© 2014 IJEDR | Volume 3, Issue 1 | ISSN: 2321-9939

Improved Distribution of Workload in Hadoop Yarn

[1]Rajneesh Kumar,
[1]Student,
Department of Computer Application, Faculty of Engineering and Technology, SRM University, Chennai , INDIA

Abstract **- Hadoop YARN is a software framework that supports data intensive distributed application. Hadoop creates clusters of machine and coordinates the work among them. It include two major component, HDFS (Hadoop Distributed File System) and MapReduce. HDFS is designed to store large amount of data reliably and provide high availability of data to user application running at client. It creates multiple data blocks and store each of the block redundantly across the pool of server s to enable reliable, extreme rapid computation. MapReduce is software framework for the analyzing and transforming a very large data set in to desired output. This paper focus on how the replicas are managed in HDFS for providing high availability of data under extreme computational requirement. Later this paper focus on possible failure that will affect the Hadoop cluster and which are failover mechanism can be deployed for protecting the cluster.**

IndexTerms **- Cluster, HDFS, MapReduce, Node Replica, High Availability, Load Balancing.**

I. INTRODUCTION

Dealing with “Big Data” requires – an in expensive, reliable storage and a new tool for analyzing structured and unstructured data. Apache Hadoop addresses both of these problems. Hadoop is the subproject of Lucene under the umbrella of Apache Software Foundation. Hadoop distribute and parallelize data processing across many nodes in a compute cluster, speeding up large computations and hiding I/O latency through increased concurrency. It is well suited for large data processing like searching and indexing in huge data set.

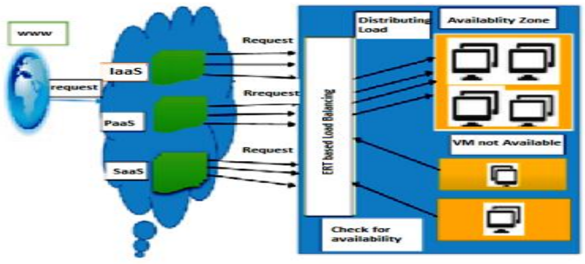
In this paper we have implemented a cloud application platform which enable the whole infrastructure having both virtual and physical components of Hadoop YARN to appear as a single node cluster , highly extensible and reliable system, which has the ability to operate itself and adaptable to the failures and changing load. It can also be provisioned either by an application or simply by a web request.

II. EXISTING SYSTEM

Virtual machine enables the abstraction of an Operating System and Application running on it from the hardware. The interior hardware infrastructure services interrelated to the Clouds are modelled in the simulator by a Datacentre element for handling service requests. These requests are application elements sandboxed within VMs, which need to be allocated a share of processing power on Datacentre’s host components. Datacentres object manages the data centre management activities such as VM creation and destruction and does the routing of user requests received from User Bases via the Internet to the VMs .The Data Centre Controller, uses a VM Load Balancer to determine which VM should be assigned the next request for processing.

III. PROPOSED SYSTEM

To provide the higher degree of satisfaction in the form of throughput and CPU Utilization we have proposed and implemented
an approach for load balancing based upon the expected response time of VM. By using this approach we are distributing our workload across several VM for a successful task flow execution. Fig 1. YARN Based Load Balancer Architecture



IV. IMPLEMENTATION

The Proposed Expected Response Time (ERT) based Load balancing algorithm is divided into three parts. The first phase is the initialization phase. In the first phase, the expected response time of each VM is to be find. In second Phase find the efficient VM, in Last Phase return the ID of efficient VM. The working of algorithm algorithm is shown using the Flow Chart shown in fig 2.

