International Journal of Latest Research in Engineering and Technology (IJLRET)
ISSN: 2454-5031
www.ijlret.com || Volume 02 - Issue 09 || September 2016 || PP. 25-29

Analysis of Data placement mechanism for heterogeneous hadoop and mapreduce

Dr. Bharti kalra[1], Dr. D.K. Chauhan[2]
[1]*(Department of CSE, Institute of Technology Gopeshwar, Chamoli, India)*
[2]*(Director Technical, Noida International University, India)*

Abstract: In this paper we proposed the algorithm for data placement with hadoop framework or initially through HDFS to distribute the data set to multiple nodes according to the capability of each node and reducing the common question span for data placement with duplication.

Keywords: Hadoop, Mapreduce, heterogeneous, data placement.

1. INTRODUCTION

In this paper we proposed the algorithm for data placement with hadoop framework or initially through HDFS to distribute the data set to multiple nodes according to the capability of each node and reducing the common question span for data placement with duplication.

1.1 INTRODUCTION TO BIG DATA

Big data can be analysed, collected, processed, analyzed and stored in many ways. Every big data source has so many characteristics, including the frequency, volume, velocity, type, and veracity of the data. When big data is processed and stored, additional dimensions come into play, such as governance, security, and policies. Selecting architecture and creating an appropriate big data solution is challenging because so many factors have to be considered [16].

This "Big data architecture and patterns" series presents a structured and pattern-based approach to simplify the task of defining an overall big data architecture [1].

1.2 INTRODUCTION TO HADOOP

The most significant platform for big data analytics is the open-source distributed data processing platform Hadoop (Apache platform), initially developed for routine functions such as aggregating web search indexes. It belongs to the class NoSQL technologies (others include CouchDB and MongoDB) that have evolved to aggregate data in unique ways [2]. Hadoop has the potential to process extremely large amounts of data by mainly allocating partitioned data sets to numerous servers (nodes), which individually solve different parts of the larger problem and then integrate them back for the final result. It can serve in the twin roles of either as a data organizer or as an analytics tool. Hadoop offers a great deal of potential in enabling enterprises to harness the data that was, until now, difficult to manage and analyze [3].

1.3 INTRODUCTION TO HDFS

The Hadoop Distributed File System (HDFS). This facilitates the underlying storage for the Hadoop cluster. When data for the analytics arrives in the cluster, HDFS breaks it into smaller parts and redistributes the parts among the different servers (nodes) engaged in the cluster. Only a small chunk of the entire data set resides on each server/node, and it is conceivable each chunk is duplicated on other servers/nodes.

1.4 INTRODUCTION TO MAPREDUCE

One of the best known methods for turning raw data into useful information is by what is known as MapReduce [8]. MapReduce is a method for taking a large data set and performing computations on it across multiple computers, in parallel. It serves as a model for how program, and is often used to refer to the actual implementation of this model [4].

In essence, MapReduce consists of two parts. The Map function does sorting and filtering, taking data and placing it inside of categories so that it can be analyzed. The Reduce function provides a summary of this data by combining it all together. While largely credited to research which took place at Google, MapReduce is now a generic term and refers to a general model used by many technologies [12].

2. PROBLEM DEFINITION

In a network, every node contains a disk. Every node is also defined as the process node.

If the data is not available in the processing node, data have to be transferred from the other node via network interconnects. Transferring data from one node to another in excess cause's network congestion and overload and it will also decrease the system performance. In cluster the processing and computing capabilities of each node might be different [5]. We can distinguish these nodes by high speed node and slow speed nodes. When a high speed node finishes the process of its operations, the node support the load sharing by handling unprocessed information or data set on the one or more slow speed nodes. When the amount transferred data by load sharing become excessive, it will be the critical issue for Hadoop's performance. To increase the performance of hadoop in heterogeneous clusters, we have to reduce the transformation of data among the high speed node and slow speed node. This will be achieved by the data placement algorithms that distribute and store data across multiple nodes on the basis of their computing and processing capabilities. Data movement will be reduced if the quantity of file fragments placed on the disk of every node is proportional to the node's processing speed.

For achieving the good I/O performance, one may create the replica of input file in hadoop distribute application in this way that every node in an exceedingly Hadoop cluster contains a copy of the input file. Such an data replication method will minimize data transfer among slow speed and high speed nodes within the cluster when executing the Hadoop application. This data replication scheme having some limitations.

1. It is very costly to make replicas in an exceedingly large-scale cluster.
2. Second, distributing a huge amount of replicas will consume scarce network data measure in Hadoop clusters.
3. Storing replicas require a huge amount of disk capability, which will increase the costing budget of Hadoop clusters.

Although all replicas will be created before the execution of Hadoop applications, significant efforts must be create to decrease the overhead of generating replicas. If the data replication scheme is used in Hadoop, one has to define the issue of high overhead for generating file replicas by implementing a least overhead file replication mechanism. as an example, Shen and Zhu developed a proactive low-overhead file replication theme for structured peer to look networks[12]. Shen and Zhu's theme could also be incorporated to beat this limitation [13].

