Page No.
Date: / /

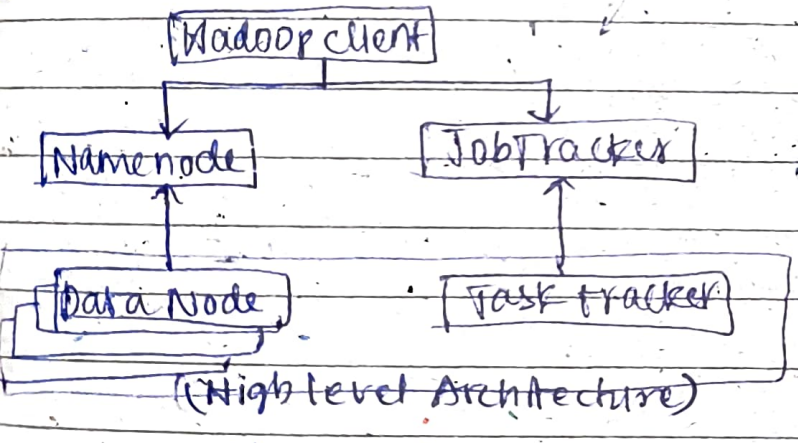
Big Data

→ Hadoop: → History, HDFS, components, analysis of data with Hadoop.

- Most well known technology used for Big Data is Hadoop.
- large scale batch data processing system.
- enables parallel data processing, Distributed cluster system, massively scalable applications, Job coordination.
- Dough cutting, Michael J. Cafarella (Apache software Foundation)
- FB, Google, apple, hp, twitter, IBM.
- Features: access to file systems, package contains Jar files, scripts, source code, documentation, etc.

core components → Hadoop distributive file system.
→ Map reduce

→ Distributed file system
→ Traditional hierarchal file org.
→ Single namespace.
→ Write once - read many.

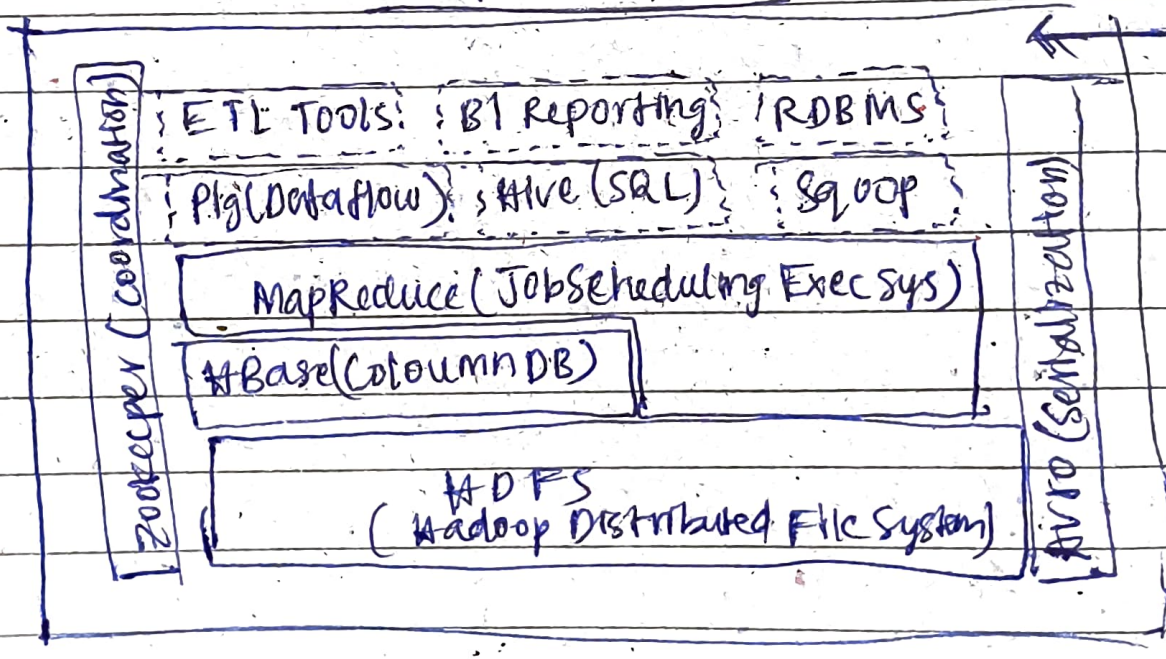


(High level Architecture)

→ Namenode Features: List of files, List of blocks for each file, List of Data Nodes for each block, file attributes, creation time, Records every change in metadata.

