Dynamic Data Placement Strategy in MapReduce-styled Data Processing Platform

Hua-Ci WANG1,a,*  Cai CHEN2,b,*  Yi LIANG3,c

1,2,3 Beijing University of Technology, China

awanghc@emails.bjut.edu.cn, bchencai@bjut.edu.cn, cqliang@bjut.edu.cn

*Corresponding author

Keywords: Data Placement, MapReduce, HDFS, Replica, CloudSim.

Abstract. Data placement is one of the core technologies of the MapReduce-styled data processing platform, which contributes much to the data processing efficiency. A dynamic data placement strategy is proposed in this paper due to that existing data placement techniques are lack of the consideration of the computing load on the data storage node and reduce the ratio of the localized processing of the hot-spot data. The strategy takes the data accessing localization ratio, the remaining computing capacity of nodes as the new factors of the data placement decision. Performance evaluation results show that the proposed data placement strategy outperforms the original strategy in HDFS file system and the average job execution time is reduce by the maximum of 12%.

Introduction

MapReduce-styled data processing platform (“MapReduce platform” for short) is one of the latest technologies in the field of massive data processing [1]. Data locality-aware processing on the MapReduce-styled platform means the massive volume of data is stored on the local disk on the computing nodes and the computing tasks are scheduled to the nodes that the consuming data resides as much as possible, so as to reduce the communication overhead caused by remote data access and improve the data processing efficiency. Consequently, increasing the possibility of local data processing is one of the main objectives of the MapReduce platform.

Data placement is one of the core technologies of the data processing platform, which implement the function that the data in the platform can be distribute stored in all storage nodes via a reasonable and effective way. Existing data placement techniques mostly aim at enhancing the efficiency of data access and reducing the data I/O bottleneck. However, when applied to the MapReduce platform, these technologies will lead to the poor data processing efficiency, due to that these technologies are lack of the consideration of the computing load on the data storage node and reduce the ratio of the localized processing of the hot-spot data.

Aiming on the problem mentioned above, a dynamic data placement strategy is proposed in this paper. The strategy takes the data accessing localization ratio, the remaining computing capacity of nodes as the new factors of the data placement decision. The strategy has two parts: one part is evaluate nodes and data block replica with decision-making factors, another part is build migration task with the migration data block replica candidate set and the destination node candidate set, and is implement with a Master/Slave framework. Performance evaluation results show that the proposed data placement strategy outperforms the original strategy in HDFS file system (the most popular data management system in MapReduce platform) and the average job execution time is reduce by the maximum of 12%.

Background & Related Work

Hadoop distributed file system (HDFS) is one of the most widely used file systems in MapReduce platform [2]. The research of dynamic data placement is based on HDFS in this paper.
HDFS is designed as a master-slave architecture with a single master. A typical HDFS file system cluster is composed of a Namenode and several Datanodes. Files stored in HDFS are consist of several blocks of data. For the fault-tolerance and concurrency, data blocks are separately stored in different nodes in a multi-replica way. In the default placement, there are three replicas. One major replica is stored in the local node or in the local rack where the client operates, and two default replicas are stored in a remote rack's two nodes with random way.

Data placement strategy is the key to the reliability and performance of file system in the MapReduce platform, and optimized replica placement strategy will effectively improve the performance of the MapReduce platform. A greedy data placement strategy proposed by [3] is to store the data according to the storage capacity and data accessing frequency, and the strategy proposed by [4] is to store the data according to the calculation and processing ability of the nodes.

Cost-Effective Dynamic Replication Management Scheme (CDRM) is proposed in the paper [5], which considers the availability of data storage nodes and load balancing. According to the node's load situation and the degree of node availability, the strategy to adjust the number of replicas and distribution can be adjusted.

Existing data placement techniques mostly aim at enhancing the efficiency of data access and reducing the data I/O bottleneck. In this paper, by adjusting the location of the data block replica to improve the probability of the data to be localized access as a means to reduce the MapReduce job execution time for the design and implementation of dynamic data placement strategy.

**Overview of Dynamic Data Placement**

Fig. 1 shows the dynamic data placement implement with a Master/Slave framework. The light-colored components have existed in HDFS, and the deep-colored components are new-added in HDFS.

