**Advanced Database Management**

hadoop and its ecosystem

Under the Guidance of Prof. Yann Chang

University of Illinois at Chicago

IDS 521 – Term Paper

**Submission By-**

Shivam Duseja

Pawanjeet Kaur

Karansinh Raj

Table of Contents

[Executive Summary 3](#_Toc38246342)

[Hadoop – Introduction 4](#_Toc38246343)

[1. Origin and Overview 4](#_Toc38246344)

[I. What is Hadoop 5](#_Toc38246345)

[II. Hadoop Architecture 5](#_Toc38246346)

[III. Characteristics of Hadoop 7](#_Toc38246347)

[IV. When should be use Hadoop? 8](#_Toc38246348)

[V. When we should not use Hadoop 8](#_Toc38246349)

[2. Key Technologies within the Hadoop Ecosystem 9](#_Toc38246350)

[I. HDFS (Hadoop Distributed File System) 9](#_Toc38246351)

[II. HDFS Architecture 9](#_Toc38246352)

[III. Map Reduce 10](#_Toc38246353)

[3. Hadoop – Projects 12](#_Toc38246354)

[I. Apache Pig 12](#_Toc38246355)

[II. Apache Hive 14](#_Toc38246356)

[III. HBase 15](#_Toc38246357)

# Executive Summary

It has been correctly said by Clive Humby – ‘Data is the new Oil’. In the recent past, we have seen an exponential increase in the amount of data and the evolution of Big Data to extract valuable information from petabytes of data. This large amount of data further led to the development of distributed and parallel computing solutions. Hadoop’s capability of processing, analyzing, and transforming Big Data has attracted a lot of customers/users making it – one of the biggest players in the market. The two main components of Hadoop – HDFS and MapReduce have enabled parallel computation by partitioning data across thousands of hosts. One of the major advantages of Hadoop is that it offers easy scalable solutions at a comparatively lower cost for petabytes of data. In order to provide a SQL-like interface for data processing on file systems, Hadoop is integrated with a data warehousing solution – Apache Hive – that is built on top of Hadoop. Apart from the SQL compatibility, Hadoop is also integrated with Apache Pig, that provides a high-level platform for running programs using – Pig Latin language.

As Apache does not provide any kind of support – this may lead to different stability and security concerns, hence, there arises a need for different vendors to help distribute Hadoop across platforms. A few vendors that support and ensure the reliability of Hadoop are – Cloudera, MapR, Hortonworks etc.

# Hadoop – Introduction

## Origin and Overview

Doug Cutting and Mike Cafarella started Hadoop in the year 2002, while they both were on Apache Nutch project. Originally Apache Nutch project was the process of building a search engine that could index around 1 billion pages.

As per Doug Cutting, Hadoop was a made-up name and not an acronym. As per him – *‘The name my kid gave a stuffed yellow elephant. Short, relatively easy to spell and pronounce, meaningless and not used elsewhere: those are my naming criteria.’* The Nutch developers created a working crawler and a search system, however, the major problem in front of them was that their architecture was not able to scale to billions of pages on the web. Google’s paper on their distributed filesystems (GFS – Google’s distributed filesystem) in 2003 turned out to be a silver lining for the Nutch developers as this paper could solve their problem of storing very large files, however, this was only a half solution to their problem. The other half solution was addressed when Google published one more paper on the technique MapReduce in 2004. Both the techniques – GFS and MapReduce were just on paper with Google.

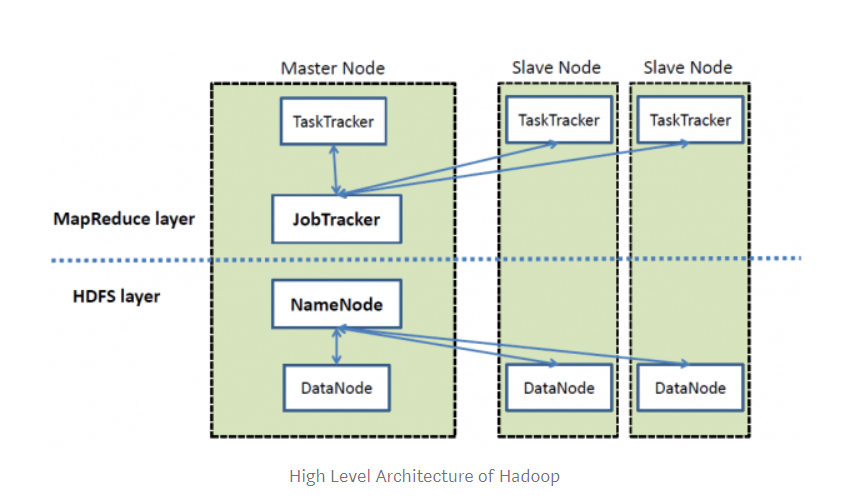
Doug Cutting, from his prior work on Apache Lucene was aware that the best way to spread a technology to more people is to make it open source. He then started working with Mike Cafarella on implementing GFS and MapReduce technique as an open-source project.

