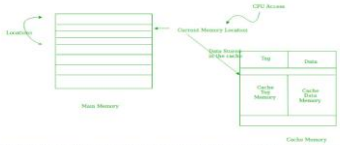
The second way is to store the data or instruction in the cache memory so that if it is needed soon again in the near future it could be fetched in a much faster way.

Cache Operation

Cache Operation: It is based on the principle of locality of reference. There are two ways with which data or instruction is fetched from main memory and get stored in cache memory. These two ways are the following:

Temporal Locality

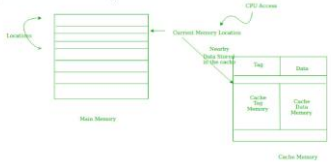
Temporal Locality – Temporal locality means current data or instruction that is being fetched may be needed soon. So we should store that data or instruction in the cache memory so that we can avoid again searching in main memory for the same data.



When CPU accesses the current main memory location for reading required data or instruction, it also gets stored in the cache memory which is based on the fact that same data or instruction may be needed in near future. This is known as temporal locality. If some data is referenced, then there is a high probability that it will be

Spatial Locality

Spatial Locality – Spatial locality means instruction or data near to the current memory location that is being fetched, may be needed soon in the near future. This is slightly different from the temporal locality. Here we are talking about nearly located memory locations while in temporal locality we were talking about the actual memory location that was being fetched.

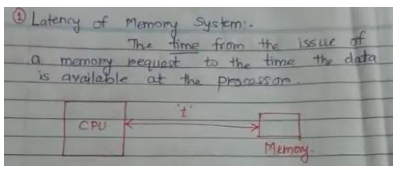


Few Calculations

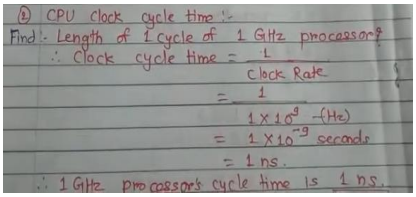
```
Hit + Miss  = Total CPU Reference
Hit Ratio(h) = Hit / (Hit+Miss)
Miss Ratio = 1 - Hit Ratio(h)
Miss Ratio = Miss / (Hit+Miss)
```

```
Tavg = Average time to access memory
For simultaneous access
Tavg = h * Tc + (1-h)*Tm
For hierarchial access
Tavg = h * Tc + (1-h)*(Tm + Tc)
```

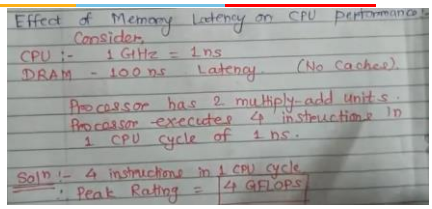

Latency Impact Example Without Cache



Contd..

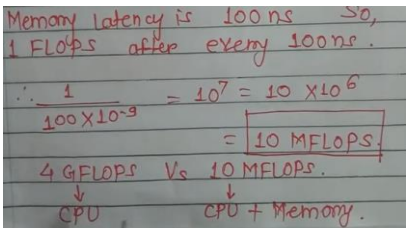


Contd..

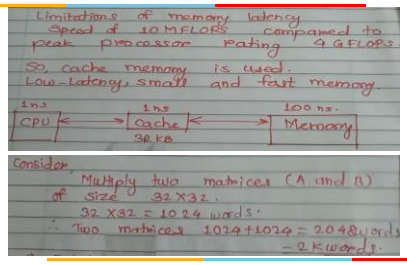


GFLOPS : Giga Floating Operations Per Second

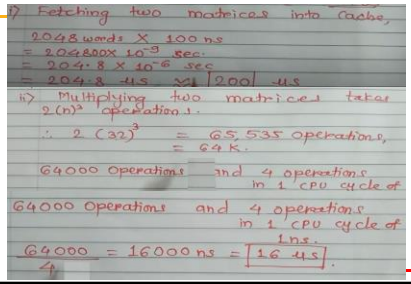
Contd..



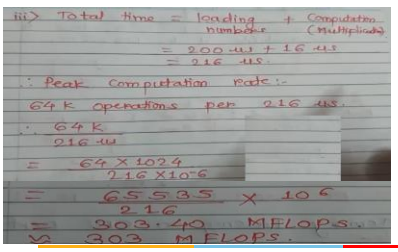
Latency Impact Example With Cache



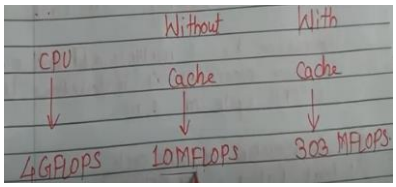
Latency Impact Example

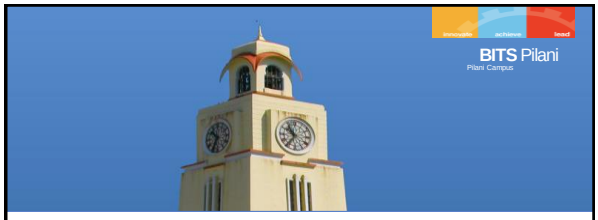


Contd..



Conclusion

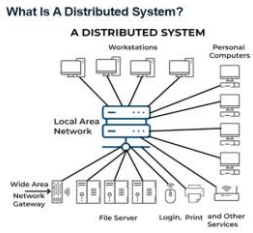




Distributed – Cluster Computing

S.NO	Parallel Computing	Distributed Computing
1.	Many operations are performed simultaneously	System components are located at different locations
2.	Single computer is required	Uses multiple computers
3.	Multiple processors perform multiple operations	Multiple computers perform multiple operations
4.	It may have shared or distributed memory	It have only distributed memory
5.	Processors communicate with each other through bus	Computer communicate with each other through message passing.
6.	Improves the system performance	Improves system scalability, fault tolerance and resource sharing capabilities

Distributed Env



A distributed system is a **collection of computer programs spread across multiple computational nodes**. Each node is a separate physical device or software process but works towards a shared objective. This setup is also known as distributed computing systems or distributed databases.
The main goal of a distributed database system is to **avoid bottlenecks and eliminate central points of failure** by allowing the nodes to communicate and coordinate through a shared network.

Benefits Of Distributed Systems

Scalability: Distributed database systems offer improved scalability as they can add more nodes to easily accommodate the increase in workload**.**

Improved reliability: It eliminates central points of failure and bottlenecks. The redundancy of nodes ensures that even if one node fails, others can take over its tasks.

Enhanced performance: These systems can easily scale horizontally by adding more nodes or vertically by increasing a node's capacity. This scalability results in enhanced performance and optimum output.

Drawbacks & Risks Of Distributed Systems
Requirement for specialized tools: Management of multiple repositories in a distributed system requires the use of specialized tools.
Development sprawl and complexity: As the system's complexity grows, organizing, managing, and improving a distributed system can become challenging.
Security risks: A distributed system is more vulnerable to cyber attacks, as data processing is distributed across multiple nodes that communicate with each other.

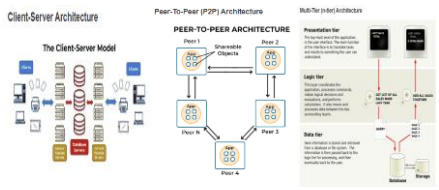
Drawbacks & Risks Of Distributed Systems

Requirement for specialized tools: Management of multiple repositories in a distributed system requires the use of specialized tools.

Development sprawl and complexity: As the system's complexity grows, organizing, managing, and improving a distributed system can become challenging.

Security risks: A distributed system is more vulnerable to cyber attacks, as data processing is distributed across multiple nodes that communicate with each other

Examples



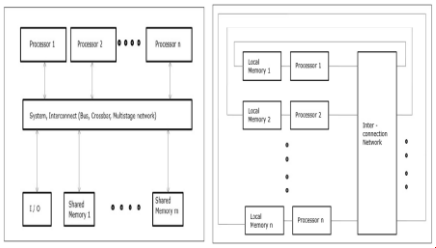
API can also viewed as Distributed Service Flavor

Definition : Parallel Systems

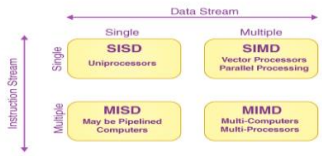
Parallel computing refers to the process of executing several processors an application or computation simultaneously. Generally, it is a kind of computing architecture where the large problems break into independent, smaller, usually similar parts that can be processed in one go.

It is done by multiple CPUs communicating via shared memory, which combines results upon completion. It helps in performing large computations as it divides the large problem between more than one processor.

Parallel Systems UMA/NUMA



Flynn's Classification



Parallel processing can happen in the data stream, the instruction stream, or both.

Parallel V/s Distributed Databases

Parallel Database:
A parallel DBMS is a DBMS that runs across multiple processors and is designed to execute operations in parallel, whenever possible. The parallel DBMS link a number of smaller machines to achieve the same throughput as expected from a single large machine.
Features :
There are parallel working of CPUs
It improves performance
It divides large tasks into various other tasks
Completes works very quickly
Distributed Database:
A Distributed database is defined as a logically related collection of data that is shared which is physically distributed over a computer network on different sites. The Distributed DBMS is defined as, the software that allows for the management of the distributed database and makes the distributed data available for the users.
Features :
It is a group of logically related shared data
The data gets split into various fragments
There may be a replication of fragments
The sites are linked by a communication network

Definition Shares/Message Pass Model

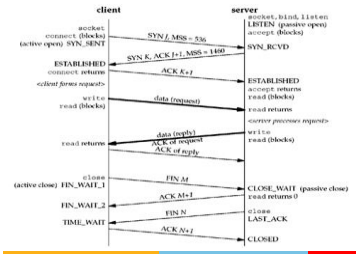
What is Shared Memory?
The fundamental model of inter-process communication is the shared memory system. In a shared memory system, the collaborating communicates with each other by establishing the shared memory region in the address space region.

If the process wishes to initiate communication and has data to share, create a shared memory region in its address space. After that, if another process wishes to communicate and tries to read the shared data, it must attach to the starting process's shared address space.

What is Message Passing?
In this message passing process model, the processes communicate with others by exchanging messages. A communication link between the processes is required for this purpose, and it must provide at least two operations:
transmit (message) and ***receive (message)***.

Message sizes might be flexible or fixed.

Message Passing Scenario



Shared V/s Message Passing

Shared Memory	Message Passing
It is mainly used for data communication.	It is mainly used for communication.
It offers a maximum speed of computation because communication is completed via the shared memory, so the system calls are only required to establish the shared memory.	It takes a huge time because it is performed via the kernel (system calls).
The code for reading and writing the data from the shared memory should be written explicitly by the developer.	No such code is required in this case because the message passing feature offers a method for communication and synchronization of activities executed by the communicating processes.
It is used to communicate between the single processor and multiprocessor systems in which the processes to be communicated are on the same machine and share the same address space.	It is most commonly utilized in a distributed setting when communicating processes are spread over multiple devices linked by a network.
It is a faster communication strategy than the message passing.	It is a relatively slower communication strategy than the shared memory.
Make sure that processes in shared memory aren't writing to the same address simultaneously.	It is useful for sharing little quantities of data without causing disputes.

Networking

Two Networking Models

Server Based Network

- Networked computers take on different roles or functions in relation to each other.
  - **Client-Server** network:
    - Requires central servers responding to client requests

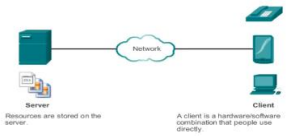
Networking Models

Two Networking Models

Server Based Network Peer to Peer Network

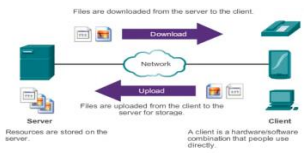
- Networked computers take on different roles or functions in relation to each other.
  - **Client-Server** network:
    - Requires central servers responding to client requests
  - **Peer-to-Peer (P2P)** network.
    - Variations: P2P networks and P2P applications.

Contd



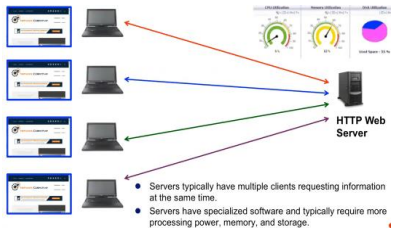
- In a client server model, 2 computers typically communicate with each other by using request/response protocols.
  - The requestor takes on the role of a **client**.
  - The responder takes on the role of a **server**.

Contd



- Files are downloaded from a **server**.
- Files / requests are uploaded from a **client**.

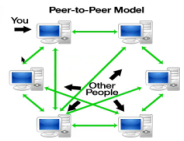
Contd



- Servers typically have multiple clients requesting information at the same time.
- Servers have specialized software and typically require more processing power, memory, and storage.

Peer to Peer

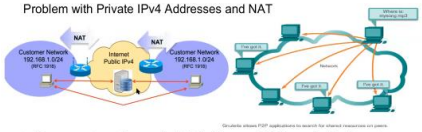
Peer-to-Peer Networking Model Concerns



- P2P networks **decentralize the resources** on a network.
  - Data can be located anywhere and on any connected device.
- In a "pure P2P" architecture, data is accessed from a peer device without the use of a dedicated server.
- Each device (known as a peer) can function as both a server and a client.

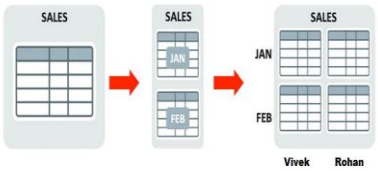
NAT-Nating

Problem with Private IPv4 Addresses and NAT



- Because most user have a private IPv4 address and NAT is required for Internet access, each peer must access **a central index server** to get the location of a resource stored on another peer.
  - The index server can also **help connect two peers**, but after connected, the **communication "may" take place between the two peers without additional communication to the index server – (TCP/UDP hole punching, NAT Transversal.**
  - Real solution: IPv6

Data Partitioning



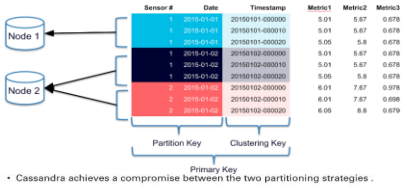
Key Value Based

- Volume 1 contains words starting with **A and B**, but volume 12 contains words starting with **T, U, V, W, X, Y, and Z**.
- Simply having one volume per two letters of the alphabet would lead to some volumes being much bigger than others.
- In order to distribute the data evenly, the partition boundaries need to adapt to the data.

Hash Key Value Based

- A good hash function takes **skewed data** and makes it **uniformly distributed.**
- Say you have a 32-bit hash function that takes a string. Whenever you give it a new string, it returns a seemingly random number between 0 and $2^{32} - 1$.
- Even if the input strings are very similar, their hashes are **evenly distributed** across that range of numbers.

Cassandra Strategy



- Cassandra achieves a compromise between the two partitioning strategies .
- A table in Cassandra can be declared with a *compound primary key* consisting of several columns. Only the first part of that key is hashed to determine the partition, but the other columns are used as a concatenated index for sorting the data in Cassandra's SSTables.

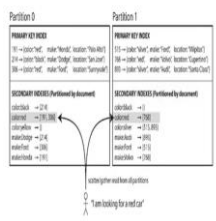
Partition key. The partition key is part of the **primary key and defines which node will store the data based on its hash value and distributes data across the nodes** in the Cassandra cluster. All rows with the same partition key will be stored together on the same node.

Clustering key(s). The clustering key(s) follow the partition key in the definition of the primary key in Cassandra. **The clustering key sorts the data within a partition**, allowing rows with the same partition key to be ordered based on their different clustering key values.

Secondary Index Approach

A secondary index usually doesn't identify a record uniquely

but rather is a way of searching for occurrences of a particular value: find all actions by user 123, find all articles containing the word hogwash, find all cars whose color is red, and so on.



- In this indexing approach, **each partition is completely separate: each partition maintains its own secondary indexes,** covering only the documents in that partition.
- It doesn't care what data is stored in other partitions.
- Whenever you write to the database—to add, remove, or update a document—you only need to deal with the partition that contains the document ID that you are writing.
- For that reason, a document-partitioned index is also known as a local index (as opposed to a global index, described in the next section).

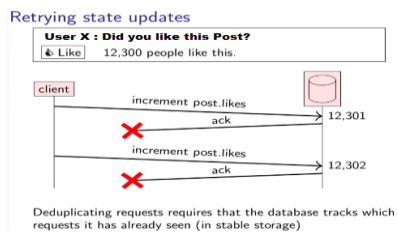
Strategies for Data Access Replication

- Keeping a copy of the same data on multiple nodes
- Databases, filesystems, caches, . . .
- A node that has a copy of the data is called a **replica**
- If some replicas are faulty, others are still accessible
- Spread load across many replicas
- Easy if the data doesn't change: just copy it
- We will focus on data changes

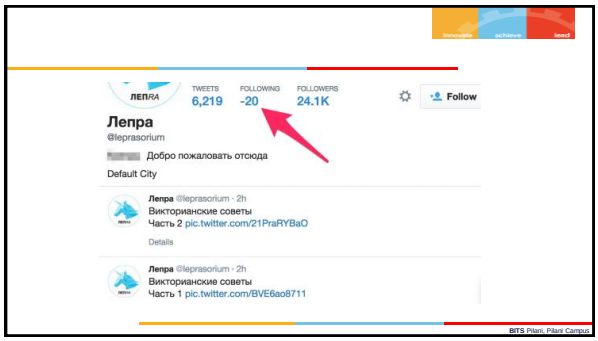
Compare to **RAID** (Redundant Array of Independent Disks): replication within a single computer

- RAID has single controller; in distributed system, each node acts independently
- Replicas can be distributed around the world, near users

Updates



Deduplicating requests requires that the database tracks which requests it has already seen (in stable storage)



Idempotence

Idempotence

A function f is idempotent if $f(x) = f(f(x))$.

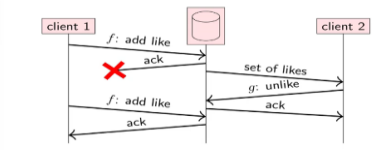
- **Not idempotent:** $f(likeCount) = likeCount + 1$
- **Idempotent:** $f(likeSet) = likeSet \cup \{userID\}$

Idempotent requests can be retried without deduplication.

Choice of retry semantics:

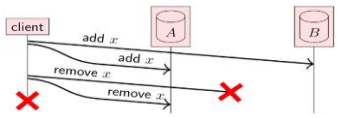
- **At-most-once** semantics:
  send request, don't retry, update may not happen
- **At-least-once** semantics:
  retry request until acknowledged, may repeat update
- **Exactly-once** semantics:
  retry + idempotence or deduplication

Idempotent : Does not work here

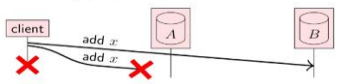


$f(likes) = likes \cup \{userID\}$
$g(likes) = likes \setminus \{userID\}$
Idempotent? $f(f(x)) = f(x)$ but $f(g(f(x))) \neq g(f(x))$

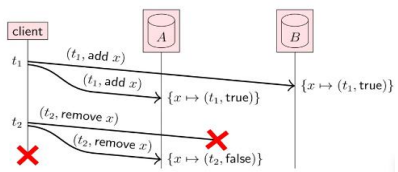
Add and Remove



Final state ($x \notin A,\ x \in B$) is the same as in this case:



Tombstone

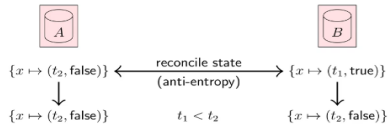


"remove x" doesn't actually remove x: it labels x with "false" to indicate it is invisible (a **tombstone**)

Every record has **logical timestamp** of last write

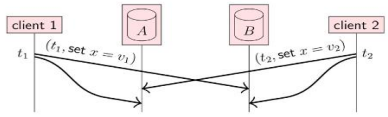
Reconciling Replicas

Replicas periodically communicate among themselves to check for any inconsistencies.



Propagate the record with the latest timestamp, discard the records with earlier timestamps (for a given key).

Conflict Due to Concurrent Write



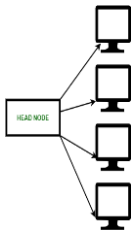
Two common approaches:

- **Last writer wins** (LWW):
  Use timestamps with total order (e.g. Lamport clock)
  Keep v_2 and discard v_1 if $t_2 > t_1$. Note: **data loss**!
- **Multi-value register**:
  Use timestamps with partial order (e.g. vector clock)
  v_2 replaces v_1 if $t_2 > t_1$; preserve both $\{v_1, v_2\}$ if $t_1 \parallel t_2$

Cluster Computing

Introduction :

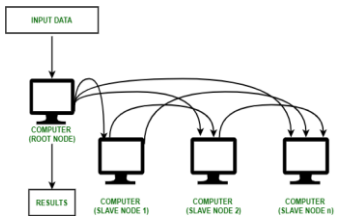
- Cluster computing is a collection of tightly or loosely connected computers that work together so that they act as a single entity.
- The connected computers execute operations all together thus creating the idea of a single system.
- The clusters are generally connected through fast local area networks (LANs)



Why is Cluster Computing important?

1. Cluster computing gives a relatively inexpensive, unconventional to the large server or mainframe computer solutions.
2. It resolves the demand for content criticality and process services in a faster way.
3. Many organizations and IT companies are implementing cluster computing to augment their scalability, availability, processing speed and resource management at economic prices.
4. It ensures that computational power is always available.
5. It provides a single general strategy for the implementation and application of parallel high-performance systems independent of certain hardware vendors and their product decisions.

Cluster computing layout



Types of Cluster computing :

- **High performance (HP) clusters :**
  - HP clusters use computer clusters and supercomputers to solve advanced computational problems.
  - These are used to perform functions that need nodes to communicate as they perform their jobs.
  - These are designed to take benefit of the parallel processing power of several nodes.

Cont....

- **Load-balancing clusters :**
  - Incoming requests are distributed for resources among several nodes running similar programs or having similar content.
  - This prevents any single node from receiving a disproportionate amount of tasks.
  - This type of distribution is generally used in a web-hosting environment.

Cont....

- **High Availability (HA) Clusters :**
  - HA clusters are designed to maintain redundant nodes that can act as backup systems in case any failure occurs.
  - Consistent computing services like business activities, complicated databases, customer services like e-websites and network file distribution are provided.
  - They are designed to give uninterrupted data availability to the customers.

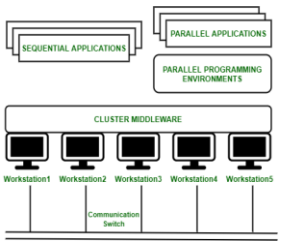
Classification of Cluster :

- **Open Cluster :**
  - IPs are needed by every node and those are accessed only through the internet or the web. This type of cluster causes enhanced security concerns.
- **Close Cluster :**
  - The nodes are hidden behind the gateway node, and they provide increased protection. They need fewer IP addresses and are good for computational tasks.

Cluster Computing Architecture :

- It is designed with an array of interconnected individual computers and computer systems operating collectively as a single standalone system.
- It is a group of workstations or computers working together as a single, integrated computing resource connected via high-speed interconnects.
- A node – Either a single or multiprocessor network having memory, input and output functions and an operating system.
- Two or more nodes are connected on a single line or every node might be connected individually through a LAN connection.

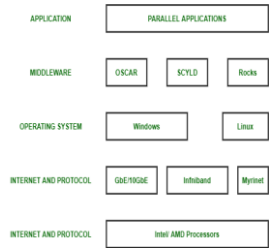
Cluster Computing Architecture Diagram



Components of a Cluster Computer :

1. Cluster Nodes
2. Cluster Operating System
3. The switch or node interconnect
4. Network switching hardware

Component Diagram



Advantages of Cluster Computing :

- **High Performance :**
  - The systems offer better and enhanced performance than that of mainframe computer networks.
- **Easy to manage :**
  - Cluster Computing is manageable and easy to implement.
- **Scalable :**
  - Resources can be added to the clusters accordingly.
- **Expandability :**
  - Computer clusters can be expanded easily by adding additional computers to the network. Cluster computing is capable of combining several additional resources or networks to the existing computer system.
- **Availability :**
  - The other nodes will be active when one node gets failed and will function as a proxy for the failed node. This makes sure for enhanced availability.
- **Flexibility :**
  - It can be upgraded to the superior specification or additional nodes can be added.

Disadvantages of Cluster Computing :

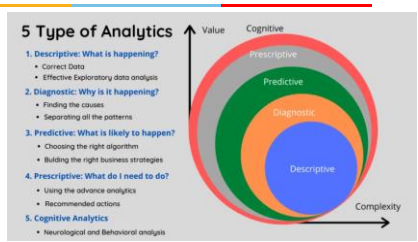
- **High cost :**
  - It is not so much cost-effective due to its high hardware and its design.
- **Problem in finding fault :**
  - It is difficult to find which component has a fault.
- **More space is needed :**
  - Infrastructure may increase as more servers are needed to manage and monitor.

Applications of Cluster Computing :

- Various complex computational problems can be solved.
- It can be used in the applications of aerodynamics, astrophysics and in data mining.
- Weather forecasting.
- Image Rendering.
- Various e-commerce applications.
- Earthquake Simulation.
- Petroleum reservoir simulation.

Data Analytics Types

Data Analytics types

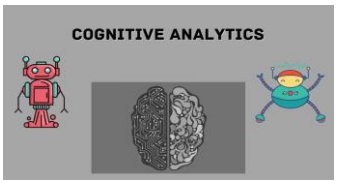


Descriptive analytics used in businesses commonly assesses historical data for finding the trends, customer patterns, areas to do improvement, etc.

Diagnostic analytics can do multiple operations on data like data discovery, data mining, and different type of bivariant data analysis like correlation, etc.

Predictive analytics comprises a variety of statistical methods from data mining, predictive modeling, machine learning, Deep learning, and past data to make future predictions or unknown events.

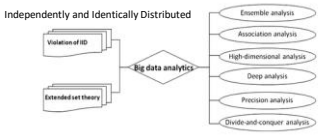
Prescriptive modeling is typically not just one individual action in analytics, but it is a combination of other actions.



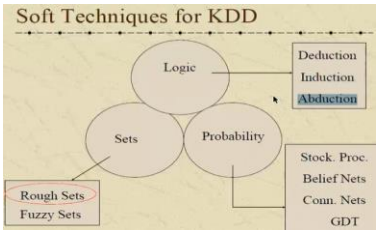
The chief advantage of using (cognitive analytics) intellectual investigation over conventional enormous information examination is that such datasets don't should be pre-labeled.

Analytics Types based on two theoretical Ideas

Two theoretical breakthroughs and six techniques in big data analytics.



Exyended Set Theory



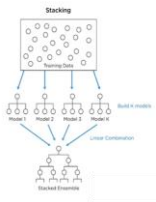
Ensemble-Techniques

Ensemble-Stacking

Advanced Ensemble Techniques

1. **Stacking**
2. **Blending**
3. **Bagging**
4. **Boosting**

- **Stacking:** It combines predictions from multiple (base-level) models to build a new model (meta-model). This meta-model is used for making predictions on the test set.
- Base level algorithms are trained based on a complete training data-set using k-fold cross validation.
- Meta model is trained on the prediction combination of all base level model as feature.
- Stacking is useful when the results of the individual algorithms are very different.



Ensemble-Blending

Blending Ensemble Technique

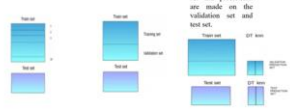
Blending follows the same approach as stacking but uses only a holdout (validation) set from the train set to make predictions.
In other words, unlike stacking, the predictions are made on the holdout set only. The holdout set and the predictions are used to build a model which is run on the test set.

Step 1: First Data is divided into train and test set.

Step 2: The train set is split into training and validation sets.

Step 3: Base models are fitted on the training set and predictions are made on the validation set and test set.

Step 4: The validation set and its predictions are used as features to build a new model and this model is used to make final predictions on the test features.



Bootstrap approach

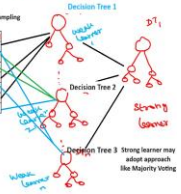
Bagging Ensemble Technique

- Bagging is combining the results of multiple models (for example, all decision trees) to get a generalized result.
- If **all the models** utilizes the **same set of data** and combine it, will it be useful?
  - Mostly, All models provides the **same result** due to the **same dataset as input**.

- **Bootstrapping** solves this problem.
- In Bootstrapping, **divide dataset** into **few subsets** with data examples replacement (row sampling).
- **Bagging /Bootstrap Aggregating uses** these subsets (bags) to get better distribution (complete set).
- The size of subsets less than the original dataset.

Row Sampling and Feature Sampling

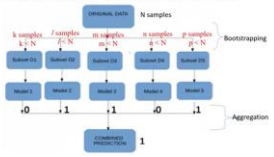
Qty	Gender	Label
5	M	0
25	F	0
10	M	1
20	F	0
5	M	1
13	F	1
0.05	M	0



Contd… approach

Bagging Ensemble Technique

- A base model (weak model) is created on each of these subsets (subsets are selected using sampling process).
- The models run in parallel, and independent of each other.
- The final predictions are determined by combining the predictions from all the models.



Boosting Ensemble

Boosting Ensemble Technique

- If a data point is incorrectly predicted by the first model, and then the next model (probably all models), will combining the predictions provide better results?
- It is solved by **boosting.**
- **Boosting** is a sequential process, where each subsequent model attempts to correct the errors of the previous model.
- The succeeding models are dependent on the previous model.

Association Type
Market Basket Analysis

High-Dimensional Type

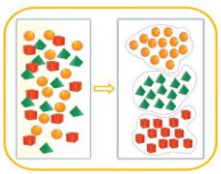
Traditional Clustering

What is clustering?

- Unsupervised learning method to classify unlabeled data into different groups

Traditional clustering methods

- K-means clustering
- Hierarchical clustering
- Density-based clustering

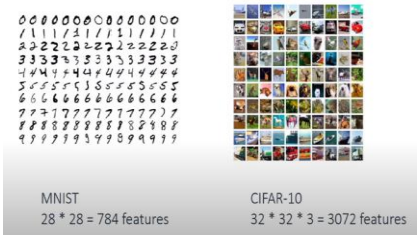


K-means approach

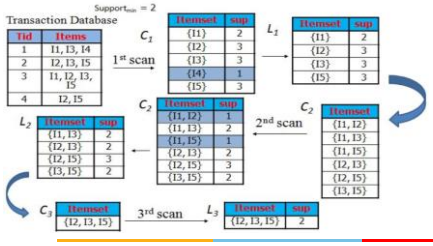
Points which are close to each other within the threshold distance are grouped together to form a cluster

In high dimensional cluster all points will become sparse and thus it becomes impossible to compute their closeness
Example: stars in galaxy

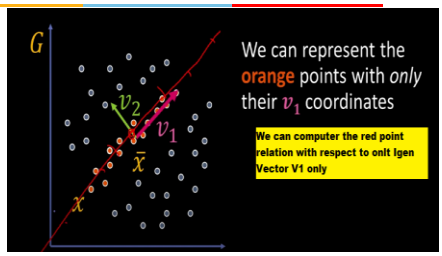
Too Heavy



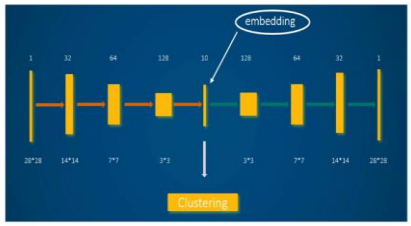
The Apriori Algorithm—An Example



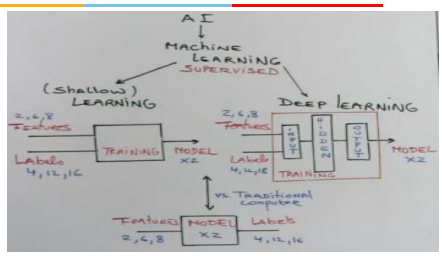
Solution : Dimension reduction



Solution : Dimension reduction



Deep Analysis Type



Precision Analysis Type

Precision analysis is used to evaluate the veracity of data from the perspective of data utility and data quality.

The fundamental difference between precision and traditional medicine is their treatment approaches. Traditional medicine often relies on generalized protocols based on the average responses of large populations. This can lead to effective treatments for some patients but less so for others. In contrast, precision medicine utilizes big data analytics to identify patterns and correlations within diverse patient datasets, enabling custom data visualization to illustrate specific biomarkers and genetic variations that influence disease progression and treatment response. As a result, precision medicine can offer therapies precisely calibrated to the patient's genetic profile and health conditions, significantly improving the chances of successful outcomes.

Divide & Conquer Analysis Type

Divide and Conquer Approach:

Divide:

- **Task Division:** Split the large dataset into smaller chunks or partitions. Each chunk can be processed independently. For instance, if you have a dataset containing billions of lines of text, you might divide it into thousands of smaller files or segments.

Conquer:

- **Local Processing:** Process each chunk independently to compute the word counts. This involves reading the text, tokenizing it into words, and counting occurrences for each word in the chunk. This step is usually performed in parallel across different machines or nodes in a distributed computing environment.

Combine:

- **Aggregation:** After processing all chunks, combine the word counts from each chunk to get the final word count for the entire dataset. This involves merging the results from all chunks and summing up the word counts for each unique word.

Analytics Contd…

What is Data Analytics?

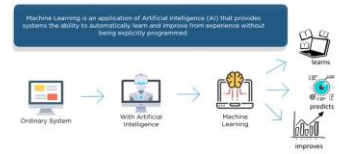
Data analytics is the process of deriving useful insights from data, It is a subset of data science, although the terms are often used interchangeably.

Data analytics itself breaks down into several sub-areas:

1. Statistical analysis,
2. machine learning,
3. Business intelligence (BI).

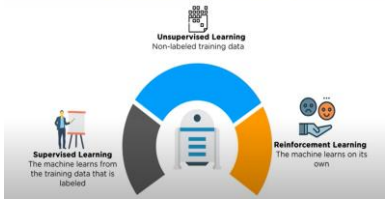
What is Machine Learning?

Machine Learning is an application of Artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.

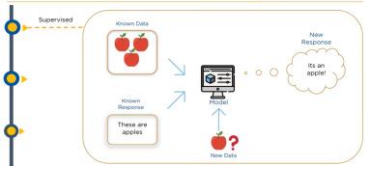


Machine Learning process

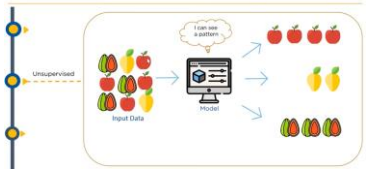
Types of Machine Learning



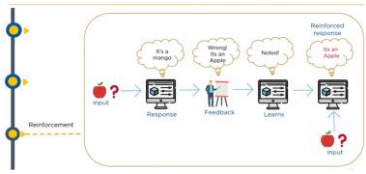
Types of Machine Learning



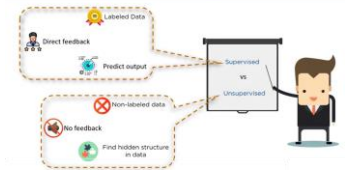
Types of Machine Learning



Types of Machine Learning



Supervised vs Unsupervised



The right Machine Learning solution?

10 ML Algorithms

1. Linear Regression
2. Decision Trees
3. Random Forest
4. Ada Boost
5. Gradient Boost
6. Logistic Regression
7. SVM
8. KNN
9. K-Means
10. Collaborative filtering

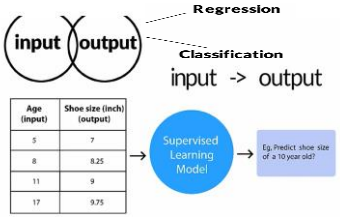
Regression

classification

VSV

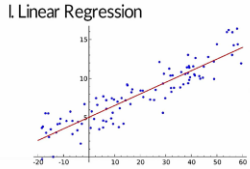
SUP

UN-SUP



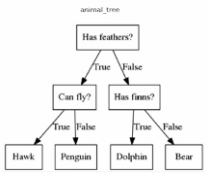
Types of Regression Models

I. Linear Regression



- Multiple Linear
- Polynomial

II. Decision Tree



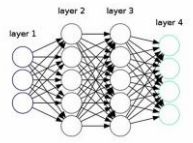
Ⅲ. Random Forests

Ensemble learning technique

↓ Blue

"Majority Wins" Model

IV. Neural Network



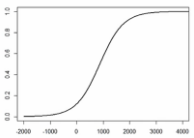
Layer 1 = Input Layer
Layer 2 = Hidden Layer
Layer 3 = Hidden Layer
Layer 4 = Output Layer

1.2 Classification

discrete

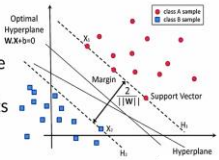
Ⅰ. Logistic regression

the output values can only be between 0 and 1



Ⅱ. Support Vector Machine

N-dimensional space that distinctly classifies data points



Ⅲ. Naive Bayes

Likelihood — Prior probability

$$P(B \mid A) = \frac{P(A \mid B)P(B)}{P(A)}$$

Posterior probability — Predictor

IV. Decision Tree, Random Forest, Neural Network

Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$P(PlayTennis = yes) = 9/14 = .64$

$P(PlayTennis = no) = 5/14 = .36$

Outlook	Y	N
sunny	2/9	3/5
overcast	4/9	0
rain	3/9	2/5

Humidity	Y	N
high	3/9	4/5
normal	6/9	1/5

Tempreature		
hot	2/9	2/5
mild	4/9	2/5
cool	3/9	1/5

Windy		
Strong	3/9	3/5
Weak	6/9	2/5

Example

$(Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong)$

$$v_{NB} = \underset{v_j \in \{yes,no\}}{\operatorname{argmax}} P(v_j) \prod_i P(a_i|v_j)$$

$$= \underset{v_j \in \{yes,no\}}{\operatorname{argmax}} P(v_j) \quad P(Outlook = sunny|v_j) P(Temperature = cool|v_j)$$

$$P(Humidity = high|v_j) P(Wind = strong|v_j)$$

$$v_{NB}(yes) = P(yes)\ P(sunny|yes)\ P(cool|yes)\ P(high|yes)\ P(strong|yes) = .0053$$

$$v_{NB}(no) = P(no)\ P(sunny|no)\ P(cool|no)\ P(high|no)\ P(strong|no) = .0206$$

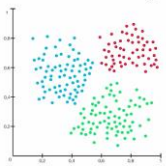
$$v_{NB}(yes) = \frac{v_{NB}(yes)}{v_{NB}(yes)+v_{NB}(no)} = 0.205 \qquad {}_{NB}(no) = \frac{v_{NB}(no)}{v_{NB}(yes)+v_{NB}(no)} = 0.795$$

2. Unsupervised Learning

Patterns from input data without references to labeled outcomes

> Clustering
> Dimensionality Reduction

2.1 Clustering



> K-means
> Hierarchical
> Mean shift
> Density-based

2.2 Dimensionality Reduction

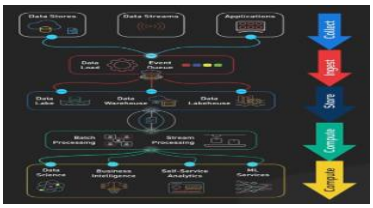
process of reducing the dimension of your feature set

> feature elimination
> feature extraction

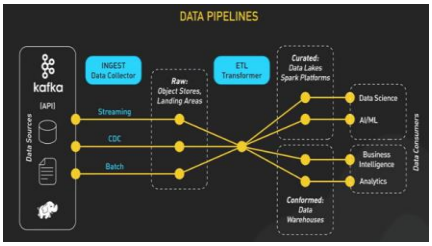
PRINCIPAL COMPONENT ANALYSIS (PCA)

Analytics Stages & CAP Therem

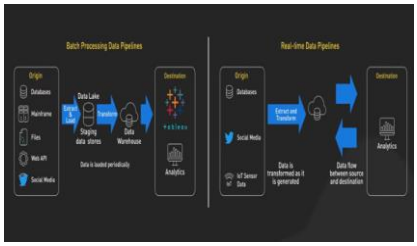
Stages…Contd…



Big Data Analytics -- Stages



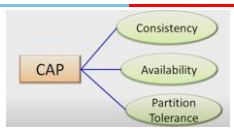
Stages…Cont.…



CAP Theorem

- Analyze different aspects of CAP theorem for choosing databases
- Formulate basic use case of CAP theorem

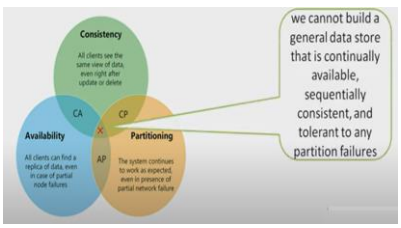
Definition

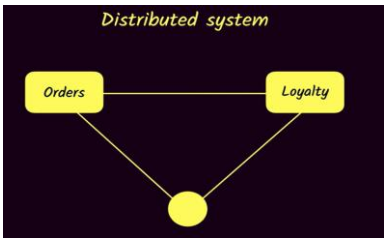


- The CAP theorem is also called as Brewer's Theorem
- It states that, in a distributed computing environment, it is impossible to provide all the three CAP guarantees.

- **CAP theorem** or Brewer's theorem states that it is impossible for a distributed computer system to simultaneously provide all three (C, A , P) guarantees.

The CAP theorem states that a distributed database system has to make a tradeoff between Consistency and Availability when a Partition occurs.

