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

Xiaodong Liu
School of Computing,
Edinburgh Napier University
10 Colinton Road,
Edinburgh, EH10 5DT, UK

Qi Liu
School of Computing,
Edinburgh Napier University
10 Colinton Road,
Edinburgh, EH10 5DT, UK
Emai1: q.liu@napier.ac.uk

***Abstract*—Hadoop is a famous distributed computing framework that is applied to process large-scale data. "Straggling tasks" have a serious impact on Hadoop performance due to imbalance of slow tasks distribution. Speculative execution (SE) presents a way to deal with Straggling tasks by monitoring the real-time progress of running tasks and replicating potential "Stragglers" on another node to increase the opportunity of completing backup tasks ahead of original. Current proposed SE strategies meet their challenges such as misjudgment of "Straggling tasks", improper selection of backup nodes, etc., which result in inefficient performance of the SE and its Hadoop system. In this paper, we propose an optimized SE strategy based on local data prediction, which collects task execution information in real time and uses Locally Weighted Regression (LWR) to predict remaining time of each running tasks, and selects an appropriate backup task node according to the actual requirements. It also combines a cost-benefit model to maximize the effectiveness of SE. According to the results, the proposed SE strategy implemented in Hadoop-2.6.0 enhances the accuracy of selecting potential Straggler task candidates, and shows better performance in various situations in a heterogeneous Hadoop environment.**

Keywords—Hadoop; Speculative Execution; Straggling Task; LWR; Prediction Accuracy

I. INTRODUCTION

In recent years, cloud computing has served as a new distributed computing model, which has attracted wide concerns and interests from the academia and industry [1]. The hadoop, which acts as the top project of Apache and one of the most popular cloud computing frameworks, has been widely adopted for its distributed features on data storage, computing and searching [2].

The process of job scheduling in a Hadoop system aims to divide a job into multiple tasks, and then provoke a JobTracker service to assign the tasks to corresponding TaskTracker nodes. Distributing tasks as fast as possible cannot guarantee that subsequent execution in each TaskTracker still maintains its superiority [3], and may lead to the so-called slow tasks, "Straggler". Speculative Execution (SE) is the current effective mechanism to recognize and correct inefficient allocation made by a JobTracker service so as to improve the fault tolerance feature of the Hadoop [4]. The main goal of an SE strategy is to find out potential stragglers and back them up to TaskTracker nodes that can complete the backup tasks faster than the original ones, thereby reduce the execution time of the job and increase the performance of the entire cluster [5].

Those existing optimized SE strategies such as LATE[6], ERUL[7], MCP[8], etc. were designed to recognize slow tasks based on the self-estimation of the tasks' remaining time, so suffer from possible inappropriate allocation due to inaccurate estimation. Therefore, we propose a new speculative execution strategy called Locally Weighted Regression based Speculative Execution (LWR-SE), which aims to predict the remaining time of a run-time task more accurately with maximum benefits as far as the entire cluster is concerned.

The rest of this paper is arranged as follows. Section II lists related research work on current optimization strategies for Hadoop speculative execution mechanism. Our design of "LWR-SE" is presented in Section III. Section IV compares and evaluates our strategy with current algorithms through experiments, finally, the whole work in this paper and critical future work are concluded in Section V.

II. RELATED WORK

In order to improve the performance of Hadoop-Naive in heterogeneous environments, Zaharia, et al. presented an SE strategy called LATE, which uses the remaining time as the speculative execution priority [6]. Wu et al. proposed a SE strategy called ERUL, presenting the linear relationship between system load and the remaining time of a task, which calculates the remaining time by the real-time system load and improves the accuracy of the prediction [7].

Since the previous strategies did not take into account cluster efficiency, MCP was proposed to calculate and maximize the benefits of launching backup tasks [8]. An optimized SE strategy called Ex-MCP was presented by taking the value of nodes into consideration compared with the MCP [9]. In addition, some other optimization strategies are proposed. Wang et al. presented an improved strategy called Partial Speculative Execution (PSE) to enhance the efficiency of SE, with no consideration of the difference between processors [10]. In [11], an efficient SE strategy (SECDT) was designed and implemented by calculating the completion time of the task based on a C4.5 decision tree algorithm. Besides, a dynamic slot allocation was put forward to improve the performance of job execution and Speculative Execution Performance Balancing (SEPB) was designed to achieve the balance between a single jobs and multiple jobs [12].

