A replica pre-placement strategy based on correlation analysis in cloud environment

Shaochun Wu ,Xiang Shuai, Liang Chen, Ling Ye, Bowen Yuan
School of Computer Engineering and Science, Shanghai University, Shanghai 200072, P. R. China
scwu@shu.edu.cn, sx700@126.com

Abstract—**Currently, The data placement of cloud storage environment encounters some problems. Based on analyzing the requests of upload files of users, we can dig out the relevance of files. For the purpose of increasing the efficiency of data access, we try to place the relevant files to the nodes that users or programs frequently use in the initial placement. To ensure load balancing, we select storage location in the set of nodes which corresponding its associated files list by using genetic algorithm. Simulations show that by using this pre-placement strategy in replica creation can not only greatly reduce the program execution time, but also relatively ensure a more load balancing of the system.**

Keywords—data placement, Hadoop, File association, genetic algorithm, load balancing

I. Introduction

Replication technology is a most commonly used technique to improve system reliability. In the distributed storage system, it has a wide range of applications. However, the placement strategy of Hadoop is relatively simple at present. It has three replicas in total, one in the local frame, one in the storage node and the last one in another node of the remote rack. And the remaining replicas are randomly placed. That is to say, Hadoop just simply places data block, but it does not consider the characteristics of the data, such as the relationship between the data. In some high performance requiring occasions, including grouping the content of files, aggregating the files and querying connections, the performance is relatively low when relating to a number of relevant documents.

Based on Hadoop platform, this article is researching the resettlement process that data upload to the data center. When the users uploaded a file, we dig out the associated files by analyzing request of file uploading and place the relevant documents to the same storage node. In the resettlement process, we should ensure load balancing. .At the same time we should try to reduce network cost during program execution and data migration at the stage of the follow-up replica's adjustment.

II. Related work

First, In the computer system, request for access of data is not completely random, but driven by the user's service request or the behavior of the program. So in the cloud computing environment, some correlations may exist between data and service, data and data, service and service. When a service or data is accessed, other related services or data may have been accessed. Therefore, the user requests and file usage are closely related with data placement strategy, we can use some strategies to analyze them, and dig out the correlation between files, so that we can carry out more efficient placement.

Yuan Dong proposed a data placement strategy based on the k-means clustering algorithm. The strategy is divided into initial workflow and workflow running phase. In the initialization stage, according to relevancy of files, it will calculate the dependency matrix. Then group the existing datasets by adopting the BEA algorithm. At the stage of running, the system will use the k-means algorithm to allocate generated datasets to the appropriate data centers base on the files dependencies. But this strategy has two shortcomings. Firstly, the number of clustering is given in advance in the k-means clustering algorithm, while in the distributed storage it cannot be determined in advance. Furthermore, by using a recursive traversal method for the transformed dependency matrix, the system will have a low efficiency in case of large number of files. Chen Tao et al. proposed a data placement algorithm CCHDP, which based on the factors of node capacity and bandwidth for clustering device, and use Consistent Hashing-Aware for data storage in every category. But this strategy does not take into account the dependencies among datasets.

In addition, data placement problem itself can be regarded as a bin-packing problem, in which a storage node can be regarded as a box, and the placement data can be considered the goods which would be loaded into the box. Because of high complexity, people usually use heuristic algorithm (likes genetic algorithm) to solve this problem. ZhengPai et al. proposed three stages of data placement strategy, which is used to research the layout plan among data centers with three goals, including the data transmission in data centers, the data dependencies, and a relative load balancing of data centers. The first stage this strategy uses genetic algorithm to generate a solution set with less time cost. During the second stage calculates damage degree of dependencies and selects a set with lower damage degree. At last it selects a solution of balanced load. Although the strategy can effectively reduce the time cost of data movements among data centers, it does not study further the strategy of replica management, and does not put forward an actual strategy how to solve the connection problem between data.

The literature above raised various strategies for the initial data placement, but none of them has considered the relevance of files and centrally stored relevant files. This article will be based on the central storage of relevant files and keep load balancing, and research the initial data placement.

III. Pre-placement process

In certain ranges, the users often use the cloud services. In addition, the location of the user is often fixed in a certain range, so the data accessing are concentrative. In this section, based on the file request record of usage, firstly, we adopt the data mining methods to place the file to the data nodes that services are frequently used, so that to reduce the execution time and the movement of data in the adjustment process. Furthermore, in order to ensure the load balancing, and prevent data gathering to one data node which will result in the barrier of the system, when the system place the file near the cluster center, it will use genetic algorithm to select the optimal node from the cluster center, and then place the replication.