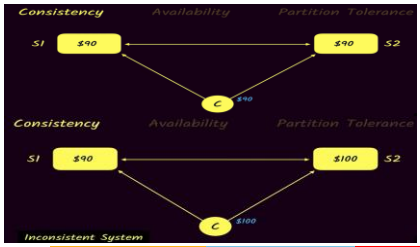




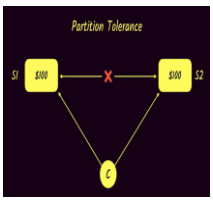
Consistency

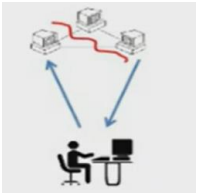
- All client has same view of data irrespective of delete or update
- It implies that every read fetches the last write

Consistent



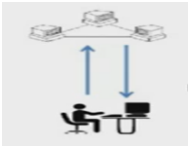
Partition Tolerance

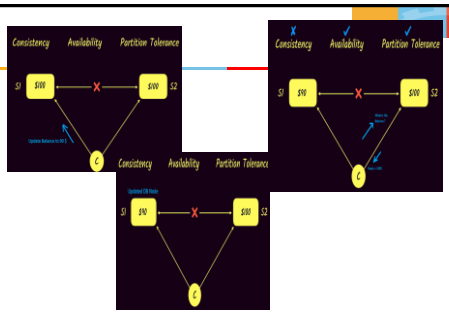




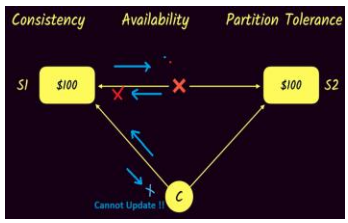
Available

- Each client can always read and write
- It implies that every reads and writes always succeed

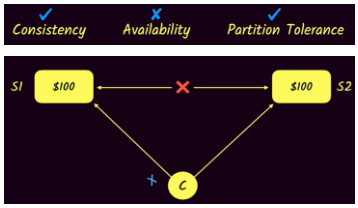


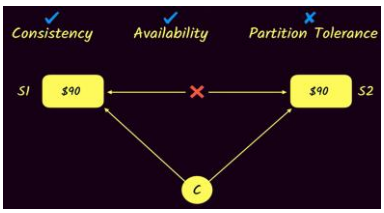


Solution



Loose Availability to obtain Consistency

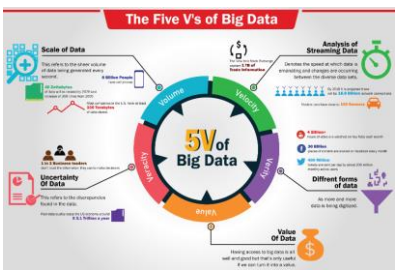




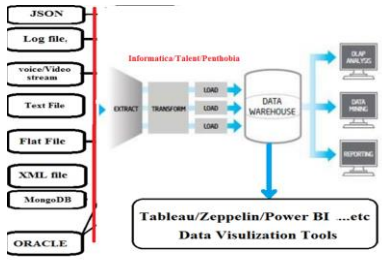
Triangular View of CAP Theorem

Big Data Analytics

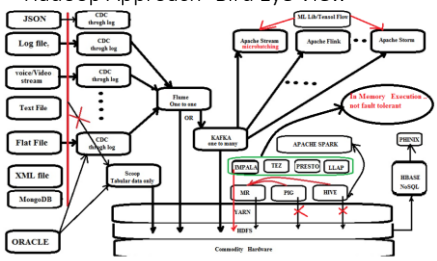
Tools Taxonomy



Traditional approach

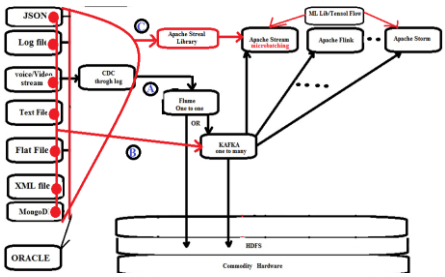


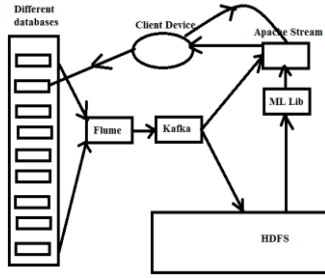
Hadoop Approach -Bird Eye View

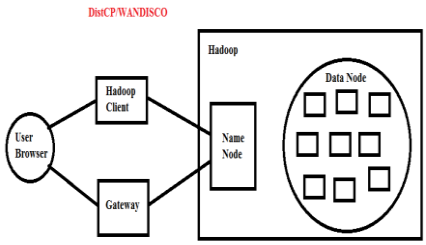


ELT APPROACH

Different Approaches – Know your requirements







Flume V/s Kafka

- F –point to point
- K - Multi
- F- Can pull data without disturbing client
- K- Need Kafka producer services on client box

Apache STORM
– Better real time processing system then Flink and Apache streaming

Big Data tools

2) Data Cleaning


Data Extraction Tools

ELT /ETL

Big Data tools

3) Data Mining

Big Data tools

4) Data Visualization


Big Data tools

5) Data reporting

Big Data tools

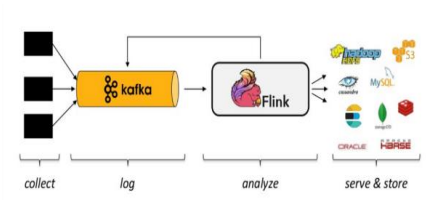
7) Data Analysis

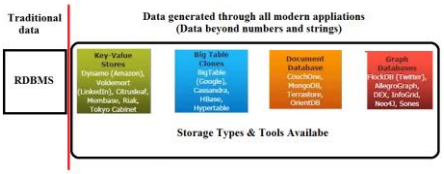

Big Data tools

8) Data Acquisition

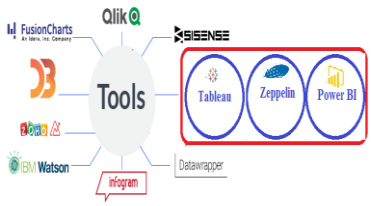
FLINK over Apache Streaming

1)Consistent Data movement
2)more realistic data streaming
3)Window based over micro batching

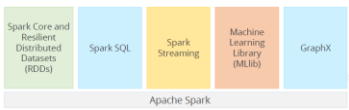




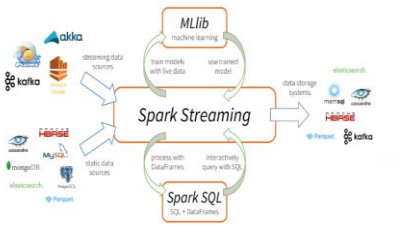
Visualization: How you want to see!!

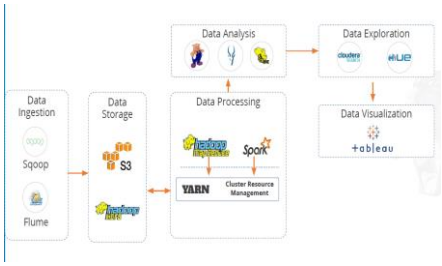


SPARK Components

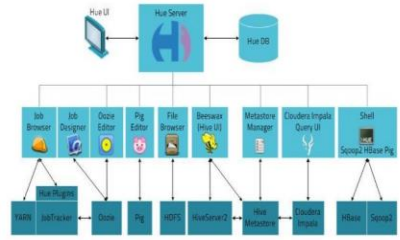


Spark Streaming





Interface to Hadoop : HUE

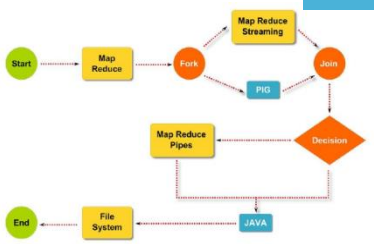


Hadoop Task Scheduler : OOZIE
Scheduling Batch Jobs

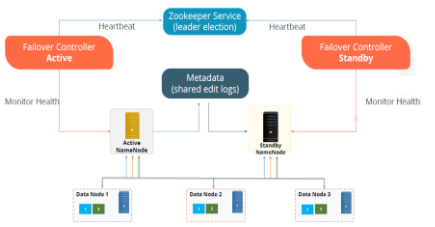
Oozie executes workflow based on

- Time Dependency (Frequency)
- Data Dependency

Sample Work Flow: OOZIE

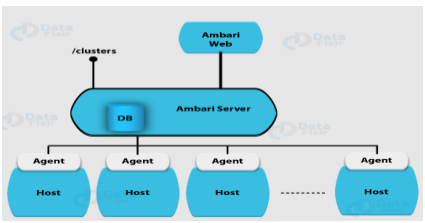


Hadoop Availability – Zookeeper
Works with HBase



Hadoop Administration : Apache Ambari

Used for management of Apache Hadoop clusters using a web UI. It also integrates with other existing applications using Ambari REST APIs.



Hadoop Administration : Apache Ambari

Ambari

Provision:

- Virtual, physical and cloud Environments.
- Deploy 10s, 100s, 1000s of Hadoop servers

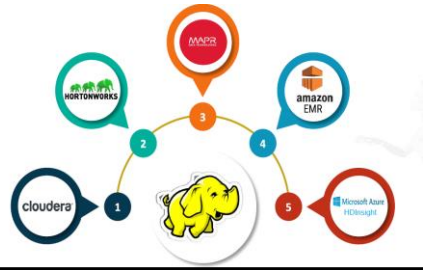
Manage:-

- Advance configuration & host Controls.
- Single point for Host controls.

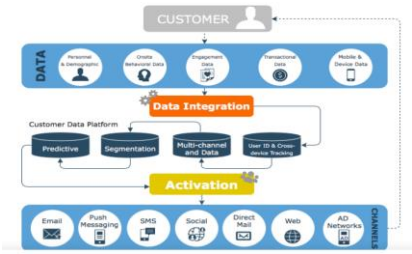
Monitor:-

- ✓ Pre-configuration metrics and alerts.
- ✓ Single pane of glass for Hadoop & system status.

Market Players



Trending - CDC



Quick Summarization

- ➢Exact need
- ➢Form in which data is available
- ➢Data type : perishable/Nonperishable

Based on the answers obtained ..Find out

- ➢What are the tools available
- ➢How to use those tools
- ➢Do you need to be a programming expert
- ➢Organization protocols/ Infra Prerequisite
- ➢paid or open source

STREAMLINE YOUR OPERATION

- Are you planning to have your own setup ? ..**Bigger Question**
- In what form data is available ?
- What is the speed at which data arrives ?
- Direct access to the data source is available ?
- Do you need to send data to multiple processing tools as well as storage device ?

Data Ingestion

- Are you going to store data using Hadoop native component or proprietary tools ? **Data storage**
- Do you need real time processing ?
- Do you need to take immediate action using data thresholds ?

Data processing

- Do you need to monitor data for decision making?

Data visualization

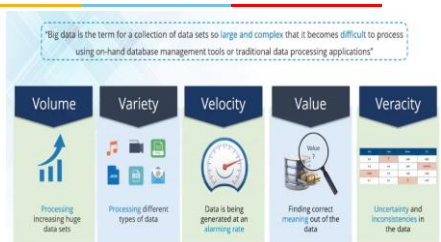
References

- https://hadoop.apache.org/ -- Apache Foundation
- https://www.ibm.com/analytics/hadoop/big-data-analytics ---IBM
- https://azure.microsoft.com/en-in/solutions/big-data/ - AZURE
- Great Learning – Raghu Raman

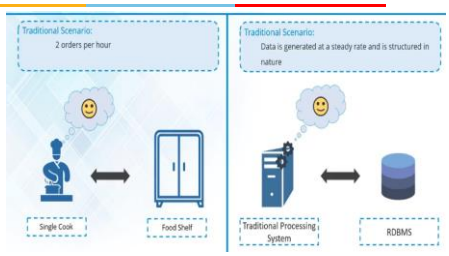
Big Data

INDEX

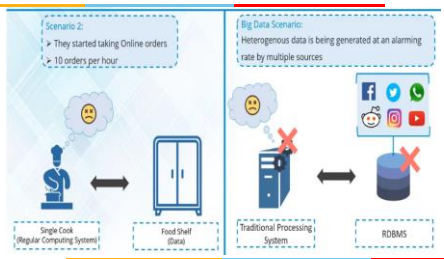
Five V's



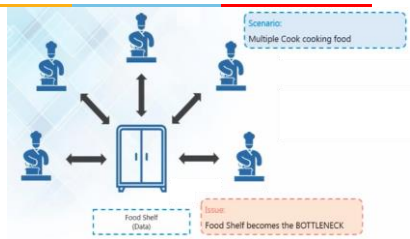
Traditional System V/s Big Data



Contd..

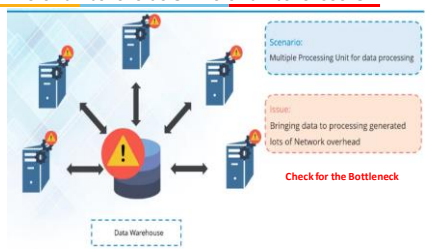


Bottleneck

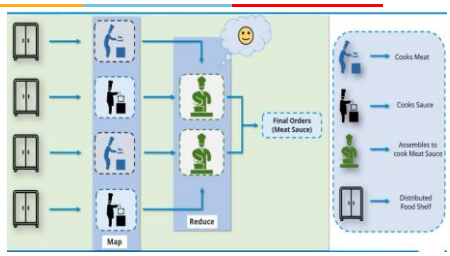


Bottleneck

More number of orders = More number of cooks



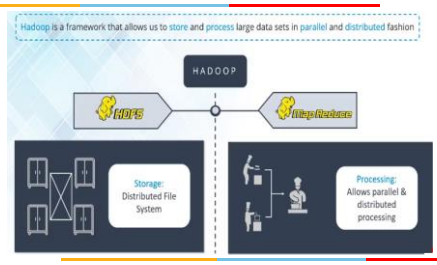
Effective Solution



Solution Framework

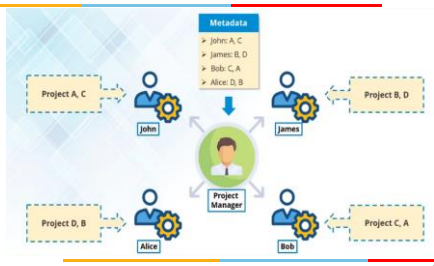
WE NEED SOME FRAMEWORK THAT CAN PROVIDES SOLUTION

Apache Hadoop

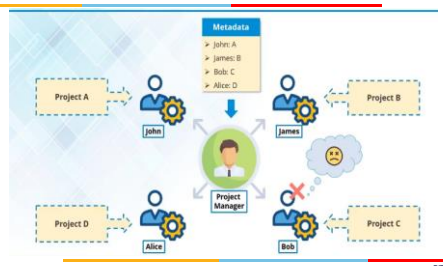


Master/Slave Approach

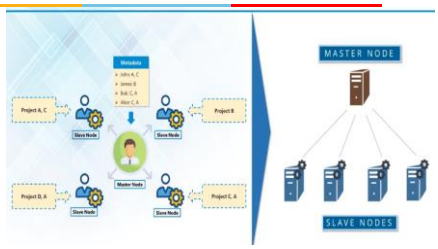
Maintain Metadata



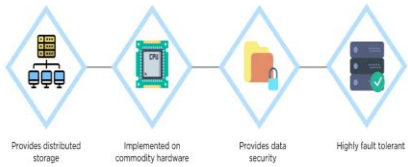
Backup Failover Solution



Evolution of Architecture

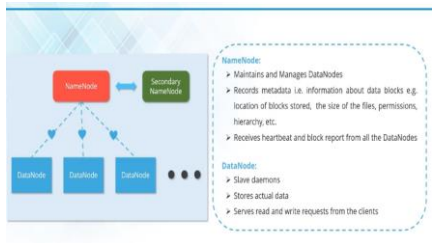


Evolution of HDFS

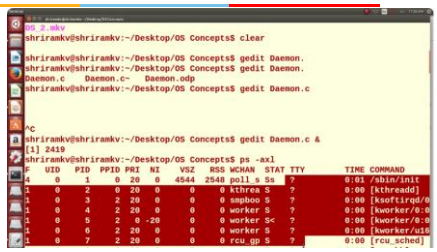


3 Core Components

Namenode & Datanode



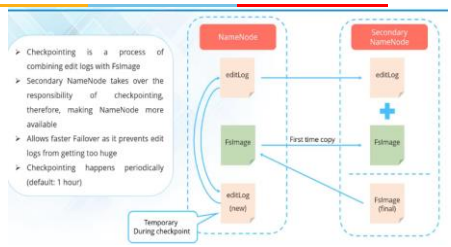
Daemon Services



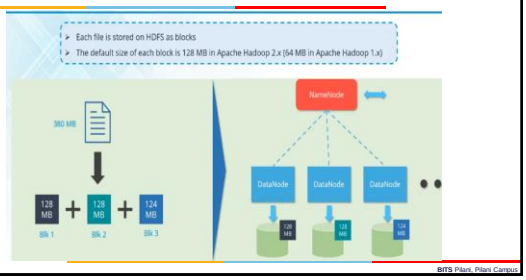
About Daemon Process

- It is a program.
- Runs in Unix/Linux without interruption or user intiation to happen.
- Executed in the background.
- Recollect, orphans.
- No terminal usage
- Command to see the daemons
- Ps -axl (results will have ? Under TTY, it tells Daemon's exisitence)

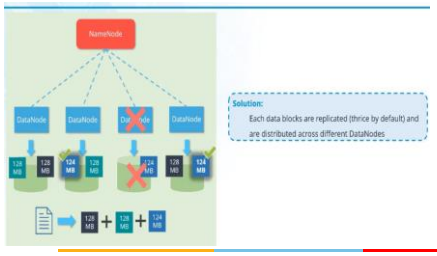
Secondary NameNode & Checkpoint



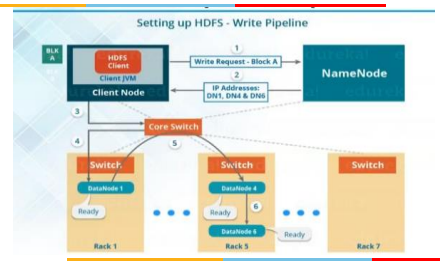
HDFS DataNode Blocks



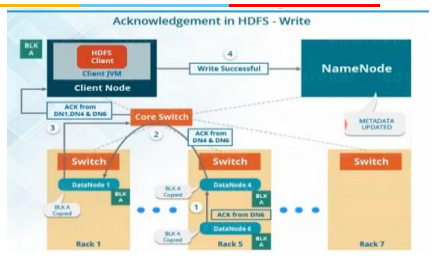
Replication



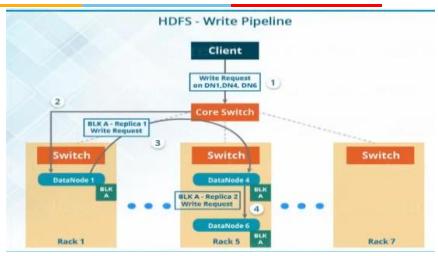
HDFS Write



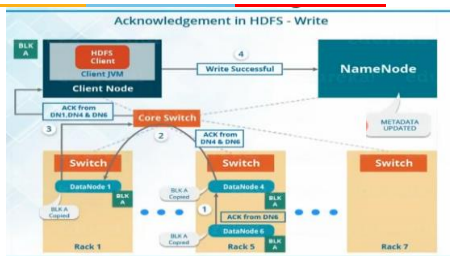
HDFS Write - Ack



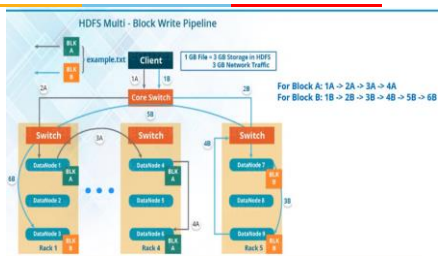
Write Request



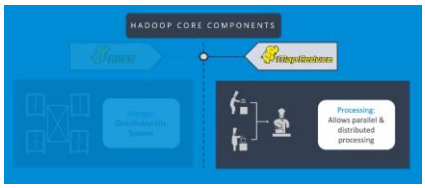
Ack



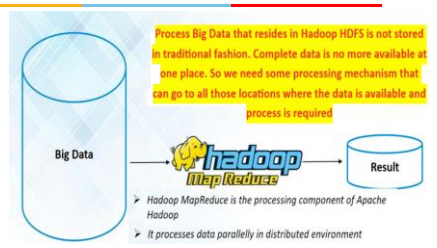
Multi Data Block write



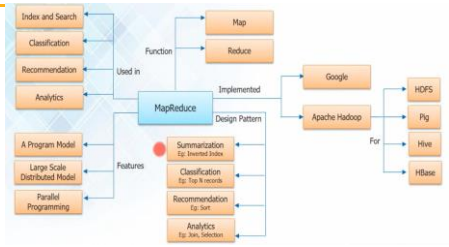
MapReduce Layer



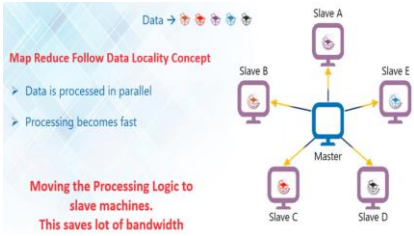
Why Map-Reduce?



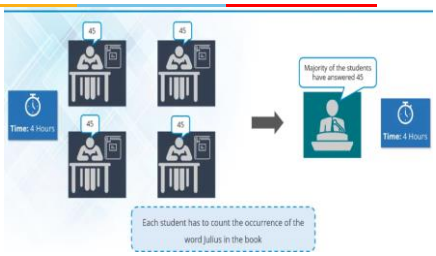
Bird Eye View



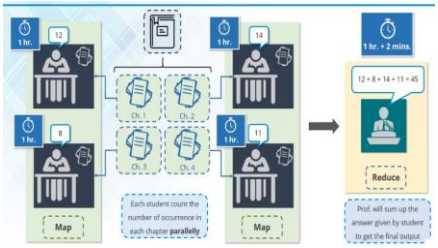
Parallel Processing scenario



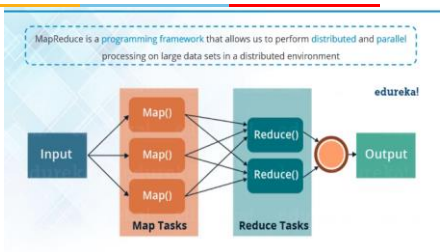
Word Count in a Book – Normal mode



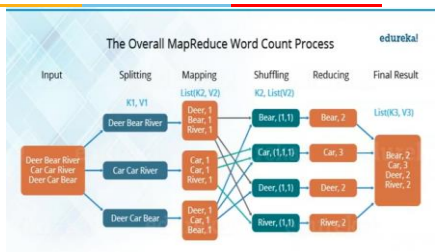
MapReduce Operation - Example



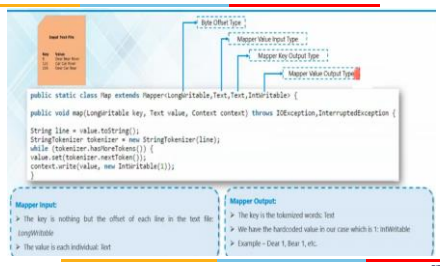
Exact Flow - MapReduce



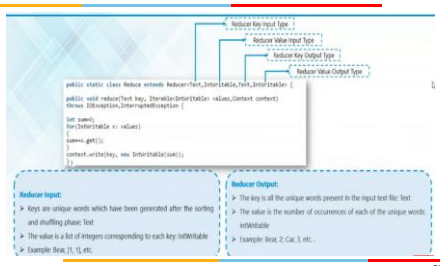
Steps with example



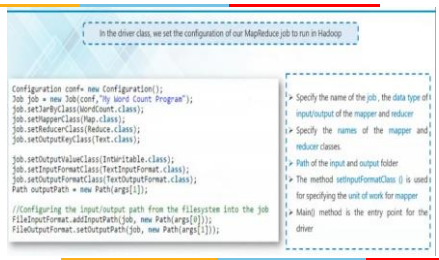
Mapper Code –Word Count Example



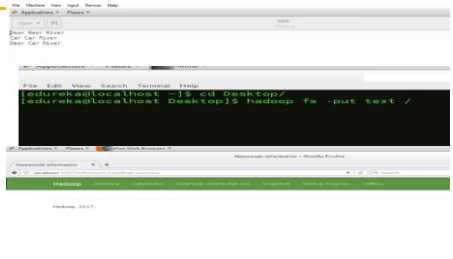
Reducer Code –Word Count Example



Driver Code for Configuration

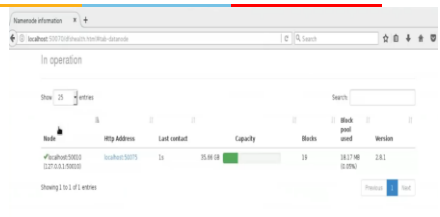


Sample Text File

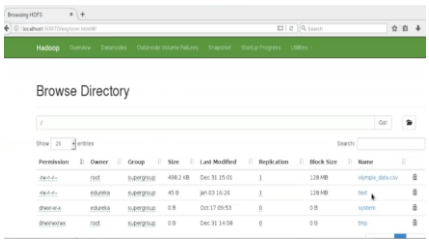


NameNode Summary

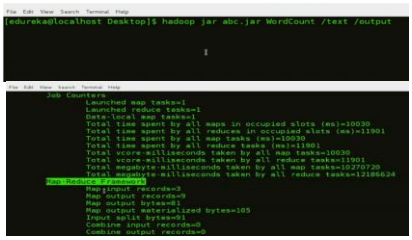
Live Node Details



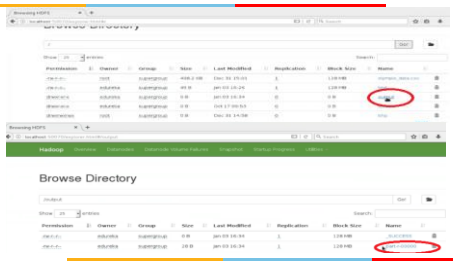
NameNode Files



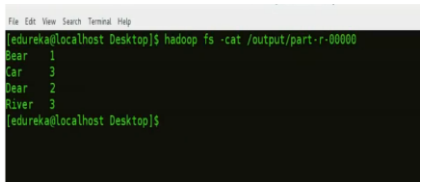
Execution



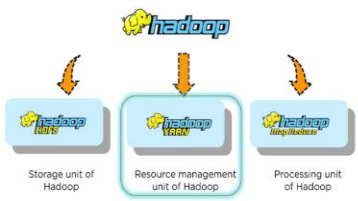
Result



Result-Contd



Basic Hadoop-2 Architecture



HDFS

Hadoop Distributed File System

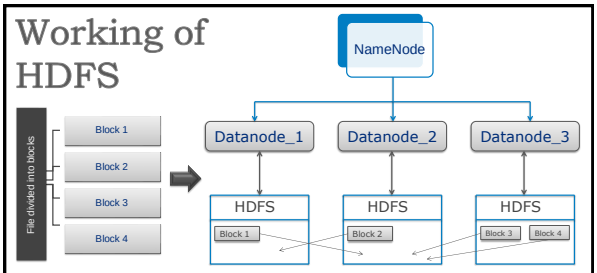
Topics Covered

- Design Goals
- Hadoop Blocks
- Rack Awareness, Replica Placement & Selection
- Permissions Model
- Anatomy of a File Write / Read on HDFS
- FileSystem Image and Edit Logs
- HDFS Check Pointing Process
- Directory Structure - NameNode, Secondary NameNode , DataNode
- Safe Mode, Trash, Name and Space Quota

HDFS Design Goals

- ***Hardware Failure*** - Detection of faults and quick, automatic recovery
- ***Streaming Data Access*** - High throughput of data access (Batch Processing of data)
- ***Large Data Sets*** - Gigabytes to terabytes in size.
- ***Simple Coherency Model –*** *Write once read many* access model for files
- *Moving computation is cheaper than moving data*

Working of HDFS



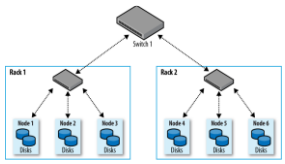
Storage & Replication of Blocks in HDFS

Blocks

- Minimum amount of data that can be read or write - 64 MB by default
- Minimize the cost of seeks
- A file can be larger than any single disk in the network
- Simplifies the storage subsystem – Same size & eliminating metadata concerns
- Provides fault tolerance and availability

Hadoop Rack Awareness

- Get maximum performance out of Hadoop cluster
- To provide reliable storage when dealing with DataNode failure issues.
- Resolution of the slave's DNS name (also IP address) to a rack id.
- Interface provided in Hadoop DNSToSwitchMapping, Default implementation is ScriptBasedMapping



Rack Topology - /rack1 & /rack2

Rack Awareness Configuration

File : hdfs-site.xml
Property : topology.node.switch.mapping.impl
Default Value : ScriptBasedMapping class

Property : topology.script.file.name
Value : <Absolute path to script file>

Sample Data (topology.data)

192.168.56.101 /dc1/rack1
192.168.56.102 /dc1/rack2
192.168.56.103 /dc1/rack2

Sample Command : ./topology.sh 192.168.56.101

Output : /dc1/rack1

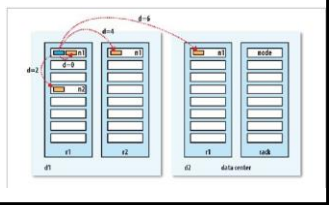
```
HADOOP_CONF=${HADOOP_HOME}/conf
while [ $# -gt 0 ] ; do
  nodeArg=$1
  exec< ${HADOOP_CONF}/topology.data
  result=""
  while read line ; do
    ar=( $line )
    if [ "${ar[0]}" = "$nodeArg" ] ; then
      result="${ar[1]}"
    fi
  done
  shift
  if [ -z "$result" ] ; then
    echo -n "/default/rack "
  else
    echo -n "$result "
  fi
done
```

Ref Hadoop Wiki **Sample Script**
http://wiki.apache.org/hadoop/topology_rack_awareness_scripts

Replica Placement

Distance b/w Hadoop Client and DataNode		
Same Node	:	d=0
Same Rack	:	d=2
Same Data Centre different rack	:	d=4
Different Data Centre	:	d=6

- Critical to HDFS reliability and performance
- Improve data reliability, availability, and network bandwidth utilization



Replica Placement cont..

Default Strategy :

a) First replica on the same node as the client.

b) Second replica is placed on a different rack from the first (off-rack) chosen at random

c) Third replica is placed on the same rack as the second, but on a different node chosen at random.

d) Further replicas are placed on random nodes on the cluster

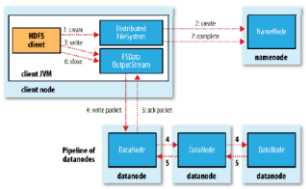
Replica Selection - HDFS tries to satisfy a read request from a replica that is closest to the reader.

Permissions Model

- Directory or file flag, permissions, replication, owner, group, file size, modification date, and full path.
- User Name : 'whoami'
- Group List : 'bash -c groups'
- Super-User & Web Server

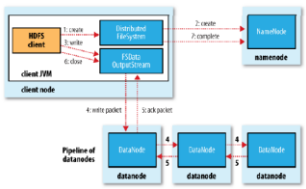
Anatomy of a File Write

- Client creates the file by calling create() method
- NameNode validates & processes the create request
- Spilt file into packets (DataQueue)
- DataStreamer asks NameNode for block / node mapping & pipelines are created among nodes.



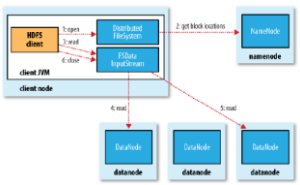
Anatomy of a File Write (cont..)

- DataStreamer streams the packets to the first DataNode
- DataNode forwards the copied packet to the next DataNode in the pipeline
- DFSOutputStream also maintains the ack queue and removes the packets after receiving acknowledgement from the DataNodes
- Client calls close() on the stream



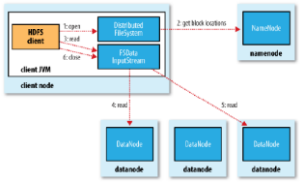
Anatomy of a File Read

- Client calls open() on the FileSystem object
- DistributedFileSystem calls the NameNode to determine the locations of the blocks
- NameNode validates request & for each block returns the list of DataNodes.
- DistributedFileSystem returns an input stream that supports file seeks to the client



Anatomy of a File Read (cont..)

- Client calls read() on the stream
- When the end of the block is reached, DFSInputStream will close the connection to the DataNode, then find the best DataNode for the next block.
- Client calls close() on the stream

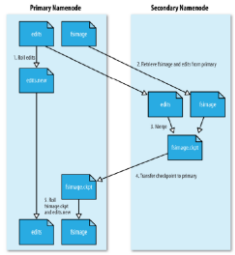


FileSystem Image and Edit Logs

- **fsimage** file is a persistent checkpoint of the file-system metadata
- When a client performs a write operation, it is first recorded in the edit log.
- The NameNode also has an in-memory representation of the files-ystem metadata, which it updates after the edit log has been modified
- Secondary NameNode is used to produce checkpoints of the primary's in-memory files-ystem metadata

Check Pointing Process

- Secondary NameNode asks the primary to roll its edits file, so new edits go to a new file.
- NameNode sends the fsimage and edits (using HTTP GET).
- Secondary NameNode loads fsimage into memory, applies each operation from edits, then creates a new consolidated fsimage file.
- Secondary NameNode sends the new fsimage back to the primary (using HTTP POST).
- Primary replaces the old fsimage with the new one. Updates the fstime file to record the time for checkpoint.



Directory Structure

NameNode (On NameNode only)
${dfs.name.dir}/current/VERSION
/edits
/fsimage
/fstime

Secondary NameNode (On SecNameNode Only)
${fs.checkpoint.dir}/current/VERSION
/edits
/fsimage
/fstime
/previous.checkpoint/VERSION
/edits
/fsimage
/fstime

DataNode (On all DataNodes)
${dfs.data.dir}/current/VERSION
/blk_<id_1>
/blk_<id_1>.meta
/blk_<id_2>
/blk_<id_2>.meta
/…
/blk_<id_64>
/blk_<id_64>.meta
/subdir0/
/subdir1/
/…

Block Count for a directory
dfs.datanode.numblocks property

Safe Mode

- On start-up, NameNode loads its image file (fsimage) into memory and applies the edits from the edit log (edits).
- It does the check pointing process itself. without recourse to the Secondary NameNode.
- Namenode is running in safe mode (offers only a read-only view to clients)
- The locations of blocks in the system are not persisted by the NameNode - this information resides with the DataNodes, in the form of a list of the blocks it is storing.
- Safe mode is needed to give the DataNodes time to check in to the NameNode with their block lists
- Safe mode is exited when the minimal replication condition is reached, plus an extension time of 30 seconds.

HDFS Safe Mode

SafeMode Properties – Configure safe mode as per cluster recommendations.

dfs.replication.min	*Default - 1*	Minimum no. of replicas for file wrtie success
dfs.safemode.threshold.pct	*Default – 0.999*	Min. portion of blocks statisfying the minimum replication to leave safe mode. Value 0 – Forces NN not to start in safemode Value 1 – Ensurs NN never leave safemode
dfs.safemode.extension	*Default – 30,000*	Extension time in milliseconds before NN leaves safemode

SafeMode Admin Commands – A command option for dfsadmin command: *hadoop dfsadmin -safemode*

get : Get the NameNode safemode status	*enter* : NameNode enters safemode
Wait : Wait for NameNode to exit safemode	*leave* : *NameNode leaves the safemode*

HDFS Trash – Recycle Bin

When a file is deleted by a user, it is not immediately removed from HDFS. HDFS moves it to a file in the /trash directory.

File	:	core-site.xml
Property	:	fs.trash.interval
Description	:	Number of minutes after which the checkpoint gets deleted.

A file remains in /trash for a configurable amount of time. After the expiry of its life in /trash, the NameNode deletes the file from the HDFS namespace.

File	:	core-site.xml
Property	:	fs.trash.checkpoint.interval
Description	:	Number of minutes between trash checkpoints. Should be smaller or equal to fs.trash.interval.

Undelete a file: User needs to navigate the /trash directory and retrieve the file by using mv command.

HDFS Quotas

Name Quota - a hard limit on the number of file and directory names in the tree rooted at that directory.

dfsadmin -setQuota <N> <directory>...	Set the name quota to be N for each directory.
dfsadmin -clrQuota <directory>...	Remove any name quota for each directory.

Space Quota - a hard limit on the number of bytes used by files in the tree rooted at that directory.

dfsadmin -setSpaceQuota <N> directory>..	Set the space quota to be N bytes for each directory.
dfsadmin -clrSpaceQuota <directory>...	Remove any spce quota for each directory.

Reporting Quota - count command of the HDFS shell reports quota values and the current count of names and bytes in use. With the -q option, also report the name quota value set for each directory, the available name quota remaining, the space quota value set, and the available space quota remaining.

fs -count -q <directory>..

FS Shell – Some Basic Commands

chgrp - Change group association of files.
Usage: hdfs dfs -chgrp [-R] GROUP URI [URI ...]

chmod - Change the permissions of files.
Usage: hdfs dfs -chmod [-R] <MODE[,MODE]... | OCTALMODE> URI [URI ...]

chown - Change the owner of files.
Usage: hdfs dfs -chown [-R] [OWNER][:[GROUP]] URI [URI]

du - Displays sizes of files and directories
Usage: hdfs dfs -du [-s] [-h] URI [URI ...]
The -s option will result in an aggregate summary of file lengths being displayed, rather than the individual files.
The -h option will format file sizes in a "human-readable" fashion.

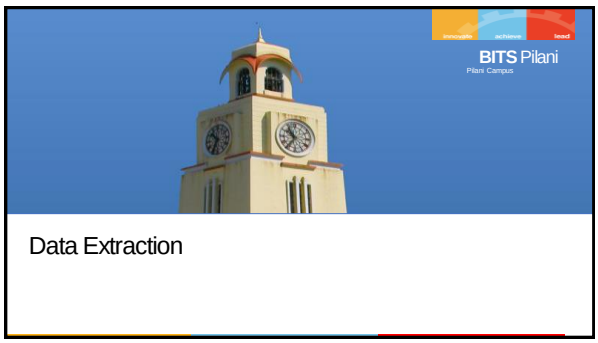
dus - Displays a summary of file lengths
Usage: hdfs dfs -dus <args>

http://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-common/FileSystemShell.html
http://hadoop.apache.org/docs/r1.2.1/file_system_shell.html

DfsAdmin Command

- bin/hadoop dfsadmin [Generic Options] [Command Options]

-safemode enter / leave / get / wait	Safe mode maintenance command. Safe mode can also be entered manually, but then it can only be turned off manually as well.
-report	Reports basic filesystem information and statistics.
-refreshNodes	Re-read the hosts and exclude files to update the set of Datanodes that are allowed to connect to the Namenode and those that should be decommissioned or recommissioned.
-metasave filename	Save Namenode's primary data structures to filename in the directory specified by hadoop.log.dir property. filename is overwritten if it exists. filename will contain one line for each of the following 1. Datanodes heart beating with Namenode 2. Blocks waiting to be replicated 3. Blocks currrently being replicated 4. Blocks waiting to be deleted



Data Extraction

Data Extraction

What is Data Extraction?

Extracting data is key in managing and analyzing information. As firms collect stacks of data from different places, finding important info becomes crucial. We gather specific info from different places like databases, files, websites, or APIs to analyze and process it better. Doing this helps us make smart decisions and understand things better.

Gathering data from various places, changing it so we can use it, and putting it where we need it for review is what data extraction is about.

Data Extraction – Capture and Transform

Why we need Data Extraction ?

1. **Facilitating Decision-Making:** It gives us what has happened (historical trends), what's happening (current patterns), and what might happen (emerging behaviours). This helps firms or organizations make plans for better business.
2. **Empowering Business Intelligence:** Business smarts need relevant and timely data for helpful insights. This makes a group more focused on data.
3. **Enabling Data Integration:** Firms often hold data in different systems. Taking out the data makes it mix better. This gives an all-around and fitting view of firm-wide data.
4. **Automation for Efficiency:** Automated data extraction processes boost efficiency and less hands-on need. Automation offers a smooth, steady way to deal with lots of data.

Data Extraction Process

- **Filtering:** It demands meticulous sifting through information, delicately pinpointing and extracting details that harmonize with predetermined standards. This pivotal stage ensures that data emerges, filtering noises and elevating the precision of subsequent evaluations.
- **Parsing:** Is a systematic exploration of the data framework, unravelling it into elements that can be further maneuvered and processed. This holds particular importance when grappling with information that lacks a clear structure or adheres to a semi-organized format.
- **Structuring:** Raw data often lacks a coherent arrangement. In the journey of data extraction, an art known as organizing is employed to structure and format the information in a manner conducive to analysis process.

Extract Product information from an E-Commerce Website

Data Extraction from CSV using Python

```
import pandas as pd
from io import StringIO

# Sample in-line CSV data for example purpose
csv_data = """Name,Age,Occupation
John,25,Engineer
Jane,30,Teacher
Bob,22,Student
Alice,35,Doctor"""

# Read the CSV data into a DataFrame
df_csv = pd.read_csv(StringIO(csv_data))

# Extract data based on the 'Occupation' column
engineers_data_csv = df_csv[df_csv['Occupation'] == 'Engineer'][['Name', 'Age']]

# Display the extracted data
print("Data from CSV:")
print(engineers_data_csv)
```

Output:

```
Data from CSV:
   Name  Age
0  John   25
```

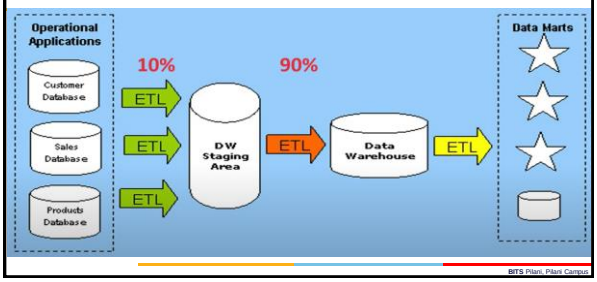

Data Extraction –Web Scraping

```
import requests
from bs4 import BeautifulSoup

# URL for web scraping
url = "https://www.geeksforgeeks.org/basic/"

# Send a GET request to the URL
response = requests.get(url)

# Check if the request was successful (status code 2
if response.status_code == 200:
    # Parse the HTML content
    soup = BeautifulSoup(response.text, 'html.parser')

    # Extract article titles
    article_titles = [title.text.strip() for title in soup.find_all('div', class_='head')]  # Adjust the class base

    # Display the extracted data
    print("Article Titles from GeeksforGeeks:")
    for title in article_titles:
        print("- " + title)
else:
    print("Failed to retrieve the webpage. Status code:", response.status_code)
```

Output:

```
Article Titles from GeeksforGeeks:
- Array Data Structure
- Tribal Leader Vishnu Deo Sai (Biography): New Chhattisgarh Chief Minister
- Protection Against False Allegations and Its Types
- Double Angle Formulas
- MariaDB CHECK Constraint
- Cristiano Ronaldo Net Worth 2024: Football Success and Endorsements
- PM Modi Proposes to host COP 33 Summit in 2028 in India
- Why Ghol fish Declared as State Fish of Gujarat?
```

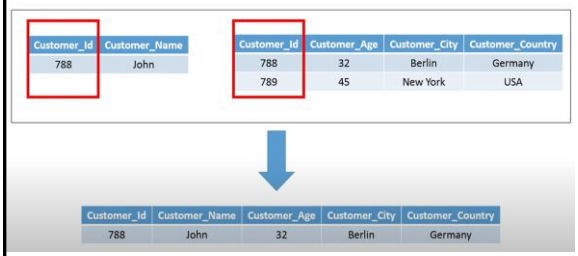
Data Transformation



Data transformation process

Data Joining
Data Duplication
Keys Restructuring
Data Cleansing
Data Validation
Data Format Revision
Data Derivation
Data Integration
Data Filtering
Data Splitting

Joining (two table join)

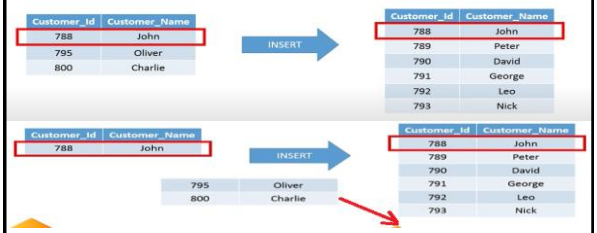


Customer_Id	Customer_Name
788	John

Customer_Id	Customer_Age	Customer_City	Customer_Country
788	32	Berlin	Germany
789	45	New York	USA

Customer_Id	Customer_Name	Customer_Age	Customer_City	Customer_Country
788	John	32	Berlin	Germany

Duplication



Customer_Id	Customer_Name
788	John
795	Oliver
800	Charlie

INSERT

Customer_Id	Customer_Name
788	John
789	Peter
790	David
791	George
792	Leo
793	Nick

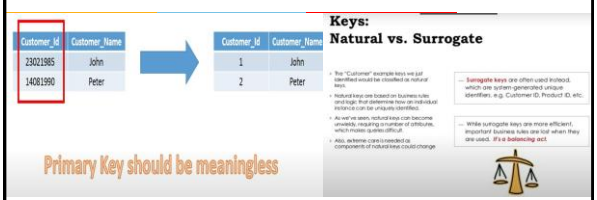
Customer_Id	Customer_Name
788	John

INSERT

Customer_Id	Customer_Name
795	Oliver
800	Charlie

Customer_Id	Customer_Name
788	John
789	Peter
790	David
791	George
792	Leo
793	Nick

Keys Restructuring



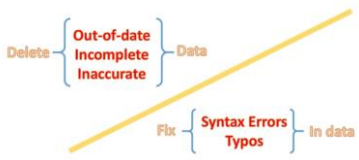
Customer_Id	Customer_Name
23021985	John
14081990	Peter

Customer_Id	Customer_Name
1	John
2	Peter

Keys:
Natural vs. Surrogate

Primary Key should be meaningless

Cleansing

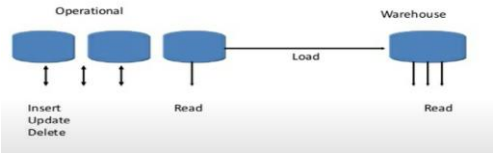


Integrity

Only Data Insert and Read Operations are possible on warehouse

Nonvolatile

Typically data in the data warehouse is not updated or deleted.



