Near-Data Prediction Based Speculative Optimization in a Distribution Environment

Mingxu Sun[1], Xueyan Wu[2], Dandan Jin[3](✉), Xiaolong Xu[3], Qi Liu[4], and Xiaodong Liu[5]

[1] School of Electrical Engineering, University of Jinan, Jinan, China
[2] Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET), Nanjing University of Information Science and Technology, Nanjing 210044, China
[3] School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China
18751971087@163.com
[4] Shandong Beiming Medical Technology Co., Ltd., Jinan, China
[5] School of Computing Edinburgh, Napier University Edinburgh, Edinburgh, UK

Abstract. Apache Hadoop is an open source software framework that supports data-intensive distributed applications and is distributed under the Apache 2.0 licensing agreement, where consumers will no longer deal with complex configuration of software and hardware but only pay for cloud services on demand. So how to make the performance of the cloud platform become more important in a consumer-centric environment. There exists imbalance between in some distribution of slow tasks, which results in straggling tasks will have a great influence on the Hadoop framework. By monitoring those tasks in real-time progress and copying the potential Stragglers to a different node, the speculative execution (SE) realizes to improve the probability of finishing those backup tasks before the original ones. The Speculative execution (SE) applies this principle and thus proposed a solution to handle the Straggling tasks. At present, the performance of the Hadoop system is unsatisfying because of the erroneous judgement and inappropriate selection for the backup nodes in the current SE policy. This paper proposes an SE optimized strategy which can be used in prediction of near data. In this strategy, the first step is gathering the real-time task execution information and the remaining runtime required for the task is predicted by a local prediction method. Then it chooses a proper backup node according to the near data and actual demand in the second step. On the other side, this model also includes a cost-effective model in order to make the performance of SE to the peak. The results show that using this strategy in Hadoop effectively improves the accuracy of alternative tasks and effects better in heterogeneous Hadoop environments in various situations, which is beneficial to consumers and cloud platform.

Keywords: Distributed systems · Hadoop · Speculative execution · Locally weighted regression · Near data prediction

M. Sun and X. Wu—Both authors are the first author due to equal contribution to this paper.

X. Zhang et al. (Eds.): CloudComp 2019/SmartGift 2019, LNICST 322, pp. 111–122, 2020.
https://doi.org/10.1007/978-3-030-48513-9_9

1 Introduction

As the internet has successfully occupied many aspects of people's lives, the amount of data stored in the consumer's private cloud will grow exponentially in the next few years, many consumers need to pay for the cloud service on demand [1]. Whilst cloud computing platforms have evolved such as Apache Storm [2], Spark [3], and Hadoop [4].

Hadoop is widely used in distributed data storage, computing and search functions areas because of the Apache top project and the prevalent cloud computing frameworks [5]. Many strategies have been designed to improve the effectiveness and efficiency of Hadoop clusters and facilitate big data storage and analytics [6], but the inefficient resource allocation in Hadoop job scheduling still bring many difficulties.

Allocation and Coordination of tasks among TaskTrackers has therefore become critical and challenging in a JobTracker due to lack of runtime information of TaskTrackers and difficulty in predicting the completion duration of each tasks [7]. The most effective mechanism to improve Hadoop's fault tolerance is Speculative Execution (SE), which identifies and corrects the inefficient allocation of JobTracker. [8]. Previous research efforts have been conducted to optimize the SE strategy, Although the purpose of these strategies is to identify the remaining time of the task through slow tasks, such self-estimation is often inappropriate due to inaccuracy. [9].

In this paper, we pay attention to real-time task execution and collect the relevant information during a task's run time. A local weighted prediction method called LWR-SE is employed to estimate the running time required for the task. In parallel, the max cost-consumption model and the more appropriate selection strategy of back up task execution nodes are combined. In this way, both cloud platform providers and consumers can take advantage of it. And Sect. 2 lists current user-centric cloud environment research and Hadoop-based fault-tolerant optimization strategies. Section 3 presents the "LWR-SE" we designed, and the reliability of the method was verified by experimental methods in Sect. 4. In Sect. 5, we summarize this article's work and list some key work to be done in the future.