In January 2008, Hadoop was released by Yahoo as an open source project to Apache Software Foundation (ASF). Further in July 2008, ASF successfully tested a 4000-node cluster with Hadoop.

### What is Hadoop

Apache Hadoop is an open source framework that is capable of storing and processing large datasets ranging from gigabytes to petabytes. Instead of using a conventional way of storing and processing data in a single large computer, Hadoop allows clustering of multiple computers which helps in parallel processing thereby analyzing massive datasets more quickly. The Hadoop ecosystem has grown significantly since its inception and today this ecosystem offers many tools and application to manage – store – process – analyze big data.

### Hadoop Architecture



Major Components of the Hadoop Architecture:

* Master – Slave mechanism
* Made up of 2 major components – HDFS and MapReduce
* **Hadoop Distributed File System (HDFS):** The HDFS component replicates data across multiple nodes thereby offering a distributed storage. Unlike other regular file systems, HDFS creates multiple blocks of data by automatically splitting it and storing it across different data nodes. This helps in ensuring higher availability and fault tolerance.
* **MapReduce:** The MapReduce component provides an analysis system that performs complex computations on large sized datasets. These complex computations are broken into multiple tasks that are assigned to individual slave nodes. Further the algorithm takes works on coordinating and consolidating the results.
* **Master:** The master node consists of the job tracker and NameNode components:
  + **NameNode** – The NameNode component holds all the useful information for operation of Hadoop cluster. It holds information about all the nodes and files within the cluster and other building blocks of these files and their locations in the cluster.
  + **Job Tracker** – The job tracker component is responsible for tracking individual jobs/tasks assigned to all the nodes and it further coordinates information exchange and results.
* The **Slave** component in the framework contains the task tracker and a DataNode component:
  + **DataNode:** Holds the data
  + **Task Tracker:** Responsible for running the tasks assigned to it.
* The above framework has no dependency on the physical location of the server.

### Characteristics of Hadoop

* **Faster:** Hadoop is extremely good in processing high volumes of data given the fact that it has the capability of processing batches in parallel. This performance is much better than a single thread server on a mainframe.
* **Distributed Processing:** Data in Hadoop is processed in parallel on a cluster of nodes. This is provided – the data is stored in a distributed manner in HDFS.
* **Fault Tolerance:** As per the Hadoop architecture, the data that is sent to one node is replicated across other nodes within the same cluster. Hence, in case the original node fails in processing the data – the other nodes would process it.
* **Reliability:** Due to Hadoop’s characteristics of data replication across nodes, the data is reliably stored on clusters thereby protecting it from any kind of machine failures.
* **Scalable Solution:** Hadoop allows users to run and process large data sets (thousands of petabytes of data) by distributing this data across multiple parallel operating servers. Apart from that, Hadoop also provides the capability to add nodes while processing, without any downtime. This is commonly known as *‘Hardware horizontal scalability’*.
* **Flexible:** Hadoop is highly capable in dealing with structures/unstructured, encoded, or formatted types of data.
* **Availability:** Data replication of Hadoop – makes it available and accessible from different paths even if the hardware machine fails.
* **Ease of use:** As Hadoop takes care of the distributed computing itself – it is relatively easier to use.
* **Cost Effective:** Hadoop is a cost-effective storage solution for very large data as it runs on a cluster of commodity hardware.
* **Locality of Data:** While submitting a new MapReduce algorithm – Hadoop moves the algorithm to the same data cluster instead of moving data to the location where algorithm is submitted.

