

Near-Data Prediction Based Speculative Optimization in a Distribution Environment

Qi Liu · Xueyan Wu · Xiaodong Liu ·
Yonghong Zhang · Yuemei Hu

Received: date / Accepted: date

Abstract Hadoop is an open source from Apache with a distributed file system and MapReduce distributed computing framework. The current Apache 2.0 license agreement supports on-demand payment by consumers for cloud platform services, helping users leverage their respective different hardware to provides cloud services. In cloud-based environment, there is a need to balance the resource requirements of workloads, optimize load performance, and the cloud compute costs to manage. When the processing power of clustered machines varies widely, such as when hardware is aging or overloaded, Hadoop offers a speculative execution (SE) optimization strategy, by monitoring task progress in real time, in the starting identical backup tasks on different nodes when multiple tasks under a job are not running at the same speed, providing the first to go. The completed calculations maintain the overall progress of the job. At present, the SE strategy's incorrect selection of backup nodes and resource constraints may result in poor Hadoop performance, and subsequent

Q. Liu

School of Electrical Engineering University of Jinan, Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET), Nanjing University of Information Science & Technology, Nanjing, 210044, China

X. Wu

School of Computer and Software Nanjing University of Information Science & Technology, Nanjing, 210044, China

X. Liu

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

Y. Zhang

School of Automation, Nanjing University of Information Science & Technology, Nanjing, Jiangsu 210044, China

Y. Hu

Network Engineering Department, Information Science and Technology School, QuFu Normal University (Rizhao Campus), ShanDong Province, Rizhao, China
E-mail: yuemeihu@qfnu.edu.cn

tasks cannot be completed execution and other problems. This paper proposes an SE optimization strategy based on near data prediction, which analyzes the prediction of real-time task execution information to predict the required running time, select backup nodes based on actual requirements and approximate data to make the SE strategy achieve the best performance. Experiments prove that in a heterogeneous Hadoop environment, the optimization strategy can effectively improve the effectiveness and accuracy of various tasks and enhance the performance of cloud computing. Platform performance can benefit consumers better than before.

Keywords Distributed Systems · Hadoop · Speculative Execution · Locally Weighted Regression · Near Data Prediction

1 Introduction

With the development of IoT transmission efficiency, in recent years, the user-centric private cloud server due to the flexibility, high performance and stability of the features have also shown explosive growth [26], and their underlying architecture is roughly the same. It is equipped with different physical structures and features depending on its application [10]. Users can elastically expand and pay for according to their needs[5]. Existing cloud computing platforms include Spark, Hadoop, and Apache Storm[15].

In the cloud computing framework, Hadoop due to high reliability, scalability, fault tolerance and efficiency in distributed data, search, storage and computing fields are widely used [13]. To address the big data storage and analysis failures caused by the master/slave cluster architecture model, there are a variety of distributed cluster strategies resolve cluster crashes against failures [3]. However, Hadoop still suffers from the resource allocation problem of job scheduling [7].

In a distributed clustered environment on a cloud platform, it is easy for a program error, unbalanced load, or uneven distribution of resources to cause the same multiple tasks run under job to improve the speed is inconsistent, slowing down the overall execution of the progress of JobTracker [17]. It is difficult to predict the total time required to complete each task, how to coordinate tasks and JobTracker becomes a pressing problem. The most effective optimization strategy is Speculative Execution (SE), which trades space for time [33]. The first results are used to improve the computation speed, but this method consumes more resources, and the problem that needs to be solved is shortage of cluster resources. However, this approach will consume more resources and needs to address the problem of proper allocation of time and resources for large job calculations in case of cluster resource shortage [32].

We propose a locally weighted prediction algorithm called LWR-SE in the taskcollecting task information in real time during execution and using a maximum cost consumption model combined with appropriate task strategies for back-up node selection to improve the resource allocation efficiency for optimal strategy. In section 2, we cite current research on user-centric cloud computing

environments and strategies for Hadoop-based performance optimization. Our proposed LWR-SE method is presented in section 3. In section 4, we designed multiple sets of comparative experiments to verify the usability of the algorithm. Section 5 conclude the work of this paper and proposes future work to be done.

