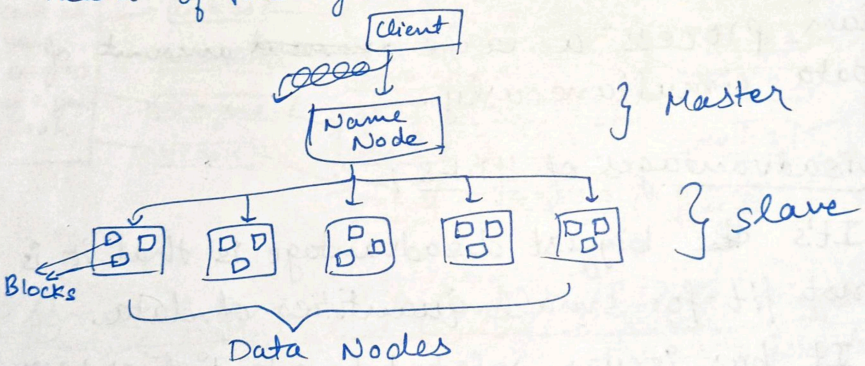
* HDFS :-

- Hadoop Distributed File System
- A scalable, fault tolerant, High performance distributed file system.
- Asynchronous replication
- Write-once & read-many (WORM)
- 3 minimum Data Nodes. cluster
- Data divided into 64 MB or 128 MB blocks. Each block replicated 3 times.
- These replicated blocks are stored in diff. Data nodes.
- HDFS consist of two core components → Name Node, Data Node.
- HDFS maintains all the coordination b/w the clusters & hardware, thus working at the heart of the system.



* Name Node :- It is the prime node which contains meta data like,
list of files, list of blocks in each file,
List of Data Nodes for each block, Creation time

* ~~Data Node :-~~

* Data Node :-

- Stores the actual data.
- Block server stores data in the local file system.
- Periodically sends a report of all existing blocks to the name node.
- Periodic validation of 'check sum'.
- It serves read, write request, performs block replication, deletion & creation upon instruction from Name node.

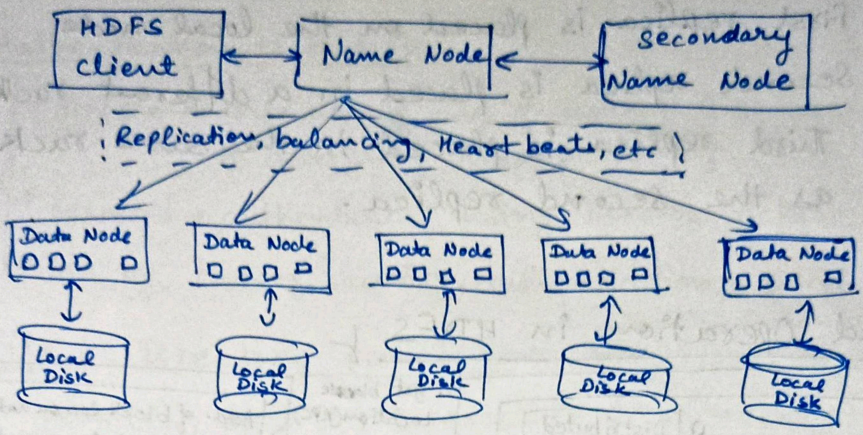
* Advantages of HDFS :-

- It is inexpensive, immutable in nature, stores data reliably, ability to tolerate faults, Scalable, block structured.
- Can process a large amount of data simultaneously.

* Disadvantages of HDFS :-

- It's biggest disadvantage is that it is not fit for small quantities of data.
- It has issues related to potential stability, restrictive & rough in nature.

* HDFS Architecture :-



* Unique features of HDFS :-

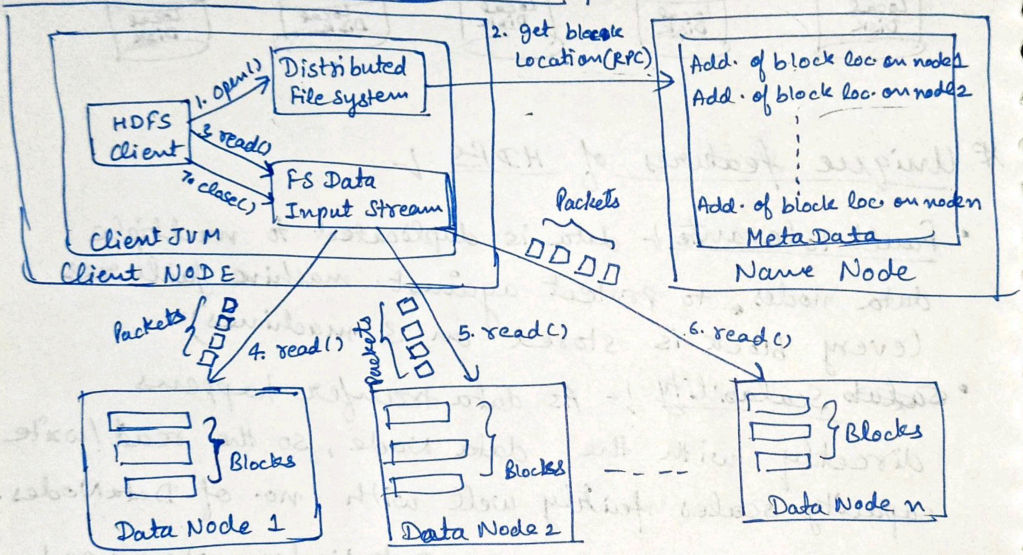
- Fault Tolerance :- data is duplicated to multiple data nodes, to protect against machine failures (every block is stored on 3 machines)
- Scalability :- As data transfer happens directly with the data Node, so the read/write capacity scales fairly well with no. of DataNodes.
- Space :- Just add more DataNodes when need more disk space & rebalance.
- Industry Standard :- Other distributed applications are built on top of HDFS.

* HDFS - Data Organization :-

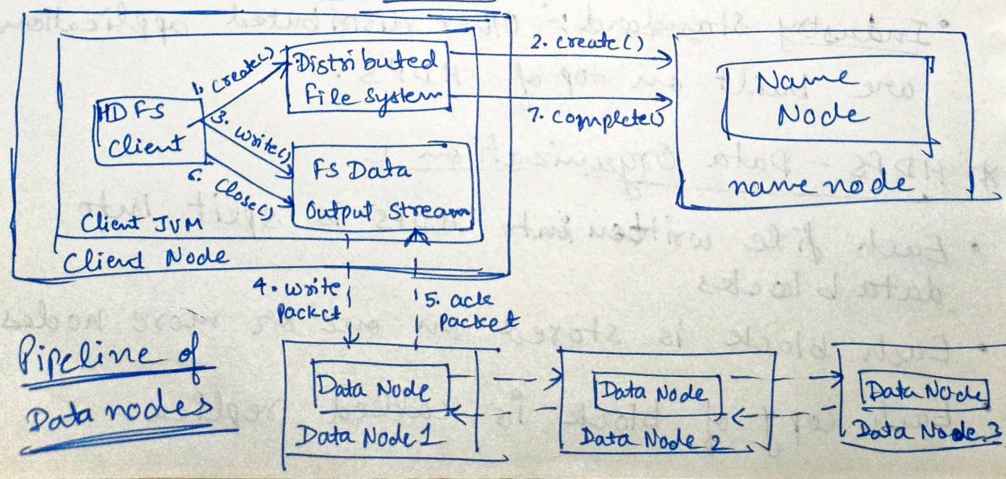
- Each file written into HDFS is split into data blocks
- Each block is stored on one or more nodes.
- Each copy of block is called replica.

