Big Data [26 Aug24]

26 August 2024 09:18

Introduction to big data

What's big data?

Data that has high

- Volume
  - size
- Variety
  - Structured
  - Semi structured
  - Quasi
  - unstructured
- Velocity
  - speed
- Veracity
  - Truthfulness of the data
  - Chances of inaccurate predictions
- Value
  - Data getting collected must provide information that adds value to the company

Why store so much data?

- To analyse and fetch results

Industry	Uses
Banking	- Transaction - Credit cards - Credit score - Investments, insurance
Healthcare	- Images (xray, mri, ctscan) - Ecg, eeg - Personalized diagnosis - Fitness routine - Early disease prediction
Energy	- EV & Hybrid vehicles (forecasting the requirements) - EV Buses
Technology	Advancement
Consumer	- Forecasting product demand - Managing logistics
Manufacturing	- Logistics - Warehouse and store locations

Challenges of traditional decision making

- Took long time to make a decision
- Required human intervention
- Lacked systematic linkage
- Limited scope of data analysis
- Withheld company's ability to make fully informed decisions
- Limitation of storage and cost cutting

After big data analytics

- Decision making is based on what you know
- Provides a comprehensive view of overall picture after analysing from various sources
- Streamlined and faster decision making
  - Improves competitive advantage
  - Faster process
  - Various tool used (framework)
- Using unstructured data

Types of data

Name	What is it	examples
Unstructured	- Data has no inherent structure - Usually stored as different types of files	- Text docs - Pdf - Images - videos
Quasi structured	- Textual data with erratic formats - can be formatted with effort and software tools	- Clickstream data
Semi structured	Textual data files with an apparent pattern	- Json files - Xml files - Csv files - Tsv files
Structured	Data having defined data model (tabular format), format	Database

Clickstream data

Data that is used to record a user's interactions with a website application. It includes

- Page views
- Click events
- Navigation paths
- Time stamps
- Session info
- User ID
- Device and browser info
- Geolocation
- Referrer
- Engagement metrics

		- Xml files - Csv files - Tsv files
Structured	Data having defined data model (tabular format), format, structure	- Database - excel

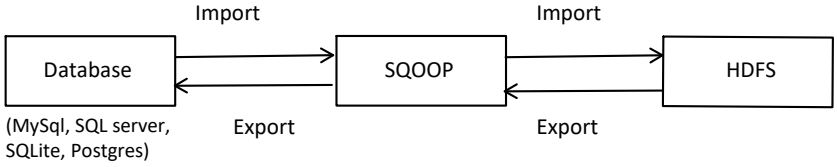
- Device and browser info
- Geolocation
- Referrer
- Engagement metrics

Case study: use of big data in Netflix

When do users watch a show	To find the most active time
Where do they watch it	Country location
On which device do they watch	Subscription packages
How often do they pause a show	How interesting is the show
How often do they rewatch a show	Re-watchability
Do they skip credits	
What are the keywords searched	To find a show easily

Big data analytics pipeline

- Data sources
- Data ingestion layer
  - Sqoop
  - Kafka
  - Flume
- Data collection layer
- Data storage layer
  - Hadoop
  - Aws
- Data processing layer
  - Batch
  - Realtime
  - Hybrid
- Data query layer
  - Spark
- Analytics
  - Spark (not used for storage)
- Data visualization
  - databricks



SQOOP

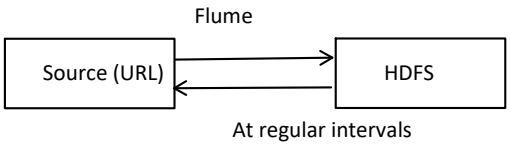
Import and export data from source (structured source) to destination (HDFS)

HDFS

- Hadoop distributed file system
- Storage system

Kafka

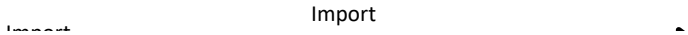
- Temporary storage



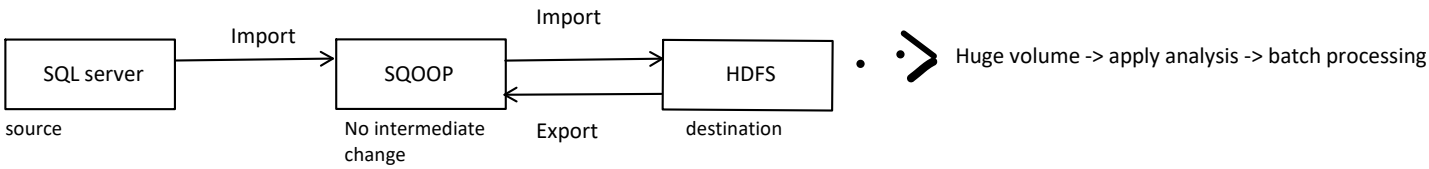
Flume

- Also used for migrating data from URL (live streaming source) and collect it to store in HDFS

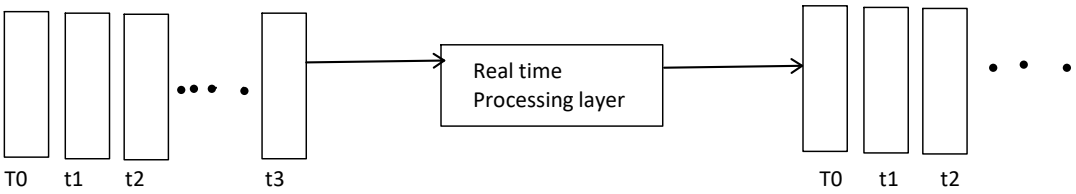
Batch processing



Batch processing



Real time processing



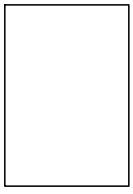
Features of big data

Distributed system in big data

Explained in more detail below, scroll down

Traditional vs distributed machine

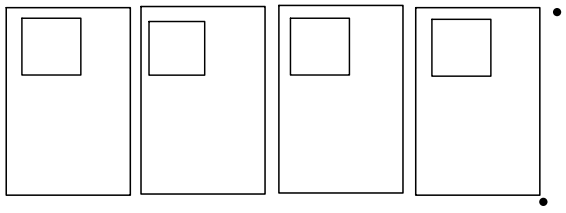
Traditional



- 1 TB storage
- Processing speed = 100 mb/sec
- 4 channel I/O

Time approx = 43.6 min

Distributed



Commodity hardware

- Processing speed = 100 mb/sec
- Number of channels = 4 I/O channel

Time approx = 43.6/4
= 10.9 min

Then if there were 100 machines, the time would reduce to 1/100th of 43.6 mins
= approx 25 seconds

How to handle increased load?