The system frame diagram is shown in figure 1.

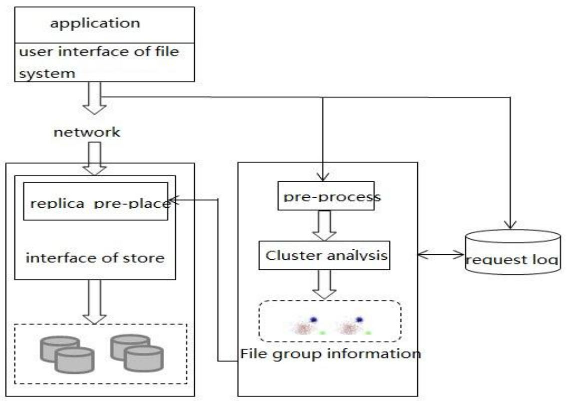

Fig.1. the frame diagram of replica pre-placement

This solution bases on the user's request, tries to place files uniformly to the storage node that services often use. Then it can adjust the layout according to the user's habit in the follow process, to optimize the system. There is the pre-placement process of the system:

- The user will sent the request for creating files to the metadata server when he uploaded files to the cluster, and the request includes the storage path of files, the service, and the IP address of the client and so on.
- The metadata server will give the information of the user's request to the data mining module when it receives the information, and then save the information into the log file.
- The data mining module will process the information of request, and then start the following process: if a suitable clustering is not found, it would adapt the HDFS's default placement strategies for the file's placement. Otherwise, we assume that the most similar cluster which data mining module returns C, the metadata server will search the file's storage node, returns the corresponding node set K of the cluster C, and use genetic algorithm to place the file in the range of the storage node which is included by set K.

A. Clustering analysis

By digging out the relevance between blocks or files, we can try to place the relevant files together, to improve the data concentricity, and make a reasonable layout of files. There are great help to make better layout of storage, and improve program's efficiency during these processes. This paper will base on clustering and analyzing the user's request, place the relevant or similar files together, and make better layout of the storage.

$$Dis\left(\mathrm{A},\mathrm{B}\right)=1-\frac{|A\cap B|}{|\max(\mathrm{A},\mathrm{B})|} \tag{1}$$

1) Distance function

In general, there is little information about the file itself when it is placed to the data center, but we can dig out the file's relevance with the following information:

- Users often save the related files in a particular directory. Therefore, files in the same directory are usually associated.
- Because applications often access the same file in a fixed sequence, there is relevance between files which are accessed by the same application.
- There is relevance between files which are accessed by the same user.
- There is relevance between files which are involved by the access sequence that appear frequently.

This paper will make the information of log file sand user's requests vectored, from which it extracts the user's name, IP address, service's names and catalogues which are in the uploaded files' record, then dig out relevance between files base on Canopy clustering algorithm. In this paper, a file consists of the following attributes: the owner, the appealing IP, Agent, Service ID, path, the request time, and action. And agent is the way to the request, service ID represents service program which process the file, and action represents that the operation is uploading file or reading file. The distance between file A and file B can be shown in the follow-up formula:

$$Dis\left(\mathrm{A},\mathrm{B}\right)=1-\frac{|A\cap B|}{|\max(\mathrm{A},\mathrm{B})|} \tag{2}$$

B. The pre-placement strategy based on the improved genetic algorithm

By clustering the file request, the file is assigned near the most similar cluster center. We should try to ensure the load balancing of each node, and avoid the excessive usage of individual nodes. This is actually a bin packing problem, that is to say, the node is regarded as the box, and the data which is being placed is regarded as items. In this paper, we adapt improved genetic algorithm to process it.

1) Chromosome coding

We assume that file is divided into S blocks according to the data block of the default size to 64M, the number of the replica of each one is R (R defaults to three), and the size of the population can be considered as S, each chromosome in the population can be seen as a copy of a segmented data block, and the size of it is R. Gene in the chromosome can be expressed as <BlockID,DataNodeID>, wherein, BlockID is the ID of data blocks, DataNodeID is the ID of the storage server which the data block would place to. In a chromosome, BlockID is fixed, only the DataNodeID is changing. Different chromosome's BlockID constitutes a file. For example, we can assume that the file F include three data block of Block1, Block2 and Block3, the replica factor is three, and like << Block1,DataNode1>,<Block1,DataNode2>,<Block1,Data Node3>> are expressed as the data block which number is 1,and it would place to the node which number is DataNode1,DataNode2 or DataNode3.