→ Data Node: Block server stores data in local file system, periodic validation of checksum, periodically sends a report of all existing blocks to Name node

Hadoop Ecosystem
(or)
Hadoop Architecture



(Hadoop)

- Hadoop is an open source software programming framework for storing a large amount of data and performing the computation.
- based on Java prog. and some native code in C and shell script
- Used for advanced level of analytics; ML, Data mining.
- Redundant, Fault-tolerant data storage, parallel computation.

Advantages	Disadvantages
• Ability to store large amount of data	• Not very effective for small data.
• High flexibility, cost effective.	• Hard cluster management
• High computational power.	• Has stability issues
• Independent tasks	• security concerns.

(HDFS)

[advantages: Inexpensive, immutable, stores data reliably, tolerate faults, scalable, block structured, can process large amount of data simultaneously]

[dis-advantages: not fit for small quantities of data, restrictive and rough in nature]

- HDFS splits files into blocks and sends across various nodes in form of large clusters.

(Namenode DataNode)

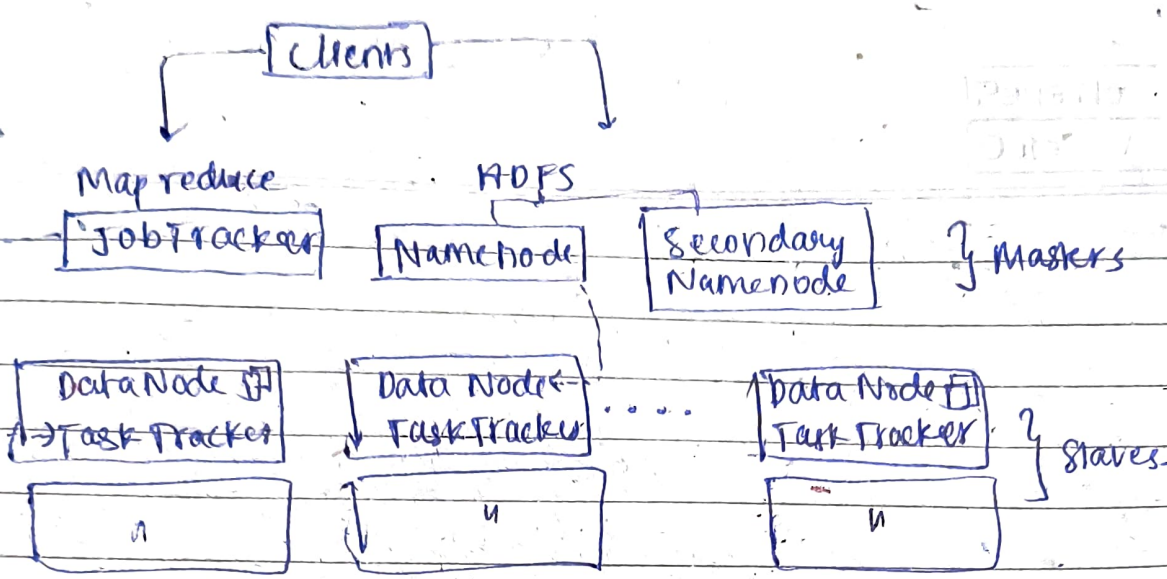
- HDFS cluster consists of a single Namenode, a master server that manages file system namespaces, regulates access to files by clients.

- DataNodes manage storage of the node.
- A file is split into one or more blocks & set of blocks are stored in DataNodes. → {read, write reqs; block creation, deletion}
  - DataNode stores each block of HDFS in seperate file.
  - Doesn't create all files in same directory
  - heuristics to determine optimal no. of files per directory.
  - Block report: list of all HDFS blocks ⇒ sends report to Namenode after filesystem startup.
- communication Protocols
- Hadoop cluster — client, Master, Slave

→ Task Tracker:
- Responsible to execute and manage tasks assigned by Job tracker
- constantly communicate with JobTracker the status of the Task.
- Heartbeat to Job tracker.

