An optimized Speculative Execution Strategy Based on Local Data Prediction in Heterogeneous Hadoop Environment

Dan-Dan Jin[1], Qi Liu[2*], Xiao-Dong Liu[3]and Nigel Linge[4]

[1] Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET),
Nanjing University of Information Science & Technology,
Nanjing, Jiangsu, China
nuist_jdd@126.com

[2] School of Computer & Software, Nanjing University of Information Science and Technology,
Nanjing, Jiangsu, China
nuist_jin@163.com

[3] School of Computing, Edinburgh Napier University,
10 Colin Road, Edinburgh EH10 5DT, UK
x.liu@napier.ac.uk

[4] The university of Salford, Salford,
Greater Manchester, M5 4WT, UK
n.linge@salford.ac.uk

Received 27 August 2017; Revised 10 October 2017 ; Accepted 10 October 2017

Abstract. Hadoop is a famous parallel computing framework that is applied to process large-scale data, but there exists such a task in hadoop framework, which is called "Straggling task" and has a serious impact on Hadoop. Speculative execution (SE) is an effective way to deal with the "Straggling task" by monitoring the real-time rate of running tasks and back up the "Straggler" on another node to increase the opportunity of completing backup task ahead of original. There are many problems in the proposed SE strategies, such as "Straggling task" misjudgment, improper selection of backup nodes, which will result in inefficient implementation of SE. In this paper, we propose an optimized SE strategy based on local data prediction, it collects task execution information in real time and uses Local regression to predict remaining time of the current task, and selects the appropriate backup task node according to the actual requirements, at the same time, it uses the consumption and benefit model to maximizes the effectiveness of SE. Finally, the strategy is implemented in Hadoop-2.6.0, the experiment proves that the optimized strategy not only enhances the accuracy of selecting the "Straggler" task candidates, but also shows better performance in heterogeneous Hadoop environment.

Keywords: Hadoop, Speculative execution, Straggling task, Local Regression, Prediction accuracy

1 Introduction

In recent years, Cloud computing has served as a new business computing model, which has become the research hotpot of the academic and IT business giants [1]. Many famous IT companies like Google, Microsoft, Amazon, Yahoo! have launched their own cloud computing service platform such as Apache Storm [2], Spark [3], Dryad [4], and Hadoop [6], and let the development of cloud computing technology as one of the important strategic road of the future [5]. And as one of the current popular cloud computing frameworks, hadoop platform is also widely used and has become the top project of Apache and been adopted by major internet companies to do some related customization [6], which is originally a part of the open-source search engine "Nutch", and implemented by Doug Cutting by referring to Google's distributed large data storage model "GFS" and the distributed parallel computing model "MapReduce" [7-8]. With the continuous improvement and development of Hadoop platform, many applications based on HDFS and MapReduce are becoming more and more abundant,

* Corresponding Author

such as Hive [9], HBase [10], which are designed to improve the performance of the cluster and let people storage and process data more easily, but these applications are based on the underlying Hadoop distributed storage framework "HDFS" [11] and computing framework "MapReduce" [8], which are the core components of Hadoop.

HDFS, which is called Hadoop Distributed File System, is a scalable distributed file system for large-scale data-intensive applications. Although it runs on a low-cost machine, its high fault tolerance makes it possible to provide high-performance services to its customers, HDFS weakens the file system POSIX (Portable Operating System Interface) requirements to improve the efficiency of data access, which uses a "write once, read many times" access mode, the user can only write in the end of the file, this mode can easily maintain data consistency and enhance the throughput of data access [11].

MapReduce is a new abstract model designed by Google, which allows application developers to represent the simple computations they will perform while hiding those messy details in a library, such as parallelization, fault tolerance, data distribution, load balancing, etc [8]. Google's abstract model was inspired by Lisp and the original representation of Map and Reduce of many other functional languages. Google recognizes that many of their calculations include the ability to apply a Map operation on the logical records of the input data to compute an intermediate key/value pair and apply the Reduce operation on all values with the same key to merge derived data appropriately. The use of functional models, combined with user-specified Map and Reduce operations, allows users to easily implement massively parallelized computations and use re-implementation as a primary mechanism to achieve fault tolerance [12].

In the Hadoop platform, each job will be divided into multiple tasks, which is called job scheduling, and all of these tasks are assigned by the JobTracker to each TaskTracker. How to coordinate the resource allocation of each TaskTracker has become the main function of JobTracker [13]. However, JobTracker cannot obtain all the information of TaskTracker in real time and it is difficult to predict the future running state of TaskTracker, it can only satisfy the task distribution time as fast as possible and cannot guarantee that the allocation in the subsequent execution still maintain its superiority [14], which will lead to the so-called slow tasks "Straggler". Table 1 summarizes the internal and external factors that leads to "Stragglers", it can be seen from the table that there are many factors that cause stragglers and these factors cannot be accurately accounted for and measured, which makes it impossible to pinpoint whether the task is a "Straggler". Speculative Execution (SE) is the current effective method to correct the wrong decision made by the scheduler due to the unpredictable problems in this kinds of nodes or tasks and optimizes the fault tolerance mechanism of hadoop [8]. The main function of speculation execution (SE) is to find out the straggler and back up to another node to perform, and hope that the backup task can complete faster than the original task, thereby reducing the execution time of the job and increasing cluster throughput.