2 Related Work

2.1 Service in User-Centric Cloud

The combination of the consumer electronics industry and cloud computing has led to a growing number of researchers focusing on user-centric cloud services. A new architecture called IDM based on privacy and reputation extensions was put forward to enhance the security of consumers' identity [10]. A new architecture "SuSSo" is designed to deal with the limitation of service continuity when across different consumer electronic devices combined with the cloud computing [11]. Abolfazli et al. gave an overall analysis and compared the different solutions on the mobile cloud computing in the fields of consumer electronics [12]. Fu et al. proposed a new useful multi-keword ranked search strategy towards the encrypted cloud data, which supports synonym queries at the same time [13]. Grzonkowski et al. raised a more secure authentication method for home networks in user-centric cloud environment [14]. Due to the complexity and difference of big data, they propose a cloud computation offloading method, named COM, dynamic schedules of data/control-constrained computing tasks are confirmed [15]. A new and systematic smart home management system, which was deployed in the cloud and acted as the community broker, is presented to provide more electronic information service [16].

2.2 Fault Tolerance in Hadoop

On the other hand, the temporal fault-tolerance aims to automatically detect and restore fault run-time tasks so that it can shorten the execution time, and improve the computing performance and reliability of a cloud system, which involves strategies on MapReduce job and task scheduling, enhancement of speculation execution (SE) strategies, etc. [17].

The original speculative execution was implemented as Hadoop-Naïve in Hadoop [18]. Its primary idea was to recognize a task as a "Straggler" if its progress is below the average level, which can cause misjudged tasks and wasted cluster resources. It goes even worse in a heterogeneous environment. With regard to the average rate used in the LATE to calculate the remaining time of running tasks, which may lead to inaccurate or even incorrect prediction. In 2015, Wu's team improved the accuracy of the prediction by calculating the remaining time of system load situation calculation task [19].

MCP proposal can maximize the startup backup task, which solves the problem that the previous SE strategy is not old, by dividing Map tasks into map and combiner stages, and Reduce tasks into copy, sort and reduce phases [20]. In 2014, an SE optimization algorithm called Ex-MCP was proposed to compare node values with MCP [21]. On top of that, there are some optimization methods put forward. A execution was proposed based on sort nodes out according to the hardware performance of the nodes [22]. Wang et al. proposed a PSE optimization strategy that can ignore the differences between different processors to improve the efficiency of speculative execution [23]. In [24] An effective speculative execution strategy (SECDT) is proposed. The completion time required for the task is calculated by decision tree [25]. Besides, an ATAS strategy can improve the Hadoop's expansive ability by increasing the estimate accuracy on the execution time of backing-up tasks [26]. Adaptive allocation scheduling can also be used for NILM algorithms based on power allocation [27]. Due to the imbalance of cloud platform performance, Edge Computing Nodes (ECNS) has been proposed as an alternative solution for cloud computing in recent years. The team of Xu uses non-dominated sorting genetic algorithm II (NSGA-II) to achieve multiple Target optimization, shortening the unloading time of computing tasks and reducing the energy consumption [28].

In general, the current SE strategy still has great difficulties in quickly backing up and accurately identifying1 potential Straggler tasks in appropriate nodes, and how to balance the overall benefits while maintaining the processing of local universities is also very large challenge.

3 Model and Algorithm

In this section, we introduce a speculative execution method named "LWR-SE". The flow chart of the method is indicated in Fig. 1, with more details discussed in the rest parts of this section.

3.1 The Recognition of Straggler Candidates

Data Collection of Running Tasks. First, confirm Stragglers by collecting detailed information such as the progress and execution time of real-time tasks. To collect features, the raw data is collected from the HDFS (progress, Timestamp) to facilitate prediction. Then convert the progress pair to (progress, execution time) in order to simplify the algorithm. The algorithm for data collection is shown as follows (Table 1).