→ HDFS:
- A scalable, Fault tolerant, High performance distributed file sys.
- Write once - read many | Min 3 data nodes
- Data divided into 64MB (or) 128 MB blocks, each block replicated 3 times (default).
- NameNode holds metadata, files are broken up and spread over Datanodes

→ Hadoop Map Reduce:
- Software framework for distributed computation. / prog. model for data processing.
- Input | Map() | copy/sort | Reduce() | output.
- Job tracker schedules and manages jobs.
- Task tracker executes individual map() and reduce() on each cluster node.



- Don't have to worry about handling
  - parallelization
  - data distribution
  - load balancing
  - fault tolerance
  - process huge amounts of data

*JobTracker: Responsible for accepting jobs from clients, divide jobs into tasks, assigning to worker nodes.

*TaskTracker: manages the execution of tasks currently assigned to that node. Each has fixed slots (two maps and two reduce).

- four transformations of Data:
  - Transformed from i/p files and fed into mappers.
  - Transformed by mappers.
  - sorted, merged and presented to reducer.
  - Transform by reducers and written to output Files.
- Developing MapReduce application: map, reduce functions, ideally with unit tests.

→ Fault tolerant, Data Locality, cost-effective solution, Scalability, parallel programming.

→ common frameworks of Hadoop:

Hive, Drill, Storm, Spoork, Pig, Tez.

→ Hive: uses HiveQl for data structuring and for writing complicated MapReduce

- Data warehouse infrastructure that provides data summarization and adhoc querying on top of Hadoop.
- Metastore: Table/partitions properties, Metadata many SQL backend
- Hive Query Language: Basic SQL, equijoin, multitable insert, batch query, Multi-Group by.

→ Pig: a SQL-Like Language and performs data transformation of unstructured data.

- High-level data-flow language and execution framework for parallel computation.
- Simple to write MapReduce program, abstraction.

→ Sqoop: a tool designed to help users import large data existing Rel. databases into their Hadoop clusters.

- Automatic data import, SQL to Hadoop, easy import.

→ zookeeper:

- A high performance coordination service for distributed applications.
- centralized service for info, config^n, naming, providing group services
- All servers store a copy of data, leader is elected at startup.

→ Avro: data serialization system that provides dynamic integration with scripting languages.

- Avro data - Expressive, smaller and Faster, Dynamic.
- Avro RPC - Leverage versioning support, cross-language access.

→ Big data Analytics: a process to extract meaningful insight from big such patterns, trends, etc..

- Structured Data: data that has a proper structure.
  Ex: Databases, CSV files; excel, etc..
- Semi-structured: neither structured, nor unstructured; no proper format, that it can be stored in a relation.
  Ex: emails, log files, word documents, etc..
- Unstructured: No format, doesn't have keys, or labels. usually large data, videos, audio etc..
- Tools: MongoDB, Hadoop, Talend, Cassandra, Storm, Spark
- Risk Management, Improve customer experience, product development and innovations, better decision making

→ Education: Enchance results, better grading system, customized programs, reduce dropouts.

