Research On A Data-Aware RPC Method For Improving Storage Performance Of Big Data System

Mareng,Marko Mawien Mawien

International School, Changsha University of Science & Technology, Changsha city,Hunan province,China

DOI: 10.29322/IJSRP.12.03.2022.p12350
http://dx.doi.org/10.29322/IJSRP.12.03.2022.p12350

Paper Received Date: 1st March 2022
Paper Acceptance Date: 15th March 2022
Paper Publication Date: 20th March 2022

Abstract Hadoop Remote Procedure Call (RPC) is increasingly used in many data centers of known technology companies such as Facebook and Yahoo together with other data middleware centers like MapReduce, HDFS (Hadoop Distributed File System) and Hbase. Due to its simplicity, efficiency, high performance use of RPC systems, it is essential to achieve dense data storage and low latency and high throughput. This paper describes how to improve data storage, throughput, and latency performance for big data. E.Sivaraman Dr.R.Manickachezian(2014) said systems are becoming increasingly data-intensive due to the explosion of data and the need to process it, therefore ,the application performance is vital to storage management in data-intensive computing systems. The author mentioned above highligted that researchers believe that applications sensitive areas are advancing in storing large volume of data. As data volumes continue to grow, the movement of data from memory to storage systems has become a critical performance bottleneck for many data centers.

However, data management on such multi-server storage systems is yet under studies to ascertain on how to utilize various storages for efficient data movement is an important research topic in RPC data perception. In this research, we propose an adoptive Heartbeat system, data-aware distribution that efficiently manages storage systems to coordinate multiple concurrent data-intensive applications. Extensive experiments are conducted and the results show that this method can significantly improve the network throughput performance and data-intensive storage of Hadoop in HDFS clusters. The thesis discusses the background of RPC, huge data application storage systems, related work and technologies, Hadoop installation, adaptive model mechanisms, evaluation and performance analysis of big data architecture and its server interfaces. The reason to focus on storage improvements is because of the great need to store information globally. A large class of communication-intensive distributed applications and software components have been ported to virtual machines, such as high-performance storage systems. This method dynamically adjusts the RPC configuration between the NameNode and the DataNode by sensing the data characters stored in the DataNode. This method can effectively reduce the processing pressure of the NameNode and improve the network throughput generated by the information transmission between the NameNode and the DataNode.

Index Terms- Big Data Ecosystem, Hadoop, HDFS Heartbeat, Throughput, Latency and Adaptive Mathematical Model

I. Introduction

Hadoop's Remote Procedure Call (RPC) is the core communication mechanism in Hadoop. RPC mainly exchanges metadata through all Hadoop components, such as MapReduce, Hadoop Distributed File System (HDFS) and Hadoop MAPREDUCE. Remote Procedure Call (RPC) is a protocol that requests services from remote computer programs through the network without understanding the underlying network technology. RPC assumes that certain protocols such as Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) exist to carry information and data between communication programs. Hadoop RPC is very extensive in the entire Hadoop application, and the communication between Client, DataNode, and NameNode depends on it. To keep data, it has to be processed and integrated into the current business, which uses advancement of technology. Furthermore, the government has to regulate other business sectors or industries to maintain both unstructured and structured data meet government's policy. As the world enjoys the debates on high demands of Big Data being the only tool to provide large storage performances, a breakthrough is reached in joining around the definition of the term "Big Data" is defined as a data collection that has grown so large it cannot be affordably or effectively managed using Google search engines, based on the task. Another trending word is "Big data Analytics "in which highest analytic techniques are designed to work on big data sets. In this regards, big data analytics is all about two main elements include, big data and analytics and how they have teamed up to create one of the most philosophical trends in business intelligence (BI) today. There are a lot methods designed to keep process and analyze large volumes of data in a massively parallel scale and for instance, Hadoop is the best example of parallel processing that store large volume of information. Based on the research conducted by Xu in 2016,the handling big data with Hadoop is facing challenges in addressing reliable storage in a distributed frame work hence storage systems can be scaled-out, but the compute to storage node ratio is still high, rendering the storage subsystem a highly contended component . Therefore, the slower of the system performances in the big data systems results to storage interruption, which reduces the performance target set by other resource managers designed or implemented in a large body of works and if it fails to provide applications with accurate performance then the data is lost completely. The transaction in Big Data system require appropriate timing prediction to complete the bounding period to achieve a meaningful outcomes as it is requires a sensitive duration to execute the job. This section addresses the problems stated above for data-intensive storage systems. According to E.Sivaraman, and R.

Manickachezian paper journal published 2014,the research involves building a virtual system in a virtual machine monitor task to intercept the normal performance in the virtual machine and redirect them to the privileged Virtual Machine(VM). However, adaptive Remote Procedures Call (RPC) traditional system is very slow in performance mostly in Hadoop Distributed File System (HDFS) system. Therefore, there is a need develop another better method to increase the storage capacity and improve network-throughput and low latency. In this thesis, adaptive model will be the better alternative to use in distributed file system for large and data-intensive. Hadoop Distributed File System (HDFS) is a part of Apache Hadoop core project, and its design ideas refer to the Google File System (GFS) system. As an underlying distributed file system framework. The main goal of HDFS RPC Mechanism is to provide extremely low latency and high throughput in VM to configure the heartbeat interval of data stored in the DataNodes, to realize the dynamic adjustment of the node's heartbeat sending mechanism. In this work, the network throughput generated by direct communication between NameNode and DataNode through the heartbeat mechanism can be reduced, and latency time of the entire system can improve the comprehensive performance of the distributed system of the data. Hadoop Distributed File System (HDF) forwards many replicas of data blocks and disseminates them on compute nodes throughout a cluster to enable reliable and fast computations to occur and this will improve RPC system in a data ware for big data.

A. Background and Significance of research

Section 1 presented introduction, theoretical background, purpose, motivation, contributions, methodology and related work. Section 2 introduced the related theories and technologies such concepts of big data ecosystem, Hadoop system HDFS, Map reduce, RPC Mechanism, Rack awareness and guide on how the experimental Hadoop-2.6.5 environment is set up. It is formed as a systematic installation guide including exact terminal commands and configuration parameters in order to set up a multi-node Hadoop cluster. Section 3 explained in details the mathematical models used to simulate the experiments and what methods that are used. Section 4 evaluated the experimental analysis and test results are analyzed and presented inform of graphs. Section 5 concludes the paper and proposes further work that can be done in continuation of this work or further work that explores the limitations set by the scope of this paper.

B. Purpose of the Paper

The main goal of this project is to provide extremely low latency, improve the network throughput and data-intensive storage performances of hadoop hdfs to realize the dynamic adjustment of the node's heartbeat sending mechanism.

C. Motivation

The main of objective Remote Procedures Call (RPC) is to make distributed computing simple and efficient in providing huge storage space and extremely low latency and high throughput in the Virtual Machine to configure the heartbeat interval of data stored in the Hadoop Distributed File System (HDFS) system. The main contribution of this paper is maintaining the local procedure call abstraction for RPC to makes it easier for programmers to use RPC.