Vertical scaling	Horizontal scaling
- Increasing capacity of single system to handle more load - Achieved by adding more resources to the machine like ○ CPU ○ RAM ○ Storage	- Adding more servers/instances to handle increased load - Achieved by ○ Adding servers ○ Load balancing ○ Fault tolerance

Scalability in big data

Aka adding/removing resources to/from distributed system

- Scalable platform accommodates rapid changes in growth of data (in traffic or volume)
- Utilizes and adds hardware/software to increase the output (throughput) and storage of data
- When the platform of a company is scalable, it is prepared for the potential growth

Fault tolerance in big data

Can it handle faults?

- It refers to working strength of a system in unfavourable conditions and how that system can handle that situation
- Ex. Replication factor in HDFS (replicating a node n times)

Data inconsistency in big data

Monitoring every change in the data

- Once data is captured, inconsistent or conflicting phenomena can occur at various granularities
- It can occur from
    - Knowledge content
    - Data information
    - Knowledge meta knowledge
    - Expertise
- Can adversely affect outcome of analysis

What's a distributed system?

Model in which components located on networked computers communicate and coordinate by passing message.

Challenges of distributed system:

- System failure
- Limited bandwidth
- High programming complexity

*Any solution for this? (Yes, its **Hadoop**)*

Hadoop & HDFS Theory [26 Aug 24]

26 August 2024 15:07

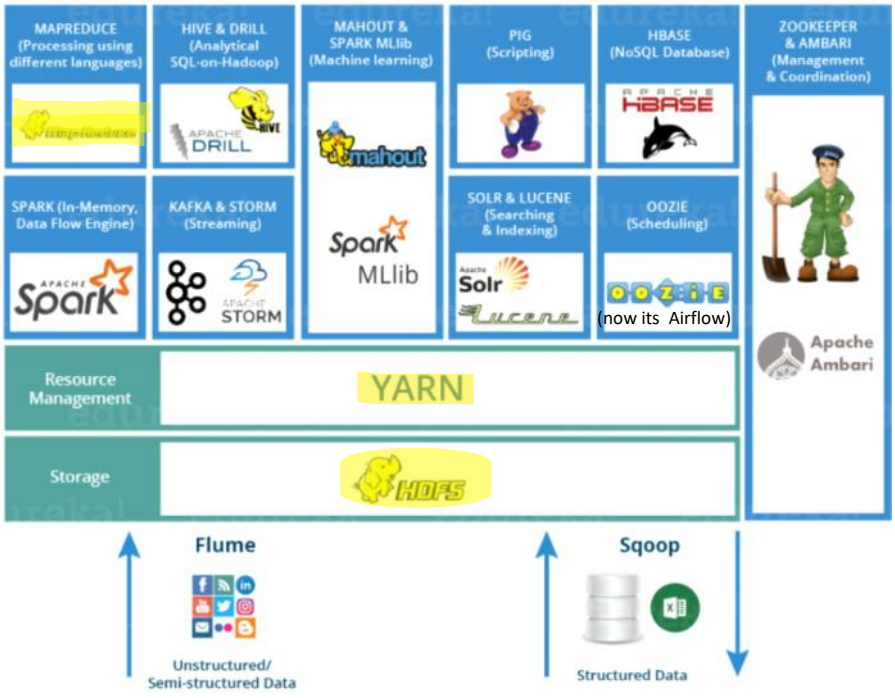
Hadoop

What's Hadoop?

Framework that allows distributed processing of large datasets across clusters of commodity computers using simple programming models

3 main components that will be discussed

- Storage (HDFS - Hadoop Distributed File System)
- Programming methodology (MapReduce)
- Resource Management while processing data (YARN - Yet Another Resource Negotiator)



Characteristics

Feature	As in
Scalable	Both horizontal and vertical scaling possible
Reliable	Highly available (copies)
Economical	Can use ordinary computers for processing
flexible	Can store huge data and keep to use for later

Traditional systems vs Hadoop

factor	*Traditional systems (RDBMS)*	*Hadoop*
flow	Data sent to program	Program sent to data
Datatypes	Structured	All kinds
Processing (cleaning the data)	Limited, no data processing	Processing coupled with data
Governance	Structured	Loosely structured
Schema	Required on write	Required on read
Speed	Fast reads (read many write many)	Fast writes (read many write once)
Cost	Software license	Support only

Read many write once

Many people can read at a time
BUT only one person can write at a time

Resources	Known entity	Growing
Best fit use	- Oltp (online transaction processing) - Complex acid transactions - Operational data store	- Data discovery - processing unstructured data - Massive storage and processing

Modes of Hadoop configuration

Hadoop is programmed using java.

Type	Feature
Standalone mode	All Hadoop services runs in a single JVM on a single machine.
Pseudo distributed mode *(learning stage)*	Each Hadoop runs on its own JVM but on a single machine.
Fully distributed mode *(enterprise level)*	Hadoop services run on individual JVM but they're in separate commodity machine in a single cluster.

HDFS - Hadoop Distributed File System

- A key component of the Hadoop ecosystem
- Designed to store and manage large volumes of data across a distributed network of machines.