3. INITIAL ALGORITHM

To define the above shortcoming of the data replication scheme, we directed to data placement methods wherever files are divided and distributed across multiple nodes in an exceedingly Hadoop cluster while not being replicated. This data placement scheme doesn't have any comprehensive theme to modify data replicas.

Placement management mechanism, two algorithms are defined and implemented into Hadoop's HDFS. The initial algorithm is to distribute file fragments to heterogeneous nodes in an exceedingly cluster. While all the fragments of file for an input data required by computing nodes are to be there in a node, these file fragments are divided to the computing nodes. The second data placement algorithm is defined to identify file fragments as to the solution of data skew problems. Two cases are there during which file fragments should be organized. First, new computing nodes will be assigned to an existing cluster to expand the cluster. Second, new data is added to an existing input data. In every case, file fragments distributed by the initial data placement formula will be disrupted.

3.1 Initial data Placement

The initial data placement formula started by initial dividing a huge input file into variety of even sized fragments. Then, the data placement formula assigns fragments to nodes in an exceedingly cluster in accordance to the nodes' processing speed. Compared with low-performance nodes, superior nodes are expected to store and method additional file fragments. Allow us to contemplate a MapReduce application and its input data in an exceedingly heterogeneous Hadoop cluster. In spite of the non-uniformity in node process power, the initial data placement theme must distribute the fragments of the input file in order that all the nodes will complete process their native data at intervals nearly an equivalent time. In our experiments we tend to determined that the computing capability of every node is sort of stable certainly tested Hadoop applications, as a result of the time interval of those Hadoop applications on every node is linearly proportional to computer file size. As such, we are able to quantify every node's process speed in an exceedingly heterogeneous cluster employing a new term known as computing magnitude relation. The computing magnitude relation of a computing node with relation to a

Hadoop application will be calculated by identification the applying. Its price noting that the computing magnitude relation of a node might vary from application to application.

3.2 DATA Dispensation

Input files fragments deliver by the initial data placement method that may be dislocate by some reasons:

(1) Recent data is appended to existing input file;

(2) Data blocks are removed from the regnant input file; and

(3) Recent data computing nodes are added into an existing cluster.

To address this dynamic data load equilibrate, we have to implement an data dispensation formula to rearrange file fragments supported computing ratios.

The data dispensation procedure is defined by the following steps.

First, as initial data placement, information related to the topology and storage space utilization of a cluster is collected by the data distribution server. Second, the server creates two node lists:

1. The list of nodes that define the number of local fragments of every node beyond ites computing capability.
2. A list of node that could manage extra local fragments by their high performance.

The first list is defined as over-utilized defined list; and the second is defined as under-utilized node list.

3. The data dispensation server continuously transfer file fragments from an over-utilized node to under-utilized node till the data load is equally dispensator.

In between of transfer data from an over-utilized mode to an under-utilized node, the server transfer the data fragments from source nodes in over-utilized node list to destination node in underutilized node list.

It has to be noted that server finalize the number of bytes rather than fragments and transfer fragments from source to destination node.

The above data transformation process is repeated till the number of fragments in every node matches is speed calculated by computing ratio.

5.4 The Data distribution formula

1. Get the Network Topology; measure the computing magnitude ratio and utilization
2. Create and sort two lists: under-utilized node list and over-utilized node list
3. Select the supply and destination node from the different lists
4. Migrate data from source node to destination node
5. Repeat step 3, 4 till any list is empty.

5.5 Dynamic Data Placement Policy (DDPP)

In a heterogeneous cluster, the computing capabilities of each node and magnitude relation of nodes are not equal. So, a dynamic data placement Policy (DDPP) Strategy is defined for dispensation of knowledge blocks.

The algorithm needed a table known as computing ratio- defined as to calculate each node’s processing speed in a heterogeneous cluster. There is a procedure to computing ratio, carried out in following steps:

1. A mapreduce applications’ given information processing function are performed separately in every node. It is assumed that every node process the same amount of data.
2. Second, now list the response time of every node performing data processing functions.
3. Minimal response time is carried out as a reference to normalize the response time measurements.
4. Fourth and last, the normalized values, is known as the computing ratio and are used by the data placement algorithm to assign input data for a given mapreduce application.

4. IMPLEMENTATION AND CASE STUDY

Suppose there are three nodes (node x, y, z) in a hadoop cluster. After executing and running an application of mapreduce on each node, collecting the response time of the application on node x, y, z is 20, 30, 40 seconds respectively. The node with the smallest response time has the priority of computing ratio. That means, node x have the shortest response time, so computing ratio of node x in respect of mapreduce application is 1, which becomes the preference to calculating the computing ratio of y, z.

Node	File fragments	Response time	Ratio	Speed
X	30	20	1	Fastest
Y	20	30	2	Average
Z	10	40	3	slowest

Table 1

Table 1 defines the computing ratio and response time for every node in a hadoop cluster. It also showing the total numbers of file fragments dispersed to every node in the cluster. For example the fastest node X have to manage the 30 files fragments whereas slowest node Z having only 10 file fragments to manage.

A least value of computing ratio of a node show that node is a very high speed node. So a high speed node can process a huge number of file fragments.