Table 1. Algorithm for data collection

```
Algorithm 1: Data Collection
Input: Job Status (JS), MapTask Report (MR), The running task attempt (RT), The id of task attempt (IT), A context object for task (TC), The progress of a running task (P), Execution time (ET)
Steps:
Get the JS from JobClient
Traverse the JS:
Get the MRs from JobId
For each MR in the MRs
    Get the RTs from MR
    Get the collection containing the IT
    Get P from TC
    If P has changed
        Write the P and ET to the file named with IT
    End If
EndFor
End Data Collection
```

Figure 1 and Fig. 2 show examples of detailed execution information when running the Wordcount and Sort datasets in the Hadoop cluster. The collected data is shown in the figures.

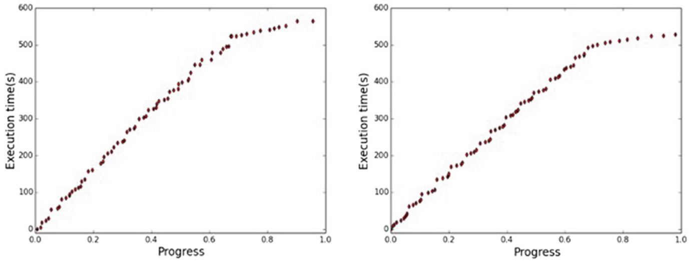

Fig. 1. The execution data collected by running Wordcount

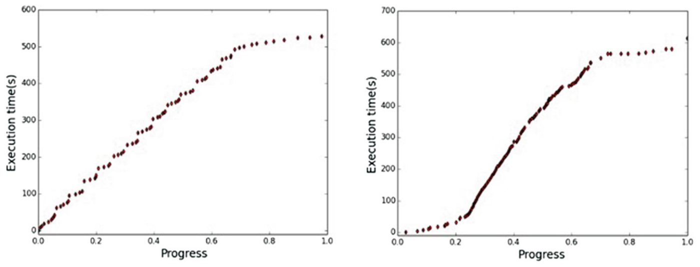

Fig. 2. The execution data collected by running Sort

A Locally Weighted Learning Model. After the running task execution information is collected and Sort datasets, similar tendency can be seen in the figures above. In order to define such non-traditional linear relationship in between, a non-parametric learning algorithm using local weighted regression is designed to establish a linear model on the non-linear datasets. The input dataset D = {(p_i,t_i)|i = 1,2,…,n}, and the predicted output as below:

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

The number of training set samples is set to n, representing the progress of different tasks. p represents the progress which is an input n + 1 dimension. t indicates the execution time of the task. θ is the regression parameter and it should satisfy that the square error Minimize between predicted and true values, which is proposed in Eqs. (2) and (3).

$$E = h_\theta\left(p^i\right) - t^i \tag{2}$$

$$\min_\theta = \sum_{i=0}^{m} \omega_i E^2 = \sum_{i=0}^{m} \omega_i \left[h_\theta\left(p^i\right) - t^i\right]^2 \tag{3}$$

Where E represents the error, (p_i, t_i) is the *ith* training samples, ω_i is the weight in the *ith* local forecasting area, which depends on the local prediction point. To simplify the description, we can transform it into a matrix representation as shown in Eq. (4).

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

Where X is a matrix, with m rows training dataset p_0, p_1…, p_m and n being set to 2. W is a matrix as the Eq. (5) shows.

$$W = \begin{pmatrix} \omega_1 & \cdots\cdots & 0 \\ \vdots & \omega_2 \cdots & \vdots \\ \vdots & \vdots \quad \ddots & \vdots \\ 0 & \cdots\cdots & \omega_n \end{pmatrix} \tag{5}$$

In addition, θ is also ensure that LWR has the minimum loss function at the predicted q, and the loss function of the LWR algorithm is shown as follows:

$$J(\theta) = \frac{\sum_{i=1}^{n} \omega_i \left[h_\theta\left(p^i\right) - t^i\right]^2}{2} = \frac{(X\theta - Y)^T W(X\theta - Y)}{2} \tag{6}$$

Then the regression parameter θ can be calculated using the least square method with a prediction point corresponding to a parameter θ. The final calculated θ is substituted into the Eq. (1) and then the execution time of the corresponding progress is predicted, as shown in Eq. (7) and (8).

