An Adaptive Approach to Better Load Balancing in a Consumer-centric Cloud Environment

Qi Liu, *Member*, IEEE, Weidong Cai, Jian Shen, Xiaodong Liu, Nigel Linge, *Member*, IEEE

Abstract — *Pay-as-you-consume, as a new type of cloud computing paradigm, has become increasingly popular since a large number of cloud services are gradually opening up to consumers. It gives consumers a great convenience, where users no longer need to buy their hardware resources, but are confronted with how to deal effectively with data from the cloud. How to improve the performance of the cloud platform as a consumer-centric cloud computing model becomes a critical issue. Existing heterogeneous distributed computing systems provide efficient parallel and high fault tolerant and reliable services, due to its characteristics of managing large-scale clusters. Though the latest cloud computing cluster meets the need for faster job execution, more effective use of computing resources is still a challenge. Presently proposed methods concentrated on improving the execution time of incoming jobs, e.g., shortening the MapReduce (MR) time. In this paper, an adaptive scheme is offered to achieve time and space efficiency in a heterogeneous cloud environment. A dynamic speculative execution strategy on real-time management of cluster resources is presented to optimize the execution time of Map phase, and a prediction model is used for fast prediction of task execution time. Combing the prediction model with a multi-objective optimization algorithm, an adaptive solution to optimize the performance of space-time is obtained. Experimental results depict that the proposed scheme can allocate tasks evenly and improve work efficiency in a heterogeneous cluster[1].*

Index Terms —Pay-as-you-consume; MapReduce; Load balancing; Prediction model; K-ELM

[1] This work is supported by the NSFC (61300238, 61300237), Marie Currie Fellowship (701697-CAR-MSCA-IFEF-ST), Basic Research Programs (Natural Science Foundation) of Jiangsu Province (BK20131004), 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 and the PAPD fund.

Q. Liu is with the College of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, 210044, CHINA (e-mail: qi.liu@nuist.edu.cn).

W. Cai is with the Jiangsu Engineering Centre for Network Monitoring, Nanjing University of Information Science and Technology, Nanjing, 210044, CHINA (e-mail: caiweidongsuzhou@163.com).

S. Jian is with the Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET), Nanjing University of Information Science and Technology, Nanjing, 210044, CHINA (e-mail: s_shenjian@126.com).

X. Liu is with School of Computing, Edinburgh Napier University, 10 Colinton Road, Edinburgh EH10 5DT, UK (E-mail: x.liu@napier.ac.uk).

N. Linge is with the School of Computing, Science and Engineering, University of Salford, Salford, M5 4WT, UK (E-mail: n.linge@salford.ac.uk).

I. INTRODUCTION

Pay-as-you-consume, as one of the cloud consuming models, is becoming more popular for its benefits, e.g. a large number of convenient services, reducing the burden of storage and flexible data access, and minimizing the cost of the hardware and software [1-2]. Industrial consumers have already set up various cloud computing services. Cloud computing that has been seen as a successful commercial distributed system provides users with on-demand services by the reasonable allocation of resources [3-10].

MapReduce (MR) is a distributed programming model proposed by Google. Currently, more and more enterprises have applied MapReduce to process data. Apache provides an open source implementation version of the MR, which enables convenient and efficient big data processing, but also brings differences and complexity on resource requirements, data delivery deadlines, etc. Such diversity brings new challenges to job scheduling and workload management.

Irrational allocation of resources using current load scheduling strategies in cloud systems can lead to inefficient job execution and may waste more storage space. Therefore, optimization schemes have been proposed [10-19], but most of them are only focused on task execution time, whereas storage space is often neglected.

This paper proposes an approach in a cloud system to achieve load balancing on both space and time. A dynamic speculative execution policy is designed to reduce the running time at the map phase. Then, a prediction model set up on the kernel-based extreme learning machine (K-ELM) [20-23], called PMK-ELM, is proposed to estimate the possible execution duration and storage space of new tasks. Depending on the characteristics of the data, a modified algorithm called DNSGA-II is presented adapting to disperse variables based on NSGA-II [24]. A new algorithm combined with the K-ELM and DNSGA-II keeps all the nodes complete the task in a similar time and maintains a comparable ratio of hard disk space usage. Feasibility and performance of the scheme are verified in a practical Hadoop environment. As it is applied in a consumer-centric cloud computing platform, both consumers and service providers can benefit from the platform.

This paper combines five sections. Related work on load balancing is reviewed in Section II. The adaptive method to achieve load balancing at map and reduce phases is discussed in Section III. Results are presented and evaluated in Section IV with comparison of corresponding algorithms. Finally, Section V concludes the paper.