- Block Replacement Policy :-
  - First replica is placed on the local node
  - Second replica is placed in a different rack
  - Third replica is placed in the same rack as the second replica.

Read Operation in HDFS :-



Write Operation in HDFS :-



* HDFS Security :-

- **Authentication to Hadoop:**
  - Simple → insecure way of using O.S. username to determine hadoop identity.
  - Kerberos → authentication using kerberos ticket.
  - Set by `hadoop.security.authentication = simple|kerberos`
- **File & Directory permissions are same like POSIX**
  - read (r), write (w) & execute (x) permissions
  - also has an owner, group & mode.
  - enabled by default (`dfs.permissions.enable = true`)
- ACL's are used for implementing permissions that differ from natural hierarchy of users & groups.

* Interfaces to HDFS :-

- Java API
- C wrapper
- HTTP protocol
- WebDAV protocol
- Shell commands.

Hadoop Security

* Problem Statement:-

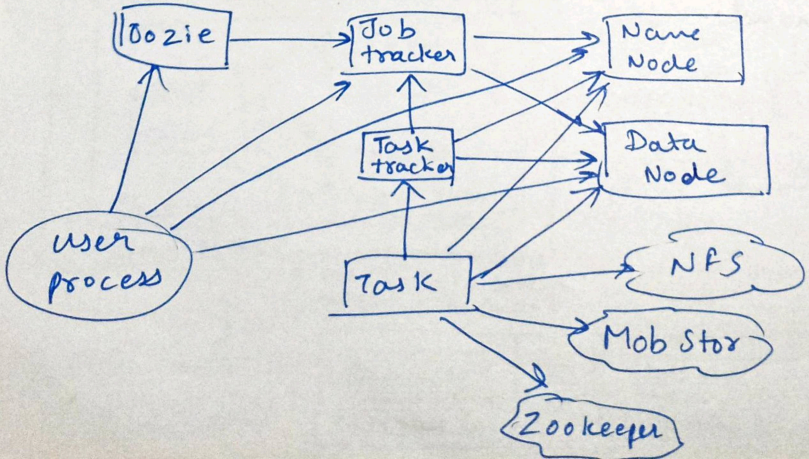
- The fundamental goal of adding Hadoop security is that YAHOO's data stored in HDFS must be secure from unauthorized access.

Furthermore, it must do so without adding significant efforts to operating or using the grid.

→ All the HDFS clients must be ~~authorized~~ authenticated to ensure that user is who they claim to be.

→ Since Data Nodes & Task trackers are entrusted with user data & credentials, they must also be authenticated to ensure they are running as part of the grid & ~~not~~ are not trojan horses

→ Kerberose will be underlying authentication service so that users can be authenticated using their system credentials.



* Security threats in Hadoop :-

- User to Service Authentication
  No user Authentication
  - on Name Node or Job tracker
  - on Data Node → user can read/write any block
  - on Job Tracker →
    - User can modify or kill others jobs
    - can modify state of Job Tracker.

- Service to Service Authentication
  - No Authentication of Data Nodes and Task trackers
  - Users can start fake data Nodes & Task Trackers

Solution to Threats :-

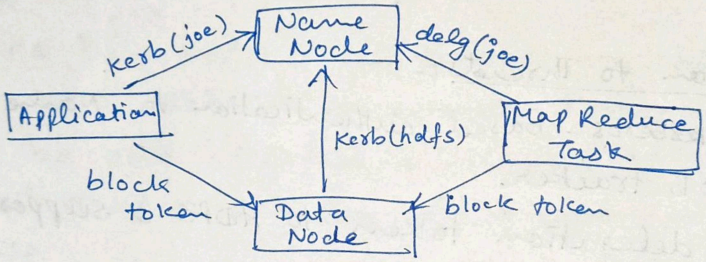
- Add kerberos-based authentication to Name Node & Job tracker
- Add delegation token to HDFS & support for them in Map Reduce
- Determine user's group membership on the Name Node & Jobtracker.
- Protect Map Reduce system directory from users
- Add authorization model to Map Reduce so that only submitting user can modify or kill Job.
- Add backyard authentication to web UI's
- Protecting against root on slave nodes.
  - Encryption of RPC messages
  - Encryption of block transfer protocol
  - Encryption of Map Reduce transient files
  - Encryption of HDFS block files.

- Passing Kerberos ticket to MapReduce tasks for third party kerborized services.

* HDFS security :-

- Users authenticate via kerberos
- MapReduce jobs can obtain delegation tokens for later use.
- When clients are reading or writing on HDFS file, the NameNode generates a block access token that will be verified by the DataNode

* HDFS Authentication :-



- Client authentication to NameNode via:
  - → kerberos
  - → Delegation token
- Client demonstrates authorization to DataNode via block access token.
- DataNode authenticates to NameNode via Kerberos.

* Delegation Token :-

- Advantages over using kerberos directly :-
  - → Dont trust Job tracker with credentials.
  - → Avoid MapReduce task authorization flood
  - → Renewable by third party (i.e. Job Tracker)
  - → Revocable when job finishes

- token ID
- token Authenticator
- Token.

* Block Access Token :-

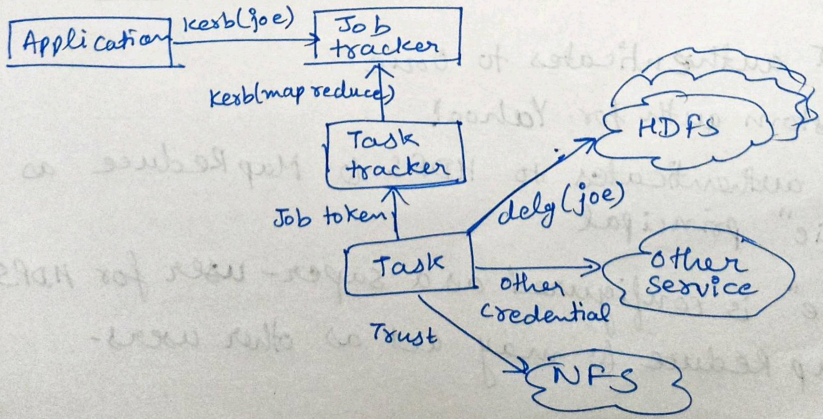
- Only Name Node knows the set of users allowed to access a specific block, so the Name Node gives an authorized client a block access token.
- Capabilities like read, write, copy or replace
- The Name Node & Data Node shares a dynamically rolled secret key to secure the token.

Same.

* Map Reduce Security :-

- Require Kerberos authentication from client.
- Secure the info. about pending & running jobs
- Job tracker creates a random job token.
  ↳ used for connecting to Task's tracker's RPC.

* Map Reduce Authentication :-

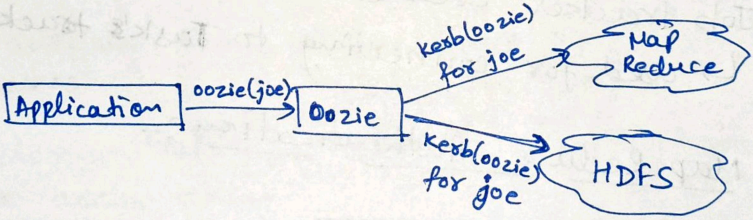


- Client authenticates to Job tracker via kerberos
- Task tracker authenticates Job tracker via kerberos
- Task authenticates to the task trackers using the job token.
- task authenticates to HDFS using a delegation token.
- NFS is not kerberized

* Web UI :-

- MapReduce makes heavy use of web UI for displaying state of cluster & running Jobs.
- HDFS also has a web browsing interface
- Use Backyard to authenticate web UI users
- Only allow submitting users of job to view job's tasks.
- HDFS web browser checks users authorization.

