ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

IPMPLIMENTING AN INNOVATIVE NAMENODE FAIL SAFE ARCHITECTURE FOR HDFS

Ms. Yojana Nanaware, Dr. Suhas D. Raut
Department of Computer Science & Engineering, N. K. Orchid College of Engineering & Technology, Solapur, Maharashtra, India

DOI: 10.5281/zenodo.159462

ABSTRACT

In data intensive computing and organization the growth of data is phenomenon and unpredictable. To handle it Hadoop is the widely used medium. For online, the client applications of Hadoop require high availability of system. Our project provides a new HDFS architecture present cloud storage scheme utilizes multiple NameNodes, has been proposed to address the issues of availability of NameNode.

KEYWORDS: Hadoop, HDFS.

INTRODUCTION

The growth of internet based applications and web services in last decade have brought change in the mindset of researchers. The traditional techniques are not enough to handle and improve such huge amount of data. There are Information technology solutions provided by organizations having great concern towards the large amount of data their machines are producing [2]. Several system architectures are developed to deal with such large data. But, most of the data growth is in unstructured format. MapReduce is a programming paradigm architecture pioneered by Google and is available in open-source implementation called Apache Hadoop. It is used by organizations like Yahoo, Facebook and other online shopping marts. Data-Intensive Computing Systems have approaches to parallelize the processing of data. The goal to design such platform is to provide high levels of reliability, efficiency, availability and scalability [4]. In our developed architecture of HDFS is exploits above mentioned all features.

EXISTING HDFS ARCHITECTURE

The goal of HDFS is to address the issues of hardware failure, high throughput data access and process large data sets of applications. Hadoop has simple coherency model of write-once-read-many for files and works on idea of moving computation to data. HDFS is portable across heterogeneous hardware and software platforms. The centralized NameNode server stores the namespace of HDFS in memory for high performance. As the storage capacity of a cluster grows, more NameNode server memory is required [5].
HDFS or Hadoop Distributed File System is a block-structured file system. Each file is divided into blocks which are having predetermined size. These blocks are stored across a cluster of one or several machines. Apache Hadoop HDFS Architecture follows a *Master/Slave Architecture*, where a cluster comprises of a single NameNode (Master node) and a number of DataNodes (Slave nodes). HDFS is constructed using Java programming language, due to which HDFS can be deployed on broad spectrum of machines that support Java. Though one can run several DataNodes on a single machine, but in practical world, these DataNodes are spread across various machines [1] [3].