D. Contributions

This paper aims to improve the storage performance of Big data systems a using RPC-aware adaptive methods in improving its performance. We develop two systems that used RPC-aware adaptive model to achieve high performance. We make the following contributions in this work:

a. We present a scalable consensus protocol for consistent data replication in We present a scalable consensus protocol for consistent data distribution in form of graphs in data block and Data Rate distribution deployments. This simplifies its design by making assumptions and using information specific to the design of modern datacenters and network infrastructure. Through adaptive model, we demonstrate the effectiveness of RPC aware design approach to improve the performance of large-scale consensus protocols.

b. We observe that adaptive like a Transmission Control Protocol (TCP) protocol are necessary to overcome the network latencies and support higher throughput. To achieve high throughput in distributed deployments, we introduce batching and pipelining while maintaining the safety of the protocol..

c. We evaluate the systems in two deployment models: a local deployment within a datacenter and a distributed cluster across up to several locations.

Our experimental results show it can achieve more than 4 million requests with distributed nodes, which is more than several times higher throughput in the same settings. Design of high performance show that the consensus protocols, and consequently the storage systems using them, can benefit from following the infrastructure-aware design approach.

E. Research Questions

The research questions are set to grasp the Hadoop setup requirement and the performance measurement perspective. How to improve Hadoop RPC in terms of latency and throughput of different data types? How to evaluate the performance of Hadoop RPC on different networks/protocols and the parameter configuration on modern clusters? How to analyze Apache Hadoop in pseudo-distributed installation modes? How to measure the performance in the pseudo-distributed setups? What kind of tools are used to test Hadoop on an ordinary machine? What kinds of hardware and soft wares are to test Hadoop in a normal Virtual box host? What are the configuration parameters for testing a semi distributed Hadoop environment? The installation and configuration chapters as well as the test setup chapter discuss the main research questions. How Does RPC improve system storage performances?

II. Related Theories and Technologies

This section introduces us to different authors who once studied about data storage intensive Remote Procedures Call (RPC) aware and how it solves how to implement local RPC with minimal overhead and system performances. It simulates the native procedure call model and does not require additional messaging, but requires the original procedure call convention. RPC establishes a simple control transfer model by running the client thread to execute the requested service in the server's address space. Distributed systems have many different general physical and logical computer resources. The software system builds a network, computing nodes distributed in various places, and they exchange information through the computer network. This related work will introduce the related work of big data ecosystem, Hadoop Distributed File System(HDFS), Mapreduce, Rack perception and adaptive mechanism model.

In HDFS, there are many data replication tasks designed to improve the performance of the system and are used to store large amounts of data. According to little research work, focus on improving dense storage, network throughput, and data latency in hadoop's HDFS cluster. Jungha Lee, JongBeom Lim, KwangSik Chung and JoonMin Gil proposed to attempt to improve data storage performance in Hadoop clusters. A. Kavitha presented a white paper in 2013 to explain Big Data in simple language. What do you mean by Big Data? What are the traits of big data? It gives brief up of MapReduce model and introductorily oracle database. They mentioned case of Patient health information system on cloud. The real tie application of Big Data can also be in patient health information (PHR). They mentioned three aspects of the system. This application using finger print or iris pattern or face pattern of patient It capable of storing image king of data and able to processing it. Traditional enterprise data includes the entire PHR right from his/her birth with the details of the doctors and prescription and all records. Social data can be used online consultation and medicine purchase. Bhandarkar, M. designed a summary form of Apache Hadoop platform of choice for developing large-scale data-intensive applications. The author developed a Hadoop Tutorial, describe how to design and develop Hadoop applications and higher-level application frameworks to crunch several terabytes of data, using anywhere from four to 4,000 computers. The solutions to most common challenges found in combining Hadoop storage performance are all discussed in this paper. A number of frameworks have been described such as; utilities developed using Hadoop that increase programmer-productivity, and application- performance.

Eltabakhet. (2011) pointed out that Hadoop became an attractive platform for large-scale data analytics. To overcome bottleneck of Hadoop, author introduce Hadoop, a lightweight extension of Hadoop that allows applications to control where data are stored. Instead, applications give hints to CoHadoop that some set of files are related and may be processed jointly CoHadoop then tries to co-locate these files for improved efficiency. Eltabakhet.approach is designed such that the strong fault tolerance properties of Hadoop are retained. Colocation can be used to improve the efficiency of many operations,. He has conducted a detailed study of joins and sessionization in the context of log processing.This data block approach aims to improve data-intensive storage, network throughput, and low latency. This helps the default Hadoop distribution to provide fixed data replication during the data storage phase. It allows the data replication factor to copy the data to the acquired node. In order to increase the storage capacity of the system, data replication must is eliminated. Therefore, there is a limit to providing optimized replication factors for data-intensive storage in the HDFS system. This chapter proposes data distribution blocks and distribution methods to improve system performance and balance data load, consider the processing speed of nodes, and provide initial data placement and adaptive models to improve data storage in the Hadoop cluster environment. However, the performance depends on the application, because it only considers the data model of mathematical analysis. As far as the data distribution block is concerned, it is more important to consider applications that share data across system nodes. The data blocks are distributed and stacked in local racks to collect and analyze the data stored in the blocks. In addition to data replication, some scheduling methods are to improve the data storage in Remote Protocol Call (RPC) to adapt to the adaptive method in the Hadoop cluster environment. It uses waiting time estimation and data transmission time to schedule tasks. It dynamically decides whether to reserve tasks for nodes that store data, or to schedule tasks to requesting nodes by transmitting data to requesting nodes.

2.1 Big Data Ecosystem

Big data ecosystem is the comprehension of massive functional components with various enabling tools capable of computing and storing large amount of data, but it also has the advantages of its systematic platform and potentials of big data analytics(David Loshin,2013).In a research presented by NASA in 1997, Big data had a problem with graphics, which proved a serious challenge in computer. Datasets was taking a large capacity of the main memory, local disk and even remote disk in the system and resolve problem of low storage, Apache Hadoop as an open source software platform for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware was in introduced by Doug Cutting and Mike Cafarella in 2002 and its services were to provide for data storage, data processing, data access, data governance, security and operations. Hadoop performance in terms of storage and analytics is a key issue for companies engaging in data centric business. Apache Hadoop is to scale up from single servers to thousands of machines, each offering local computation and storage.