* Oozie :-



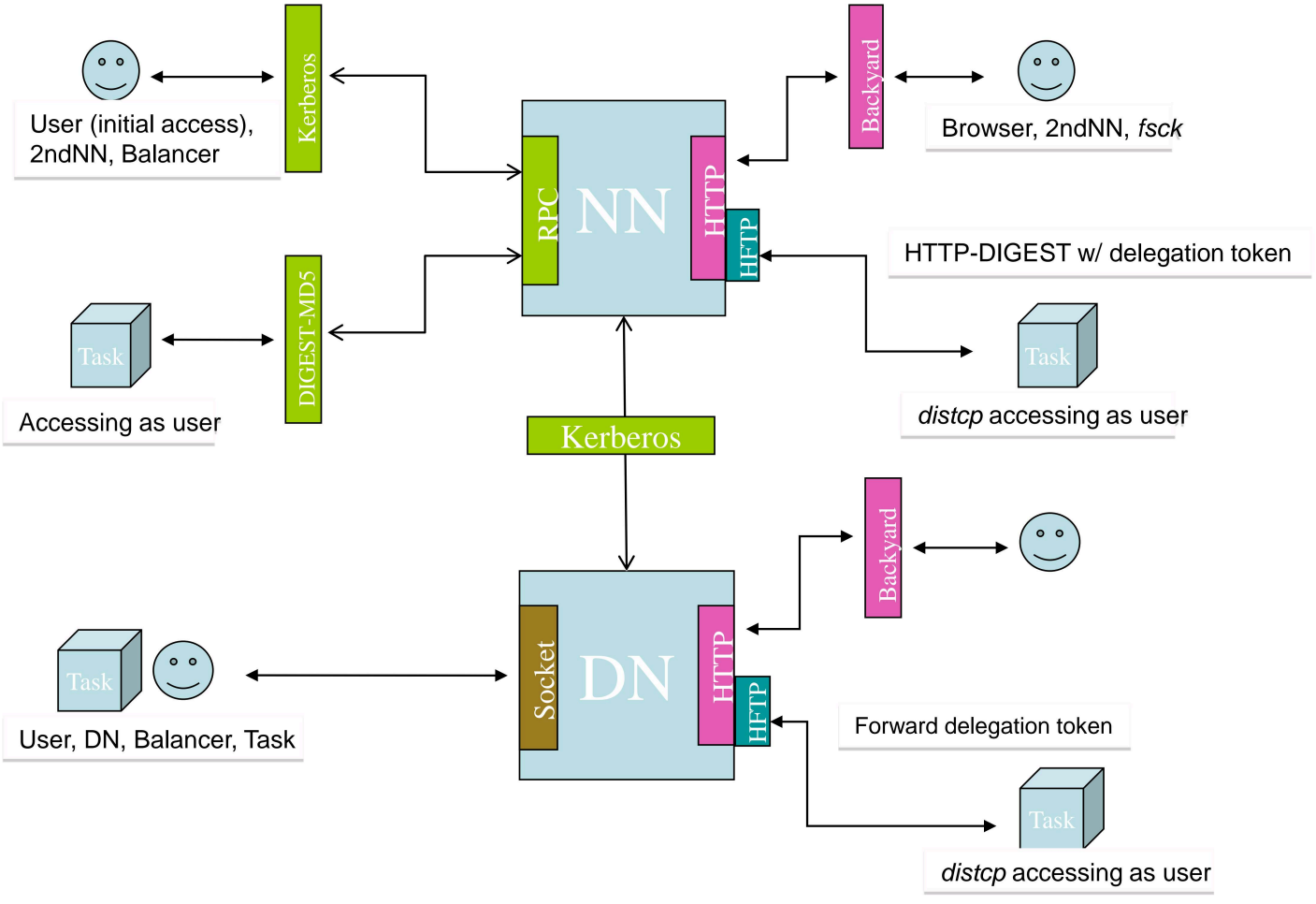
- Client authenticates to Oozie
  ↳ Custom auth for Yahoo!
- Oozie authenticates to HDFS & MapReduce as "oozie" principal
- "Oozie" is configured as a super-user for HDFS & MapReduce & may act as other users.

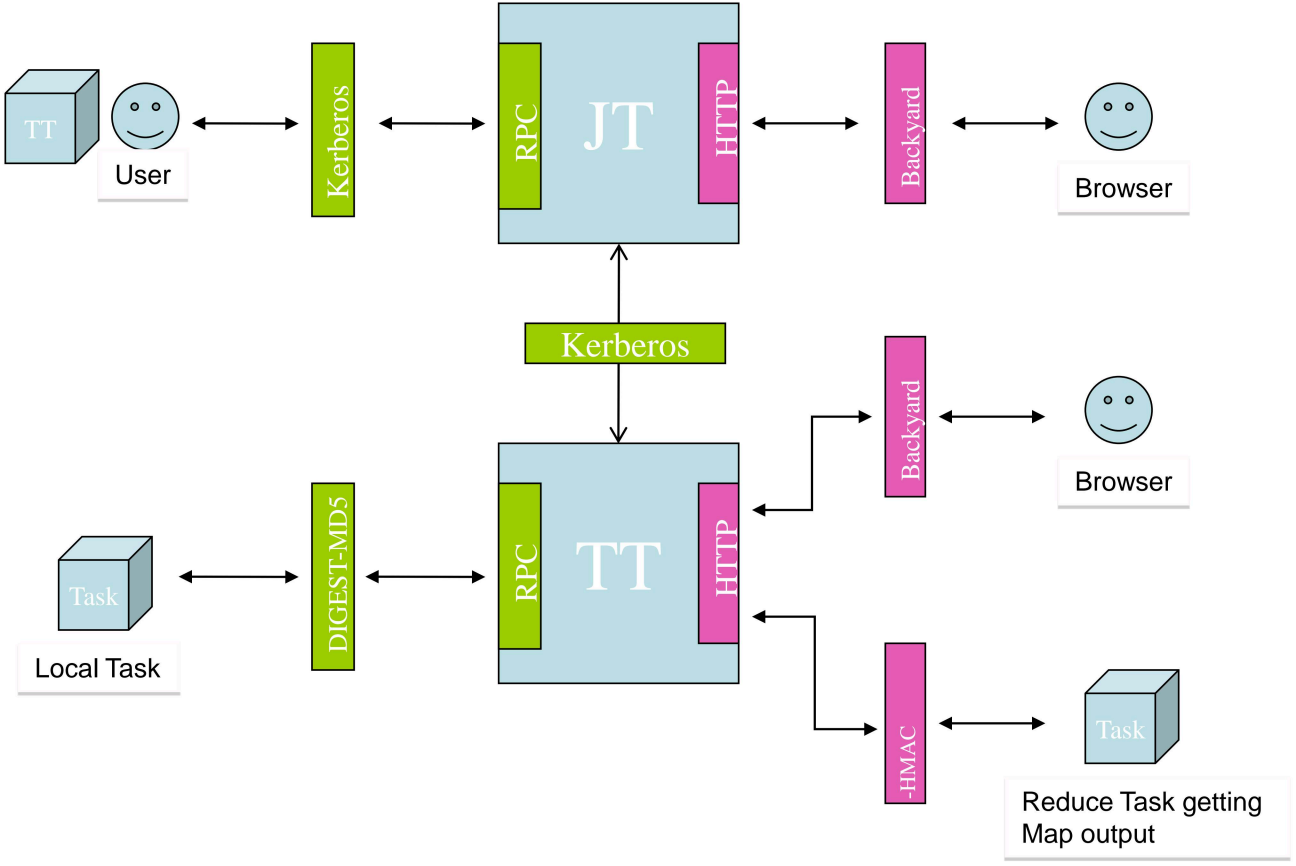
Proxy Services Trust Model

- Requires trust that service (eg. Oozie) principal is secure.
- Explored and rejected
  - Having user headless principals stored on Oozie machine “x/oozie” for user “x”
  - Passing user headless principal keytab to Oozie
  - Generalizing delegation token to have token granting tokens.