2 Related Work

2.1 User-Centric Cloud System

The acceleration of the development process in both electronics and traditional industries has led to an increasing number of research directions towards user-centric cloud services. Sanchez's team researched IDM architecture based on user privacy protection, based on user-centric cloud the dynamic and heterogeneous nature of the platform proposes dynamic federated identity management to provide security for the consumer cloud computing paradigm [24]. Kalyampudi et al. present sending estimates to validate load balancing methods to optimize service continuity issues between different devices [11]. By comparing the analysis of the user-centric cloud computing domain, Abolfazli's team proposes a cloud-based mobile enhancement model, optimizing computing power [1]. Fu's team proposes a multi-keyword ranked search model for the cloud computing paradigm of the consumer-centric per-use billing, in which the encrypted Support for synonymous queries in the cloud [4]. Xu et al. for intensive data a network topology and non-genetic algorithm based on minimizing manufacturing cycles and optimizing load balancing [37]. Hamdanis team proposed a load balancing model based on cloud server weights by fuzzy logic expresses the weights of different nodes to improve load balancing performance of the cloud platform and improve the overall performance of user-centric clouds [8]. High-speed and stable wireless communication has also greatly affected high-tech fields such as medical robots [28]. Due to the large differences in the types of big data in cloud platforms, Xu's team proposed a data-control-based offloading method called COM that restricts the dynamic scheduling of tasks by confirm the data offload model [38]. For data privacy issues, a recommended domain algorithm combining QoS and LSH is proposed, and it effectively solves the problem of protecting sensitive information [23]. A mobile crowdsourcing technology [21] applied to dynamic environments and a distributed privacy service that introduces local sensitive hashing (LSH) technology are proposed to ensure privacy in multi-dimensional data [6]. Xu et al. proposed an EN resource optimization scheme that balances the load and protects privacy [36]. An architecture for HAR is proposed for de-noising, normalizing, and segmenting data signals to extract feature vectors [29]. Lee's team has proposed a consumer-centric, cloud-based smart home management system that uses building automation devices and home networks deepen the integration of communities and environments and optimize modules for energy efficiency, scenarios and safety information [12]. Cao et al. use SDN-based UARP ap-

proach to reduce energy consumption and address execution uncertainty [40]. A cross-platform SR-Amplified algorithm based on LSH can maintain a balance between efficiency and users [2], a cross-cloud platform service based on collaborative filtering can ensure the scalability of choice decisions while ensuring data accuracy [41]. An energy-saving QoS-aware VM scheduling method called QVMS can effectively solve the VM migration and overuse of resources caused by explosive cloud expansion problems [22].

2.2 Fault Tolerance in Hadoop

Currently there are three main types of failures in Hadoop: DN node failures, network failures, and data corruptions, and fault tolerance mechanisms through the It is based on the MapReduce scheduling strategy and speculation, and can be used to restore a failed task, effectively shorten the execution time of tasks and ensure system stability.. The main approaches are MapReduce-based task scheduling strategy and speculation. execution (SE) strategy[18].

Current jobs for cloud clusters can be executed quickly, but efficient use of computing resources and fault tolerance remains a challenge. Liu's team proposed a dynamic SE strategy based on real-time cluster management for consumer-centric heterogeneous cloud environments, combined with a multicomponent, multicomponent, multicomponent, and multicomponent cloud management. Objective optimization algorithms improve the efficiency of time and space [19]. Xin et al. Man proposes an optimized SE strategy for the slow-running straggler task in Hadoop by using the Linear relationship model as a dynamic load-aware strategy (ERUL) and time and scaling cost (exMCP) for task information make improvements to overcome the misleading LATE algorithm, reduce job runtime and improve cloud computing efficiency [9].

