International Research Journal of Advanced Engineering and Science

ISSN: 2455-9024

Load Balancing Using Distributed Cluster

Mohammed Muddasir N[1], Pooja R Palankar[2], Rakshit Raje Urs K K[3]

[1]Assistant Professor, Dept of IS&E, VVCE, Mysore, India

[2, 3]Student, Dept of IS&E, VVCE, Mysore, India

Abstract**—** *Access to high performance service on the web is a major challenge faced by web service providers. Performance of web service could be improved with high end configured fast servers. This could be an ideal situation if the web service is accessed consistently throughout the year. But these high end servers would be a burden on organization providing web services with variations in the web traffic pattern. These organizations are looking for alternatives to provide fast web access to their respective traffic patterns. This could be done by distributing the load across a cluster configured with only commodity hardware instead of high end expensive servers. We present a proof of concept for the above using single server performance measured using jmeter and distributed cluster using hadoop and testdfs I/O provided by the apache foundation. We are comparing the access speed of files on a server with access speed of files on distributed cluster. We are taking throughput as the criteria to assess individual system performance.*

Keywords**—** *Hadoop, MapReduce.*

I. Introduction

Web service has become the most preferred business in today's fast growing internet age with the advent of online shopping, social media, online customer services like booking movie tickets, booking tickets for sports events like cricket or football. Web service providers have to deploy expensive servers that are reliable and tolerant to failure and also take back up from time to time to recover from any loss of data. At times there are situation where web service providers need to provide extra services only for one or two hours or probably one or two days. Let's take an example when famous Indian cricketer Sachin Tendulkar played his last match there was a huge rush at the booking counters. The online booking systems almost crashed [1-3] by the number of hits. Online booking providers faced difficulty in handling the traffic and only way the online booking websites could survive the crash are by deploying additional servers. Cost of deploying new servers was very high and these servers are an underutilized resource the next day of match because there is not much traffic after this event. The online booking provider could have got his share of profit, but deploying a new server cost a lot of money. In this scenario we could use a hadoop cluster as a web service provider. When there is need for more service new machine could be added to the cluster and once there is not much requirement these new machines could be inactivated on the cluster. The advantage of hadoop cluster is it's made up of commodity hardware and not state of the art expensive server and hence cost of deploying them and de deploying is very less compared to deploying servers.

To ascertain this we have developed a proof of concept using hadoop, jetty server with apache and jmeter performance testing tool. We have taken five node hadoop clusters one is a master and four are other machines are slaves. We have checked the performance of reading the data from these five nodes. We run the TestDFSIO mapreduce program given by apache foundation with hadoop. This program first generates the file that is its doing the write operation then it will read those files. Writing and reading can be selected in the parameter list provided for running the program. Also we have deployed a jmeter performance testing tool on other machines other than the cluster machines that will help us know the capacity of reading data from one machine without hadoop. We have taken through put as the parameter. Through put is the number of request that could be served per user. As the number of user increases gradually the throughput decreases. The service we focus here is only on reading of data. We have developed a 2MB web page in case of standalone system that we are considering as a server and increase the number of users who access this web page. By increasing the number of users we find the through put decreases, this we have presented in our results. In case of hadoop we have not created a web page but checking its reading capability by increasing the number of files it could read at a time and gradually increasing the number of files will again decrease the through put. This through put could be improved in case of hadoop by adding new nodes but in case of standalone server machine it's not feasible because of the cost involved. Adding a new machine is simple and cost effective as compared to adding a new server.