Table.1 Factors leading to "Straggler"

Internal factors of Cluster	External Factors
Node resource heterogeneity	External application resources competition
Resource competition between different tasks	Input data skew
Network performance	Faulty node hardware

Hadoop originally implemented the naïve speculative execution called "Hadoop-Naive", but its performance is poor in heterogeneous environments due to it uses task progress to determine whether the task is a "Straggler", so many researchers began to optimize the SE from all aspects and several optimized strategies are listed and compared in Ref. [15], Such as LATE, MCP, ERUL and so on. These Strategies judge the slow tasks mainly based on the self-estimation of the task remaining time, but inaccurate estimation will cause inappropriate allocation, so the accuracy of estimating the remaining time of running tasks is necessary to be improved. Based on the research on speculative execution, we propose a new speculative execution called "LSE", which aims to predict the remaining time of the current task more accurately, and ensure that the benefits of the cluster is maximized while starting the backup task.

1.1 Our Contributions

In this paper, we pay attention to the data of the real-time task execution and collect the task information during task running, then apply the local regression algorithm, which is better than linear regression, to predict the remaining time of task more accurately than the general linear regression algorithm, finally, we ensure that the backup task will bring maximum benefits to the cluster. The main contributions of our paper are listed as follows.

[1] we collect task execution information during task execution and find that there exists a certain linear relationship between task execution time and progress, which is not implemented in the currently existing speculative execution strategy, then analyze and exact data features in order to predict the following data tendency;

[2] Based on the data collection of task execution information, then we implements the local regression, we can predict the remaining time of running tasks more accurately because the local regression doesn't use all the data to do prediction but use local data, which is an optimized Non-parametric prediction method;

[3] We calculate the benefits and cost of enabling and disabling backup tasks to ensure the benefits of speculative execution is maximum;

1.2 Organization

The rest of this paper is arranged as follows. Section 2 lists some related works on the current optimization strategies for Hadoop fault-tolerant mechanism. Our design of "LSE" is presented in Section 3. Section 4 compares our strategy with some current algorithms by experiments, finally, we will conclude the whole paper and prospect for future work in Section 5.

2 Related Work

In recent years, Hadoop is widely used in the Internet industry, the optimization of fault-tolerant mechanism of Hadoop gained wide attention. The fault tolerance of Hadoop consists of storage fault-tolerance and computation fault-tolerance. The main purposes of storage fault-tolerance include improving the reliability and high speed of data access, automatic fast recovery of faults and minimizing the loss, while the computation fault-tolerance aims to automatically detect and restore fault tasks in order to improve the system's computing performance and reliability. Storage fault tolerance is primarily targeted at HDFS optimization while computation fault-tolerance includes Hadoop job scheduling strategy, task scheduling strategy improvement, enhancing speculation execution strategy etc. We will introduce these work in detail.

2.1 Storage Fault-tolerance

HDFS uses the master-slave mode for data storage and management, and let master node to manage the storage node and store data block-related metadata from multiple slave nodes to store each data block in order to facilitate the management of large data block storage. Once the master node fails, the whole cluster will be paralyzed, so it is necessary to improve the fault tolerance of the master node. In order to solve this problem, NameNode Federation technology was proposed, it achieve NameNode horizontal extension by setting a number of independent NameNode, these NameNode constitute an alliance between them and do not coordinate with each other [16]. Due to the low efficiency of load balancing in HDFS Federation, a distributed storage system based on the extensible metadata service of HDFS called "Clover" was put forward [17], which provides persistence metadata storage by way of establishing "Share Storage Pool" (SSP) above the NameNode. Facebook raised the "AvatarNode" method to avoid the single-point failure of NameNode. Apart from this, the load balance of data in HDFS is another research hotspot in Hadoop, existing load balance strategies mainly consider the disk usage of the storage node to judge whether the whole is balanced [6]. Cost-effective Dynamic Replication Management (CDRM), which is considered as a cost-effective method, is used to adjust the distribution of data blocks by cluster load variation combined with node CPU processing capability, memory, network bandwidth and disk storage performance [18]. Shafer, Rixner, et al. designed an adaptive feedback load balancing model based on HDFS, it integrates disk utilization and service capability of node to describe the load of balance, which is more accurate for systems with frequent data reading and writing [19].

2.2 Computation Fault-tolerance

Some basic MapReduce framework computation fault-tolerant mechanisms are mentioned in [5], which includes periodic heartbeat mechanism to determine the communication flow, basic task re-executed rules to deal with task or work node failure, and speculative execution which detects slow tasks early and select appropriate nodes to perform backup tasks. Although these mechanisms are implemented in Hadoop, some of them do not perform well and require further refinement. In this paper, we focus on the optimization of speculative execution strategies, and then we will introduce some of the research results in recent years.

Since Hadoop's Naïve speculative execution strategy is based on an isomorphic cluster, a task is determined as a "Straggler" when its progress is below average, which will exist misjudged tasks and increase the cluster resource consumption. In order to improve the performance of Hadoop-Naive in heterogeneous environments, Zaharia, et al. put forward the LATE strategy which uses the remaining time as the speculative execution priority [20]. LATE use the average rate to calculate the remaining time of running tasks, which will lead to error, H. Wu et al. find out the linear relationship between system load and the remaining time of the task and propose the "ERUL" strategy, which calculates the remaining time by the real-time system load and improves the accuracy of the prediction [21]. Since the previous strategy did not take into account the cluster efficiency issues, MCP was proposed to enable the backup task to maximize the benefits of the cluster, which divided Map tasks into map and combiner stages while reduce tasks are divided into copy, sort and reduce phases, it calculates the benefits of launching backup tasks and let the benefits to be maximum [22]. Even though MCP takes the efficiency into consideration, it still ignores the value of nodes, Ex-MCP was raised by taking the value of the node into consideration comparing with MCP [23]. In addition, some other optimization strategies are proposed. A smart speculative execution was brought forward based on node classification according to the hardware performance of the node [24]. Wang and Lu et al. presented an improved strategy called "Partial Speculative Execution (PSE)"to enhance the efficiency of speculative execution, but it ignored the difference between nodes′ processors [25]. A new speculative execution strategy based on C4.5 decision tree calculates the completion time of the task based on C4.5 decision tree · which is more efficient and feasible [26]. The Adaptive Task Allocation Scheduler (ATAS) was provided to make improvements on increasing the Hadoop's ability to respond by adapting a more accurate way to estimate the time of backing up tasks [27].