The MCP(Maximize Cost Performanee) strategy is a strategy that is implemented in the It is based on LATE and is improved by a phased approach using EWMA prediction algorithm predicts the execution speed of each stage and calculates the task's Remaining completion time to accurately select dropped tasks and build a cost-benefit model to improve system performance at minimal cost. Wu's team proposes an MCP-based optimization algorithm that considers the impact of dynamic load on the system, improves the accuracy of task remaining time estimation, and reduces job runtime. [31]. An LBBDD-based optimization framework to solve multi-node periodic tasks in asynchronous buffer communication [43], a schedulability analysis integrating PTS and MCS can reduce program space requirements [44]. Different optimization strategies have been proposed, and Liu's team proposed a node ranking strategy based on user hardware performance to optimize tasks execution method [16]. Wang's team proposed an SE-based task checkpoint model PSE to eliminate duplicate data processing costs and improve MapReduce performance [30]. Li's team proposed a decision tree based speculative execution strategy SECDDT[14], which calculates the time required to complete a task through a decision tree, improves the accuracy of time estimates. Tang's team uses the DynamicMR model to pre-

determine slot assignment constraint task mappings, solve data localization problems, balance job performance, and optimize resource allocation maps. [25]. A two-stage offloading optimization strategy UTO was proposed to maximize ECU resource utilization and minimize implementation time costs [34]. An strategy named ATAS can improve the Hadoops expandability by increasing the accuracy of estimating the time required for the task optimize user-centric cloud data scalability by improving the accuracy of backup job time estimates of backing-up tasks [42]. An improved FasterRCNN framework can improve the accuracy of data detection and guarantee real-time transmission [27]. The user-centric smart home is also able to extract features by optimizing the data sampling model and optimizing the cloud using the cloud adaptive distribution algorithm [20]. Xu’s team proposed an offloading model based on blockchain technology to ensure data integrity by using blockchain technology in edge computing [39]. Due to the time-consuming nature of backbone data propagation in cloud-based data centers, users who are sensitive to latency performance have proposed edge computing nodes. The team at Xu is targeting alternative solutions to the Edge Computing Nodes (ECNS). Dense tasks are deployed at edge nodes using a non-dominant ordering genetic algorithm to shorten tasks using decision makers and weight selection Unload time for multi-objective optimization [35].

Speculative execution strategy is currently widely used for fast backups of tasks. However, there are still great difficulties in accurately identifying stragglers in tasks and maintaining efficient processing of local tasks. In general, how to achieve the highest local prediction accuracy while ensuring the balance of tasks and job execution schedule in the cloud system is a pressing issue.

3 Model and Algorithm

3.1 Identification of Straggler Candidates

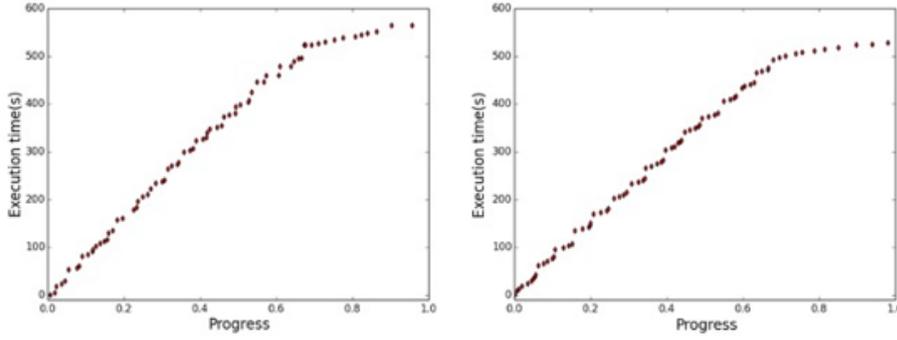
Collection of Features for Real-time Tasks. To confirm the current stragglers by collecting information such as real-time task status and MapTask report. It also collects raw data from HDFS, task progress and timestamps to confirm features and make predictions. To optimize the time complexity, the data pairs are converted to (progress, time of execution):

Figure 1 and 2 show the execution duration information when running the Wordcount dataset and Sort dataset examples. The data is collected in a Hadoop cluster environment.