The function of Slave nodes is to collect, compute and predict the decision-making information. During the processing, the role of information collection is to get the data block replica access information from HDFS client, and the CPU and RAM information from NodeManager. The collected information can be defined as 3-tuple: (replica-access, CPU, RAM); then compute the collected information; and predict the statistical information by using gray prediction.

The components in master node includes decision-making information acquisition, that is to gather all decision-making information from slave nodes, data block replica evaluation which is to calculate the migration value of data block replica, node evaluation which is to calculate the migration value of nodes, destination node selection which is to select suitable nodes into migration destination node candidate set, data block replica selection which is to select suitable data block replicas into migration data block replica candidate set, and migration task building is to choose the best destination node from migration destination node candidate set for each data block replica which is in the migration data block replica candidate set.
Dynamic Data Placement Strategy

The basic idea of the proposed dynamic data placement strategy is to adjust placement of data block replica to improve the localized processing rate of hot-spot. Dynamic data placement strategy is composed of the definition as follows and the data block replica migration algorithm.

Dynamic data placement strategy is based on some new decision factors which can be collected as the decision-making information from the slave nodes, so we first defined the decision-making factors.

Definition 1 The accessing frequency of data block replica: The accessing frequency is the times of data block replica accessing in a time period which can be expressed as:

\[ R = \sum_{r_i} \text{value} | \text{value} = \begin{cases} 1, & r_i, \text{time} \notin [t, t + T] \\ 0, & r_i, \text{time} \in [t, t + T] \end{cases} \]

where, \( r_i \) is an event of data block replica accessing; \( r_i, \text{value} \) is the assignment value to data block replica on event \( r_i \); \( r_i, \text{time} \) is happen time of event \( r_i \); \( t \) is the beginning of the time period; \( T \) is the interval of the time period.

Definition 2 The ratio of localized accessing of data block replica: The ratio of localized accessing of data block replica \( L \) can be express as: \( L = \frac{R_{\text{local-access}}}{R_{\text{total-access}}} \); where, \( R_{\text{local-access}} \) is the times of data block replica localized accessing in a time period; \( R_{\text{total-access}} \) is the all times of data block replica accessing in a time period.

Definition 3 The node remaining computing capacity: The node remaining computing capacity is the average of remaining numbers of CPU core, the average capacity of RAM in a time period and can be expressed as:

\[ \bar{C} = \frac{1}{n} \sum_{i=1}^{n} e_i, \text{cvalue}, \quad \bar{M} = \frac{1}{n} \sum_{i=1}^{n} e_i, \text{mvalue} | e_i, \text{time} \in [t, t + T] \]

where, \( e_i \) is an collection event for collecting remaining computing capacity; \( e_i, \text{cvalue} \) is the numbers of CPU core that is got by event; \( e_i, \text{mvalue} \) is the capacity of RAM that is got by event; \( t \) and \( T \) is the same thing which is described in Definition 1.

Evaluation of the Migration Value of the Data Block Replica and the Node

The migration value of node can be expressed as:

\[ v' = w_1 \times v^c + w_2 \times v^m | w_1 + w_2 = 1; \]

where, \( v^c \) is the normalized value of \( \bar{C} \); \( v^m \) is the normalized value of \( \bar{M} \); \( w_1 \) is the weight of \( v^c \) while evaluating migration value of node; \( w_2 \) is the weight of \( v^m \) while evaluating migration value of nodes;

The migration value of replica can be expressed as:

\[ v'' = w_1 \times v^b + w_2 \times v^c + w_3 \times v^m | w_1 + w_2 + w_3 = 1; \]

where, \( v^c \) and \( v^m \) are same as \( v^c \) and \( v^m \) which are described above, besides, \( v^b \) and \( v^m \) here is based on the node that stores current data block replica; \( w_1, w_2 \) and \( w_3 \) are the weights of \( v^b \), \( v^c \) and \( v^m \) while evaluating migration value of replica; \( v^b \) is the normalized value of the accessing value of data block replica which can be defined as: \( B = L + k/R \); where, \( L \) is the ratio of localized accessing in the current data block replica; \( R \) is the accessing frequency of current data block replica; \( k \) is the weight the the access frequency of data block replica while calculate \( B \).