$$\frac{\partial J(\theta)}{\partial \theta} = X^T WX\theta - X^T WY = 0 \tag{7}$$

$$\theta = \left(X^T WX\right)^{-1} X^T WY \tag{8}$$

The target of LWR is to find θ that minimizes for present prediction, during which the most important process is to compute the weight function, which can be obtained in two steps:

Step 1: Distance Calculation. The local region is firstly determined using Euler distance when predicting the value of the local point, as described in Eq. (9).

$$d = \sqrt{\sum_{i=1}^{n} \left(p^{(q)} - p^{(i)}\right)^2} \tag{9}$$

Step 2: Weight Calculation. The calculation of the weight function depends on the distance d. The greater the distance from the predicted point, the smaller the weight will be assigned. Use the Gaussian kernel function in (10), γ can controls the rate at which the weight decreases with distance. Set to 0.08 in this paper.

$$\omega(d) = \exp\left(-\frac{d^2}{2\gamma^2}\right) = \exp\left(-\frac{\sum_{i=1}^{n} \left(p^{(q)} - p^{(i)}\right)^2}{2\gamma^2}\right) \tag{10}$$

According to the consumption and benefits of launching or non-launching a backup task in the cluster, we can compute the profits of launching or non- launching the backup task to the cluster, the profits of launching speculative execution or not can be obtained as the following Equations.

$$profit_{backup} = \alpha \times \left(t_{rem} - t_{backup}\right) - 2 \times \beta \times t_{backup} \tag{11}$$

$$profit_{not\_backup} = -\beta \times t_{backup} \tag{12}$$

α and β represent the weight of benefit and the cluster cost. When satisfying the following formula, the identified Straggler backup task will be launched so that it can reach the maximum efficiency.

$$profit_{backup} > profit_{not\_backup} \Leftrightarrow \frac{t_{rem}}{t_{backup}} > \frac{\alpha + 2\beta}{\alpha + \beta} \tag{13}$$

Here we let ζ replace β/α, then the above Equation can be simplified as follows.

$$profit_{backup} > profit_{not\_backup} \Leftrightarrow \frac{t_{rem}}{t_{backup}} > \frac{1 + 2\zeta}{1 + \zeta} \tag{14}$$

$$\zeta = load\_factor = \frac{num_{pending\_tasks}}{mum_{free\_slots}} \tag{15}$$

Where t_{backup} is the running time of a backup task, ς is the load factor of the cluster, which is the ratio of the number of pending tasks to the number of the free containers in the cluster.

4 Results and Evaluation

In this section, we first test the performance of our model based on linear predictions and actual values. After that, the LWR-SE strategy is evaluated compared to Hadoop-None, LATE, and MCP in a heterogeneous cloud environment in three different scenarios.

4.1 Experimental Environment Preparation

We use 64-bit Ubuntu Server to be our operating system and our experimental platform is Hadoop-2.6.0. There are eight virtual nodes in the Hadoop cluster and each server is consist of four Intel® Xeon® CPU, 288 GB memory in total and up to 10 TB hard drive. In Table 3, it shows some detail information about each node. In the framework of Hadoop, it is common to use the datasets such as Wordcount and Sort as the experimental workloads. They are available on the Purdue MapReduce Benchmarks Suite (Table 2).

Table 2. 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 LWR Model

Data Prediction of LWR. The prediction results using the LWR model in the Wordcount and the Sort datasets are depicted in Fig. 3 and Fig. 4 respectively, where the red line represents prediction error rates. It can be depicted that the predictive accuracy of the LWR model much outperforms the linear regression, especially while the progress reaches 80% and over. RMSE is used to evaluate the accuracy of the prediction, and the calculation Equation is as follows.