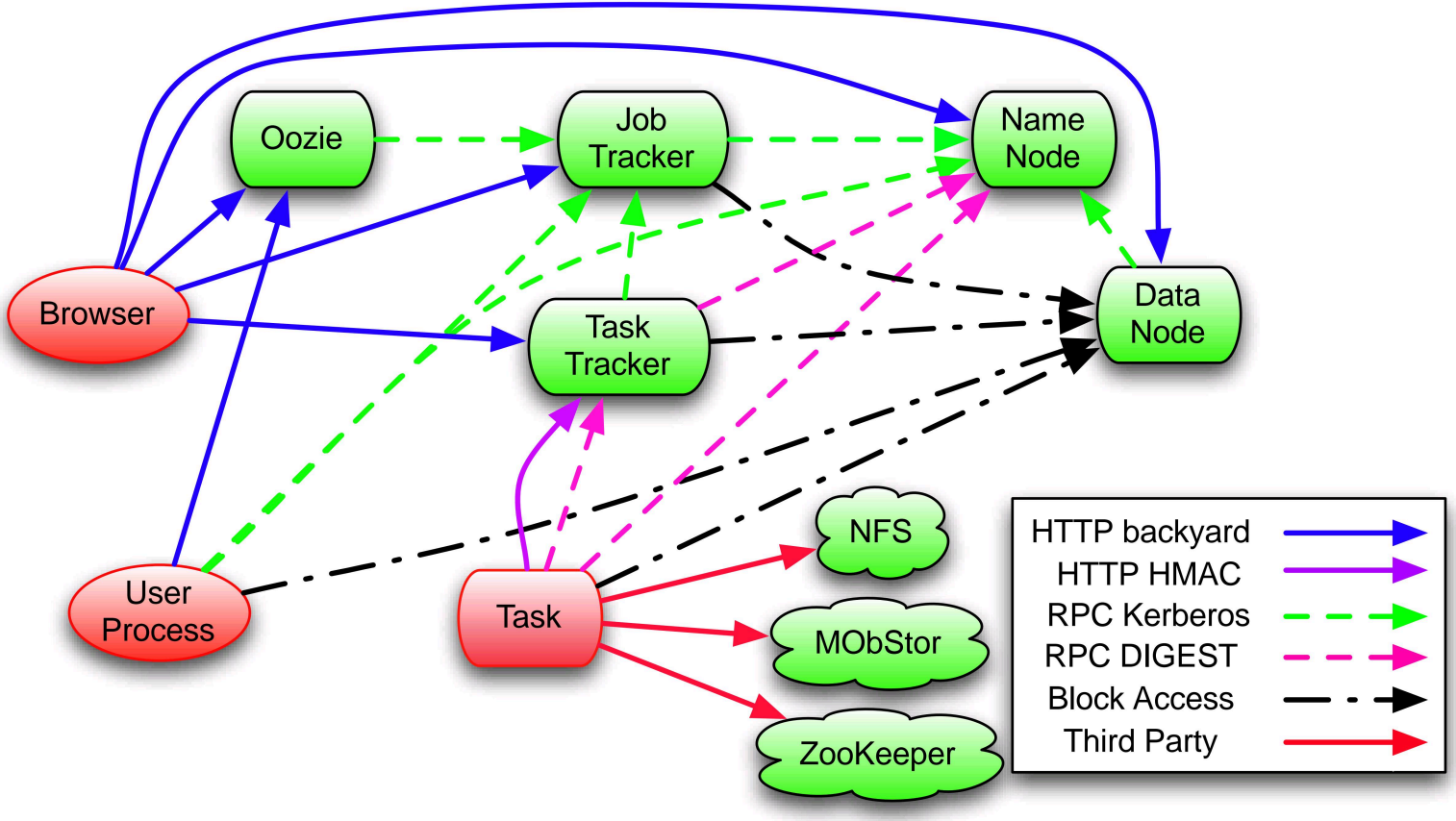
Summary

- RPC
  - Kerberos
    - Application to NameNode, JobTracker
    - DataNode to NameNode
    - TaskTracker to JobTracker
- Block Access Token
- Backyard
  - User to Web UI


User (initial access), 2ndNN, Balancer
Kerberos
Backyard
Browser, 2ndNN, fsck
RPC
NN
HTTP
HFTP
HTTP-DIGEST w/ delegation token
Task
DIGEST-MD5
Accessing as user
Task
distcp accessing as user
Kerberos
Backyard
Task
Socket
DN
HTTP
HFTP
User, DN, Balancer, Task
Forward delegation token
Task
distcp accessing as user


TT
User
Kerberos
RPC
JT
HTTP
Backyard
Browser
Kerberos
Backyard
Browser
Task
Local Task
DIGEST-MD5
RPC
TT
HTTP
-HMAC
Task
Reduce Task getting
Map output

Authentication Paths



Pluggability

- Pluggability in Hadoop supports different environments
- HTTP browser user authentication
  - Yahoo – Backyard
  - External – SPNEGO or Kerberos login module
- RPC transport
  - SASL supports DIGEST-MD5, Kerberos, and others
- Acquiring credentials
  - JAAS supports Kerberos, and others

Performance

- The authentication should not introduce substantial performance penalties.
- Delegation token design to avoid authentication flood by MapReduce tasks
- Required to be less than 3% on GridMix.

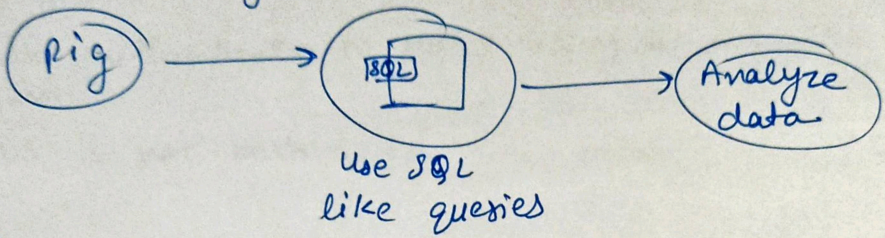
Reliability and Availability

- The Kerberos KDC can not be a single point of failure.
  - Kerberos clients automatically fail over to secondary KDC's
  - Secondary KDC's can be sync'ed automatically from the primary since the data rarely changes.
- The cluster must remain stable when Kerberos fails.
  - The slaves (TaskTrackers and DataNodes) will lose their ability to reconnect to the master, when their RPC socket closes, their service ticket has expired, and both the primary and secondary KDC's have failed.
  - Decided not to use special tokens to handle this case.
- Once the MapReduce job is submitted, the KDC is not required for the job to continue running.

Operations and Monitoring

- The number of Kerberos authorizations will be logged on the NameNode and JobTracker.
- Authorization failures will be logged.
- Authentication failures will be logged.
- The authorization logs will be a separate log4j logger, so they can be directed to a separate file.

Pig :- • Pig is a scripting platform that runs on Hadoop clusters, designed to process & analyze large data sets.



