THE INTERNATIONAL JOURNAL OF SCIENCE & TECHNOLEDGE

An Innovative Name Node Fail Safe Architecture for HDFS

Yojana Nanaware
Student, N. K. Orchid College of Engineering & Technology, Maharashtra, India
Dr. Suhas D. Raut
Associate Professor, Dean of IT Department,
N. K. Orchid College of Engineering & Technology, Maharashtra, India

Abstract:
Widely used by organization, in data intensive computing is Hadoop. Client applications of Hadoop are online with unpredictable growth rate requires high reliability and scalability of system. As HDFS namespace grows, it additional memory demands are satisfied by Namenode servers vertical scalability which creates performance and availability issue. This project provides a new HDFS architecture present cloud storage scheme utilizes multiple Name Nodes, has been proposed to address the issues of scalability and availability of Name Nodes.

Keywords: *Hadoop, HDFS*

1. Introduction

Phenomenon growth in web services and internet based applications, they handle the large amount of data by using traditional techniques, relational database management systems etc. But still the data growth is in unstructured form. So to provide high level of reliability, scalability, efficiency and availability, the Hadoop cluster architecture is a good platform. Hadoop cluster performs on voluminous data with the help of three parts: Client machines, Master nodes and Slave nodes [1].

2. Literature Survey

There are some existing solutions [5], enhancing the availability of HDFS by recovering long time Name Node. These solutions are the Hot Stand By, Apache Hadoop HDFS Architecture, and HDFS Federation.
In Hot Stand By, HDFS has a Backup Name Node that does the data backup and has the potential to be a "Hot" Standby. This consists two Name Nodes i.e. one is Active Name Node and another is Hot Stand By and Zookeeper ensemble i.e. preferably 3 Zookeeper servers.
Apache Hadoop HDFS Architecture is a block-structured file system having predetermined size blocks stored across cluster of one or more several machines. It follows Master/Slave Architecture, where one Name Node as Master and number of Data Nodes as Slaves comprises in cluster [1].

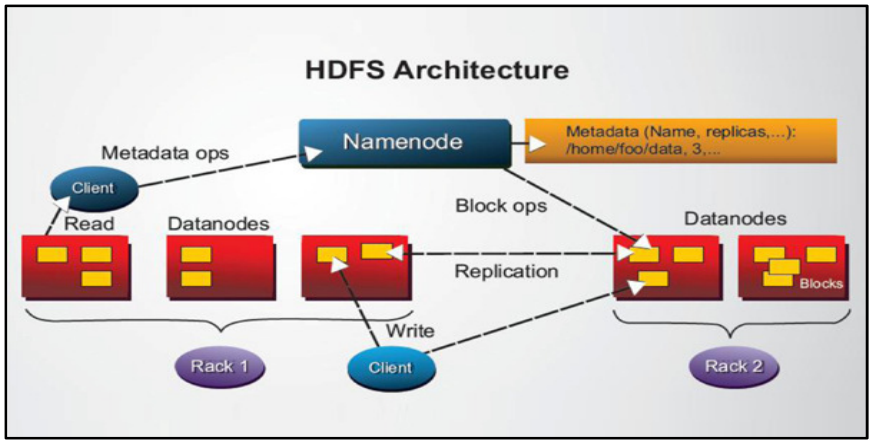

Figure 1: Basic HDFS Architecture

HDFS Federation is expansion of HDFS architecture enables block storage layer with the help of namespace layer performs block management and storage layer stores blocks on local file system and allows read/write access. It allows horizontal scalability and

Name Nodes are don't require inter coordination. But HDFS Federation is also backward compatible, so the single Name Node configuration will also work without any changes.

3. Problem Formulation

As the current HDFS Federation architecture is backward compatible, with simple design, it's require to present the cloud storage scheme to utilize multiple Name Nodes provide high scalability, high availability, and better fault tolerant with reliable HDFS architecture.

