International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Enhancing the Hadoop Performance through Data Placement in Heterogeneous Hadoop Cluster

A Ankita Poovaiah[1], Gopal B[2]

[1]M.Tech Student, Department of CSE, Mangalore Institute of Technology and Engineering, Mangalore, Karnataka, India

[2]Assistant Professor, Department of CSE, Mangalore Institute of Technology and Engineering, Mangalore, Karnataka, India

Abstract: ***In the present world large volumes of data are getting generated and these records and data details have to be maintained for future purpose. Keeping these large bulks of data and using them becomes difficult. To overcome this and make it easy to store, use and work with it a tool called Hadoop is used. Hadoop uses the concept of a cluster that is many small nodes together form a cluster. Nodes with varying configurations (like varying RAM sizes, processors) form a heterogeneous cluster. Data placement technique in heterogeneous cluster is complicated. The data placement technique in heterogeneous cluster helps in the efficient use of resources and when combined with the MapReduce programming model increases the performance. Data placement can be done by forming racks. In this work we enhance the performance of Hadoop in heterogeneous cluster by first creating racks for data placement and then modifying certain parameters of Hadoop tool. The techniques are implemented and evaluated in Hadoop 1.0.3.***

Keywords: Big Data, Hadoop, Heterogeneous Cluster, Data Placement.

1. Introduction

Big data is a combination of volume, variety and velocity. Big data is generated mainly due to the high usage of social networks, Internet. It creates huge volumes of data which can go up to petabytes and terabytes. It has a variety of data that can be of the form structured, unstructured or semi structured. Hadoop is a tool used to manage this big data so that the data can be used and handled in a very efficient and easy way. Hadoop is an open source tool and it mainly uses a MapReduce programming model which uses JAVA. Hadoop works well in homogeneous clusters where all the nodes have the same configurations. In homogeneous clusters Hadoop splits the input data into equal size or data blocks (by default 64MB) on the Hadoop distributed file system (HDFS) across all the nodes present in the cluster [1]. In a heterogeneous cluster the nodes will have different configurations like varying RAM sizes, processors. Data placement technique when applied in a heterogeneous cluster helps to efficiently utilize the available resources. Data placement in a heterogeneous cluster can be done by creating racks. These racks are created based on the nodes internet protocol addresses. The Hadoop performance can be boosted by modifying the parameters which are present in the configuration file of Hadoop. MapReduce is a flexible data processing tool used in Hadoop and it works in two phases that is map and reduce phase [2]. The paper is organized in the following sections. Section II gives the related work and section III provides the proposed enhancements. In section IV and V experimentation results are provided along with the conclusion respectively.

2. Related Work

Many research works are conducted on data placement and enhancement of Hadoop tool. This segment lists various related work towards the performance enhancement in Hadoop.

Jiong Xie et.al. [3] provide the data placement in heterogeneous cluster where initially the same task is assigned to all the nodes and it provides the response time. Based on this response time the computing ratio is assigned to the nodes and these computing ratios define the data to be placed and tasks to be allotted to the nodes. The data placement varies for every application. This works well only for a particular application.

Madhavi Vaidya [4] in parallel processing of cluster by MapReduce states that MapReduce library automatically parallelizes the computation. It even handles issues like fault tolerance, data distribution and load balancing. It aims to provide the overview of a MapReduce programming model and its applications. The scalability of MapReduce is proven to be very high because a job in MapReduce model is partitioned into numerous small tasks running on multiple machines in a large scale cluster. The data locality issue in heterogeneous environments can reduce the MapReduce performance when it is run on multiple nodes.

B.ThirumalaRao et.al. [5] states in performance issues of heterogeneous Hadoop clusters in cloud computing how large volumes of data can be stored in a reliable and inexpensive way. It also uses a new tool to analyze the structured and the unstructured data. It describes the hardware parameters that affect the Hadoop performance. It provides the performance challenges of Hadoop in heterogeneous clusters.