Validation

Range checks

- i.e. is between 1 and 100

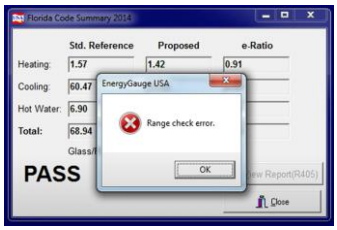
Please type in your age:

Type checks

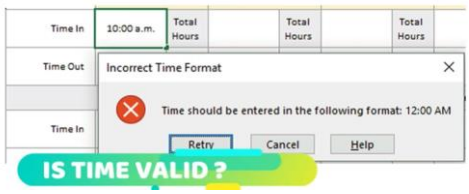
- i.e is numeric and not text

Please type in your age:

In OLTP Systems



Time Validation



Formatting

male	M
female	F

20/03/1989	20-March-1989
21/08/1999	21-April-1999

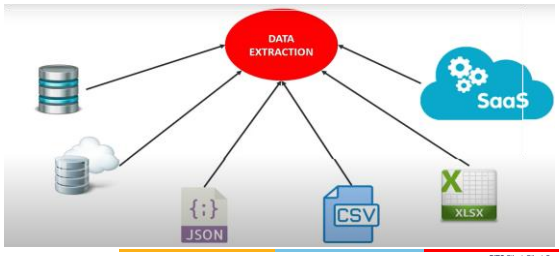
Gender	DOB
Male	1989-03-20
Female	1999-08-21

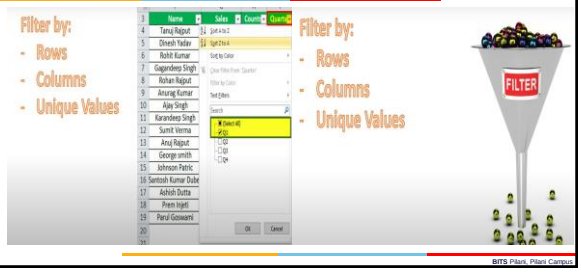
Derivation

Customer_Name	Customer_dob
John	20-08-1990
Peter	15-07-1999

Customer_Name	Customer_dob	Customer_Age (years)
John	20-08-1990	30
Peter	15-07-1999	21

Intigration





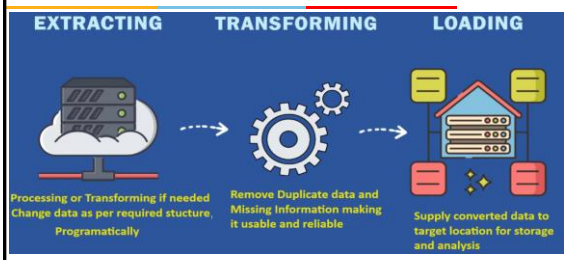
Splitting

Address
Carlton Street 15, 78845, Munich; Bavaria; Germany

Street_Name	House_Number	Postal_Code	City	State	Country
Carlton Street	15	78845	Munich	Bavaria	Germany

Avoid joins

ETL Process



Data Extraction tool-Example

Data Ingestion

Data Ingestion is the process of importing and loading data into a system. It's one of the most critical steps in any data analytics workflow. A company must ingest data from various sources.

Data ingestion and ETL are two very different processes. Data ingestion is importing data into a database or other storage engine, while ETL is extracting, transforming, and loading.

Data extraction is the first step in the data ingestion process, which involves pulling data from sources and preparing it for use.

Extraction liberates data so you can fully use its potential. Data ingestion transforms information into insight

Difference

Data Ingestion

Data ingestion is a process that involves copying data from an external source (like a database) into another storage location (like a database). In this case, it's typically done without any changes to the data.

For example, if you have an Amazon S3 bucket containing some files that need to be imported into your database, then data ingestion would be required to move those files into your database location.

ETL

ETL stands for extract transform load; it's a process that involves taking data from one system and transforming it so that it can be loaded into another system for use there.

In this case, rather than just copying data from one location to another without making any changes.

Ingestion and Integration

1) Data Ingestion - The act or process of introducing data into a database or other storage repository. Often this involves using an ETL (extract, transform, load) tool to move information from a source system (like Salesforce) into another repository like SQL Server or Oracle.

2) Data Integration - The process of combining multiple datasets into one dataset or data model that can be used by applications, particularly those from different vendors like Salesforce and Microsoft Dynamics CRM.

Ingestion Types

Real-time ingestion involves streaming data into a data warehouse in real-time, often using cloud-based systems that can ingest the data quickly, store it in the cloud, and then release it to users almost immediately.

Batch ingestion involves collecting large amounts of raw data from various sources into one place and then processing it later. This type of ingestion is used when you need to order a large amount of information before processing it all at once.

Ingestion process

Data ingestion process begins by prioritizing data sources, validating individual files and routing data items to the correct destination

1)Collection of Data from the source

2)Filtering

3)Route to one or more data stores

Ingestion challenges

1. **Coding and maintenance** are two enormous challenges that can take time to overcome. Sometimes it's easier to throw out old data than figure out how to organize it so that you can use it for future projects.
2. **Latency** is another challenge companies face when trying to ingest new data. If you're waiting too long between ingesting your data and using it in another application or process, then there may be significant delays in getting things done!
3. **Data quality** is also a challenge—how often have you had to clean up or reprocess old data because there wasn't enough information or detail? Sometimes we'll even need to go back through old files multiple times before they're ready for our purposes!
4. Finally, there's the problem of **capturing all this information** in the first place—how do we even begin collecting all this data without losing any of its required information?

Parallelism

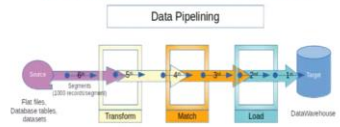
Methods of Parallelism

- Data Pipelining
- Data Partitioning
- Combining both Pipelining and Partitioning
- Dynamic Data Repartitioning

Pipelining

Data Pipelining

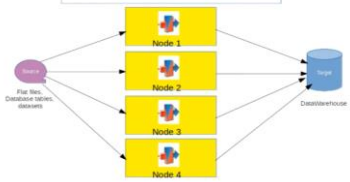
Entire data is partitioned and each partition Data will be Moving from one stage to next stage.



Partitioning

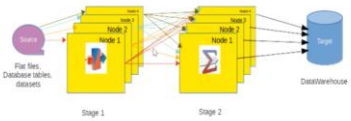
Data Partitioning Parallelism

All Stages will be running parallelly. Each note will have all stages running parallelly



Dynamic Data Repartitioning

Dynamic Data Repartitioning

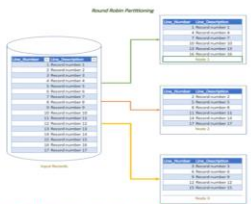


Data Clubbed based on customer last name

Data Partitioned Based on City

Round Robin Partition

Round Robin Partitioning

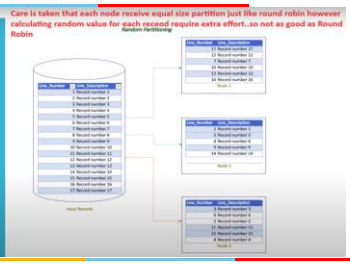


Approximately equal sized partitions are made

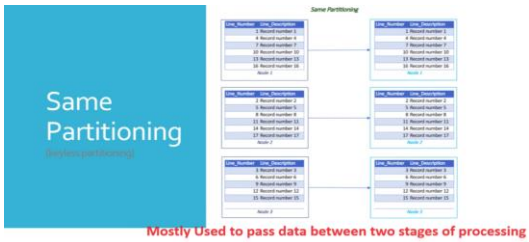
Random Partitioning

Random Partitioning

Care is taken that each node receive equal size partition just like round robin however calculating random value for each receod require extra effort..so not as good as Round Robin

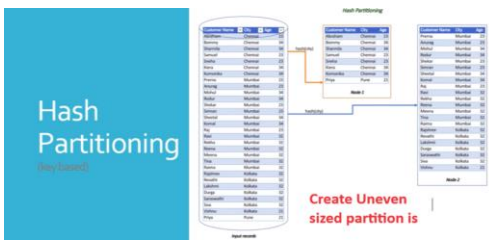


Same Partitioning



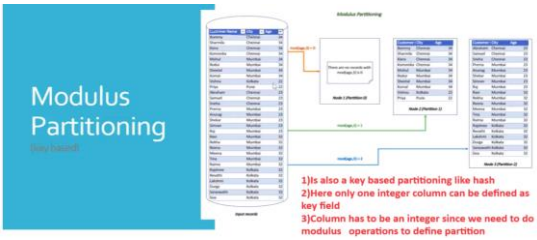
Mostly Used to pass data between two stages of processing

Partition and Divide-and-conquer techniques in Distributed system



Create Uneven sized partition is

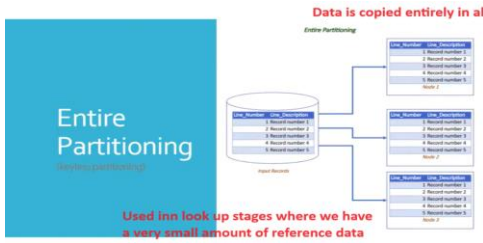
Modulus Partition



1)Is also a key based partitioning like hash
2)Here only one integer column can be defined as key field
3)Column has to be an integer since we need to do modulus operations to define partition

Entire Partitioning

Data is copied entirely in all nodes



Used inn look up stages where we have a very small amount of reference data

In-node and over-the-network latencies

In-Node Latency: In-node latency refers to the time it takes for a process or thread to access data or perform computation within the local memory of a single node. In-node latency is typically much lower compared to network latencies since data can be accessed and processed directly from local memory, avoiding the communication overhead associated with network transmission.

Over-the-Network Latency: Over-the-network latency refers to the time it takes for data to be transmitted between nodes over the network. Network latencies are typically higher than in-node latencies due to factors such as network congestion, routing delays, and transmission times.

The difference between in-node and over-the-network latencies is crucial in distributed systems, as it determines the cost of data communication and influences the design of communication patterns between nodes. Minimizing over-the-network latencies is essential to achieve better performance and responsiveness in distributed systems.

Data and Task Locality

Data Locality: Data locality refers to the degree to which data accessed by a process or task is physically located close to the processing unit (CPU, GPU) that needs it. High data locality means that the data accessed is already present in the local memory of the processing unit, reducing the need for expensive network communication or disk I/O. Data locality is vital in reducing access latencies and optimizing data processing in distributed systems.

Task Locality: Task locality refers to the placement of related computation or tasks close to each other, often within the same node. Task locality can improve performance by reducing communication costs between tasks, as communication within a node is faster than communication between nodes.

In distributed systems, optimizing data and task locality helps to reduce communication overhead and latency, leading to improved system performance and efficiency.

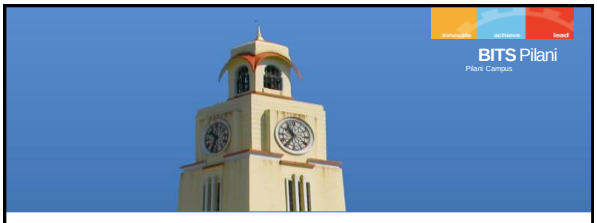
Communication cost

Communication cost refers to the resources, time, and bandwidth required to exchange data or messages between nodes in a distributed system. Communication costs include both the time taken to transmit data over the network (network latency) and any additional processing overhead required for serialization, deserialization, and handling communication protocols.

Minimizing communication cost is essential for achieving efficient data exchange and coordination among distributed nodes. Strategies such as data partitioning, data replication, and message aggregation can help reduce communication overhead and optimize data access and processing in distributed systems.

Conclusion

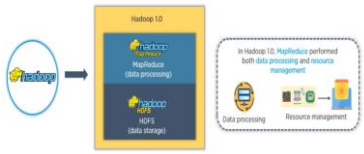
Efficient management of these factors can lead to better performance, reduced latency, and improved resource utilization in distributed computing environments.



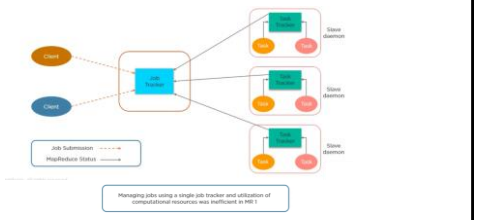
YARN

Todays Agenda

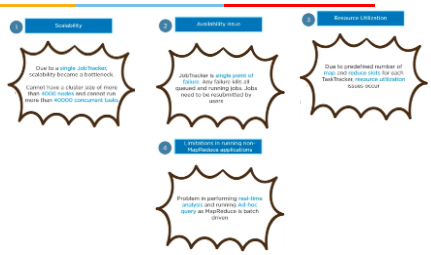
Hadoop-1



Hadoop-1

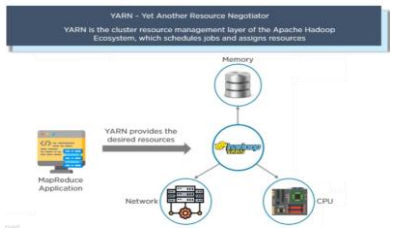


Limitations

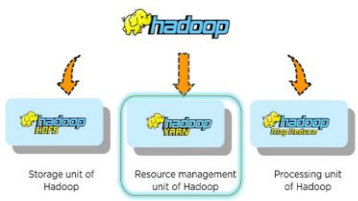


Solution

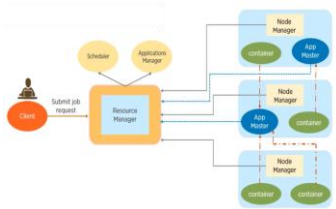
What is YARN ?



Basic Hadoop-2 Architecture

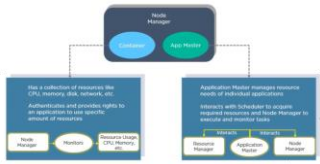


Architecture

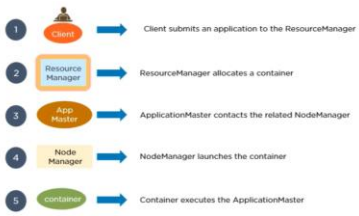


Resource Manager

Node Manager

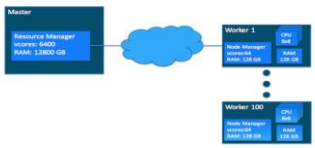


Team work



Global View

YARN currently defines two resources, *vcores* and *memory*. Each NodeManager tracks its own local resources and communicates its resource configuration to the ResourceManager, which keeps a running total of the cluster's available resources. By keeping track of the total, the ResourceManager knows how to allocate resources as they are requested. (Vcore has a special meaning in YARN. You can think of it simply as a "usage share of a CPU core." If you expect your tasks to be less CPU-intensive (sometimes called I/O-intensive), you can set the ratio of vcores to physical cores higher than 1 to maximize your use of hardware resources.)



Container

Containers are an important YARN concept. You can think of a container as a request to hold resources on the YARN cluster. Currently, a container hold request consists of vcore and memory shown in Figure

Container as a hold (left), and container as a running process (right)

Once a hold has been granted on a host, the NodeManager launches a process called a *task*. The right side of Figure shows the task running as a process inside a container.

Running Process/App Master

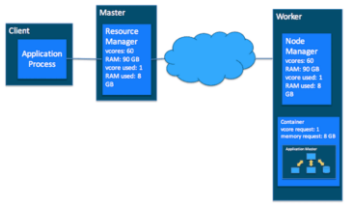
1. The application starts and talks to the ResourceManager for the cluster:

Figure 5: Application starting up before tasks are assigned to the cluster

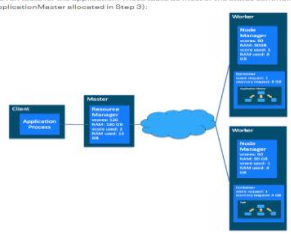
2. The ResourceManager makes a single container request on behalf of the application:

Step 3

3. The ApplicationMaster starts running within that container:



4. The ApplicationMaster requests subsequent containers from the ResourceManager that are allocated to run tasks for the application. Those tasks do most of the status communication with the ApplicationMaster allocated in Step 3):



Putting It Together

Figure 10: Merged MapReduce/YARN Application Running on a Cluster

In a MapReduce application, there are multiple map tasks, each running in a container on a worker host somewhere in the cluster. Similarly, there are multiple reduce tasks, also each running in a container on a worker host.

Simultaneously on the YARN side, the ResourceManager, NodeManager, and ApplicationMaster work together to manage the cluster's resources and ensure that the tasks, as well as the corresponding application, finish cleanly.

YARN Installation facts

In reality, there are two reasons why the full set of resources on a node cannot be allocated to YARN:

1. Non-Apache Hadoop services are also required to be running on a node (overhead).
2. Other Hadoop-related components require dedicated resources and cannot be shared with YARN (such as when running CDH).

The Intel-backed group develops CDH, a distribution of Hadoop that includes several other open-source projects, such as Impala and Search. It also offers security and integration features. The Impala framework is an interactive SQL query engine that allows direct queries of data stored in HDFS, Apache HBase, or AWS S3.

YARN in RM UI

As mentioned before, the ResourceManager has a snapshot of the available resources on the YARN cluster.

Example: Assume you have the following configuration on your 50 Worker nodes:

1. `yarn.nodemanager.resource.memory-mb = 90000`
2. `yarn.nodemanager.resource.vcores = 60`

Doing the math, your cluster totals should be:

1. memory: 50*90GB=4500GB=4.5TB
2. vcores: 50*60 vcores= 3000 vcores

On the ResourceManager Web UI page, the cluster metrics table shows the total memory and total vcores for the cluster, as seen in Figure 2 below.

Container Config

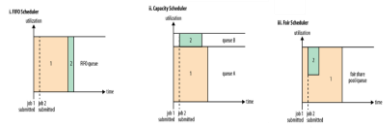
At this point, the YARN Cluster is properly set up in terms of Resources. YARN uses these resource limits for allocation, and enforces those limits on the cluster.

- YARN Container Memory Sizing
  - Minimum: `yarn.scheduler.minimum-allocation-mb`
  - Maximum: `yarn.scheduler.maximum-allocation-mb`
- YARN Container VCore Sizing
  - Minimum: `yarn.scheduler.minimum-allocation-vcores`
  - Maximum: `yarn.scheduler.maximum-allocation-vcores`
- YARN Container Allocation Size Increments
  - Memory Increment: `yarn.scheduler.increment-allocation-mb`
  - VCore Increment: `yarn.scheduler.increment-allocation-vcores`

Restrictions

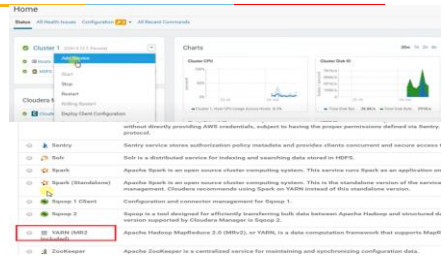
- Memory properties:
  - Minimum required value of 0 for `yarn.scheduler.minimum-allocation-mb`.
  - Any of the memory sizing properties must be less than or equal to `yarn.nodemanager.resource.memory-mb`.
  - Maximum value must be greater than or equal to the minimum value.
- VCore properties:
  - Minimum required value of 0 for `yarn.scheduler.minimum-allocation-vcores`.
  - Any of the vcore sizing properties must be less than or equal to `yarn.nodemanager.resource.vcores`.
  - Maximum value must be greater than or equal to the minimum value.
  - Recommended value of 1 for `yarn.scheduler.increment-allocation-vcores`. Higher values will likely be wasteful.

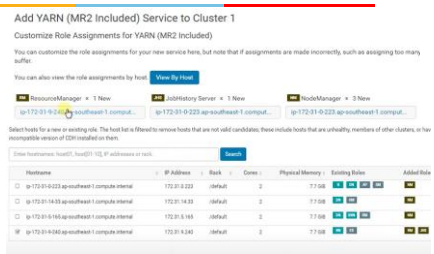
Scheduling YARN



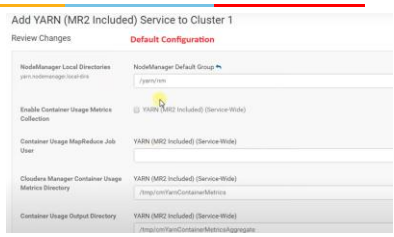
Jump To Next PPT

Add YARN Service

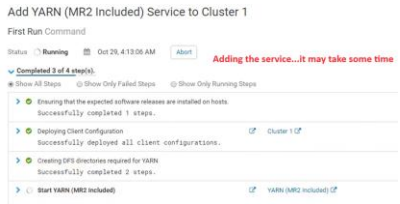




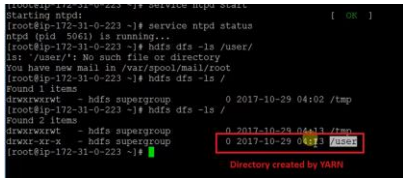
Default config



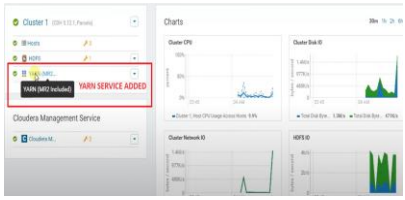
Adding Services

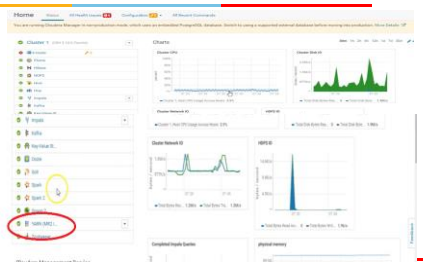


Contd..

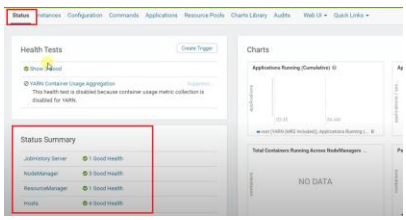


Service added Successful y

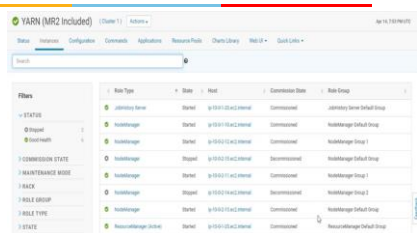




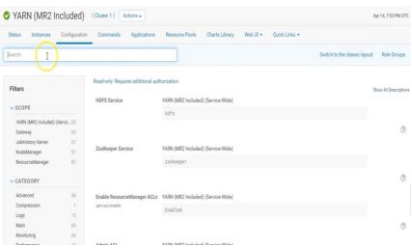
Status page



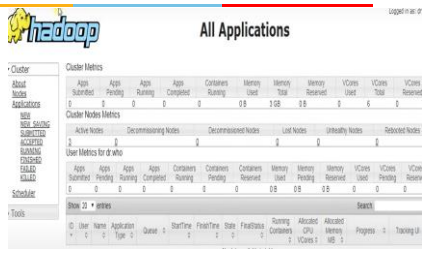
Instances/Roles



Configuration

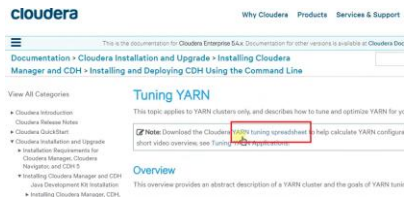


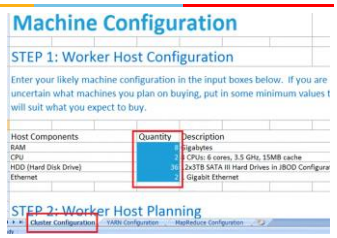
Resource Manager UI

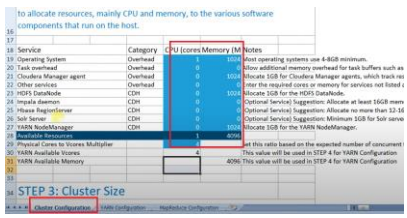


Job History UI

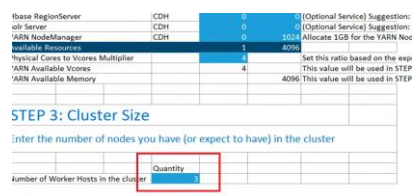
YARN Tunning



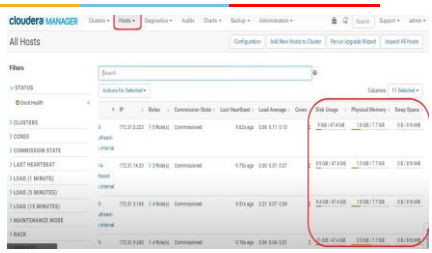




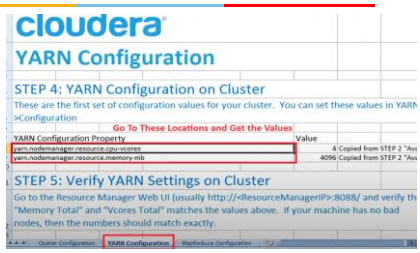
Details of worker nodes



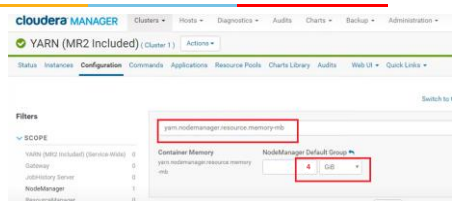
Make note of node statistics



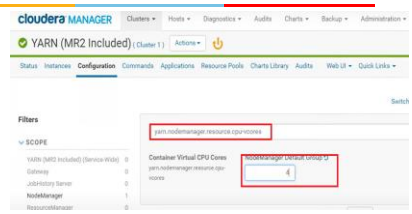
YARN Configuration



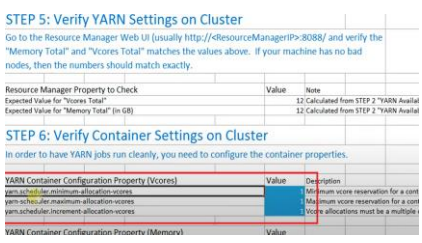
Memory Available update



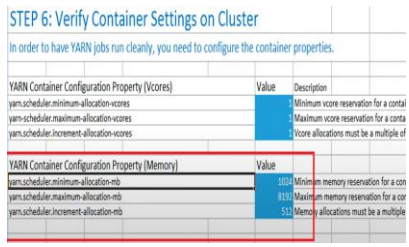
CPU Cores Available Update



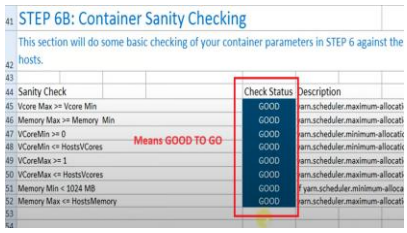
Few More Details



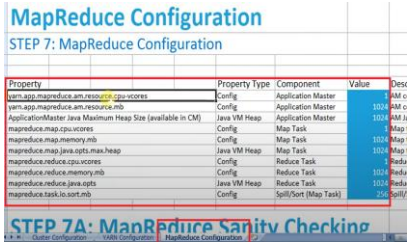
Scheduler Info

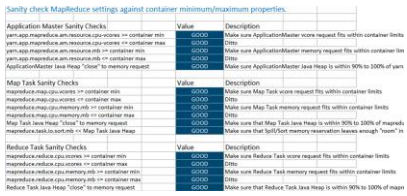


Check

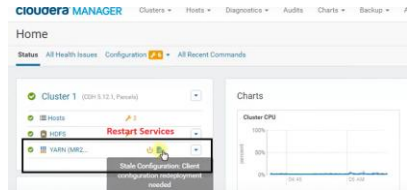


Map-Reduce Config

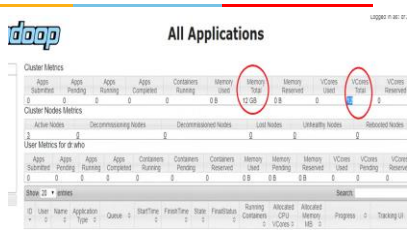




Restart Services



Verify

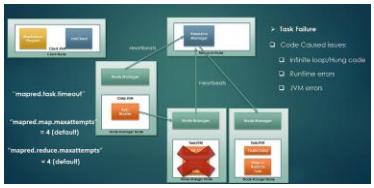


YARN Failure

YARN Failure Cases

- Task Failure
- Application Master Failure
- Node Manager Failure
- Resource Manager Failure

Task Failure



After Exhausting all attempts …Complete job will be marked as failed

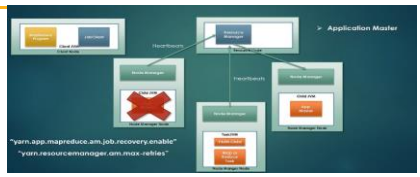
"mapred.map.failures.maxpercent"

"mapred.reduce.failures.maxpercent"

Failure of one or two tasks cannot be marked as complete Job Failure

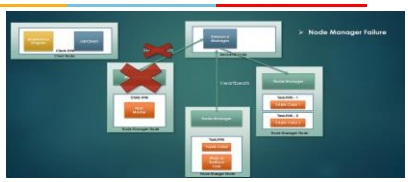
Above "mapred.map" and "mapred.reduce" properties are Used to Decide Expectable Percentage of Failure before deciding that Job has Failed
In case of task failure Resource Manager will start the Application Manager in a new container

App Master Failure



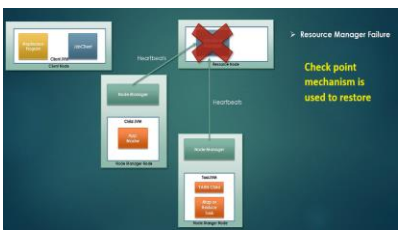
Task that has run under Application Master need not be resubmitted , it can be recovered. Provided property ..would be needed to be set to enable.
When Application master fail, Resource master stops getting heart beats from Application master. Number of attempts of Application Master is determined by the property "**yarn.resourcemanager.am.max-retries**"

Node Manager Failure



If Application Master was running under the failed Node Manager than steps described In Application Master are followed. All the remaining tasks are re spawned on new Node Manager. If task under specific Node Manager fail often and crosses the threshold then It is remode from available pool and is **Black Listed**

Resource Manager Failure



Hadoop Cluster Capacity Planning

WHAT IS HADOOP CLUSTER?
FACTORS DECIDING HADOOP CLUSTER
HARDWARE REQUIREMENTS
OPERATING SYSTEM REQUIREMENTS

What is a Hadoop cluster?

Factors influencing cluster size

Volume of Data

It is important for a Hadoop Admin to know about the volume of Data he needs to deal with and accordingly plan, organize, and set up the Hadoop Cluster with the appropriate **number of nodes** for an Efficient Data Management

Data Storage

The obtained data is encrypted and compressed using various **Data Encryption** and **Data Compression** algorithms so that the data security is achieved, and the space consumed to save the data is as minimal as possible.

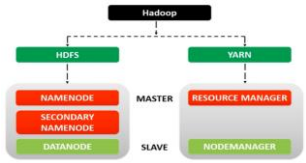
Data Retention

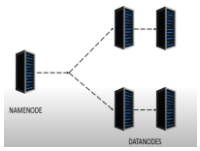
Data Retention is a process where the user gets to remove outdated, invalid, and unnecessary data from the Hadoop Storage to save space and improve cluster computation speeds.

Types of Work-Loads

This factor is purely **performance-oriented**. All this factor deals with is the performance of the cluster, the Workload on the processor can be classified into three types. Intensive, normal, and low

H/W Requirements



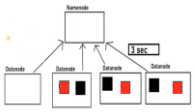


Name Node

- **Master daemon** manages the Data Nodes.
- Records the metadata of all the files
- Receives Heartbeat and a block report from Data Nodes.

Data Node

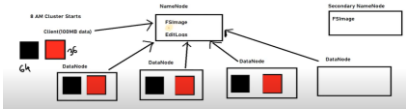
- **Slave daemons** runs on slave machine
- The actual data is stored on Data Nodes
- Responsible for serving read & write requests.



If NameNode fails entire HDFS file system is lost→**NameNode is the single point of failure in HDFS**

Secondary NN and Back Up NN

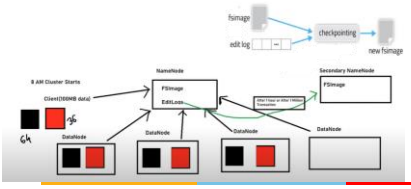
When Cluster Started , both FSImage , EditLogs has no value



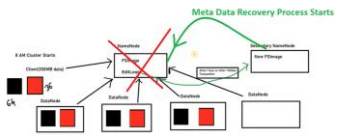
The metadata information related to the client request will be written down in the FSImage & EditLogs of NameNode

Checkpointing….Updates

After 1 Hour or after 1 M transaction whatever happens first



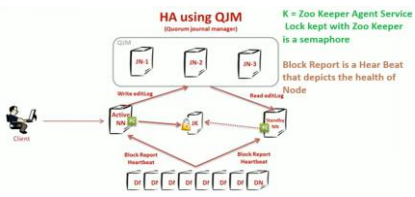
Safe Mode – Read Only Mode



- Safemode in Apache Hadoop is a maintenance state of NameNode, during which NameNode doesn't allow any modifications to the file system.
- In Safemode, HDFS cluster is in read-only and doesn't replicate or delete Data Blocks.

Back Up NN better Than Sec NN

Back Up Node



Thus …High Availability Hadoop Cluster Does not Need Secondary Name Node
Standby node also performs the Checkpointing operation.

H/W requirements for Name node and Job Tracker

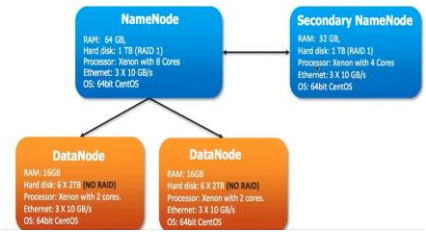
Component	Requirement
Operating System	1 Terabyte Harddisk Space
FS-Image	2 Terabyte Harddisk Space
Other Softwares(Zookeeper)	1 Terabyte Harddisk Space
Processor	Octa-Core Processor 2.5 GHz
RAM	128 GB
Intenet	10 GBPS

H/W req. for Data Node and Task trackers

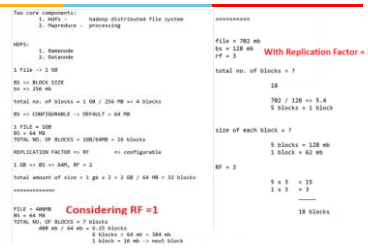
Number of Nodes	24 nodes(4 Terabytes each)
Processor	Octa-Core Processor 2.5 GHz
RAM	128 GB
Internet	10 GBPS

S/W Req. –Suggested OS

About Hadoop Cluster – A Typical Cluster Example



Few Facts about File Blocks



Observation

1 GB
BS = 256 MB
RF = 3

Large Data Set

TOTAL NO. OF BLOCKS => 12 BLOCKS
1GB / 256 MB = 4 X RF (3) = 12 BLOCKS

1 GB
1024 FILES = >
SIZE OF EACH FILE = 1MB

Small Data Set

BS = 256 MB
RF = 3

TOTAL NO. OF BLOCKS => 1024 X 3 = 3072 BLOCKS

If the Size of the File is less than the Size of the Block
Than size of the file becomes the size of the Block -- IMPORTANT

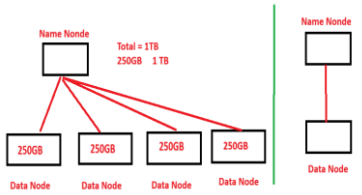
How Big Can Be the Size of My Cluster?

Everything in Hadoop can be viewed as an Object
(Hadoop is written in JAVA)
So Everything . File, Directory as well as Block is an Object
A File of 128 MB having Block Size of 64mb

So this file will split in to 2 Block + One More will be for file name. So 2 Objects + 1 Object = **3 Objects**
Every Object occupy 150 Bytes hence totally **150 X 3 = 450**
Bytes of memory will be occupies **IN THE NAME NODE**

If My Block Size is 128mb than total memory requirement Reduces to 2 Blocks ….So it is not advised to have smaller size files
So…Total Size of Memory in Name Node divide by File size required to be stored in Name node gives us count of total Number of Objects that Name node can accommodate

Name Node Size Matters



Whatever is the total capacity ..Number of Object that Name Node can accommodate is same

Sample Calculation

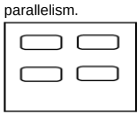
If Name Node memory = 64 GB
Lets consider some memory space for NN OS
Lets consider some memory space for NN Daemon

Remaining memory = may be 50 GB
Consider Block size = 64mb
Consider File size = 128mb
So total object size memory required on Name Node
Is 2 Block + 1 Block = 3 Block = 3 X 150 Bytes = 450 Bytes
Now total object that our NN can accommodate
= 50gb/450 bytes
So we can have these many Files in the cluster
If we divide 50gb/150 bytes ..we will get number of Objects

Name Node - Multithreaded Daemon

So we need to increase the HANDLER COUNT if the cluster is of bigger size
Max allowed = 60

So it is better to have multiple CPU CORE which will run multiple threads and achieve parallelism.



With 2 Threads So 4 Cores means 8 threads

Quad Core CPU Each having Virtualization as well as Hyper Threading Facility

Data Node Statistics

Default Mapper & Reducer Size

If Data Node had 4gb of Memory
(considering no memory used for OS & Daemon)

Mapper default size = 1gb
Reducer default size = 1.5gb

Hence we can accommodate 4 mappers
OR
1 mapper & 2 Reducers

IMPORTANT : Make sure that there is no swapping
Swapping of Processing (Mapper or Reducer) will reduce the Performance of the Data node

If we have 4gb of memory and 4 mappers even if Data node is capable
To accommodate the processing if CPU has only one CORE than only one Processing logic (One mapper at a time) can be executed.
Hence it is IMPORTANT to have Multi-Core CPU

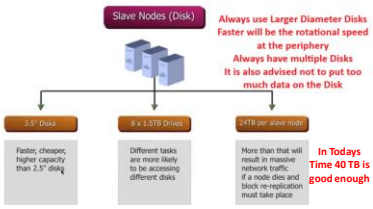
How Many Tasks Can I Run on My CPU?

This is give as = 1.5 X CORES

Example = If you have 2 core CPU than we can run 3 tasks
Per CPU since 1.5 X 2 = 3

Here Two tasks can run on 2 CPU and third can run once
Any one of the task from any one of the core completes

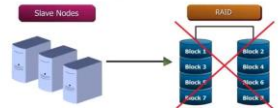
Too much is too bad



Say No to RAID on Data Nodes

Slave Nodes do not benefit from using RAID* storage

- HDFS provides redundancy by replicating blocks across multiple nodes
- RAID striping (RAID 0) is actually slower than the JBOD configuration used by HDFS
- RAID 0 read and write operations are limited by the speed of the slowest disk in the RAID array
- Disk operations on JBOD are independent, so the average speed is greater than that of the slowest disk

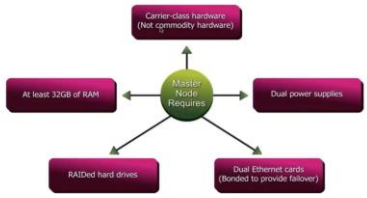


If we use RAID (RAID1 – MIRRORING) than we will be dividing memory by 6 since
1 x 3 + 1 x 3 = 6 Considering Replication =3
Speed of the RAID is decided by the Slowest Disk used.
Instead JBOD is suggested since with JBOD each Disk is free to run Independently
And so the total speed will be the average

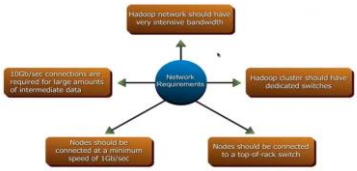
Just a Buch of Disk - JBOD

If we use RAID (RAID1 – MIRRORING) than we will be dividing memory by 6 since
1 x 3 + 1 x 3 = 6 Considering Replication =3
Speed of the RAID is decided by the Slowest Disk used.
Instead JBOD is suggested since with JBOD each Disk is free to run Independently
And so the total speed will be the average of all

Master Node H/W Reccomendatios

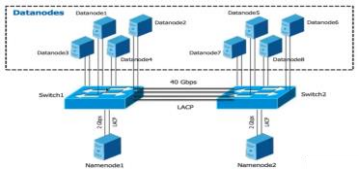


Network Considerations



Hadoop Use Jumbo Frames that uses 9000 bytes instead of MTU that support 1500 bytes

Advantages of Multiple NIC



Configure multiple NIC to behave as Bonded Interfaces.
Link Aggregation Control Protocol (LACP) gets triggered
If eth0 has 10 mbps and eth1 also has 10 mbps then
It will support Throughput of 20 mbps
If one of the NIC fail yjan system will continue working using remaining NIC

Planning Hadoop Cluster - Data Size

Hadoop Storage (HS) = CRS / (1-i)

- C= Compression Ratio
- R= Replication Factor
- S= Size of the data to be moved into Hadoop
- i= Intermediate Factor

Calculating Required Number of nodes

- Data Compression, hence, C is 1.
- The standard replication factor for Hadoop is 3.
- The Intermediate factor is 0.25,
- Then the calculation for Hadoop, in this case:

HS = (1*3*S) / (1-(1/4)
HS = 4S

Expected Result is 4 Times the Initial Storage

Calculating Required Number of nodes

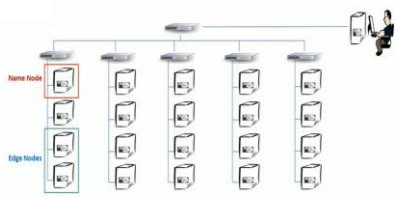
The expected Hadoop Storage instance, in this case, is 4 times the initial storage. The following formula can be used to estimate the number of data nodes.

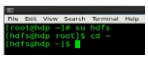
N = HS/D = (CRS/(1-i)) / D

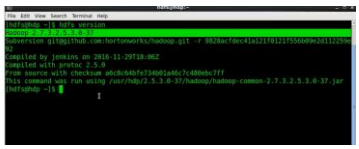
N = 5000/25 = 200

D = Available disk space of Each Node
Consider.. Each node to have 27 disks of 1 TB.
2 TB space goes to OS. Hence ...27-2 =25
Assuming Initial data to be 5000 TB
That is how we will need 200 Nodes

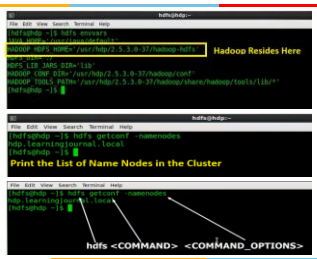
Edge node – For Day to Day Operations



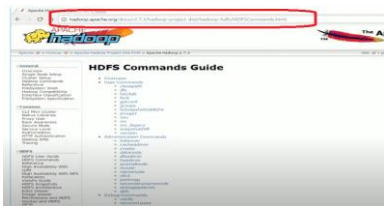




Hadoop Commands



List of Commands



Works Just Like LINUX

Commands to copy data

1. cp
2. copyFromLocal
3. Put
4. appendToFile

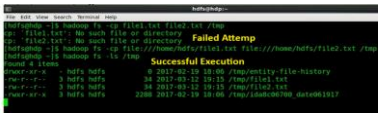
Snippet from documentation

cp

Usage: hadoop fs -cp [-f] [-p | -p[topax]] URI [URI ...] <dest>

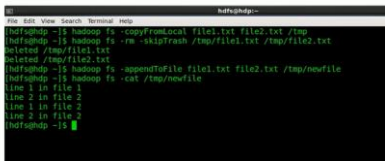
Copy files from source to destination. This command allows multiple sources as well in which case the destination must be a directory.

File Copy from



Copy data from One Directory to another or From Another File system to DHFS

Copy to Local and Append



Sizing parameters

1. Storage Considerations
2. Processing Considerations
3. Number of Data Nodes Required
4. Additional Resources

Storage Considerations

The number of data nodes you need is determined by the size of the data, how it will be analyzed, and the number of replicas you will have. By default, Apache Hadoop has 3 copies.
In this case, if we want to store X GB of data we need X*3 GB of storage for the forecasted period.

Processing Considerations

In addition to having enough space to store your data, you will need room for data processing, computing, and miscellaneous other tasks.

We can assume that, on an average day, only 10% of data is being processed, and a data process creates three times temporary data.

Therefore, you need to account for around 30% of your total storage as extra space.

Contd…

Number of Data Nodes Required

The final calculation for the number of data nodes required for your system will be dependent on your JBOD ("just a bunch of disks") capacity.

For example: Let's say that you need 500GB of space. If you have a JBOD of 12 disks, and each disk can store 6TB of data, then the data node capacity, or the maximum amount of data that each node can store, will be 72 TB. Data nodes can be added as the data grows, so to start with its better to select the lowest number of data nodes required.

In this case, the number of data nodes required to store 500GB of data equals 500/72, or approximately 7.