II. Literature Review

Hadoop is an echo system of open source projects from Apache. The main feature of hadoop is distributed file system called as Hadoop Distributed File System (HDFS) [4]. The idea of distributed file system came from the limitation of read/write access on a hard drive. In simple terms the more the size of the hard drive the longer it takes to read/write on the entire hard drive. Reading one terabyte of data all at once with the available speed of 100Mb/s takes atleast two and half hour to complete. Reading this data stored in 100 disks in parallel takes about 2.5 mins that is huge reduction in reading time [5]. Hadoop accomplished just that reading data from a massive parallel computing environment. Hadoop is a master slave architecture one of the machines is designated as master and other machines as slaves. Master is responsible only for knowing where the data is stored and keeping track of the slaves. Slaves are responsible for storing the data. The setup is called a hadoop cluster. Big organizations like Yahoo run 4000 node and Facebook run 2300 node cluster [6], node is nothing but a single machine. Machines used are commodity hardwares hence chances of break down is relatively high as compared to sophisticated servers so what hadoop does is it makes copies of the data and stores in more than one node so that if one node goes down data can still be recovered from the other available nodes. Data to be stores is partitioned in to chunks. Each chunk of data is replicated based on a replication factor property in HDFS. Industry standard is replication factor or 3. Each chuck of data is stored on different machines i.e. on slave machines. Master keeps track of the chuck where it is stored and which data source the chuck belongs to. If the master node fails individual chunks cannot be interpreted for the original data, so the master is a very important part of the echo system. Back up of the master is made using secondary master if master goes down secondary master will be used to recover the chunks of data into original data. The chunks will be stored on different slave machines and master will keep track of this information and whenever a slave machine fails the chunk is lost from that machine but could be got back from other machines that would have a copy of the chuck. This entire task is done by the master machine. Hadoop uses the MapReduce [7] programming paradigm with implementations in JAVA, PIG, HIVE. We can write MapReduce programs in JAVA that is a application programming language, PIG that is a scripting language, and also HIVE which is a query language. Hadoop gives flexibility for different types of developers with their area of expertise to work on HDFS. Whether one writes java, or pig, or hive all code is run in the form of Mapper and Reducers that work on name value pair. The job of mappers is to identify key value pairs and reducer is to aggregate related key value pairs. This could be understood using simple word count program. Considering a set of files having some text word count will count the accordance of each word in all the documents. Mapper will just identify each word and give a count of 1 to each word. First time a word appears Mapper will put 1 in front of that word. Next time the same word appears it will again put 1 in front of that word. Reducer job is to take each word and sum up all the 1's for that word. Hadoop uses the jetty web server. Jetty is a machine to machine communication with large software firms. It an open source project developed by the eclipse foundation. We run TestDfsIO for testing the performance of read/write on a hadoop cluster. This will give throughput, average and standard deviation of input output. Throughput mb/sec for a TestDFSIO job using N map tasks is defined as follows [8]. The index 1 <= i <= N denotes the individual map tasks: Throughput (N) = $\sum_{i=0}^{n} filesize(i) / \sum_{i=0}^{n} time(i)$

Websites are built on a 3-tier or n-tier architecture that has user interface layer like web browser, single or multiple application logic processing layers and finally the data base layer. Database layer is storing of data on the traditional database management system. Finally the DBMS is the one responsible for writing and reading the data. DBMS reads data from a single hard drive which has limited speed of reading the data if it's huge. Here the volume is concerned with the number of request rather than the number of users. That is if we take a website service the size of the files may not be huge but if the number of request are huge then we have to read the file multiple times from the hard drive. This reading from the single source (server) will limit the number of users that could get the file in a reasonable time. If suppose the users increase then everyone will be affected and the users have to wait a long time. All this because we are still reading from the single hard drive that has limited access time. If it's a static web site then reading and writing data depends on the operating system. Although DBMS is not involved here again the same problem of single hard drive surfaces that will limit the number of users that could be served. Throughput in jmeter is calculated as (number of requests/total time). Total time is start time of first request and the end time of the last request [9]. Any gap between the requests is also considered because it adds to the latency and increases the load on the server.

III. Related Work

There are several comparative studies that try to compare the performance of hadoop. One such study [10] compares the performance of storing image data on HDFS with Meta data on Hbase. They call this as a hybrid architecture where they have considered random reads and random writes to read Meta data from Hbase and MySQL and retrieve the image files from HDFS. They have compared Hbase-HDFS architecture with MySQL-HDFD architecture. They have stored the image on HDFS and Meta data on Hbase and MySQL. In another study [11] they have compared the performance of hadoop cluster using rack aware data nodes. They have segregated the data nodes as to belonging to different racks. Experiments are performed by varying the replication factor from 3 which is the default in hadoop up to 8. The replication factors are set to a value that is greater than the number of nodes in the cluster. The first result is about computations speed calculating the value of PI using multiple replication factors is shown they show that as the replication factor increases the multiple map tasks are performed and reduces the time to compute. The next experiment is results are shown for large data files with multiple replication factors where the large data could be read fast for greater replication factor. Here for certain threshold value the performance increases but after the threshold value of replication factor the performance falls. In another study [12] the objective is how to deploy hadoop for large number of small files. Here they mention hadoop is used for processing large amount of data which usually is stored in one file but some organizations have huge data but in different files stored as small chunks so how hadoop could be leveraged for these types of data. They have build index after combining data from multiple small files in to one large file, they have considered files that a close (neighboring files) and that which have latest updates. Their experiments show this optimization technique improves the performance of reads and writes in a scenario like this. They have in the experiment environment used 5 node closeted to tune the files operations like storing, reading, deleting and updating. The tuning here refers to the technique of combining and indexing the various small files into big files. They show the result of both normal HDFS and tuned HDFS it shows a huge difference.