In summary, many companied like Yahoo!, Facebook et al. disabled their own speculative execution strategy due to the misjudgment of slow tasks [20], which will increase the cluster execution time and waster the cluster resources. Therefore, there are still many challenges existing in optimization of speculative execution strategy, one of the most urgent is to how to find the real "Straggler" task and back up it on the fast node at the right time. The following paper is to present our optimized speculative execution strategy from these three points.

3 Model and Algorithm

3.1 Overview of Hadoop Naïve Speculative Execution Strategy

In Hadoop-Naïve, the primary method to determine whether a task is worth starting a backup task is to first assume that it starts a backup task · and then estimate the execution time of the backup task as $estimatedEndTime_2$. similarly, according to the computing speed of running task, we can estimate the most likely completion time $estimatedEndTime_1$, so the difference between the $estimatedEndTime_1$ and $estimatedEndTime_2$ is larger, which indicates that the value of launching backup task is greater and hadoop tends to start the backup task for such tasks, detailed calculations are as follows.

$$estimatedEndTime_1 = estimatedRunTime + taskAttempStartTime \tag{1}$$

$$estimatedRuntime = (currentTimestamp - taskAttemptStartTime) / progress \tag{2}$$

$$estimatedEndTime_2 = currentTimestamp + averageRunTime \tag{3}$$

Where *currenTimestamp* is the current time, *taskAttemptStartTime* is the task start time and *averageRunTime* represents the average running time of the successfully completed tasks. At the same time we can see that the biggest drawback in the Hadoop-Naïve is that using average progress rate to predict the remaining time will result in errors and misjudgments of slow tasks.

3.2 Our Design

Due to the misjudgments of slow tasks caused by inaccurate task remaining time prediction and the incorrect choice of backup task node, the performance of SE in Hadoop is still low. In order to better predict the performance of the running tasks to bring the maximum benefit to the cluster while launching the backup tasks, we try to collect the real-time tasks' execution information, and find that there is a certain linear relationship between task execution time and progress, then we use a new prediction method to estimate the remaining execution time of the real-time tasks and detect "Stragglers" candidates in time , and combine the cosumption and benefits model to decide whether to start backup tasks. The flow chart of the whole method is designed as shown in Figure 1.