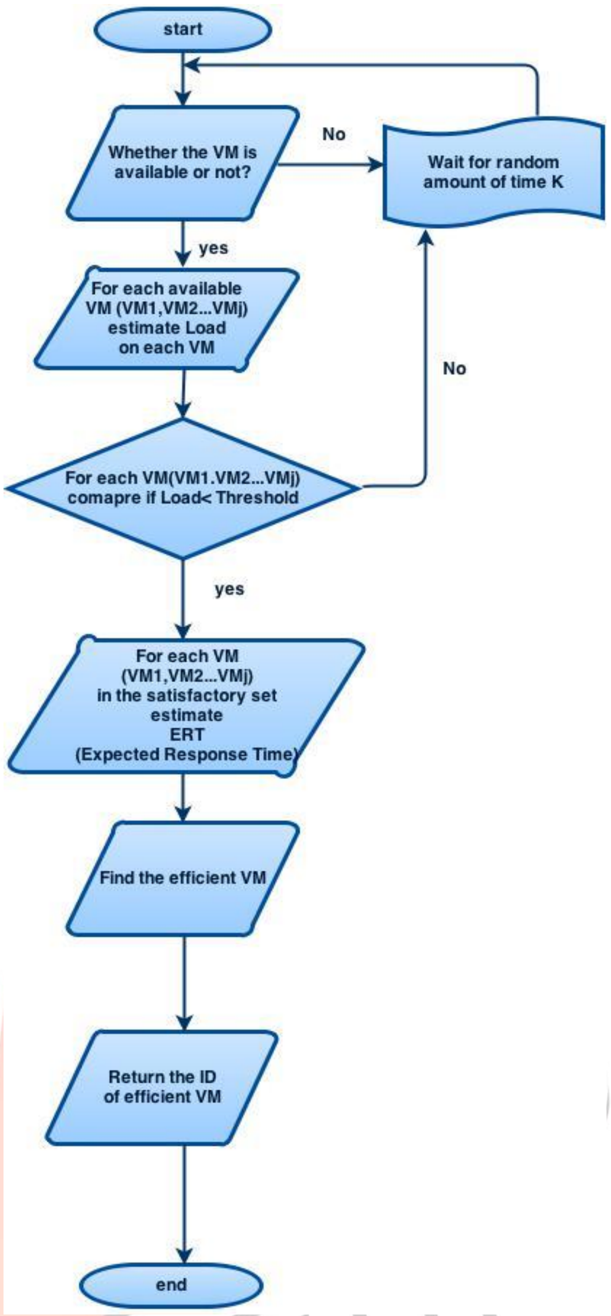

start
Whether the VM is available or not?
No
Wait for random amount of time K
yes
For each available VM (VM1,VM2...VMj) estimate Load on each VM
No
For each VM(VM1.VM2...VMj) comapre if Load< Threshold
yes
For each VM (VM1,VM2...VMj) in the satisfactory set estimate ERT (Expected Response Time)
Find the efficient VM
Return the ID of efficient VM
end

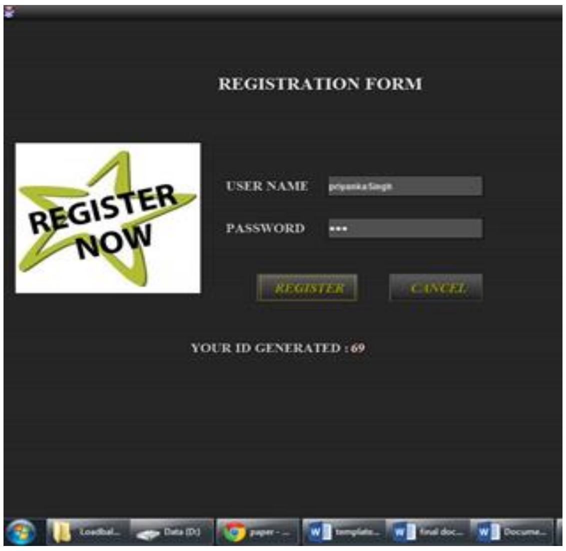
Fig2. Flowchart for ERT based Load Balancing Algorithm.

To distribute the load across several nodes (VMs) we have presented the algorithm as shown in flow diagram represented in figure 3. We are assuming the threshold as a fraction from 0 to 100 where we are taking the default value as 10. Because we are assuming that none of the virtual machine will be idle after initialization.

V. MODULES

The modules that we have implemented are

A. Bucket Creation : Browser allows to easily create Buckets in all regions supported by commercial cloud. Once created a new bucket, consumer can create virtual folders to organize files, and upload and download files to and from commercial cloud database.