• Pig operates on various types of data like structured, semi-structured & unstructured data

* Map Reduce	HIVE	PIG
① Compiled lang.	SQL like query	Scripting lang.
② Needs to write long complex codes	No need to write complex codes	No need to write complex codes.
③ Can process structured, semi-structured & unstructured data.	Can process only structured data	Can process structured, semi-structured & unstructured data
④ Lower level of abstraction	Higher level of abstraction	Higher level of abstraction.
⑤ Supports partitioning features	⑤ Supports partitioning features	No concept of partitioning in pig.
⑥ MapReduce uses Java & Python	Hive uses a SQL like query lang. known as HiveQL	Pig Latin is used which is a procedural data flow lang.
⑦ Code performance is good	Code performance is lesser than MapReduce & Pig	Code performance is lesser than MapRe but better than

* Components of Pig :-

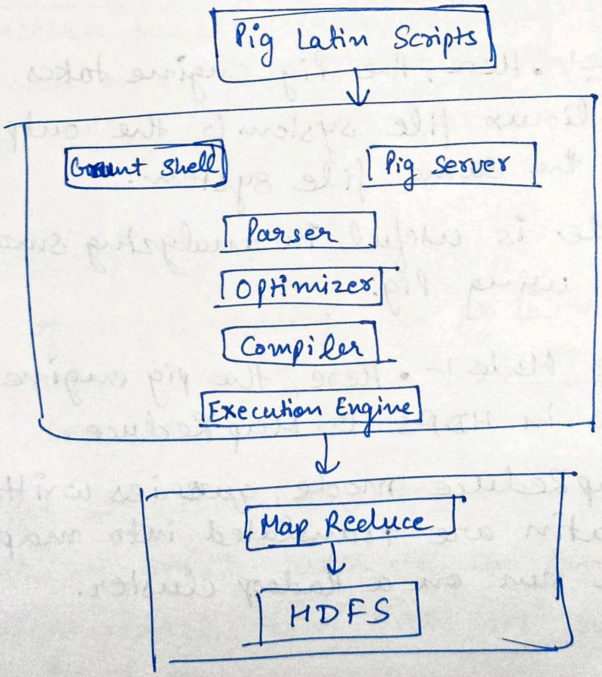
Pig Latin

- It is a procedural data flow lang. used in Pig to analyze data
- It is easy to program using Pig Latin & it is similar to SQL

Runtime engine.

- It represents the execution env. created to run Pig Latin Prog.
- It is also a compiler that produces Map Reduce programs.
- Uses HDFS for storing & retrieving data.

* Pig Architecture :-



* Working of Pig :-

① Load data & write Pig script

② Pig Operations

③ Execution of the Plan

↳ In this stage, the results are shown on the screen otherwise stored in HDFS as per the code.

* Pig Execution Modes :-

Ⓘ ↳ Works in two modes, depending on where the data is residing & where the Pig script is going to run..

① Local Mode :- • Here, the Pig engine takes input from the linux file system & the output is stored in the same file system.

• Local Mode is useful in analyzing small data set using Pig.

② Map Reduce Mode :- • Here, the pig engine interacts & executes in HDFS & Map Reduce.

• In the Map Reduce mode, queries written in Pig Latin are translated into map Reduce jobs. & are run on a Hadoop cluster.

③ Interactive

(11) There are three modes in pig, depending on how a Pig Latin code can be written

→ (1) Interactive Mode :- Means coding & executing the script line by line.

→ (2) Batch Mode :- all scripts are coded in a file with the extention .pig & the file is directly executed.

→ (3) Embedded Mode :- Pig lets it's users define their own functions (UDF's) in programming lang. such as Java.

* Use Case - Twitter :-

500 Million tweets are made on a daily basis → Hadoop & Map Reduce was used to process & analyze this data → Analyzing the no. of tweets created by a user in the tweet table

Features of Pig :-

(1) Ease of Programming, as pig Latin is similar to SQL. Lesser lines of code needs to be written.

(2) Short development time as the code is simpler

(3) Handles all kinds of data.

(4) It lets us create user defined functions.

(5) It offers large set of operators such as join, filter & so on.

(6) Allow multiple queries to process parallely.

(7) Optimization & compilation is easy as it is done automatically

Criteria	HIVE	PIG	SQL
Language used	Uses HIVEQL, a declarative lang.	Uses Pig Latin, a procedural data-flow lang.	SQL itself is a declarative lang.
Definition	An open source built with an analytical focus used for Analysis queries	An open-source & high-level data flow lang. with a Multi-query approach	General purpose database lang. for analytical & transactional queries.
Developed By	facebook	Yahoo	Oracle
Suitable for	Batch Processing OLAP	Complex&nested data structure	Bussiness demands for fast data analysis
Operational for	Structured Data	Structured & semi-structured data	Relational database Management.
schema Support	Support schema for data insertion	Doesn't support schema	Strictly support schema for data storing
Mainly used by	Data Analysts	Researchers & Programmers	Data Analysts, Data Scientists, & Programmers

* When to use Hive :-

- Facebook widely use Hive for analytical purpose on large datasets ↑
- To Query large datasets →
- For Someone familiar with SQL ⇒ It will be easy to use due to similarities b/w both.
- For extensibility ↦ Hive contains a range of user API's that helps in building custom behaviour for the query engine.

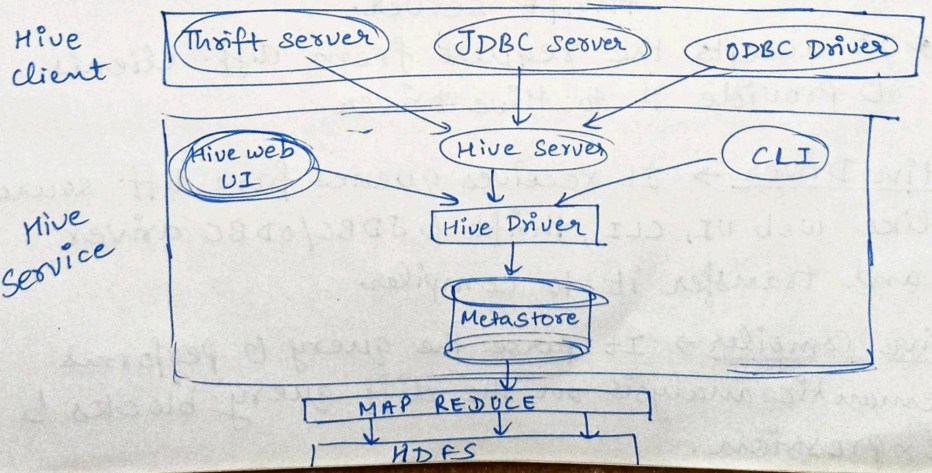
* When to use Pig :-

- For fast Processing → Apache Pig is faster than HIVE because it uses a multi-query approach. It is famous worldwide for speed.
- As a programmer with scripting knowledge, they can learn Pig very easily & efficiently.
- When you don't want to work with schema, because there is no need of schema for data loading related works.

* When to use SQL :-

- For better performance → SQL is better at pulling data quickly & frequently. HIVE is slow in case of Online Transaction. SQL uses OLAP (Online Analytical Processing) for better performance.
- When Datasets are small → It works well with small data & performs much better for small data.
- For frequent data manipulation.

HIVE Architecture :-



- Thrive Service → It is a cross-language service provider that serves the request from all those programming language that supports thrive.
- JDBC Driver → Used to establish connection b/w hive & Java applications.
- ODBC Driver → It allows the applications that supports ODBC protocol to connect to Hive
- Hive CLi :- Here we can execute Hive queries & commands.
- Hive web user Interface → It is alternative of Hive CLI. It provides a web-based GUI for execution of Hive queries & commands
- Hive MetaStore → It is a central repository that stores all the structural information of various tables & partitions in the warehouse.
  - It also includes metadata of columns & its types information.
- Hive Server → It is Reffered to as Apache thrift server.
  - It accepts the request from diff. clients & provide it to Hive Driver.
- Hive Driver → It receives queries from diff. sources like web UI, CLI, thrift & JDBC/ODBC driver and transfer it to compiler.
- Hive Compiler → It parse the query & performs semantic analysis on the diff. query blocks & expression.