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

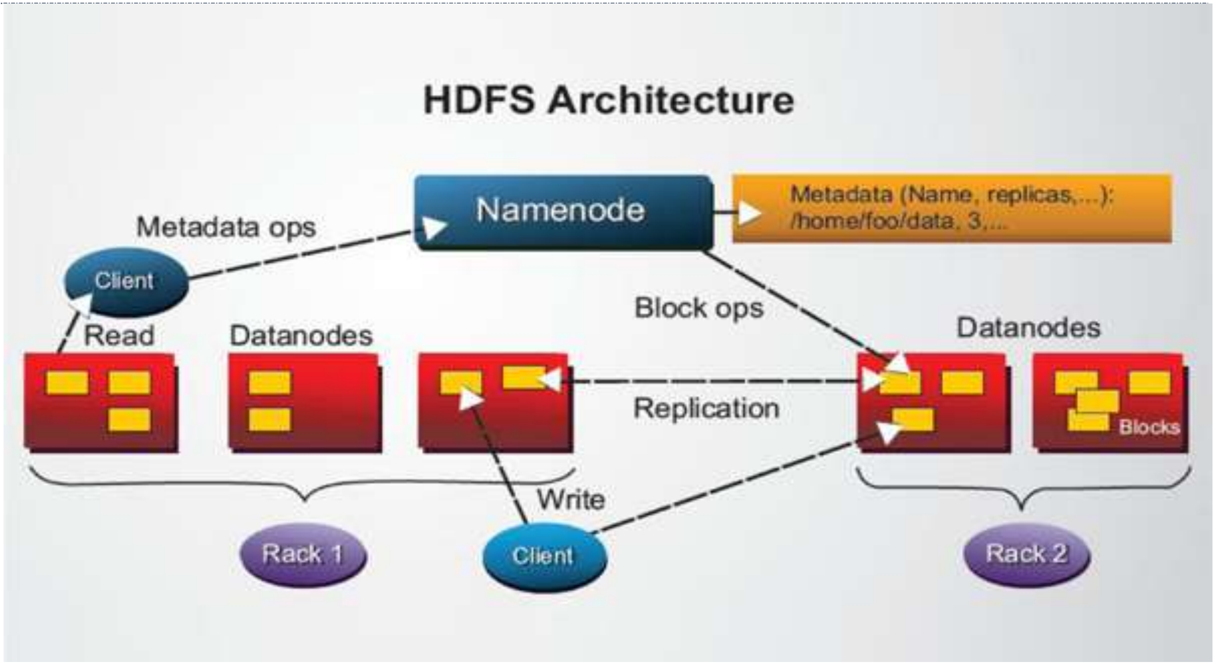

Figure 1: Basic HDFS Architecture

Figure 1, shows the basic HDFS architecture and their modules that help to work architecture efficiently.
NameNode is the master of the HDFS that maintains and manages the blocks present on the DataNodes (slave nodes). NameNode is a very high-availability server that manages the file system namespace and controls access to files by clients. There is just one NameNode in the Hadoop which is the single point of failure in the entire Hadoop HDFS cluster. The HDFS architecture is built in such a way that the user data is never stored in the NameNode.
DataNodes are the slave nodes in HDFS. Unlike NameNode, DataNode is commodity hardware, that is, a non-expensive system which is not of high quality or high-availability. DataNode is a block server that stores the data in the local file.
Secondary NameNode gives an impression that it is a substitute of the NameNode. The Secondary NameNode is one which constantly reads all the file systems and metadata from the RAM of the NameNode and writes in the file system. It is responsible for combining the *editlogs* with *fsimage* from the NameNode. It downloads the EditLogs from the NameNode at regular intervals and applies to fsimage. The new fsimage is copied back to the NameNode, which is used whenever the NameNode is started the next time. However, as the secondary NameNode is unable to process the metadata onto the disk, it is not a substitute to the NameNode. So if the NameNode fails, the entire Hadoop HDFS goes down and you will lose the entire RAM present in the RAM. It just performs regular checkpoints in HDFS. Just a helper, a checkpoint node [1]!.

ISSUES IN EXISTING SYSTEM

As the current HDFS architecture is backward compatible, with simple design, it's require to present the cloud storage scheme [6] to utilize multiple NameNodes provide high availability, and better fault tolerant with reliable HDFS architecture.

SYSTEM ARCHITECTURE

System Architecture helps to the client to perform his I/O operations using different modules very efficiently as shown in Figure 2. Client interacts with NLS module to get the file location and NLS module responses client by using filemap that generate mapping relationship between the NameNode and client. This filemap update as per client request i.e. performs optimized updating.
Each NameNode runs the program that generates configuration file having information about its own system. Also each NameNode consist the information about the active NameNodes & their respected live DataNodes.

All NameNodes in this system are co-ordinate to each other and they are co-exists. When one of the Namenode goes fail, our System Architecture transfers the DataNodes under failure NameNodes to active NameNodes manually.

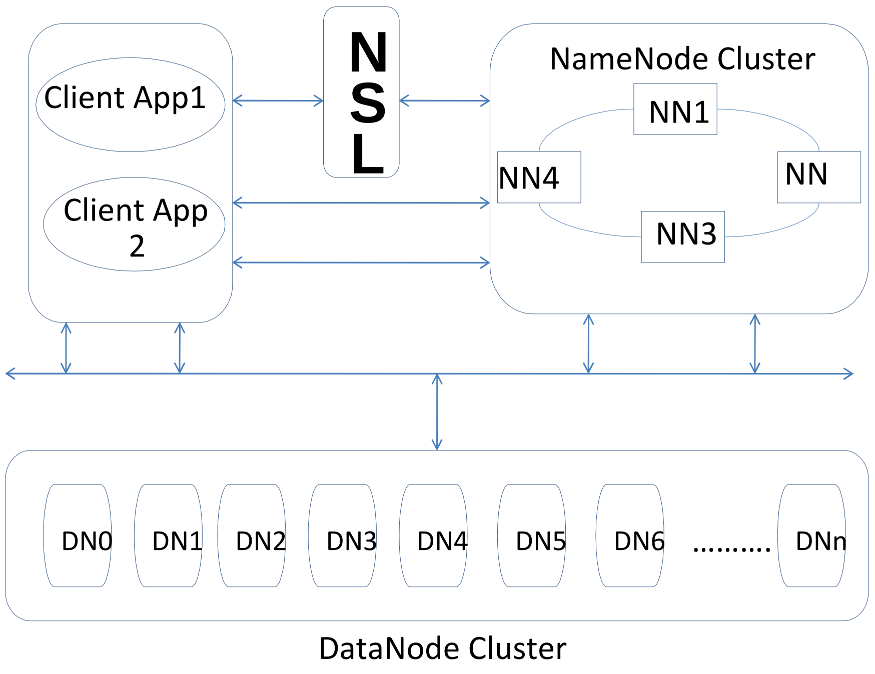

Figure 2: System Architecture

RESULTS

Implemented system architecture provides the degree of natural load balancing, giving the availability of NameNodes and flexibility of naming by using user friendly environment for small and large files which having heterogeneous formats.

CONCLUSION

We developed a better fault tolerant and widely available HDFS architecture who's NameNode is distributed and there Datanodes work under other Namnodes manually when their own NameNode fails.

We are trying to implement this System Architecture, where transforming a DataNode from failure NameNode to active NameNode by automatically instead of manual connection.

REFERENCES

[1] Apache™Hadoop®. Hadoop documentation, http://hadoop.apache.org/, (2014) February 11.
[2] OpenStack, Swift documentation, http://docs.openstack.org/developer/swift/, (2014) June 20.
[3] Wang, F., Qiu, J., Yang, J., Dong, B., Li, X. and Li, Y., November 2009, Hadoop High Availability through Metadata Replication, IBM China Research Laboratory, ACM.
[4] Towards a scalable HDFS architecture, FaragAzzedin IEEE 2013.
[5] NCluster: Using Multiple Active Namenodes to Achieve High Availability for HDFS, 2013 IEEE International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing.
[6] Hadoop Framework to Provide Fault Tolerance in the Cluster, ASEE 2014 Zone I Conference, April 3-5, 2014, University of Bridgeport, Bridgeport, CT, USA.