2.2 Hadoop System

Hadoop consists of a storage part Hadoop Distribution File System (HDFS) and a processing part known as MapReduce. The focus in this thesis is the HDFS part. The main idea of HDFS is that it splits files into blocks that are then distributed across different nodes in a cluster. That is called distributed storage and all data blocks of a file have the same size except the last block which can be smaller. The number of blocks of a file is the exact multiple of the configured block size, plus an additional final block with the number of bytes that is left. Hadoop Distributed File System (HDFS) is

used to store big data sets and the default block size is 128MB and if the block size is small like for instance in the Linux file system , Hadoop Distributed File System(HDFS) will generate huge amounts of overhead and traffic in order to manage the large number of blocks and metadata files that is not beneficial, so the block size is therefore set to 128 MB,

2.3 Hadoop Heartbeat

In distributed processing systems, a heartbeat is basically intended to periodically check the node's liveness. Beyond checking liveness of daemons, controlling tasks, and other information, tasks are incorporated inside heartbeats to reduce the number of messages between nodes. In particular, the NameNode and JobTracker are central entities that manage the other nodes in the cluster. Thus, Hadoop restricts the number of messages sent to these nodes in order to reduce their loads. This paper focuses on Hadoop 2.6.5 architecture. In Hadoop 2.6.5, there are two kinds of heartbeats: the Hadoop Distributed File System (HDFS)heartbeat and the Yarn heartbeat. In recent big data services, Hadoop Distributed File System (HDFS) can help to improve the quality of service, the system performance and reliability of data intensive applications. It has received more and more attention from academia and industry. The figure 2 below shows Big Data Ecosystem Architecture

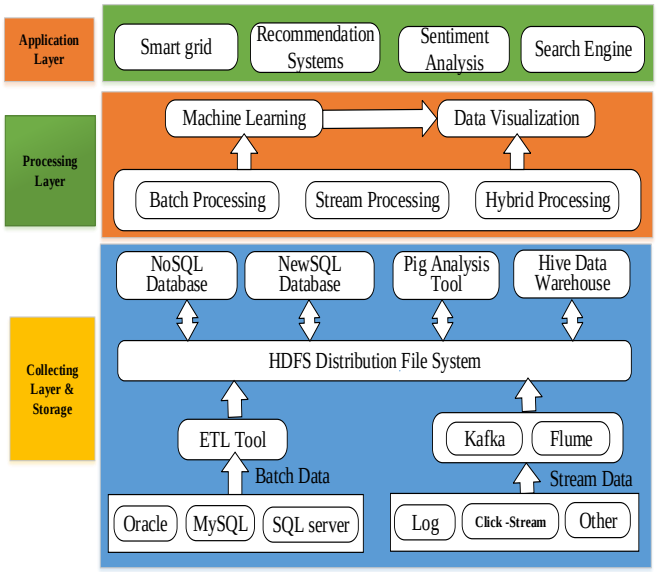


Figure 2: HDFS Architecture Data Flow

2.4 Hadoop **Distributed File System (HDFS) Heartbeat**

The Hadoop Distributed File System(HDFS) is a file system that supports a distributed environment. It is composed of a NameNode and DataNodes. The NameNode is a server that manages the metadata of the file system. DataNodes save the actual data blocks and deliver those blocks to the read and write data when clients need that data. The HDFS heartbeat is the heartbeat between the NameNode and DataNodes. A DataNode sends information about its liveness as well as its status change. When a client reads a certain file from the HDFS or writes to the HDFS, the client accesses the NameNode and receives the position of the block (e.g. the node's IP address that has the block.). Then the client directly accesses the DataNode and requests block read and write operations. Because the NameNode has scalability issues, it does not send a message to a DataNode proactively. Instead, the NameNode communicates with a

DataNode in response to a message from that DataNode when needed. A DataNode sends periodic heartbeat signals to the NameNode. Thus, if there is a change in a DataNode's status caused by the completion of a read or write operation, a time delay might exist for the status update to the NameNode. The maximum delay is the heartbeat cycle. For this reason, the heartbeat cycle can influence the execution times of the read and write operations in the HDFS. The described process in which clients write blocks in the HDFS is shown in figure 1 below

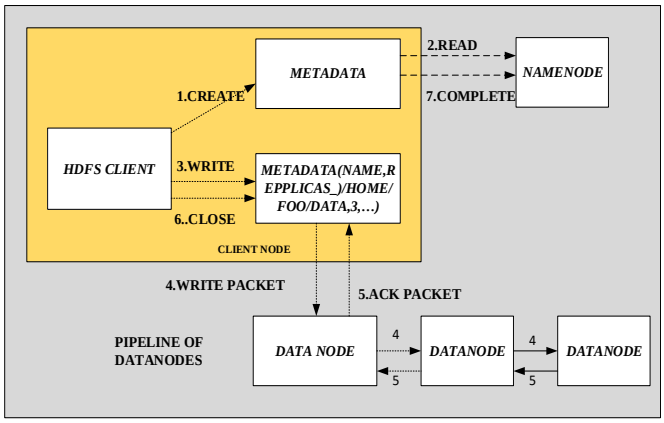


Figure 1 HDFS Heartbeat

In HDFS, the heartbeat period for DataNode>NameNode is set to /etc/hadoop/hdfs-site.xml in the Hadoop folder. The default value is set to three seconds. In a large system with many DataNodes, a short heartbeat period may put excessive load on the NameNode. However, for Hadoop clusters with less than a hundred DataNodes, DataNode updates with short heartbeat cycles will not cause excessive load on the NameNode. Therefore, by reducing the heartbeat period below the default value, you can expect an improvement in HDFS performance.

2.5 Mapreduce

MapReduce can be referred as processing engine and as a unit of works that executes across multiple computing nodes which completes in multiple phases i.e. map phase and reduce phase. In the map phase, each map task processes a block of input dataset that is generally stored in a distributed file system. The input dataset is split into blocks of pre-defined size(128MB), and distributed over cluster nodes. The map tasks read the data blocks and applies the user-defined map function. The map output the map function to collect it into physical memory. It is passed from physical memory to the local disk of the processing node performing the mapping task. The location of the map output is passed to the master node responsible for forwarding to the node executing the reduce task. In the reduce phase, the reduce task reads the map output from the remote location, and sorts it by the key of the map output, so that all values of the same key are grouped together. After that, the reduce task iterates over the sorted map output file and passes the unique intermediate key and associated value to the reduce function. The output of the reduce function is written to the distributed file system as the

final output. The number of final output files depends on the number of reduce tasks initiated