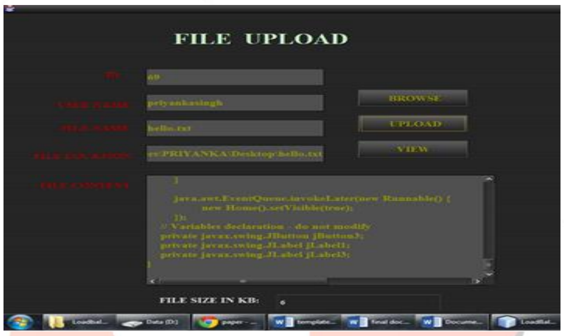
REGISTRATION FORM

USER NAME: priyanka Singh
PASSWORD: •••

YOUR ID GENERATED : 69

Fig 4. Bucket Creation Module

B. Uploading Instances: VM import/export enables to easily import virtual machine image from existing environment to commercial cloud instances and export them back to on-premises environment. This offering allows to leverage an existing investment in the virtual machine that built to meet IT security. configuration management and compliance requirements by seamlessly bringing those virtual machines into commercial cloud as ready to use instances. Consumer can easily export imported instances back to an on-premises virtualization infrastructure, allowing to deploy workload across IT infrastructure.



FILE UPLOAD

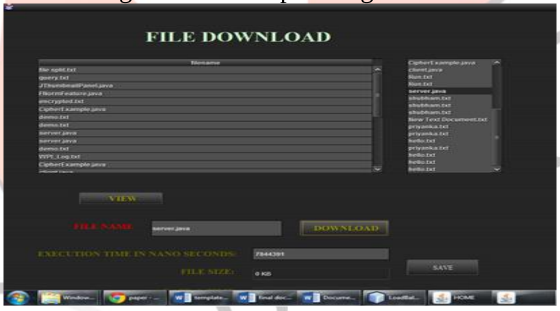
ID: 69
USER NAME: priyankasingh
FILE NAME: hello.txt
FILE LOCATION: e:\PRIYANKA\Desktop\hello.txt
FILE CONTENT:

```
    }

    java.awt.EventQueue.invokeLater(new Runnable() {
         new Home().setVisible(true);
    });
  // Variables declaration - do not modify
  private javax.swing.JButton jButton3;
  private javax.swing.JLabel jLabel1;
  private javax.swing.JLabel jLabel3;
}
```

FILE SIZE IN KB: 6

Fig 5. Instance Uploading Module



FILE DOWNLOAD

filename
file split.txt
query.txt
JThumbnailPanel.java
FNormFeature.java
encrypted.txt
CipherExample.java
demo.txt
demo.txt
server.java
server.java
demo.txt
WPI_Log.txt
CipherExample.java

CipherExample.java
client.java
lion.txt
lion.txt
server.java
shubham.txt
shubham.txt
shubham.txt
New Text Document.txt
priyanka.txt
priyanka.txt
hello.txt
priyanka.txt
hello.txt
hello.txt
hello.txt