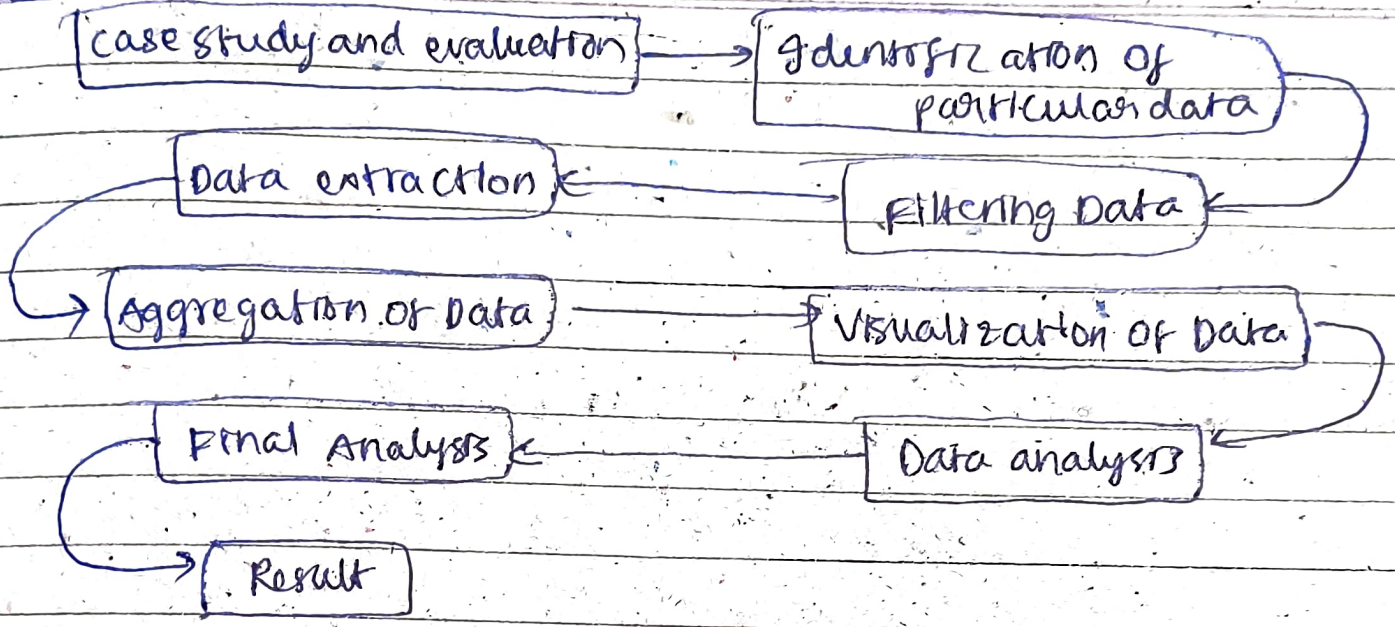
→ Healthcare: Health tracking, Machinery improve, fraud detection and prevn, Real time alerts.

→ Weather Forecasts: Flooding, storms, Snow/rain, power lines, windspeeds, roads/bridges, flights cancelled.

→ Agriculture: Boost productivity, predict yields, Food safety, savings.

→ Manufacturing: Product quality, tracking, planning.

Process of Big data Analytics:



→ Google Big Data:

google developed several open source tools and techniques in Big data ecosystem. With help of these, google can fetch right answer or info in millseconds.

- Sorting tools, synonyms, Knowledge graphs, google translate, tracking cookies, realtime data feeds, etc.

→ IBM's Weather Forecasting:

IBM's Deep Thunder, forecasts extremely specific locations, such as single airport.

→ Hadoop Ecosystem:

- Map Reduce for Indexing, learning (Machine learning)
- HBase for storage and fast access
- Storm for incremental update
- Rel DB for most recent derived data
- API facade for input; API for querying learning.

→ Main components of Hadoop ecosystem:
HDFS, MapReduce, YARN, Hadoop Common.

→ HDFS, YARN: Yet another resource negotiator, MapReduce; PIG, HIVE, HBase: NoSQL Database; Mahout, Spark MLlib: Machine learning algo libraries; Solar, Lucene: searching/indexing, Zookeeper: Managing cluster, Oozie: Job Scheduling.

→ Apache HBase:
• It's a NoSQL database which support all kinds of data, thus capable of handling anything of Hadoop Database.

→ YARN:
• Yet another resource negotiator, manage resources across the clusters. Scheduling and resource allocation.
i) Resource Manager
ii) Nodes Manager
iii) Application Manager
• privilege of allocating resources. CPU, memory, bandwidth per machine.

→ Hadoop 1 limitations:
Scalability: Max cluster size ~ 5,000 Nodes:
Max concurrent task ~ 40,000.
Coarse synchronization in JobTracker.
availability, Hard partition of resources into map and reduce slo
Lacks support for Alternate paradigms and services.

⇒ Key features of Map Reduce:

RDBMS	Hadoop
→ Giga bytes (Terabytes)	Petabytes (Exabytes)
→ Interactive and Batch	Batch - Not Interactive.
→ Read/write many times	Write once Read many
→ Static schema	Dynamic schema
Nonlinear	Linear
can be Immediate response	Has Latency

⇒ Hadoop Streaming:

Feature of Hadoop distribution, that allows users to write Hadoop MapReduce programs using C++, Python, Ruby, Perl, etc.