A Learning Model Based on Local Weighting Algorithms. As shown in Tab.1, the trends in the execution information collected by the running task cannot be fitted using a linear model, and we designed a model based on local non-parametric learning model of the weighted regression algorithm, defining the contribution of the regression coefficients by weights, designing the kernel parameter matrix to make the objective function minimize and op-

Table 1 Algorithm for Task Data Collection

Collection of Features for Real-time Tasks : Data Collection
Input:
MapTask report (MR), Job status (JS), ID of task (IT), Task context (TC), Progress of running task (P), Running task attempt (RT), 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
End For
End Data Collection

**Fig. 1** Implementation data collected by running the Wordcount dataset

minimize the linear model under fit problem. Input dataset is $D = \{(p_i, t_i) | i = 1, 2, \dots, n\}$, the prediction result model is:

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

The execution of different tasks is defined as n , as the number of training set samples, and each set needs to be relearned for each training session, retain the training set sample. p is an $n + 1$ dimensional data representing the current progress of the task. t is the time of execution. θ is used as the non-linear model regression parameters are used to minimize the squared difference between real

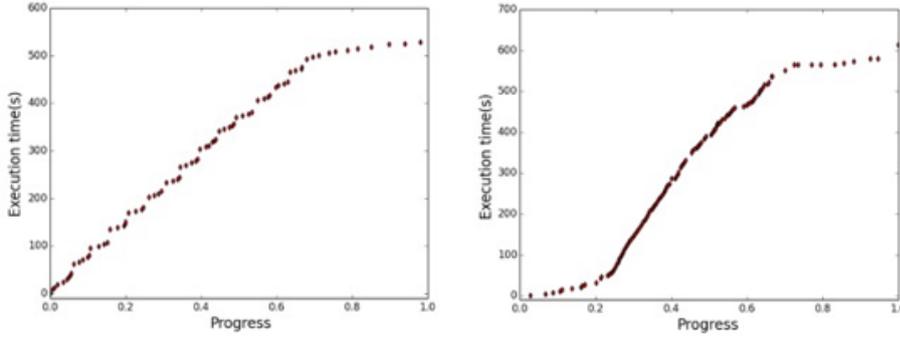


Fig. 2 Implementation data collected by running the Sort dataset

data and predicted data so that the training results satisfy the prediction as shown in Equation(2) and (3).

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

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

Where E represents the error, (p^i, t^i) is the number of sample, ω_i is the weight of the local prediction region near the point to be measured, the size of which was determined by prediction point. We optimized it by transforming it into the kernel function matrix, as shown in Equation (4).

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

X is used as a matrix, where the training data is m rows. p_0, p_1 till p_m, p_n being set to 2. W is a matrix as the Equation (5) shows.

$$W = \begin{pmatrix} \omega_1 & L & L & 0 \\ M & \omega_2 & L & M \\ M & M & O & M \\ 0 & L & L & \omega_n \end{pmatrix} \quad (5)$$

The loss function with θ as the weight parameter updating the equation so that LWR-SE minimizes the loss function when predicting the value of q .

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

Afterwards, the least squares method is used as the updating equation for the regression parameter θ to calculate the parameter θ corresponding to the point to be measured, and finally bring the parameter values into Eq. (1) to solve for the predicted execution time, as in Equation (7) and (8) shown..

$$\frac{\partial J(\theta)}{\partial \theta} = X^T W X \theta - X^T W Y = 0 \quad (7)$$

$$\theta = (X^T W X)^{-1} X^T W Y \quad (8)$$

The goal of the LWR-SE algorithm is find the θ value that minimizes the loss function, so the weights need to be taking full account of the distances between the predicted points and other points, the weight function is calculated as follows:

Step 1: Calculating Distance. Calculate the distance between the point to be measured and nearby points by Euler distance, as shown in equation (9):

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

Step 2: Calculating Weight. Where $p^{(q)}$ is the prediction point, $p^{(i)}$ is all the points near the prediction point, and γ determines the size of the weights of nearby points. The larger the value of γ , the smaller the effect of increasing distance on decreasing weights, γ is set to 0.08 in experiments. Equation (10) uses the Gaussian nuclear function:

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

Among the cluster's tasks, calculate the resource consumption and benefits to the cluster from the choice between starting a backup task and not starting a backup task via equation (11):