It converts HiveQL statements into Map Reduce jobs.

- Hive Execution engine → It executes the incoming tasks in the order of their dependencies.

#

RDBMS	HIVE
① It is used to maintain the database.	① It is used to maintain a data warehouse.
② It uses SQL	② It uses HQL
③ Schema is fixed.	③ schema varies
④ Normalized data is stored	④ Normalized & de-normalized both types of data is stored
⑤ Tables in rdms are sparse	⑤ The table in hive is dense

Advantages of Hadoop :-

① Varied Data Sources → Data like structured, unstructured form. Data like text, XML, images CSV files, etc.

② Cost effective → Economical because it uses a cluster of commodity hardware (cheap machine) to store data.
Hence cost of adding nodes is not much high.

③ Performance → Hadoop's distributed processing & distributed storage architecture process huge amount of data at high speed. (because runs sub-tasks parallely))

④ Fault Tolerant → As we stores atleast 3 copies of a Data at Data Nodes (diff. locations) so fault at one Data Node won't affect the data blocks.

⑤ Highly Available → It supports multiple standby Name Nodes making the system even more highly available as it continue working even if two or more Name Nodes crash.

⑥ Low Network traffic

⑦ High Throughput

⑧ Open Source

⑨ Scalable

⑩ Ease of use

⑪ Compatibility

⑫ Multiple Lang. Supported.

Disadvantages of Hadoop :-

(1) Issue with small files → Hadoop is suitable for small no. of large files but fails in case of large no. of small files because it overloads the NameNode & makes it difficult for Hadoop to function.

(2) Vulnerable by Nature → Hadoop is written in Java which is widely used prog. lang. hence it is easily exploited by cyber criminals.

(3) Processing Overhead → In Hadoop, the data is read from the disk & written to disk which makes read/write operations very expensive in case of TB & PB of data.

(4) Support only Batch Processing → Due to this, it is not efficient in stream processing. It cannot produce output in real-time with low latency.

(5) Security → It uses Kerberos authentication which is hard to manage. It is missing encryption at storage & network levels which are a major point of concern.

(6) Iterative Processing.

Hadoop Admin Responsibilities :-

1. Responsible for implementation & administration of Hadoop instructure.
2. Testing HDFS, Hive, Pig & MapReduce access for applications.
3. Cluster maintenance tasks like Backup, Recovery, Upgrade, Patching.
4. Performance tuning & capacity planning for clusters.
5. Monitor Hadoop cluster & deploy security.

Hadoop Administration

HDFS administration	MapReduce admin
Monitoring the HDFS file structure, location & the updated files	Monitoring the list of applications, configuration of nodes, etc.

* HDFS Monitoring :-

→ HDFS contains the user directories, input files & output files.
Use the mapreduce commands, put & get for storing & retrieving.

* MapReduce Job Monitoring :-

→ A MapReduce application is a collection of jobs (Map job, combiner, Partitioner & Reduce job)

→ It is mandatory to monitor & maintain the following :-

- Configuration of data Nodes where the application is suitable
- The no. of data nodes & resources used per application.

System maintainance operations such as ~~upgrading~~ updating O.S. & applying security patches or hotfixes are routine operations in any data centers. DataNodes undergoing such operations can go offline for a few minutes to several hours.

By design, Apache Hadoop HDFS can handle data nodes going down. However there might be some cases which could lead to temperory data unavailability.

HDFS supports foll. features for performance planned maintainance activity :-

① Rolling upgrade ② Decommission

③ HDFS supports using Maintainance state.

* Rolling upgrade :- This process helps to upgrade the cluster software without taking the cluster offline. When many DataNodes from the same or diff. racks go down for upgrades,

it is preffered to choose the DataNodes in such a way that the replica blocks or files must be available.

* Decommission → This feature is used to reduce the data availability problem encountered during rolling upgrade scenarios.
  - When Decommissioning is requested for DataNode, the Name Node transitions them into "decommission-in progress" state where all of their blocks are replicated onto other live DataNode in order to satisfy the global block replication config. When all blocks are replicated to other Data Node, the state of Name Node is changed to final decommissioned state.

* Maintenance State :-
  - It aims to overcome the draw backs of Rolling upgrade & Decommission features & make the planned maintenance activity much more seamless.
  - Only applies to HDFS DataNode roles.
  - It avoids unnecessary replication of blocks, i.e., it doesn't schedule replication right away for blocks on Data Nodes requesting maintainance activity.

- Even when these data nodes are down for maintenance, the cluster continues to run using the minimum replication limit for replicas instead of desired level set by the global block replication factor.
- The NameNode configuration defines the minimum no. of live ~~blocks~~ replicas that all blocks of DataNode undergoing maintainance need to satisfy.
- For DataNode in maintainance state,
  - if all blocks have atleat this min. replicas on other live DataNode, then no replication.
  - otherwise replication will be triggered with range (0 to dfs.replication).
- If Min. replication factor equals to global replication factor, the cluster needs atleast dfs. replication replica blocks & now it ~~beha~~ behaves like decommission feature.
- if min. replication factor = 0, means the NameNode does not verify any min. replication factor for the blocks on DataNode undergoing for maintainance. An admin. might prefer setting this value.
- if min. replication factor = 1, means cluster can ~~work~~ function with just one live replica.

HBase :-

- HBase is a column-oriented non-relational database management system that runs on top of HDFS.
- HBase provides a fault-tolerant way of storing sparse data sets, which are common in many big data use cases.
- It is well suited for realtime data processing or random read/write access to large volumes of data.
- HBase applications are written in Java
- HBase relies on Zookeeper for high performance coordination.
- HBase works well with Hive.

* How HBase Works :-

- HBase architecture is a great database for storing unstructured data. You can access data using HBase API.
- In HBase, data lives within individual columns indexed by unique row key, called the primary key.
- Hence it makes data retrieval faster.
- HBase distributes the data & the requests across all the servers in an HBase cluster, which makes it possible to query PB of data in matter of milliseconds.

* HBase is Used for :-

- HBase is ideal for high-scale real-time applications, such as social media app or a streaming app.
- As schema is not fixed in HBase, developers can add new data without conforming to a schema model.
- It's common for web applications databases to consist of billion of rows of data. Here RDBMS performs poorly because it slows down exponentially as data grows. HBase can overcome this issue because it is designed to handle large chunks of data.

* Advantages/ Key Featueres of HBase :-

① High Scalability :- With HBase you can scale your application across thousands of servers because HBase applications scale linearly

② Speed :- It comes with low latency read write access to huge amount of structured, semi-structured & unstructured data. This happens by distributing the data to region servers where each of those servers stores a portion of tabel's data. This makes data read & write faster if all data lived on the same server.

③ Fault Tolerance → HBase data is split across many hosts in a cluster, thus the system can withstand the failure of an individual host.

Diff b/w RDBMS & HBase :-

① Data Model :- • RDBMS uses a traditional data model, where data is stored in tables with predefined columns & rows.

• HBase uses column-family data model, & it is often reffered as a NoSQL.

② Scaling :- • RDBMS scale vertically, which means adding more resources to single machine to increase performance.