### When should be use Hadoop?

* **Volume and Diversity of Data:** Hadoop should be used - when the data volume is very large (gigabytes or petabytes) and is coming from a different source in a variety of formats.
* **Lifetime availability of Data:** Hadoop does not limit the size of different clusters and also provides a functionality of adding nodes to a cluster, if required. This feature stands out - if we want this data to be live and available forever.
* **Multiple Frameworks:** Hadoop is easily integrable with different technologies/tools like R, Python, Spark, HBase, MongoDB, NoSQL databases, etc.
* **Design for Future Planning:** Hadoop provides a functionality to plan for future scalabilities. Before using Hadoop infrastructure – the first step should be to understand the complexity of our processed and the rate at which data is expected to grow.

### When we should not use Hadoop

* **Processing smaller data volumes:** In order to process smaller data volumes – tools like MS Excel, MySQL should be used as Hadoop would be slower and costlier in this scenario. One way to use multiple smaller datasets is to combine them (provided they are of same format and type) and create a large dataset. Hadoop can then be used to store and process this large dataset.
* **Replaceable Infrastructures:** We can run multiple processes with Hadoop; however, we cannot simply replace existing databases. There are generally different tools for different jobs. For e.g. data can be stored and processed in the Hadoop ecosystem – which can then be passed into other relational databases and other BI tools like Tableau, MicroStrategy etc.
* **Sensitive Data:** When the data is highly sensitive, it cannot be directly moved to Hadoop because of the possibility of data breach. If Hadoop has to be used, data should be encrypted first before loading it into the Hadoop ecosystem.

## Key Technologies within the Hadoop Ecosystem

### HDFS (Hadoop Distributed File System)

HDFS is a distributed file system that follows a network-based approach to store different files across various systems. HDFS is designed to work on product hardware having low costs that can handle large volumes of data. This is a highly self-healing and fault tolerant distributed file system.