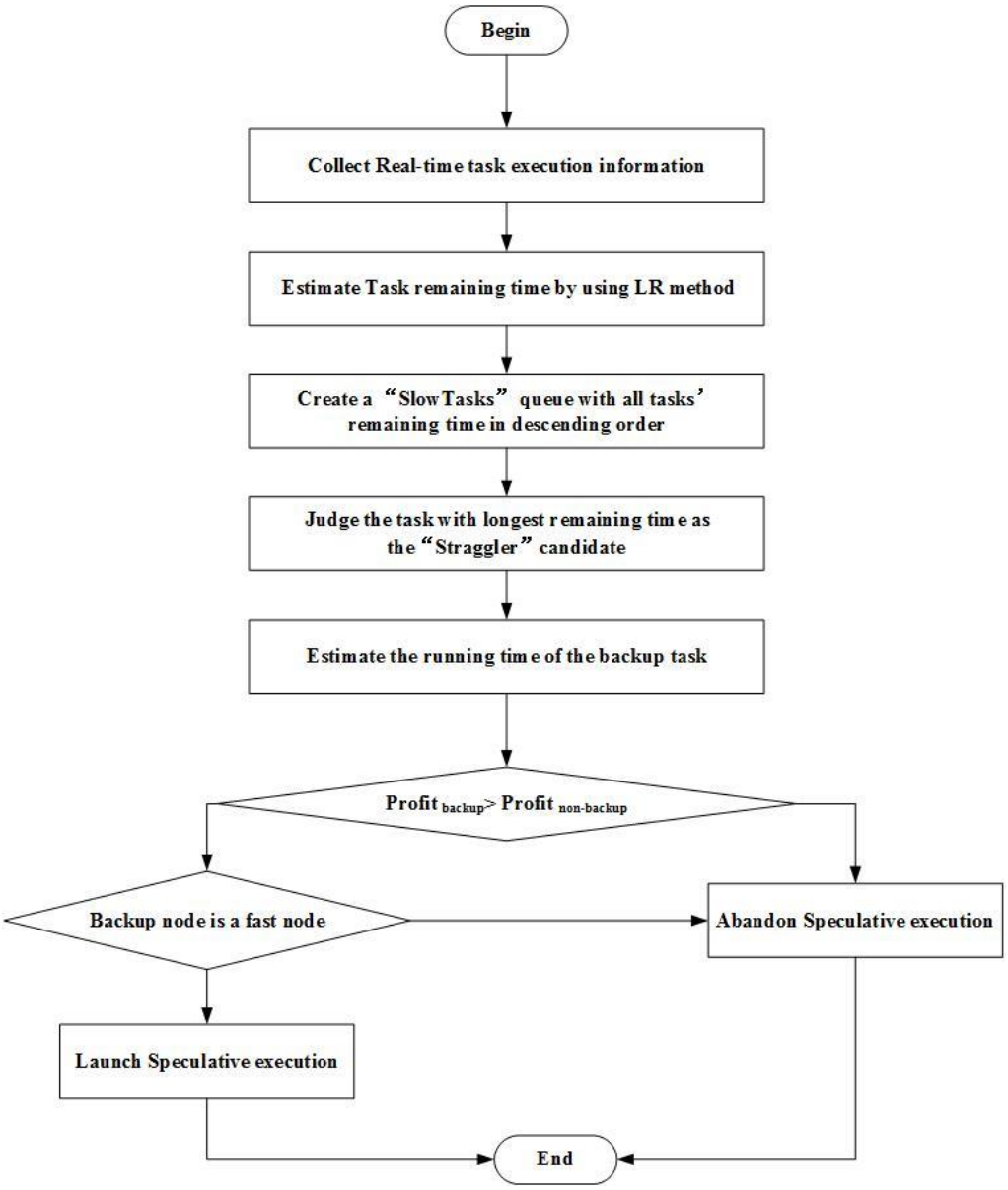


Fig. 1. Flow chart of the LSE strategy

3.2.1 Selecting the "Straggler" Candidates

1. Data Collection of Each Running Task

The first step to determine the "Stragglers" is to collect the detailed information of running tasks, which includes progress and execution time. The original collection format is (progress, Timestamp) , which is written in HDFS in order to extract data features for the following data prediction, the timestamp format is directly converted to the execution time in order to facilitate the following calculation, so the final collection format is (*progress, execution time*).

Fig 2 show examples of detailed execution information when running the Wordcount datasets in the hadoop cluster. The collected data is shown in the figure.

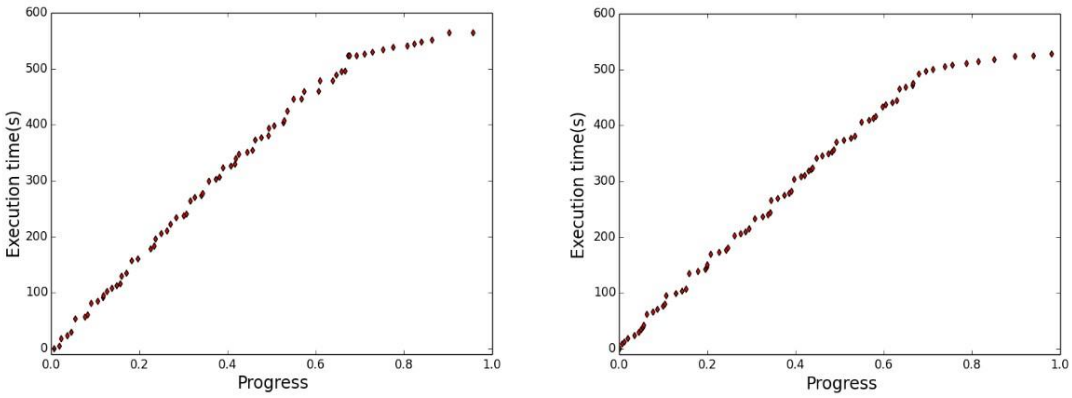


Fig. 2. The execution data collected by running Wordcount. (a) ,(b) show the two different groups of data which are collected from two map tasks when running WordCount.

2. Data Analysis using the Local Regression model

After collecting the execution information of multiple running tasks in the WordCount dataset, we can see from the Fig 2 that there exists the similar tendency between progress and execution time. Due to the non-traditional linear relationship between the progress and execution time of running tasks, we uset the local regession method to estimate the remaining time of the real-time task. It is a typical non-parametric learning algorithm in order to solve the problem that how to establish a linear model on the non-linear data sets. When predicting the value of a point, it selects the points close to the current point rather than all the points to do linear regression, which is the main idea of the local regression.

As mentioned in the previous section, the information storage format for the real-time task collection is (*Progress, Timestamp*). First, the format of the timestamp needs to be converted into the consuming time t, then the input dataset $D=\{(p_i,T_i)|i=1,2,...n\}$, the predicted output of a given input vector is shown in Equation (4):

$$\hat{t} = h_\theta(P) = \sum_{i=0}^{n} \theta_i P_i = \theta^T P \tag{4}$$

Where n is the number of the training samples, in fact, it represents the progress of different tasks, P is the input vector with $n+1$ variables which includes p_0 , p_1 .... p_n , p_0=1, p_1 to p_n for the corresponding progress of the task P, and t is the output variable that indicates the task execution time. θ is the regression parameter and it is needed to ensure that the square error E between the predicted value and the true value is minimized, which is shown in the Equation(5).

$$\min_{\theta} = \sum_{i=0}^{m} \omega_i E^2 = \sum_{i=0}^{m} \omega_i \left[h_\theta(p_i) - t_i \right]^2 \tag{5}$$

In addition, θ is also needed to ensure that the loss function of LR at the prediction q is minimum, the loss function of the local regression algorithm $L(\theta)$ is designed as the following Equations (6).

$$L(\theta) = \frac{\sum_{i=1}^{n} \omega_q \left[h_\theta(P_q) - t_q \right]^2}{2} = \frac{(X\theta - Y)^T W (X\theta - Y)}{2} \tag{6}$$

Where X is an input matrix $X \in \Re^{m \times n}$, in this matrix, rows are the training dataset P_0, P_1.... P_m, n is set to be 2, Y is the output vector, $Y = [t_1, t_2, ...t_m]$, W is a diagonal weight matrix as the Equation (7) shows.

$$W = \begin{pmatrix} \omega_1 & \cdots & \cdots & 0 \\ \vdots & \omega_2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \omega_n \end{pmatrix} \tag{7}$$