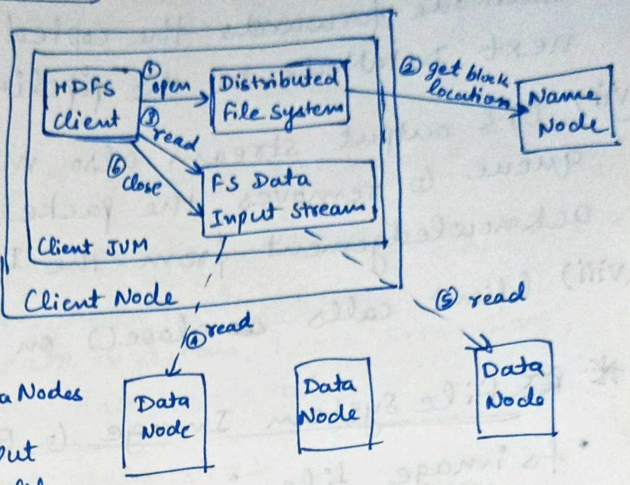
• HBase scales Horizontally, which means adding more machines to system to inc. performance.

③ Consistency :- • RDBMS provides strong consistency, which means all the nodes in system see the same data at the same time.

• HBase provides eventual consistency, which means different nodes may see different data at different times but eventually they will all converge same data.

④ Speed : HBase is faster than RDBMS when it comes to process large amount of data. in real-time.

Anatomy of file Read :-



(i) Client call open() on filesystem object

(ii) DFS calls the name node for location of blocks

(iii) Name Node validates request & for each block returns the list of DataNodes

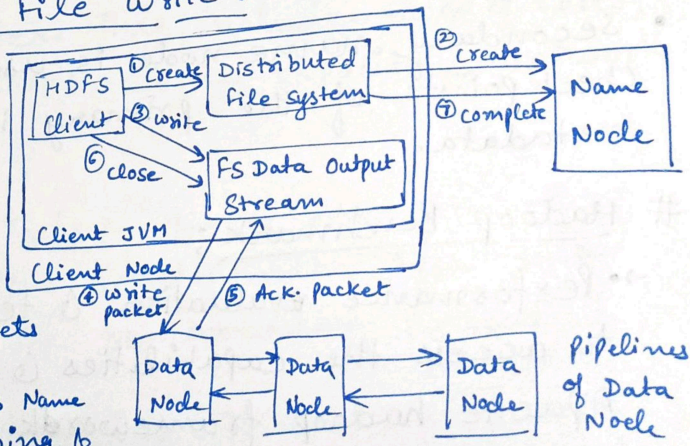
(iv) DFS returns an input stream that supports file seeks to the client.

(v) Client call read() on the stream

(vi) When the end of the block is reached, DFS input Stream will close the connection of DataNode, the find the best DataNode for next block.

(vii) Client calls close() on the stream.

Anatomy of a file Write :-



(i) Client creates file by create() method.

(ii) Name Node validates & process the create request

(iii) Split files into packets

(iv) Data streamer asks Name Node for block mapping & pipelines are created among nodes

(v) Data Streamer streams the packets to first DataNode.

(vi) DataNode forwards the copied packet to the next DataNode in the pipeline.

(vii) DFS output stream also maintains the ack. queue & removes the packets after receiving acknowledgement from the DataNode.

(viii) Client calls close() on the stream.

* File System Image & Edit Logs :- (Imp. Components of Name Node)

- fs image file is a persistent checkpoint of the file-system metadata.
- When a client performs a write operation, it is first recorded in the edit logs.
- The name node also has an in-memory representation of the file-system metadata, which it updates after the edit log has been modified.
- Secondary name node is used to produce checkpoints of the primary in-memory filesystem metadata.

Hadoop Benchmark :-

→ • Performance evaluation & testing process used to access the capabilities & efficiency of the Apache hadoop framework or other hadoop-based solutions.

- It is crucial for understanding how well hadoop-based systems can handle big data workloads

* Key components of Hadoop Benchmark :-

- Workloads :- Defines specific workloads or tasks to be executed on Hadoop cluster.
  Tasks can be → data ingestion, batch processing, realtime processing, data retrieval, etc.
- Data sets :- Different data sizes.
  Data sets range from small scale to large multiterabyte datasets.
- Metrics :- Various performance metrics are collected to evaluate the system efficiency.
  Metrics like throughput, latency, resource utilization & fault tolerance.
- Cluster Configuration :- Hardware & Software configuration of Hadoop cluster. This includes no. of cores, memory, CPU cores, storage capacity & network bandwidth.
- Benchmarking Tools :- Tools like TeraSort, TestDFSIO, GridMix & HiBench. These tools help automate the benchmarking process.
- Benchmarking Senarios
- Scaling Tests
- Comparative Analysis.

* Primary Goals of Hadoop Benchmarking:-

- Identify performance bottlenecks & areas for optimization.
- Validate the configuration & ensure that it meets the requirements of a specific use case.
- Evaluate the impact of changes, such as upgrading Hadoop components or adding nodes to cluster.
- Determine the overall efficiency & reliability of the Hadoop cluster under diff. conditions.

* Backup Node:- • It keeps an in-memory, up-to-date copy of the file system namespace.
• It is always synchronized with the active Namenode state.

Java Interfaces to HDFS :-

→ • Most Hadoop filesystems interactions are mediated through Java API.
• The filesystem shell, is a Java application using the Java file system class.

① HTTP :-

- The HTTP Rest API is exposed by WebHDFS protocol makes it easier for other languages to interact with HDFS.
- The HTTP interface is slower than native Java, so should be avoided for very large data transfer if possible.

- Two ways of accessing HDFS over HTTP:-
  - Direct HTTP requests to client.
  - via proxy, which accesses HDFS on the client's behalf using the usual DFS API.

② Web HDFS:-

- This concept is based on HTTP operations like Get, put, post & delete.
- For get, operation can be open, getfilestatus, liststatus.
- For put, operations can be create, mkdir, rename.
- For post, operation can be Append.
- For delete, operation can be Delete.
- The requirement of webHDFS is that the client needs to have a direct connection to namenode & datanode. via predefined ports.

* Advantages of WebHDFS:-

- Calls are much quicker than a regular "hadoop fs" command. you can easily see the difference on cluster with terabytes of data.
- It is helpful for non-java clients which needs access to HDFS.

* WebHDFS	Http FS
(i) It needs access to all nodes of the cluster & when some data is read, it is transmitted from that node directly.	(i) Here a single node will act similiar to a 'gateway' & will be single point of data transfer to the client node
(ii) Do not chok during large data transfer	(ii) It gets choked during large data transfer
(iii) Footprint is high.	(iii) Footprint is minimized.

* Hoop :-

- A rewrite of Hdfs Proxy
- It aims to replace Hdfs Proxy
- Hoop has a HTTP Rest API.
- Like Hdfs Proxy, It runs as external servers to provide a proxy service.
- As it is the proxy running outside HDFS, it can't take advantages of some features such as redirecting clients to the corresponding data nodes for providing data locality.
- It has advantages like, it can be extended to control & limit bandwidth, or to carry out authentication translation from one mechanism to HDFS's native kerberos auth.
- It can also serve proxy service to other file systems like Amazon S3.

(3) C :-

- Hadoop provides a C library called libhdfs that mirrors the Java FileSystem interface.
- It works using the Java Native Interface (JNI) to call a Java filesystem client.
- It is also a libwebhdfs library that uses the WebHdfs interface.
- C Api is similar to Java one, but still lags some features, so newer features might not be supported.

(4) NFS :-

- It is possible to mount HDFS on a local client's filesystem using Hadoop's NFSV3 gateway.
- Then, using unix utilities to interact with the filesystem, upload files, and in general

use POSIX libraries to access the filesystem from any prog. lang.

- Appending to a file works, but not the random modifications, since HDFS can only write to the end of a file.