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

$$profit_{not_backup} = -\beta \times t_{backup} \quad (12)$$

In equation (11), (12), α denotes the benefit weight of starting the backup task, and β represents the cost of the required cluster. When the following equation is satisfied, the identified backup task of straggler that maximizes efficiency is automatically started.

$$profit_{backup} > profit_{not_backup} \Leftrightarrow \frac{t_{rem}}{t_{backup}} > \frac{\alpha + 2\beta}{\alpha + \beta} \quad (13)$$

Here we let ζ replace β/α , equation(13) can be simplified as follows:

$$profit_{backup} > profit_{not_backup} \Leftrightarrow \frac{t_{rem}}{t_{backup}} > \frac{1 + 2\zeta}{1 + \zeta} \quad (14)$$

$$\zeta = load_{factor} = \frac{num_{pending_tasks}}{num_{free_slots}} \quad (15)$$

In the above equation, t_{backup} represents the runtime required for the straggler backup task, ζ represents task queue ratio to idle resources in the cluster as the cluster's current load factor.

4 Results and Evaluation

In this section, we design performance tests with the aforementioned LWR-SE algorithm in heterogeneous cloud with three settings based on the linear prediction and actual value comparison of LWR-SE optimization strategies with LATE, MCP and Hadoop-None conduct comparative testing and evaluation.

4.1 Experimental Operating Environment

Experiments were conducted using Hadoop-2.6.0 with 64-bit Ubuntu Server operating system to build a server with 8 Hadoop cluster of 10 virtual nodes, where each server consists of a 10TB hard disk, 288GB of memory and four Intel Xeon CPU composition. The details of eight virtual nodes are shown in Table 2. The experimental load uses the Wordcount and Sort datasets under the Purdue MapReduce benchmark suite in the Hadoop framework.

Table 2 The Details of Eight Virtual Nodes

ID	Memory	Core Processors
No. 1	10GB	8
No. 2	8GB	4
No. 3	8GB	1
No. 4	8GB	8
No. 5	4GB	8
No. 6	4GB	4
No. 7	18GB	4
No. 8	12GB	8

4.2 LWR-SE Algorithm Performance Evaluation

Data Prediction Based on LWR-SE Algorithm. As shown in figure 3 and 4, the predictions of the LWR-SE model when using the Wordcount data set and Sort data set, the results are much more accurate than linear regression, and the red line in the figure represents the prediction error rate. The LWR-SE model also provides a better fit to the data when the task progress exceeds 80%. RMES was used experimentally as an evaluation mechanism for prediction accuracy, as shown in equation (16):

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (p - p_i)^2}{n}} \quad (16)$$

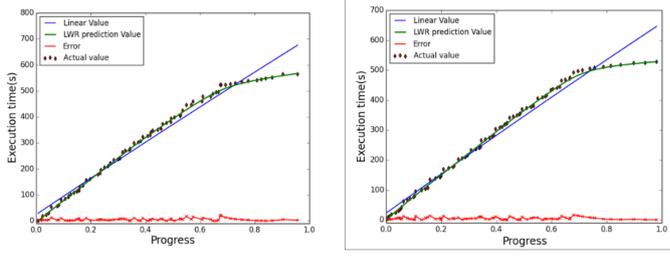


Fig. 3 LWR-SE model and linear regression model were run using the Wordcount dataset

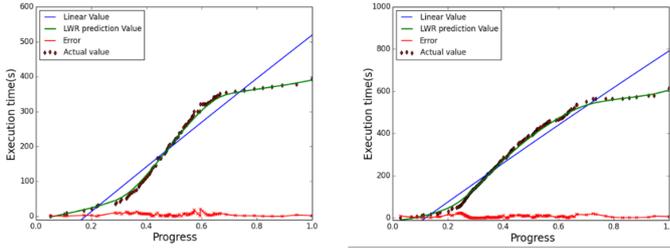


Fig. 4 LWR-SE model and linear regression model were run using the Sort dataset

Where p_i is prediction and p denotes the real value of the p-point. Tables 3 and 4 show the RMSE values calculated for 15 different tasks, which are derived from Wordcount and Sort dataset were randomly selected. Experiments showed RMSE values of 1.56 and 1.75 for the Wordcount and Sort datasets. This is due to contention for finite resources in the Reduce process and the non-data of copying phrases in the MapReduce. The localization produces some unusually large outliers. If these anomalies are removed, the average predicted RMSE for the two is aggregates can be reduced to 0.91 and 0.86.