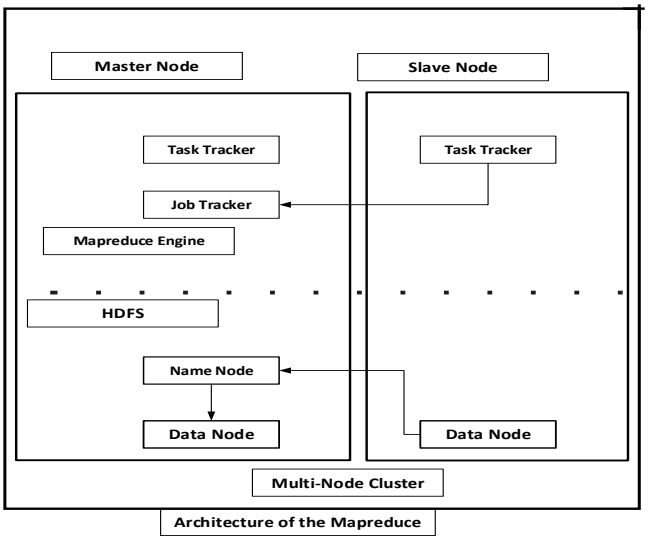


Figure 4: Mapreduce Architecture

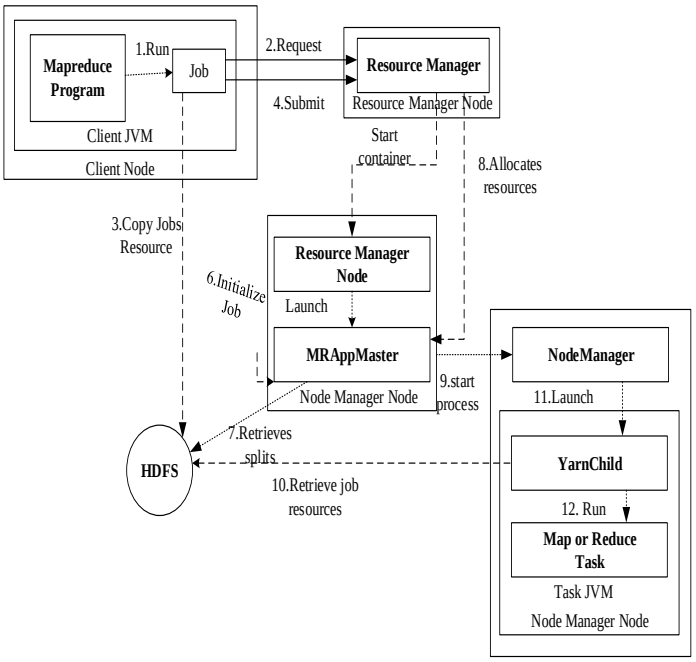


Figure 5: YARN Architecture

2.6 YARN Heartbeat

In Hadoop 2.6.5, Yarn is supported, as shown in Fig. 2. Yarn is a new resource management and scheduling method that separates the job scheduling and monitoring functions that were managed by JobTracker in Hadoop 2.0. Yarn's Resource Manager (RM) and Application Master (AM) manage the necessary functions. In particular, RM manages all of the worker nodes in clusters and receives reports on the status of each worker node. When jobs require resources, RM assigns the necessary resources to meet the specific requirements of each job. AM manages each job and is located in each server. It manages worker nodes, determining whether they satisfactorily execute tasks. If the tasks are completed, Application Master (AM) disappears after collecting and sending the results to RM and the client. Each worker node is managed by Node Manager (NM). In Yarn, each state is reported via heartbeat to reduce the load on the central server in the same manner as the HDFS. Heartbeats can be divided into two types:

I. Node Manager >Resource Manager heartbeat
II. Application Master >Resource Manager heartbeat

In Type I, NM sends a heartbeat to RM for each cycle, and this heartbeat contains the status information of the current node. In Type II, AM sends a heartbeat to RM for each cycle, and this heartbeat contains the status information for the current job being performed, as well as the job results. The default value of each heartbeat is one second and it can be configurable if needed. For Type I NM >RM heartbeat can be set in Hadoop the /etc/hadoop/yarn-site.xml file. For Type II AM >RM heartbeat can be set in the /etc/hadoop/mapred-site.xml file.

2.7 Hadoop Installation and Configuration

This section, the mean goal is to set up an experimental Hadoop environment and that will form a guide on how to fully set up multi node Hadoop clusters. The setup begins with a single node hadoop set up and then multi node setup comes next. The Hadoop is installed on Ubuntu 16.04LTS mounted on virtual Box machines. In this setup, many important commands and configuration are presented by Apache Software Foundation's official website [Foua] and [Hadoop Tutorials 2017] in order to ensure all files are correctly stored within HDFS. The contents of this setup was done with help of the. In addition, the environmental variables configuration files need to be configured so that the Hadoop single node setup works without any problem. These files are HDFS-site.xml, yarn-site.xml, mapred-site.xml, Core-site.xml and the Hadoop-env.sh

2.7.1 Local Standalone mode

As it was presented by Turkington in 2013 that mode is Hadoop's default mode and so its components like Namenode,Datanode,JobTracker and TaskTracker don't require standalone mode configuration.

2.7.2 Pseudo-Distbuted Mode

According to Turkington ,2013 installation guide, Pseudo-Distributed mode is regarded a mini cluster setup on a single host and a seperate JVM is spawned for every Hadoop components as they could communicate across network sockets, effectively producing a fully functioning and optimized mini-cluster on a single host. So, in case of this mode, changes in configuration files will be required for all the three files- mapred-site.xml, core-site.xml, hdfs-site.xml(Fitsum Belay,2017).

2.7.3 Fully Distributed Mode

In this mode, spreads Hadoop across several machines in the cluster and the fully distributed mode is the one that can scale Hadoop across a cluster of machines. The upcoming multi-node cluster setup helps in the successful setup and configuration of Hadoop cluster. (Turkington ,2013).

2.7.4 Java installation

Hadoop needs Java regardless of which type of Hadoop installation is used. Hadoop requires Java version 8 or above. The screenshot from the node assures whether Java is successfully installed. If the above command generates no output, a full Java install is required. After Java is properly installed, the Java environmental variable is set in the bashrc file.

2.7.5 Hadoop Configuration Files

In order to ensure Hadoop setup works correctly, the environmental need proper configuration to avoid system failure. The files which need to be properly configured are HDFS site.xml,Mapred-site.xml,core-site.xml,yarn-site.xml and Hadoop-env-site.sh [Apache Software Foundation's official website ,Foua]

III. Adaptive Heartbeat Mathematical Model

In this section, nodes perform the fixed number of tasks in accordance with task slots configured in Hadoop. When a task tracker notices that there are one or more empty task slots, it sends request messages for tasks to job trackers. A job tracker preferentially assigns a task to the node that holds the data block. If there is no such node with node storage, it assigns the task to a node with rack or rack-off storage. When the data problem arises, for such a reason, some data blocks transfer to nodes with overhead. This problem can be alleviated by using a adaptive heartbeat model. The proposed method adaptively determines whether it replicates a data file or not, according to the current replication factor value and the predicted access count for the data file. To this end, it is necessary to maintain and analyze metadata for data blocks.