Hadoop in the cloud :-

* Amazon Elastic Map/Reduce (EMR) :-

→ It is a managed service that allows you to process & analyze large datasets using the latest versions of big data processing framework such as Apache Hadoop, Spark, HBase & Presto on fully customizable clusters

→ Key features :-

- Ability to launch Amazon EMR cluster in minutes, with no need to manage node configuration, cluster setup or tuning.
- Simple & predictable pricing - flat hourly rate for every instance-hour.
- Ability to provision one, hundreds or thousands of computer instances to process data at any scale.
- Amazon provides demand based service, i.e. when job is done user can terminate the cluster & store data in Amazon S3. Hence paying for only the actual use time

* On Google Cloud :-

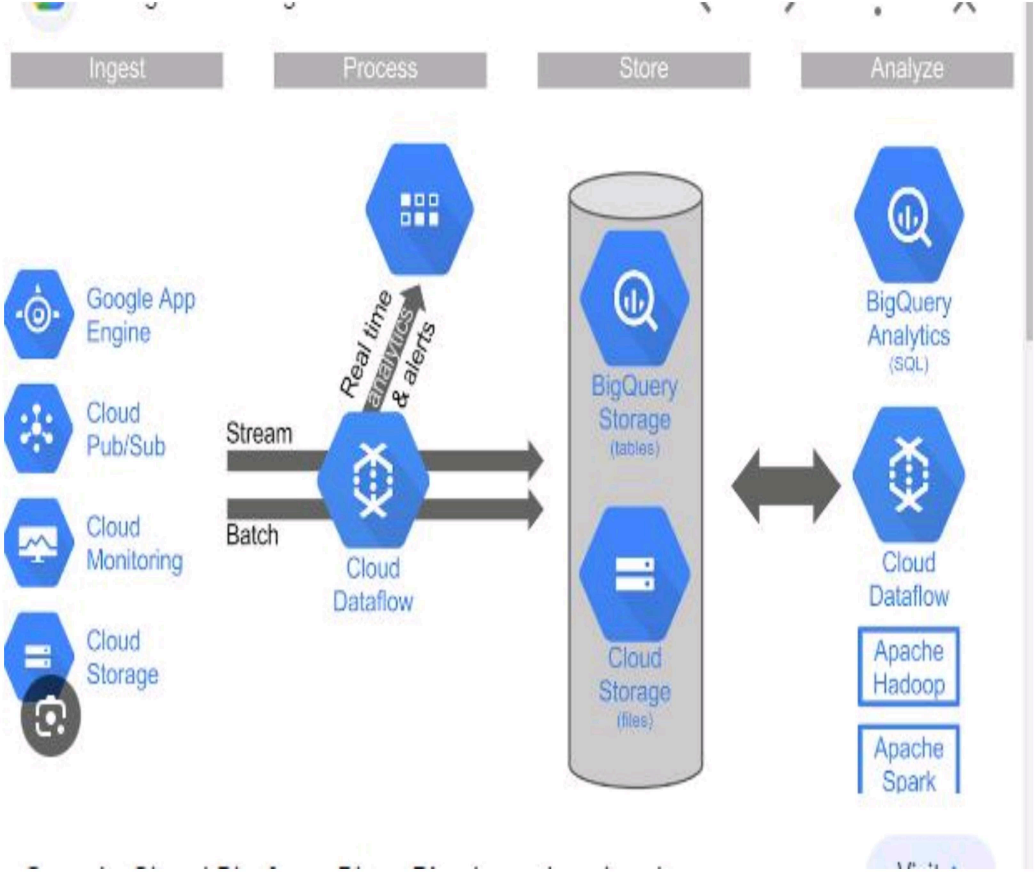
→ Google Dataproc is a fully-managed cloud service for running Apace Hadoop & spark clusters.

→ It provides enterprise-grade security, governance, & support & can be used for general purpose data processing, analytics & ML.

→ Dataproc uses cloud storage (GCS) to data for processing stores it in GCS, Bigtable or BigQuery.

→ Features of Dataproc :-

- Supports opensource tools such as spark or Hadoop.
- Lets you customize virtual machines (VM's) i.e. scale up or scale down acc. to needs.
- Provides on-demand clusters to help you reduce cost
- Integrates tight with google cloud service.


Ingest
Process
Store
Analyze
Google App Engine
Cloud Pub/Sub
Cloud Monitoring
Cloud Storage
Stream
Batch
Cloud Dataflow
Real time analytics & alerts
BigQuery Storage
(tables)
Cloud Storage
(files)
BigQuery Analytics
(SQL)
Cloud Dataflow
Apache Hadoop
Apache Spark

The Cost And Benefits Of Hadoop

Hadoop is ideal for batch processing terabyte to petabyte scale. Because Hadoop can store and process any type of data, from plain text files to binary files like images, or different versions of data collected over time, it's ideal if your use case requires you to store large volumes of unstructured data. On the surface, it appears to be a cost-effective way to handle these big data workloads because it uses commodity clusters, and can scale from a single cluster to thousands of clusters. And the software itself is open-source, so it's free. But on the other hand, Hadoop is not a single solution—it's a framework that requires you to build your data warehouse from the ground up. That means your solution can cost a lot of time and require specialized engineering resources. But once it's built, you can continue to scale to suit your needs.

How Hadoop Has Helped Evolve Big Data In The Cloud

Many Hadoop ecosystem projects are easily reusable in the cloud and have been widely adopted in cloud architectures. Most notably, Spark and Kafka were developed as part of the Hadoop ecosystem, but they have adapted even better to the cloud than to the environment for which they were originally built. Spark and Kafka work very well on the cloud alongside other scalable message bus systems like Google Pub/Sub, Amazon Kinesis, or Azure EventData Hub. There are also some good distributed SQL engines built for the cloud, including Amazon Redshift, Google BigQuery and Azure SQL Data Warehouse. These are better alternatives for running SQL workloads than Hadoop.

Can You Run Hadoop In The Cloud?

Technically yes, but cloud and Hadoop architectures were created for different purposes . While Hadoop was really meant for physical data centers, cloud environments were built to provide better elasticity and speed of source provisioning. Maintaining a large permanent cluster in the cloud is expensive, especially at scale. So to run a permanent HDFS you might need to use cloud storage solutions and other SQL engine alternatives. Another challenge is that there is really no universal solution to support security. While you might have a preferred cloud solution from one vendor, your Hadoop vendor may use another cloud platform. Using a mix of solutions can result in a steep learning curve, and possibly switching costs. And when it comes to ephemeral clusters in the cloud, governance and security become even more challenging. There is not yet an ideal solution to resolve them. So yes, it is possible, but the costs and barriers that are created when you run Hadoop in the cloud generally make alternatives more appropriate. Additionally, in many cases you lose the ability to embrace the fully-managed nature of cloud offerings and the capability they provide. Running Hadoop on the cloud, hence should not be viewed as a simple technical choice, but a choice based on overall capability and long term strategic vision.

Hadoop In IaaS

Pros

- Complete Control
- On-Demand Cluster Sizing
- Storage - Local or Cloud

Cons

- Only VMs managed for HA
- Administration required
- Clusters need to stay active

Hadoop In PaaS

Pros

- Fully managed – SLA bound
- Flexible resizing
- Pay-on-use
- Customization Options
- Deployed in minutes

Cons

- Forgo some control

Big Data as a Service

Pros

- Abstracted from clusters
- Automated resource alignment
- Easy to use interface and APIs
- Familiar languages

Cons

- Forgo complete control
- Priced extravagantly

Hadoop Cluster Modes

Hadoop can run in any of the following three modes:

Standalone (or Local) Mode

- → No daemons, everything runs in a single JVM.
- → Suitable for running MapReduce programs during development.
- → Has no DFS.

Pseudo-Distributed Mode

- → Hadoop daemons run on the local machine.

Fully-Distributed Mode

- → Hadoop daemons run on a cluster of machines.