2) *The guarantee of solution validity*

In order to prevent the invalid placement solution generating, we should judge the effectiveness of the solution when the system generates the initial population or the chromosome is crossing and mutating. Specifically, the system needs to check the gene fragment of each chromosome in each generation, and then count the data block which will be placed to the specified node, to judge whether the remaining space is enough. If the space is not enough, the solution is invalid. It needs to re-select the node.

3) *The generation of initial population*

The metadata server will check the file in the file list which is returned by the Canopy cluster algorithm, and then returns the storage node of the file. After processing all of the files, we will get the corresponding storage node set of the file list, as the node list which is being selected by the population which is generated by the genetic algorithm. For each chromosome in the initial population, we can assume that the number of the replication is R, the system will randomly select one node in the node list above, as the first replication's storage node which each data block will be placed to. Similarly, the system will choose other nodes as the first R replication's storage node that each data block will be placed to.

4) *Fitness Function*

This paper argues that the difference on the nodes utilization rate is smaller, the usage of the node is better. We can assume that DataNode(i)(i=1,2,..,R) is the storage node of a chromosome, Capacity(i) (i=1,2,..,R) is the total capacity of the storage node, Size(i) (i=1,2,..,R) is the used storage space of the node, Avg is the average of the utilization rate of the storage nodes which each chromosome of population is placed to ,and the fitness of a chromosome(f) can represent the variance of the storage utilization.

$$f=E[Size(i)/ Capacity(i) - Avg]2 \quad (i=1,2,..,R) \qquad (3)$$

5) *Crossover*

We can select same location from two chromosomes as the cross point randomly, and exchange the gene after the cross point so that offspring can inherit good genes from parent. We assumed that the chromosomes data block number is 1, its chromosomes number is<<1,DataNode1>,<1, DataNode2>,<1, DataNode3>>, and the second is <<2, DataNode1>,<2, DataNode3>,<2, DataNode5>>, and the cross point is between second gene and third gene, the offspring that generated by crossing is

<<1,DataNode1>,<1, DataNode2>,<1, DataNode5>>,
<<2, DataNode1>,<2, DataNode3>,<2, DataNode3>>

6) *Mutation*

In this paper, the mutation operation is base on knowledge. In the chromosome, the data block number is fixed, only the storage node number can be changed in the corresponding storage node set of the file list that include by the cluster.

IV. Simulation

A. *Simulation environment*

This simulation is running at 6 PCs. Each node is composed of Intel (R) Core2 dual-core processor, 4G memory and a 160G hard disk, and tested in the LAN environment. And the node will install Ubuntu Linux operating system (version Ubuntu10.10),Hadoop 0.20.203,jdk1.6.0_27. Hadoop's NameNode and JobTracker are running at the same node, and DataNode and TaskTracker are running at the other node at the same time. To achieve better effect of the test, we have made changes to the configuration of Hadoop platform for the simulation, set the maximum value of the JVM heap to 1Gand support JVM reuse, set the size of the reorder buffer to 512M, and shut down the continuous executive of map and reduce. We also have simulated that user use the system to process the text. In order to verify the change of the response time, we would consider the situation of text processing, including some similar files, and each file's record would use the format of table UserVisits in the literature[7], and each item of the record would be separated by '\t' in the file. We assume that there are 100 user using this system, each user uploaded 20 files, and 10 of them is less than or equal to 64M, 6 of them is about 128M, 4 of them is about 320M. That is to say, the file fits into the data block. The user saves the uploaded files to the corresponding directory of his ID, the owner of the file is the user, every user's IP is different, and Agent is the browser. The contents of the file randomly generated the corresponding data set according the requirement in the table UserVisits.

B. *The analysis of simulation results*

To compare the pre-placement strategy in this paper and default placement strategy, the simulation recorded effect of the execution time and load balancing with the Hadoop default placement strategy, and redeploy and execute the placement strategy after it add to the Hadoop source code. We have done several test and taken the average of the final result.

1) Comparative analysis of the pre-placement's response time

To analyze the changes of system response time, the simulation will select 10 users from 100usersrandomlyafter all users have uploaded files, and do aggregate operation on every user's files, that is to say, the simulation will group the record by its IP, count its total amount of adRevene in the past time, and sort it by time.

The result is shown in Figure 2.