Note: Number of Data nodes* = (no. of disks*)

Summarizing

While setting up the cluster, we need to know the below parameters:

- What is the volume of data for which the cluster is being set? (For example, 100 TB.)
- The retention policy of the data. (For example, 2 years.)
- The kinds of workloads you have — CPU intensive, i.e. query; I/O intensive, i.e. ingestion, memory intensive, i.e. Spark processing. (For example, 30% jobs memory and CPU intensive, 70% I/O and medium CPU intensive.)
- The storage mechanism for the data — plain Text/AVRO/Parque/Jason/ORC/etc. or compresses GZIP, Snappy. (For example, 30% container storage 70% compressed.)

Estimating Data Node Requirement

Data Nodes Requirements

With the above parameters in hand, we can plan for commodity machines required for the cluster. (These might not be exactly what is required, but after installation, we can fine tune the environment by scaling up/down the cluster.) The nodes that will be required depends on data to be stored/analyzed.

By default, the Hadoop ecosystem creates three replicas of data. So if we go with a default value of 3, we need storage of *100TB *3=300 TB* for storing data of one year. We have a retention policy of two years, therefore, the storage required will be *1 year data* retention period=300*2=600 TB.*

Assume 30% of data is in container storage and 70% of data is in a Snappy compressed Parque format. From various studies, we found that Parquet Snappy compresses data to 70-80%.

We have taken it 70%. Here is the storage requirement calculation:

Total storage required for data =total storage % in container storage + total storage * %in compressed format*expected compression*

600.30+600*.70*(1-.70)=180+420*.30=180+420*.30=306 TB.*

Contd…

In addition to the data, we need space for processing/computation the data plus for some other tasks. We need to decide how much should go to the extra space. We also assume that on an average day, only 10% of data is being processed and a data process creates three times temporary data. So, we need around 30% of total storage as extra storage.

Hence, the total storage required for data and other activities is *306+306*.30=397.8 TB.*

As for the data node, JBOD is recommended. We need to allocate 20% of data storage to the JBOD file system. Therefore, the data storage requirement will go up by 20%. Now, the final figure we arrive at is *397.8(1+.20)=477.36 ~ 478 TB.*

Let's say *DS=478 TB*.

Additional considerations

Now, we need to calculate the number of data nodes required for 478 TB storage. Suppose we have a JBOD of 12 disks, each disk worth of 4 TB. Data node capacity will be 48 TB.

The number of required data nodes is *478/48 ~ 10*.

In general, the number of data nodes required is *Node= DS/(no. of disks in JBOD*diskspace per disk)*.

Note: We do not need to set up the whole cluster on the first day. We can scale up the cluster as data grows from small to big. We can start with 25% of total nodes to 100% as data grows.

Now, let's discuss data nodes for batch processing (Hive, MapReduce, Pig, etc.) and for in-memory processing.

As per our assumption, 70% of data needs to be processed in batch mode with Hive, MapReduce, etc.

10.70=7* nodes are assigned for batch processing and the other 3 nodes are for in-memory processing with Spark, Storm, etc.

CPU Cores and Tasks per Node

For batch processing, a *2*6*-core processor (hyper-threaded) was chosen, and for in-memory processing, a *2*8* cores processor was chosen. For batch processing nodes, while one core is counted for CPU-heavy processes, .7 core can be assumed for medium-CPU intensive processes. As we have assumption, 30% heavy processing jobs and 70% medium processing jobs, Batch processing nodes can handle [(no. of cores* %heavy processing jobs/cores required to process heavy job)+ (no. of cores* %medium processing jobs/cores required to process medium job)]. Therefore tasks performed by data nodes will be;

12.30/1+12*.70*/.7=3.6+12=15.6 ~15* tasks per node.

As hyperthreading is enabled, if the task includes two threads, we can assume 15*2~30 tasks per node.

RAM Requirements for Data Node

Now, let's calculate RAM required per data node. RAM requirements depend on the below parameters.

*RAM Required=DataNode process memory+DataNode TaskTracker memory+OS memory+CPU's core number *Memory per CPU core*

At the starting stage, we have allocated four GB memory for each parameter, which can be scaled up as required. Therefore, RAM required will be *RAM=4+4+4+12*4=60* GB RAM for batch data nodes and *RAM=4+4+4+16*4=76* GB for in-memory processing data nodes.

Cluster Calculations

Estimate the number of data nodes based on data size and increase pattern :

Estimating the hardware requirement is always challenging in Hadoop environment because we never know when data storage demand can increase for a business.
Understanding following factors in detail to conclude for the current scenario of adding right numbers to the cluster.

Scenario : If 8TB is the available disk space per node (10 disks with 1 TB, 2 disk for operating system/System Logs etc. were excluded.). **Assuming initial data size is 600 TB. How will you estimate the number of data nodes (n)?**

Parameters

Cluster Size : Hadoop

- The actual size of data to store – **600 TB**
- **Data trending analysis-prediction**: At what pace the data will increase in the future (per day/week/month/quarter/year)
- **Replication Factor** plays an important role – default 3x replicas.
- **Logs:** Hardware machine overhead (OS, logs etc.) – 2 disks were considered
- **Disk I/O:** Intermediate mapper and reducer data output on hard disk - 1x
- **Storage capacity**: Space utilization between 60 % to 70 %, disk utilization.
- **Compression ratio**

Cluster Size

Cluster Size : Hadoop

Let's do some calculation to find the number of data nodes required to store 600 TB of data:

Base calculation:

Data Size – 600 TB
Replication factor – 3
Intermediate data – 1
Total Storage requirement – (3+1) * 600 = 2400 TB
Available disk size for storage – 8 TB
Total number of required data nodes (approx.): 2400/8 = 300 machines
Actual Calculation: Base Calculation + Disk space utilization + Compression ratio

Actual Calculations

Cluster Size : Hadoop

Disk space utilization – 65 % (differ business to business)
Compression ratio – 2.3
Total Storage requirement – 2400/2.3 = 1043.5 TB
Available disk size for storage – 8*0.65 = 5.2 TB
Total number of required data nodes (approx.): 1043.5/5.2 = 201 machines
Actual usable cluster size (100 %): (201*8*2.3)/4 = 925 TB

Case: Business has predicted 20 % data increase in a quarter, and we need to predict the new machines to be added in a year

Contd..

Data increase – 20 % over a quarter
Additional data:
1st quarter: 1043.5 * 0.2 = 208.7 TB
2nd quarter: 1043.5 * 1.2 * 0.2 = 250.44 TB
3rd quarter: 1043.5 * (1.2)^2 * 0.2 = 300.5 TB
4th quarter: 1043.5 * (1.2)^3 * 0.2 = 360.6 TB

Available disk size for storage – 8*0.65 = 5.2 TB

Additional data nodes requirement (approx.):
1st quarter: 208.7/5.2 = **41 machines** — 8
2nd quarter: 250.44/5.2 = **49 machines** — 9
3rd quarter: 300.5/5.2 = **58 machines** — 12
4th quarter: 360.6/5.2 = **70 machines**
29
With these numbers we can predict next year additional machines requirement for the cluster :

(last quarter + 29) and so on.

PIG

- Entry of Apache Pig
- Pig vs MapReduce
- Twitter Case Study on Apache Pig
- Apache Pig Architecture
- Pig Components
- Pig Data Model & Operators
- Running Pig Commands and Pig Scripts (Log Analysis)

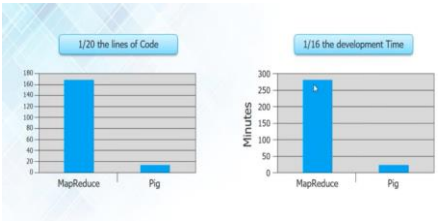
No Need to be a Programmer

Developed by YAHOO

- An open-source high-level dataflow system
- Introduced by Yahoo
- Provides abstraction over MapReduce
- Two main components – the *Pig Latin* language and the *Pig Execution*

Fun Fact:
✓ ***10 lines of pig latin= approx. 200 lines of Map-Reduce Java Program***

Clear Winner



Pig v/s MapReduce

- High-level data flow tool
- No need to write complex programs
- Built-in support for data operations like joins, filters, ordering, sorting etc.
- Provides nested data types like tuples, bags, and maps

- Low-level data processing paradigm
- You need write programs in Java/Python etc.
- Performing data operations in MapReduce is a humongous task
- Nested data types are not there in MapReduce

Pig Specific

- Can take any data
  - Structured data
  - Semi-Structured data
  - Unstructured data
- Easy to learn, Easy to write and Easy to read
  - Data Flow Language
  - Reads like a series of steps
- Extensible by UDF (User Defined Functions)
  - Java
  - Python
  - JavaScript
  - Ruby

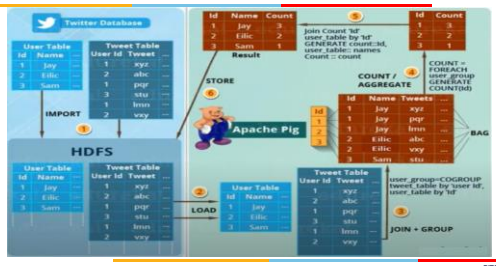
- Provides common data operations filters, joins, ordering, etc. and nested data types tuples, bags, and maps missing from MapReduce.
- An ad-hoc way of creating and executing map-reduce jobs on very large data sets
- Open source and actively supported by a community of developers.

Pig – Hello World

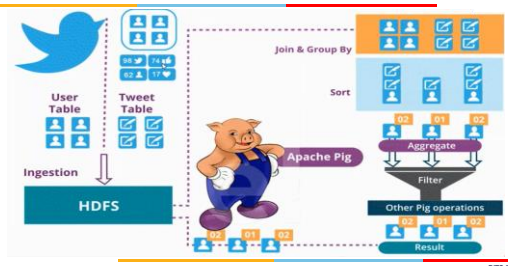
- Twitter's data was growing at an accelerating rate (i.e. 10 TB/day).
- Thus, Twitter decided to move the archived data to HDFS and adopt Hadoop for extracting the business values out of it.
- Their major aim was to analyse data stored in Hadoop to come up with the multiple insights on a daily, weekly or monthly basis.

Let me talk about one of the insight they wanted to know.
Analyzing how many tweets are stored per user, in the given tweet tables?

Working



Overview



Architecture

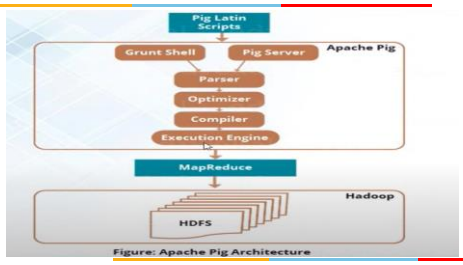
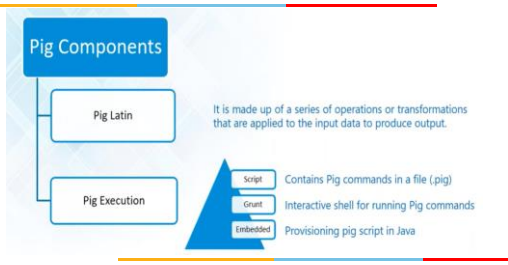


Figure: Apache Pig Architecture

Components



Modes of Execution

You can run Apache Pig in 2 modes:

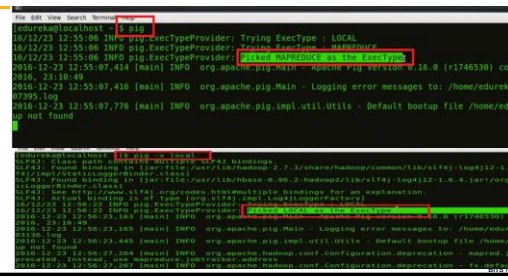
MapReduce Mode – This is the default mode, which requires access to a Hadoop cluster and HDFS installation. The input and output in this mode are present on HDFS.

Command: pig

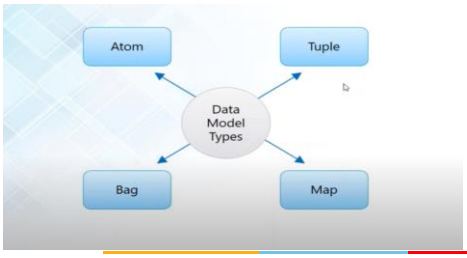
Local Mode – With access to a single machine, all files are installed and run using a local host and file system. Here the local mode is specified using '-x flag' (pig -x local). The input and output in this mode are present on local file system.

Command: pig –x local

MapReduce & Local Mode



Data Models



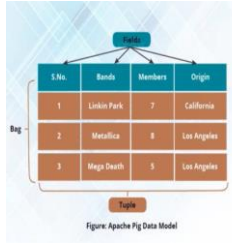


Figure: Apache Pig Data Model

- Tuple is an ordered set of fields which may contain different data types for each field.

  Example of tuple – (1, Linkin Park, 7, California)

- A Bag is a collection of a set of tuples and these tuples are subset of rows or entire rows of a table.

  Example of a bag – {(Linkin Park, 7, California), (Metallica, 8), (Mega Death, Los Angeles)}

Map & Atom

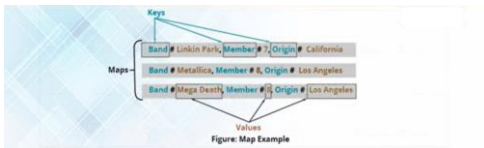


Figure: Map Example

- A Map is key-value pairs used to represent data elements.

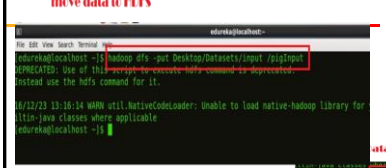
  Example of maps– [band#Linkin Park, members#7], [band#Metallica, members#8]

- Atoms are basic data types which are used in all the languages like string, int, float, long, double, char[], byte[]

Operators for DB Operatrions

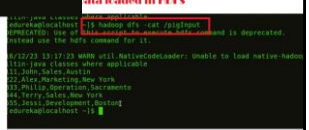
Operator	Description
LOAD	Load data from the local file system or HDFS storage into Pig
FOREACH	Generates data transformations based on columns of data
FILTER	Selects tuples from a relation based on a condition
JOIN	Join the relations based on the column
ORDER BY	Sort a relation based on one or more fields
STORE	Save results to the local file system or HDFS
DISTINCT	Removes duplicate tuples in a relation
GROUP	Groups together the tuples with the same group key (key field)
COGROUP	It is same as GROUP. But COGROUP is used when multiple relations re involved

move data to HDFS

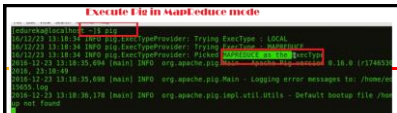


Sample Data

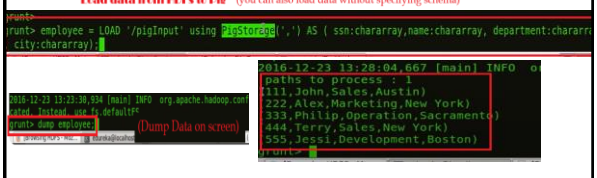
ata loaded in HDFS



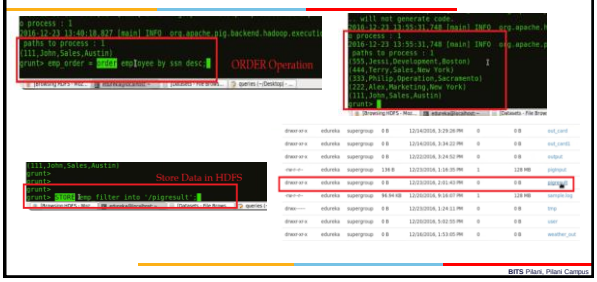
Execute Pig in MapReduce mode



Load data from HDFS to Pig (you can also load data without specifying schema)



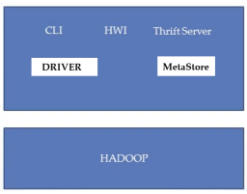
Use of "foreach"



HIVE

- Hive Architecture
- Hive Components
- Metastore
- Limitations of Hive
- Where to use Hive
- Differences with traditional RDBMS
- Type System
- Hive Data Model

Architecture



Components

Shell - Environment
MetaStore - Metadata
Driver- Management of Life Cycle of Hql Statements
Compiler- Convert Hql to MR
Execution Engine- Execution of Job

Metastore

- Embedded metastore → Driver → MS→ Derby
- Local Metastore: Driver → MS, Driver → MS → MySql
- Remote Metastore: Driver → MS, Driver → MS → MySql

Limitations

- Not recommended for row level updates .
- Latency for Hive Queries Is High
- Not designed for OLTP

Difference w.r.t RDBMS

Schema on Read V/s Schema on Write

Data Types

- Boolean :- true/false
- Integers :- TinyInt - 1 byte integer
  SmallInt - 2 byte integer
  INT - 4 byte integer
  BigInt - 8 byte integer
- Floating point :- Float / Double
- String

Data Types

NUMERIC TYPES	DESCRIPTION
TINYINT	1-byte signed integer, from -128 to 127
SMALLINT	2-byte signed integer, from -32,768 to 32,767
INT/INTEGER	4-byte signed integer, from -2,147,483,648 to 2,147,483,647
BIGINT	8-byte signed integer, from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,80
FLOAT	4-byte single precision floating point number
DOUBLE	8-byte double precision floating point number
DOUBLE PRECISION	Alias for DOUBLE, only available starting with Hive 2.2.0
DECIMAL	It accepts a precision of 38 digits.
NUMERIC	Same as DECIMAL type.

DATE/TIME TYPES	DESCRIPTION
TIMESTAMP	Accepts Both Date and Time
DATE	Accepts just Date
INTERVAL	Interval

Data Types

STRING TYPES	DESCRIPTION
STRING	The string is an unbounded type. Not required to specify the lenght. It can accept max up to 32,767 bytes.
VARCHAR	Variable length of characters. It is bounded meaning you still need to specify the length like VARCHAR(10).
CHAR	Fixed length of Characters. if you define char(10) and assigning 5 chars, the remaining 5 characters space will be wasted.

Complex Data Types

- ❑ Struts
- ❑ Maps
- ❑ Array

ACID -limitations

- To support ACID, Hive tables should be created with **TRANSACTIONAL** table property.
- Currently, Hive supports ACID transactions on tables that store **ORC** file format.
- Enable ACID support by setting transaction manager to **DbTxnManager**
- Transaction tables cannot be accessed from the non-ACID Transaction Manager (**DummyTxnManager**) session.
- **External tables** cannot be created to support ACID since the changes on external tables are beyond Hive control.
- **LOAD** is not supported on ACID transactional Tables. hence use **INSERT INTO**.
- On the Transactional session, all operations are auto commit as **BEGIN**, **COMMIT**, and **ROLLBACK** are not yet supported.

Data Models

- Database
- Table
- Partition
- Buckets & Clusters

Partitioning - Static

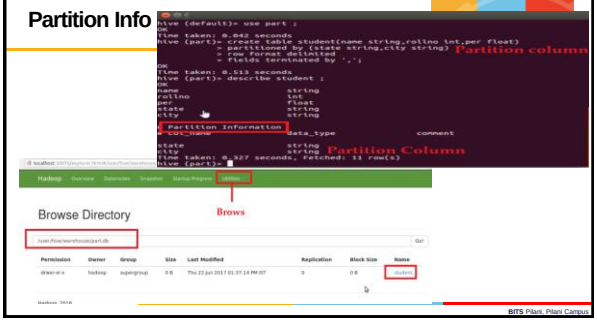
```
hive> create database part ;
OK
Time taken: 0.439 seconds
hive> set hive.cli.print.current.db=true;
hive (default)> use part ;
OK
Time taken: 0.842 seconds
hive (part)> create table student(name string,rollno int,per float)
```

Create database

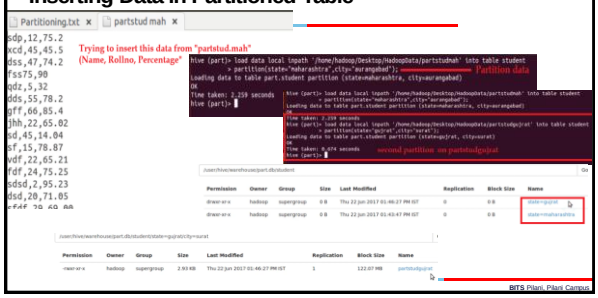
Use database already created

Create table in database

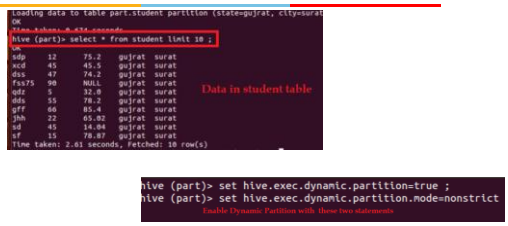
Partition Info



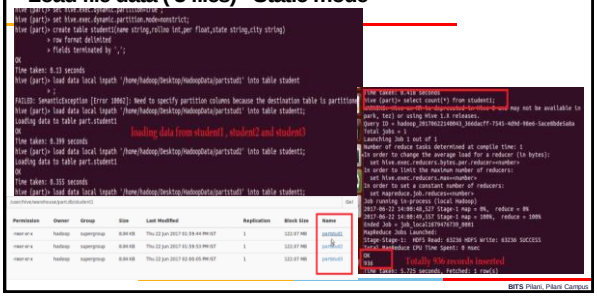
Inserting Data in Partitioned Table



Partitioning Dynamic



Load file data (3 files) –Static mode

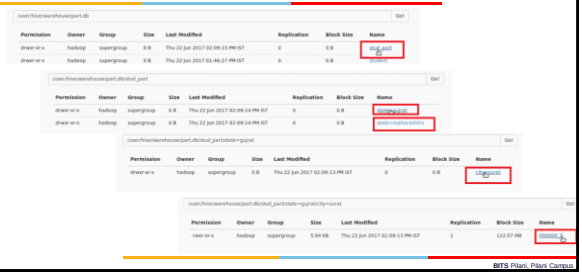


Insert data from table to partitioned table

```
hive (part)> insert into stud_part
           > partition(state,city)
           > select name,rollno,per
           > ,state,
           > city
           > from student1;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be avail
park, tez) or using Hive 1.X releases.
Query ID = hadoop_20170622140910_9f43aa75-98c0-448b-84b3-f36b6c0
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce opera
Job running in-process (local Hadoop)
2017-06-22 14:09:13,096 Stage-1 map = 0%,  reduce = 0%
2017-06-22 14:09:14,111 Stage-1 map = 100%,  reduce = 0%
```

Insert data from student1 to "stud_part" table
having partition on state and city

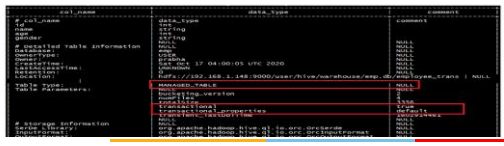
Data Presentation in dynamic mode



Transactional table

```
CREATE TABLE emp.employee_trans (
  id int,
  name string,
  age int,
  gender string)
  STORED AS ORC
  TBLPROPERTIES ('transactional'='true');
```

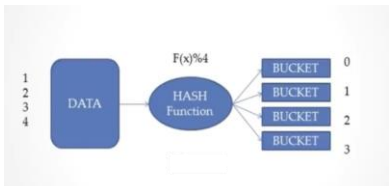
You can run the DESCRIBE FORMATTED emp.employee to check if the table is created with the transactional data_type as TRUE.

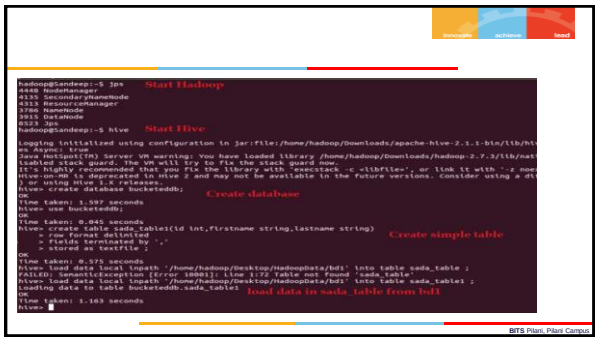


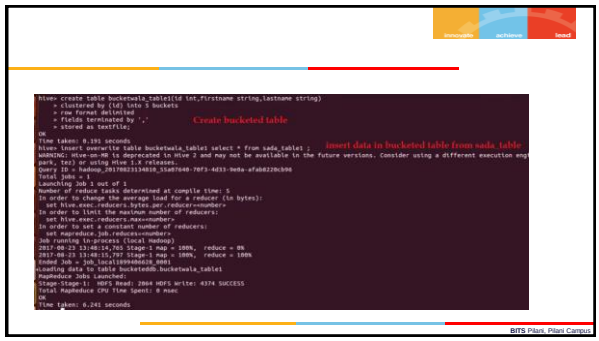
Bucketing

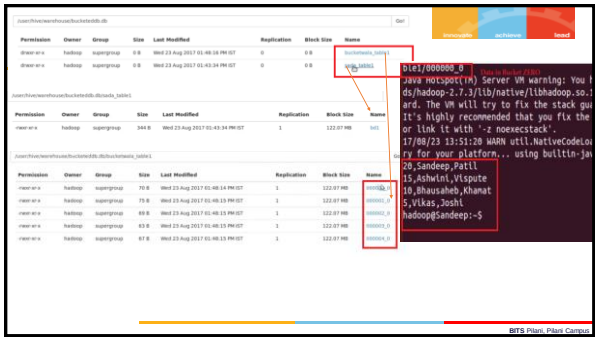
- Data is divided into buckets
- Used for Data Sampling
- Helpful for implementing Map-side Join

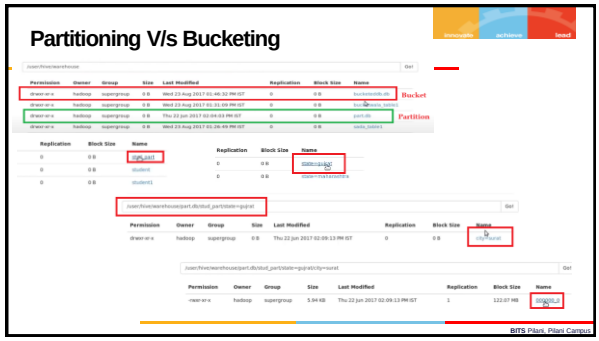
Hashing Technique











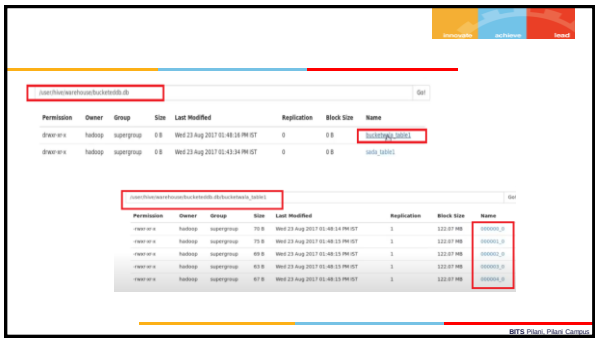


Table Types

- Internal Table
- External Table

Internal/Managed & External Table

Internal tables are also known as Managed tables that are owned and managed by Hive. By default, Hive creates a table as an Internal table and owned the table structure and the files.

In other words, Hive completely manages the lifecycle of the table (metadata & data) similar to tables in RDBMS.

For Internal tables, Hive by default stores the files at the data warehouse location which is located at `/user/hive/warehouse`

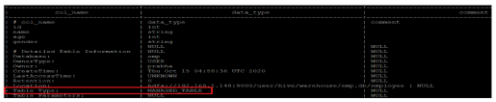
When you drop an internal table, it drops the data and also drops the metadata of the table.

Below is an example of creating internal table.

Internal / Managed Table

```
CREATE TABLE IF NOT EXISTS emp.employee (
 id int,
 name string,
 age int,
 gender string )
 COMMENT 'Employee Table'
 ROW FORMAT DELIMITED
 FIELDS TERMINATED BY ',';
```

Use `DESCRIBE FORMATTED emp.employee;` to get the description of the table and you should see Table Type as `MANAGED_TABLE`.



External Table

Data in External tables are not owned or managed by Hive. To create an External table you need to use `EXTERNAL` clause.

Hive default stores external table files also at Hive managed data warehouse location but recommends to use external location using `LOCATION` clause.

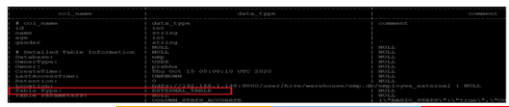
Dropping an external table just drops the metadata but not the actual data. The actual data is still accessible outside of Hive.

Below is an example of creating an external table in Hive. If you noticed we use `EXTERNAL` and `LOCATION` options.

External Table

```
CREATE EXTERNAL TABLE emp.employee_external (
 id int,
 name string,
 age int,
 gender string)
 ROW FORMAT DELIMITED
 FIELDS TERMINATED BY ','
 LOCATION '/user/hive/data/employee_external';
```

Use `DESCRIBE FORMATTED emp.employee_external;` to get the description of the table and you should see Table Type as `EXTERNAL_TABLE`.



Dropping table

Regardless of the Internal and external table, Hive manages the table definition and its partition information in Hive Metastore. Dropping an internal table deletes the table metadata from Metastore and also removes all its data/files from HDFS.

Dropping an external table, just drop the metadata of the table from Metastore and keeps the actual data as-is om HDFS location.

Key differences -- Summary

INTERNAL OR MANAGED TABLE	EXTERNAL TABLE
By default, Hive creates an Internal or Managed Table.	Use EXTERNAL option/clause to create an external table
Hive owns the metadata, table data by managing the lifecycle of the table	Hive manages the table metadata but not the underlying file.
Dropping an Internal table drops metadata from Hive Metastore and files from HDFS	Dropping an external table drops just metadata from Metastore with out touching actual file on HDFS.
Hive supports ARCHIVE, UNARCHIVE, TRUNCATE, MERGE, CONCATENATE operations	Not supported
Supports ACID/Transactional	Not supported
Supports result caching	Not supported

Hadoop…Hbase….Hive….Pig

Hadoop:

- Hadoop is basically 2 things: a **Distributed FileSystem (HDFS) + Computation or Processing framework (MapReduce).**
- Like all other FS, HDFS also provides us storage, but in a fault tolerant manner with high throughput and lower risk of data loss (because of the replication). But, being a FS, HDFS lacks **random read and write access**.
- This is where HBase comes into picture. It's a **distributed, scalable, big data store**, modelled after Google's BigTable. It stores data as key/value pairs.

HBase:

- Apache HBase is an open source NoSQL database that provides real-time read/write access to those large datasets.
- HBase scales linearly to handle huge data sets with billions of rows and millions of columns, and it easily combines data sources that use a wide variety of different structures and schemas.
- HBase is natively integrated with Hadoop and works seamlessly alongside other data access engines through YARN.

Hive:

- It provides us data warehousing facilities on top of an existing Hadoop cluster. Along with that it provides an SQL like interface which makes work easier.
- While Pig is basically a **dataflow language** that allows us to process enormous amounts of data very easily and quickly.
- Pig basically has 2 parts: the Pig **Interpreter** and the language, **PigLatin**. We can write Pig script in PigLatin and using Pig interpreter process them.
- Pig makes our life a lot easier, otherwise writing MapReduce is always not easy.

Pig V/s Hive

S.No.	Pig	Hive
1.	Pig operates on the client side of a cluster.	Hive operates on the server side of a cluster.
2.	Pig uses pig-latin language.	Hive uses HiveQL language.
3.	Pig is a Procedural Data Flow Language.	Hive is a Declarative SQLish Language.
4.	It was developed by Yahoo.	It was developed by Facebook.
5.	It is used by Researchers and Programmers.	It is mainly used by Data Analysts.
6.	It is used to handle structured and semi-structured data.	It is mainly used to handle structured data.
7.	It is used for programming.	It is used for creating reports.
8.	Pig scripts end with .pig extension.	In Hive, all extensions are supported.
9.	It does not support partitioning.	It supports partitioning.
10.	It loads data quickly.	It loads data slowly.
11.	It does not support JDBC.	It supports JDBC.
12.	It does not support ODBC.	It supports ODBC.
13.	Pig does not have a dedicated metadata database.	Hive makes use of the exact variation of dedicated SQL-DDL language by defining tables beforehand.

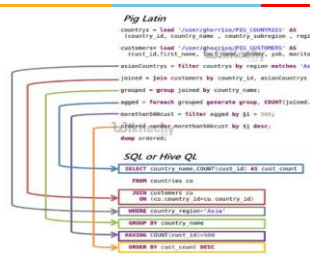
Hive V/s HBase

S. No.	Parameters	Hive	HBase
1.	Basics	Hive is a query engine that uses queries that are mostly similar to SQL queries.	It is Data storage, particularly for unstructured data.
2.	Used for	It is mainly used for batch processing (that means OLAP-based).	It is extensively used for transactional processing (that means OLTP).
3.	Processing	It cannot be used for real-time processing since immediate analysis results are unable to obtain. In other words, the operations in Hive require batch processing, they normally take a long time to complete.	It can be used to process data in real-time. Transactional operations are faster than non-transactional operations (since HBase stores data in the form of key-value pairs).
4.	Queries	It is used only for analytical queries. It is mostly used to analyze Big Data.	It is used for real-time querying. It is mostly used to query Big Data.
5.	Runs on	Hive runs on the top of Hadoop.	HBase runs on the top of HDFS (Hadoop Distributed File System).
6.	Database	Apache Hive is not a database.	It supports the NoSQL database.
7.	Schema	It has a schema model.	It is free from the schema model.

Contd…

8.	Latency	Made for high latency operations as batch processing takes time.	Made for low-level latency operations.
9.	Cost	It is expensive as compared to HBase.	It is cost-effective as compared to Hive.
10.	Query Language	Hive uses HQL (Hive Query Language).	To conduct CRUD (Create, Read, Update, and Delete) activities, HBase does not have a specialized query language. HBase includes a Ruby-based shell where you can use Get, Put, and Scan functions to edit your data.
11.	Level of Consistency	Eventual consistency	Immediate consistency
12.	Secondary Indexes	It does not support Secondary Indexes.	It supports Secondary Indexes.
13.	Example	Hubspot	Facebook

Clear Difference



Flume

1. What is Flume?
   1. Overview
   2. Architecture
2. Down the rabbit hole
   1. Events
   2. Sources
   3. Channels
   4. Sinks
   5. Configuration
   6. Data manipulation with interceptors
   7. Multiplexing
   8. Customer Serializers
3. Use Cases

What is Flume?

Flume is a data collection and aggregation framework, which operates in distributed topologies.

It is stream - oriented, fault tolerant and offers linear scalability.

It offers low latency with high throughput

Its configuration is declarative, yet allows easy extensibility

It is quite mature – it has been around for quite a while, enjoys support from multiple vendors with thousands of large-scale deployments.

What is Flume?

Flume defines a simple pipeline structure with three roles:

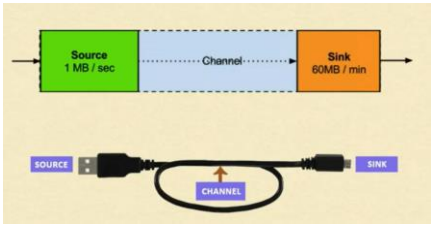
- Source
- Channel
- Sinks

Sources define where data comes from, e.g. a file, a message Queue (Kafka, JMS)

Channels are pipes connecting Sources with Sinks

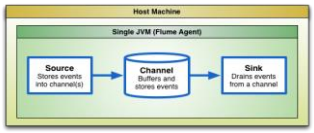
Sinks are the destination of the data pipelined from *Sources*

Channel …is like buffer

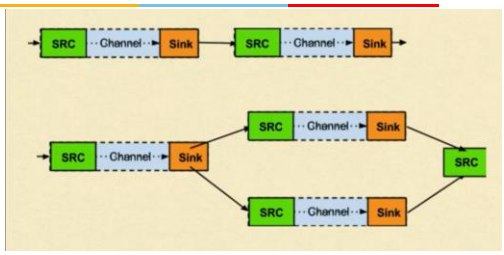


What is Flume?

These three roles run within a JVM process called *Agents*. Data is packed into *Events*, which are a simple Avro wrapper around any type of record, typically a line from a log file.



Transfer to Single or Multiple source



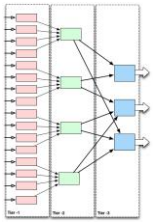
Supports a large variety of sources Including:

- tail (like unix tail -f),
- syslog,
- log4j - allowing java applications to write logs to HDFS via flume

Flume nodes can be arranged in arbitrary topologies.
Typically there is a node running on each source machine
With tiers of aggregating nodes that the data flows through on its way to HDFS.

Delivery reliability:
best-effort delivery - doesn't tolerate any node failures
end-to-end - which guarantees delivery in node failures

Contd..



Agents can be chained into a tree structure to allow multiple collection from various sources, concentrated and piped to one destination.

Shown here:

- Many agents (1 per host) at edge tier
- 4 "collector" agents in that data center
- 3 centralized agents writing to HDFS

Sources

Event-driven or polling-based
Most sources can accept batches of events
Stock source implementations:

- Spool Directory
- HTTP
- Kafka
- JMS
- Netcat
- AvroRPC

AvroRPC is use by Flume to communicate between Agents

Channels

Passive "Glue" between Sources and Sinks
Channel type determines the reliability guarantees
Stock channel types:

- JDBC – has performance issues
- Memory – lower latency for small writes, but not durable
- File – provides durability; most people use this
- New: Kafka – topics operate as channels, durable and fast

Sinks

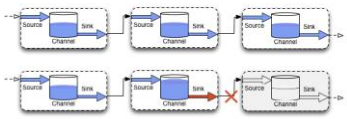
All sinks are polling-based
Most sinks can process batches of events at a time
Stock sink implementations:

- Kafka
- HDFS
- HBase (2 variants – Sinc and Async)
- Solr
- ElasticSearch
- Avro-RPC, Thrift-RPC – Flume agent inter-connect
- File Roller – write to local filesystem
- Null, Logger– for testing purposes

Pipeline

Source: Puts events into the local channel
Channel: Store events until someone takes them
Sink: Takes events from the local channel

- On failure, sinks backoff and retry forever until success



Flume configuration…for whole cluster

Read from Port (web) Push to HDFS

```
# Name the components on this agent
a1.sources = r1
a1.sinks = s1,s2
a1.channels = c1

a2.sources = r1
a2.sinks = s1,s2
a2.channels = c1

# Describe/configure the source
a1.sources.r1.type = netcat
a1.sources.r1.bind = localhost
a1.sources.r1.port = 44444

# Describe the sink
#a1.sinks.k1.type = logger
a1.sinks.s1.type = hdfs
a1.sinks.s1.hdfs.path = hdfs://a.cloudxlab.com/user/student/sgiri/flume/webdata

a1.sinks.s2.type = jdbc
a1.sinks.s2.connection.path = hdfs://a.cloudxlab.com/user/student/sgiri/flume/webdata1
```

a1 Agent

a2 Agent

Config File

```
a1.sources = r1
a1.sinks = hdfs-Cluster1-sink
a1.channels = c1

# Describe/configure the source
a1.sources.r1.type = netcat
a1.sources.r1.bind = localhost
a1.sources.r1.port = 44444

# Describe the sink
#a1.sinks.k1.type = logger
a1.sinks.hdfs-Cluster1-sink.type = hdfs
a1.sinks.hdfs-Cluster1-sink.hdfs.path = hdfs:///user/sandeepgiri9034/flume-webdata

# Use a channel which buffers events in memory
a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel
a1.sources.r1.channels = c1
#a1.sinks.k1.channel = c1
a1.sinks.hdfs-Cluster1-sink.channel = c1
```


Configuration

Declarative configuration with property files

- Define your source
- Define the sink
- Declare the channel for source and sink

```
####### Spool Config

SpoolAgent.sources = MySpooler
SpoolAgent.channels = MemChannel
SpoolAgent.sinks = HDFS

SpoolAgent.channels.MemChannel.type = memory
SpoolAgent.channels.MemChannel.capacity = 500
SpoolAgent.channels.MemChannel.transactionCapacity = 200

SpoolAgent.sources.MySpooler.channels = MemChannel
SpoolAgent.sources.MySpooler.type = spooldir
SpoolAgent.sources.MySpooler.spoolDir = /var/log/
SpoolAgent.sources.MySpooler.fileHeader = false

SpoolAgent.sinks.HDFS.channel = MemChannel
SpoolAgent.sinks.HDFS.type = hdfs
SpoolAgent.sinks.HDFS.hdfs.path = hdfs://cluster/data/spool/
SpoolAgent.sinks.HDFS.hdfs.fileType = DataStream
SpoolAgent.sinks.HDFS.hdfs.writeFormat = Text
SpoolAgent.sinks.HDFS.hdfs.batchSize = 100
SpoolAgent.sinks.HDFS.hdfs.rollSize = 0
SpoolAgent.sinks.HDFS.hdfs.rollCount = 0
SpoolAgent.sinks.HDFS.hdfs.rollInterval = 300
```

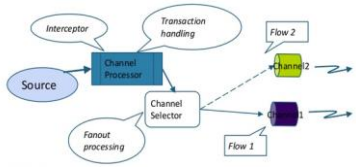

Interceptors

Incoming data can be transformed and enriched at the source
One or more interceptors can modify your events and set headers
These headers can be used for sorting within sinks and routing to different sinks
Stock Interceptor Implementations

- Timestamp (adds a timestamp header)
- Hostname (adds the flume host as a header)
- Regex (extracts the applied regex to a header)
- Static (sets a static constant header)

Selectors

Multiplexing selectors allows header sensitive dispatch to specific destination sinks.
Replicating selectors allow simultaneous writes to multiple sinks.

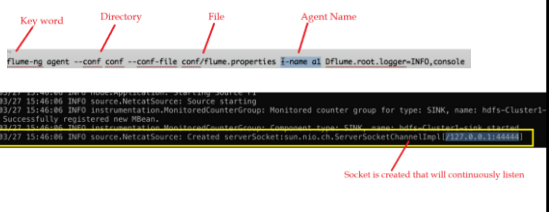


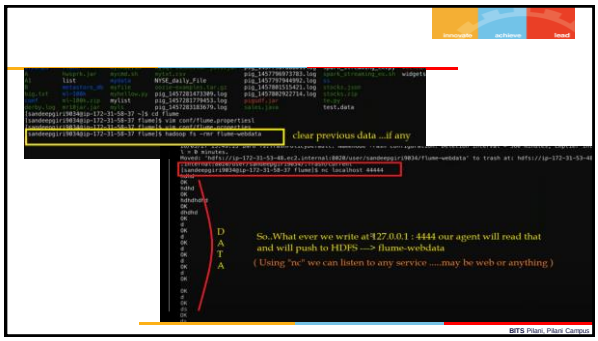
Serializers

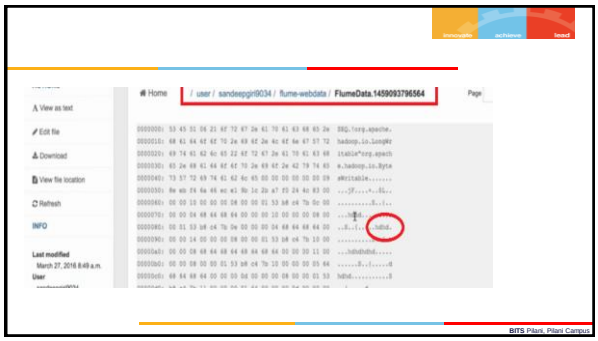
Serializers customize the format written to sink.
Built-in serializers are generally fine for text
More complicated use case such as Avro require customer serializers
HDFS, HBase and FileRoller Sinks support customer Serializers.
Custom Serializers implement specific interfaces / extend abstract classes:

- AsyncHbaseEventSerializer (HBase)
- AbstractAvroEventSerializer (HDFS)
- EventSerializer (Generic)

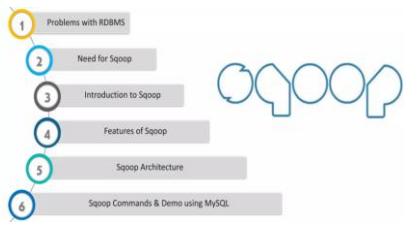
Some practical



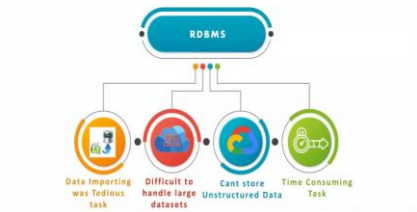




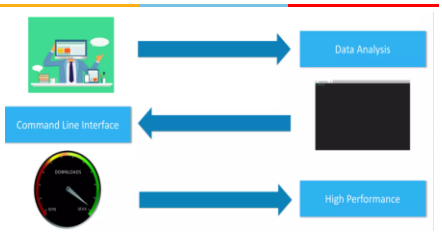
Sqoop



Problem with RDBMS



Why Sqoop?