In summary, existing SE strategies still meet difficulties in the accuracy when recognizing potential “Straggler” tasks and the efficiency during backing it up in an appropriate fast node. Furthermore, maintaining the trade-off between local effectiveness and overall benefit becomes also challenging.

III. MODEL AND ALGORITHM

In order to better evaluate and predict running tasks and to bring maximum benefit to the cluster while launching the backup tasks, we propose a new method called “LWR-SE”. In this strategy, consequent decision on whether to start backing up is made depending on the recognition of a Straggler, the benefits/costs to replicate it and the selection of a suitable backup node.

A. The Recognition of Straggler Candidates

- Collect the detailed information of run-time tasks.

The first step to determine a Stragglers is to collect the detailed information of running tasks, which includes their progress and execution time. Original collected data on task progress, formatted as (progress, Timestamp), are retrieved from the HDFS in order to extract data features for prediction. The progress pair is then converted to (progress, execution-time), where the timestamp is directly converted to the execution time in order to facilitate following calculation.

- Design a locally weighted regression model to adapt task progress to achieve local prediction accuracy.

After collecting the execution information of multiple running tasks in the Wordcount dataset, a non-traditional linear relationship between progress and execution time can be found.

With regard to the converted datasets formatted as (progress, execution-time), the input dataset D={(p_i, t_i)|$i=1,2,\dots,n$}$(p_i, t_i)|i = 1,2,\dots,n\}$, the predicted output of a given input vector is shown in Equation (1).

$$\hat{t} = h_\theta(p) = \sum_{i=0}^{n} \theta_i p_i = \theta^T p \tag{1}$$

Where n is the number of the training samples. p is the input vector with $n+1$ variables which includes p_0, $p_1 \dots$ p_n, where p_0=1, and p_1 to p_n are for the corresponding progress of the task p. t is the output variable that indicates the consuming time of task execution.

θ is needed to ensure that the loss function $J(\theta)$ of the LWR at the prediction point (p_q, t_q) is minimum, the loss function is designed as follows:

$$J(\theta) = \frac{\sum_{i=1}^{n} \omega_i \left[h_\theta\left(p^i\right) - t^i \right]^2}{2} = \frac{(X\theta - Y)^T W (X\theta - Y)}{2} \tag{2}$$

$$\frac{\partial J(\theta)}{\partial \theta} = X^T W X \theta - X^T W Y = 0 \tag{3}$$

$$\theta = \left(X^T W X\right)^{-1} X^T W Y \tag{4}$$

Where X is an input matrix, Y is the output vector. W is a diagonal weight function matrix.

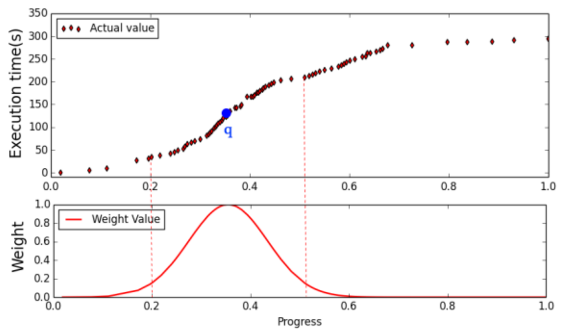
A Gaussian kernel function is therefore used to calculate the weight function $\omega(d)$ as in Equation (5), where γ is the wave-length parameter and is set to 0.08 in this paper.

$$\omega(d) = \exp\left(-\frac{d^2}{2\gamma^2}\right) = \exp\left(-\frac{\sum_{i=1}^{n}\left(p^{(q)} - p^{(i)}\right)^2}{2\gamma^2}\right) \tag{5}$$

- Predict the remaining time of the run-time tasks at particular local query points.

The Fig. 1 below shows the variation 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. 1. The weight function at the prediction point q.



B. The Benefit Calculation of Replicating Stragglers

According to the designed LWR model, the remaining time of the running task t_{rem} can be obtained and then sorted in a descending order. The task which has the longest task remaining time is judged as a “Straggler”candidate and then a cost-benefit model is designed to guarantee the benefit of the entire cluster. The benefits and consumption of launching a backup task are considered and calculated, as shown in TABLE I.