4. Proposed Architecture

Our proposed architecture contains basic HDFS framework but with multiple Name Nodes performs coordination which is very important among each Name Node servers while the node server goes down and comes up or while new file and directories or objects/ blocks are added to the namespace [2]. Also there is NLS (Name Node Location Service) module used to locate Name Node for clients. The Name Nodes in architecture still manages Datanodes, whereas our proposed architecture using clustering techniques to improves Namenodes scalability [4][6].

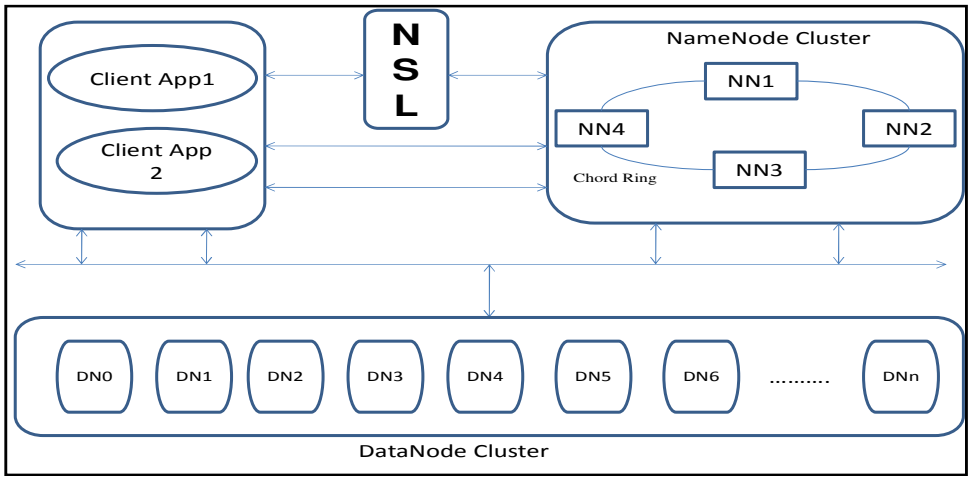

Figure 2: Proposed System Architecture

In Figure.2 Name Node module and the Data Node module are standard HDFS modules. A Name Node location service (NLS) module is added to the system. It allows multiple Name Nodes coexisting in the system and each Name Node managed the Data Nodes. NLS module collected all of the Name Nodes' basic information and responses to any client's file location query requests. This system is design to provide cloud storage service for a large number of users and each user had a separate directory to store their own files and subdirectories. The NLS module only maintains the mapping relationship between the user id and the corresponding Name Node. All the files and subdirectories belong to the same user would be maintaining by the same Name Node.

Each Name Node maintains its successor and predecessor Name Node lists [3]. So a Name Node is aware of its predecessor and successor and can communicate to both sides. It forms a bidirectional ring of Name Node. Periodically, each Name Node is monitor by its successor. So successor finds whether the Name Node is up or down. If it finds it down, then it will notify to all other Name Node that its predecessor is out of service. Predecessor of down node becomes predecessor of Name Node.

5. Expected Result

As per proposed system it provides scalability, improves robustness, makes loosely organized peer-to-peer applications, gives large amount of flexibility using Chord keys, and shows degree of natural load balance using distributed hash function.

6. Conclusion

We propose a fault tolerant, highly available and widely scalable HDFS architecture whose Name Node is distributed and will not suffer HDFS failure in case of a single Name Node failure. We achieve this by utilizing the Chord protocol and integrate it with HDFS Name Node.

7. References

i. Apache™ Hadoop®. Hadoop documentation, http://hadoop.apache.org/, (2014) February 11.
ii. Towards a scalable HDFS architecture, Farag Azzedin IEEE 2013.
iii. Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications.
iv. NCluster: Using Multiple Active Name Nodes to Achieve High Availability for HDFS, Zhanye Wang IEEE 2013.
v. Towards Better Fault Tolerance in HDFS – A Survey Paper, IJIRT 2014.
vi. Hadoop Framework to Provide Fault Tolerance in the Cluster, ASEE 2014 Zone I Conference, April 3-5, 2014, University of Bridgeport, Bridgeport, CT, USA.