FILE NAME: server.java
EXECUTION TIME IN NANO SECONDS: 7844391
FILE SIZE: 0 KB

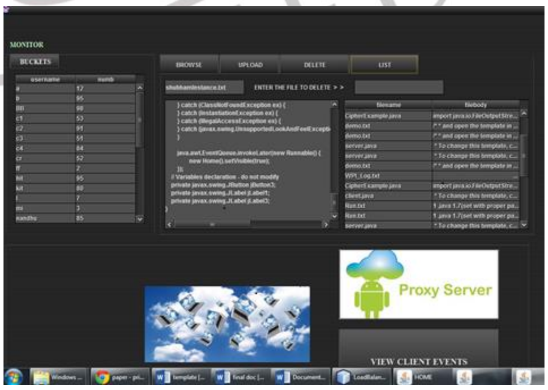
Fig 6. Instance Downloading Module

C. Monitoring Instances: ERT based load balancer provides a reliable, scalable, and flexible monitoring solution that can start using within minutes. No longer need to set up, manage, or scale own monitoring systems and infrastructure. Using this approach admin can easily monitor the istances.it also lets admin to programmatically retrieve consumer instances. It has the features for monitoring data, view graphs, and set alarms to help you troubleshoot, spot trends, and take automated action based on the state of Cloud environment. The Monitoring Module is coded using Java language which is such as:

```
public class monitring extends javax.swing.JFrame
{   public monitring() {
     try {
          Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/loadbalancing", "root", "root");
          st = con.createStatement();
       } catch (Exception e)
 {
          e.printStackTrace();
```

```
        }
        initComponents();
       }
 private void jButton5ActionPerformed(java.awt.event.ActionEvent evt)
 {
         try {
           FileInputStream fis = null;
                 String filebody=jTextArea2.getText();
                String filename=jTextField1.getText();
           FileWriter fstream = new                    FileWriter("proxyserver\\"+filename);
  BufferedWriter out = new BufferedWriter(fstream);
  out.write(filebody);
               System.out.println(filebody);
                    File file = new File(jTextField1.getText());
                    fis = new FileInputStream(jTextArea2.getText());
                    String str=jTextArea2.getText();
            st.executeUpdate("insert into upload values('admin','admin','"+filename+"','"+filebody+"','fs','time')");
             FileOutputStream fos = new FileOutputStream("F:\\"+filename);
               fos.write(str);
                   fos.close();
            out.close();
            JOptionPane.showMessageDialog(this, "File Has Been Uploaded Successfully");
        }
 catch (Exception ex) {
Logger.getLogger(upload.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
        proxyserver pr=new proxyserver();
        pr.setVisible(true);
    }
    private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
   clientevent ce = null;
        try {
            try {
                ce = new clientevent();
            }
 catch (Exception ex)
{
Logger.getLogger(monitring.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
```

In this way using this module Admin can monitor log event as well as characteristic of virtual machines on several instances.



username	numb
a	17
b	95
bb	90
c1	53
c2	91
c3	51
c4	84
cr	52
ff	2
kk	95
kit	80
l	7
mi	3
nandhu	85

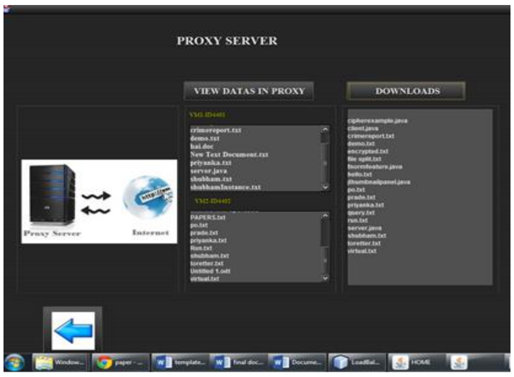
shubhaminstance.txt

```
    } catch (ClassNotFoundException ex) {
    } catch (InstantiationException ex) {
    } catch (IllegalAccessException ex) {
    } catch (javax.swing.UnsupportedLookAndFeelExcepti
    }

    java.awt.EventQueue.invokeLater(new Runnable() {
        new Home().setVisible(true);
    });
 // Variables declaration - do not modify
 private javax.swing.JButton jButton3;
 private javax.swing.JLabel jLabel1;
 private javax.swing.JLabel jLabel3;
}
```

filename	filebody
CipherExample.java	import java.io.FileOutputStre...
demo.txt	/* * and open the template in ...
demo.txt	/* * and open the template in ...
server.java	* To change this template, c...
server.java	* To change this template, c...
demo.txt	/* * and open the template in ...
WPI_Log.txt	...
CipherExample.java	import java.io.FileOutputStre...
client.java	* To change this template, c...
Ran.txt	1 java 1.7(set with proper pa...
Ran.txt	1 java 1.7(set with proper pa...
server.java	* To change this template, c...