TABLE I. CONSUMPTION AND BENEFITS OF SE

	SE Enabled	**SE Disabled**
Cluster Consumption	Two slots for t_{backup}	One slot for t_{rem}
Cluster Benefits	One slot for $t_{rem} - t_{backup}$	

Then, a backup task of a recognized “Straggler” will be launched only while satisfying the following conditions in order to ensure the maximum efficiency of the cluster.

$$Profit_{SE-Enabled} \rangle Profit_{SE-Disabled} \quad (6)$$

In our design, t_{rem} is the remaining time predicted by the LWR model, t_{backup} can be further computed as in Equation (7), t_{avg} is the average execution time of completed tasks. μ is introduced to avoid the influence of the data skew of the input data.

$$t_{backup} = t_{avg} \times \mu \quad (7)$$

$$\mu = \frac{data_{input}}{data_{avg}} \quad (8)$$

C. *The Selection of Backup Nodes*

To enhance the performance of SE, we proposes a new method to measure and assess potential backup nodes by dividing the nodes into two good-at groups, i.e. "Map-Fast" nodes and "Reduce-Fast" nodes. In terms of backup node candidates with high computing abilities, it takes real-time process rate of the backup nodes as a critical metric, and determines the backup node eventually by following conditions.

$$PR_{map} - PR_{avg\_map} \rangle std_{map} \quad (9)$$

$$PR_{reduce} - PR_{avg\_reduce} \rangle std_{reduce} \quad (10)$$

Where PR represents the processing rate of node candidates, and std_{map} and std_{Reduce} are the standard deviation of progressing rate in these two stages. If the Condition (9) is satisfied, it is determined as a "Map-Fast" node, while if the Condition (10) is satisfied, it is a "Reduce-Fast" node. In order to ensure a backup task is completed ahead of the original Straggler, it will be placed in a corresponding fast node to execute depending on what stage it resides in.

IV. RESULTS AND EVALUATION

In this section, the performance of our proposed Locally Weighted Regression based SE strategy (LWR-SE) 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.

A. *Experimental Environment Preparation*

The experimental cluster uses 64-bit Ubuntu Server 12.04 as the operating system and Hadoop-2.6.0 as a test platform. The heterogeneous Hadoop cluster is built on a server with eight virtual nodes. The server contains four Intel® Xeon® CPU E5649 2.53 GHz (24 core processors in total), 10 TB of hard drive and 288GB of memory. The detailed specification of each node is listed in TABLE II. 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 II. 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

B. *Performance evaluation of LWR-SE strategy in Heterogeneous environment*

- Normal Load Scenario

In the normal load scenario, a low-load cluster has been configured with efficient resources. The execution time of each job and the cluster throughput running Wordcount dataset is calculated, with experimental results shown in Fig. 2 and Fig. 3.

As can be seen from Fig. 2 and Fig. 3, when a Wordcount job is processed, the job execution time taken by the LWR-SE is less than the MCP by 5.3%, less than the LATE by 21.4%, less than the Hadoop-None by 27.9% on average. Moreover, the LWR-SE improves cluster throughput by 6.9% over the MCP, 22.9% over the LATE, and 43.3% over the Hadoop-None on average.

In summary, the LWR-SE strategy achieves improvements over the MCP, LATE and Hadoop-None in a normal cluster load scenario with no data skew. Since the MCP predicts the remaining time of running tasks timely combined with a partial selection strategy for the selection of backup nodes, the LWR-SE depicts slight optimization over the MCP but great improvements over the LATE and Hadoop-None.

Fig. 2. Job Execution Time of different SE strategies on Wordcount jobs in a normal load scenario

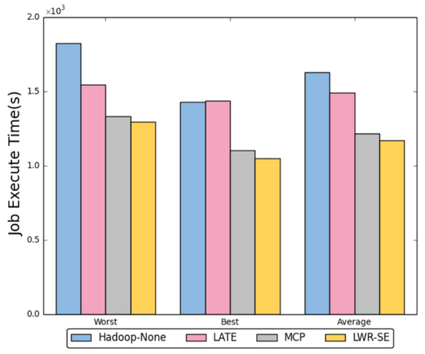
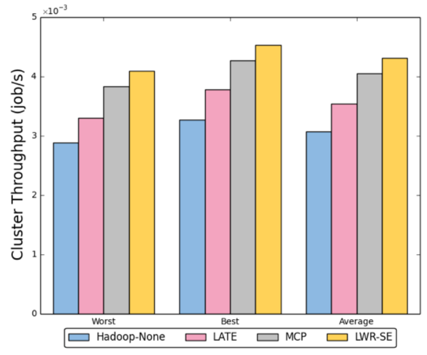