### HDFS Architecture

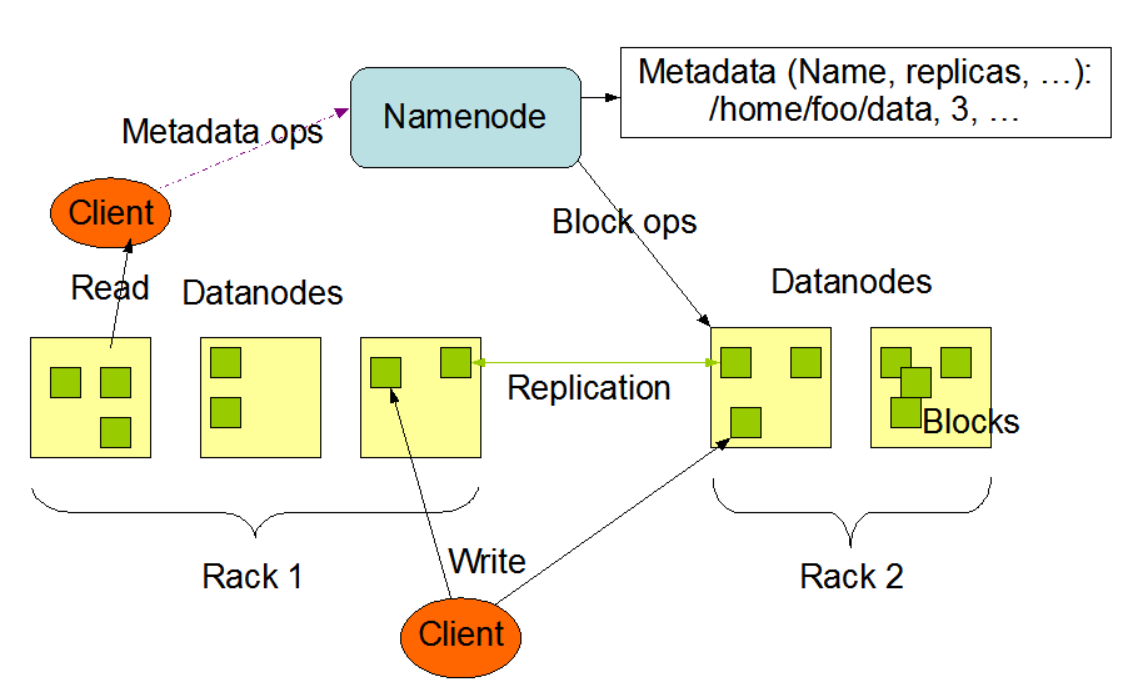


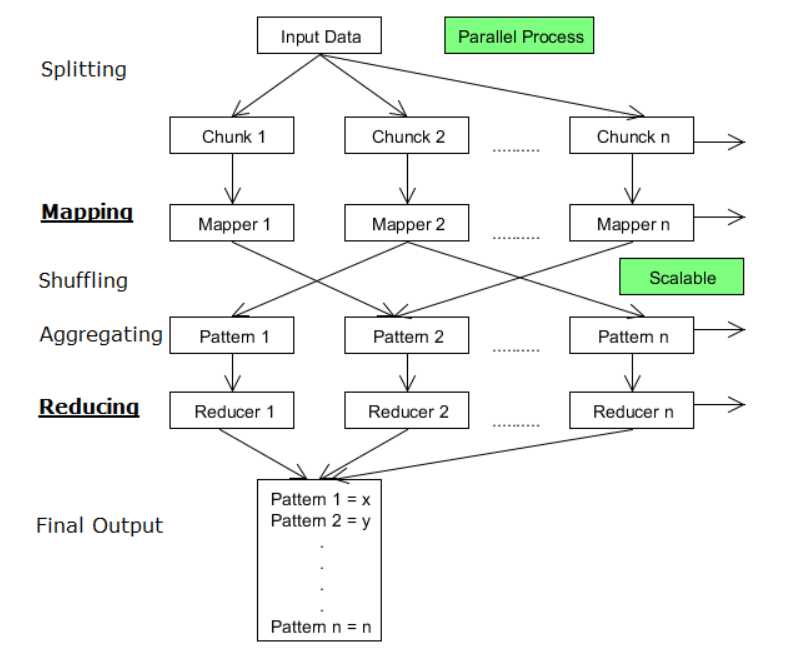
Figure on the previous page shows the HDFS architecture – that follows a master/slave mechanism. HDFS architecture is designed in such a way that the master daemon contains NameNode, secondary NameNode and job tracker whereas the slave daemons contains the data node and task tracker. The NameNode is responsible for keeping the directory tree of all files in the system and tracking the location of file within the cluster. It also maintains the file system and the file system further contains all the information about the meta data. This is in general a single point of failure for HDFS cluster and when it goes down – the entire file system goes offline. However, when the primary node goes down, the secondary node comes into picture. This helps to maintain the file size containing HDFS modification logs within certain limits at name node.

The task tracker monitors and instantiates the individual maps and thereby reduces work. Every task tracker resides on top of the data node and has a certain slot allocated to take care of different scheduling tasks in the form of MapReduce jobs. This sits on top of the NameNode that manages MapReduce tasks and distributes individual tasks to the task tracker machine. The data node is responsible for storing the data in Hadoop File System in the form of HDFS blocks that has a default size of 64 MB. Initially the service is established when the data node connects to the name node which further responds to the requests back from the name node for different file system operations.

### Map Reduce

The Map Reduce algorithm is a parallel programming model that is used for processing huge volume of data. It helps in writing applications to process high amount of data in parallel across different large clusters. It further provides automatic parallelization, I/O scheduling, monitoring, distributed fault- tolerance and status updates. As the computational process generally occurs on both structured and unstructured datasets, thus, making it reliable, fault-tolerant that supports thousands of nodes.

#### Map Reduce Algorithm



A general Map Reduce job is further divided into 4 phases:

* **Input Split:** The map reduce input split is divided into fixed sizes chunks which is a fixed piece of input being consumed by a single map.
* **Mapper:** Data residing in the input split is passed on to the mapper function to produce mapper output values. An intermittent output is produced in this section depending upon the business/user requirement.
* **Shuffling/Aggregation:** The output from the mapper function is fed to the shuffling/aggregation phase. This phase further consolidates the data records for processing in the reducer function
* **Reducer:** The shuffled/aggregated output is fed to the Reducer for final processing. This function returns a final output summarizing the job/task at hand (for example – determining frequency of words, clustering etc.)

In general, this algorithm cannot control the order in which the mappers or reducers are running, however, a reducer job would not start until and unless all the mapper jobs are complete. These algorithms/programs are usually written in scripting languages (Java, Python) using streaming APIs.