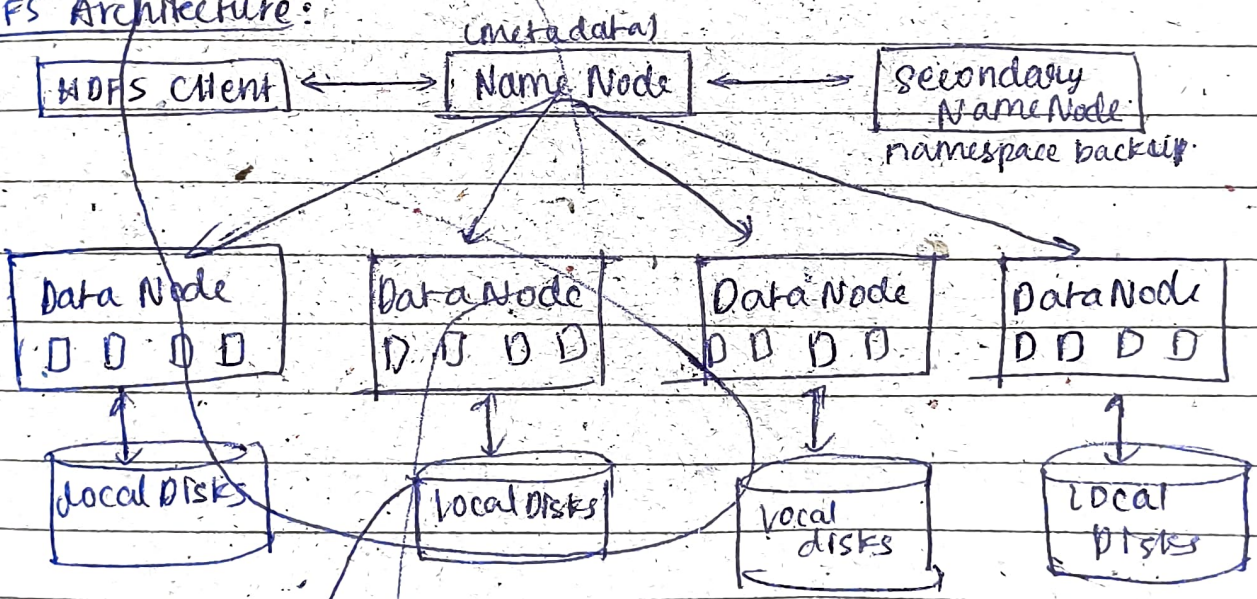
→ Mapper, reduce, Driver

Unit-3

→ Introduction to HDFS:

- Is a distributed filesystem that is fault tolerant, scalable and extremely easy to expand.
- primary distributed storage for Hadoop applications.
- provides interfaces for applications to move themselves closer to data.
- There a two main components:
  - NameNode: Is heart of an HDFS filesystem, it maintains and manages the namespace metadata. (like an index page)
  - DataNode: This is where actual data is stored by HDFS.

→ HDFS Architecture:



→ Unique features:

- Failure tolerant: data is duplicated across multiple DataNodes to protect against machine failures. (default 3 nodes)
- Scalability: data transfer happen directly with DataNodes. So read/write capacity fairly scales.
- Space: Just add more dataNodes for mode necessary data.
- Industry standard: Other distributed apps are built on top of HDFS (HBase, Map-Reduce).

→ Data Organization:

- Each file is split into data blocks.
- Each block is stored on one or more nodes.
- Block placement policy:
  - First replica is placed on local node.
  - Second on different rack.
  - Third on same rack as second replica.

→ Read operation:

open() → [get block locations from NameNode] → read() → (packets) read() read() read() → close()

→ Write operation:

create() → create() → write() → write packet → ack packet → close() → complete

→ HDFS Security:

- Authentication to Hadoop:
  - simple - insecure way of using OS username to determine hadoop identity.
  - kerberos - authenticating using kerberos ticket

[hadoop.security.authentication = simple|kerberos]

- read(r), write(w), execute(x)
- owner, group and mode.
- dfs.namenode.acls.enabled = true
  - ↳ used for implementing permissions that differ from natural hierarchy.