Fig. 3. Cluster Throughtput of different SE strategies on Wordcount jobs in a normal load scenario



- Busy Load with Data Skew Scenario

In a practical cloud environment, data skew situation is common, especially in the Map stage, which will result in "Stragglers" misjudgments due to the different size of input data. In order to create a data skew scenario, the Wordcount jobs have been proposed with 30GB of its dataset in total and 100MB of input data in each. The input data were divided into two data blocks due to the Hadoop self-split strategy, which are 64MB and 36MB.

Fig. 4. Job Execution Time of different SE strategies on Wordcount jobs in a busy load with data skew scenario

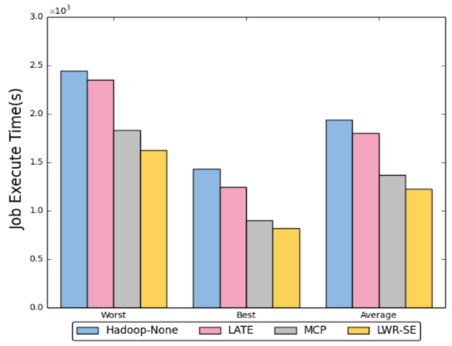
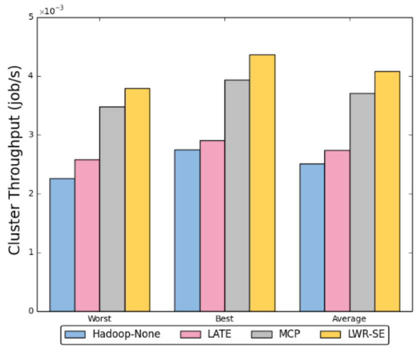


Fig. 5. Cluster Throughtput of different SE strategies on Wordcount jobs in a busy load with data skew scenario



According to Fig. 4 and Fig. 5, the performance of the LWR-SE has been significantly improved, which consumes the job execution time 10.5% less than the MCP, 31.9% less than the LATE and 36.9% less than the Hadoop-None. Moreover, the LWR-SE improves the cluster throughput by 8.1% over the MCP, 48.1% over the LATE and 62.8% over the Hadoop-None. The LWR-SE makes large improvement over the LATE and Hadoop-None since they have not taken data skew into account. Even compared with the MCP, which considers the problem of data skew, the LWR-SE has depicted about 10% of improvement on both metrics due to its higher prediction accuracy prediction of remaining time than the MCP.

V. Conclusions

In this paper, we have proposed a novel speculative execution strategy, LWR-SE inspired by the non-linear relationship between job execution time and progress. Different from previous strategies, the LWR-SE collects real-time information of tasks during task execution, predicts the remaining time of running tasks based on a locally weighted learning method for locally higher prediction accuracy, and ensures the effectiveness of speculative execution and the maximum benefits of the entire cloud system. The experimental results have shown that the LWR-SE outperforms the MCP, LATE and Hadoop-None in three different heterogeneous scenarios designed with either normal or busy workloads.

Acknowledgment

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 701697.

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, vol. 39, no. 1, pp. 50-55, 2008.

[2] Z. Li, H. Shen, W. Ligon, J. Denton, "An Exploration of Designing a Hybrid Scale-Up/Out Hadoop Architecture Based on Performance Measurements," IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 2, pp. 386-400, 2017.

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

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

[5] H. Xu, W. Lau, "Speculative Execution for a single job in a MapReduce-Like system," International conference on Cluster computing, 2014, pp. 586-593.

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

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

[8] Q. Chen, C. Liu, Z. Xiao, "Improving MapReduce Performance Using Smart Speculative Execution Strategy," IEEE Transactions on Computers, vol. 63, no. 4, pp. 954-967, 2014.

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

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

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

[12] S. Tang, B. Lee, B, He, "DynamicMR: A Dynamic Slot Allocation Optimization Framework for MapReduce Clusters," IEEE Transactions on Cloud Computing, vol. 2, no. 3, pp. 333-347, 2014.