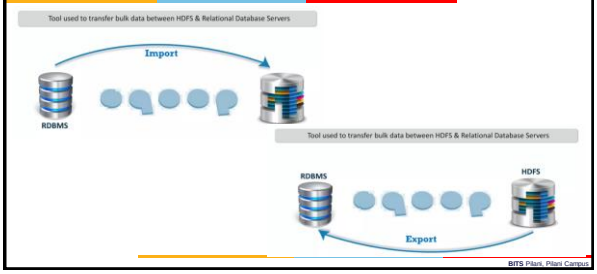
Why Sqoop cont..

- SQL Servers are already deployed opulent worldwide
- Nightly processing is done on SQL servers for years
- As Hadoop making ways into enterprise, there was a need to move certain part of data from traditional SQL DB (RD) to Hadoop
- Transferring data using scripts is inefficient and time consuming.
- Traditional DB already have reporting, data visualization etc. applications built in enterprise
  - Bringing processed data from Hadoop to those application is the need

Why Sqoop cont..

- From RDB to Hadoop
  - Users must consider details like ensuring consistency of data, the consumption of production system resources, data preparation for provisioning downstream pipeline.
- Hadoop to RDB
  - Directly accessing data residing on external systems from within the map reduce applications complicates applications and exposes the production system to the risk of excessive load originating from cluster nodes.

What is Sqoop?



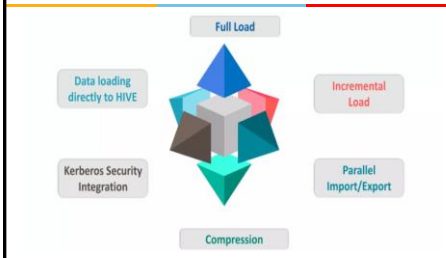
What Sqoop Provides?

- Sqoop allows easy import and export of data from structured data stores
  - RD, Enterprise data warehouses, and NoSQL systems
- Provision data from external system on to HDFS
  - Once data is moved populate tables in Hive and HBase.
- Sqoop integrates with Oozie, allowing you to schedule and automate import and export tasks.
- Sqoop uses a connector based architecture which supports plugins that provide connectivity to new external systems.

What Sqoop Provide?

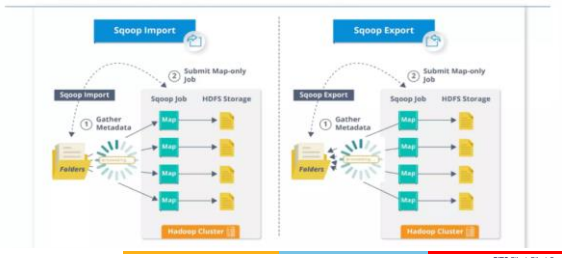
- Sqoop runs in Hadoop Cluster
- Sqoop has access to Hadoop Core
  - Sqoop use Mappers to slice the incoming
  - Data is placed to HDFS
- Sqoop Import
  - From RD/NoSql DB to Hadoop
- Export
  - From Hadoop to RD/NoSql DB

Features



Architecture

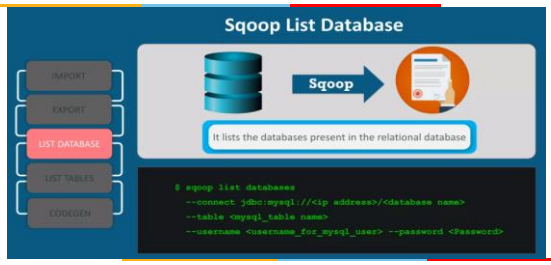
Sqoop Import & Export



Cont.…

Cont.…

List



List Cont.…

Import All Tables

```
$ sqoop import-all-tables (generic-args) (import-args)
$ sqoop-import-all-tables (generic-args) (import-args)
```

Example

Let us take an example of importing all tables from the **userdb** database. The list of tables that the database **userdb** contains is as follows.

```
+--------------------+
|    Tables          |
+--------------------+
|    emp             |
|    emp_add         |
|    emp_contact     |
+--------------------+
```

The following command is used to import all the tables from the **userdb** database.

```
$ sqoop import-all-tables \
--connect jdbc:mysql://localhost/userdb \
--username root
```

Note – If you are using the import-all-tables, it is mandatory that every table in that database must have a primary key field.

Export

```
$ sqoop export (generic-args) (export-args)
$ sqoop-export (generic-args) (export-args)
```

Example

Let us take an example of the employee data in file, in HDFS. The employee data is available in **emp_data** file in 'emp/' directory in HDFS. The **emp_data** is as follows.

```
1201, gopal,     manager, 50000, TP
1202, manisha,   preader, 50000, TP
1203, kalil,     php dev, 30000, AC
1204, prasanth,  php dev, 30000, AC
1205, kranthi,   admin,   20000, TP
1206, satish p,  grp des, 20000, GR
```

It is mandatory that the table to be exported is created manually and is present in the database from where it has to be exported.

Export Cont.…

The following query is used to create the table 'employee' in mysql command line.

```
$ mysql
mysql> USE db;
mysql> CREATE TABLE employee (
   id INT NOT NULL PRIMARY KEY,
   name VARCHAR(20),
   deg VARCHAR(20),
   salary INT,
   dept VARCHAR(10));
```

The following command is used to export the table data (which is in **emp_data** file on HDFS) to the employee table in db database of Mysql database server.

```
$ sqoop export \
--connect jdbc:mysql://localhost/db \
--username root \
--table employee \
--export-dir /emp/emp_data
```

The following command is used to verify the table in mysql command line.

```
mysql>select * from employee;
```

If the given data is stored successfully, then you can find the following table of given employee data.

```
+------+--------------+-------------+-------------------+--------+
| Id   | Name         | Designation | Salary            | Dept   |
+------+--------------+-------------+-------------------+--------+
| 1201 | gopal        | manager     | 50000             | TP     |
| 1202 | manisha      | preader     | 50000             | TP     |
| 1203 | kalil        | php dev     | 30000             | AC     |
| 1204 | prasanth     | php dev     | 30000             | AC     |
| 1205 | kranthi      | admin       | 20000             | TP     |
| 1206 | satish p     | grp des     | 20000             | GR     |
+------+--------------+-------------+-------------------+--------+
```


Sqoop JOB

Sqoop job creates and saves the import and export commands. It specifies parameters to identify and recall the saved job.

This re-calling or re-executing is used in the incremental import, which can import the updated rows from RDBMS table to HDFS.

Create & Verify job with the name "**myjob**", which can import the table data from RDBMS table to HDFS

Create Job (--create)

```
$ sqoop job --create myjob \
-- import \
--connect jdbc:mysql://localhost/db \
--username root \
--table employee --m 1
```

Verify Job (--list)

'**--list**' argument is used to verify the saved jobs. The following command is used to verify the list of saved Sqoop jobs.

```
$ sqoop job --list
```

It shows the list of saved jobs.

```
Available jobs:
  myjob
```


Inspect & Execute JOB

Inspect Job (--show)

'**--show**' argument is used to inspect or verify particular jobs and their details. The following command and sample output is used to verify a job called **myjob**.

```
$ sqoop job --show myjob
```

It shows the tools and their options, which are used in **myjob**.

```
Job: myjob
Tool: import Options:
----------------------------
direct.import = true
codegen.input.delimiters.record = 0
hdfs.append.dir = false
db.table = employee
...
incremental.last.value = 1206
...
```

Execute Job (--exec)

'**--exec**' option is used to execute a saved job. The following command is used to execute a saved job called **myjob**.

```
$ sqoop job --exec myjob
```

It shows you the following output.

```
10/08/19 13:08:45 INFO tool.CodeGenTool: Beginning code generation
...
```


Sqoop Eval Tool

It allows users to execute user-defined queries against respective database servers and preview the result in the console. So, the user can expect the resultant table data to import. Using eval, we can evaluate any type of SQL query that can be either DDL or DML statement.

Example: Select & Insert Query Evaluation

```
$ sqoop eval \
--connect jdbc:mysql://localhost/db \
--username root \
--query "SELECT * FROM employee LIMIT 3"
```

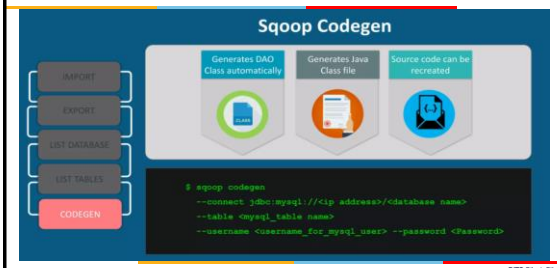
If the command executes successfully, then it will produce the following output on the terminal.

```
+------+--------------+-------------+-------------------+--------+
| Id   | Name         | Designation | Salary            | Dept   |
+------+--------------+-------------+-------------------+--------+
| 1201 | gopal        | manager     | 50000             | TP     |
| 1202 | manisha      | preader     | 50000             | TP     |
| 1203 | khalil       | php dev     | 30000             | AC     |
+------+--------------+-------------+-------------------+--------+
```

```
$ sqoop eval \
--connect jdbc:mysql://localhost/db \
--username root \
-e "INSERT INTO employee VALUES(1207,'Raju','UI dev',15000,'TP')"
```

If the command executes successfully, then it will display the status of the updated rows on the console.

Sqoop Codegen



Codegen Tool

It generates DAO class in Java, based on the Table Schema structure. The Java definition is instantiated as a part of the import process. The main usage of this tool is to check if Java lost the Java code. If so, it will create a new version of Java with the default delimiter between fields.

Example

The following command is used to execute the given example.

```
$ sqoop codegen \
--connect jdbc:mysql://localhost/userdb \
--username root \
--table emp
```

If the command executes successfully, then it will produce the following output on the terminal.

```
14/12/23 02:34:40 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5
14/12/23 02:34:41 INFO tool.CodeGenTool: Beginning code generation
----------
14/12/23 02:34:42 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/local/hadoop
Note: /tmp/sqoop-hadoop/compile/9a300a1f94899df4a9b10f9935ed9f91/emp.java uses or
  overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
14/12/23 02:34:47 INFO orm.CompilationManager: Writing jar file:
  /tmp/sqoop-hadoop/compile/9a300a1f94899df4a9b10f9935ed9f91/emp.jar
```

Let us take a look at the output. The path, which is in bold, is the location that the Java code of the **emp** table generates and stores. Let us verify the files in that location using the following commands.

```
$ cd /tmp/sqoop-hadoop/compile/9a300a1f94899df4a9b10f9935ed9f91/
$ ls
emp.class
emp.jar
emp.java
```

If you want to verify in depth, compare the **emp** table in the **userdb** database and **emp.java** in the following directory

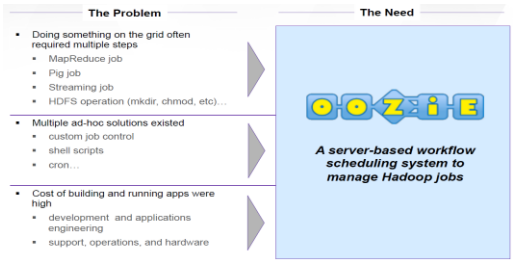
/tmp/sqoop-hadoop/compile/9a300a1f94899df4a9b10f9935ed9f91/.

Apache Oozie: Workflow Scheduler

Overview of Oozie

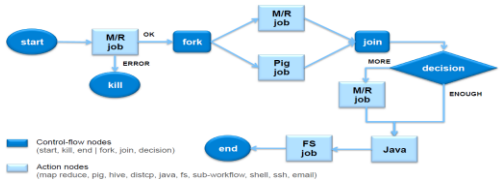
- Oozie is an open-source Apache project that provides a framework for coordinating and scheduling Hadoop jobs. Oozie is not restricted to just MapReduce jobs; you can use Oozie to schedule Pig, Hive, Sqoop, Streaming jobs, and even Java programs.
- Oozie is a Java web application that runs in a Tomcat instance.

Why Oozie?

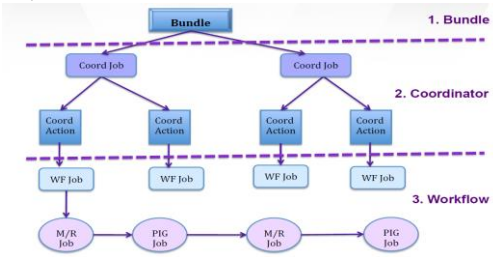


Oozie as a DAG

- Oozie executes workflow defined as DAG of jobs (Directed Acyclic Graph)
- The job type includes MapReduce, Pig, Hive, shell script, custom Java code etc.
- Introduced in Oozie 1.x



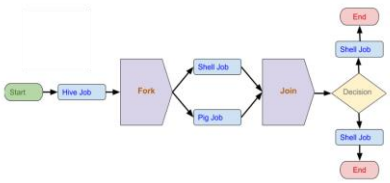
Layer of Abstraction



Oozie-Workflow

- An Apache Oozie workflow is a sequence of actions organized as a directed acyclic graph (DAG). These actions depend on each other, so that the next action can only be executed after the previous action has been completed.
- Different types of actions can be created as required. The workflow and any scripts or .jar files must be positioned in the HDFS path before executing the workflow.
- If we want to run several jobs in parallel, we can use Fork. For each use of Fork, a join must be used at the end of the Fork. Join assumes that all nodes running in parallel are children of a single Fork, as shown in the following diagram.

Sample Workflow



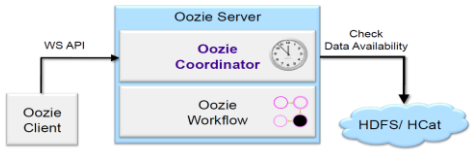
Oozie-Coordinator

- The Oozie **Coordinator** system allows the user to define and execute recurrent and interdependent workflow jobs (data application pipelines).
- Connect workflow jobs that run regularly, but at different time intervals. The outputs of multiple subsequent runs of a workflow become the input to the next workflow.

Example: The outputs of last 4 runs of a workflow that runs every 15 minutes become the input of another workflow that runs every 60 minutes. Chaining together these workflows result it is referred as a data application pipeline.

Oozie-Coordinator

- Oozie executes workflow based on
  - time dependency (frequency)
  - data dependency
- Introduced in 2.x



Oozie -Bundle

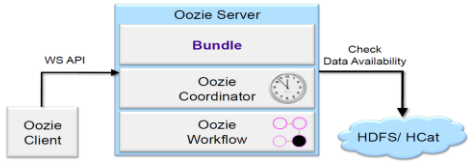
- Bundle is a higher-level oozie abstraction that will batch a set of coordinator applications. The user will be able to start/stop/suspend/resume/rerun in the bundle level resulting a better and easy operational control.
- Oozie **Bundle** system allows the user to define and execute a bunch of coordinator applications often called a data pipeline. There is no explicit dependency among the coordinator applications in a bundle. However, a user could use the data dependency of coordinator applications to create an implicit data application pipeline.

Some Key Terms

- **Kick-off-time:** The time when a bundle should start and submit coordinator applications.
- **Bundle Application:** A bundle application defines a set of coordinator applications and when to start those. Normally, bundle applications are parameterized. A bundle application is written in XML.
- **Bundle Job:** A bundle job is an executable instance of a bundle application. A job submission is done by submitting a job configuration that resolves all parameters in the application definition.
- **Bundle Definition Language:** The language used to describe bundle applications.

Bundle

- Users can define and execute a "bundle" of coordinator apps
  - large scale data processing (inter-related coordinators)
  - operability and manageability of pipelines
- User can start/stop/suspend/resume/rerun in the bundle level
- Introduced in 3.x, bundles are optional

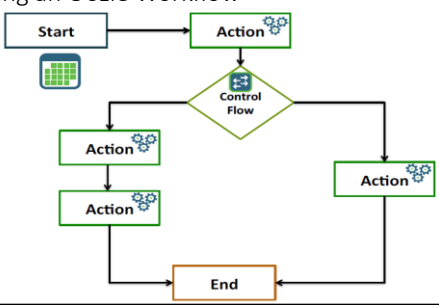


Overview of Oozie

Oozie has two main capabilities:

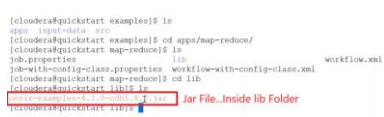
1. Oozie Workflow: a collection of actions (defined in a **workflow.xml** file).
   - Pig Actions
   - Hive Actions
   - MapReduce Actions
2. Oozie Coordinator: a recurring workflow (defined in a **coordinator.xml** file).
   - Schedule a Job Based on Time
   - Schedule a Job Based on Data Availability

Defining an Oozie Workflow



Three Important Files

1. You need the JAR file
2. workflow.xml
3. job.properties



Workflow.xml

```
<!--
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<workflow-app xmlns="uri:oozie:workflow:0.2" name="map-reduce-wf">
    <start to="mr-node"/>
    <action name="mr-node">
        <map-reduce>
            <job-tracker>${jobTracker}</job-tracker>
            <name-node>${nameNode}</name-node>
            <prepare>
                <delete path="${nameNode}/user/${wf:user()}/${examplesRoot}/output-data/${outputDir
}"/>
            </prepare>
            <configuration>
                <property>
                    <name>mapred.job.queue.name</name>
                    <value>${queueName}</value>
                </property>
                <property>
```

Start node name

Action node name

JobTracker or Resource Manager parameter

Name Node Parameter

Workflow.xml

```
                <delete path="${nameNode}/user/${wf:user()}/${examplesRoot}/output-data/${outpu
</prepare>
<configuration>
    <property>
        <name>mapred.job.queue.name</name>
        <value>${queueName}</value>
    </property>
    <property>
        <name>mapred.mapper.class</name>
        <value>org.apache.oozie.example.SampleMapper</value>
    </property>
    <property>
        <name>mapred.reducer.class</name>
        <value>org.apache.oozie.example.SampleReducer</value>
    </property>
    <property>
        <name>mapred.map.tasks</name>
        <value>1</value>
    </property>
    <property>
        <name>mapred.input.dir</name>
        <value>/user/${wf:user()}/${examplesRoot}/input-data/text</value>
    </property>
    <property>
        <name>mapred.output.dir</name>
                <value>/user/${wf:user()}/${examplesRoot}/output-data/${outputDir}</value>
            </property>
        </configuration>
    </map-reduce>
    <ok to="end"/>
    <error to="fail"/>
</action>
<kill name="fail">
    <message>Map/Reduce failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
</kill>
<end name="end"/>
```

Delete the directory if existing

Mapper Class

Reducer class

Mapper Input Directory

Mapper Output Directory

OK if successful

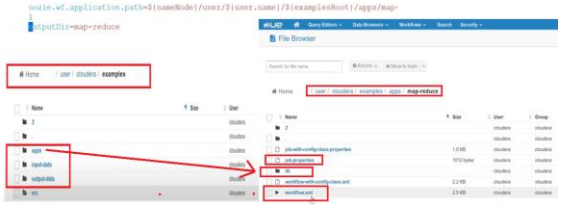
fail if Not Successfull

Error Message ..if not successful

job.properties file

```
nameNode=hdfs://localhost:8020
jobTracker=localhost:8032
queueName=default
examplesRoot=examples

oozie.wf.application.path=${nameNode}/user/${user.name}/${examplesRoot}/apps/map-
reduce
outputDir=map-reduce
```

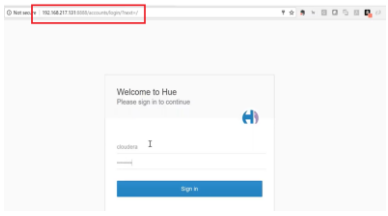
Defining the Parameter



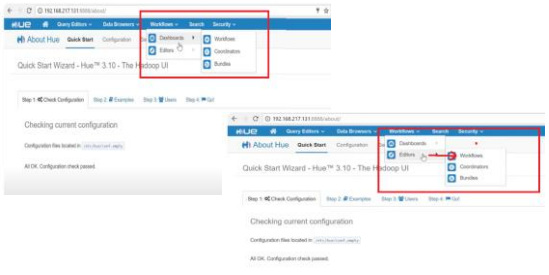
Submit the Job

```
cloudera@quickstart:~/oozie
[cloudera@quickstart oozie]$ ls
examples  oozie-examples.tar.gz
[cloudera@quickstart oozie]$ oozie job -oozie http://localhost:11000/oozie -config examples/apps/m
p-reduce/job.properties -run
job: 0000000-171030184541100-oozie-oozi-W
[cloudera@quickstart oozie]$
```

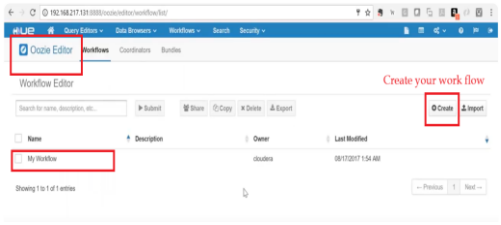
Through HUE : Example



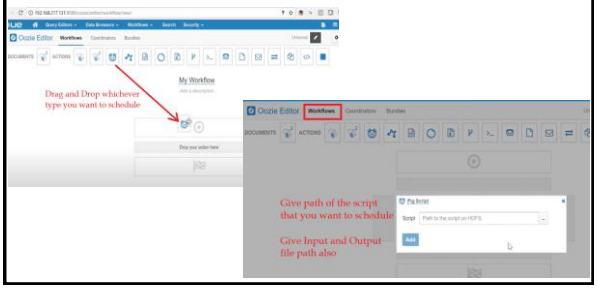
HUE Interface



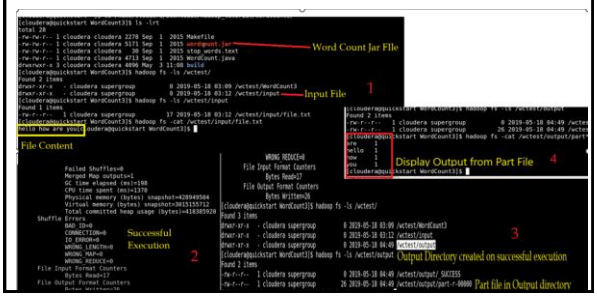
Oozie Editor



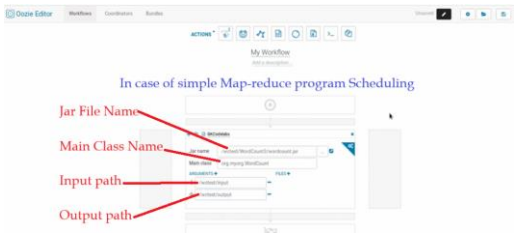
Working with Oozie Interface



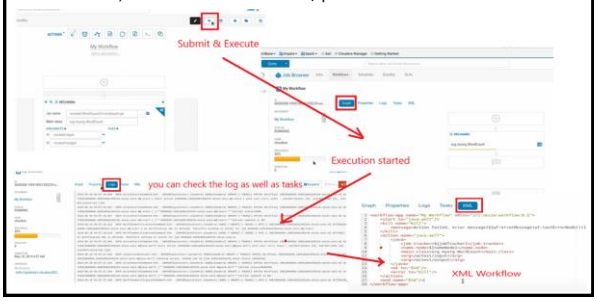
Example : Simple Map-Reduce, Word Count Program



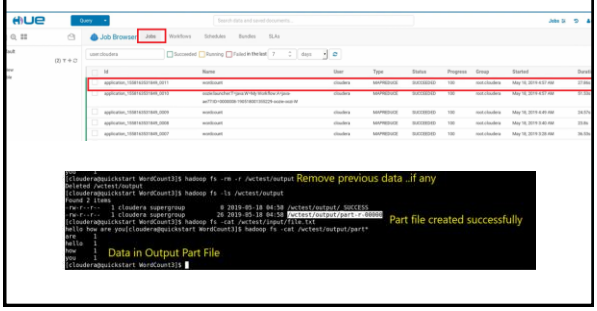
Scheduling simple Map-Reduce through UI



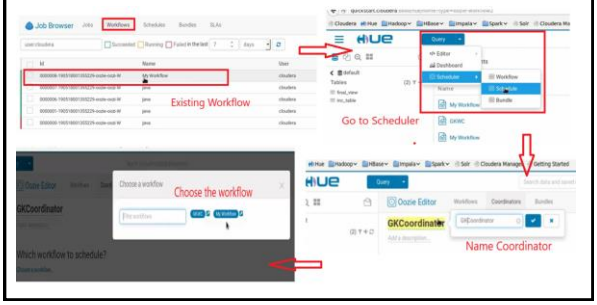
Submit, Execute & check O/p



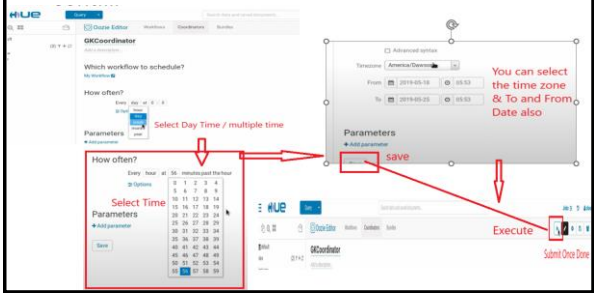
O/p



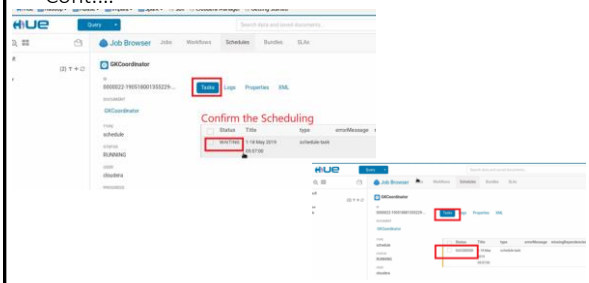
Scheduling using Start & End Time through Coordinator

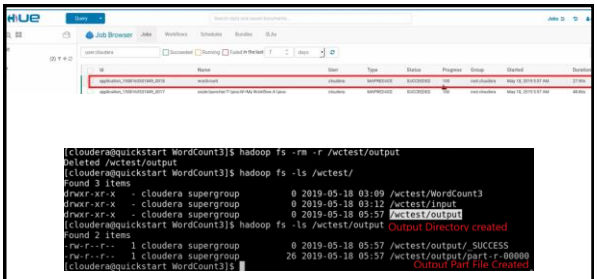


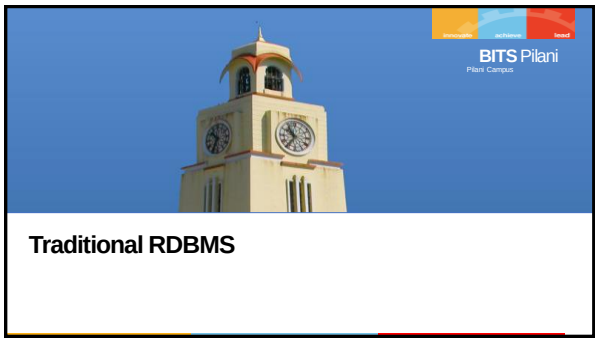
Cont....



Cont....







Traditional RDBMS

User

ID	FName	LName	DOB	city	state	country	pin
111	ABC	XYZ	01/01/2000	BANGALORE	KA	India	001
112	EFG	HIJ	21/01/2000	BANGALORE	KA	India	001

Order

ID	UserID	Product	Amount
200	111	Mobile	1000
201	112	Laptop	5000
203	112	Keyboard	500

SQL Joins

SQL JOIN operations are used to combine rows from two or more tables, and are of two types:

•Conditional Join: Combine rows based on a condition over one or more common columns (typically primary or foreign keys).

•There are 4 types of conditional joins: inner, left, right, and full.

•Cross Join: Cartesian Product of two tables.

Inner Join

An Inner Join produces a row combining a row from both tables only if the key is present in *both* tables.
For example, if you had run two marketing campaigns on different channels that captured potential customer info in two tables:
•Table A: Name and phone number
•Table B: Name and email

```
Table A:
      Name      |      Phone
----------------+----------------
 Warren Buffett | +1-123-456-7890
 Bill Gates     | +1-987-654-3210

Table B:
      Name      |      Email
----------------+----------------
 Bill Gates     | bgates@live.com
 Larry Page     | lpage@gmail.com
```

Inner Join Cont....

You can find all customer leads for which you have both their phone number and email address using an Inner Join:

SQL

```sql
SELECT A.Name, A.Phone, B.Email
FROM A
INNER JOIN B
ON A.name = B.name;
```

The word `INNER` is optional and you can omit it if you want. The result will be:

Code

```
      Name      |      Phone      |      Email
----------------+-----------------+-----------------
 Bill Gates     | +1-987-654-3210 | bgates@live.com
```

Left Outer Join

A Left Join returns all rows from the left table and the matched rows from the right table.
Say, your sales team decides to reach out to all customer leads over the phone, but, if possible, wants to follow up by email too.

With a Left Join, you can create a list of customers where you have their phone numbers for sure but may also have their email:

SQL

```sql
SELECT A.Name, A.Phone, B.Email
FROM A
LEFT OUTER JOIN B
ON A.name = B.name;
```

The word `OUTER` is optional and you can omit it if you want. The result will be:

Code

```
      Name      |      Phone      |      Email
----------------+-----------------+-----------------
 Warren Buffett | +1-123-456-7890 |      null
 Bill Gates     | +1-987-654-3210 | bgates@live.com
```

Right Outer Join

A Right Join returns all rows from the right table and the matched rows from the left table.
Say, you want the opposite: all customers with an email address and, if possible, their phone numbers too:

SQL

```sql
SELECT B.Name, A.Phone, B.Email
FROM A
RIGHT OUTER JOIN B
ON A.name = B.name;
```

The word `OUTER` is optional and you can omit it if you want. The result will be:

Code

```
      Name      |      Phone      |      Email
----------------+-----------------+-----------------
 Bill Gates     | +1-987-654-3210 | bgates@live.com
 Larry Page     |      null       | lpage@gmail.com
```

Full Outer Join

A Full Join returns all rows from both tables matching them whenever possible.

Say, you want to have a consolidated list of all customers having a phone number or email address or, if possible, both:

SQL

```sql
SELECT
  CASE
    WHEN A.Name IS NOT NULL THEN A.Name
    ELSE B.Name
  END AS Name,
  A.Phone, B.Email
FROM A
FULL OUTER JOIN B
ON A.name = B.name;
```

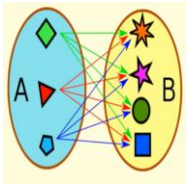
The word `OUTER` is optional and you can omit it if you want. In some databases, a simple `SELECT *` might work and you will not need that `CASE` statement. The result will be:

Code

```
      Name      |      Phone      |      Email
----------------+-----------------+-----------------
 Warren Buffett | +1-123-456-7890 |      null
 Bill Gates     | +1-987-654-3210 | bgates@live.com
 Larry Page     |      null       | lpage@gmail.com
```

Cross Join

- QL Cross Join is the Cartesian Product of two tables. It pairs each row of the left table with each row of the right table.
- Say, you have a table CarType with 7 rows having values: Sedan, Hatchback, Coupe, MPV, SUV, Pickup Truck, and Convertible.
- And you also have a table Color with 5 rows: Red, Black, Blue, Silver Grey, and Green.
- A Cross Join between the two will generate a table with 35 rows having all possible combinations of car types and colors:



Cross Join Example

SQL

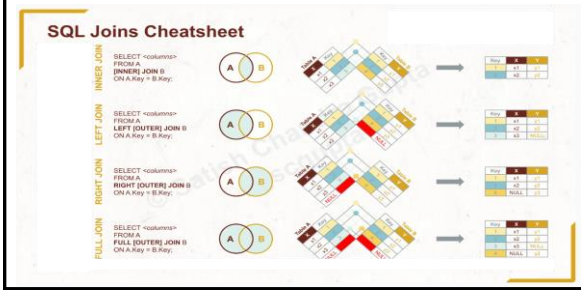
```
SELECT *
FROM CarType
CROSS JOIN Color;
```

SQL Statement:

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
CROSS JOIN Orders;
```

For total 3 customers in customers table and 4 Orders in Orders table total 12 row output Will be generated

SQL Joins - Cheatsheet

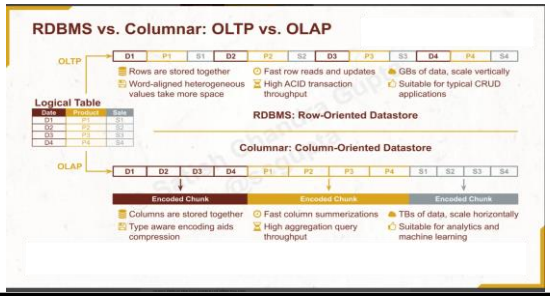


POINTER

Data warehouses are Columnar databases, i.e., values in a column are stored together (in RDBMS, the rows are stored together).

Therefore, SELECT * is a particularly bad idea.

Real Time use of Row & Columnar DB -OLAP VS OLTP



Why NoSQL

Relational databases are optimized for transaction operations. Transactions often update multiple records in multiple tables. Indexes are optimized for frequent low-latency writes of ACID Transactions.

While transactions are on rows (records), analytics properties are computed on columns (attributes). OLAP applications need an optimized column-read operation on a table. Columnar databases are designed for high-throughput of column aggregations. That's why Columnar DBs are row-oriented databases.

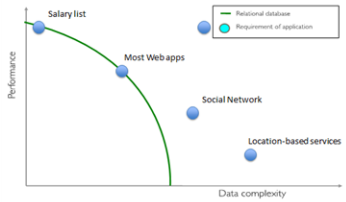
However, the primary RDBMS operation is low-latency high-frequency ACID transactions. That does not scale to the Big Data scale common in analytics applications.

Why NOSQL now?? Ans. Driving Trends

11

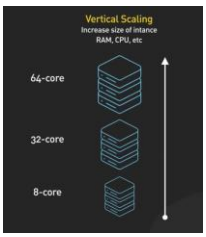
Side note: RDBMS performance

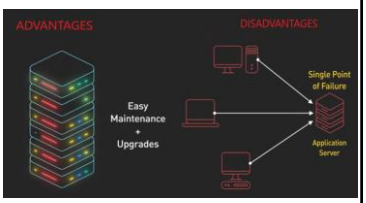
12



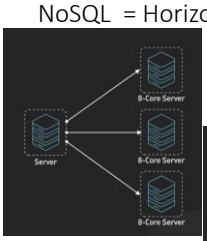
Contrast

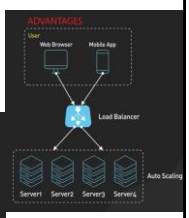
RDBMS = Vertical Scaling

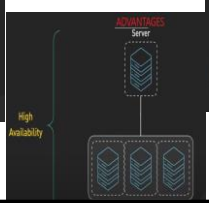




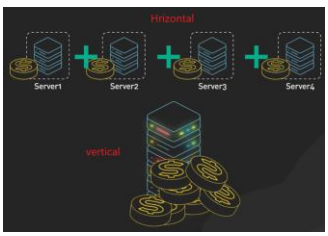
NoSQL = Horizontal Scaling



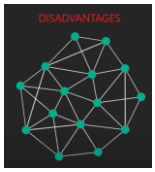


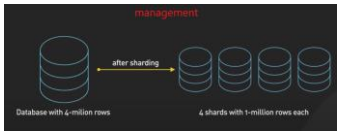


Cost…

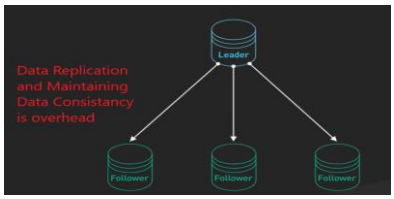


Disadvantages -- Horizontal

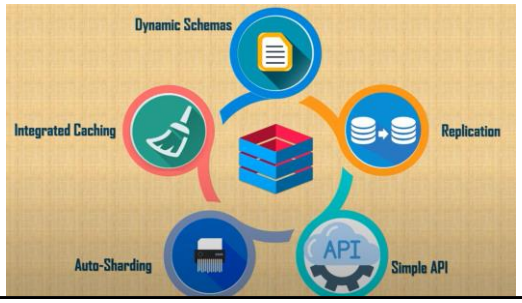




Disadvantages

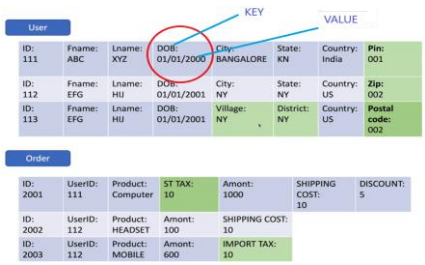


NoSQL Features

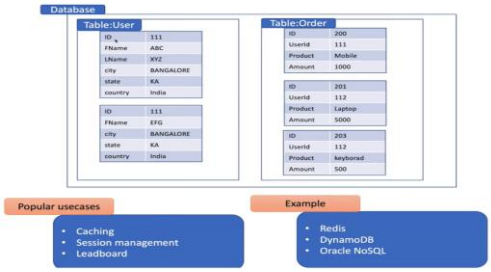


NoSQL DB Types

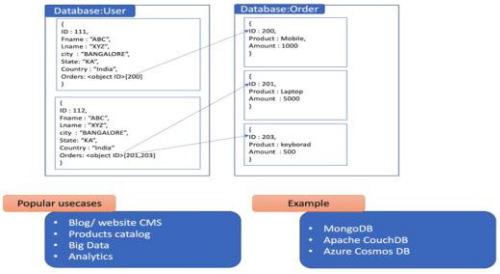
NoSQL DB



Key-Value Pair



Document DB



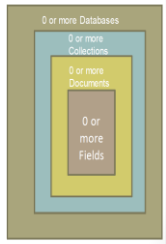
MongoDB JSON