3.1 NameNodes and DataNodes Interaction in Adaptive Model

In this part, steps by step data node and namenode interaction will be discussed in details. The adaptive mechanism model of cluster in HDFS consists of a single NameNode (Master node) and all the other nodes are DataNodes (Slave nodes),.If all checks pass the NameNode create file and return success to the client see event ClientCreateFile,NameNode can easily assign DataNode to store that block and send back the DataNode name to Namenode and this will start send block to DataNode,if the file is larger than one block will again connect to NameNode to get location of new block, see the adaptive mechanism in Figure 7

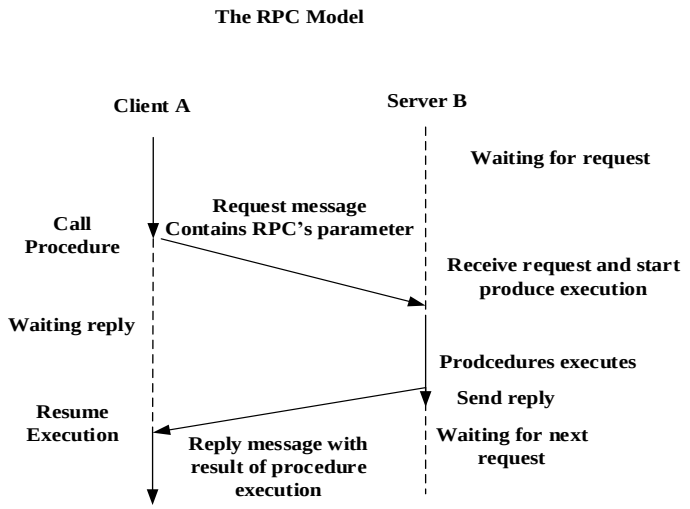


Figure: 6 RPC MODEL

It is similar to a commonly used procedure call model .It works in the following manner:

1. For making a procedure call, the caller places arguments to the procedure in some well-specified location.
2. Control is then transferred to the sequence of instructions that constitutes the body of the procedure.
3. The procedure body is executed in a newly created execution environment that includes copies of the arguments given in the calling instruction.
4. After the procedure execution is over, control returns to the calling point , returning a result .

3.1 HDFS Heartbeat Model

In this paper, a running cycle means a specific and fixed running time duration of the HDFS system. Node i counts the number of data blocks in a jth running cycle,$Y_{ij}(i1,2,\dots N, j = 1,2,\dots k)$ we assume that $\mathbf{y}_{ij}$ will affect the heartbeat sending time interval tij of node i in the current cycle. The greater the value of , the shorter the heartbeat sending interval t_{ij} in jth cycle, and the faster the heartbeat sending frequency. Conversely, the smaller the Y_{ij} , the longer the heartbeat sending interval t_{ij} in jth cycle, and the slower the heartbeat sending frequency. In our model, N refers to the expected mean number of data blocks stored by all nodes in the cluster derived from the historical records, and it has the following form:

$$\boldsymbol{N = F(y_{ij}) \int_{i=1}^{n} (y_{ij})} \qquad (1)$$

DataNode uses $\mathbf{Y_{ij}}$ in jth cycle to calculate the heartbeat sending interval $\mathbf{t_{ij}}$ of the node . Using formula (1), the calculation method of the heartbeat interval for node i is:

$$\boldsymbol{t = around\left\{t. = \frac{w.N}{y_{ij}}\right\}} \qquad (2)$$

Table 1 Definition of Related Symbols

Symbol	Description
N	The expected number of data blocks store in all the nodes in HDFS cluster
t`	The default fixed heartbeat interval
M	The mean average value of distribution rate
Y_{ij}	The number of distribution rate per a block size
M	The number of nodes in the system
t_{ij}	The distribution transmission between nodes
T=Round{}	means rounding function of the distribution rate used in system RPC configuration
P_{ij}	This is the length between nodes load internally
P	The constant network loads between the nodes
T ART	It represents latency
T'_{ART}	The constant value of latency
w	It is a system evaluation function
$\int_{n=1}^{n} \frac{T}{t_{ij}}$	The integral total number of transmission rate forwarded
T	The distribution rate of data load

In the model, t refers to the initial system RPC heartbeat configuration. The heartbeat factor ω is used to control the influence of the mean value M on the heartbeat interval to avoid excessive changes in the heartbeat interval and increase the burden on the system. Round is a function for rounding, which can be used to

simplify system RPC configuration. The mathematical model dynamically adjusts the heartbeat-sending interval of each DataNode by sensing the stored data. Assume that the length of the heartbeat information sent between nodes is the same without considering other influencing factors in the interior HDFS network, in jth cycle, the system network load P_{ij} generated by all the nodes is:

$$\boldsymbol{p_{ij}} = \int_{i=1}^{n} \frac{\boldsymbol{T}}{\boldsymbol{t_{ij}}} . \boldsymbol{L} \int_{n=1}^{n} \frac{\boldsymbol{T}}{\boldsymbol{round}\left\{t' . \frac{\boldsymbol{w.N}}{\boldsymbol{y_{ij}}}\right\}} . \boldsymbol{L} \qquad (3)$$

Where T is the interval time of one running cycle, and $\int_{n=1}^{n} \frac{T}{t_{ij}}$ is the total number of heartbeat packets sent by the DataNodes in the system. For distributed systems, the fault tolerance of the system is also crucial. The time interval between two continuous heartbeats determines the system response time for node fault. If the NameNode receives no heartbeats from one DataNode within a heartbeat interval, the DataNode is considered to be in an abnormal state.

Normally, the abnormal response time of a distributed system equals the heartbeat interval. Assuming that the number of data blocks in the node Ni is large, it has a high probability that it will affect the completion of the task in the node when an abnormal state occurs. Conversely, if the number of data blocks is small, the task affects probability when an abnormal state occurs will also decrease. For this reason, in $\mathbf{j_{th}}$ cycle, the total abnormal response time of the system is:

$$\mathbf{T'_{ART}} = \int_{i=1}^{n} \mathbf{P_{ij}} . \mathbf{t_{ij}} = \int_{k=1}^{n} \frac{\mathbf{y_{ij}}}{\int_{k=1}^{n} \mathbf{y_{ij}}} . \mathbf{round}\left\{\mathbf{t'} \frac{\mathbf{w.N}}{\mathbf{y_{ij}}}\right\} \quad (4)$$

$$(\mathrm{i} = 1,2 \ldots, \mathrm{N}; \mathrm{j} = 1,2 \ldots, \mathrm{K})$$

In order to obtain better overall system performance, the network load M of the system and the abnormal response time T_{ART} are needed to be clustered. To sum up, we define the system optimization function with the adaptive heartbeat interval as:

$$\mathbf{P} = \frac{\boldsymbol{Q}}{\boldsymbol{Q'}} + \frac{\boldsymbol{TART}}{\boldsymbol{T'ART}} =; \int_{i=1}^{n} \mathbf{P_{ij}} . \mathbf{t_{ij}} = \int_{k=1}^{n} \frac{\mathbf{y_{ij}}}{\int_{k=1}^{n} \mathbf{y_{ij}}} . \mathbf{round}\left\{\mathbf{t'} \frac{\mathbf{w.N}}{\mathbf{y_{ij}}}\right\} (5)$$

$(i = 1,2 \ldots, N; j = 1,2 \ldots, K)$

In order to obtain better overall system performance, it is necessary to cluster the system's network load Q and abnormal response time TART. In summary, we define the system clustering function of adaptive heartbeat interval as:

$$\mathbf{P} = \frac{\boldsymbol{M}}{\boldsymbol{m'}} + \frac{\boldsymbol{TART}}{\boldsymbol{T'ART}} =; \frac{\int_{i=1}^{n} \frac{\boldsymbol{p_{ij}}}{\boldsymbol{p_{ij}}} . \boldsymbol{t_{ij}}}{\int_{n=1}^{n} \frac{\boldsymbol{P_{ij}}}{\boldsymbol{p_{ij}}} . \boldsymbol{t_{ij}}} + \frac{\int_{k=1}^{n} \frac{\boldsymbol{Y_{ij}}}{\boldsymbol{y_{ij}}} . \boldsymbol{t_{ij}}}{\int_{k=1}^{n} \frac{\boldsymbol{Y_{ij}}}{\boldsymbol{y_{ij}}} \boldsymbol{t'}} \quad (6)$$

$(i = 1,2 \ldots, N; j = 1,2 \ldots, K)$

In above mathematical model,$\boldsymbol{M}$ and **T ART** respectively refer to the network load and total system abnormal response time in a typical HDFS system using the default fixed heartbeat interval t in each running cycle. Our goal of this article is to get a smaller$\boldsymbol{w}$. When **M** $\boldsymbol{M'}$ and $\boldsymbol{TART}$ $\boldsymbol{T'ART}$ are both less than 1, it is considered that our proposed method outperform the compared system method with the default settings.