→ HDFS Defaults:

- Blocksize = 64 MB, Replication Factor = 3, Web UI port = 50070

→ Interfaces to HDFS:

Java API (DFS), C wrapper (libhdfs), HTTP protocol, WebDAV protocol, Shell commands

→ User commands:

① hdfs dfs -ls

② hdfs dfs -du -h / [file loc]

③ copy data:

(to hdfs) { hdfs dfs -mkdir tdata
hdfs dfs -copyFromLocal <file> tdata
hdfs dfs -ls -R
↳ [-R is used for listing every file]

(to local) → hdfs dfs -copyToLocal <file> <file>.hdfs

④ hdfs dfs -getfacl tdata/genera.csv

⑤ hdfs dfs -stat "%r" <file>

⑥ write:

echo "- - - -" | hdfs dfs -put - <files>
hdfs dfs -ls -R
hdfs dfs -cat tdataset/tfile.txt

⑦ remove:

hdfs dfs -rm tdataset/tfile.txt

⑧ hdfs dfsadmin -report
-printTopology
-getDatanodeInfo

⑨

Page No.
Date: / /

→ Hadoop Security :

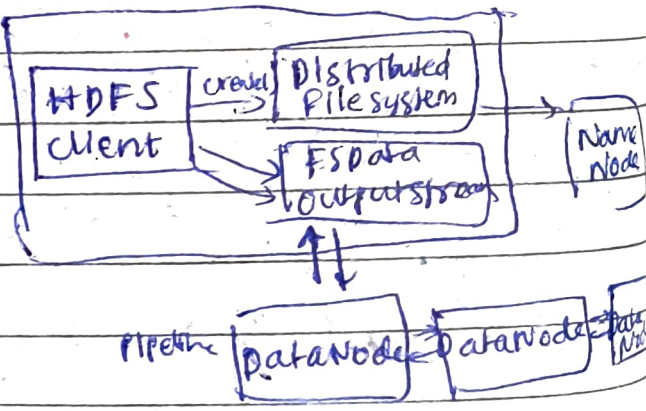
In order to secure from unauthorized access of the data stored in HDFS, Kerberos is used to provide security authentication of map/reduce users, HDFS clients. etc.

- Add Kerberos based authentication to NameNode and JobTracker
- Determine user's group membership on NameNode and JobTracker
- Protect MapReduce system directory from users.
- authorization to MapReduce.
- User to service authentication, service to service Authentication.
- users authenticate via kerberos.
- MapReduce Jobs can obtain delegation tokens for later use
- When clients are reading/writing an HDFS file, Namenode generates a block token that will be verified by DataNode

---

→ Anatomy of a File write:

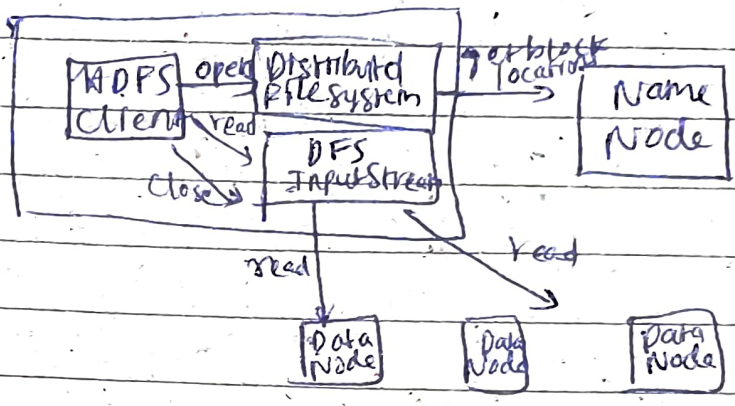
- Client create file by create()
- NameNode validates & processes
- Split file into packets (DataQueue)
- Pipeline created
- Data streamer streams packets
- DataNode forward packets
- DFSoutStream also maintains que of ack.
- Client calls close().



Page No.
Date: / /

→ Anatomy of a File Read:

- Client calls open()
- NameNode gives locations.
- NameNode validates req & returns
  - lists DataNodes
- DFS returns an InputStream
- Client calls read()
- At end of block, DFS InputStream will close connection.
- close() by client.



~~→ FSDataInputStream:~~

→ open() : public FSDataInputStream open(Path f)

write Data : public FSDataOutputStream create(Path f)

→ Map Reduce — Hive — Pig

compiled lang — SQL like open