IV. IMPLEMENTATION

Implementation involved setting up hadoop cluster with five machines and configuring jmeter on a single machine. For the distributed environment we had set up a Hadoop cluster that has 5 node with 1 node being the master and 4 nodes being the slaves. All the machines had ubuntu linux version 12.04 LTS. Each machine had to be configured with password less SSH login. With passwordless SSH we could have all the machines talk to each other without providing password using a secured shell. This is the key for configuring hadoop on any number of machines. If every time we had to enter password to access a machine then it would be tedious as well as time consuming and defeat the whole purpose of distributed cluster. The idea is all these five machines behave as one unit or at least look to work as one unit with the master taking control of the entire environment. If the master fails then we cannot make sense of the data in slave nodes. Although secondary master is configured as a replacement for master in case master node fails we have not implemented secondary master in our setup. We have installed jdk version 7 and java 6 on all machines. Formatting the HDFS filesystem via the namenode before we start our new multi-node cluster, we must format Hadoop's distributed filesystem (HDFS) via the name node. We need to do this the first time when we set up a hadoop cluster. Formatting is done using the dfs format command that will initialize a new directory on the name node. After formatting the HDFS start command is used to start the daemon processes on master and slave nodes. The daemon on the master are called name nodes and that on slave are called data nodes. Also MapReduce daemons are started on the master called as job tracker and same on slave is called as task tracker. Once all the deamons are running the environment is set for conducting the experiment. We use the browser based interface for checking live nodes and dead nodes. Data nodes send heart beat messages to master node at intervals of 3 seconds. We could check the same using the JPS command provided in java. We next run TestDFSIO on write mode to create the files on the hadoop cluster. These are the files which we are going to read and measure the throughput. We run the above program and created the required amount of files as shown in the results section. Once all the files were created we run the same program in read mode and record the throughput. We have done this for different set of files as shown in the result section of this paper.

Next we measured the throughput on a single machine running jmeter. We created a web page of 2MB and did a performance test of this page on 100 through 500 users as shown in the result section and measure the throughput.

V. RESULTS

We have measured the throughput on hadoop cluster and local machine. Table I shows the number of files used for measuring the throughput and throughput values for 2node, 5node hadoop cluster and local machine. Figure 1 shows bar graph of the through put measured on a single machine for same number of files. Figure 2 shows the bar graph of the throughput measured for different number of files on 2 node hadoop cluster and 5 node hadoop clusters. In the second graph the throughput is given by jmeter. The throughput decreases in case of local machine as the number of files increases as shown in figure 1. In case of hadoop also the throughput decreases as seen in the case of 2 node cluster, but as we increase the number of nodes from two to five the throughput increases. This shows that using hadoop clusters we could achieve higher throughput by just augmenting additional nodes to the existing cluster. Whereas the same cannot be done for servers because of the cost associated with adding a new server usually high. Unit of measurement for throughput is mb/sec.

TABLE I. Number of files used for read on hadoop cluster and local machine.