Table 3 RMSE values for the LWR-SE algorithm based on the Wordcount dataset.

	Task 01	Task 02	Task 03	Task 04	Task 05
RMSE(s)	0.89	1.01	0.94	0.48	0.67
	Task 06	Task 07	Task 08	Task 09	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

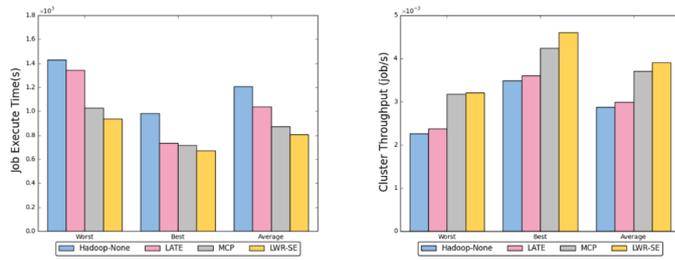


Fig. 5 The results of running Wordcount jobs under busy load condition.

Table 4 RMSE values for the LWR-SE algorithm based on the Sort dataset.

	Task 01	Task 02	Task 03	Task 04	Task 05
RMSE(s)	0.98	0.97	1.31	1.03	1.13
	Task 06	Task 07	Task 08	Task 09	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 Evaluation of LWR-SE

We designed the effect evaluation of the LWR-SE algorithm for three different load scenarios under clustered work: normal condition, busy condition and data skew busy condition. The experimental results are divided into three types: best, average and worst.

Since load scenarios at busy times provide very limited resources to the cluster, tasks cannot have additional backups, and to ensure inferred execution. To ensure cluster performance and avoid low accuracy of data due to unreasonable usage of cluster resources, we design a simultaneous computation-intensive and I/O intensive tasks are configured as busy load condition. The two dataset tasks are set to commit every 150 seconds.

In figure (5) and (6), when running the Sort job under a busy load condition, the LWR- The SE has better performance, with the LWR-SE showing a 9.7 % improvement in JET average compared to the MCP, LWR-SE comparison with LATE shows a 24.9% increase in JET average. LWR-SE and Hadoop-None, the comparison shows a 30.6 % increase in JET average. When CT is considered, LWR-SE increases cluster throughput by 9.3% over MCP and by 36.1% over LATE.

5 Conclusion

Based on the relationship between Hadoop tasks and job progress, this paper proposes the LWR-SE optimization strategy, which has higher local prediction accuracy and ensures the maximum benefit of the user-centric cloud

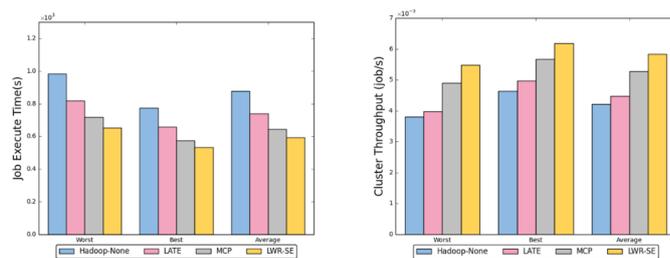


Fig. 6 The results of running Sort jobs under busy load condition.

system. Testing of model performance through linear prediction and comparison with different strategies in three environments with heterogeneous clouds. Experimental results show that LWR-SE works better than LATE, MCP and Hadoop-None.