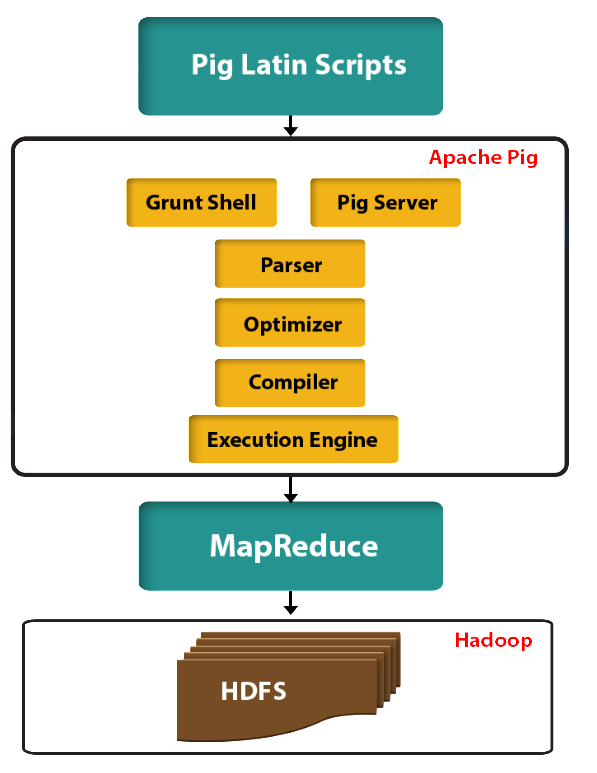
## Hadoop – Projects

### Apache Pig

In order to analyze data using the Pig script in Hadoop – Apache Pig is most commonly used. Apache Pig, being a high-level data processing language, offers different types of data types, operators, and data processing operations. The developers must write a Pig script using the Pig Latin language in order to perform different tasks and these tasks can be executed using different execution mechanisms like Embedded and Grunt Shell.

Once the scripts have executed, they undergo different transformations in order to produce the required/desired output by the Apache Pig framework.

As per the Pig architecture, these scripts are transformed into a number of Map Reduce jobs making the process much simpler and faster. A high-level architecture of Apache Pig is displayed on the page below –



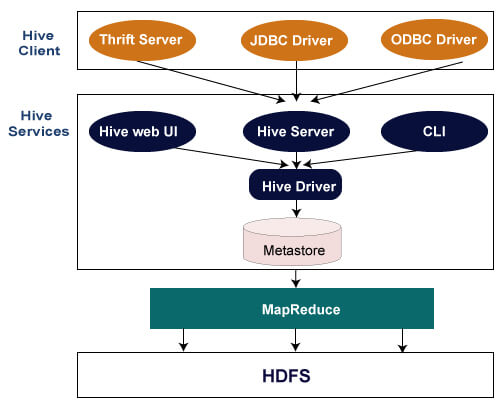
#### Major Components of the Pig Architecture

* **Parser:** As per the architecture, the parser is responsible for handling the pig scripts. Different types of validations like – syntax, types checks, etc. are performed by the parser. Directed Acyclic Graph (DAG) is the output generated by the parser which basically represents the statements of Pig Latin with different logical operators.
* **Optimizer:** Various logical optimizations like pushing, projecting, etc. are performed by the logical optimizers (whose input is DAG).
* **Compiler:** Once the DAG is optimized into a logical plan, the compiler compiles it into a series of MapReduce jobs which is further used by the execution engine.
* **Execution Engine:** The MapReduce jobs produced in the compiler step are further fed in a sorted order to Hadoop. These MapReduce jobs/tasks run on Hadoop in order to generate the desired/required results.

### Apache Hive

In order to support SQL-like query processing, Hive is a data warehousing solution built on top of Hadoop. These queries within the Hive ecosystem are known as HiveQL or HQL and are compiled into Hadoop based Map Reduce jobs. HQL also offers a feasibility to connect different customized map scripts to SQL-like queries.

#### Hive Architecture and its Components



The above figure shows a high-level architecture view of Hive and its components. From the figure, we can see the flow in which a simple query is processed in Hive which is finally processed by the MapReduce algorithm.