Then the regression parameter θ can be calculated using the least square method as shown in the Equation (8) (9) (a prediction point is corresponding to a parameter θ), the final calculated θ is substituted into the formula (4) and then the execution time of the corresponding progress is predicted.

$$\frac{\partial L(\theta)}{\partial \theta} = X^T W X \theta - X^T W Y = 0 \tag{8}$$

$$\theta = \left(X^T W X\right)^{-1} X^T W Y \tag{9}$$

The calculation of the weight function W depends on the distance d. The greater the distance from the predicted point, the smaller the weight will be assigned, otherwise the bigger. Here we use the Gaussian kernel function as follows, where η is the wave-length function that controls the rate at which the weight decreases with distance and is set to be 0.08 by training a large amount of data.

$$\omega(d) = \exp\left(-\frac{d^2}{2\eta^2}\right) \tag{10}$$

d is the Euler distance of the local region (p_k, p_m), which is described as the formula (11) shows:

$$d = \sqrt{\sum_{i=1}^{n}\left(p_k - p_m\right)^2} \tag{11}$$

The Fig 3 below shows the change of the weight function at the prediction point q. The red dotted line shows the selection of the local area when predicting the point q, and the red curve indicates the change of the weight function at the time of prediction. As can be seen, the weight gradually decreases with the increase of distance.

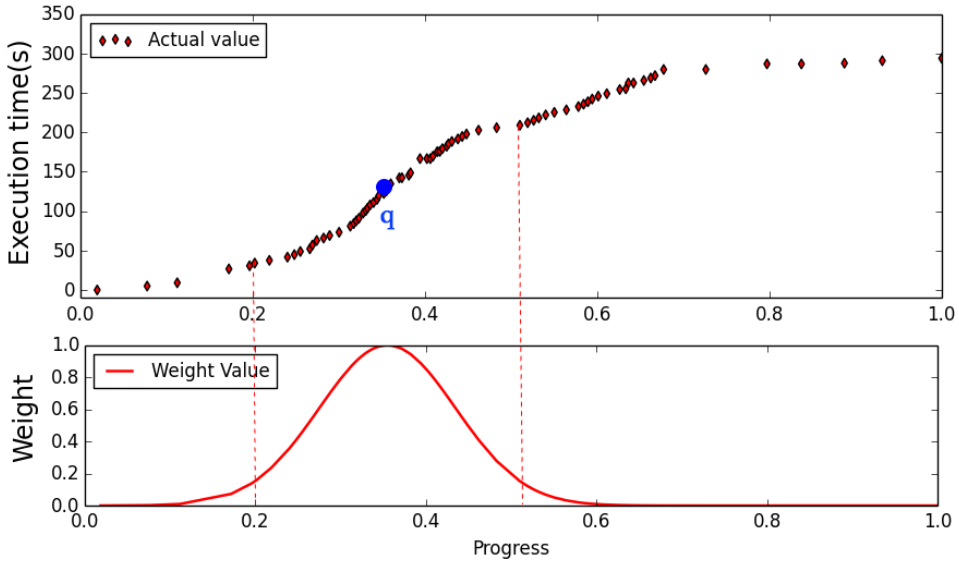


Fig 3. The weight function at the prediction point q

3.2.2 The Benefit Guarantee of Speculative Execution

According to the local regression method, the remaining time of the running task t_{rem} can be obtained, and then sorted in descending order. The task which has the longest task remaining time is judged as a "Straggler" candidates, and then the consumption-benefit model proposed is used to guarantee the profit of cluster. First, the benefits and costs of launching a backup task can be calculated as Table 2 shows.

Table 2. Consumption and Benefits of Speculative Execution

	Launch speculative execution	Non speculative execution
Cluster Consumption	Two slots for t_{backup}	One slot for t_{rem}
Cluster Benefits	One slot for $t_{rem} - t_{backup}$	

Second, we can compute the profits of launching or no-launching the backup task to the cluster. Finally, the "Straggler" tasks will launch backup tasks while satisfying the following condition (12) in order to ensure the maximum efficiency of the cluster.

$$profit_{backup} > profit_{not\_backup} \Leftrightarrow \frac{t_{rem}}{t_{backup}} > \frac{1+2\lambda}{1+\lambda} \tag{12}$$

$$\lambda = load\_factor = \frac{num_{pending\_tasks}}{num_{free\_slots}} \tag{13}$$

Where t_{rem} is the remaining time of the running task, t_{backup} is the execution time of backup task, λ is the load factor of the cluster, which is the ratio of the number of pending tasks to the number of the free slots in the cluster.

In our design, t_{rem} is the remaining time predicted by the local regression method and t_{backup} can be computed as follows, where t_{avg} is the average execution time of the completed task. We introduce a parameter φ to avoid the influence of the amount of the input data on the remaining time of all tasks, and it is the ratio of the amount of input data to the average amount of data.

$$t_{backup} = t_{avg} \times \varphi \tag{14}$$

$$\varphi = \frac{data_{input}}{data_{avg}} \tag{15}$$

3.2.3 The Selection of the Backup Candidate nodes

Factors that affect the performance of speculative execution include not only the accurate selection of the "Straggler" tasks, but also choosing the proper backup nodes with high computing performance. Hadoop original SE strategy does not consider the selection of backup nodes, which results in that the backup task is assigned to node with poor performance, thus affecting the performance of cluster. This paper proposes a new standard to measure the processing capability of backup nodes, different from the previous strategy, it divides the node execution capability into two conditions according to different types of tasks, which includes "Map-Fast" nodes and "Reduce-Fast" nodes. It takes the process rate as the capability of each node, and then it determines the backup node belongs to which type of fast nodes by the following condition.

$$PR_{map} - PR_{avg\_map} \rangle std_{map} \tag{16}$$

$$PR_{reduce} - PR_{avg\_reduce} \rangle std_{reduce} \tag{17}$$

Where PR represents the processing rate of nodes, the std_{map} and std_{reduce} are the standard deviation of progressing rate of these two types. If the expression (16) is satisfied, it is determined as the "Map-Fast" nodes while if the equation (17) is satisfied, it is "Reduce-Fast" nodes. In order to ensure that the backup task completes before the original task, the backup task will be placed on the corresponding fast node to execute.