Files/Nodes	2 Node Hadoop Cluster	5 Node Hadoop Cluster	Local Server
100	1.79	1.930	0.00553
500	0.438	0.454	0.00528
1000	0.16	0.250	0.00495
1500	0.054	0.058	0.00287

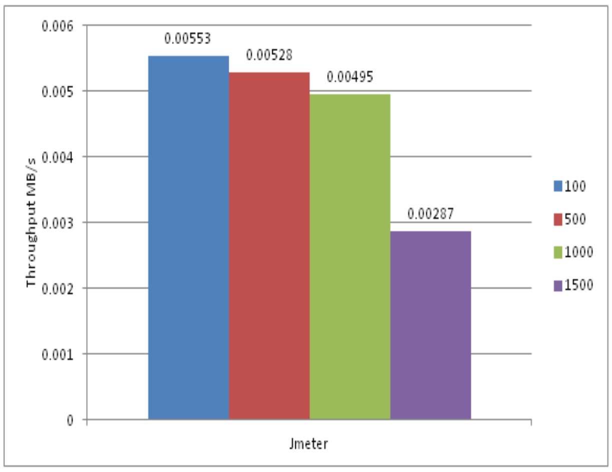

Fig. 1. Throughput on Jmeter.

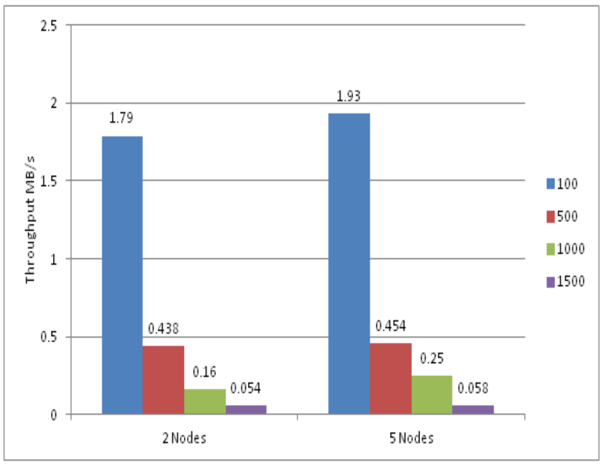

Fig. 2. Throughput on Hadoop Cluster.

VI. CONCLUSION

We could conclude from the result that performance in terms of throughput could be enhanced by adding more nodes on a hadoop cluster. The load distribution could be used to provide web service in case the traffic patterns are varying in nature like the example given in the introduction section. Performance is a major challenge in today's age of growing demand for information. More the faster and accurate information is given its better for decision makers. Hadoop comes as performance enhancer in a cost effective way by not using sophisticated hardware's but using commodity hardware's. Hadoop cluster are mainly used in big data analytics.

REFERENCES

[1] M. Report, "http://www.dnaindia.com," 2013. [Online]. Available: http://www.dnaindia.com/sport/report-online-ticketing-site-crashes-sachin-tendulkar-fans-bitter-1917500.

[2] M. Report, "http://www.medianama.com," 2013. [Online]. Available: http://www.medianama.com/2013/11/223-kyazoonga-down-sachin-last-test/.

[3] M. Report, "http://www.rediff.com," 2013. [Online]. Available: http://www.rediff.com/cricket/report/sachin-tendulkar-200th-test-website-crashes-ahead-of-online-ticket-sales/20131111.htm.

[4] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies*, pp. 1-10, 2010.

[5] T. White, Hadoop Definitive Guide, 2009.

[6] H. Mallipeddi, "https://www.quora.com," Facebook, 2010. [Online]. Available:https://www.quora.com/What-are-some-of-the-largest-Hadoop-clusters-to-date.

[7] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation*, vol. 6, 2004.

[8] M. G. Noll, "http://www.michael-noll.com," Applied Research. Big Data. Distributed Systems. Open Source., 11 apr 2011. [Online]. Available:http://www.michael-noll.com/blog/2011/04/09/benchmarking-and-stress-testing-an-hadoop-cluster-with-terasort-testdfsio-nnbench-mrbench/.

[9] E. Halili, Apache JMeter, 2008.

[10] M. N. Vora, "Hadoop-HBase for large-scale data," in *International Conference on Computer Science and Network Technology*, 2011.

[11] M. Ramane, S. Krishnamoorthy, and S. Gowtham, "An experimental evaluation of performance of a hadoop cluster on replica management," *International Journal on Computational Sciences & Applications*, vol. 4, no. 5, pp. 87, 2014.

[12] X. Liu, J. Han, Y. Zhong, C. Han, and X. He, "Implementing WebGIS on hadoop: A case study of improving small file I/O performance on HDFS," *IEEE International Conference on Cluster Computing and Workshops, CLUSTER '09.*, New Orleans, 2009.