HDFS Architecture

Master slave architecture

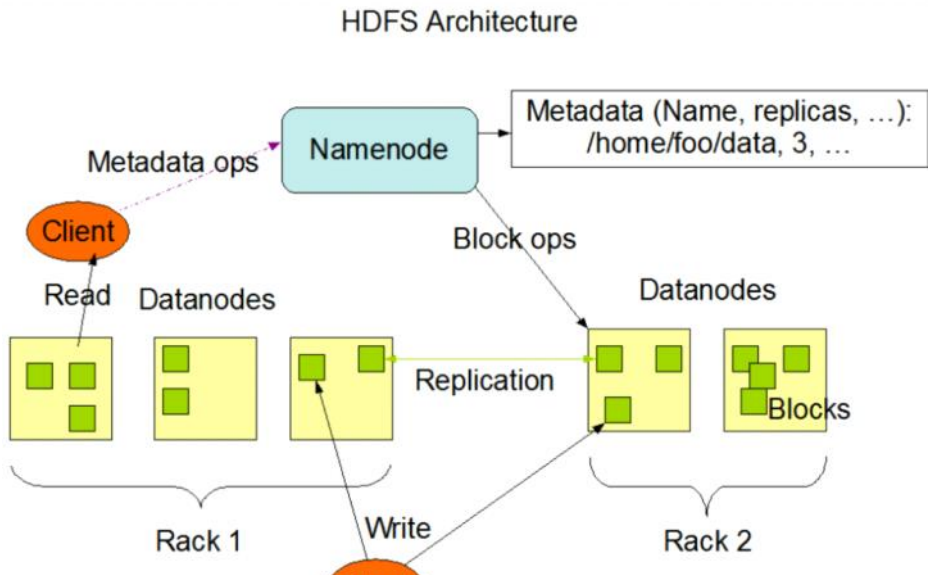
Designed to

- handle high throughput (output) access to data
- provide fault tolerance
- Provide high scalability

Components:

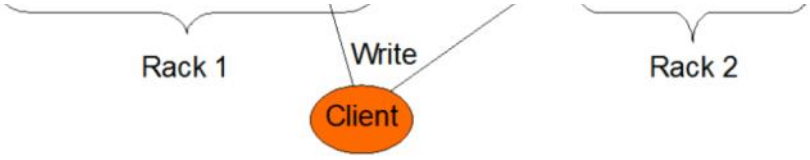
1. Namenode
   - Acts as master server
   - Manages metadata of the file system
   - Role - to keep file system info (path to data blocks, replicas, file permissions, etc.)
2. Datanode
   - Acts as worder/slave node
   - Holds the actual block of data
3. Secondary Namenode
   - Helps keep the information organized
   - Makes recovery faster by creating regular snapshots of the data's state (checkpointing)

The client only interacts with the Namenode.



How is data efficiently stored? How are the resources efficiently used?

- Number of blocks in a Datanode is defined by a default block size of 128 MB
  so if file size is 512 MB
  - it will be split into 4 blocks
  - While making sure that they're close to Namenode
  - And don't forget that each one of them replicates too
- All data blocks will be replicated by 3 (default value)
- Namenode tries to access the nearest Datanode while performing Read & Write operations.
- Replicas will not be placed in the



Read & Write operations.

- Replicas will not be placed in the same rack.
    - If no rack is available, then it'll atleast make sure to not place within the same Datanode

What is a rack?

- Set of nodes (servers)
- Physically located together in the same physical location within the same centre.

HDFS and YARN is still relevant. MapReduce, not as much. Spark changed a lot of things.

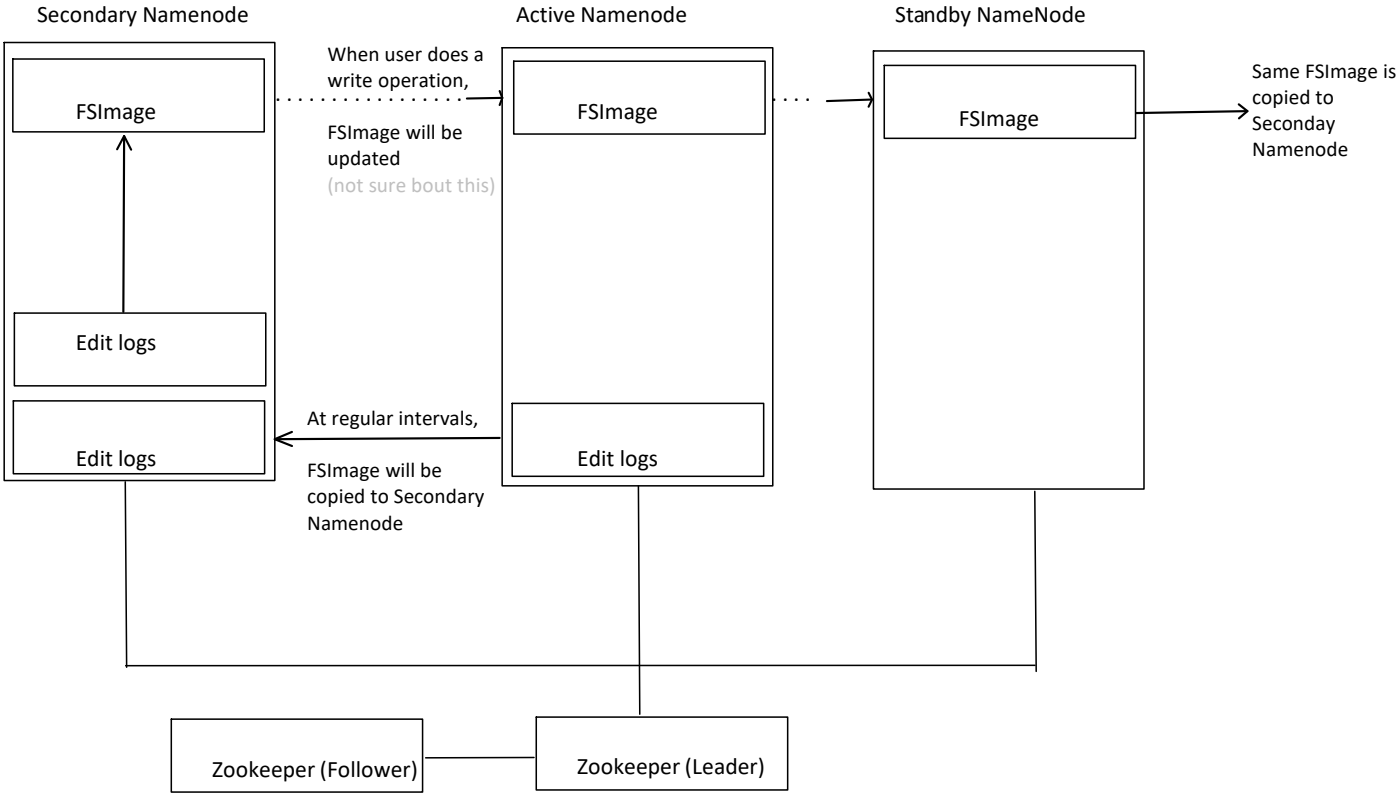
What happens when a Datanode goes down?