3. Proposed Enhancements

We propose the techniques for enhancing the Hadoop performance in heterogeneous cluster. Initially the Hadoop cluster is set up by considering the nodes with varying configurations like different Random access memory sizes and processors (like dual core or quad core processors). The cluster has a master node and can have any number of slave nodes. After the cluster is set up the administrators have to know the configurations of each master and its slave nodes.

3.1 Data Placement and Distribution

After the multi node cluster is set up [6] and a Hadoop file is created the following process has to be done. The IP addresses of all the slave nodes have to be maintained along with the configurations. Here, the data assignment will be done manually by creating racks. Initially, a topology script has to be created in the Hadoop's conf file, this helps to determine the rack location of the nodes. In the topology script file the Hadoop path has to be set and this script must be executable. For this topology script to get the rack details another data file has to be created in the same path and in this data file the IP addresses of the slave nodes have to be added. It should be added according to the way the administrator wants the data to be stored in the nodes. A better performing node like (larger RAM size or better processing node) can be given the first location and the others accordingly. The data gets stored accordingly in the particular racks. So, better performance racks are provided with more data and so on.

After this the number of requests can also be done manually for each node by providing the value in the core site of Hadoop. By default it will be hundred requests but this can be increased or decreased based on the nodes computing capacity. Then, the HDFS and MapReduce have to be restarted. This is how the data placement and data distribution will be done manually in heterogeneous clusters. This helps in the better utilization of the available resources clusters. This helps in the better utilization of the available resources.

3.2 Performance Enhancements

The performance of Hadoop can be enhanced using the following three techniques. That is increasing the number of mapper and reducer slots, reusing the JVM and increasing the IO sort buffer and factor[7].When these techniques are used individually performance can be improved. These techniques are combined with each other and then performance is measured. It is seen that the performance enhancement is high when the approaches are used in a combined manner rather than an individual approach.

a) Increasing the number of mapper and reducer slots

The Hadoop path will have a conf file present in it. In that conf file when listing of contents is done an xml file called mapred-site will be present. In this mapred-site file there will be two factors present. The first factor is mapred. tasktracker.map. tasks. maximum and the second is mapred.tasktracker.reduce.tasks.maximum. These constraints can be calculated using the following formulas.

Maximum number of = (CPU cores – 1 reserved core for mapper slots Hadoop daemon) * x

In the above equation x is the value for the hyper threading factor. If the hyper threading factor is deactivated then the value of x is 0.95 and if it is activated the value of x becomes 1.75. Then once the number of mapper slots is calculated the cluster capacity is computed using the following formula:

Cluster mapper = Maximum number of mapper slots * number capacity of nodes

Similarly, for the next parameter the following formulas are used:

Maximum number of = (CPU cores – 1 reserved core) * x reducer slots

Here again x determines the hyper threading factor. If enabled then 1.75 and if disabled then multiply by 0.95. Then the cluster reducer capability is computed as follows:

Cluster reducer = Maximum number of reducer slots * capacity number of nodes

The above calculated values are then provided to the Hadoop's conf file using the property and its values in the mentioned parameters. This can be executed for a normal word count program. To get better performance results, inputs of varying sizes can be given.

b) Reusing the Java Virtual Machine (JVM)

Reusing is a technique for minimizing sources such as CPU, memory space. When we deal with loads of data records, it is always cheaper to reuse an existing instance rather than creating a new one. By default Hadoop creates a new JVM to run the map or reduce task. The Hadoop's path configuration file is used again and the parameter mapred.job.reuse.jvm.num.tasksis modified. When this factor is allowed by setting a value multiple tasks can be executed sequentially with one JVM. Change this variable to run the desired number of tasks. For instance, if the value is fixed to 2 then it runs two tasks with single JVM. If the value is fixed as -1, the jobs that the JVM can execute are boundless. This technique can be implemented for any application and performance can be measured.