Fig 6. Monitoring Instance Module

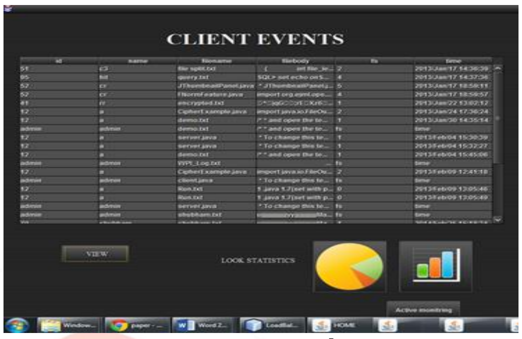


crimereport.txt
demo.txt
hai.doc
New Text Document.txt
priyanka.txt
server.java
shubham.txt
shubhamInstance.txt

PAPERS.txt
po.txt
prade.txt
priyanka.txt
Run.txt
shubham.txt
toretter.txt
Untitled 1.odt
virtual.txt

cipherexample.java
client.java
crimereport.txt
demo.txt
encrypted.txt
file split.txt
fnormfeature.java
hello.txt
jthumbnailpanel.java
po.txt
prade.txt
priyanka.txt
query.txt
run.txt
server.java
shubham.txt
toretter.txt
virtual.txt

Fig 7. Monitoring Availability of VM



id	name	filename	filebody	fs	time
51	c3	file split.txt	{ ... int file_le...	2	2013/Jan/17 14:36:39
95	fol	query.txt	SQL> set echo onS...	4	2013/Jan/17 14:37:36
57	cr	JThumbnailPanel.java	* JThumbnailPanel.j...	5	2013/Jan/17 18:58:11
52	cr	FNormFeature.java	import org.ejml.ope...	4	2013/Jan/17 18:58:57
41	rr	encrypted.txt	[illegible]	1	2013/Jan/22 13:02:12
12	a	CipherExample.java	import java.io.FileOu...	2	2013/Jan/24 17:36:24
12	a	demo.txt	/* * and open the te...	1	2013/Jan/30 14:35:14
admin	admin	demo.txt	/* * and open the te...	fs	time
12	a	server.java	* To change this te...	1	2013/Feb/04 15:30:39
12	a	server.java	* To change this te...	1	2013/Feb/04 15:32:27
12	a	demo.txt	/* * and open the te...	1	2013/Feb/04 15:45:06
admin	admin	WPI_Log.txt	...	fs	time
12	a	CipherExample.java	import java.io.FileOu...	2	2013/Feb/09 12:41:18
admin	admin	client.java	* To change this te...	fs	time
12	a	Run.txt	1 java 1.7(set with p...	0	2013/Feb/09 13:05:46
12	a	Run.txt	1 java 1.7(set with p...	0	2013/Feb/09 13:05:49
admin	admin	server.java	* To change this te...	fs	time
admin	admin	shubham.txt	[illegible]	fs	time