Acknowledgements This work has received funding from National Natural Science Foundation of China (No. 41911530242, 41975142), 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. Abolfazli S, Sanaei Z, Alizadeh M, Gani A, Xia F (2014) An experimental analysis on cloud-based mobile augmentation in mobile cloud computing. *IEEE Transactions on Consumer Electronics* 60(1):146–154
2. Chi X, Yan C, Wang H, Rafique W, Qi L (2020) Amplified locality-sensitive hashing-based recommender systems with privacy protection. *Concurrency and Computation: Practice and Experience* p e5681
3. Dean J, Ghemawat S (2008) Mapreduce: simplified data processing on large clusters. *Communications of the ACM* 51(1):107–113
4. Fu Z, Sun X, Linge N, Zhou L (2014) Achieving effective cloud search services: multi-keyword ranked search over encrypted cloud data supporting synonym query. *IEEE Transactions on Consumer Electronics* 60(1):164–172
5. Giselsson P, Boyd S (2017) Linear convergence and metric selection for douglas-rachford splitting and admm. *IEEE Transactions on Automatic Control* 62(2):532–544
6. Gong W, Qi L, Xu Y (2018) Privacy-aware multidimensional mobile service quality prediction and recommendation in distributed fog environment. *Wireless Communications and Mobile Computing* 2018
7. Gu Z, Qiu M (2018) Introduction to the special issue on ?embedded artificial intelligence and smart computing?

8. Hamdani M, Aklouf Y, Bouarara HA (2019) Improved fuzzy load-balancing algorithm for cloud computing system. In: Proceedings of the 9th International Conference on Information Systems and Technologies, pp 1–4
9. Huang X, Zhang L, Li R, Wan L, Li K (2016) Novel heuristic speculative execution strategies in heterogeneous distributed environments. *Computers & Electrical Engineering* 50:166–179
10. Iqbal MH, Soomro TR (2015) Big data analysis: Apache storm perspective. *International journal of computer trends and technology* 19(1):9–14
11. Kalyampudi PL, Krishna PV, Kuppani S, Saritha V (2019) A work load prediction strategy for power optimization on cloud based data centre using deep machine learning. *Evolutionary Intelligence* pp 1–9
12. Lee YT, Hsiao WH, Huang CM, Seng-cho TC (2016) An integrated cloud-based smart home management system with community hierarchy. *IEEE Transactions on Consumer Electronics* 62(1):1–9
13. Li J, Liu Y, Pan J, Zhang P, Chen W, Wang L (2020) Map-balance-reduce: an improved parallel programming model for load balancing of mapreduce. *Future Generation Computer Systems* 105:993–1001
14. Li Y, Yang Q, Lai S, Li B (2015) A new speculative execution algorithm based on c4. 5 decision tree for hadoop. In: International Conference of Young Computer Scientists, Engineers and Educators, Springer, pp 284–291
15. Li Z, Shen H, Ligon W, Denton J (2016) An exploration of designing a hybrid scale-up/out hadoop architecture based on performance measurements. *IEEE Transactions on Parallel and Distributed Systems* 28(2):386–400
16. Liu Q, Cai W, Fu Z, Shen J, Linge N (2016) A smart strategy for speculative execution based on hardware resource in a heterogeneous distributed environment. *International Journal of Grid and Distributed Computing* 9(2):203–214
17. Liu Q, Cai W, Jin D, Shen J, Fu Z, Liu X, Linge N (2016) Estimation accuracy on execution time of run-time tasks in a heterogeneous distributed environment. *Sensors* 16(9):1386
18. Liu Q, Cai W, Shen J, Fu Z, Liu X, Linge N (2016) A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment. *Security and Communication Networks* 9(17):4002–4012
19. Liu Q, Cai W, Shen J, Liu X, Linge N (2016) An adaptive approach to better load balancing in a consumer-centric cloud environment. *IEEE Transactions on Consumer Electronics* 62(3):243–250
20. Liu Q, Chen F, Chen F, Wu Z, Liu X, Linge N (2018) Home appliances classification based on multi-feature using elm. *IJSNet* 28(1):34–42
21. Qi L, Dou W, Wang W, Li G, Yu H, Wan S (2018) Dynamic mobile crowd-sourcing selection for electricity load forecasting. *IEEE Access* 6:46926–46937