c) Increasing the Sort Buffer Size and Sort Factor

Every job when executed writes the output to the buffer. Once the buffer reaches the threshold value the tasks start writing the output to the disks. Maximizing the buffer size minimizes the spills to the disk. This helps to decrease the IO time on both mappers as well as reducer sides. The constraints are agreed for the value in the **Hadoop** path's conf file. In this, an xml file will be present called as mapred-site. The two parameters used here are io.sort.mb and io.sort.factor can be increased so that more memory is allocated to the merging and sorting processes. If the allotted memory to the jobs is not enough to complete the job then garbage collector activities will increase. To avoid more garbage collector activity and the task out of memory exception, we should set io.sort.mb parameter to a value more than 0.25 and less 0.50. This parameter is linked with the mapred.child.java.opts. By default partial memory is allocated to mapred.child.java.optsis assigned to io.sort.mb parameter. The formula to calculate mapred.child.java.opts factoris:

(Map task maximum + reduce task maximum) * Memory to allocate (in MB) < available RAM – reserved memory.

To set the sort buffer the following formula is used:

io.sort.mb=10*io.sort.factor

This value is then set in the Hadoop path's configuration file and any application can be executed. This is run for a word count application.

d) Combined Approach

Here, the two techniques that are increasing the number of mapper and reducer slots and reuse of JVM are combined. Both these procedures factors are set in the Hadoop path's conf file and any application can be executed to measure the enactment. After applying the combined method for any task run it takes less response time and hence provides a better performance for a large input file or data set.

4. Experimentation Results

4.1 Cluster Setup

We used a minimum of three nodes to set up a multi node cluster [6]. Once the multi node cluster is established which will consist of a master node and many slave nodes connected to each other by means of IP addresses. We have to get the path of the Hadoop file. In the Hadoop file the configuration file has to be located and in this file a topology script file has to be created for the racks. This topology file has to be an executable one. In this script file the Hadoop path has to be specified and in the same conf file another data extension data file has to be created which will contain all the IP addresses of the slave nodes. The IP addresses should be arranged in ascending order based on the node configuration details. A high configuration node should be given priority so that when the racks are created large sized data will get placed on the high configuration node. This is how data placement is done and this helps in the efficient utilization of the resource.

4.2 Performance Evaluation

We took an application of word count which takes a book file as an input. The book file is divided into two input sources one is a small book file about 25MB and a book file of size around 155MB. The parameters were evaluated and the performance was measured.

Consider the Hadoop path's configuration file and the parameters will be modified in the xml file of mapred. First for increasing the mapper and reducer slots, we considered dual core processor and two slave machines. Hence, CPU core is four and the machine does not support hyper threading factor. Therefore, maximum number of mapper slots and cluster mapper capacity will be:

Maximum number of = (CPU cores – 1 reserved core for mapper slots Hadoop daemon) * x

Maximum number of = (4 – 1) * 0.95 = 2.85 ≈ 3mapper slots

Cluster mapper = Maximum number of mapper slots capacity * number of nodes
= 2.85 * 2 = 5.7 ≈ 6

Similarly, the maximum number of reducer slots and cluster capacity is computed as below:

Maximum number of reducer slots = (4-1) * 0.95
= 2.85 ≈ 3
Cluster reducer capacity = 2.85 * 2
= 5.7 ≈ 6

Next technique reusing the JVM, here the parameter mapred.job.reuse.jvm.num.tasks is modified. We set the value to -1 where the number of tasks that the JVM can execute is not limited. Then the first technique and the second techniques are combined and the performance is measured and the graph is plotted for these parameters before and after modifications.