- The blocks get replicated (to maintain replication factor) and are put in other Datanodes.
- What if the previous node comes back up?
    - The other one will be dropped/deleted.

How many Namenodes are there? What if the Namenode goes down?

*There will be ONLY one Namenode working at a time, and it's called **active Namenode***

HDFS HA (High Availability) Architecture



Explaining the terms (some from the image)

- FSImage
    - *Stores snapshot of the entire filesystem's metadata*
    - FileSystem Namespace
    - Path to file system in cluster's DataNode

- Edit logs

Fun fact
In older version of HDFS, the cluster used to be down frequently, since

- Path to file system in cluster's DataNode

- Edit logs
  - Changes made in the file that are not present in the FSImage

- Zookeeper
  - It has a
    - Leader - directly connected to cluster's Namenode
    - Follower - backup
  - It
    - acts as a coordinator that monitors health of Active NameNode
    - Decides if the automatic fail over should take place (where it will make Standby Namenode replace Active Namenode)
- Heartbeat
  - Regular signal sent from Datanode to Namenode
  - Used to indicate that the Datanode is still alive and functioning.

Fun fact
In older version of HDFS, the cluster used to be down frequently, since logs and FSImages took a lot of space in Memory (RAM).
Due to memory shortage, the cluster used to regularly go down.

This got resolved in Hadoop 2 after introducing Secondary Namenode

Checkpointing

Process of creating consistent snapshot of filesystem's metadata (which is stored in FSImage)

- Primary function of the secondary Namenode
- It periodically merges Namenode's edit log (changes in the file system) with the current file system (FSImage)

But why do we need this?

Recovery, performance, stability

- The Namenode keeps edit log of all modifications made to the file system.
- Over time, the log can grow large, which is why secondary Namenode helps to convert the edit logs into file system (FSImage) by creating checkpointing.
- If the log grows large,
  - Namenode will take more time to convert it into FSImage (by restarting it)
  - And the cluster will be down during this phase.

- Thus secondary Namenode helps reduce recovery time of the cluster.
- Due to check pointing, the downtime of the cluster is minimized

Hadoop Practical [27 Aug24]

27 August 2024 09:41

Linux commands

mv	move file. syntax: *sudo mv <filepath> <destpath>*
sudo	super user do equivalent of admin
cd	change directory
ls	list
ls -l	list all details
tar xvzf	extracting a file x — extraction v — verbocity z — filtering out .z zip file f filename
sudo mv *<oldfilename> <newname>*	to rename file
history	list of previous commands entered
cd ~	move to home directory
sudo apt get name	to install
sudo apt get remove	to remove
777 first digit — users permission second digit — group's permission third digit — others	Rwxrwxrwx 4 — r — read 2 — w — write 1 — x — execute 7 — rwx — all three
whereis filename	to find location of a file to find variables, make sure to use $ before var name
|	to join two commands together

javaws

- stands for java web start
- used to launch java application on web browser

SSH

- secure shell
- to manage remote & local machine
- *communication between Namenode and Datanode*
- Ipv 6 needs to be disabled (only IPV4 will be installed)

HDFS commands

when writing commands, start with either

- **hdfs dfs**
- **hadoop fs**

FILESYSTEM COMMANDS:

`~$ start-dfs.sh && start-yarn.sh`	start shell
`hadoop fs -mkdir -p /newdir1/newdir2/newdir3`	make new dir
`$ hdfs dfs -touchz /newdir1/tempfile.txt`	create new empty file
`hdfs dfs -cat /newdir/newfile.txt`	display content of file

`$ hdfs dfs -copyFromLocal newfile.txt /newdir`	copy the **file** from local system and put in hadoop dir
`$ hdfs dfs -appendToFile newfile.txt /newdir1/tempfile.txt`	copy **content** of file from local to one file to another
`hadoop@hadoop-VirtualBox:~$ hdfs dfs -appendToFile newfile1.txt newfile.txt /newdir1/tempfile.txt`	copy content from multiple sources to one destination (local to hadoop)
`hadoop fs -cat /newdir1/tempfile.txt > newfile2.txt`	copy content from hadoop file to local file. if permission denied, enter *sudo chmod 777 filename*
`$ hdfs dfs -put newfile2.txt /hdfs`	alternative for *copyFromLocal*
`hdfs dfs -cp /hdfs/newfile2.txt /newdir`	copying the file from one directory of the hadoop to another one (within hadoop itself)
`hdfs dfs -copyToLocal /newdir1/tempfile.txt`	copying from hadoop to local. if destination is not mentioned, it will automatically be home dir.
`$ hdfs dfs -ls /`	display directories in hdfs home

`$ hdfs dfs -ls -R /`
or
`$ hdfs dfs -ls -R hdfs://localhost:9000/`
	display all directories and its files in hadoop
	