Major components of Hive are as follows:

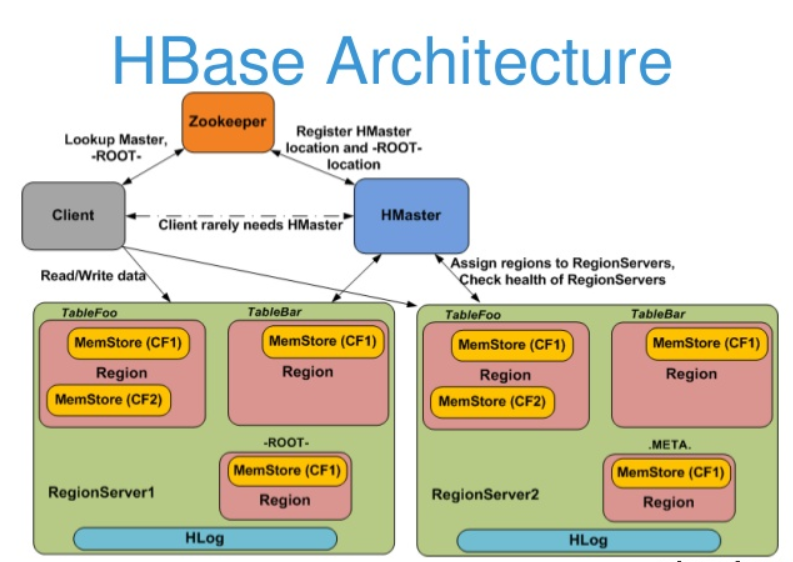
* **Hive Client:** Hive is one of the most versatile languages in a sense that it supports applications written in different languages like – Java, Python, C++ etc. using JDBC/ODBC and Thrift drivers. Hence, a Hive application can be developed easily written in the language of their choice.
* **Hive Service:** Apache hive provides various kinds of services like web interface, CLI, in order to perform/process queries in a more user-friendly manner.
* **Processing and Resource Management:** The MapReduce framework/algorithm is used internally to execute the queries and process the data.
* **Storage:** As Hive is a data warehousing solution built on top of Hadoop, it uses the underlying HDFS architecture for distributed storage and processing.

### HBase

The datasets have become sparser with the increased application of Big Data. HBase is a columnar database with a Hadoop Distributed File System most suitable for scenarios wherein the datasets are sparse. HBase is not a relational database and does not support SQL-like structured queries to analyze or process data. These HBase applications are generally written in Java which is pretty similar to the MapReduce algorithm.

A simple HBase system is made up of a set of tables. Similar to a traditional database, each table is made up of rows and columns, having a primary key, however, all the access attempts to the HBase tables should use this primary key.

The figure on the below page shows a high-level view of HBase architecture and its components:



The Major components of the HBase architecture are –

* **HMaster:** HBase HMaster is the process that performs load balancing by assigning regions to region servers within the Hadoop cluster. Major tasks taken care by the HMaster are –
  + Monitoring and Managing the Hadoop cluster
  + All the different kinds of DDL operations are managed by HMaster
  + Responsible for administrative tasks (interface for creating, deleting, updating tables)
  + It is also responsible for the changes at the schema level as well as changes involving any metadata operations.
* **Region Server:** Different client requests like – read, write, delete, update and handled by working nodes (Regional servers). This regional server process runs on every node within the Hadoop cluster. This includes the following components –
  + **MemStore:** This refers to the write cache – that stores new data which has not been written to the disk yet. Each column family has MemStore within a region.
  + **Block Cache:** This refers to the read cache. As per the architecture, the most frequently read data is stored in the read cache, however, when the block cache is full, the recently used data is disposed of.
  + **WAL (Write Ahead Log):** Any new data that is not permanently written is stored in the form of a file (WAL)
  + **HFile** is the storage file responsible for saving rows on a disk as sorted key values.
* **Zookeeper:** It is used as a distributed coordinating service for different regional assignments that is used to recover any regional server crash by loading it into other functioning region servers. The major responsibility of Zookeeper is to act as a centralized monitoring server thereby maintaining configuration and distributing synchronization information.

In order to approach regions, the Zookeeper must be contacted first. The Region servers and HMaster are registered with the Zookeeper service whereas the clients require zookeeper quorum to connect to different region and HMaster servers.

References:

<https://www.geeksforgeeks.org/hadoop-history-or-evolution/>

<https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html>