```
{"widget": {
    "debug": "on",
    "window": {
        "title": "Sample Konfabulator Widget",
        "name": "main_window",
        "width": 500,
        "height": 500
    },
    "image": {
        "src": "Images/Sun.png",
        "name": "sun1",
        "hOffset": 250,
        "vOffset": 250,
        "alignment": "center"
    },
    "text": {
        "data": "Click Here",
        "size": 36,
        "style": "bold",
        "name": "text1",
        "hOffset": 250,
        "vOffset": 100,
        "alignment": "center",
        "onMouseUp": "sun1.opacity = (sun1.opacity / 100) * 90;"
    }
}}
```

Structure

Mapping with RDBMS

- A MongoDB instance may have zero or more 'databases'
- A database may have zero or more 'collections'.
- A collection may have zero or more 'documents'.
- A document may have one or more 'fields'.
- MongoDB 'Indexes' function much like their RDBMS counterparts.



RDBMS		MongoDB
Database	➡	Database
Table, View	➡	Collection
Row	➡	Document (BSON)
Column	➡	Field
Index	➡	Index
Join	➡	Embedded Document
Foreign Key	➡	Reference
Partition	➡	Shard

Replication and Partitioning

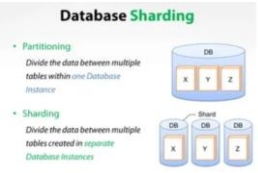
Replication (Copying data)— Keeping a copy of same data on multiple servers that are connected via a network.

Partitioning — Splitting up a large monolithic database into multiple smaller databases based on data cohesion, e.g. Horizontal (sharding) and Vertical (increase server size) partitioning.

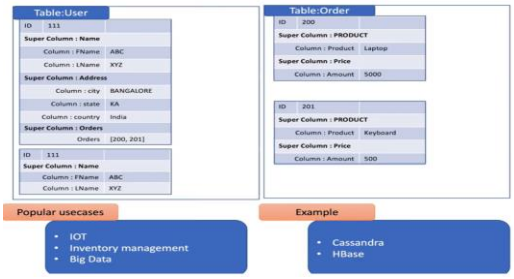
Partitioning and Sharding

All orders placed in January can be stored in one partition, all orders placed in February can be stored in another partition, and so on. Each partition can then be stored on a separate server. This way, when the company needs to retrieve orders from a particular time period, it can query only the relevant partition, which can significantly improve performance.

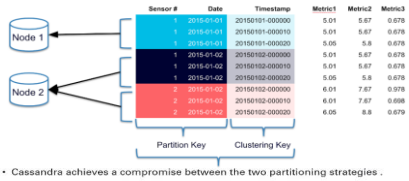
All orders from customers in North America can be stored on one server, all orders from customers in Europe can be stored on another server, and so on. This way, when the company needs to retrieve orders from a particular region, it can query only the relevant server, which can help improve performance and reduce the load on any one server.



Wide Column DB

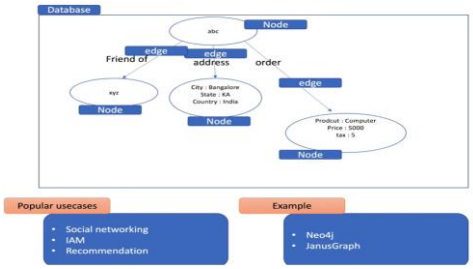


Cassandra Strategy



- Cassandra achieves a compromise between the two partitioning strategies .
- A table in Cassandra can be declared with a *compound primary key* consisting of several columns. Only the first part of that key is hashed to determine the partition, but the other columns are used as a concatenated index for sorting the data in Cassandra's SSTables.

Graph DB



Before Cloud

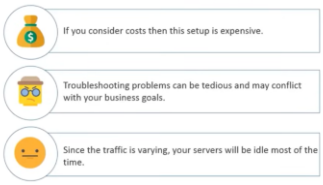
Suppose you want to host a website, these are the following things that you would need to do:


Buy a stack of servers.

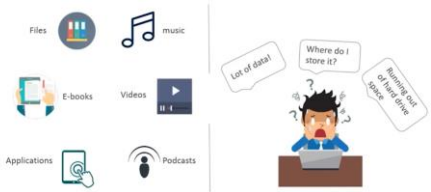

Monitoring and Maintain servers.


High traffic? More servers.

Potential Problem



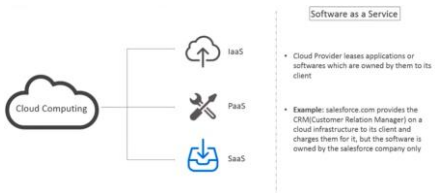
Why Cloud?



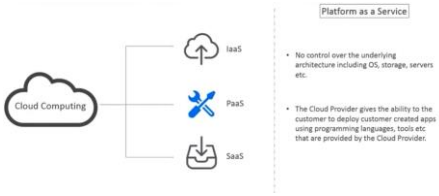
How cloud can support?

What cloud provide?

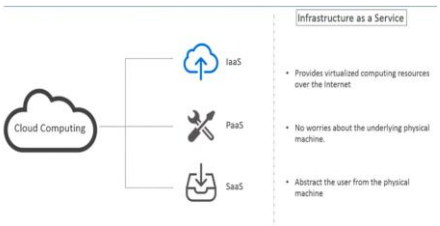
Service Models : SaaS



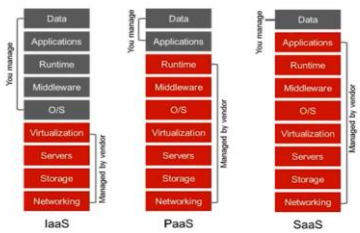
Service Models : PaaS



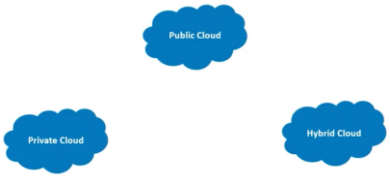
Service Models : IaaS



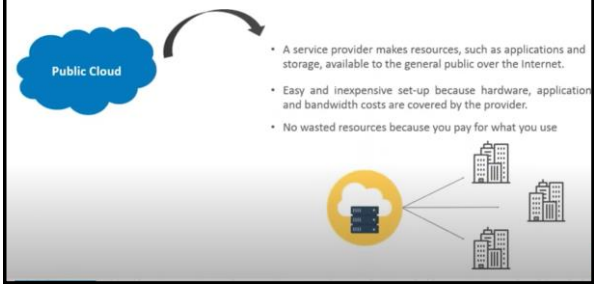
Service Model Architecture



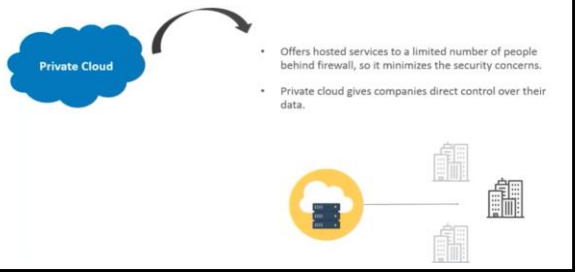
Cloud Deployment Models



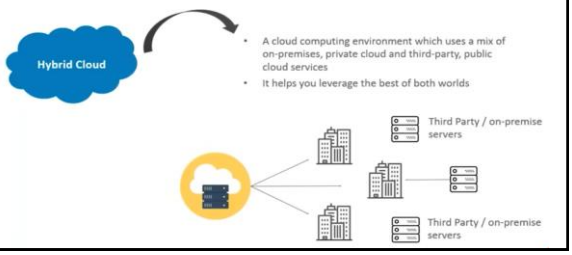
Public



Private



Hybrid



On Premises VS On Cloud -- Comparison

- Write Your own piece of code on your own server
- Provide your own storage
- You will be fully responsible for Security
- You need to take care of availability
- You are supposed to maintain and monitor all applications
- You will have complete control over your data and access

Colo Deployment :Colocation deployment, also known as "colo" deployment, is when a business rents space in a data center to store their computing hardware and servers

- The time to deploy a new workload in an On Prem or Colo deployment is typicall at least a week:
- The purchase needs to be approved and the server ordered
- The server needs to be delivered
- It needs to be physically racked up and cabled
- The server team need to install and configure the OS, patches, any standard software and the applications
- The network team need to configure switches, routers and firewalls
- The storage team need to configure the storage system and SAN switches
- The lead time is quicker when provisioning virtual machines but it still takes time for the different teams to manually complete their tasks

On Cloud

Database , Storage , Computation, Compliance, Security , analytics and many other services

YOU JUST FOCUS ON YOUR BUZINESS NEEDS

Few Issues – On Cloud

Virtualization

Virtualization is a technique, which allows to share single physical instance of an application or resource among multiple organizations or tenants (customers). It does so by **assigning a logical name** to a physical resource and providing a **pointer to that physical resource** on demand.

Virtualization – What it is?

- Creating a virtual machine over existing operating system and hardware is referred as Hardware Virtualization. Virtual Machines provide an environment that is logically separated from the underlying hardware.
- The machine on which the virtual machine is created is known as **host machine** and **virtual machine** is referred as a **guest machine.** This virtual machine is managed by a software or firmware, which is known as **hypervisor.**

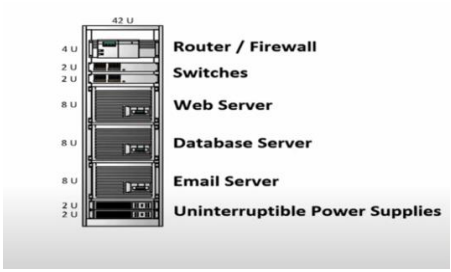
Virtualization .. Cont....

Virtualization is the underlying core technology of cloud computing

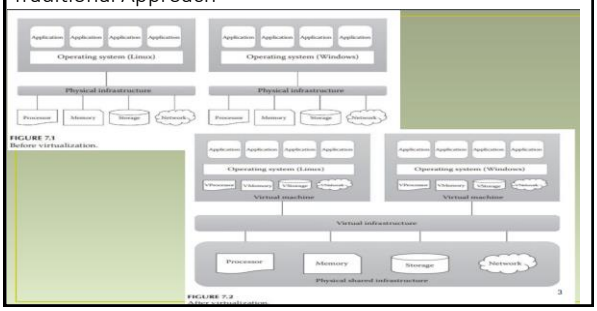
Advantages

- • Using virtualization, the physical infrastructure owned by the service provider is shared among many users, increasing the resource utilization.
- • Virtualization provides efficient resource utilization and increased return on investment (ROI).
- • Ultimately, it results in low capital expenditures (CapEx) and operational expenditures (OpEx).
- • Promotes the green IT by reducing energy wastage.
- • Dynamic data center • Improves disaster recovery

Traditional Approach

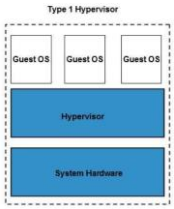


Traditional Approach

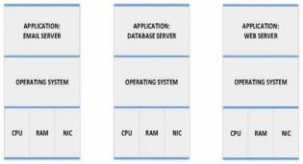


Hypervisor

The **hypervisor** is a firmware or low-level program that acts as a Virtual Machine Manager. There are two types of hypervisor:
Type 1 hypervisor executes on bare system. LynxSecure, RTS Hypervisor, Oracle VM, Sun xVM Server, VirtualLogic VLX are examples of Type 1 hypervisor. The following diagram shows the Type 1 hypervisor.

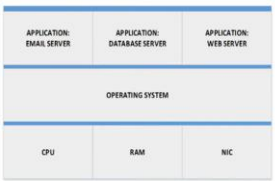


Before Virtualization



- Server utilization (CPU, RAM, NIC etc.) around 15%
- I had to pay for each separate server, and they're all using power, space and cooling

Some common BAD Practice

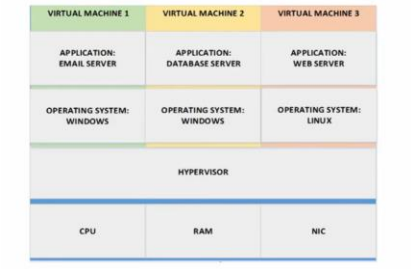


- Putting multiple applications on the same server would improve utilization

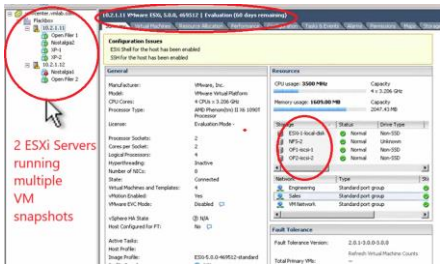
Hypervisor –Native/Bare-Metal-type 1

- Type 1 Hypervisors run directly on the system hardware
- VMware ESXi (part of the vSphere suite)
- Microsoft Hyper-V
- Red Hat KVM
- Oracle VM Server
- Citrix XenServer

Server Virtualization

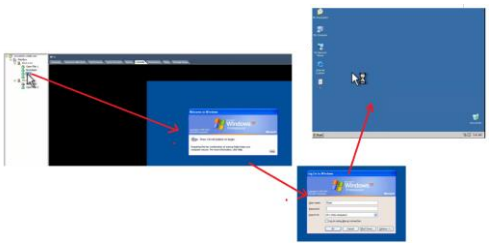


Implementation

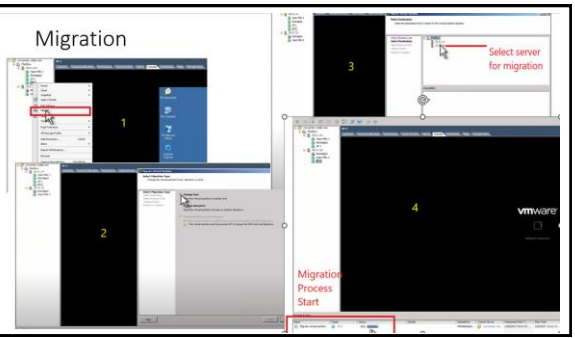


10.11 machine And 10.12 machine will have separate Secondary memory

XP-1 Instance



Migration

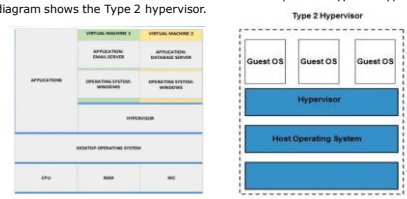


Hypervisor – Hosted Hypervisor- type 2

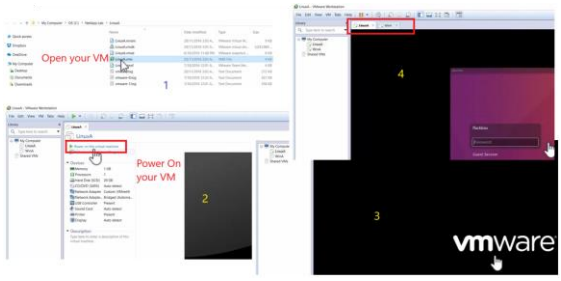
- Type 2 Hypervisors run on top of a host operating system
- VMware Workstation, Player and Fusion
- VirtualBox
- QEMU
- Parallels

Hypervisor

The **type1 hypervisor** does not have any host operating system because they are installed on a bare system.
Type 2 hypervisor is a software interface that emulates the devices with which a system normally interacts. Containers, KVM, Microsoft Hyper V, VMWare Fusion, Virtual Server 2005 R2, Windows Virtual PC and **VMWare workstation 6.0** are examples of Type 2 hypervisor. The following diagram shows the Type 2 hypervisor.



Hypervison-Type2 Implementation



Summary

Summary of Hypervisors

Hypervisor	Vendor	Type	License
Xen	University of Cambridge Computer Laboratory	Type 1	GNU GPL v2
VMWare ESXi	VMware, Inc.	Type 1	Proprietary
Hyper-V	Microsoft	Type 1	Proprietary
KVM	Open virtualization alliance	Type 2	GNU general public license
VMWare workstation	VMware, Inc.	Type 2	Shareware
Oracle Virtualbox	Oracle Corporation	Type 2	GNU general public license version 2

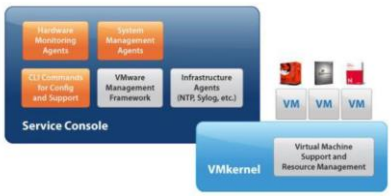
ESX & ESXi

ESX (Elastic Sky X) is the VMware's enterprise server virtualization platform.

In ESX, VMkernel is the virtualization kernel which is managed by a console operating system which is also called as Service console.

Which is linux based and its main purpose is to provide a Management interface for the host and lot of management agents and other third party software agents are installed on the service console to provide the functionalists like hardware management and monitoring of ESX hypervisor.

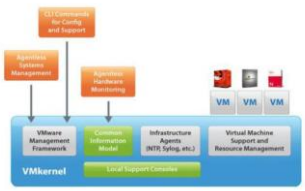
ESX



ESXi

ESXi (Elastic sky X Integrated) is also the VMware's enterprise server virtualization platform. In ESXi, Service console is removed. All the VMware related agents and third party agents such as management and monitoring agents can also run directly on the VMkernel. ESXi uses Direct Console User Interface (DCUI) instead of a service console to perform management of ESXi server. ESXi installation will happen very quickly as compared to ESX installation.

ESXi



Hardware Virtualization Types

Hardware virtualization is further subdivided into the following types:

- ✓Full Virtualization – In it, the complete simulation of the actual hardware takes place to allow software to run an unmodified guest OS.
- ✓Para Virtualization – In this type of virtualization, software unmodified runs in modified OS as a separate system.
- ✓Partial Virtualization – In this type of hardware virtualization, the software may need modification to run.

Full Virtualization

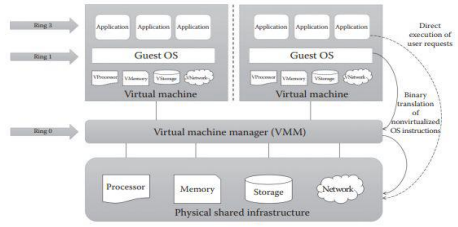


FIGURE 7.10
Full virtualization.

Para Virtualization

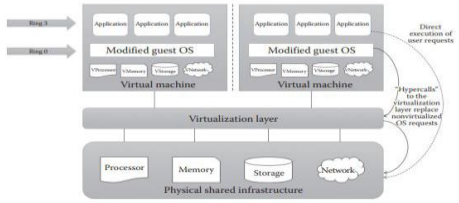
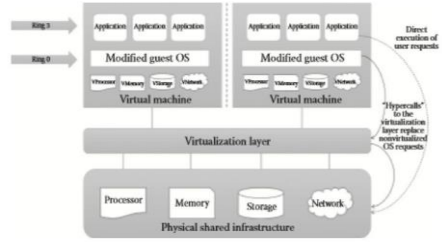


FIGURE 7.11
Paravirtualization.

Partial Virtualization

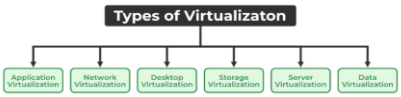


Summary

Summary of the Different Approaches to Virtualization

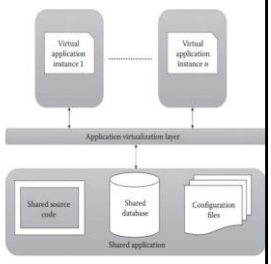
	Full Virtualization	Paravirtualization	Hardware-Assisted Virtualization
Technique	Binary translation and direct execution	Hypercalls	OS requests trap to VMM without binary translation or paravirtualization
Guest OS modification	No	Yes	No
Compatibility	Excellent compatibility	Poor compatibility	Excellent compatibility
Is guest OS hypervisor independent?	Yes	No	Yes
Performance	Good	Better in certain cases	Fair
Position of VMM and privilege level	Ring 0 Root privilege	Below ring 0	Below ring 0 Root privilege
Position of guest OS and privilege level	Ring 1 Nonroot privilege	Ring 0 Root privilege	Ring 0 Nonroot privilege
Popular vendor(s)	VMware ESX	Xen	Microsoft, Virtual Iron, and XenSorce

Software Virtualization Types



Software/Application Virtualization

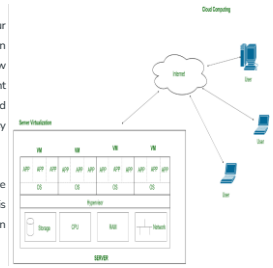
It lessens the time needed for keeping up tasks, like system upgrades and backups. In other words, if an application is deployed on a virtual server, then only essential parts need to be refreshed which leads to fewer changes on the customer's side compared with physical devices. This lets companies keep their business operations running effortlessly while having negligible downtime because of maintenance activities associated with systems



Server Virtualization

Now Consider situation, You are using Mac OS on your machine but particular application for your project can be operated only on Windows. You can either buy new machine running windows or create virtual environment in which windows can be installed and used. Second option is better because of less cost and easy implementation. This scenario is called Virtualization.

In it, virtual CPU, RAM, NIC and other resources are provided to OS which it needed to run. This resources is virtually provided and controlled by an application called Hypervisor.

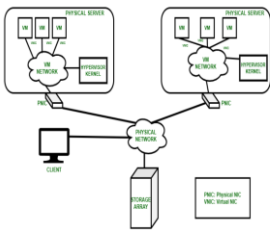


Network Virtualization

It grants organisations the capacity to create separate, logically detached networks on top of their pre-existing infrastructure, allowing for simpler implementation of assorted applications and services.

By virtualising their network configuration, businesses can take advantage of increased scalability, security and dependability while also reducing IT overhead costs.

It essentially abstracts all underlying physical hardware elements and produces multiple "virtual" networking layers which can function independently of one another – granting companies an extra measure of control over their structure as well as greater flexibility when it comes to deployment options!



Network Virtualization

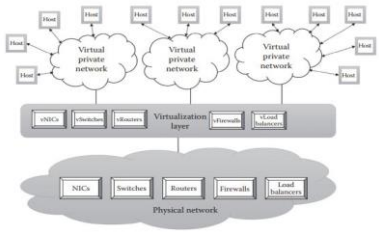
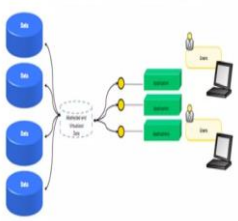


FIGURE 7.6
Network virtualization.

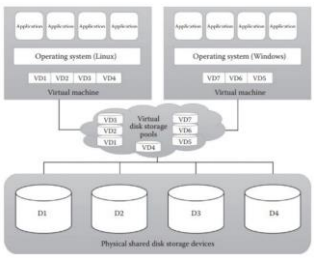
Storage Virtualization

Storage virtualization is one of the most widely used types of virtualization in cloud computing. This technology allows organizations to access, manage and store data across different servers without needing physical storage – allowing them to share resources between their networks as if they were all located together. With this tech, teams can gain access to information from any location regardless of where it's physically stored – which has multiple benefits for businesses wanting to optimize their storage solutions.

- Improved storage management in a heterogeneous IT environment
- Easy updates, better availability
- Reduced downtime
- Better storage utilization
- Automated management



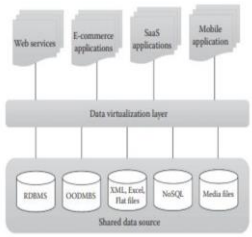
Storage Virtualization



Data Virtualization

Data virtualization is the ability to retrieve the data without knowing its type and the physical location where it is stored.

- It aggregates the heterogeneous data from the different sources to a single logical/virtual volume of data.
- This logical data can be accessed from any applications such as web services, E-commerce applications, web portals, Software as a Service (SaaS) applications, and mobile application.
- Data virtualization hides the type of the data and the location of the data for the application that access it.
- It also ensures the single point access to data by aggregating data from different sources. It is mainly used in data integration, business intelligence, and cloud computing



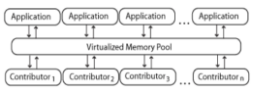
Desktop Virtualization

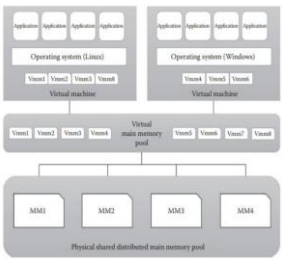
It provides the work convenience and security. As one can access remotely, you are able to work from any location and on any PC. It provides a lot of flexibility for employees to work from home or on the go. It also protects confidential data from being lost or stolen by keeping it safe on central servers.

Memory Virtualization

It introduces a way to decouple memory from the server to provide a shared, distributed or networked function. It enhances performance by providing greater memory capacity without any addition to the main memory. That's why a portion of the disk drive serves as an extension of the main memory.

Application-level integration – Applications running on connected computers directly connect to the memory pool through an API or the file system.





Impact of Virtualization

	Before	After
Servers	> 1,000	> 50
Storage	> Direct attach	> Tiered SAN and NAS
Network	> 3000 cables/ports	> 300 cables/ports
Facilities	> 200 racks	> 10 racks
	> 400 power whips	> 20 power whips

Characteristics of Cloud Computing

1. On-Demand Self-Service

With cloud computing, you can provision computing services, like server time and network storage, automatically. You won't need to interact with the service provider. Cloud customers can access their cloud accounts through a web self-service portal to view their cloud services, monitor their usage, and provision and de-provision services.

2. Broad Network Access

Another essential cloud computing characteristic is broad network access. You can access cloud services over the network and on portable devices like mobile phones, tablets, laptops, and desktop computers. A public cloud uses the internet; a private cloud uses a local area network. Latency and bandwidth both play a major role in cloud computing and broad network access, as they affect the quality of service.

3. Resource Pooling
With resource pooling, multiple customers can share physical resources using a multi-tenant model. This model assigns and reassigns physical and virtual resources based on demand. Multi-tenancy allows customers to share the same applications or infrastructure while maintaining privacy and security. Though customers won't know the exact location of their resources, they may be able to specify the location at a higher level of abstraction, such as a country, state, or data center. Memory, processing, and bandwidth are among the resources that customers can pool.

4. Rapid Elasticity
Cloud services can be elastically provisioned and released, sometimes automatically, so customers can scale quickly based on demand. The capabilities available for provisioning are practically unlimited. Customers can engage with these capabilities at any time in any quantity. Customers can also scale cloud use, capacity, and cost without extra contracts or fees. With rapid elasticity, you won't need to buy computer hardware. Instead, can use the cloud provider's cloud computing resources.

5. Measured Service
In cloud systems, a metering capability optimizes resource usage at a level of abstraction appropriate to the type of service. For example, you can use a measured service for storage, processing, bandwidth, and users. Payment is based on actual consumption by the customer via a pay-for-what-you-use model. Monitoring, controlling, and reporting resource use creates a transparent experience for both consumers and providers of the service.

Other Cloud Computing Characteristics

Resiliency and Availability

Resilience in cloud computing refers to the ability of a service to recover quickly from any disruption. Cloud resiliency is measured by how fast its servers, databases, and networks restart and recover after any damage. To prevent data loss, cloud services create a copy of the stored data. If one server loses data for any reason, the copy version from the other server restores.
Availability is a related key concept in cloud computing. The benefit of cloud services is that you can access them remotely, so there are no geographic restrictions when using cloud resources.

Flexibility
Companies need to scale as their business grows. The cloud provides customers with more freedom to scale as they please without restarting the server. They can also choose from several payment options to avoid overspending on resources they won't need.

Remote Work
Cloud computing helps users work remotely. Remote workers can safely and quickly access corporate data via their devices, including laptops and smartphones. Employees who work remotely can also communicate with each other and perform their jobs effectively using the cloud.

On Demand Services

- 'A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.'

Rapid Elasticity

- 'Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.'
  - Servers can be quickly provisioned and decommissioned based on current demand.
  - Elasticity allows customers to achieve cost savings and is often a core justification for adoption.

Cont..

- If 10 servers are required for a 3 month project, the company can provision them within minutes, pay a small monthly OpEx fee to run them rather than a large upfront CapEx cost, and decommission them at the end of the 3 months
- If a company experiences seasonal demand, such as an ecommerce store at Christmas, additional front end web servers can be **automatically** provisioned and added to a load balanced server farm, and then automatically decommissioned when the demand dissipates.

NOTE:

Capital expenditures (CapEx) are costs that often yield long-term benefits to a company. CapEx assets often have a useful life of more than one year.

Operating expenses (OpEx) are costs that often have a much shorter-term benefit.

Agenda

- What is Cloud storage?
- Types of storage
- Before Amazon S3
- What is S3?
- Benefits of S3
- Objects and Buckets
- How does Amazon S3 work
- Features of S3

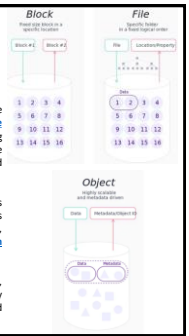
What is Cloud Storage?

Cloud storage provides a web service where your data can be stored, accessed and easily backed up by users over the internet

Cloud storage is
- reliable,
- scalable and
- secure

than traditional on-premises storage systems

Storage Types



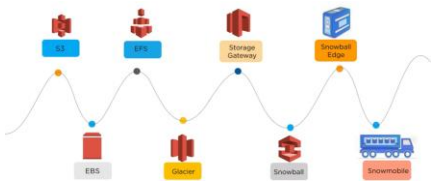
There Are 3 Types of Cloud Storage

1. **Object Storage** – The enormous scalability and metadata capabilities of object storage are frequently tapped into by cloud-based applications. Simple Storage Service (Amazon S3) and Amazon Glacier are excellent object storage options for building modern applications from the ground up that require scale and adaptability. These solutions can also be used to ingest existing data stores for analytics, backup, and archiving purposes.

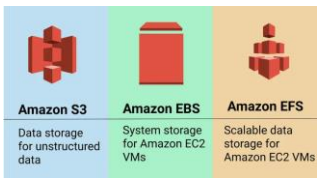
2. **File Storage** – A filing system is required since many applications need to access shared files. A Network Attached Storage (NAS) server is typically used to support this type of storage. In situations like big content repositories, development environments, media stores, or user home directories, file storage systems like Elastic File System (Amazon EFS) are ideal.

3. **Block Storage** – Other business applications, such as ERP or database systems, frequently need exclusive, low-latency storage for every host. This is frequently compared to a cargo area network (SAN) or direct-attached storage (DAS). Block-based cloud storage options such as Amazon EBS's Elastic Block Store and EC2 Instance Storage

Storage Types

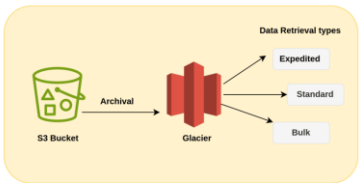


Storage Types cont....



Glacier

Amazon Web Services (AWS) Glacier is a cloud-based storage service designed for long-term data archival and backup. It's part of AWS's extensive suite of storage solutions, and it offers a cost-effective way to store data that you don't need to access frequently but must retain for compliance, regulatory, or business continuity purposes.



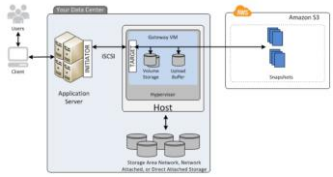
Storage Gateway

AWS Storage Gateway is a hybrid cloud storage service that allows your on-premise storage & IT infrastructure to seamlessly integrate with AWS Cloud Storage Services. It Can be AWS Provided Hardware or Compatible Virtual Machine.

Purpose of Using AWS Storage Gateway(hybrid Cloud Storage) :

- To Fulfill Licensing Requirements.
- To Achieve Data-Compliance Requirements.
- To Reduce Storage & Management Cost.
- For Easy and Effective Application Storage-Lifecycle & Backup Automation.
- For Hybrid Cloud & Easy Cloud Migration.

Storage Gateway



Snowball

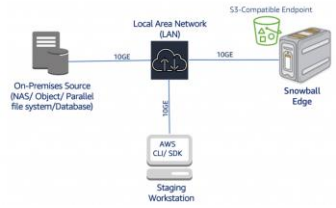
AWS Snowball is a service that allows users to transfer large amounts of data into and out of Amazon Web Services (AWS) using secure, rugged devices. Snowball devices can help address challenges associated with large-scale data transfers, such as high network costs, long transfer times, and security concerns.

It is commonly used for migrating data from on-premises data sources such as NAS arrays, databases, data warehouses, or other storage systems to AWS using Snowball Edge. The Snowball Edge Storage Optimized (SBE -SO) device has a raw storage capacity of 100 TB with a built-in Amazon S3 compatible endpoint, Amazon EC2 compute, and block storage capabilities.

This architecture uses a temporary "staging workstation." The workstation mounts the data source using the NFS or SMB protocol. For migrations of data from Hadoop (HDFS/QFS) or other file systems, this architecture enables the use of native connectors and libraries on the staging workstation to mount the data source. Once you mount the data source, use the AWS Command Line Interface (CLI) commands (copy and sync) to transfer your data into the Snowball Edge

Network File System (NFS) and Server Message Block (SMB) are file access storage protocols that allow users to share files and directories over a network.

Snowball



Snowball and Snowmobile

Snowball comes in two different models, Snowball and Snowball Edge, each with its own set of features. The original Snowball is designed for transferring up to 80 terabytes of data, while the Snowball Edge can transfer up to 100 terabytes of data and also includes computing capabilities, making it suitable for data processing tasks. Both devices are highly secure and include tamper-resistant enclosures, 256-bit encryption, and a chain of custody tracking mechanism to ensure data security during transit.

AWS Snowball is a physical device that customers can use to transfer data. It is a small, ruggedized device that customers can use to transfer data via a local network connection. This is useful for customers who have a large amount of data to transfer and don't have a fast internet connection.

AWS Snowmobile, on the other hand, is a large truck that can be used to transfer petabytes of data to the AWS cloud. This is useful for customers who have an extremely large amount of data to transfer, such as a data center migration. In summary, Snowball is for small-scale data migration, and Snowmobile is for extremely large-scale data migration.

Why Storage Services

About S3

Amazon S3 (Simple Storage Service) provides object storage which is built for storing and recovering any amount of information or data from anywhere over the internet

- Amazon S3 provides storage through web services interface
- It is designed for developers where web-scale computing can be easier for them
- It provides 99.9999999999% durability and 99.99% availability of objects
- It can store computer files up to 5 terabytes in size

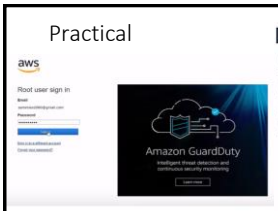
Benefits of S3

AWS Buckets and Objects

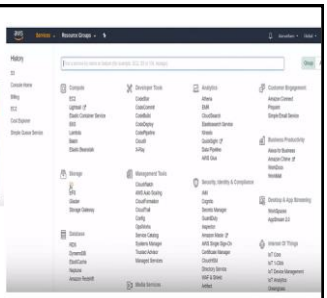
An object consists of data, key (assigned name), and metadata. A bucket is used to store objects. When data is added to a bucket, Amazon S3 creates a unique version ID and allocates it to the object.

Example of an object, bucket, and link address

Practical

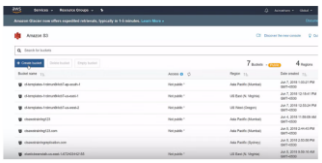


Logging into AWS



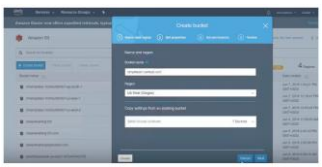
Selecting S3 from Service offerings

Bucket List



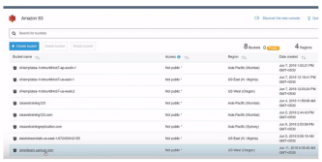
Amazon S3 bucket list (usually empty for first-time users); create a bucket by clicking on the "Create bucket" button

Create Bucket



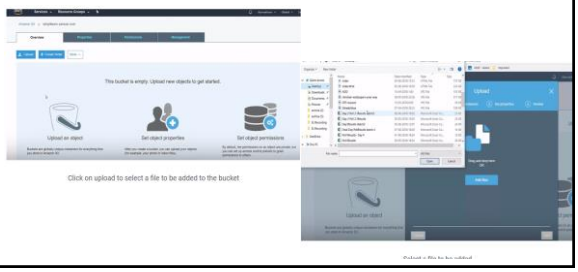
Create a bucket by setting up name, region, and other options; finish off the process by pressing the "Create" button

Select Newly Created Bucket



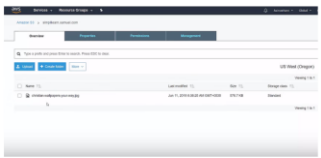
Select the created bucket

Upload File



Click on upload to select a file to be added to the bucket

Done



The file is now uploaded into the bucket

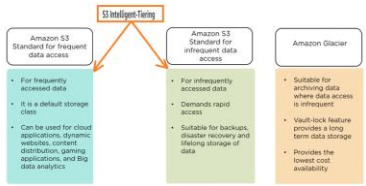
How S3 Works?

Storage Classes

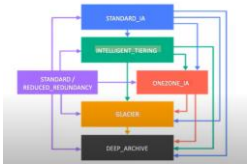
	S3 Standard	S3 Intelligent-Tiering	S3 Standard-IA	S3 One Zone-IA	S3 Glacier	S3 Glacier Deep Archive
Designed for durability	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)
Designed for availability	99.99%	99.9%	99.9%	99.5%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99.9%	99.9%
Availability Zones	≥3	≥3	≥3	1	≥3	≥3
Minimum capacity charge per object	N/A	N/A	128KB	128KB	40KB	40KB
Minimum storage duration charge	N/A	30 days	30 days	30 days	90 days	180 days
Retrieval fee	N/A	N/A	per GB retrieved	per GB retrieved	per GB retrieved	per GB retrieved

Storage Classes

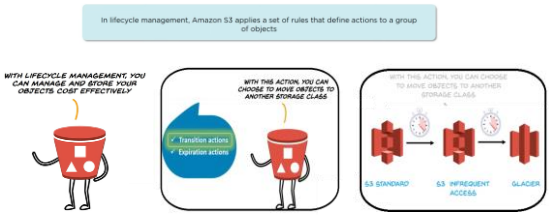
Data Movement



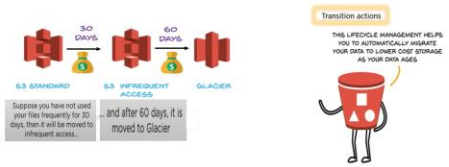
Data Movement



Life Cycle Management

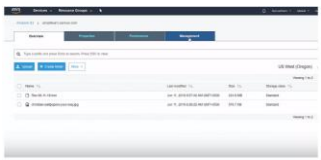


Transition Action - Example

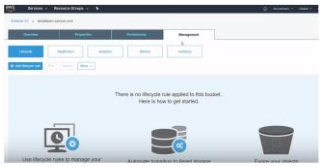


Expiration Action

Life Cycle Management - Practical



Set Rules



Select "Lifecycle" and then click on the "Add lifecycle rule."

Rule Name and Scope

Add a rule name and scope

Configure transaction options

Set Expiration Details

Set up expiration details

Save and use

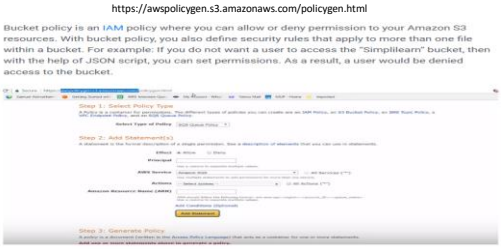
Bucket Policy

- Bucket policy is an IAM policy where you can allow and deny permission to your Amazon S3 resources
- With bucket policy, you also define security rules that apply to more than one file within a bucket
- For example: If you do not want a user to access the "Simplilearn" bucket, then with the help of JSON script you can set permissions

An Identity and Access Management (IAM) bucket policy is a mechanism that controls access to resources in a bucket, such as an Amazon S3 bucket or Google Cloud bucket:

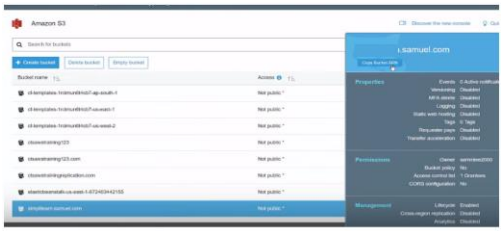
Bucket Policy Tool

https://awspolicygen.s3.amazonaws.com/policygen.html



Use an online tool to generate a policy. Select the type of policy as an S3 bucket policy. Select the appropriate effect. In this case, denying access.

Find ARN



Find the ARN of the bucket

What is ARN?

Amazon Resource Name (ARN) is a unique string that identifies an Amazon S3 bucket in the Amazon Web Services (AWS) public cloud:

ARNs follow a naming convention that includes:

- **Namespaces**: The individual elements of the ARN syntax
- **Partition and service**: The locations where the resource resides
- **Region**: The region code of the endpoint of a specific service
- **Account-id**: The user's AWS account number

Example:
arn:*partition*:*service*:*region*:*account-id*:*resource-id*
arn:*partition*:*service*:*region*:*account-id*:*resource-type*/*resource-id*
arn:*partition*:*service*:*region*:*account-id*:*resource-type*:*resource-id*

ARN Example

partition
The partition in which the resource is located. A *partition* is a group of AWS Regions. Each AWS account is scoped to one partition.

The following are the supported partitions:
- aws - AWS Regions
- aws-cn - China Regions
- aws-us-gov - AWS GovCloud (US) Regions

service
The service namespace that identifies the AWS product.

region
The Region code. For example, `us-east-2` for US East (Ohio). For the list of Region codes, see Regional endpoints in the *AWS General Reference*.

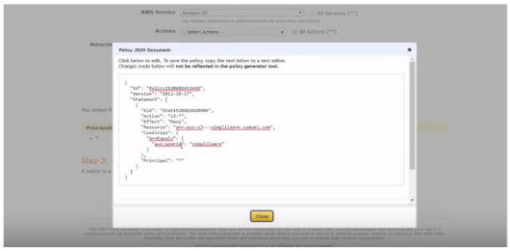
account-id
The ID of the AWS account that owns the resource, without the hyphens. For example, `123456789012`.

resource-type
The resource type. For example, `vpc` for a virtual private cloud (VPC).

resource-id
The resource identifier. This is the name of the resource, the ID of the resource, or a resource path. Some resource identifiers include a parent resource (sub-resource-type/parent-resource/sub-resource) or a qualifier such as a version (resource-type:resource-name:qualifier).

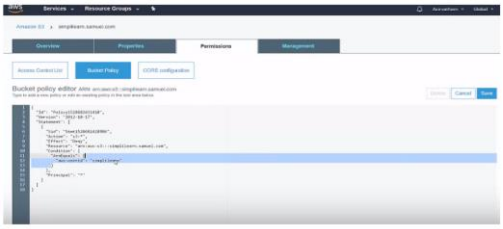
Examples

IAM user
arn:aws:iam::*123456789012*:user/*johndoe*



Set up additional conditions and set up a JSON script to deny access to a particular user. In this case, "simplilearn."

Save Permission



Go back to the bucket and set up a bucket policy under "Permissions." Then click on "Save."

Two Methods

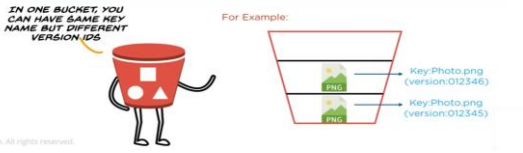
- ✓ Amazon S3 provides IT teams a highly durable, protected and scalable infrastructure designed for object storage
- ✓ Amazon S3 protects your data using 2 methods:
  - ❑ Data Encryption and
  - ❑ Versioning

Data Protection

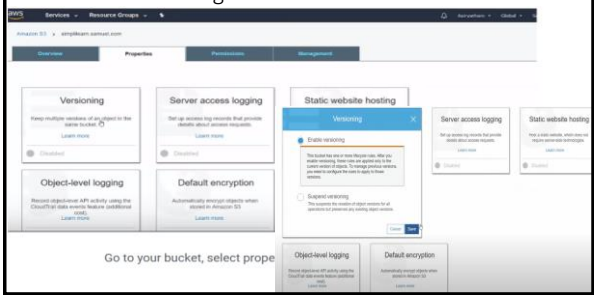
- ✓ It refers to protection of data while it's being transmitted and at rest
- ✓ Data Encryption can happen in two ways:

Versioning

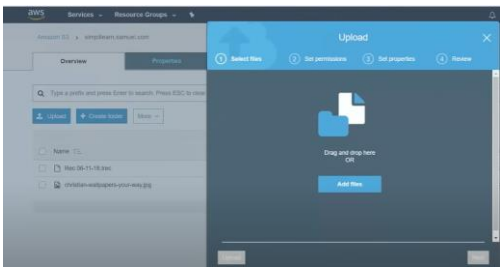
- ✓ It can be utilized to preserve, recover and restore early versions of every object you store in your Amazon S3 bucket
- ✓ Unintentional erase or overwriting of objects can be easily regained with versioning

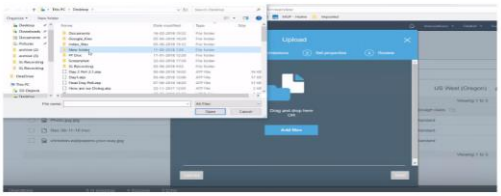


Enable Versioning

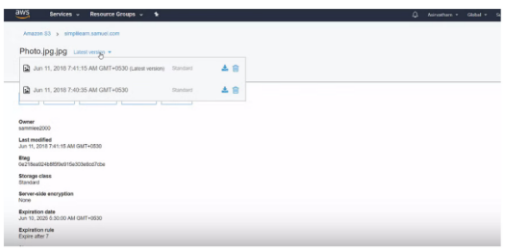


Upload an Object



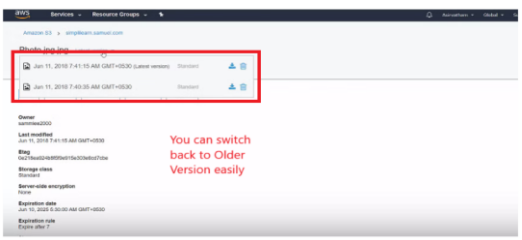


Upload a File and then
Upload another file of the same name



Select the file and alternate between its current and older versions

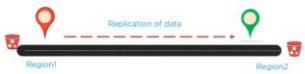
Switch Back Possible



Select the file and alternate between its current and older versions

Cross Region Replication

Cross-Region Replication provides automatic copying of every object uploaded to your buckets (source bucket and destination bucket) in different AWS regions



Note: Versioning must be turned on to enable CRR

Create New Bucket in Different Region

Cross-region replication provides automatic copying of every object uploaded to your buckets (source and destination bucket) in different AWS regions. Versioning needs to be turned on to enable CRR.

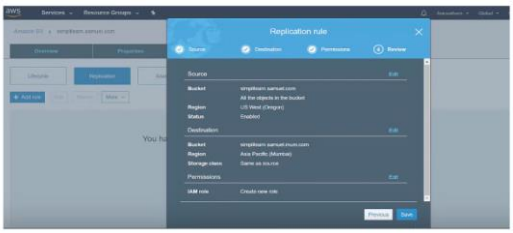
Create a new bucket in a different region

Upload & start replication

Select uploaded file, go to "Management" and then replication.

Enable Versioning on Source as well as Destination Bucket

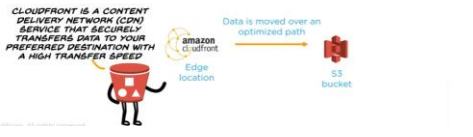
Here, click on "Add Rule."



Select the source, destination, and IAM rule

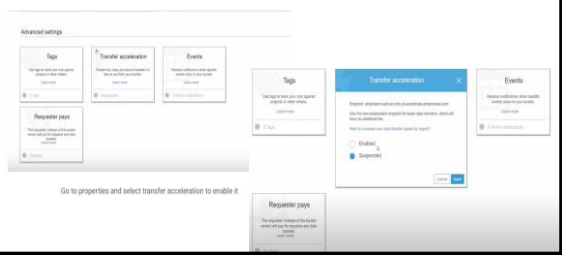
Transfer Acceleration

- ✓ It enables fast, easy and secure transfers of files over long distances between your client and S3 bucket
- ✓ The edge locations around the world provided by Amazon CloudFront are taken advantage by transfer acceleration
- ✓ It works via carrying data over an optimized network bridge that keeps running between the AWS Edge Location (closest region to your clients) and your Amazon S3 bucket



Process

https://s3-accelerate-speedtest.s3-accelerate.amazonaws.com/en/accelerate-speed-comparsion.html



Check for the speed test to see effect

https://s3-accelerate-speedtest.s3-accelerate.amazonaws.com/en/accelerate-speed-comparsion.html

Amazon EBS- Is a Hard Disk With Root Volume

- EBS Volume
- EBS Snapshot
- Data Life Cycle Manager

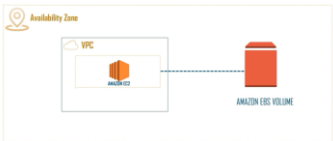
Elastic Block Store volume – External-EBS

Amazon Elastic Block Store (EBS) is a block storage service that allows users to store data on Amazon Web Services (AWS):

- **Persistent data**: EBS stores data on AWS servers even when the EC2 instances are shut down.
- **Scalability**: EBS allows users to scale storage capacity at low subscription-based pricing.
- **High availability**: EBS guarantees 99.999% availability.
- **Encryption**: EBS offers encryption of data at rest using Amazon-managed keys or keys created through Amazon Key Management Service (KMS).
- **Snapshots**: EBS allows users to create point-in-time backups of Amazon EBS volumes.
- **Replication**: EBS automatically replicates every provisioned volume to other storage devices in the same Availability Zone.

EBS is designed to be used with Amazon Elastic Compute Cloud (EC2) instances. Users can attach EBS volumes to EC2 instances and use them like local hard drives

About EBS Volume



These are External Block Storage

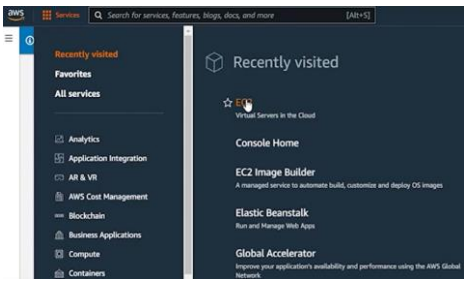
EBS Volume can be attached to only a single ECS Instance at a time.

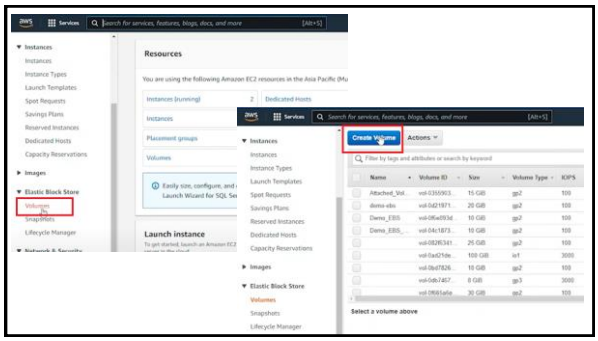
Both EBS Volume and EC2 instance Must be in same availability zone

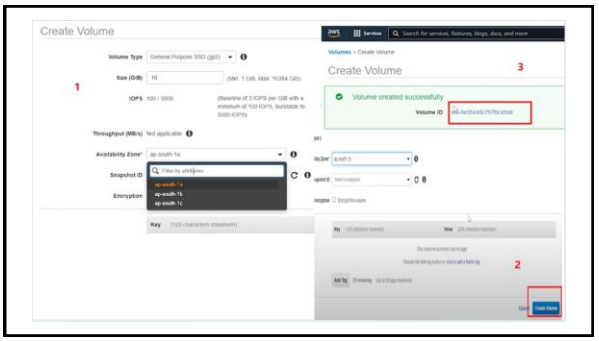
EBS Volumes are replicated by AWS Across multiple server but in the same availability zone.

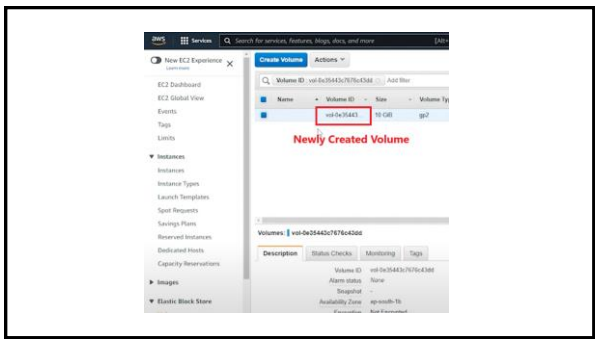
This is to avoid data loss resulting from failure of any AWS component

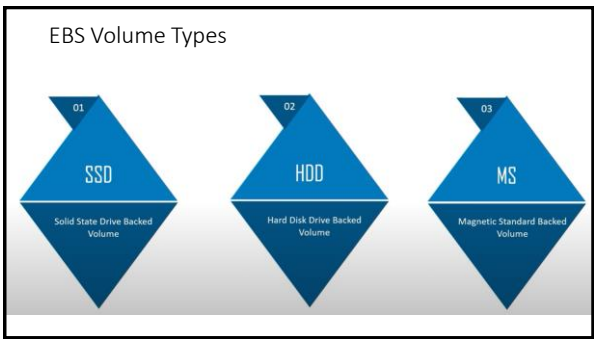
EBS Features

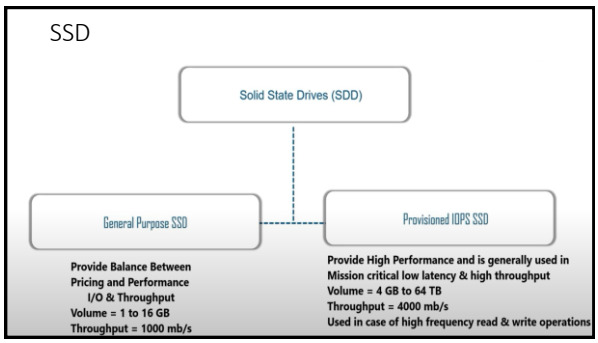


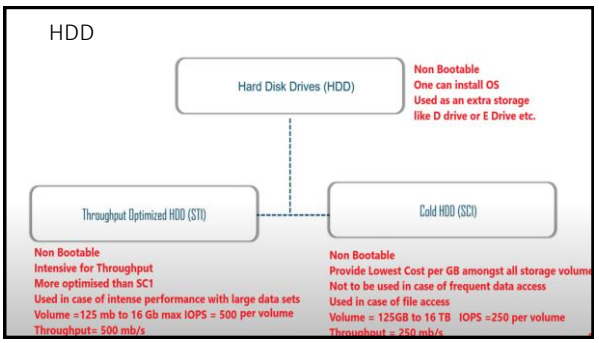

Resources
Launch instance
Create Volume
Actions
Select a volume above


Create Volume
Volume created successfully
1
2
3


Create Volume
Actions
Newly Created Volume


EBS Volume Types
01
SSD
Solid State Drive Backed Volume
02
HDD
Hard Disk Drive Backed Volume
03
MS
Magnetic Standard Backed Volume


SSD
Solid State Drives (SDD)
General Purpose SSD
Provisioned IOPS SSD
Provide Balance Between Pricing and Performance I/O & Throughput
Volume = 1 to 16 GB
Throughput = 1000 mb/s
Provide High Performance and is generally used in Mission critical low latency & high throughput
Volume = 4 GB to 64 TB
Throughput = 4000 mb/s
Used in case of high frequency read & write operations


HDD
Hard Disk Drives (HDD)
Non Bootable
One can install OS
Used as an extra storage like D drive or E Drive etc.
Throughput Optimized HDD (ST1)
Cold HDD (SC1)
Non Bootable
Intensive for Throughput
More optimised than SC1
Used in case of intense performance with large data sets
Volume =125 mb to 16 Gb max IOPS = 500 per volume
Throughput= 500 mb/s
Non Bootable
Provide Lowest Cost per GB amongst all storage volume
Not to be used in case of frequent data access
Used in case of file access
Volume = 125GB to 16 TB IOPS =250 per volume

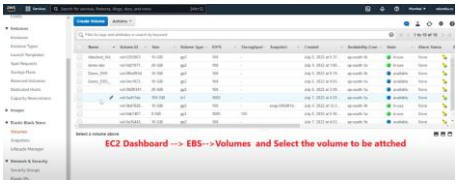
Magnetic Volumes

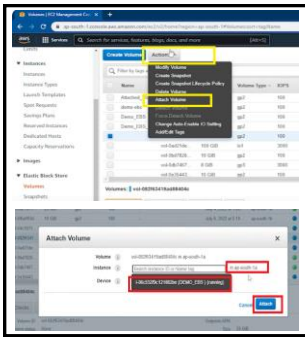
is used when data is to accesses infrequently and storage cost is more important

Volume = 1 GB to 1 TB IOPS = 40 to 200 per volume

Throughput = 40 to 90 mb/s

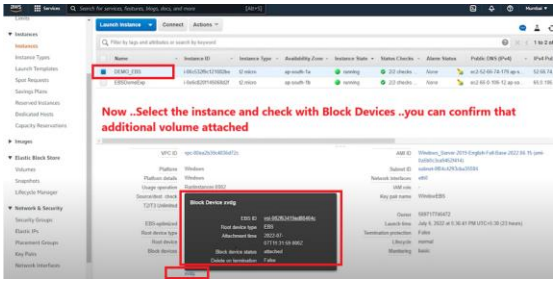
Attach Volume to instance



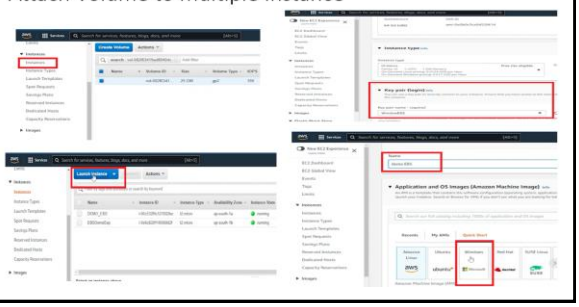


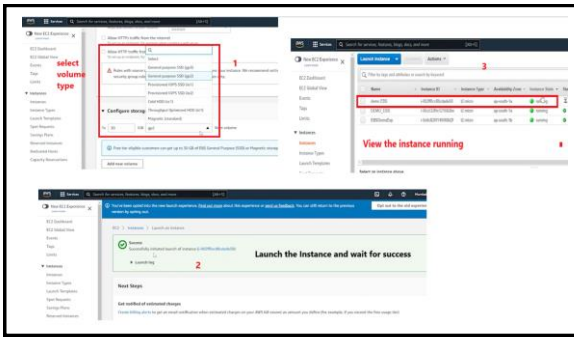
Make Sure that your
Instance and Volume
Are in same Data center
or
Availability ZONE

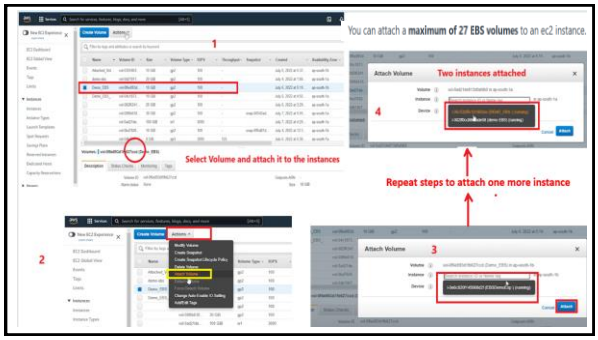
Confirm volume attached with Instance



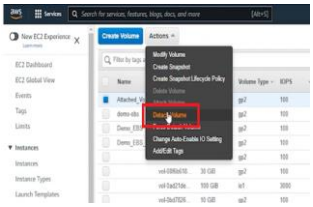
Attach Volume to multiple Instance







Detach and Delete



EBS Snapshot – What can we do with that?


Store or Restore Data


Copy data


Delete Data

You Can Create 5000 Volumes Per AWS Account And can have up to 10000 snapshots Per account

You can back up the data on your Amazon EBS volumes by making point-in-time copies, known as *Amazon EBS snapshots*. A snapshot is an **incremental backup**, which means that we save only the blocks on the volume that have changed since the most recent snapshot. This minimizes the time required to create the snapshot and saves on storage costs by not duplicating data.

DISCLAIMER BY AWS

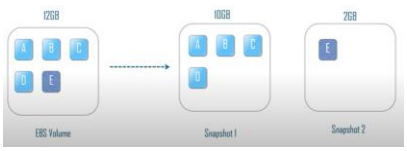
Important

AWS does not automatically back up the data stored on your EBS volumes. For data resiliency and disaster recovery, it is your responsibility to create EBS snapshots on a regular basis, or to set up automatic snapshot creation by using Automate backups with Amazon Data Lifecycle Manager or AWS Backup.

About EBS Snapshots – point in time

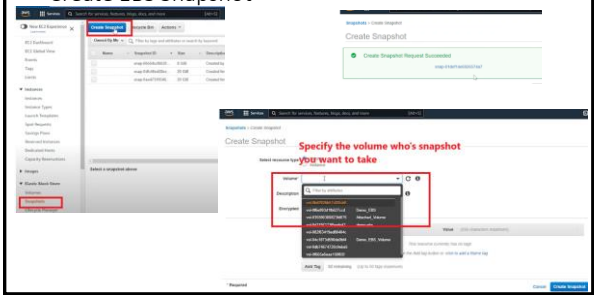
- EBS Snapshots are point-in-time images or copies of your EBS Volume
- Per AWS Account, up-to 5000 Volumes can be created
- Per Account, up-to 10,000 EBS Snapshots can be created
- EBS Volumes are AZ Specific, Snapshots are Region Specific
- EBS Snapshots are stored on S3

Incremental Backup



EBS Snapshot Features

Create EBS Snapshot

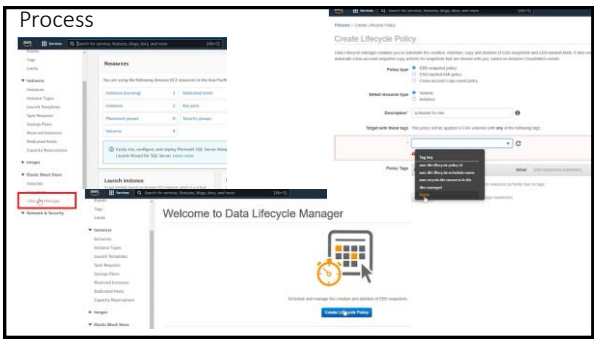


EBS Lifecycle

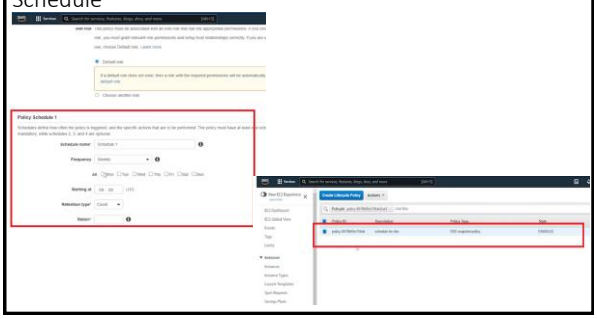
Simple Solution

1,2 & go

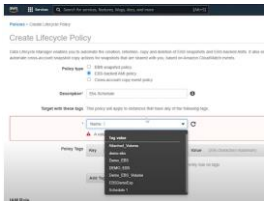
Process



Schedule

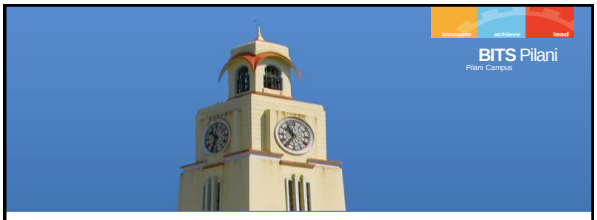


EBS Backed AMI Policy



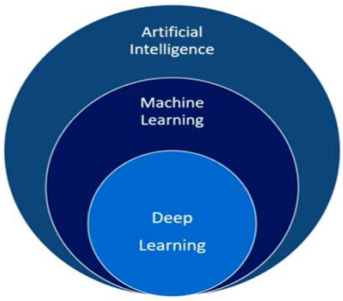
An AMI lifecycle policy in Amazon Elastic Block Store (EBS) is a policy that automates the creation, retention, and deregistration of EBS-backed Amazon Machine Images (AMIs). You can use Amazon Data Lifecycle Manager (DLM) to create and manage these policies.

You can modify as well as delete the Policy



AI/ML/DL

- Intuition of AI vs ML vs DL
- Artificial Intelligence
- Machine Learning
- Deep Learning
- ML vs DL
- Data Science



1.Applied AI(weak AI)- perform some specific tasks.

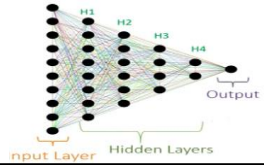
Alexa

Google Assistant

2.Generalized AI(strong AI)- acts like humans.

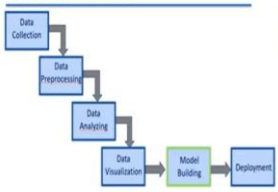
- Machine Learning is a subset of AI.
- Machine Learning is a set of algorithms that train on a data set to make predictions or take actions in order to optimize some systems.

- Deep Learning is a subset of Machine Learning Where learning method is based on data representation or feature learning.
- "Deep" refers to 1 or more hidden layers in this case.
- In Deep Learning data goes through multiple numbers of non-linear transformation obtain an output.

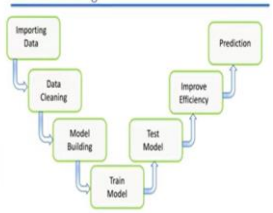


	Machine	Deep
1. Data dependencies	Small Size data	Large Size data
2. Hardware dependencies	Regular Machine	Server Class
3. Feature Engineering	Work with Data	Work with feature
4. Execution Time	Normal Time	Long Time

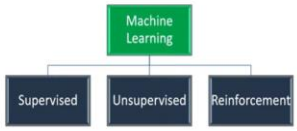
Data Science Flow Chart



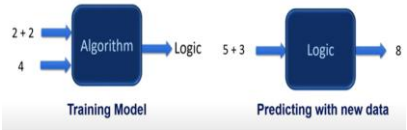
Model Building



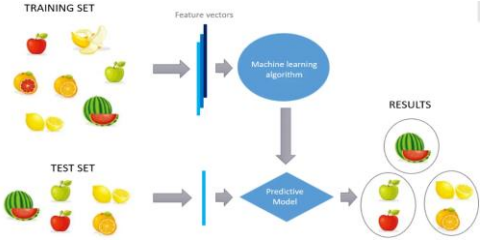
Types of Machine Learning in Artificial Intelligence



Supervised Learning



Unsupervised Learning



Supervised Learning

1. Regression
   -Continuous Value

2. Classification
   -Categorical Value

1. Regression
   -Linear Regression
   -Multiple Linear Regression
   -Polynomial Regressoin
   -Support Vector Regression
   -Decision Tree Regression
   -Random Forest Regression

2. Classification
   -Logistic Regression
   -K-Nearest Neighbors (KNN)
   -Support Vector Machine (SVM)
   -Naïve Bayes
   -Decision Tree Classification
   -Random Forest Classification

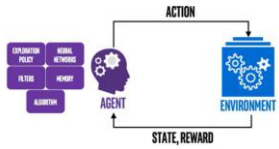
Unsupervised Learning

1. Clustering
   -K-Means Clustering
   -Hierarchical Clustering

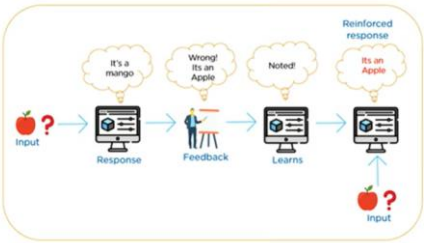
2. Association
   -Apriori
   -Eclat

Reinforcement Learning

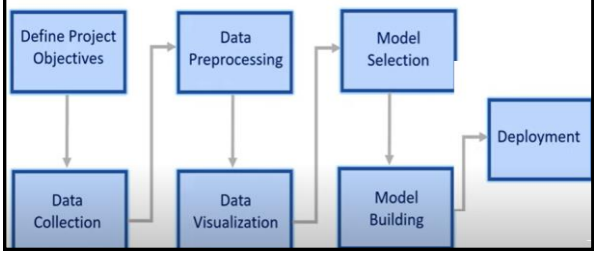
Reinforcement learning is a type of machine learning where an agent learns to behave in a environment by performing actions and seeing the results.



Reinforcement Learning



Machine Learning Life Cycle



Data Collection

- Primary Data – Collected by researcher from first-hand source.
- Secondary Data – Collected by someone else and already been passed through the statistical process.

Data Preprocessing

1. Data Cleaning :
   -Filling Missing Data
   -Smoothing Noisy Data
2. Data Transformation
   -Normalization
3. Dimensionality Reduction

Data Visualization

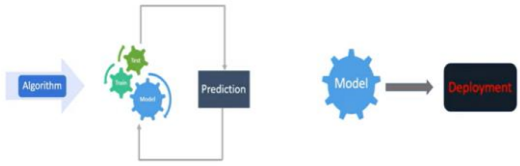
Graphical representation of data.

Model Selection

- Linear Regression
- Logistic Regression
- Random Forest
- K-Means clustering

Model Building

Model Deployment



TOOLS

Here's a list of the top Python libraries for statistical analysis:

1. *NumPy*
2. *SciPy*
3. *Pandas*
4. *StatsModels*

Here's a list of the top Python libraries for data visualization:

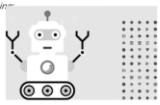
1. *Matplotlib*
2. *Seaborn*
3. *Plotly*
4. *Bokeh*

Here's a list of the top Python libraries for Machine Learning:

1. *Scikit-learn*
2. *XGBoost*
3. *Eli5*