Conclusion RPC provides programmers with a familiar mechanism for building distributed systems. RPC facility is not an universal panacea for all types of distributed applications, it does provide a valuable communication mechanism that is suitable for building a fairly large number of distributed applications

IV. EVALUATION

In this section, the methods for clustering of remote procedures call in data-aware Hadoop intensive storage to verify systems experiment. This is to analyze and improve the network throughput and low latency as well as increasing the performance of Hadoop intensive storage for RPC aware in the system servers.. In the experiments, the distribution rate and heartbeat interval between Namenode and dataNode in the HDFS system will improve. The adaptive model is to communicate data blocks and distribution rate between the NameNode and DataNodes in a HDFS system .In this adaptive model, it will increase the network throughput, latency time, system storage performances in the virtual servers.

4.1 Experimental setup

We set up Hadoop cluster using an Intel Xeon server machine. The system performance and the storage of HDFS distributions block and Distribution will evaluate and analyze in graphs to demonstrate the performance improvements of RPC aware intensive storage. In this chapter, a virtual Box was first installed then a Ubuntu 16.04LTS mounted on it and hadoop-2,6.5 and java 8 were installed on Ubuntu Linux. The 2 Virtual Machines (VMs) on the server were configured for the Hadoop cluster to run smoothly.. In this paper, we show how changes in the heartbeat period can influence system performance. The server hardware consists of an Intel Xeon E3-1240V3 Quad-core processor, 4GB DDR3 RAM, a 1TB HDD (data storage), a 256GB SSD (data storage), a 120GB SSD (operating system), and SWAP/System space. The software environment includes the operating system (Ubuntu Desktop 16.04 LTS), a Java virtual machine (OpenJDK 1.8.0), and Hadoop (Apache Hadoop 2.6.5).

V. Experimental results and Performance Analysis

In this section, the methods for clustering of remote procedures call in data-aware Hadoop intensive storage to verify systems experiment. This is to analyze and improve the network throughput and low latency as well as increasing the performance of Hadoop intensive storage for RPC aware in the system servers.. In the experiments, the distribution rate and heartbeat interval between Namenode and dataNode in the HDFS system will improve. The adaptive model is to communicate data blocks and distribution rate between the NameNode and DataNodes in a HDFS system .In this adaptive model, it will increase the network throughput, latency time, system storage performances in the virtual servers.

5.1 Data Block Distribution

In this data block, the node storage situation of the clusters in each running load, different nodes are proposed to adjust heartbeat intervals dynamically. According to the experiment, the average number block M blocks in cluster is 1000.In this design, the maximum heartbeat interval is 10s, this is done to avoid too much long heartbeat intervals in the adaptive mechanism. This mechanism fully utilizes the performance of the node, which reduces the load on the server to a certain extent and improves the efficiency of the entire transmission process. In addition, in order to study the influence of block number stored by nodes in clusters with different heartbeat sending intervals. We also simulated three different clusters in three cases. The experimental configuration is the same as Cases, 6000 blocks are uploaded to the different clusters in the three cases. There are a total of 10 slave nodes in a distributed cluster, and the experiment will randomly distribute 1000 data blocks and store them in 10 slave nodes in the cluster.The historical number records calculate the mean M of each cluster, and the heart factor was 0.5. The designed experiments are used to observe the effect of node storage in different clusters on the heartbeat interval transmission. In order to study the different allocation possibilities of data blocks on the cluster, three clusters with different allocation situations are simulated: To update the data block and distribution rate, a **10** experimental sets is set up to conduct how the system is performing in every running node within the system.

Case **A** containing data blocks ranging from **500=<1600 blocks**, Case **B** with data block from **1700=>3700** and

Case **C** with data **2700>=4700** respectively. The above data blocks will be distributed and stored in **10** nodes cluster in form of graphs. In the graphical presentation shown below, one can see the number of nodes as high as **600** nodes uploaded into data blocks.

The reason of distributing nodes into the data blocks is to ensure that system storage is large enough to store a good number of nodes in HDFS system.

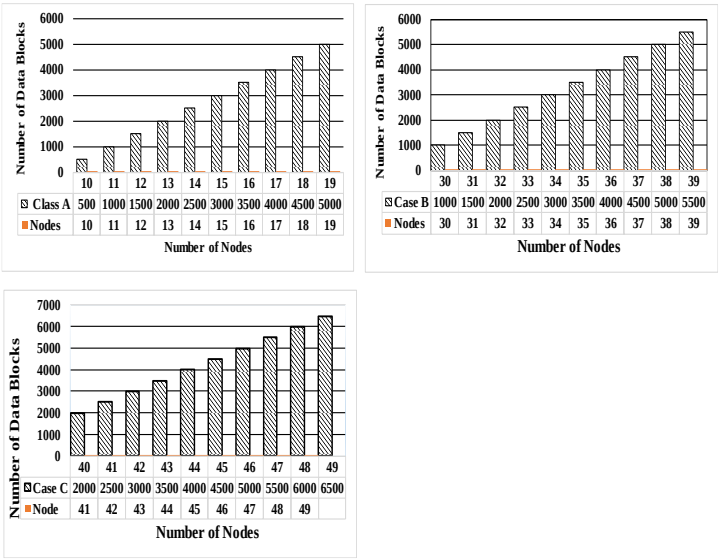


Figure: 8 Graphs showing Nodes distribution and Execution

5.3 Data Block Heartbeat Distribution

This experiment will randomly distribute 1600, 2700 and 4700 data blocks and store them in 10s in the cluster. In this study, the replication heartbeat factor is 3 and The detailed heartbeat period is configured as follows. For the experiment, we reduce the HDFS heartbeat from 3 seconds to 1 second, and reduce the other two heartbeats from 1 second to 0.5 seconds. These experiments are to observe the effect of node storage in different clusters on the heartbeat interval transmission. In order to clearly know the allocation possibilities of data blocks on the cluster, here is the distribution rates of three clusters with different allocation simulated in percentages and blocks:

For case A we distribute 10% to 20% of 1600 data blocks uploaded to cluster which seen belong in graph A.

For case B we distribute 30%~40% of the 2700 data blocks uploaded to the nodes in the cluster shown in graph B.