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

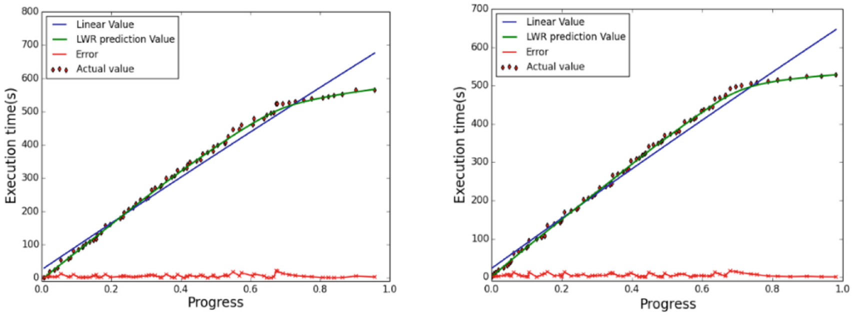

Fig. 3. Comparison of LWR and Linear Regression during running a Wordcount task

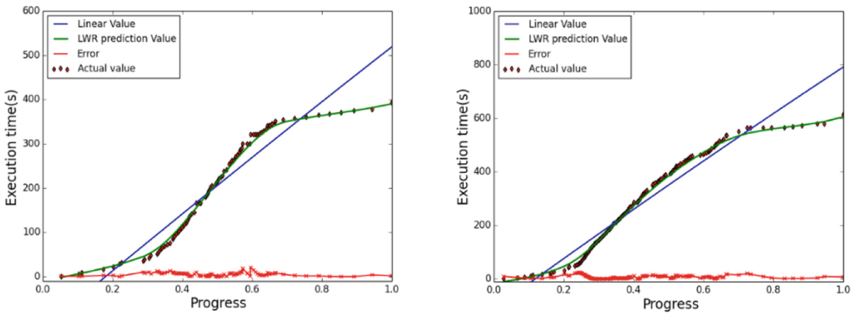

Fig. 4. Comparison of LWR and Linear Regression during running a Sort task

Where p is actual value and p_i represents the prediction value. Table 3 and Table 4 show the RMSE results of fifteen datasets/tasks, which are randomly selected from the Wordcount and Sort tasks. The average prediction RMSE of Wordcount and Sort are 1.56 and 1.75. This happens due to some unusual large values, which are mainly caused by resource contention and the non-data locality in the copy phrase during the Reduce process. If the outliers are ignored, we can find that average prediction RMSE of Wordcount and Sort drops to 0.91 and 0.86.

Table 3. RMSE of LWR for wordcount workloads

	Task 1	Task 2	Task 3	Task 4	Task 5
RMSE(s)	0.89	1.01	0.94	0.48	0.67
	Task 6	Task 7	Task 8	Task 9	Task 10
RMSE(s)	0.77	0.85	0.61	0.84	1.15
	Task 11	Task 12	Task 13	Task 14	Task 15
RMSE(s)	10.55	1.09	1.74	1.1	0.65

Table 4. RMSE of LWR for sort workloads

	Task 1	Task 2	Task 3	Task 4	Task 5
RMSE(s)	0.98	0.97	1.31	1.03	1.13
	Task 6	Task 7	Task 8	Task 9	Task 10
RMSE(s)	0.16	0.63	0.91	14.2	0.05
	Task 11	Task 12	Task 13	Task 14	Task 15
RMSE(s)	0.7	0.96	0.85	1.14	1.34

4.3 Evaluate the Performance of the LWR-SE Strategy in Heterogeneous Situations

Three different kinds of cluster workload scenarios are configured to evaluate the performance of the LWR-SE, they are Normal Load Scenario, Busy Load Scenario, and Busy Load with Data Skew Scenario. In addition, the final results are shown as the best, worse and average outcomes of each strategy.

Performance of the LWR-SE Strategy in a Busy Load Scenario. Abusy load scenario provides the cluster with limited resources to supply additional replication. It therefore is more necessary to ensure the accuracy of speculative execution, since low accuracy can cause the cluster resources to be irrationally occupied and consequently slow down the performance of the whole cluster. The busy load scenario was configured by running other computing-intensive and/or IO-intensive tasks simultaneously. Wordcount and Sort jobs were set up to submit every 150 s (Fig. 5).