Here's a list of the top Python libraries for Deep Learning:

1. *TensorFlow*
2. *Keras*
3. *Pytorch*



Here's a list of the top Python libraries for NLP:

1. *NLTK*
2. *SpaCy*
3. *Gensim*

Supervised Algo -- K-Nearest Neighbour

Sepal Length	Sepal Width	Species
5.3	3.7	Setosa
5.1	3.8	Setosa
7.2	3.0	Virginica
5.4	3.4	Setosa
5.1	3.3	Setosa
5.4	3.9	Setosa
7.4	2.8	Virginica
6.1	2.8	Verscicolor
7.3	2.9	Virginica
6.0	2.7	Verscicolor
5.8	2.8	Virginica
6.3	2.3	Verscicolor
5.1	2.5	Verscicolor
6.3	2.5	Verscicolor
5.5	2.4	Verscicolor

Sepal Length	Sepal Width	Species
5.2	3.1	?

Step 1: Find Distance

Distance (Sepal Length, Sepal Width) = $\sqrt{(x-a)^2 + (y-b)^2}$

Distance (Sepal Length, Sepal Width) = $\sqrt{(5.2-5.3)^2 + (3.1-3.7)^2}$

Distance (Sepal Length, Sepal Width) = 0.608

Sepal Length	Sepal Width	Species	Distance
5.3	3.7	Setosa	0.608

K-Nearest Neighbour

Sepal Length	Sepal Width	Species	Distance	Rank
5.3	3.7	Setosa	0.608	3
5.1	3.8	Setosa	0.707	6
7.2	3.0	Virginica	2.002	13
5.4	3.4	Setosa	0.36	2
5.1	3.3	Setosa	0.22	1
5.4	3.9	Setosa	0.82	8
7.4	2.8	Virginica	2.22	15
6.1	2.8	Verscicolor	0.94	10
7.3	2.9	Virginica	2.1	14
6.0	2.7	Verscicolor	0.89	9
5.8	2.8	Virginica	0.67	5
6.3	2.3	Verscicolor	1.36	12
5.1	2.5	Verscicolor	0.60	4
6.3	2.5	Verscicolor	1.25	11
5.5	2.4	Verscicolor	0.75	7

Step 2: Find Rank

K-Nearest Neighbour

Sepal Length	Sepal Width	Species	Distance	Rank
5.3	3.7	Setosa	0.608	3
5.1	3.8	Setosa	0.707	6
7.2	3.0	Virginica	2.002	13
5.4	3.4	Setosa	0.36	2
5.1	3.3	Setosa	0.22	1
5.4	3.9	Setosa	0.82	8
7.4	2.8	Virginica	2.22	15
6.1	2.8	Verscicolor	0.94	10
7.3	2.9	Virginica	2.1	14
6.0	2.7	Verscicolor	0.89	9
5.8	2.8	Virginica	0.67	5
6.3	2.3	Verscicolor	1.36	12
5.1	2.5	Verscicolor	0.60	4
6.3	2.5	Verscicolor	1.25	11
5.5	2.4	Verscicolor	0.75	7

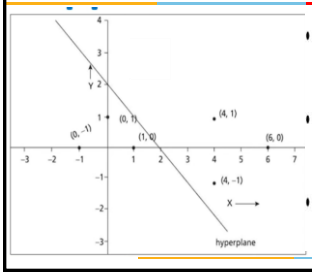
Step 3: Find the Nearest Neighbor

If k = 1 – Setosa

If k = 2 – Setosa

If k = 5 – Setosa

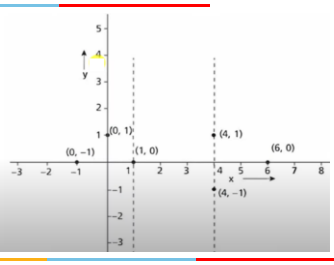
Support Vector Machine



- Points (4, 1), (4, -1) and (6, 0) belong to class positive and
- points (1, 0), (0, 1) and (0, -1) belong to negative class.
- Draw an optimal hyperplane to classify the points.

Support Vector Machine

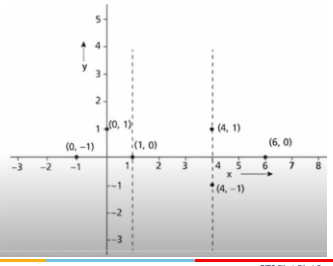
- Points (4, 1), (4, -1) and (6, 0) belong to class positive
- points (1, 0), (0, 1) and (0, -1) belong to negative class.



Support Vector Machine

- It can be observed that the support vectors are (1, 0), (4, 1) and (4, -1)

$$s_1 = \begin{pmatrix}1\\0\end{pmatrix}, s_2 = \begin{pmatrix}4\\1\end{pmatrix}, s_3 = \begin{pmatrix}4\\-1\end{pmatrix}$$



Support Vector Machine

$$s_1 = \begin{pmatrix}1\\0\end{pmatrix}, s_2 = \begin{pmatrix}4\\1\end{pmatrix}, s_3 = \begin{pmatrix}4\\-1\end{pmatrix}$$

- The augmented vector can be obtained by adding the bias given as follows:

$$\tilde{s}_1 = \begin{pmatrix}1\\0\\1\end{pmatrix}, \tilde{s}_2 = \begin{pmatrix}4\\1\\1\end{pmatrix}, \tilde{s}_3 = \begin{pmatrix}4\\-1\\1\end{pmatrix}$$

From these, a set of three equations can be obtained based on these three support vectors as follows:

$$\alpha_1\tilde{s}_1\tilde{s}_1 + \alpha_2\tilde{s}_2\tilde{s}_1 + \alpha_3\tilde{s}_3\tilde{s}_1 = -1$$
$$\alpha_1\tilde{s}_1\tilde{s}_2 + \alpha_2\tilde{s}_2\tilde{s}_2 + \alpha_3\tilde{s}_3\tilde{s}_2 = +1$$
$$\alpha_1\tilde{s}_1\tilde{s}_3 + \alpha_2\tilde{s}_2\tilde{s}_3 + \alpha_3\tilde{s}_3\tilde{s}_3 = +1$$

Support Vector Machine

$$\alpha_1\begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}1\\0\\1\end{pmatrix} + \alpha_2\begin{pmatrix}4\\1\\1\end{pmatrix}\begin{pmatrix}1\\0\\1\end{pmatrix} + \alpha_3\begin{pmatrix}4\\-1\\1\end{pmatrix}\begin{pmatrix}1\\0\\1\end{pmatrix}$$
$$= 2\alpha_1 + 5\alpha_2 + 5\alpha_3 = -1$$

$$\alpha_1\begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}4\\1\\1\end{pmatrix} + \alpha_2\begin{pmatrix}4\\1\\1\end{pmatrix}\begin{pmatrix}4\\1\\1\end{pmatrix} + \alpha_3\begin{pmatrix}4\\-1\\1\end{pmatrix}\begin{pmatrix}4\\1\\1\end{pmatrix}$$
$$= 5\alpha_1 + 18\alpha_2 + 16\alpha_3 = +1$$

$$\alpha_1\begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}4\\-1\\1\end{pmatrix} + \alpha_2\begin{pmatrix}4\\1\\1\end{pmatrix}\begin{pmatrix}4\\-1\\1\end{pmatrix} + \alpha_3\begin{pmatrix}4\\-1\\1\end{pmatrix}\begin{pmatrix}4\\-1\\1\end{pmatrix}$$
$$= 5\alpha_1 + 16\alpha_2 + 18\alpha_3 = +1$$

$$\tilde{s}_1 = \begin{pmatrix}1\\0\\1\end{pmatrix}, \tilde{s}_2 = \begin{pmatrix}4\\1\\1\end{pmatrix}, \tilde{s}_3 = \begin{pmatrix}4\\-1\\1\end{pmatrix}$$

$$\alpha_1\tilde{s}_1\tilde{s}_1 + \alpha_2\tilde{s}_2\tilde{s}_1 + \alpha_3\tilde{s}_3\tilde{s}_1 = -1$$
$$\alpha_1\tilde{s}_1\tilde{s}_2 + \alpha_2\tilde{s}_2\tilde{s}_2 + \alpha_3\tilde{s}_3\tilde{s}_2 = +1$$
$$\alpha_1\tilde{s}_1\tilde{s}_3 + \alpha_2\tilde{s}_2\tilde{s}_3 + \alpha_3\tilde{s}_3\tilde{s}_3 = +1$$

Support Vector Machine

Solving these three simultaneous equations with three unknowns yields the values:

$$\alpha_1 = -3$$
$$\alpha_2 = +1$$
$$\alpha_3 = 0$$

Solving these three simultaneous equations with three unknowns yields the values:

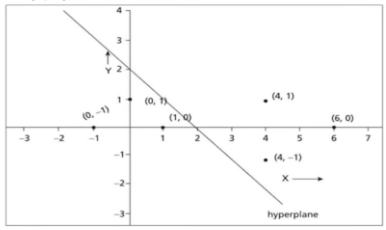
$$\alpha_1 = -3$$
$$\alpha_2 = +1$$
$$\alpha_3 = 0$$

The optimal Hyperplane is given as:

$$w = \sum_i^3 \alpha_i \times \tilde{s}_i$$
$$= -3 \times \begin{pmatrix}1\\0\\1\end{pmatrix} + 1 \times \begin{pmatrix}4\\1\\1\end{pmatrix} + 0 \times \begin{pmatrix}4\\-1\\1\end{pmatrix} = \begin{pmatrix}1\\1\\-2\end{pmatrix}$$

Support Vector Machine

The hyperplane is (1, 1) with an offset -2.



Navie Bayes

Naive Bayes is a probabilistic classifier algorithm that uses Bayes' Theorem to calculate the joint probabilities of values and their attributes within a set of cases. It's a simple, popular algorithm that's used in many industrial applications, including: **Spam filtering**:

Navie Bayes -Example

No.	Color	Type	Origin	Stolen
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

$X = \{Red, SUV, Domestic\}$

$$P(X|y) = \frac{P(y|X) \cdot P(X)}{P(y)}$$

$P(X|Yes) = ?$

$P(X|No) = ?$

Navie Bayes -Example

$$P(Red|Yes) = \frac{P(Yes|Red) \cdot P(Red)}{P(Yes)} = \frac{3/5 \cdot 5/10}{5/10} = \frac{3}{5}$$

$$P(SUV|Yes) = \frac{P(Yes|SUV) \cdot P(SUV)}{P(Yes)} = \frac{\frac{1}{5} \cdot \frac{1}{10}}{\frac{5}{10}} = \frac{1}{5}$$

$$P(Domestic|Yes) = \frac{P(Yes|Domestic) \cdot P(Domestic)}{P(Yes)} = \frac{\frac{2}{5} \cdot \frac{5}{10}}{\frac{5}{10}} = \frac{2}{5}$$

$(\because P + Q = 1 \Rightarrow Q = 1 - P)$

$P(Red|No) = 1 - \frac{3}{5} = \frac{2}{5}$, $P(SUV|No) = 1 - \frac{1}{3} = \frac{4}{5}$

Navie Bayes -Example

$$P(X|Yes) = P(Yes) \cdot P(Red|Yes) \cdot P(SUV|Yes) \cdot P(Domestic|Yes)$$

$$\frac{1}{2} \cdot \frac{3}{5} \cdot \frac{1}{5} \cdot \frac{2}{5} = \frac{3}{125} = 0.024$$

$$P(X|No) = P(No) \cdot P(Red|No) \cdot P(SUV|No) \cdot P(Domestic|No)$$

$$= \frac{1}{2} \cdot \frac{3}{5} \cdot \frac{4}{5} \cdot \frac{3}{5} = \frac{12}{125} = 4 \times 0.024 = 0.096$$

$P(X|No) > P(X|Yes)$

Therefore (No) ✓

Attribute: Outlook

Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$Values(Outlook) = Sunny, Overcast, Rain$

$S = [9+, 5-]$ $\quad Entropy(S) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$

$S_{Sunny} \leftarrow [2+, 3-]$ $\quad Entropy(S_{Sunny}) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971$

$S_{Overcast} \leftarrow [4+, 0-]$ $\quad Entropy(S_{Overcast}) = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0$

$S_{Rain} \leftarrow [3+, 2-]$ $\quad Entropy(S_{Rain}) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971$

$$Gain(S, Outlook) = Entropy(S) - \sum_{v \in \{Sunny, Overcast, Rain\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S, Outlook) = Entropy(S) - \frac{5}{14} Entropy(S_{Sunny}) - \frac{4}{14} Entropy(S_{Overcast}) - \frac{5}{14} Entropy(S_{Rain})$$

$$Gain(S, Outlook) = 0.94 - \frac{5}{14} 0.971 - \frac{4}{14} 0 - \frac{5}{14} 0.971 = 0.2464$$

Decision Tree

Attribute: Temp

Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$Values(Temp) = Hot, Mild, Cool$

$S = [9+, 5-]$ $\quad Entropy(S) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$

$S_{Hot} \leftarrow [2+, 2-]$ $\quad Entropy(S_{Hot}) = -\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4} = 1.0$

$S_{Mild} \leftarrow [4+, 2-]$ $\quad Entropy(S_{Mild}) = -\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6} = 0.9183$

$S_{Cool} \leftarrow [3+, 1-]$ $\quad Entropy(S_{Cool}) = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.8113$

$$Gain(S, Temp) = Entropy(S) - \sum_{v \in \{Hot, Mild, Cool\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S, Temp) = Entropy(S) - \frac{4}{14} Entropy(S_{Hot}) - \frac{6}{14} Entropy(S_{Mild}) - \frac{4}{14} Entropy(S_{Cool})$$

$$Gain(S, Temp) = 0.94 - \frac{4}{14} 1.0 - \frac{6}{14} 0.9183 - \frac{4}{14} 0.8113 = 0.0289$$

Decision Tree

Attribute: Humidity

Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$Values(Humidity) = High, Normal$

$S = [9+, 5-]$ $\quad Entropy(S) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$

$S_{High} \leftarrow [3+, 4-]$ $\quad Entropy(S_{High}) = -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} = 0.9852$

$S_{Normal} \leftarrow [6+, 1-]$ $\quad Entropy(S_{Normal}) = -\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7} = 0.5916$

$$Gain(S, Humidity) = Entropy(S) - \sum_{v \in \{High, Normal\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S, Humidity) = Entropy(S) - \frac{7}{14} Entropy(S_{High}) - \frac{7}{14} Entropy(S_{Normal})$$

$$Gain(S, Humidity) = 0.94 - \frac{7}{14} 0.9852 - \frac{7}{14} 0.5916 = 0.1516$$

Decision Tree

Attribute: Wind

Values (Wind) = Strong, Weak

Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$S = [9+, 5-]$ $\quad Entropy(S) = -\frac{9}{14}log_2\frac{9}{14} - \frac{5}{14}log_2\frac{5}{14} = 0.94$

$S_{Strong} \leftarrow [3+, 3-]$ $\quad Entropy(S_{Strong}) = 1.0$

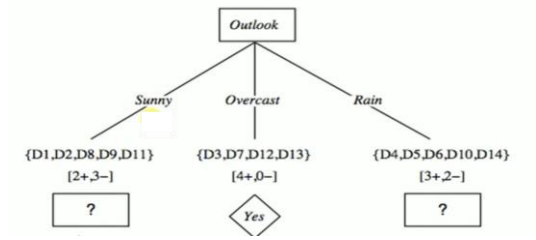
$S_{Weak} \leftarrow [6+, 2-]$ $\quad Entropy(S_{Weak}) = -\frac{6}{8}log_2\frac{6}{8} - \frac{2}{8}log_2\frac{2}{8} = 0.8113$

$$Gain\,(S, Wind) = Entropy(S) - \sum_{v \in \{Strong, Weak\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S, Wind) = Entropy(S) - \frac{6}{14} Entropy(S_{Strong}) - \frac{8}{14} Entropy(S_{Weak})$$

$$Gain(S, Wind) = 0.94 - \frac{6}{14}\,1.0 - \frac{8}{14}\,0.8113 = 0.0478$$

Decision Tree



Decision Tree

Day	Temp	Humidity	Wind	Play Tennis
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

Attribute: Temp

Values (Temp) = Hot, Mild, Cool

$S_{Sunny} = [2+, 3-]$ $\quad Entropy(S_{Sunny}) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.97$

$S_{Hot} \leftarrow [0+, 2-]$ $\quad Entropy(S_{Hot}) = 0.0$

$S_{Mild} \leftarrow [1+, 1-]$ $\quad Entropy(S_{Mild}) = 1.0$

$S_{Cool} \leftarrow [1+, 0-]$ $\quad Entropy(S_{Cool}) = 0.0$

$$Gain\,(S_{Sunny}, Temp) = Entropy(S) - \sum_{v \in \{Hot, Mild, Cool\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Sunny}, Temp) = Entropy(S) - \frac{2}{5} Entropy(S_{Hot}) - \frac{2}{5} Entropy(S_{Mild}) - \frac{1}{5} Entropy(S_{Cool})$$

$$Gain(S_{Sunny}, Temp) = 0.97 - \frac{2}{5}\,0.0 - \frac{2}{5}\,1 - \frac{1}{5}\,0.0 = 0.570$$

Decision Tree

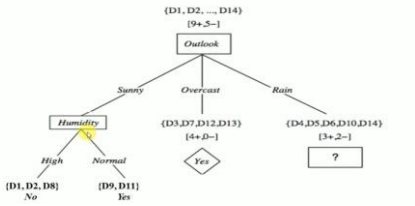
Day	Temp	Humidity	Wind	Play Tennis
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

$Gain(S_{sunny}, Temp) = 0.570$

$Gain(S_{sunny}, Humidity) = 0.97$

$Gain(S_{sunny}, Wind) = 0.0192$

Decision Tree



Regression

Regression analysis is a set of statistical methods used for the estimation of relationships between a dependent variable and one or more independent variables. It can be utilized to assess the strength of the relationship between variables and for modeling the future relationship between them.

Logistic Regression

- The dataset of pass or fail in an exam of 5 students is given in the table.
- Use logistic regression as classifier to answer the following questions.

1. Calculate the probability of pass for the student who studied 33 hours.
2. At least how many hours student should study that makes he will pass the course with the probability of more than 95%.

Hours Study	Pass (1) / Fail (0)
29	0
15	0
33	1
28	1
39	1

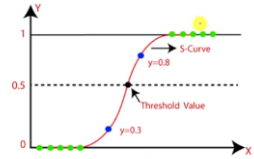
Assume the model suggested by the optimizer for odds of passing the course is,

$$\log(odds) = -64 + 2 * hours$$

Logistic Regression

- We use Sigmoid Function in logistic regression

- $s(x) = \frac{1}{1+e^{-x}}$



Logistic Regression

1. Calculate the probability of pass for the student who studied 33 hours.

- $p = \frac{1}{1+e^{-z}}$ $\qquad s(x) = \frac{1}{1+e^{-x}}$
- $z = -64 + 2 * 33 = -64 + 66 = 2$
- $p = \frac{1}{1+e^{-2}} = 0.88$
- That is, if student studies 33 hours, then there is **88% chance** that the student will pass the exam

Hours Study	Pass (1) / Fail (0)
29	0
15	0
33	1
28	1
39	1

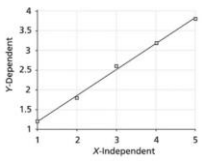
$$\log(odds) = z = -64 + 2 * hours$$

Linear Regression

- Let us consider an example where the five weeks' sales data (in Thousands) is given as shown in Table.
- Apply linear regression technique to predict the 7th and 12th week sales.

x_i (Week)	y_i (Sales in Thousands)
1	1.2
2	1.8
3	2.6
4	3.2
5	3.8

Linear Regression



x_i (Week)	y_i (Sales in Thousands)
1	1.2
2	1.8
3	2.6
4	3.2
5	3.8

Linear Regression

- Linear regression equation is given by
- $y = a_0 + a_1 * x + e$
- *where*
- $a_1 = \frac{(\bar{x}\bar{y}) - (\bar{x})(\bar{y})}{\bar{x^2} - \bar{x}^2}$
- $a_0 = \bar{y} - a_1 * \bar{x}$

x_i (Week)	y_i (Sales in Thousands)
1	1.2
2	1.8
3	2.6
4	3.2
5	3.8

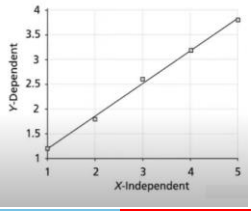
Linear Regression

- Here, there are 5 items, i.e., i = 1, 2, 3, 4, 5.

	x_i (Week)	y_i (Sales in Thousands)	x_i^2	$x_i * y_i$
	1	1.2	1	1.2
	2	1.8	4	3.6
	3	2.6	9	7.8
	4	3.2	16	12.8
	5	3.8	25	19
Sum	15	12.6	55	44.4
Average	$\bar{x} = 3$	$\bar{y} = 2.52$	$\overline{x^2} = 11$	$\overline{xy} = 8.88$

Linear Regression

- $\bar{x} = 3$ $\bar{y} = 2.52$ $x^2 = 11$ $\overline{xy} = 8.88$
- $a_1 = \frac{(\overline{xy})-(\bar{x})(\bar{y})}{\overline{x^2}-\bar{x}^2} = \frac{8.88-3*2.52}{11-3^2} = 0.66$
- $a_0 = \bar{y} - a_1 * \bar{x} = 2.52 - 0.66 * 3 = 0.54$
- **Regression equation is**
- $y = a_0 + a_1 * x$
- $y = 0.54 + 0.66 * x$



Linear Regression

- **Regression equation is**
- $y = a_0 + a_1 * x$
- $y = 0.54 + 0.66 * x$
- The predicted 7th week sale (when x = 7) is,
- **y = 0.54 + 0.66 x 7 = 5.16**
- the predicted 12th week sale (when x = 12) is,
- **y = 0.54 + 0.66 x 12 = 8.46**

Multi Linear Regression

- In linear regression model we have one dependent and one independent variable.
- Multiple regression model involves multiple predictors or independent variables and one dependent variable.
- This is an extension of the linear regression problem.

Multi Linear Regression

- The multiple regression of two variables x1 and x2 is given as follows:

$$y = f(x_1, x_2)$$

$$y = a_0 + a_1 x_1 + a_2 x_2$$

- In general, this is given for 'n' independent variables as:

$$y = f(x_1, x_2, \ldots . x_n)$$

$$y = a_0 + a_1 x_1 + a_2 x_2 + \ldots \ldots + a_n x_n + \varepsilon$$

- Here, $x_1, x_2, \ldots . x_n$ are predictor variables, y is the dependent variable, $(a_0, a_1, a_2, \ldots . a_n)$ are the coefficients of the regression equation and ε is the error term.

Multi Linear Regression

- Here, the matrices for Y and X are given as follows:

$$X = \begin{pmatrix} 1 & 1 & 4 \\ 1 & 2 & 5 \\ 1 & 3 & 8 \\ 1 & 4 & 2 \end{pmatrix} \text{ and } Y = \begin{pmatrix} 1 \\ 6 \\ 8 \\ 12 \end{pmatrix}$$

- The coefficient of the multiple regression equation is given as

$$a = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix}$$

x1 Product 1 Sales	x2 Product 2 Sales	Y Weekly Sales
1	4	1
2	5	6
3	8	8
4	2	12

Multi Linear Regression

- The regression coefficient for multiple regression is calculated the same way as linear regression:

$$\hat{a} = ((X^T X)^{-1} X^T) Y$$

x1 Product 1 Sales	x2 Product 2 Sales	Y Weekly Sales
1	4	1
2	5	6
3	8	8
4	2	12

Multi Linear Regression

- The regression coefficient for multiple regression is calculated the same way as linear regression:

$$\hat{a} = ((X^T X)^{-1} X^T) Y$$

x1 Product 1 Sales	x2 Product 2 Sales	Y Weekly Sales
1	4	1
2	5	6
3	8	8
4	2	12

$$X^T X = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 4 & 5 & 8 & 2 \end{pmatrix} \begin{pmatrix} 1 & 1 & 4 \\ 1 & 2 & 5 \\ 1 & 3 & 8 \\ 1 & 4 & 2 \end{pmatrix} = \begin{pmatrix} 4 & 10 & 19 \\ 10 & 30 & 46 \\ 19 & 46 & 109 \end{pmatrix}$$

Multi Linear Regression

$$X^T X)^{-1} X^T = \begin{pmatrix} 3.15 & -0.59 & -0.30 \\ -0.59 & 0.20 & 0.016 \\ -0.30 & 0.016 & 0.054 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 4 & 5 & 8 & 2 \end{pmatrix} = \begin{pmatrix} 0.05 & 0.47 & -1.02 & 0.19 \\ -0.32 & -0.098 & 0.155 & 0.26 \\ -0.065 & 0.005 & 0.185 & -0.125 \end{pmatrix}$$

$$\hat{a} = ((X^T X)^{-1} X^T) Y = \begin{pmatrix} 0.05 & 0.47 & -1.02 & 0.19 \\ -0.32 & -0.098 & 0.155 & 0.26 \\ -0.065 & 0.005 & 0.185 & -0.125 \end{pmatrix} \times \begin{pmatrix} 1 \\ 6 \\ 8 \\ 12 \end{pmatrix} = \begin{pmatrix} -1.69 \\ 3.48 \\ -0.05 \end{pmatrix}$$

Multi Linear Regression

$$a_0 = -1.69$$
$$a_1 = 3.48$$
$$a_2 = -0.05$$

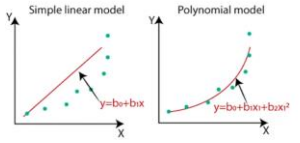
x1 Product 1 Sales	x2 Product 2 Sales	Y Weekly Sales
1	4	1
2	5	6
3	8	8
4	2	12

- $y = a_0 + a_1 x_1 + a_2 x_2$
- Hence, the constructed model is:
- $y = -1.69 + 3.48 x_1 - 0.05 x_2$

X1 & X2 ----Independent
Y -- Dependent

Polynomial Regression

- If the relationship between the independent and dependent variables is not linear, then linear regression cannot be used as it will result in large errors.



- The problem of non-linear regression can be solved by two methods:
  1. Transformation of non-linear data to linear data, so that the linea regression can handle the data
  2. Using polynomial regression

Transformations

- The trick is to convert non-linear data to linear data that can be handled using the linear regression method.
- Let us consider an exponential function $y = ae^{bx}$.
- The transformation can be done by applying log function to both sides to get:

$$\ln(y) = \ln(a) + \ln(e^{bx})$$
$$\ln(y) = \ln(a) + bx * \ln(e)$$
$$\ln(y) = \ln(a) + bx$$

Polynomial Regression

- It can handle non-linear relationships among variables by using n^{th} degree of a polynomial.
- Instead of applying transformations, polynomial regression can be directly used to deal with different levels of curvilinearity.
- For example, the second-degree polynomial (called quadratic transformation) is given as: $y = a_0 + a_1x + a_2x^2$ and third degree polynomial is called cubic transformation given as: $y = a_0 + a_1x + a_2x^2 + a_3x^3$
- Generally, polynomials of maximum degree 4 are used, as higher order polynomials take some strange shapes and make the curve more flexible.
- It leads to a situation of overfitting and hence is avoided.

- Consider the polynomial of 2nd degree.
- The polynomial equation in given by $y = a_0 + a_1x + a_2x^2$
- The coefficients a_0, a_1 *and* a_2 are calculated using the formula,

$$a = X^{-1}B$$

- Where,

$$X = \begin{bmatrix} n & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} \quad B = \begin{bmatrix} \sum y_i \\ \sum (x_i, y_i) \\ \sum (x_i^2, y_i) \end{bmatrix}$$

x_i	y_i	x_iy_i	x_i^2	x_i^2y	x_i^3	x_i^4
1	1	1	1	1	1	1
2	4	8	4	16	8	16
3	9	27	9	81	27	81
4	15	60	16	240	64	256
$\sum x_i = 10$	$\sum y_i = 29$	$\sum x_iy_i = 96$	$\sum x_i^2 = 30$	$\sum x_i^2y_i = 338$	$\sum x_i^3 = 100$	$\sum x_i^4 = 354$

$$a = X^{-1}B \quad X = \begin{bmatrix} n & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} \quad B = \begin{bmatrix} \sum y_i \\ \sum (x_i, y_i) \\ \sum (x_i^2, y_i) \end{bmatrix} \quad \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 4 & 10 & 30 \\ 10 & 30 & 100 \\ 30 & 100 & 354 \end{bmatrix}^{-1} \times \begin{bmatrix} 29 \\ 96 \\ 338 \end{bmatrix} = \begin{pmatrix} -0.75 \\ 0.95 \\ 0.75 \end{pmatrix}$$

$$a = X^{-1}B \quad \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 4 & 10 & 30 \\ 10 & 30 & 100 \\ 30 & 100 & 354 \end{bmatrix}^{-1} \times \begin{bmatrix} 29 \\ 96 \\ 338 \end{bmatrix} = \begin{pmatrix} -0.75 \\ 0.95 \\ 0.75 \end{pmatrix} \quad y = -0.75 + 0.95\,x + 0.75\,x^2$$

Unsupervised algo -- Clustering

Clustering is an unsupervised machine learning technique designed to group **unlabeled examples** based on their similarity to each other. (If the examples are labeled, this kind of grouping is called **classification**.) Consider a hypothetical patient study designed to evaluate a new treatment protocol.

K-Means

- Suppose that the data mining task is to cluster points into three clusters,
- where the points are
- *A1*(2, 10), *A2*(2, 5), *A3*(8, 4), *B1*(5, 8), *B2*(7, 5), *B3*(6, 4), *C1*(1, 2), *C2*(4, 9).
- The distance function is Euclidean distance.
- Suppose initially we assign *A1*, *B1*, and *C1* as the center of each cluster, respectively.

K-Means

Initial Centroids:
A1: (2, 10)
B1: (5, 8)
C1: (1, 2)

Data Points			Distance to 2, 10	Distance to 5, 8	Distance to 1, 2	Cluster	New Cluster
A1	2	10	0.00	3.61	8.06	1	
A2	2	5	5.00	4.24	3.16	3	
A3	8	4	8.49	5.00	7.28	2	
B1	5	8	3.61	0.00	7.21	2	
B2	7	5	7.07	3.61	6.71	2	
B3	6	4	7.21	4.12	5.39	2	
C1	1	2	8.06	7.21	0.00	3	
C2	4	9	2.24	1.41	7.62	2	

$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

K-Means

Current Centroids:
A1: (2, 10)
B1: (6, 6)
C1: (1.5, 3.5)

New Centroids:
A1: (3, 9.5)
B1: (6.5, 5.25)
C1: (1.5, 3.5)

Data Points			Distance to 2, 10	Distance to 6, 6	Distance to 1.5, 1.5	Cluster	New Cluster
A1	2	10	0.00	5.66	6.52	1	1
A2	2	5	5.00	4.12	1.58	3	3
A3	8	4	8.49	2.83	6.52	2	2
B1	5	8	3.61	2.24	5.70	2	2
B2	7	5	7.07	1.41	5.70	2	2
B3	6	4	7.21	2.00	4.53	2	2
C1	1	2	8.06	6.40	1.58	3	3
C2	4	9	2.24	3.61	6.04	2	1

$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

K-Means

Current Centroids:
A1: (3, 9.5)
B1: (6.5, 5.25)
C1: (1.5, 3.5)

New Centroids:
A1: (3.67, 9)
B1: (7, 4.33)
C1: (1.5, 3.5)

Data Points			Distance to 3, 9.5	Distance to 6.5, 5.25	Distance to 1.5, 3.5	Cluster	New Cluster
A1	2	10	1.12	6.54	6.52	1	1
A2	2	5	4.61	4.51	1.58	3	3
A3	8	4	7.43	1.95	6.52	2	2
B1	5	8	2.50	3.13	5.70	2	1
B2	7	5	6.02	0.56	5.70	2	2
B3	6	4	6.26	1.35	4.53	2	2
C1	1	2	7.76	6.39	1.58	3	3
C2	4	9	1.12	4.51	6.04	1	1

$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

K-Means

Current Centroids:
A1: (3.67, 9)
B1: (7, 4.33)
C1: (1.5, 3.5)

Data Points			Distance to 3.67, 9	Distance to 7, 4.33	Distance to 1.5, 3.5	Cluster	New Cluster
A1	2	10	1.94	7.56	6.52	1	1
A2	2	5	4.33	5.04	1.58	3	3
A3	8	4	6.62	1.05	6.52	2	2
B1	5	8	1.67	4.18	5.70	1	1
B2	7	5	5.21	0.67	5.70	2	2
B3	6	4	5.52	1.05	4.53	2	2
C1	1	2	7.49	6.44	1.58	3	3
C2	4	9	0.33	5.55	6.04	1	1

$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Agglomerative Hierarchical Clustering

Agglomerative Hierarchical Clustering Solved Example

- Consider the following set of 6 one dimensional data points:
- 18, 22, 25, 42, 27, 43
- Apply the **agglomerative hierarchical clustering** algorithm to build the hierarchical clustering **dendogram**.
- Merge the clusters using **Min distance** and update the proximity matrix accordingly.
- Clearly show the **proximity matrix** corresponding to each iteration of the algorithm.

- Step – 1

	18	22	25	27	42	43
18	0	4	7	9	24	25
22	4	0	3	5	20	21
25	7	3	0	2	17	18
27	9	5	2	0	15	16
42	24	20	17	15	0	1
43	25	21	18	16	1	0

(42, 43)

- Step – 2

	18	22	25	27	42, 43
18	0	4	7	9	24
22	4	0	3	5	20
25	7	3	0	2	17
27	9	5	2	0	15
42, 43	24	20	17	15	0

- Step – 2

	18	22	25	27	42, 43
18	0	4	7	9	24
22	4	0	3	5	20
25	7	3	0	2	17
27	9	5	2	0	15
42, 43	24	20	17	15	0

(42, 43), (25, 27) • Step – 3

	18	22	25, 27	42, 43
18	0	4	7	24
22	4	0	3	20
25, 27	7	3	0	15
42, 43	24	20	15	0

- Step – 3

	18	22	25, 27	42, 43
18	0	4	7	24
22	4	0	3	20
25, 27	7	3	0	15
42, 43	24	20	15	0

(42, 43), ((25, 27), 22) • Step – 4

	18	22, 25, 27	42, 43
18	0	4	24
22, 25, 27	4	0	15
42, 43	24	15	0

- Step – 4

	18	22, 25, 27	42, 43
18	0	4	24
22, 25, 27	4	0	15
42, 43	24	15	0

(42, 43), (((25, 27), 22), 18) • Step – 5

	18, 22, 25, 27	42, 43
18, 22, 25, 27	0	15
42, 43	15	0

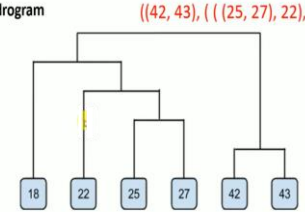
Final step

- Step – 6

	18, 22, 25, 27, 42, 43
18, 22, 25, 27, 42, 43	0

Dendrogram

- Dendrogram ((42, 43), (((25, 27), 22), 18))



A dendrogram illustrates how similar objects are related to each other. The branches of the tree represent categories, or classes, of objects that share similar

Association

Association rule learning is a type of unsupervised learning technique that checks for the dependency of one data item on another data item and maps accordingly

Apriory

Transaction ID	Items Bought
1	{Bread, Butter, Milk}
2	{Bread, Butter}
3	{Beer, Cookies, Diapers}
4	{Milk, Diapers, Bread, Butter}
5	{Beer, Diapers}

Generate candidate itemsets (Ck) and qualified frequent itemsets (Lk) step by step until the largest frequent itemset is generated.

1-Itemset	Support_count
Bread	3
Butter	3
Milk	2
Beer	2
Cookies	1
Diapers	3

Bread, Butter, Milk, Diapers, Beer

Minimum Support Count

min_support = 40%,

min_support_count

= min_support x itemset_count

= 40% x 5

= 2

1-Frequent Itemset	Support_count
Bread	3
Butter	3
Diapers	3
Milk	2
Beer	2

T ID	Items Bought
1	{Bread, Butter, Milk}
2	{Bread, Butter}
3	{Beer, Cookies, Diapers}
4	{Milk, Diapers, Bread, Butter}
5	{Beer, Diapers}

2-Itemset	Support_count
Bread, Butter	3
Bread, Diapers	1
Bread, Milk	2
Bread, Beer	0
Butter, Diapers	1
Butter, Milk	2
Butter, Beer	0
Diapers, Milk	1
Diapers, Beer	2
Milk, Beer	0

2-Frequent Itemset	Support_count
Bread, Butter	3
Bread, Milk	2
Butter, Milk	2
Diapers, Beer	2

T ID	Items Bought
1	{Bread, Butter, Milk}
2	{Bread, Butter}
3	{Beer, Cookies, Diapers}
4	{Milk, Diapers, Bread, Butter}
5	{Beer, Diapers}

Bread, Butter, Milk, Diapers, Beer

3-Itemset	Support_count
Bread, Butter, Milk	2
Bread, Butter, Diapers	1
Bread, Butter, Beer	0
Bread, Milk, Diapers	1
Bread, Milk, Beer	0
Bread, Diapers, Beer	0
Butter, Milk, Diapers	1
Butter, Milk, Beer	0
Butter, Diapers, Beer	0
Milk, Diapers, Beer	0

3-Frequent Itemset	Support_count
Bread, Butter, Milk	2

1-Frequent Itemset	Support_count
Bread	3
Butter	3
Diapers	3
Milk	2
Beer	2

2-Frequent Itemset	Support_count
Bread, Butter	3
Bread, Milk	2
Butter, Milk	2
Diapers, Beer	2

3-Frequent Itemset	Support_count
Bread, Butter, Milk	2

- min_confidence = 70%
- Confidence (X→Y) = P(Y | X) = P(X ∪ Y) / P(X)
- We have 5 frequent itemsets:
- {Bread, Butter}, {Bread, Milk}, {Butter, Milk}, {Diapers, Beer} and {Bread, Butter, Milk}.
- Therefore, candidate rules are:
- For {Bread, Butter},
  - bread>butter = 3/3 = 100% (Strong)
  - butter>bread = 3/3 = 100% (Strong)
- For {Bread, Milk}
  - bread>milk = 2/3 = 67%
  - milk>bread = 2/2 =100% (Strong)

- For {Bread, Butter, Milk}
  - bread,butter>milk = 2/3 = 67% ✗
  - bread,milk>butter = 2/2 = 100% (Strong)
  - milk,butter>bread = 2/2 = 100% (Strong)
  - bread>butter,milk = 2/3 = 67% ✗
  - butter>bread,milk = 2/3 = 67% ✗
  - milk>bread,butter = 2/2=100% (Strong)

- Therefore, candidate rules are:
- For {Butter, Milk}
  - butter>milk = 2/3 = 67% ✗
  - milk>butter = 2/2 = 100% (Strong)

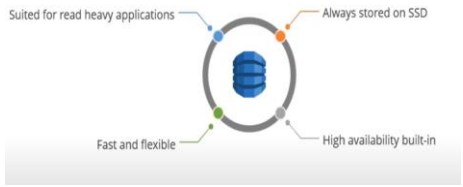
Why DynamoDB Why?

- ✓ DynamoDB is a key:value store of the NoSQL family developed and offered by Amazon as part of AWS
- ✓ High Scale, High Performance & Fully Managed DB Service
- ✓ Accessible via Web Service APIs
- ✓ Provides speed, Scalability & Ease of use
- ✓ Takes care of hardware or software provisioning, setup and configuration, software patching, operating a reliable, distributed database cluster, or partitioning data over multiple instances as you scale.

Features

- ✓ Scalable
- ✓ Fast, Predictable Performance
- ✓ Easy Administration
- ✓ Built-in Fault Tolerance
- ✓ Flexible
- ✓ Strong Consistency, Atomic Counters
- ✓ Integrated Monitoring
- ✓ Elastic MapReduce Integration
- ✓ Secure

Benefits



Concept Of DynamoDB

- ✓ **The Amazon DynamoDB data model concepts include tables, items and attributes.**
- ✓ **In Amazon DynamoDB, a database is a collection of tables. A table is a collection of items and each item is a collection of attribute**
- ✓ **DynamoDB allows several "tables", where a record ("Item"), is identified by one or two "Attributes", named:**
  - ✓ A "Hash Key", which works as a Primary Key
  - ✓ Optionally, a "Range Key", which lets you build Composite Keys
- ✓ Beside the Key Attributes, everything else is **unstructured**.
- ✓ **For the Keys, there are three data types:** String, Binary, Number (anything with up to 38 significant digits and between 10^-128 and 10^126).
- ✓ You can also have a **"Set" datatype** (for String, Binary, and Numbers), though they are not indexed.

Tables & Forums with Unique Subject

Table Name	Primary Key Type	Hash Attribute Name and Type	Range Attribute Name and Type
ProductCatalog (Id, ...)	Hash	Attribute Name: Id Type: Number	-
Forum (Name, ...)	Hash	Attribute Name: Name Type: String	-
Thread (ForumName, Subject, ...)	Hash and Range	Attribute Name: ForumName Type: String	Attribute Name: Subject Type: String
Reply (Id, ReplyDateTime, ...)	Hash and Range	Attribute Name: Id Type: String	Attribute Name: ReplyDateTime Type: String

- ✓ The ProductCatalog table represents a table in which each product item is uniquely identified by an Id
- ✓ The Forum, Thread, and Reply tables are modeled after the AWS forums. Each AWS service maintains one or more forums. Customers start a thread by posting a message that has a unique subject. Each thread might receive one or more replies at different times. These replies are stored in the Reply table

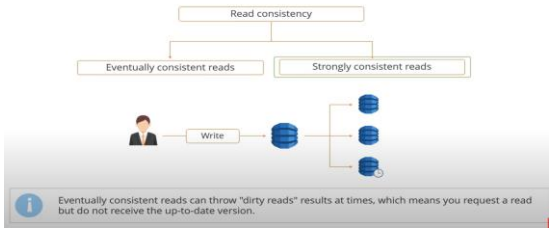
DynamoDB Concept

- ✓ **The following operations are possible:**
  - ✓ PutItem / GetItem / DeleteItem, where you DynamoDB Record (or "Item") directly, provided you have the HashKey (and Range Key). Except the latter, there are "Batch" variants available
- ✓ **Query**, where you can look up Items via Conditions based on the Primary Key Attributes.
- ✓ **Scan**, where a whole table is scanned.

DynamoDB Capacity to Manage…

- ✓ **Hardware provisioning**
- ✓ **Cross-availability zone replication**
- ✓ **Monitoring and handling of hardware failures**
  - ✓ Replicas automatically regenerated whenever necessary
- ✓ **Changing the level of provisioned throughput**
  - ✓ Data might need to be redistributed around the cluster
  - ✓ No service disruption or performance impact

Read Consistency



DynamoDB Modes

DynamoDB Provisioned Mode

You define some capacity, and DynamoDB provisions that capacity for you. This is pretty similar to provisioning an Auto Scaling Group of EC2 instances, but imagine the size of the instance is fixed, and it's one group for reads and another one for writes. Here's how that capacity translates into actual read and write operations.

Capacity in Provisioned Mode

Capacity is provisioned separately for reads and writes, and it's measured in Capacity Units.

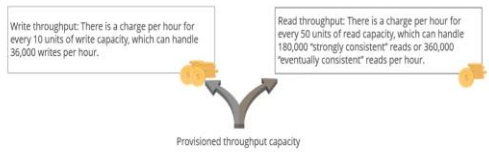
1 Read Capacity Unit (RCU) is equivalent to 1 strongly consistent read of up to 4 KB, per second. Eventually consistent reads consume half that capacity. Reads over 4 KB consume 1 RCU (1/2 for eventually consistent) per 4 KB, rounded up. That means if you have 5 RCUs, you can perform 10 eventually consistent reads every second, or 2 strongly consistent reads for 7 KB of data each (remember it's rounded up) plus 1 strongly consistent read for 1 KB of data (again, it's rounded up).

Write Capacity Units (WCU) work the same, but for writes. **1 WCU = 1 write per second, of up to 1 KB**. So, with 5 WCUs, you can perform 1 write operation per second of 4.5 KB, or 5 writes of less than 1 KB.

DynamoDB Read & Write

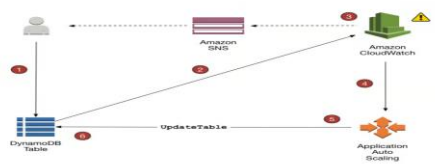
- ✓ A unit of Write Capacity enables you to perform one write per second for items of up to 1Kb in size
- ✓ a unit of Read Capacity enables you to perform one strongly consistent read per second (or two eventually consistent reads per second) of items of up to 1Kb in size. Larger items will require more capacity.
  - Units of Capacity required for writes = Number of item writes per second x item size (rounded up to the nearest KB)
  - Units of Capacity required for reads* = Number of item reads per second x item size (rounded up to the nearest KB)
- ✓ If your items are less than 1KB in size, then each unit of Read Capacity will give you 1 read/second of capacity and each unit of Write Capacity will give you 1 write/second of capacity.
  - ✓ For example, if your items are 512 bytes and you need to read 100 items per second from your table, then you need to provision 100 units of Read Capacity.
- ✓ If your items are larger than 1KB in size, then you should calculate the number of units of Read Capacity and Write Capacity that you need.

Provisioned Throughput Capacity



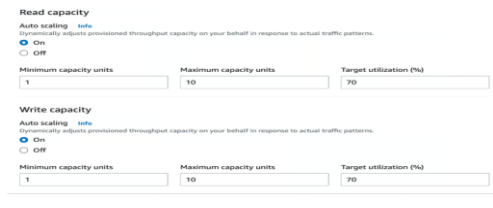
Scaling Provision Mode

This is the real scaling. DynamoDB tables continuously send metrics to CloudWatch. CloudWatch triggers alarms when those metrics cross a certain threshold. DynamoDB gets notified about that and modifies Capacity Units accordingly.



On DynamoDB you enable Auto Scaling, set a minimum and maximum capacity units, and set a target utilization (%). You can enable scaling separately for Reads and Writes.

Table Capacity



In the table metrics (handled by CloudWatch) you can view provisioned and consumed capacity, and throttled request count.

Capacity in On-Demand Mode

The cost of reads and writes stays the same: a read operation consumes 1 Read Request Unit (RRU) for every 4 KB read (half if it's eventually consistent), and a write operation consumes 1 WRU (Write Request Unit) for every 1 KB written.

Here's the difference: There is no capacity you can set. You're billed for every actual operation, and DynamoDB manages capacity automatically and transparently. However, it does have a set capacity, it does scale, and understanding how it does is important.

Scaling in On-Demand Mode

Every newly-created table in On-Demand mode starts with 4.000 WCUs and 12.000 RCUs (yeah, that's a lot). You're not billed for those capacity units though; you'll only be billed for actual operations.

Every time your peak usage goes over 50% of the current assigned capacity, DynamoDB increases the capacity of that table to double your peak. So, suppose you used 5.000 WRUs, now your table's WCUs are 10.000. This growth has a cooldown period of 30 minutes, meaning it won't happen again until 30 minutes after the last increase.

WCU stands for Write Capacity Unit, which is a unit of measurement for write requests to a table in Amazon DynamoDB. One WCU allows for one write request per second for items up to 1 KB in size. For items larger than 1 KB, additional WCUs are required

RCU stands for "Read Capacity Unit". It represents the number of times per second that data can be read from a table

Switching from Provisioned Mode to On-Demand Mode

You can switch modes in either direction, but you can only do so once every 24 hours. If you switch from Provisioned Mode to On-Demand mode, the table's initial RCUs are the maximum of 12.000, your current RCUs, or double the units of the highest peak. Same for WCUs, the maximum between 4.000, your current WCUs, or double the units of the highest peak.

If you switch from On-Demand mode to Provisioned Mode, you need to set up your capacity or auto scaling manually.

In either case the switch takes up to 30 minutes, during which the table continues to function like before the switch.

Apache Spark

Apache Spark ..Why?

What is Apache Spark ?

Apache Spark is a Unified Processing Engine and Set of Libraries for Parallel Database Processing on Computer Cluster

Unified

Spark is designed to support wide variety of tasks over the same computing engine

Ex: Data Scientist as well as Data Engineers both can use same platform for their analysis, transformation and modelling.

Engineers: Data Analysis
Scientists: Modelling and Prediction

Computing Engine

Spark is purely computing engine. It does not store any data

Spark can Connect with different data sources
Ex: HDFS, JDBC/ODBC, AZURE..etc

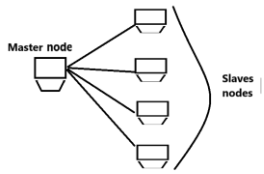
Spark works with almost all data storage systems.

Libraries

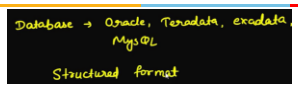
Spark had ready to use libraries
Spark SQL
Spark Streams
ML Lib
..etc

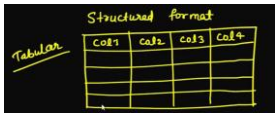
Computer Cluster for Parallel Processing

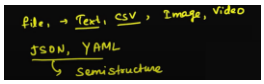
Spark works on computer cluster



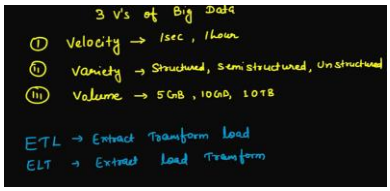
Why Spark?



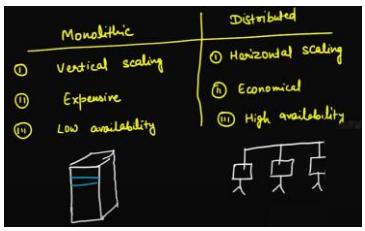




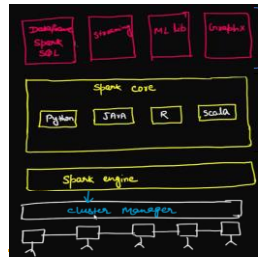
Issues



Approaches



Architecture

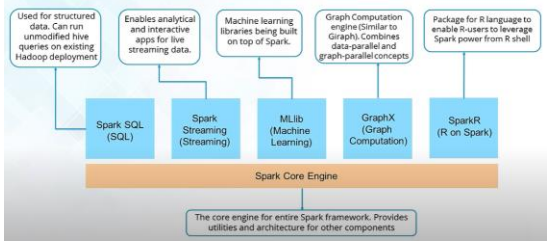


Librarie
High Level API
Dataframe/Dataset

Low Level API
RDD
SPARK CORE

Ex: YARN, MESOS, Kubernetes

Another view



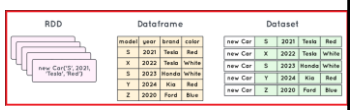
RDD V/S DataFrame V/S DataSet

Spark RDD stands for **Resilient Distributed Dataset** which is the core data abstraction API and is available since very first release of Spark (**Spark 1.0**).

It is a lower-level API for manipulating distributed collection of data. The RDD APIs exposes some extremely useful methods which can be used to get very tight control over underlying physical data structure.

It is an immutable (read only) collection of partitioned data distributed on different machines. RDD enables in-memory computation on large clusters to speed up big data processing in a fault tolerant manner.

Feature	RDD	DataFrame	DataSet
Immutable	Yes	Yes	Yes
Fault tolerant	Yes	Yes	Yes
Type-safe	Yes	No	Yes
Schema	No	Yes	Yes
Execution optimization	No	Yes	Yes
Level	Low	High	High



RDD

1. **Immutable collection:** RDD is an immutable partitioned collection distributed on different nodes. A partition is a basic unit of parallelism in Spark. The immutability helps to achieve fault tolerance and consistency.
2. **Distributed data:** RDD is a collection of distributed data which helps in big data processing by distributing the workload to different nodes in the cluster.
3. **Lazy evaluation:** The defined transformations do not gets evaluated until an action is called. It helps Spark in optimizing the overall transformations in one go.
4. **Fault tolerant:** RDD can be recomputed in case of any failure using DAG(Directed acyclic graph) of transformations defined for that RDD.
5. **Multi-language support:** RDD APIs supports **Python, R, Scala, and Java** programming languages.

Limitation : No optimization engine: RDD does not have an in-built optimization engine.

DataFrames

Spark 1.3 introduced two new data abstraction APIs – **DataFrame and DataSet**. The DataFrame APIs organizes the data into named columns like a table in relational database. It enables programmers to define schema on a distributed collection of data. Each row in a DataFrame is of object type row. Like an SQL table, each column must have same number of rows in a DataFrame. In short, DataFrame is lazily evaluated plan which specifies the operations needs to be performed on the distributed collection of the data. DataFrame is also an immutable collection.

Below are the features

1. **In-built Optimization:** When an action is called on a DataFrame, the Catalyst engine analyzes the code and resolves the references. Then, it creates a logical plan. After that, the created logical plan gets translated into an optimized physical plan. Finally, this physical plan gets executed on the cluster.
2. **Hive compatible:** The DataFrame is fully compatible with Hive query language. We can access all hive data, queries, UDFs, etc using Spark SQL from hive MetaStore and can execute queries against these hive databases.
3. **Structured, semi-structured, and highly structured data support:** DataFrame APIs supports manipulation of all kind of data from structured data files to semi-structured data files and highly structured parquet files.
4. **Multi-language support:** DataFrame APIs are available in **Python, R, Scala, and Java**.
5. **Schema support:** We can define a schema manually or we can read a schema from a data source which defines the column names and their data types.

Limitation: **Type safety:** Each row in a DataFrame is of object type row and hence is not strictly typed. That is why DataFrame does not support compile time safety.

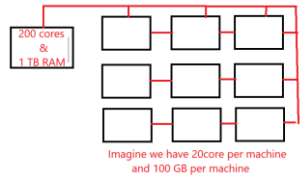
DataSet

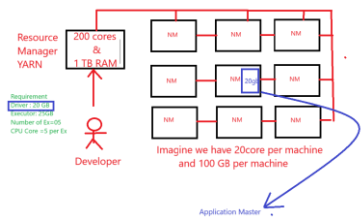
As an extension to the DataFrame APIs, **Spark 1.3** also introduced DataSet APIs which provides strictly typed and object-oriented programming interface in Spark. It is immutable, type-safe collection of distributed data. Like DataFrame, DataSet APIs also uses Catalyst engine in order to enable execution optimization. DataSet is an extension to the DataFrame APIs. Features and limitations of the DataSet are as below:

Features:
1.Combination of RDD and DataFrame: DataSet enables functional programming like RDD APIs and relational queries and execution optimization like DataFrame APIs. Thus, it provides the benefit of best of both worlds – RDDs and DataFrames.
2.Type-safe: Unlike DataFrames, DataSet APIs provides compile time type safety. It conforms the specification at compile time using defined case classes (for Scala) or Java beans (for Java).

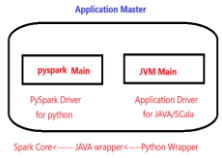
Limitations:
1.Limited language support: DataSet is only available to JVM based langauges like Java and Scala. Python and R do not support DataSet because these are dynamically typed languages.
2.High garbage collection: JVM types can cause high garbage collection and object instantiation cost.

How it Works





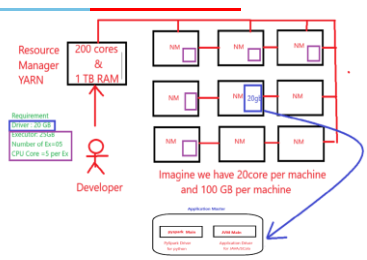
Application Master



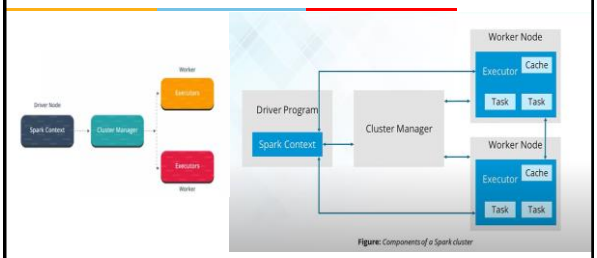
Executors and Driver

"driver" is the central coordinating process that manages the overall execution of a Spark application,

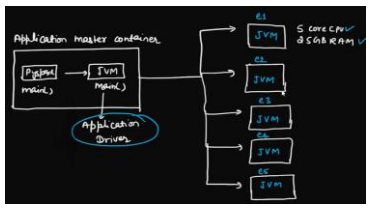
"executors" are distributed worker processes running on different nodes in the cluster that actually perform the computation tasks assigned by the driver; essentially, the driver is the brain that directs the work, and the executors are the workers who do the actual processing on the data.



Another View



Cont.…

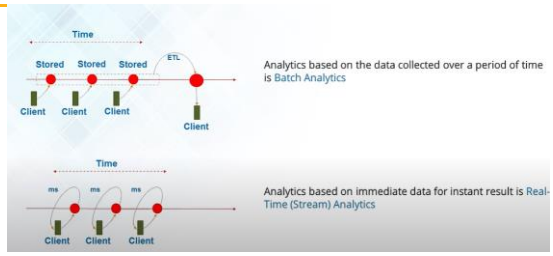


compare

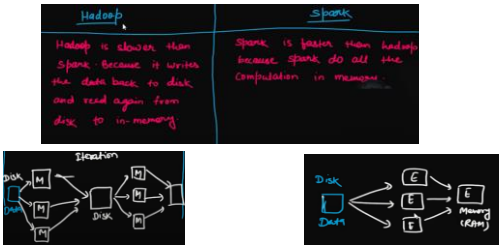
Hadoop is Build for Batch Data Processing
Difficult to write code in Hadoop…Hive was built as easier alternative

Spark is build for Stream data processing
Very Easy to write and debug the code
Spark provide low and high level API

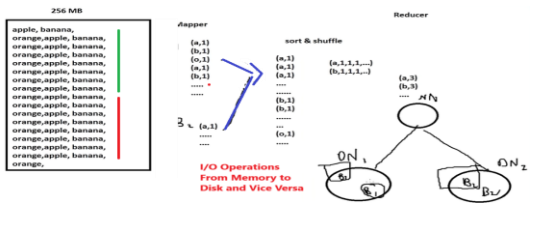
Batch and Real Time Processing



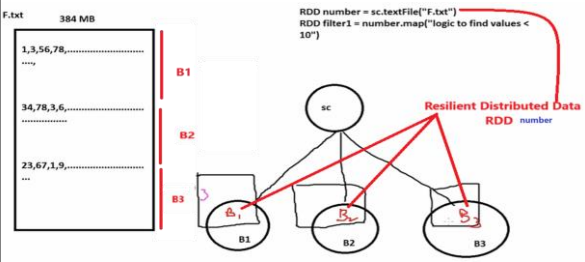
Compare



Slow operations –in Map Reduce



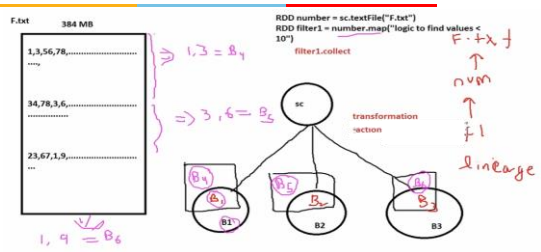
Faster processing of Spark



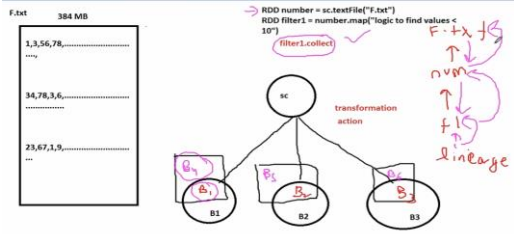
Hadoop has Data stored in blocks
It replicated these blocks to handle failure

Spark use DAG to provide Fault tolerance

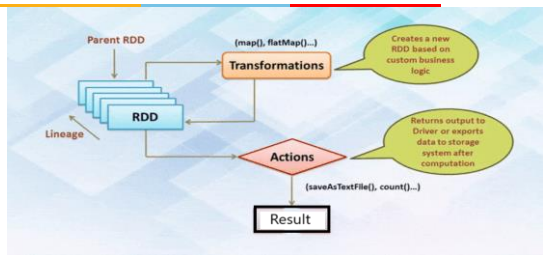
Filter..Collect..transform…action



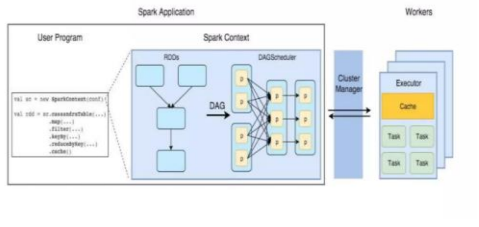
Lazy Evaluation - Lineage



Spark Lazy Evaluation



Another View



Some Actions - Count

In PySpark, actions are operations that trigger the execution of a computation on a DataFrame and return a result to the driver program. Here's a breakdown of the actions you mentioned:

count():

- **Purpose:** Returns the number of rows in a DataFrame.
- **Example:**

Python

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()
df = spark.createDataFrame([(1, "Alice"), (2, "Bob")], ["id", "name"])

row_count = df.count()
print(row_count)  # Output: 2
```