	R for recursive

`$ hdfs dfs -rm /hdfs/newfile2.txt`	delete a file/dir
`$ hdfs dfs -rm -r /newdir`	to delete a non empty directory
`hdfs dfs -rmdir /hdfs`	delete an empty directory
`hdfs dfs -expunge`	to empty the trash
`hadoop@hadoop-VirtualBox:~$ hdfs dfs -ls /hdfs/*` `-rw-r--r--  1 hadoop supergroup  69 2024-08-27 13:34 /hdfs/newfile.txt`	display details of a particular file
`$ hdfs dfs -chmod 766 /hdfs/newfile.txt`	change access permissions of a file
`hdfs dfs -chown hadoop:hadoop /hdfs`	change ownership *user:group*
`$ hdfs dfs -chown -R hadoop:hadoop /hdfs`	change ownership of file and folder recursively
`hadoop@hadoop-VirtualBox:~$ hdfs dfs -stat /hdfs/newfile.txt` `2024-08-27 10:34:49`	to show status of file
`hadoop@hadoop-VirtualBox:~$ hdfs dfs -stat %r /hdfs/newfile.txt` `1`	displays number of replicas
`hadoop@hadoop-VirtualBox:~$ hdfs dfs -stat %b /hdfs/newfile.txt` `69`	displays byte size of the file
`hadoop@hadoop-VirtualBox:~$ hdfs dfs -setrep 3 /hdfs/newfile.txt` `Replication 3 set: /hdfs/newfile.txt`	set replication factor to 3

`hadoop@hadoop-VirtualBox:~$ hdfs dfs -du /`
`69  /hdfs`
`219  /newdir1`
`hadoop@hadoop-VirtualBox:~$ hdfs dfs -du -h /`
`69  /hdfs`
`219  /newdir1`
	size occupied by each of the directories shown
	
	-h is human readable format

`hadoop@hadoop-VirtualBox:~$ hdfs dfs -df -h /`
`Filesystem  Size  Used  Available  Use%`
`hdfs://localhost:9000  19.6 G  52 K  11.0 G  0%`
	size occupied by the whole cluster

`hadoop@hadoop-VirtualBox:~$ hdfs dfs -count /` `5 2 288 /`	number of directories, number of files, total size
`hadoop@hadoop-VirtualBox:~$ hdfs dfs -count /hdfs` `1 1 69 /hdfs`	same as above but for a directory

CLUSTER MAINTAINENCE COMMANDS

`hadoop@hadoop-VirtualBox:~$ hdfs fsck /`
`Connecting to namenode via http://localhost:50070/fsck?ugi=hadoop&path=%2F`
`FSCK started by hadoop (auth:SIMPLE) from /127.0.0.1 for path / at Tue Aug 27 14:28:41 EAT 2024`
`.`
`/hdfs/newfile.txt: Under replicated BP-188004968-127.0.1.1-1724742366239:blk_1073741830_1007. Target Replicas is 3 but found 1 replica(s).`
`.Status: HEALTHY`
` Total size: 288 B`
` Total dirs: 5`
` Total files: 2`
` Total symlinks: 0`
` Total blocks (validated): 2 (avg. block size 144 B)`
` Minimally replicated blocks: 2 (100.0 %)`
` Over-replicated blocks: 0 (0.0 %)`
` Under-replicated blocks: 1 (50.0 %)`
	status of the cluster is shown
	
	(*file system check)*

`hdfs fsck / -files`	same as above but status of file shown

`hadoop@hadoop-VirtualBox:~$ hdfs fsck / -files -blocks -locations`
`Connecting to namenode via http://localhost:50070/fsck?ugi=hadoop&files=1&blocks=1&locations=1&path=%2F`
`FSCK started by hadoop (auth:SIMPLE) from /127.0.0.1 for path / at Tue Aug 27 14:33:36 EAT 2024`
`/ <dir>`
`/hdfs <dir>`
`/hdfs/newfile.txt 69 bytes, 1 block(s): Under replicated BP-188004968-127.0.1.1-1724742366239:blk_1073741830_1007. Target Replicas is 3 but found 1 replica(s).`
`0. BP-188004968-127.0.1.1-1724742366239:blk_1073741830_1007 len=69 repl=1 [DatanodeInfoWithStorage[127.0.0.1:50010,DS-a70687a0-8a0a-4d6f-9e86-616bc750a4d9,DISK]]`
 	
`/newdir1 <dir>`
`/newdir1/newdir2 <dir>`
`/newdir1/newdir2/newdir3 <dir>`
`/newdir1/tempfile.txt 219 bytes, 1 block(s): OK`
`0. BP-188004968-127.0.1.1-1724742366239:blk_1073741826_1003 len=219 repl=1 [DatanodeInfoWithStorage[127.0.0.1:50010,DS-a70687a0-8a0a-4d6f-9e86-616bc750a4d9,DISK]]`
	info bout both files and blocks

`$ hdfs fsck / -files -blocks -locations -racks`	racks too

`hadoop@hadoop-VirtualBox:~$ hdfs balancer -threshold 1`
`Time Stamp Iteration# Bytes Already Moved Bytes Left To Move Bytes Being Moved`
`The cluster is balanced. Exiting...`
`Aug 27, 2024 2:38:45 PM 0 0 B 0 B -1 B`
`Aug 27, 2024 2:38:45 PM Balancing took 2.361 seconds`
	HDFS balancer utility
	over period of time data becomes unbalanced all across Data nodes

`-$ stop-yarn.sh && stop-dfs.sh`	stop cluster services

hadoop fs -Ddfs.blocksize = 134217728 -put /home/hduser/test/test.text /hdfs	to change block size of a specific file in a cluster

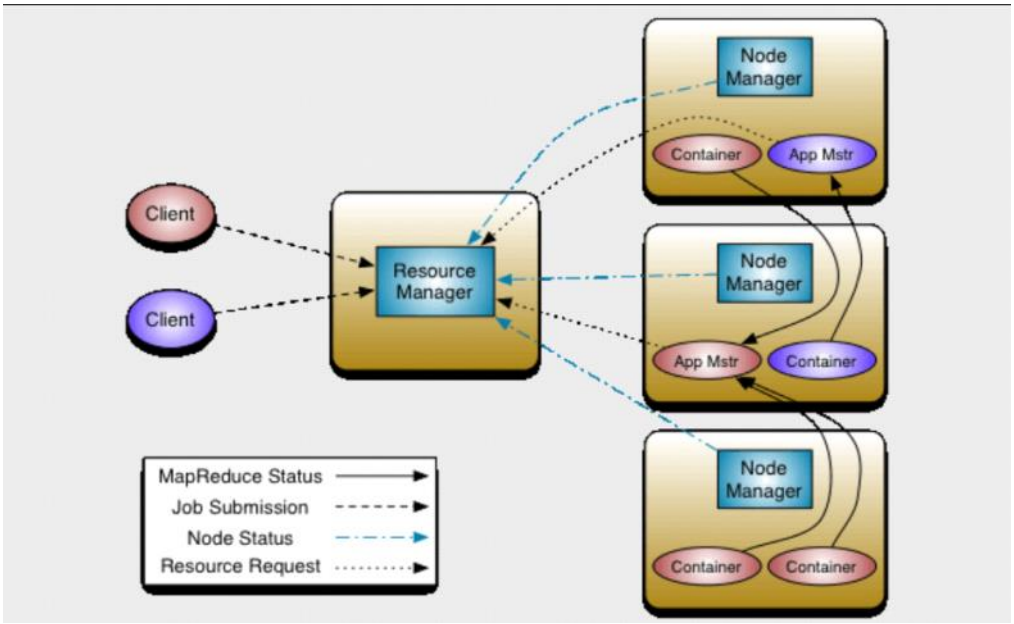
YARN [27 Aug24]