Fig 8. Monitoring Client Events

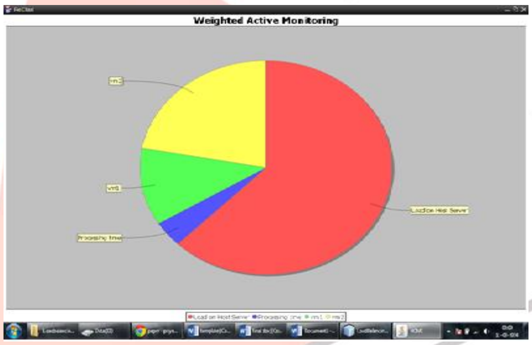


Fig 9. Weighted Active Monitoring

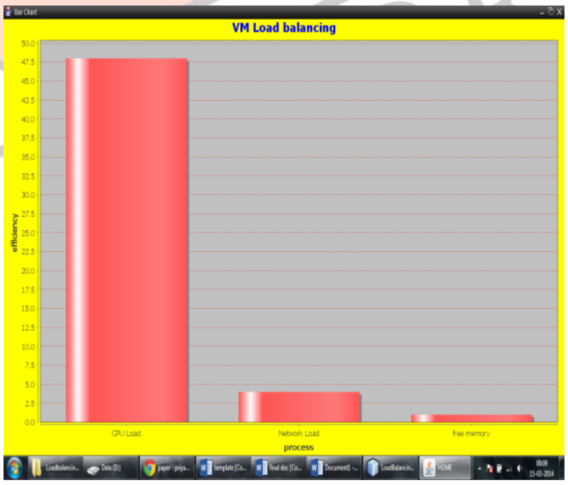


Fig 10. Monitoring the ERT and efficiency of each VM for request

D. Performance

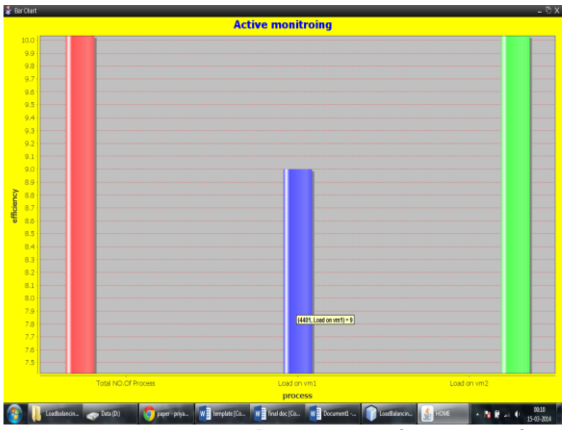


Fig 11. For each request performance of VMs

VI. CONCLUSION

In this paper we have proposed and implemented an ERT based load balancing concept through which the request from the cloud consumer can be handled efficiently and for the simultaneous requests load can be distributed so that to minimize the overhead and to avoid the performance bottleneck at any node. This approach also provide active monitoring for the clients events .admin can monitor all the instances imported and exported by the consumers and also can monitor event log for several consumers at each virtual machine. at last we are analyzing the performance of the virtual machines in terms of efficiency and throughput based upon different load at several time slot. our future work is to implement ERT based algorithm in hybrid cloud environment. and to monitor the performance of ERT based algorithm in hybrid cloud.

REFERENCES

[1] Bowman, M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems. .
[2] Ding, W. and Marchionini, G. 1997 A Study on Video Browsing Strategies. Technical Report. University of Maryland at College Park.
[3] Fröhlich, B. and Plate, J. 2000. The cubic mouse: a new device for three-dimensional input. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems
[4] Tavel, P. 2007 Modeling and Simulation Design. AK Peters Ltd.
[5] Sannella, M. J. 1994 Constraint Satisfaction and Debugging for Interactive User Interfaces. Doctoral Thesis. UMI Order Number: UMI Order No. GAX95-09398., University of Washington.
[6] Forman, G. 2003. An extensive empirical study of feature selection metrics for text classification. J. Mach. Learn. Res. 3 (Mar. 2003), 1289-1305.
[7] Brown, L. D., Hua, H., and Gao, C. 2003. A widget framework for augmented interaction in SCAPE.
[8] Y.T. Yu, M.F. Lau, "A comparison of MC/DC, MUMCUT and several other coverage criteria for logical decisions", Journal of Systems and Software, 2005, in press.
[9] Spector, A. Z. 1989. Achieving application requirements. In Distributed Systems, S. Mullender.
[10] Bowman, M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems. .
[11] Ding, W. and Marchionini, G. 1997 A Study on Video Browsing Strategies. Technical Report. University of Maryland at College Park.
[12] Fröhlich, B. and Plate, J. 2000. The cubic mouse: a new device for three-dimensional input. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems
[13] Tavel, P. 2007 Modeling and Simulation Design. AK Peters Ltd.
[14] Sannella, M. J. 1994 Constraint Satisfaction and Debugging for Interactive User Interfaces. Doctoral Thesis. UMI Order Number: UMI Order No. GAX95-09398., University of Washington.
[15] Forman, G. 2003. An extensive empirical study of feature selection metrics for text classification. J. Mach. Learn. Res. 3 (Mar. 2003), 1289-1305.
[16] Brown, L. D., Hua, H., and Gao, C. 2003. A widget framework for augmented interaction in SCAPE.
[17] Y.T. Yu, M.F. Lau, "A comparison of MC/DC, MUMCUT and several other coverage criteria for logical decisions", Journal of Systems and Software, 2005, in press.
[18] Spector, A. Z. 1989. Achieving application requirements. In Distributed Systems, S. Mullender