Some Actions - Show

show():

- **Purpose:** Displays the first few rows of a DataFrame in a tabular format.
- **Example:**

Python

```
df.show()
# Output:
# +---+-----+
# | id| name|
# +---+-----+
# |  1|Alice|
# |  2|  Bob|
# +---+-----+
```


Some Actions - collect

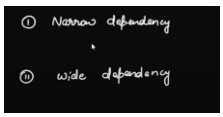
collect():

- **Purpose:** Retrieves all the rows of a DataFrame as a list of Row objects on the driver program.
- **Caution:** Use with caution on large datasets, as it can cause memory issues on the driver.
- **Example:**

Python

```
data = df.collect()
print(data)  # Output: [Row(id=1, name='Alice'), Row(id=2, name='Bob')]
```


Transformation Types



•**Narrow transformations**

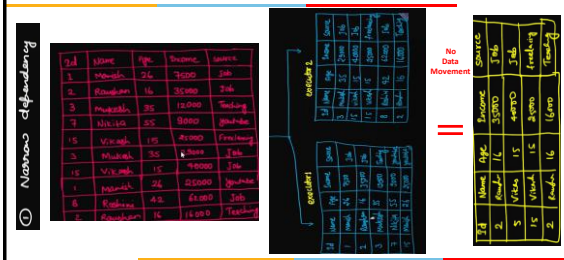
•Each input partition is used to compute one output partition. These transformations are preferred because they are more efficient and require less data movement. Examples of narrow transformations include map(), filter(), and union().

•**Wide transformations**

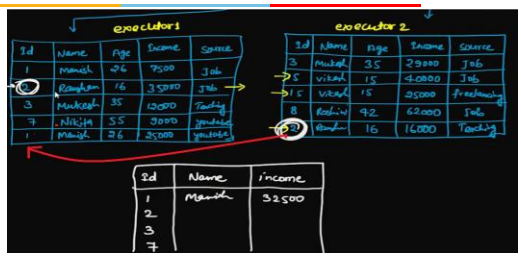
•Each input partition is used to compute multiple output partitions. These transformations are more resource-intensive and time-consuming than narrow transformations, especially when dealing with large datasets. Wide transformations may change the number of partitions in the output RDD or DataFrame.

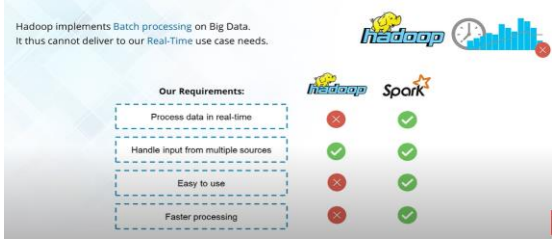
The choice between narrow and wide transformations is important for optimizing performance and resource utilization. Developers can use this knowledge to design jobs that execute efficiently across the cluster.

Show name of employee with age less than 18?
Find total income of each employee



Wide transformation—Costly affair





Compare

Hadoop Use : Kerberos for Authentication
ACL for Authorization

Spark: Do not have its own strong security features

On having access to HDFS it gains ACL controls
On having access to YARN Spark gains Kerberos Authentication credentials

Spark Gels well

Some Misconceptions

OUTLINE

- **SPARK & ITS FEATURE**
- **SPARK ARCHITECTURE**
- **RESILIENT DISTRIBUTED DATASETS(RDDs)**
- **DIRECT ACYCLIC GRAPH(DAG)**
- **ADVANTAGES & DRAWBACKS**
- **CONCLUSION**

INTRODUCTION

- Apache Spark : an open source cluster computing framework for real-time data processing
- According to ***Spark Certified Experts***: Sparks performance is up to 100 times faster in memory and 10 times faster on disk when compared to Hadoop
- The main feature of Apache Spark is its ***in-memory cluster computing*** that increases the processing speed of an application

FEATURES OF APACHE SPARK

FEATURES OF APACHE SPARK

- **Speed:**
  Spark runs up to 100 times faster than Hadoop MapReduce for large-scale data processing
- **Powerful Caching:**
  Simple programming layer provides powerful caching and disk persistence capabilities.
- **Deployment:**
  It can be deployed through ***Mesos, Hadoop via YARN, or Spark's own cluster manager***

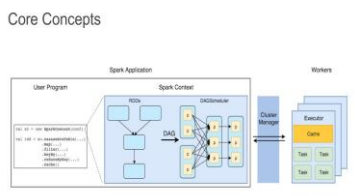
FEATURES OF APACHE SPARK

- **Real-Time:**
  It offers Real-time computation & low latency because of ***in-memory computation***
- **Polyglot:**
  Spark provides high-level APIs in Java, Scala, Python, and R. Spark code can be written in any of these four languages. It also provides a shell in Scala and Python

SPARK ARCHITECTURE

Figure:-Apache spark architecture

CORE CONCEPTS



SPARK ARCHITECTURE

- **SPARK DRIVE :-**
- Separate process to execute user application
- Creates SparkContext to schedual
- Jobs execution & negotiate with cluster manager
- **EXECUTORS :-**
- Run tasks scheduled by driver
- Store computation result in memory,on disk or off-heap
- Interact with storage systems

SPARK ARCHITECTURE

- **CLUSTER MANAGER :-**
- Spark context works with the ***cluster manager*** to manage various jobs
- The driver program & Spark context takes care of the job execution within the cluster

SPARK ARCHITECTURE

- Apache Spark Architecture is based on two main abstractions:
- *Resilient Distributed Dataset (RDD)*
- *Directed Acyclic Graph (DAG)*

RDD

- Simple view
  - RDD is collection of data items split into partitions and stored in memory on worker nodes of the cluster
- Complex view
  - RDD is an interface for data transformation
  - RDD refers to the data stored either in persisted store (HDFS, Cassandra, HBase, etc.) or in cache (memory, memory+disks, disk only, etc.) or in another RDD

Resilient Distributed Dataset (RDD)

RDD

- Complex view (cont'd)
  - Partitions are recomputed on failure or cache eviction
  - Metadata stored for interface
    - *Partitions* – set of data splits associated with this RDD
    - *Dependencies* – list of parent RDDs involved in computation
    - *Compute* – function to compute partition of the RDD given the parent partitions from the *Dependencies*
    - *Preferred Locations* – where is the best place to put computations on this partition (data locality)
    - *Partitioner* – how the data is split into partitions

Resilient Distributed Dataset (RDD)

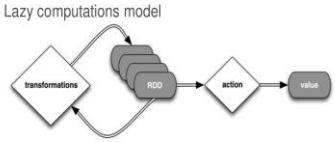
RDD

- RDD is the main and only tool for data manipulation in Spark
- Two classes of operations
  - Transformations
  - Actions

Resilient Distributed Dataset (RDD)

RDD

Lazy computations model



Transformation cause only metadata change

OPERATION OF RDD:-

- RDDs can perform two types of operations:
- **Transformations:** They are the operations that are applied to create a new RDD.
- **Actions:** They are applied on an RDD to instruct Apache Spark to apply computation and pass the result back to the driver.

DIRECT ACYCLIC GRAPH(DAG)

DAG

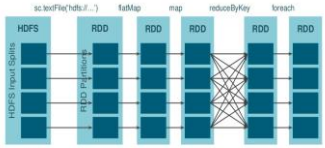
Direct Acyclic Graph – sequence of computations performed on data

- *Node* – RDD partition
- *Edge* – transformation on top of data
- *Acyclic* – graph cannot return to the older partition
- *Direct* – transformation is an action that transitions data partition state (from A to B)

DIRECT ACYCLIC GRAPH(DAG)

DAG

WordCount example



ADVANTAGES & drawbacks

- **ADVANTAGES:**
- Integration with Hadoop
- Faster
- Real time stream processing
- **DRAWBACKS:**
- No File Management system
- No Support for Real-Time Processing
- Cost Effective
- Manual Optimization

Conclusion

- SPARK makes it easy to write and run complicated data processing
- It enables computation of tasks at a very large scale
- Although spark has many limitations, it is still trending in the big data world
- Due to these drawbacks, many technologies are overtaking Spark
- Such as Flink offers complete real-time processing than the spark
- In this way somehow other technologies overcoming the drawbacks of Spark