The Data Block Replica Migration Algorithm

The data block replica migration algorithm mainly includes three step. The first step is to select appropriate data block replica as the migration data block replica candidate set. For each of replicas has a migration value when evaluate processing has be done, we will sort the replicas by their migration value ascendingly and put the replica that its migration value is lower than threshold into migration data block replica candidate set, and the size of above set should be controlled at most MAX which is configured by administrator. The second step is to select suitable migration destination node candidate set. For each of nodes has a migration value after evaluate processing has be done, we put the node that its migration value is higher than threshold into migration destination node candidate set. The third step is to build migration for all replicas in migration data block replica candidate set, and it can be as follows:
In the dynamic data placement strategy, for each replica in migration data block replica candidate set, a migration task should be built (Line 3). The data block replica's access history contains remote access record as $\text{Set<Datanode, count>}$ . There are two situations of data block replica's remote access history, one situation shows that most remote access are request by a small part of remote nodes. In this situation, data block replica should be migrated to those nodes as much as possible (Line 6). The another situation shows that remote access are request by remote nodes evenly. In this situation, choose a random node from candidate node set as the destination node (Line 7). Besides, each migration should make sure that replicas of one data block should place on different nodes and different racks (Line 9~10).

Performance Analysis

Extension of CloudSim

CloudSim[6] is a simulation software. Because of the natural flexibility, it has been popularized as time goes on. This paper simulates the distributed file system based on CloudSim 3.0.3 project by adds HDFSFile class, Block class, BlockReplica class, Namenode class and Datanode class etc. This paper further realizes the simulation of MapReduce Platform based on [7], adds HDFSFile dependence for Job Object, add Block dependence for Map Task etc.

Simulation Experiment and Results Analysis

Based on extended CloudSim simulation Platform, this paper does experiments on dynamic data placement strategy in this paper and then compares our simulation results with simulation results of HDFS default data placement strategy.

Extended CloudSim project's configuration files includes cluster configuration files, job configuration files and data configuration files. According to cluster configuration files, we distribute 144 different performance hosts on 10 racks and set data transmission speed between different racks to 1MB/S, transmission speed between different nodes in same rack to 10MB/S and
disk read speed to 133MB/S. This paper examines the performance of optimizing strategy by using two evaluation indices job average execution time and local Map task ratio.

Table 1. Result Comparison of the Proposed Data Placement Strategy to the Original Strategy of HDFS.

<table>
<thead>
<tr>
<th>Job submissions in per hour</th>
<th>Default data placement strategy</th>
<th>Dynamic data placement strategy added</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Job average execution time [s]</td>
<td>Local Map Task ratio [%]</td>
</tr>
<tr>
<td>640</td>
<td>96.50875</td>
<td>90.5982188</td>
</tr>
<tr>
<td>780</td>
<td>97.82455128</td>
<td>84.5551336</td>
</tr>
<tr>
<td>920</td>
<td>100.539163</td>
<td>76.6660861</td>
</tr>
<tr>
<td>1060</td>
<td>106.6621415</td>
<td>68.8642076</td>
</tr>
<tr>
<td>1200</td>
<td>111.170225</td>
<td>60.7274257</td>
</tr>
</tbody>
</table>

Figure 2. Performance Comparison of the Proposed Data Placement Strategy to the Original Strategy of HDFS.

Jobs in MapReduce Platform are submitted by users to the platform to run during running process. Jobs submission is a random event, and it has fixed frequency. It means that the time interval of two successive events obey exponential distribution. To preferably evaluate the performance of our strategy, this paper compares the HDFS default data placement simulation results with the dynamic data placement strategy extension under different amount of job submissions in per hour. The amount of jobs in the experiment are separately 640, 780, 920, 1060 and 1200. The experiment uses custom programs to create corresponding configuration files of different amount of jobs which satisfy Poisson distribution. Running results of jobs is shown in Table 1 and the figure of running results of jobs is shown in Fig. 2.

The results show in Fig. 2 and Table 1 demonstrates that under condition of same amount of jobs submission, MapReduce platform which adds dynamic data placement strategy has better performance than platform without dynamic data placement in job average execution time and local Map task ratio. And the platform has best optimization affect when the amount of jobs submission per hour is 1060. The application of dynamic data placement strategy decreases the execute time by 12% at best and 7.88% in average.
Summary
This paper proposes a dynamic data placement strategy, which increases the local execution ratio of the data processing task and reduces the job average execution time in MapReduce platform. It overall improves the performance of MapReduce platform. In the future, we will further consider to add network condition in dynamic placement.

References