4 Results and Evaluation

This section evaluates the optimized strategy LSE primarily by comparing it to Hadoop-None, LATE, and MCP. Since LSE estimates the remaining time of running tasks more accurately on the basis of MCP and it is more efficient than LATE in the the estimation of the remaining time and the cluster benefit guarantee. So, in the experimental part, not only the accuracy of prediction needs to be evaluated, but also its performance in a cloud environment.

4.1 Experimental preparation

The experimental cluster uses 64-bit Ubuntu Server 12.04 as the operating system and Hadoop-2.6.0 version as a test platform. The heterogeneous hadoop cluster is built on a server with eight nodes, the processor of the server is four Intel® Xeon® CPU E5649 2.53 GHz , the hard drive is 10 TB and the memory is 288 GB, the detailed information of each node is listed in Table 3. The Wordcount dataset is used as the experimental workloads, which is downloaded from the Purdue mapReduce Benchmarks Suite and widely used to test the performance of the optimized Hadoop frame.

Table 3. The detailed information of each node

NodeID	Memory(GB)	Core Processors
Node 1	10	8
Node 2	8	4
Node 3	8	1
Node 4	8	8
Node 5	4	8
Node 6	4	4
Node 7	18	4
Node 8	12	8

4.2 Performance evaluation of the Optimized model

The Local Regression model is used to estimate the remaining time of the current task, which takes the non-linear relationship between progress and execution time into consideration, Fig. 4 shows the prediction results using Local Regression in the Wordcount, where the red line represents the Local Regression prediction error, and we can see that the predictive accuracy of Local Regression improves a lot comparing to linear regression. Also, RMSE is used to evaluate the accuracy of the prediction, and the calculation formula is as follows.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(P - P_i)^2}{n}} \tag{18}$$

Where P is actual value and P_i represents the prediction value. Table 4 shows the RMSE results of fifteen sets of data, which are randomly selected from the Wordcount. The average prediction RMSE of Wordcount is 1.03, but we can find that there exists some unusually large values, which are mainly caused by resource contention and the non-data locality in the copy phrase of reduce process. If these outliers are ignored, we can find that the average prediction RMSE of Wordcount drops to 0.57.

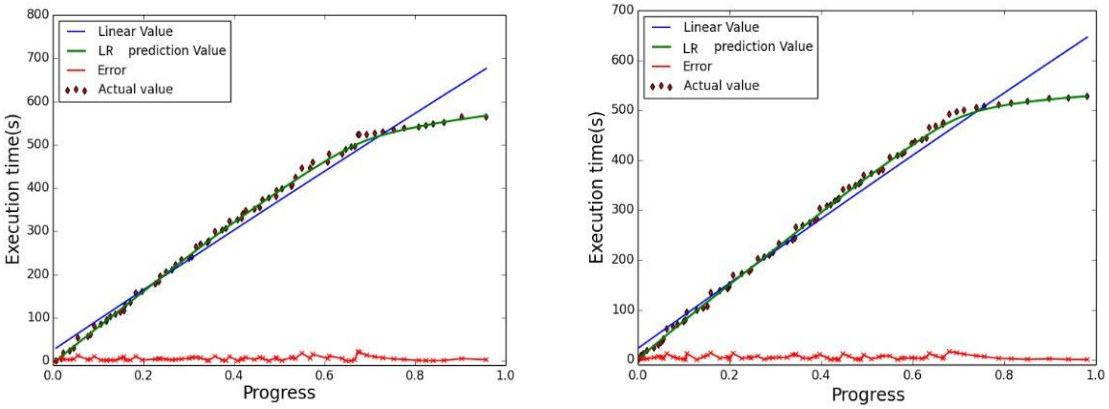


Fig. 4. Comparison of Local Regression and Linear Regression during running a Wordcount task. (a), (b) show the two different groups of training data collected by running WordCount.

Table 4. RMSE of Local Regression for Wordcount workloads

	Task 1	Task 2	Task 3	Task 4
RMSE(s)	0.36	0.49	0.43	0.78
	Task 5	Task 6	Task 7	Task 8
RMSE(s)	2.19	0.77	2.64	0.58

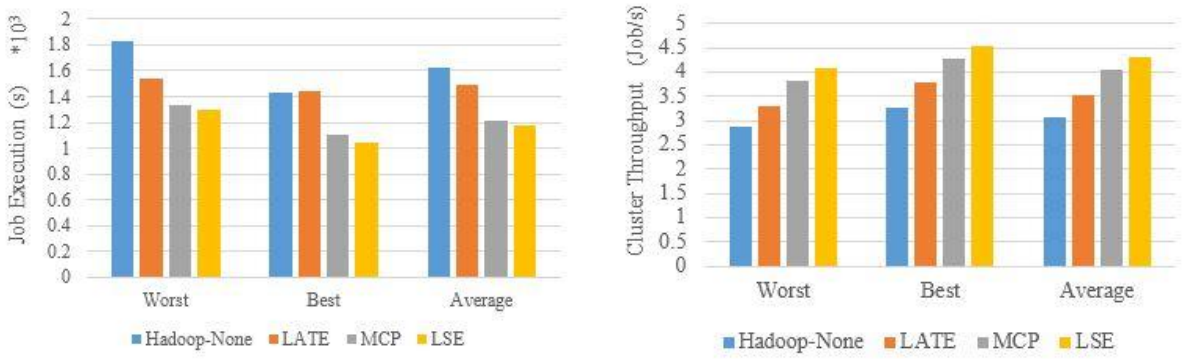
4.3 Performance evaluation of LSE strategy in Heterogeneous environment