27 August 2024 16:38

Yet Another Resource Negotiator

Refer this for framework
got added in Hadoop version 2

Architecture



snapshot
refer theory

- two services
  - Resource manager - job scheduler and application manager (master)
  - Node manager (slave)

- job scheduler:
  focuses on managing and scheduling jobs/tasks
  1. user submits job
  2. it goes to job scheduler
  3. job is scheduled using FIFO, FAIR, capacity scheduler
  4. job scheduler will allocate initial resources for the job

- application manager
  manages overall lifecycle of applications
  1. it will accept job from job scheduler
  2. will request node manager to allocate containers (resources - RAM, CPU, network, data blocks)
  3. it will monitor job execution.
  4. if it requires more resources, it will request to distribute resources as needed
  5. if the job fails, it will request to restart job

- App master
  monitors containers, manages lifecycle of the application
  1. monitor resources
  2. negotiate about resources for running job
  3. run one container for each job
  4. will kill itself once job is finished
  5. *bidirectional between app master and app manager*

- Node manager
  manages and allocates resources on individual nodes within the cluster
  - sends status of slave node using heartbeat

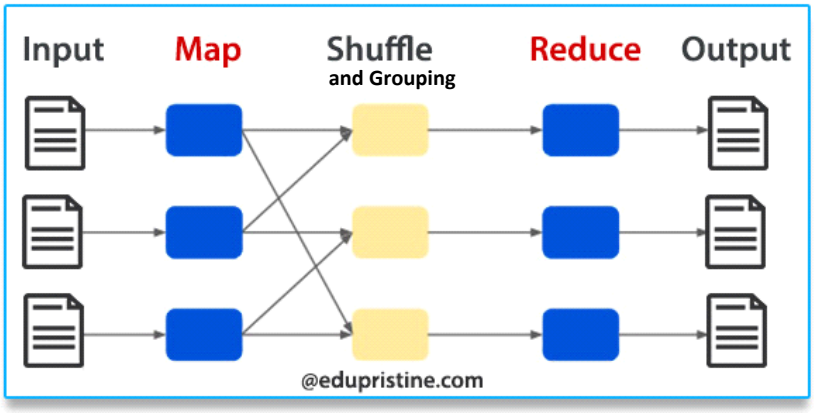
what's a container?
resource allocation unit

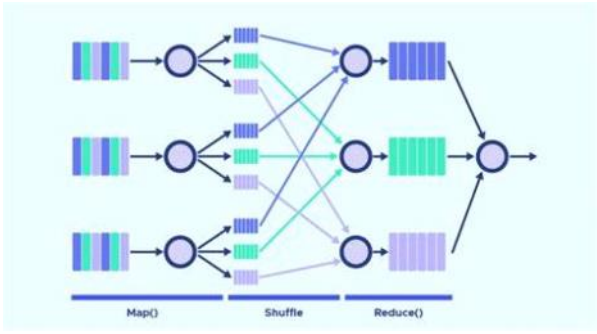
- virtual space where the task will be executed
- since its an in-disk task ie program moves to where location is present
- each job has one container
- killed after job is done

MapReduce [27 Aug24]