II. Related Work

A. Consumer-centric Cloud Services

A useful search service is presented in [3], in which encrypted cloud data is supported by multi-keyword ranked search. An IdM architecture was presented to enhance privacy and dynamic federation in a cloud [4]. Abolfazli et al. investigated the influence of different parameters to optimize the performance Mobile Argumentation based on cloud [5]. Based on cloud providers, a new middleware architecture was proposed to allow sessions to be transferred to another device [6]. A Program Recommendation system called PDPR system was implemented under a cloud environment. The proposed system can recommend the program to consumers by analyzing the viewing pattern [7]. Eom et al. presented an integrated smart home management system with community hierarchy based on cloud system [8]. A sharing cloud service was proposed [9], which provided an enhanced user authentication for home networking.

B. Load Balancing

In a heterogeneous environment, due to the different performance of each node, the node data are difficult to obtain a balanced load. [11] proposed a method to allocate more data to a node that has better performance. By monitoring running map tasks and reduce tasks, Hadoop ensures that all tasks running on each node can substantially complete. This method can detect a variety of load skew. However, complex implementation considerably changed Hadoop. In cloud systems, a general method for performance measurement and load efficiency has been tested and presented. For example, a prediction model based on SVM has been proposed in [12] to optimize the performance of a heterogeneous cluster. HAP is designed to control the distribution of the results produced by map tasks. SVM is then used for calculating the estimated data threshold. However, various division and merging would lead to extra time. Also, training phase of HAP consumes a lot of time. Matsunaga et al. proposed a novel method that can provide an accurate performance evaluation of cloud environments for distributed applications. Jing et al. proposed a prediction model based on classification and regression trees for forecasting the resource consumption of a MapReduce application [14].

Deployment efficiency on visualization has also been investigated. A general approach was introduced to estimate the resource requirements of applications running in a virtualized environment [15]. Dynamic resource demands have been studied. When starting a new VM instance, model based on adaptive resource provision was then presented for resource allocation in a cloud system [16].

Besides above methods, optimizing the speculative execution strategy in MapReduce has raised researchers' attention. A speculative execution strategy called Longest Approximate Time to End (LATE) algorithm was proposed [17]. But the running time of every stage is not stable, and the standard error used in LATE cannot represent all cases. MCP [18] was therefore proposed to solve the problems of LATE. Though it has optimized LATE a lot, average running time of nodes that utilized in MCP is unreasonable, due to the fact that the running time is largely dependent on the performance of some node.

Although many schemes have been put forward, achieving load balancing in a cloud system is still not well solved, especially in a heterogeneous one. Our previous work [19] has been issued in ICCE, however, in this paper, a novel speculative execution strategy, called dynamic strategy, is newly presented.

III. Approach to Load Balancing

A. Dynamic Speculative Execution Strategy

Speculative execution strategy is initially proposed by Google, which can be employed to backup tasks running at slow speed. However, the native strategy [18] in MRV2 suffers from low efficiency, an optimized strategy is, therefore, necessary. In dynamic strategy, real-time resources in the cluster are well considered before a backup task is launched. Resource in MRV2 is call container. The number of left tasks is marked as *N*, currently available containers as *C*.

Three states are defined, as follow:

(1) $N>C$

(2) $N=C$

(3) $N<C$

A speculator continuously detects which case current state is. As it is activated, dynamic strategy would be adopted according to different cases.

When there are not enough resources existing now, the speculator would kill the native task and starts a new task when the time saved by the new task reaches a half of the original remaining time. Because of the higher priority, the backup task would immediately start on the original node.

As case (2) that indicates resources on every node are equals the resource required. Following steps are taken:

1) Estimate remaining time of all running tasks based on the average consuming time of each node;
2) Select the task which has the longest remaining time, mark as A1 and remaining time of it T_{rem} ;
3) Find the task that has the shortest remaining time T_{rem}' and mark it as B1;
4) Try to speculate on the running time if A1 runs on the node that has the shortest remaining time, mark it as T_{new}. When meeting $T_{rem}'+T_{new}<T_{rem}$, the current task would be backed up to another node;

State (3) represents that enough resources can be assigned to backup tasks in the cluster. Under this circumstance, due to the data locality, the backed-up task has higher priority during the scheduling procedure. Every node has a similar volume of tasks, however, tasks on the low-performance node (e.g. Node A) will not finish on time in a heterogeneous environment. The

finishing timestamp of the current task is calculated by the latest five groups of data according to the linear regression. It is T_{rem} while that of a backup task is recorded as T_{rem}' . It is calculated based on the mean value of a node or the mean value of the cluster when no tasks running on the same node have finished. The mean values of each node are stored in set *MVS*. The profit of starting the backup task is calculated according to the difference between T_{rem} and T_{rem}', and it is defined as a function, called GetProfit. Detailed flow is illustrated in Algorithm 1.