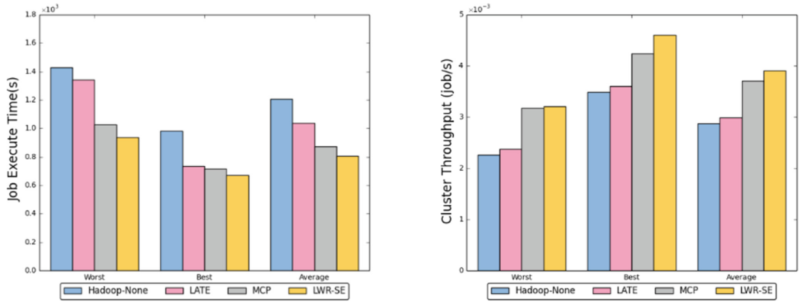

Fig. 5. Performance of strategies in Wordcount jobs in the busy load scenario

As can be seen in Fig. 6, the LWR-SE also fits well when running Sort jobs in the busy load scenario. in terms of the JET, on average cases, LWR-SE completed 9.7% earlier than MCP, 24.9% earlier than LATE and 30.6% earlier than Hadoop-None. When considering CT, the cluster throughput of LWR-SE increased by 9.3% compared with MCP and 36.1% over LATE.

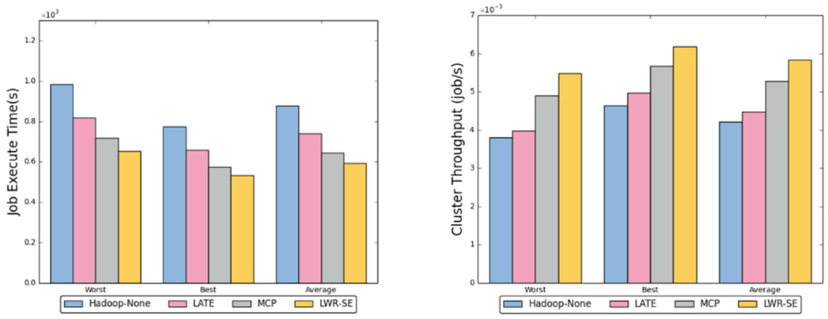

Fig. 6. Performance of strategies in the sorting job in the busy load scenario

5 Conclusion

In this paper, we propose a strategy named LWR-SE based on the relationship between tasks and job execution schedule, which can obtain higher local prediction accuracy and can guarantee the cloud system. Maximize benefits. The experimental results show that it is superior to MCP, LATE and Hadoop-None.

Acknowledgement. This work has received funding from 5150 Spring Specialists (05492018012, 05762018039), Major Program of the National Social Science Fund of China (Grant No. 17ZDA092), 333 High-Level Talent Cultivation Project of Jiangsu Province (BRA2018332), Royal Society of Edinburgh, UK and China Natural Science Foundation Council (RSE Reference: 62967_Liu_2018_2) under their Joint International Projects funding scheme and basic Research Programs (Natural Science Foundation) of Jiangsu Province (BK20191398).