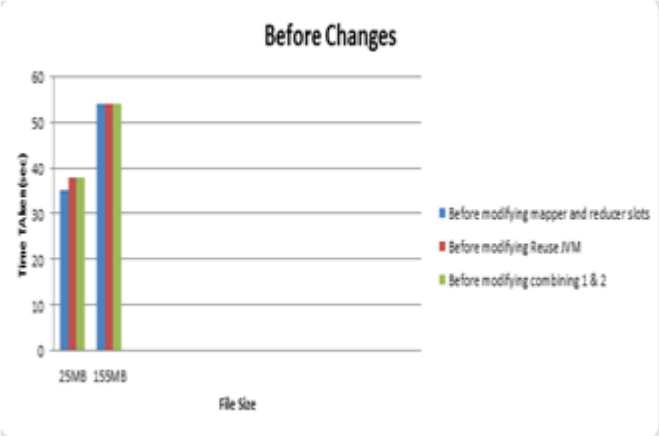

Figure 1: Before modifying the parameters comparing the performance

Here, before the parameters are modified the response time the word count application takes for the book files of varying sizes is more. The response time indicates the performance. The lesser the response time the better the performance. After applying the changes and modifying the parameters in the xml file of Hadoop's configuration the values are added as follows:

```
<property>
<name>mapred.tasktracker.map.tasks.maximum</name>
<value>3</value>
</property>
<property>
<name>mapred.tasktracker.reduce.tasks.maximum</name
>
<value>3</value>
</property>
<property>
<name>mapred.job.reuse.jvm.num.tasks</name>
<value>-1</value>
</property>
```

The graph below shows the changes after adding the properties above in the configuration file and shows less response time providing a better performance for files of varying input sizes.

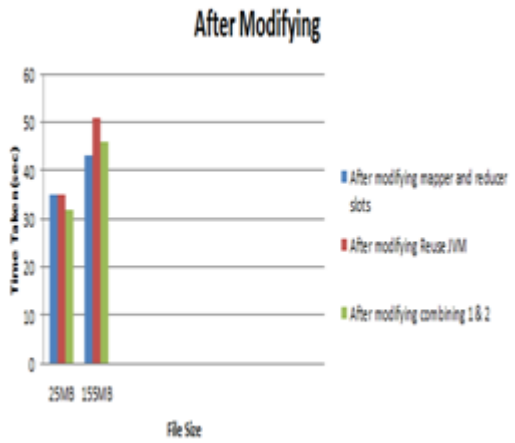

Figure 2: After modifying the parameters comparing the performance

The above graph shows that the combined approach gives a better performance when compared to individual parameters.

5. Conclusion and Future Work

This paper aims at enhancing the performance of the Hadoop tool by manually doing the data placement in the nodes of the cluster and then applying the techniques of increasing the mapper and reducer slots and reusing the available JVM's for every tasks rather than creating a new one for every job run. The performance can also be improved by using the sort buffer and the sort factor. These approaches are compared based on the response time it provides for a word count application that is run. The combined approach gives a better enhancement when compared to the individual parameters. The future work will focus on applying a data prefetching mechanism where the tasks can be run on any node irrespective of where the data is present. This can be done by creating a buffer to save the data for processing of the tasks.

References

[1] Rajashekar M. Arasanal, DaanishU.Rumani, "Improving MapReduce performance through complexity and Performance based data placement in heterogeneous Hadoop clusters".
[2] Tao Gu, Chuang Zuo, Qun Liao, Yulu Yang and Tao li,"Improving MapReduce performance by data prefetching in heterogeneous or shared environments". International journal of grid and distributed computing vol.6, 2013.
[3] Jiong Xie, Shu Yin, XiaojunRuan, Zhiyang Ding, Yun Tian, James Majors, Adam Manzanares and Xiao Qin "Improving MapReduce performance through data placement in heterogeneous Hadoop clusters". Proc.19 heterogeneity computing workshop, Atlanta April, 2010.
[4] Madhavi Vaidya, "Parallel processing of cluster by MapReduce". International journal of distributed and parallel systems, vol.3, January 2012.
[5] B. ThirumalaRao, N.V Sridevi, V.Krishna Reddy, L.S.S Reddy " Performance issues of heterogeneous hadoop Clusters in cloud computing". Global journal of computer science and technology vol XI, May 2011.
[6] Apache, Hadoop, http://hadoop.apache.org/MichaelNoll Multinode Hadoop Cluster Installation.
[7] CBT Nuggetshttp://cbtnuggets.troubleshooting. administrating.optimizinghadoop.com

Author Profile

A Ankita Poovaiah completed the Bachelor's Degree in Information Science and Engineering from Visvesvaraya Technological University (VTU). Currently pursuing M.Tech Degree in Computer Science and Engineering at Mangalore Institute of Technology and Engineering.

Mr. Gopal B received Master Degree in Computer Science and Engineering from NITK. He is currently working as Assistant Professor in the Department of Computer Science and Engineering, Mangalore Institute of Technology and Engineering.