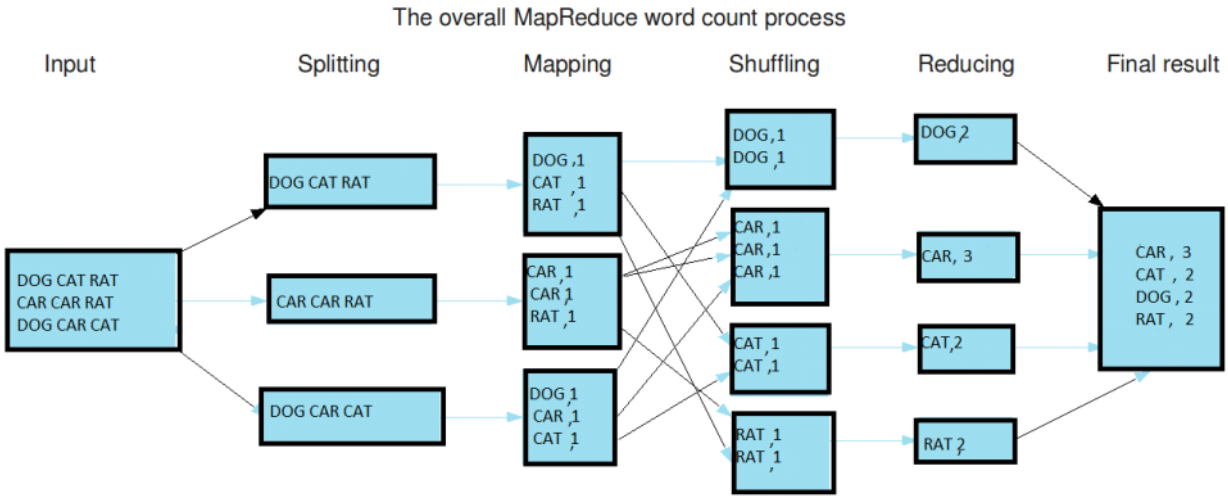
27 August 2024 17:29

Framework



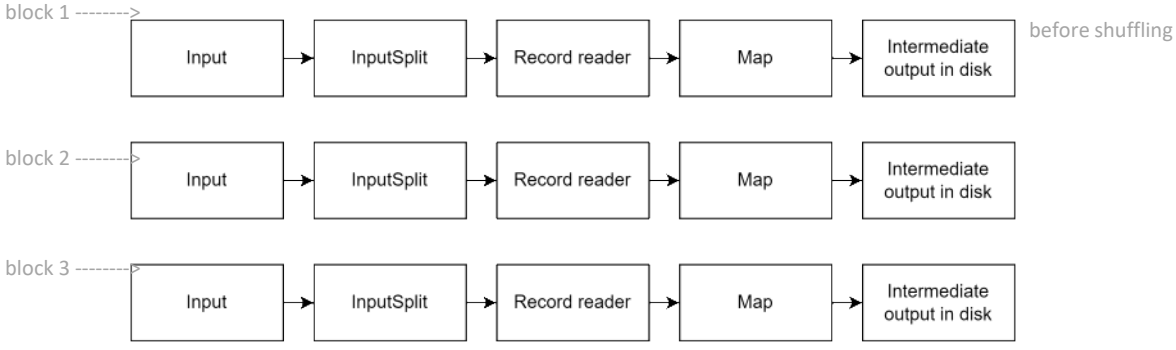


Example - word count



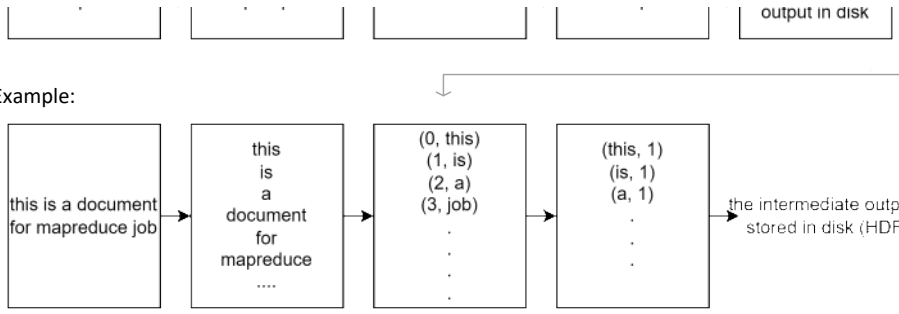
Mapper - Input split and record reader

mapper can be independent but not reducer
in mapping, same function is applied to all



Example:

this part is important in java but not in python



Partitioning

- its the pre reducer: ensures same key goes to same reduce
- determines how the output of mapper is distributed across reducers
- it decides which key value pair will go to which reducer
- load balancing
    - it helps in balancing load among reducers

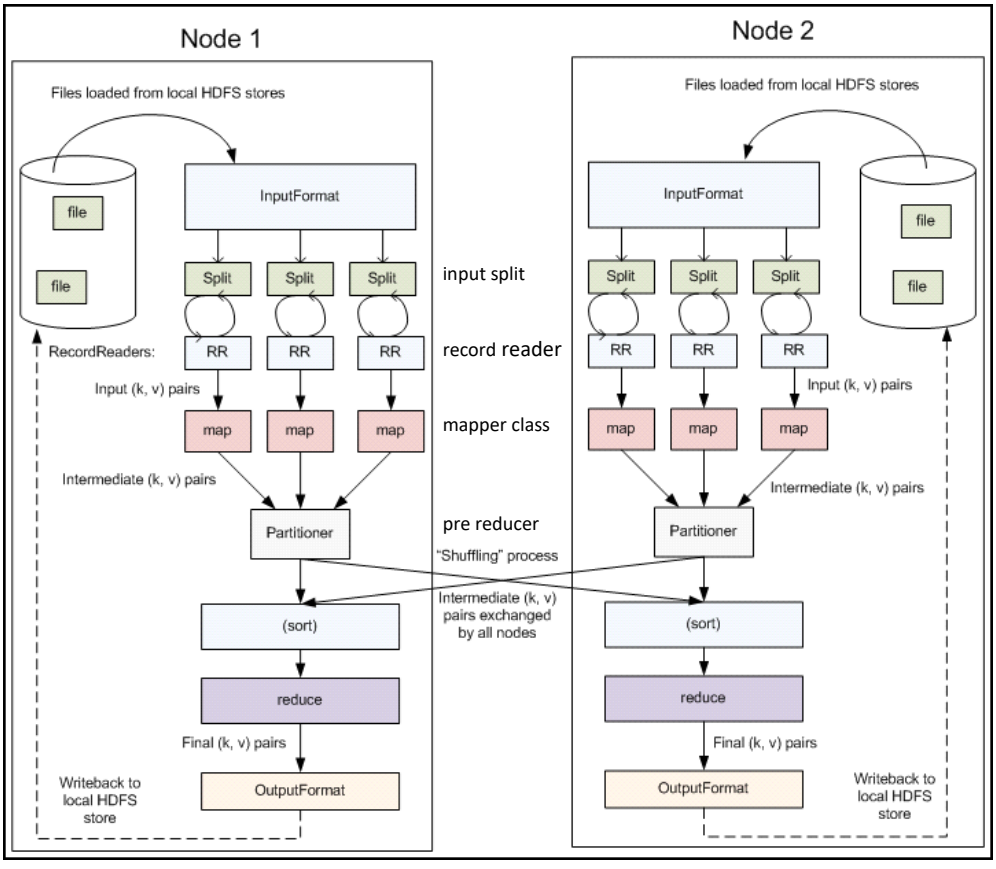
combiner

- its an *optional* local reducer which performs partial aggregation of mapper output before its sent to the reducers
- it reduces volume of data which is transferred between mapper and reducer
- ex. in a word count program, the combiner will do reducing within one block (if a word is repeated more than once)

Reducer

- reducer will do aggregate operation

The whole process is an in disk operation, so even the result will be saved in a file and automatically generated.
just make sure that the path to folder is defined



Understanding the code (refer VirtualBox)

Here's a breakdown of the `sort -k1,1` command:

- `sort`: This is the command used to sort lines of text files.
- `-k1,1`: This specifies the key for sorting. `-k` is an option that defines the sort key. The format `-k1,1` indicates that the sorting should be done based on the first field.