References

1. Vaquero, L.F., Rodero, L., Caceres, J.: A break in the clouds: towards a cloud definition. Acm Sigcomm Comput. Commun. Rev. **39**(1), 50–55 (2008)
2. Iqbal, M.H., Soomro, T.R.: Big data analysis: apache storm perspective. Int. J. Comput. Trends Technol. **19**(1), 9–14 (2015)
3. Zaharia, M., Chowdhury, M., Franklin, M.J.: Spark: cluster computing with working sets. In: Proceedings USENIX Conference on Hot Topics in Cloud Computing, pp. 1765–1773. Springer, Heidelberg (2010)
4. Li, Z., Shen, H., Ligon, W.: An exploration of designing a hybrid scale-up/out hadoop architecture based on performance measurements. IEEE Trans. Parallel Distrib. Syst **28**(2), 386–400 (2017)
5. Gunarathne, T., Wu, T.L., Qiu, J.: MapReduce in the clouds for science. In: Proceedings Second International Conference on Cloud computing, pp. 565–572 (2010)
6. Dean, J., Ghemawa, S.: MapReduce: simplified data processing on large clusters. In: Proceedings OSDI, pp. 107–113 (2004)
7. Liu, Q., Cai, W., Jin, D.: Estimation accuracy on execution time of run-time tasks in a heterogeneous distributed environment. Sensors **16**(9), 1386 (2016)
8. Xu, H., Lau, W.C.: Optimization for speculative execution in big data processing clusters. IEEE Trans. Parallel Distrib. Syst. **28**(2), 530–545 (2017)
9. Xu, H., Lau, W.C.: Optimization for speculative execution in a mapreduce-like cluster. In: Proceedings IEEE Conference on Computer Communications (INFOCOM), pp. 1071–1079 (2015)
10. Sanchez, R., Almenares, F., Arias, P.: Enhancing privacy and dynamic federation in IdM for consumer cloud computing. IEEE Trans. Consum. Electron. **58**(1), 95–103 (2012)
11. Cabarcos, P.A., Mendoza, F.A., Guerrero, R.S.: SuSSo: seamless and ubiquitous single sign-on for cloud service continuity across devices. IEEE Trans. Consum. Electron. **58**(4), 1425–1433 (2012)
12. Abolfazli, S., Sanaei, Z., Alizadeh, M.: An experimental analysis on cloud-based mobile augmentation in mobile cloud computing. IEEE Trans. Consum. Electron. **58**(1), 146–154 (2014)
13. Fu, Z., Sun, X., Linge, N.: Achieving effective cloud search services: multi-keyword ranked search over encrypted cloud data supporting synonym query. IEEE Trans. Consum. Electron. **60**(1), 164–172 (2014)
14. Eom, B., Lee, C., Lee, H.: An adaptive remote display scheme to deliver mobile cloud services. IEEE Trans. Consum. Electron. **60**(3), 540–547 (2014)
15. Xu, X., Xue, Y., Yuan, Y.: An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles. Fut. Gener. Comput. Syst. **96**(1), 89–100 (2019)
16. Lee, Y.: An integrated cloud-based smart home management system with community hierarchy. IEEE Trans. Consum. Electron. **62**(1), 1–9 (2016)
17. Liu, Q., Cai, W., Shen, J.: A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment. Secur. Commun. Netw. **7**(17), 4002–4012 (2016)
18. Liu, Q., Cai, W., Shen, J.: An adaptive approach to better load balancing in a consumer-centric cloud environment. IEEE Trans. Consum. Electron. **62**(3), 243–250 (2016)
19. Huang, X., Zhang, L., Li, R.: Novel heuristic speculative execution strategies in heterogeneous distributed environments. Comput. Electric. Eng. **50**, 166–179 (2015)
20. Chen, Q., Liu, C., Xiao, Z.: Improving MapReduce performance using smart speculative execution strategy. IEEE Trans. Comput. **63**(4), 954–967 (2014)

21. Wu, H., Li, K., Tang, Z.: A Heuristic speculative execution strategy in heterogeneous distributed environments. In: Proceedings Sixth International symposium on Parallel Architectures, Algorithms and Programming (PAAP), pp. 268–273 (2014)
22. Liu, Q., Cai, W., Shen, J.: A smart strategy for speculative execution based on hardware resource in a heterogeneous distributed environment. Int. J. Grid Distrib. Comput. **9**(1), 203–214 (2015)
23. Wang, Y., Lu, W., Lou, R.: Improving MapReduce performance with partial speculative execution. J. Grid Comput. **13**(1), 587–604 (2015)
24. Li, Y., Yang, Q., Lai, S.: A new speculative execution algorithm based on C4.5 decision tree for hadoop. In: Proceedings the International Conference of Young Computer Scientists, Engineers and Educators (ICYCSEE 2015), pp. 284–291 (2015)
25. Tang, S., Lee, B., He, B.: DynamicMR: a dynamic slot allocation optimization framework for MapReduce clusters. IEEE Trans. Cloud Comput. **2**(3), 333–347 (2014)
26. Yang, S., Chen, Y.: Design adaptive task allocation scheduler to improve MapReduce performance in heterogeneous clouds. J. Netw. Comput. Appl. **57**(1), 61–70 (2015)
27. Liu, Q., Chen, F., Chen, F.: Home appliances classification based on multi-feature using ELM. Int. J. Sensor Netw. **28**(1), 34–42 (2018)
28. Xu, X., Li, Y., Huang, T.: An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks. J. Netw. Comput. Appl. **133**(1), 75–85 (2019)