For case C we distribute 50% to 70% of the 3700 data blocks uploaded to the nodes in the cluster shown in **graph C.** In this process, the distribution rate of case A will be 600-1600 which less than or equal to 600, the generated network load is greater than or equal to 1600 and continues to increase and in Case B, the distribution rate of Case B is 1700-2700 which probably greater than about 1700, the generated network load is less than or equal to 3700 and the same process is repeated in case C.

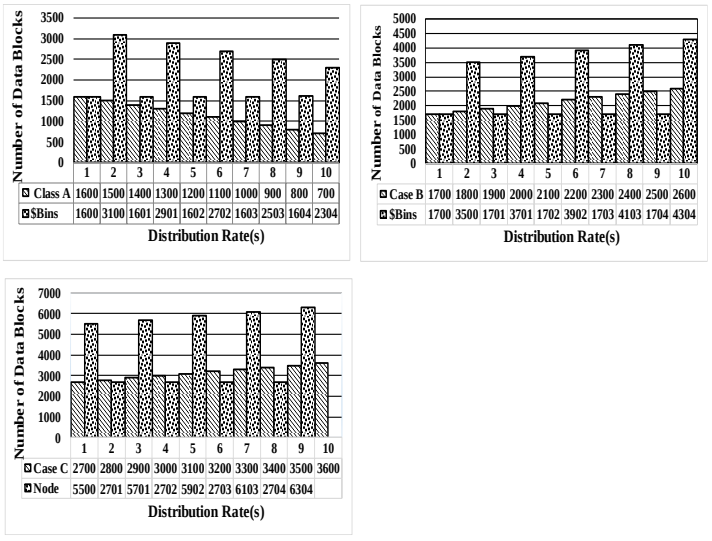


Figure: 8 Data Block Distribution Rate

5.4 Network Throughput And Low Latency

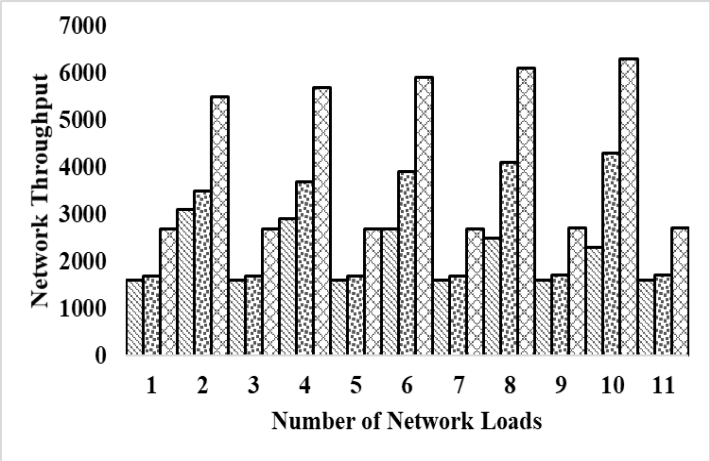
This graph figure 10 below clearly explain the network throughput represented in form of data blocks against times. These graphs are clustered from Case A, Case B and Case C.

In order to improve the network throughputs and latency time generated by longer amount of time, the total system time range runs from 0.5 to 7.5 based on the experimental data of 10 load ranges from 1000 to 4700 blocks respectively. Compared to the experimental results for Case A, the results for Case B show a performance increase of 20%. Compared to Case B, Case C show a performance increase of 40%. With file write, DataNodes must

update the status to the NameNode. Therefore, reducing the HDFS heartbeat periods increases the total processing time. When the Yarn heartbeats (AM>RM heartbeats and NM>RM heartbeats) are shortened to the same value, the overall performance improves. In particular, compared to Case A, the Case B results show a performance increase of 30%. Compared to Case B, the Case C results show a performance increase of 20%. Compared to Case B, Case C show a performance increase of 40%. With file write, DataNodes must update the status to the NameNode. Therefore, reducing the HDFS heartbeat periods increases the total processing time. When the Yarn heartbeats (AM>RM heartbeats

cycles. The network load ranges from 1000 to 4700 blocks respectively. Compared to the experimental results for Case A, the results for Case B show a performance increase of 20%. Compared to Case B, Case C show a performance increase of 40%. With file write, DataNodes must update the status to the NameNode. Therefore, reducing the HDFS heartbeat periods increases the total processing time. When the Yarn heartbeats

(AM>RM heartbeats and NM>RM heartbeats) are shortened to the same value, the overall performance improves. In particular, compared to Case A, the Case B results show a performance increase of 30%. Compared to Case B, the Case C results show a performance increase of 20%. Compared to Case B, Case C show a performance increase of 40%. With file write, DataNodes must update the status to the NameNode. Therefore, reducing the HDFS heartbeat periods increases the total processing time. When the Yarn heartbeats (AM>RM heartbeats and NM>RM heartbeats) are shortened to the same value, the overall performance improves. In particular, compared to Case A, the Case B results show a performance increase of 30%. Compared to Case B, the Case C results show a performance increase of 20%. This is because the task scheduling latency is reduced. there are noticeable increases in performance. However, when the NM>RM heartbeat is shortened (compare Case A, B to C), only a slight improvement is observed, and no significant differences exist. This indicates that the process time between the RM and AM is a performance bottleneck in which the RM allocates resources to the AM, and the AM reports the results back to the RM. Therefore, we can achieve the most performance gains by reducing the AM>RM heartbeat. Finally, if we shorten all three heartbeats (Case C), we obtain a performance improvement of 70% over the default case.



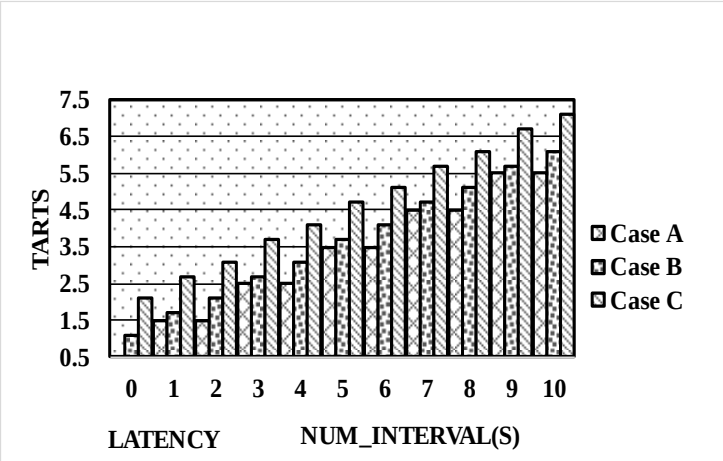


Figure 9: Shows Data Block Distribution Rate

5.5 Section Summary

Data in the era of big data has become an important foundation for the development of production materials and technology in this era. As the data processing, requirements for massive data sets of different data types are getting higher and higher, a simple data processing framework can no longer meet the data needs of the current big data environment. Issues such as scale, reliability, fault tolerance and high speed. The distributed architecture is scalable and can provide a good solution for the storage and processing of massive amounts of data. Therefore, the research of distributed systems under the framework of big data has attracted more and more attention. In a distributed architecture, because the geographic locations of the nodes in the system may be far apart, the normal operation of the system must know the status information of each node at different times, which depends on the information interaction between the nodes in the system.