1. Performance in the Normal-Load cluster

In this section, the performance of new proposed strategy “LSE” is evaluated in a heterogeneous environment by comparing it with Hadoop-None, Hadoop-LATE, Hadoop-MCP in terms of job execution time and cluster throughput.

In the hetergeneous cluster, the cluster resources are in a shortage, so ensuring the effectiveness of speculative execution is necessary. According to the Hadoop's task initialization strategy, we divide a file into several file blocks, the size of each block is 64MB and each block acts as a Map task. Then we avoid the problem of data skew by setting the input file size to be an integer multiple of 64MB and calculate the execution time of each job and the cluster throughput during running Wordcount dataset. Experimental results is shown in Figure 4.

As can be seen from Fig. 5., For Wordcount, on average, LSE consumes job execution time 5.3\% less than MCP and 21.4\% less than LATE and 27.9\% less than Hadoop-None, moreover, LSE improves cluster throughput by 6.9\% over MCP and 22.9\% over LATE and 43.3\% over Hadoop-None.



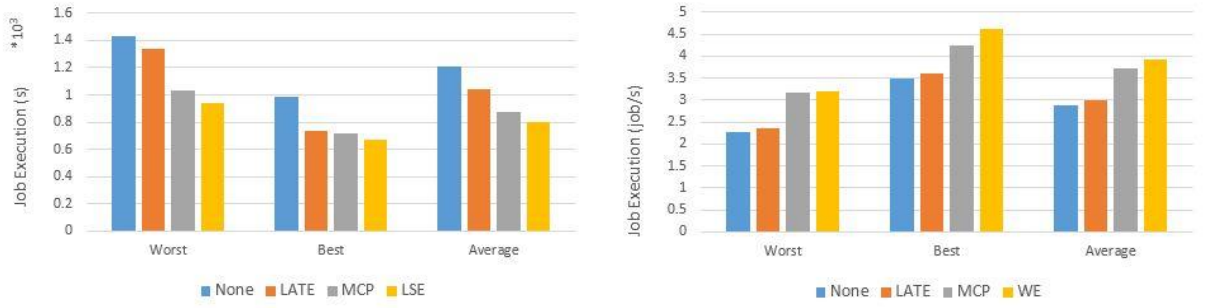
(a) (b)

Fig. 5. Performance comparison of different strategies under Wordcount workloads in the low-load cluster. (a) shows the the Job Execution Time and (b) shows the Cluster Throughput.

2. Performance in the high-load cluster

When the load of cluster is high, that is the cluster resource is in short supply, it is more necessary to ensure the accuracy of speculative execution. If the accuracy is not high, the cluster resources will be occupied and the performance of the whole cluster will be slowed down. In order to simulate the high-load cluster, we open some other run computing-intensive or IO-intensive tasks on some nodes when cluster has run for a period of time and submit a Wordcount job every 150 seconds, then we calculate the execution time and cluster of each job. Experimental comparison is shown in Figure. 10.

Fig.6 shows the performance comparison results, on average, LSE decreases job execution time by 7.7% than MCP and 22.6% than LATE and 33.3% than Hadoop-None, moreover, LSE makes more improvements on cluster throughput 5.4% than MCP and 33.4% than LATE and 39.2% than Hadoop-None when executing Wordcount Tasks. This indicates that LSE can still adapt well when the load of cluster is high.



(a) (b)

Fig. 6. Performance comparison of different strategies under Wordcount workloads in the high-load cluster

In summary, the LSE strategy in this paper gains a certain optimization compared to MCP and LATE, Hadoop-None in the cluster. Since LSE predicts the remaining time of running tasks timely according to the collect-

ed data and combines with the partial selection strategy in MCP, which is an optimization of MCP and greatly improves the accuracy of speculative execution compared to LATE.

5 Conclusions

Inspired by the non-linear relationship between job execution and progress, we propose a new speculative execution strategy in this paper, different from previous strategies, LSE first collects tasks' information in real-time during task execution, then uses local regression method to predict remaining time of running tasks, which aims to improve data prediction accuracy, and finally combines some determination methods of MCP to ensure the effectiveness of speculative execution. The experimental results show that the LSE fits better than MCP and LATE in both high-load and low-load cluster, as well as dealing with data skew in Map phase.

6 Acknowledgement

This work is supported by the NSFC (61300238, 61300237, 61232016, 1405254, and 61373133); Marie Curie Fellowship (701697-CAR-MSCA-IFEF-ST); the 2014 Project of six personnel in Jiangsu Province under Grant No. 2014-WLW-013; the 2015 Project of six personnel in Jiangsu Province under Grant No. R2015L06; Basic Research Programs (Natural Science Foundation) of Jiangsu Province (BK20131004); and the PAPD fund.

References

[1] L. F. Vaquero, L. Rodero-Merino, J. Caceres, M. Lindner, A break in the clouds: towards a cloud definition, Acm Sigcomm Computer Communication Review 39 (1) (2008) 50-55.