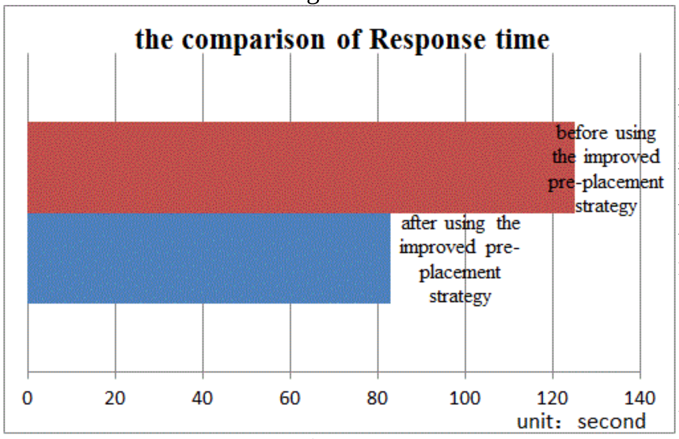

Fig.2. the comparison of Response time

Form the char, we adapt the default placement strategy which execution time is 125s, and then make a simple clustering to place the relevant files together. It can reduce the execution time greatly and improve 26%efficiency.

2) Comparative analysis of the pre-placement's load balancing

To analyze the impact of the system load balancing, system will count the usage of the six nodes, and check the used space of each disk. The simulation will count the disk usage of each node after using the pre-placement strategy, and compare with the initial HDFS.

The result is shown in Figure3.

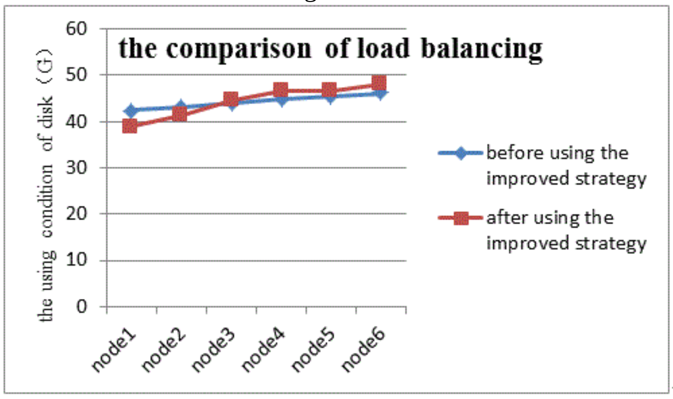

Fig.3. the comparison of load balancing

From the chart, the horizontal axis stands for the six storage nodes in the cluster, the vertical axis stands for the disk usage of each node. The simulation will sort the used space of each node by size in ascending. From the chart we can see that used space variance of each node is about 1.5% before improving. After using the strategy in this paper, the system places files that users often use to the same node, and the variance is enlarged about 3.5%, but no data set is stored centrally in certain nodes.

V. Conclusions

At the present, research into the replica placement is still in its infancy, the strategy is relatively simple. This paper analyze user's request in the file placement process, use the genetic algorithm to assign the file which will be placed near the most similar cluster center, and keeps load balancing. The test results shows that the system response time is improved 26% after using the placement strategy that we propose, and the utilization of each node is relatively balanced.

Acknowledgment

We thank all of the anonymous reviewers for their insightful suggestions and comments that will significantly improve the quality of our manuscript. And this work was financially supported by the Ocean Public Welfare Project of The Ministry of Science and Technology under Grant No.201105033, the National Natural Science Foundation of China (No.40976108).

References

[1] D.Borthakur,"Hdfs architecture," Hadoop 0.19 Documentation. [Online]. Available: http://hadoop.apache.org/core/docs/current/hdfs design.html 2008

[2] Yuan Dong A data placement strategy in scientific cloud workflows Future Generation Computer Systems, Vol. 26, no. 8, pp. 1200-1214 Oct 2010

[3] Tao Chen,Nong Xiao,Fang Liu,Changsheng FuThe data layout algorithm based on clustering and consistent Hash Journal of software, Vol.21, No.12, pp.3175?3185 December 2010

[4] Pai Zheng, Lizhen Cui, Haiyang Wang etc. Cloud computing environment for data intensive applications data layout strategy and method [J]. Journal of computer: 1472-1480 2010.8

[5] Andrew McCallum, Kamal Nigam, Lyle H. Ungar Efficient Clustering of High Dimensional Data Sets with Application to Reference Matching Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining KDD 00 .ACM Press, Pages: 169-178 2000

[6] Wu Shaochun,Yuan Bowen. An adaptive Simulated Annealing Genetic Algorithm for the data Placement Problem in SAAS. ICNECS2011 International Conference

[7] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J.DeWitt, S. Madden, and M. Stonebraker. A comparison of approaches to large-scale data analysis In SIGMOD, pages 165-178. ACM, 2009