The interaction between nodes in the current distributed system mainly relies on the RPC-based heartbeat sending mechanism. There are usually two driving methods to stimulate the operation of the heartbeat mechanism: one is to send heartbeat information periodically, and the other is event-driven. Send heartbeat information type. However, these two mechanisms have their own problems. When there are too many nodes in the system, sending heartbeats periodically may burden the nodes. For example, the master node must receive and process massive heartbeats in a short period of time. Information, this may cause the master node to crash. Although the event-driven heartbeat mechanism can reduce the sending of unnecessary heartbeat information between nodes, when a node fails, the information interaction between the system nodes using this heartbeat mechanism needs to trigger the corresponding event, which causes the node to fail to operate normally. Work. Timely failures Report the node status to the system administrator, reducing system performance. As the number of nodes in a distributed system continues to increase, the traditional RPC heartbeat sending mechanism is too simple. This article proposes an adaptive mechanism method to improve the performance of HDFS from the perspective of information exchange between HDFS system nodes:

RPC heartbeat realizes the dynamic adjustment of the heartbeat sending interval of the slave node in different cycles by sensing the number of data blocks stored by the slave node in the current cycle. When the slave node stores fewer data blocks in a certain period, the information update on the slave node may also be less. At this time, the heartbeat sending interval can be increased to reduce the consumption of heartbeat transmission; on the contrary, when the node has many internal data blocks in a certain period, the possibility of receiving update tasks on the master node will also increase. At this time, the heartbeat sending interval can be reduced, the data on the node can be updated in time, and the real-time performance of the cluster data update can be further enhanced.

The experimental results prove that the above method dynamically adjusts the heartbeat sending interval of the nodes, and the network load generated by each cycle of the cluster under different experimental conditions is less than the network load generated by the HDFS system using the default fixed heartbeat mechanism. At the same time, the cluster system using the adaptive heartbeat mechanism is abnormal The response time is less than the default system abnormal response time of 3 seconds, and the system tuning function value derived from the experimental data is less than 2. Therefore, considering that the proposed method is compared with the default heartbeat setting of the HDFS system, tuning is performed.

This chapter mainly verifies the big data system with adaptive RPC heartbeat through experimental simulation. By adjusting the heartbeat sending interval between the NameNode and DataNode in the HDFS

system, the network load generated by the system during the information exchange process and the abnormal response time of the system are reduced. The experimental results prove that by dynamically adjusting the heartbeat sending interval of the nodes, the network load generated by the cluster in each cycle under different experimental conditions is less than the network load generated by the HDFS system using the default fixed heartbeat mechanism. At the same time, the abnormal reaction time of the cluster system using the adaptive heartbeat mechanism is less than 3 seconds of the default system abnormal reaction time, and the system tuning function value obtained from the experimental data is less than 2, which proves that the proposed methods are similar. Compared with the default heartbeat setting of the system, it has been tuned.

5. Conclusion and Future Work

In this paper, I have presented the design, implementation and performance evaluation of Virtual Machine Remote Procedures call (VMRPC). VMRPC trades transparency for efficient communications, and provides fast responsiveness and high throughput when deployed for communicating components in virtualized environments. It is specifically designed for the applications that require high-volume data transfer between VMs. My evaluation shows that the performance of VM- RPC is an order of magnitude better than the traditional RPC systems and the existing inter-domain communication mechanisms in processing data-intensive applications, and VMRPC also improves the performance of a networked file system. With the availability of time and resources, further this system can be extended by making use of large number of machines for creating cluster and storing the data on machines having upgraded hardware in order to increase the capacity of the system. it can be applicable to corporate organizations as well as educational and social institutes for secure data sharing and storing. As our future work, we plan to develop new features such as dynamic heap management, non-blocking RPC, and add support for communication between the domain servers.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] http://hadoopblog.blogspot.com/2010/02/hadoop-namenode-highavailability.htm, Feburary 6, 2010.

[2] Manghui Tu, Peng Li, I-Ling Yen, BhavaniThuraisingham, LatifurKhan.Secure Data Objects Replication in Data Grid., IEEE Transactions on Dependable and Secure computing, Vol. 7, No. 1,2010

[3] Simone Leo, Gianluigi Zanetti Pydoop:a Python MapReduce and HDFS API for Hadoop. HPDC 2010: 819-825

[4] Susmit Bagchi:On reliable distributed IPC/RPC design for interactive mobile applications. MEDES 2010: 33-38October 2010 302 pages ISBN:9781450300476 DOI:10.1145/1936254

[5] K onstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler Yahoo!, "The HadoopDistr ibuted File System," IEEE NASA storage conference,http:// storageconference.org/2010/Papers/MSST/Shvachko.pdf.

[6] Konstantin V. Shvachko:HDFS Scalability: The Limits to Growth. login Usenix Mag. 35(2) (2010)

[7] Hao Chen, Lin Shi, Jianhua Sun:VMRPC: A high efficiency and light weight RPC system for virtual machines. IWQoS 2010: 1-9

[8] H. Zhu and H. Chen, "Adaptive failure detection via heartbeat under hadoop," in Pro. APSCC, Jeju, Korea (South), pp. 231- 238, 2011:

[9] Feng Wang, Jie Qiu, Jie Yang, Bo Dong, Xinhui Li and Ying Li,"Hadoop High Availability through Metadata Replication" CloudDB'09 Proceedings of the first international workshop on Cloud datamanagement

[10] Ekpe Okorafor1 and Mensah Kwabena Patrick,"Availability ofJobtracker machine in hadoop/mapreduce zookeeper coordinatedclusters", Advanced Computing: An International Journal (ACIJ),Vol.3, No.3, May 2012.

[11] Jiong Xie et al., "Improving MapReduce performance through data placement in heterogeneous Hadoop clusters," 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010, pp. 1-9, doi: 10.1109/IPDPSW.2010.5470880.

[12] Preethika Kasu, Prince Hamandawana, Tae-Sun Chung:DLFT: Data and Layout Aware Fault Tolerance Framework for Big Data Transfer Systems. IEEE Access 9: 22939-22954 (2021)

[13] Ekpe Okorafor1 and Mensah Kwabena Patrick,"Availability ofJobtracker machine in hadoop/mapreduce zookeeper coordinatedclusters", Advanced Computing: An International Journal (ACIJ),Vol.3, No.3, May 2012.

[14] K. Talattinis, A. Sidiropoulou, K. Chalkias and G. Stephanides, "Parallel Collection of Live Data Using Hadoop," 2010 14th Panhellenic Conference on Informatics, 2010, pp. 66-71, doi: 10.1109/PCI.2010.47.

AUTHORS First Author – Mareng,Marko Mawien Mawien, International School, Changsha University of Science & Technology, Changsha city,Hunan province,China, **Email address:**mawienjasus@gmail.com