[2] M.H. Iqbal, T.R. Soomro, Big Data Analysis: Apache Storm Perspective, International Journal of Computer Trends & Technology 19 (1) (2015) 9-14.

[3] M. Zaharia, M. Chowdhury, M.J. Franklin, S. Shenker, I. Stoica, Spark: cluster computing with working sets, in: Proc. USENIX Conference on Hot Topics in Cloud Computing. USENIX Association, 2010, pp. 1765-1773.

[4] M. Isard, M. Budiu, Y. Yu, A. Birrell, D. Fetterly, Dryad: distributed data-parallel programs from sequential building blocks, ACM SIGOPS Operating Systems Review 41 (3) (2007) 59-72.

[5] T. Gunarathne, T.L. Wu, J. Qiu, G. Fox, MapReduce in the Clouds for Science, in: Proc. Second international conference on Cloud computing, 2010, pp. 565-572.

[6] Apache Hadoop, <http:// Hadoop. Apache. Org/>, 2016.

[7] S. Ghemawat, H. Gobioff, S. T. Leung, The Google File System, ACM SIGOPS Operating System Review 37 (5) (2003) 29-43.

[8] J. Dean, S. Ghemawa, MapReduce: Simplified Data Processing on Large Clusters, in: Proc. Conference on Symposium on Operating Systems Design & Implementation, 2004, pp.107-113.

[9] A. Thusoo, J.S. Sarma, N. Jain, Z. Shao, P. Chakka, "Hive: a warehousing solution over a map-reduce framework" in: Proc. Vldb Endowment, 2009, pp.1626-1629.

[10] B. Vijayalakshmi, P. R. Ravi, The Down of Big Data-Hbase, in: Proc. It in Business, Industry and Government, IEEE, 2014.

[11] F. Chang, J. Dean, S. Ghemawa, A Distributed Storage System for Structured Data, ACM Transactions on Computer Systems 26 (2) (2008) 1–26.

[12] A. Wierman, M. Nuyens, Scheduling despite inexact job-size information, in: Proc. International Conference on Measurement and Modeling of Computer Systems, 2008, pp. 25-36.

[13] D. G. Yoo, K.M Sim, A Comparative Review of Job Scheduling For MapReduce, in: Proc. IEEE International Conference on Cloud Computing and Intelligence Systems, 2011, pp. 353–358.

[14] S. N. Nenavath, N. Atul, A Review of Adaptive Approaches to MapReduce Scheduling in Heterogeneous Environments, in: Proc. International Conference on Advances in Computing, Communications and Informatics, 2014, pp. 677–683.

[15] Q. Liu, D. Jin, X. Liu, N. Linge, A Survey of Speculative Execution Strategy in MapReduce, in: Proc. the 2nd International Conference on Cloud Computing and Security, Nanjing, 2016, pp. 232-243.

[16] C. Lin, J. Liao, A job-oriented load-distribution scheme for cost-effective NameNode service in HDFS, International Journal of Web & Grid Services 10 (4) (2014) 319-337.

[17] Y. Wang, J. Zhou, C. Ma, W. Wang, A distributed file system of expandable metadata service derived from HDFS, in: Proc. International conference on Cluster computing, 2012, pp. 126-134.

[18] Q. Wei, B. Veeravalli, B, Gong, L. Zeng, D. Feng, CDRM: A Cost-Effective Dynamic Replication Management Scheme for Cloud Storage Cluster, in: Proc. International conference on Cluster computing, 2010, pp. 188-196.

[19] J. Shafer, S. Rixner, A. L. Cox, The Hadoop distributed filesystem: Balancing portability and performance, in: Proc. International Symposium on Performance Analysis of Systems & Software, 2010, pp. 122-133

[20] M. Zaharia, A. Konwinski, A. Joseph, R. Katz, I. Stoica, Improving MapReduce performance in heterogeneous environments, in: Proc. 8th USENIX Conference on Operating Systems Design and Implementation (OSDI), 2008, pp. 29–42.

[21] X. Huang, L. Zhang, R. Li, L. Wan, K. Li, Novel Heuristic Speculative Execution Strategies in Heterogeneous Distributed Environments. Computers and Electrical Engineering (2015).

[22] Q. Chen, C. Liu, Z. Xiao, Improving MapReduce Performance Using Smart Speculative Execution Strategy, IEEE Transactions on Computers 63 (4) (2014) 954-967.

[23] H. Wu, K. Li, Z. Tang, L. Zhang, A Heuristic speculative execution strategy in heterogeneous distributed environments, in: Proc. The Sixth International symposium on Parallel Architectures, Algorithms and Programming (PAAP), 2014, pp. 268–273.

[24] Q. Liu, W. Cai, J. Shen, Z. Fu, N. Linge, A Smart Strategy for Speculative Execution based on Hardware Resource in a Heterogeneous Distributed Environment, International Journal of Grid Distributed Computing 9 (2015) 203–214.

[25] Y. Wang, W. Lu, R. Lou, B. Wei, Improving MapReduce Performance with Partial Speculative Execution, Journal of Grid Computing 13 (2015) 587–604.

[26] Y. Li, Q. Yang, S. Lai, B. Li, A New Speculative Execution Algorithm based on C4.5 Decision Tree for Hadoop, In: Proc. the International Conference of Young Computer Scientists, Engineers and Educators (ICYCSEE 2015), 2015, pp. 284–291.

[27] S. Yang, Y. Chen, Design Adaptive Task Allocation Scheduler to improve MapReduce Performance in Heterogeneous Clouds. Journal of Network & Computer Applications 57 (2015) 61–70.