4.1 Case study

In this section we describe how this algorithm works on the two different mapreduce applications in a heterogeneous hadoop cluster. These two applications are - wordcount and grep. Wordcount is, application used for counting the alphabets, words within a file or data, and whereas, grep, is an application is used to find out normal expression in data. The file or data is applied to the application is around the size of 1GB.

We applied the previously define algorithm to find out the computing ratio of 6 computing nodes for grep and word count applications of mapreduce.

Node number	Wordcount ratio	Grep ratio
1	2	2
2	4	1
3	3	2.5
4	1	2.7
5	4	2.9
6	4	2.9

Table 2

Table representing the computing ratio in terms of heterogeneity of hadoop cluster with grep and wordcount application.

Result given in the table showing that computing ratio is application dependent of hadoop cluster. Node 2 is 2.9 times faster as compared to the node 5 and 6 when defined in terms of grep application. Node 4 is 4 times faster as compared to node 5 and 6 when defined in terms of wordcount application.

The fig 1 shows the response time of grep and word count application and suggest that computing ratio are independent of input file size.

Conclusion

In this paper we design and implemented a data placement algorithm in HDFS for heterogeneous cluster. This mechanism define the performance of hadoop cluster based on computing ratio and try to handle the data redundant problem data allocation in the cluster.

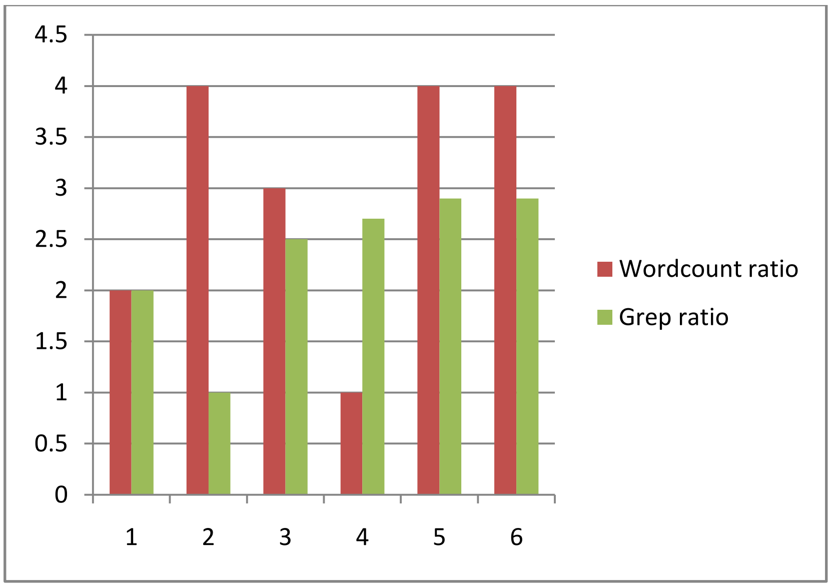

Fig 1

REFERENCES

[1] http://www.ibm.com/developerworks/library/bd-archpatterns1/
[2] https://opensource.com/resources/big-data
[3] http://searchbusinessanalytics.techtarget.com/feature/Comparing-the-leading-big-data-analytics-software-options
[4] Borkar, V.R., Carey, M.J., and C. Li. Big Data Platforms: What's Next? XRDS, 19(1), 44–49, 2012.
[5] Sathi, A. Big Data Analytics. MC Press Online LLC, 2012.
[6] Zikopoulos, P.C., Eaton, C., deRoos, D., Deutsch, T., and G. Lapis. Understanding Big Data—Analytics for Enterprise Class Hadoop and Streaming Data. New York: McGraw-Hill, 2012.
[7] http://www.ittoday.info/ITPerformanceImprovement/Articles/2014-07Raghupathi.html
[8] https://www.ibm.com/software/data/infosphere/hadoop/mapreduce/
[9] http://kasunpanorama.blogspot.in/2013/05/fundamentals-of-map-reduce.html
[10] http://www.bmcsoftware.in/guides/hadoop-introduction.html
[11] T.Chao, H.Zhou, Y.He, and L.Zha. A Dynamic MapReduce Scheduler for Heterogeneous Workloads.IEEE Computer Society, 2009.
[12] M.Isard, M.Budiu, Y.Yu, A.Birrell, and D.Fetterly. Dryad:distributed data-parallel programs from sequential buildingblocks. In EuroSys '07: Proceedings of the 2nd ACMSIGOPS/EuroSys European Conference on Computer Systems2007, pages 59–72. ACM, 2007.
[13] J. Dean and S. Ghemawat. Mapreduce: Simplified dataprocessing on large clusters. OSDI '04, pages 137–150, 2008.
[14] JiongXie, Shu Yin, XiaojunRuan, Zhiyang Ding, Yun Tian,Improving MapReduce Performance through Data Placement inHeterogeneous Hadoop Clusters, Proc. 19th Int'l Heterogeneity in Computing Workshop, Atlanta, Georgia, April 2010.
[15] http://www.oracle.com/technetwork/database/database-technologies/bdc/r-advanalytics-for-hadoop/overview/index.html
[16] https://www.oreilly.com/ideas/what-is-big-data