Understanding `-k1,1`

- `-k1,1`: This tells `sort` to use the first field (column) as the key for sorting.
  - `1,1` specifies the start and end positions of the key, meaning it starts and ends at the first field.

Storm [29 Aug24]

29 August 2024 14:47

- free
- open source
- distributed
- realtime

- streaming data framework with high ingestion rates
- ensures that every message is processed atleast once across the topology

Limitations

- no framework level support
- storm development and installation was challenging

Hadoop - Q&As

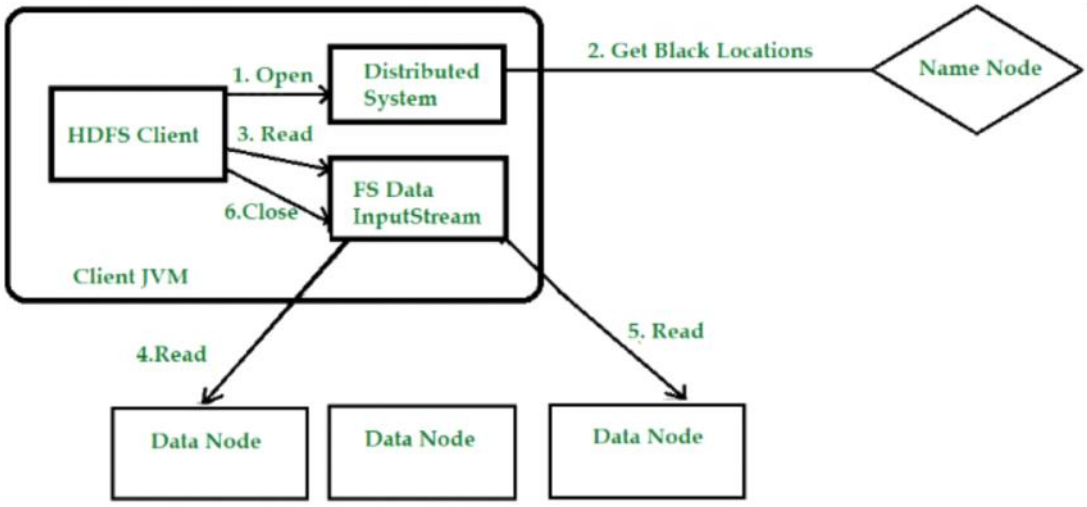
29 August 2024 11:10

Topic	Question	Answer
HDFS	how does hadoop achieve fault tolerance?	- replication of data nodes - secondary namenode
HDFS	if a file size is 129 MB, how many data blocks will be created?	2 (128+1)
HDFS	how does Hadoop handle data consistency?	- write once, read many - once file is written, it cannot be modified - ensures that all datanodes read same version of the file
YARN vs MapReduce	How does YARN differ from Original MapReduce framework in Hadoop v1?	Jobtracker - in v1, only JobTracker was responsilbe for resource management and job scheduling. this led to - scalability - job tracker failures due to overburden of tasks -in v2, the tasks were split into resource manager and app master
MapReduce	Suppose MapReduce job has 10 map tasks and 5 reduce tasks. If one of the node running map fails, what impact will it have on overall job execution and how does hadoop handle it?	if a node running a map task fails, Hadoop will reschedule failed map tasks on another available node. the system ensures that the job is done by re running only failed map task. the fault tolerance mechanism ensures that overall job execution is resilient to individual node failure *the reduce job starts ONLY after mapping job is finished*
HDFS	In a hadoop cluster where multiple applications are running simultaneously, what would be the impact of increasing replication factor of HDFS on overall system performance?	- positive • data reliability increases • fault tolerance improved - negative • storage utilization decreases • extra load on NameNode • write performance decreases (multiple write operation) • network traffic
MapReduce	How would you optimize performance of MapReduce job that can be running longer than expected?	- resource allocation - data locality : minimize data transfer across the network by ensuring that tasks run on nodes where data is located - balancing number of maps and reduce tasks that align with cluster's capacity

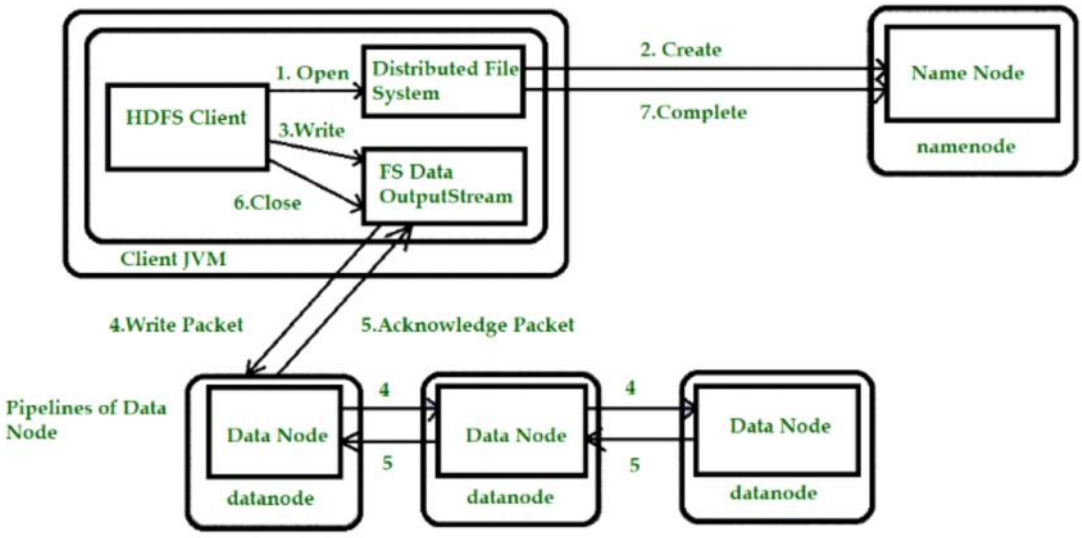
References

26 August 2024 16:05

HDFS read and write



read



write