22. Qi L, Chen Y, Yuan Y, Fu S, Zhang X, Xu X (2019) A qos-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems. *World Wide Web* pp 1–23
23. Qi L, Zhang X, Li S, Wan S, Wen Y, Gong W (2020) Spatial-temporal data-driven service recommendation with privacy-preservation. *Information Sciences* 515:91–102
24. Sanchez R, Almenares F, Arias P, Diaz-Sanchez D, Marin A (2012) Enhancing privacy and dynamic federation in idm for consumer cloud computing. *IEEE Transactions on Consumer Electronics* 58(1):95–103
25. Tang S, Lee BS, He B (2014) Dynamicmr: A dynamic slot allocation optimization framework for mapreduce clusters. *IEEE Transactions on Cloud Computing* 2(3):333–347
26. Vaquero LM, Roderomerino L, Caceres J, Lindner M (2008) A break in the clouds: Towards a cloud definition. *Acm Sigcomm Computer Communication Review* 39(1):50–55
27. Wan S, Goudos S (2020) Faster r-cnn for multi-class fruit detection using a robotic vision system. *Computer Networks* 168:107036
28. Wan S, Gu Z, Ni Q (2019) Cognitive computing and wireless communications on the edge for healthcare service robots. *Computer Communications*
29. Wan S, Qi L, Xu X, Tong C, Gu Z (2019) Deep learning models for real-time human activity recognition with smartphones. *Mobile Networks and Applications* pp 1–13
30. Wang Y, Lu W, Lou R, Wei B (2015) Improving mapreduce performance with partial speculative execution. *Journal of grid computing* 13(4):587–604
31. Wu H, Li K, Tang Z, Zhang L (2014) A heuristic speculative execution strategy in heterogeneous distributed environments. In: 2014 Sixth International Symposium on Parallel Architectures, Algorithms and Programming, IEEE, pp 268–273
32. Xu H, Lau WC (2015) Optimization for speculative execution in a mapreduce-like cluster. In: 2015 IEEE Conference on Computer Communications (INFOCOM), IEEE, pp 1071–1079
33. Xu H, Lau WC (2016) Optimization for speculative execution in big data processing clusters. *IEEE Transactions on Parallel and Distributed Systems* 28(2):530–545
34. Xu X, He C, Xu Z, Qi L, Wan S, Bhuiyan MZA (2019) Joint optimization of offloading utility and privacy for edge computing enabled iot. *IEEE Internet of Things Journal*
35. Xu X, Li Y, Huang T, Xue Y, Peng K, Qi L, Dou W (2019) An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks. *Journal of Network and Computer Applications* 133:75–85
36. Xu X, Liu X, Xu Z, Wang C, Wan S, Yang X (2019) Joint optimization of resource utilization and load balance with privacy preservation for edge services in 5g networks. *Mobile Networks and Applications* pp 1–12

37. Xu X, Mo R, Dai F, Lin W, Wan S, Dou W (2019) Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud. *IEEE Transactions on Industrial Informatics*
38. Xu X, Xue Y, Qi L, Yuan Y, Zhang X, Umer T, Wan S (2019) An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles. *Future Generation Computer Systems* 96:89–100
39. Xu X, Zhang X, Gao H, Xue Y, Qi L, Dou W (2019) Become: Blockchain-enabled computation offloading for iot in mobile edge computing. *IEEE Transactions on Industrial Informatics*
40. Xu X, Cao H, Geng Q, Liu X, Dai F, Wang C (2020) Dynamic resource provisioning for workflow scheduling under uncertainty in edge computing environment. *Concurrency and Computation: Practice and Experience* pp 56–74
41. Xu Y, Qi L, Dou W, Yu J (2017) Privacy-preserving and scalable service recommendation based on simhash in a distributed cloud environment. *Complexity* 2017
42. Yang SJ, Chen YR (2015) Design adaptive task allocation scheduler to improve mapreduce performance in heterogeneous clouds. *Journal of Network and Computer Applications* 57:61–70
43. Zhang M, Zheng N, Li H, Gu Z (2018) A decomposition-based approach to optimization of ttp-based distributed embedded systems. *Journal of Systems Architecture* 91:53–61
44. Zhao Q, Gu Z, Zeng H, Zheng N (2018) Schedulability analysis and stack size minimization with preemption thresholds and mixed-criticality scheduling. *Journal of Systems Architecture* 83:57–74