Algorithm 1. Finding the task when there are enough resources

Input: Mean value set (*MVS*), Nodes set(*NS*), Currently running tasks set (*CRTS*), HostName having the longest average time (*HN)*, Speculation Set *(SS)*

```
1  bestProfit=0; bestId=null;
2  For each task in CRTS
3      If AllowedSpeculativeTasks > SpeculationsAlready
4        If GetProfit(taskID)>bestProfit
5           bestProfit= GetProfit(taskID);
6           bestId=taskID;
7        EndIf
8        If |MVS|>|NS| and current task' host name is HN
9                addSpeculativeAttempt(taskID); //start backup
                 Add (taskID, hostname) to SS;
10               continue;
11       ElseIf |MVS|=|NS|-1 and progress < 0.2
12               addSpeculativeAttempt(taskID);
13               continue;
14       EndIf
15     EndIf
16   EndFor
17  If bestProfit>0, saved time>20% and progress<0.2
18     addSpeculativeAttempt(bestId);
19  EndIf
```

Then, the following tasks would be transferred to those nodes that have enough resources. However, it is not reasonable to start a backup task immediately for the purpose of saving resources. A backup task would be launched only when 20% of the time can be saved, or it is currently running on the slowest node (the node has the longest mean running time). If the condition is fulfilled, the task A1 would be backed up.

B. Execution time prediction based on K-ELM

Extreme learning machine based on kernel function (K-ELM) that proposed by Huang, has been proved to be one of most famous algorithms in the machine learning scope. In this section, a prediction model based on K-ELM (PMK-ELM) is presented to estimate the running time of reduce tasks when they are allocated to different nodes.

Following steps indicate the progress of establishing a prediction model for execution time based on K-ELM in detail.

Step 1: Data Acquisition. Historical data of applications are provided by a log analysis tool. The data format is as {*Time, Reducer Id, Node Id, Input size*}.

Step 2: Data Preprocessing. Samples containing high network latency are firstly filtered. Then, the datasets are then divided into training samples and test samples. The former samples are used for training the prediction model using K-ELM, whereas the latter is for examining whether prediction model has been well trained.

Step 3: Model Training. To build PMK-ELM, training parameters of the model are obtained by using the training samples. The specific processes are as follows:

(1) Set weights and the threshold value;

(2) Use activation function to work out the hidden layer output matrix;

(3) Calculate output layer weights.

Step 4: Data validation. Test samples generated by Step 2 are then used for evaluation the performance of the PMK-ELM. According to the parameters retrieved in Step 3, the predictive values are then compared with the actual values to verify the prediction performance.

C. DNSGA-II

1) Mathematical model

When map phase finishes, the intermediate data will be assigned to different reducers; however, the amount of data allocated to each reducer is not consistent with the performance, which consequently causes uneven allocation of reducers to nodes. For the sake of making reduce phase consumes less time and hard disk space occupation more balanced, following conditions should be met:

a) Disk usage of a data node should be more than the data amount to be assigned to itself;

b) A reducer can only be allocated to a node (if speculation is disabled), but a data node can deal with multiple reducers, as shown in Fig. 1.

Intermediate data generated by map tasks can be divided into *m* splits while there are *n* data nodes in the clusters. If the execution time that each reducer needs is described as t_{mn}, then a matrix *T* can be used to represent the execution time, as shown below:

$$T = \begin{pmatrix} t_{11} & \cdots & t_{1n} \\ \vdots & \ddots & \vdots \\ t_{m1} & \cdots & t_{mn} \end{pmatrix} \quad (1)$$

In order to evaluate the usage of storage space, the percentage of input size sr_{mn} from total available size sl_{mn} is calculated and noted as s_{mn}.

$$s_{mn} = sr_{mn} / sl_{mn} \quad (2)$$

Then, the hard disk space ratio of each split can be described as (3). Finally, the elements of *T* and *S* are combined into a new matrix TS , and the new elements are expressed as $(t, s)_{mn}$. The real execution time of data node *i* can be described as t_i,

whereas the split size percentage can be represented as s_i.

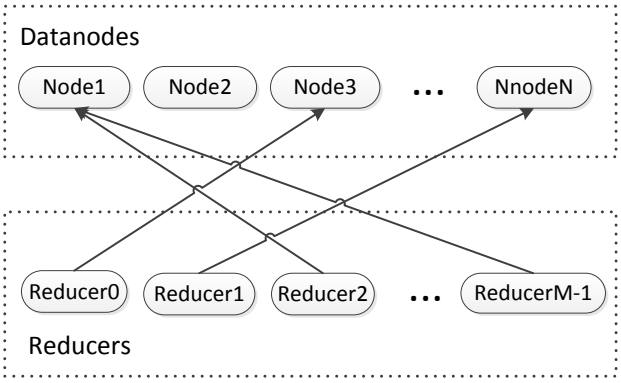


Fig. 1. Relationship between data nodes and reducers

$$S = \begin{pmatrix} s_{11} & \cdots & s_{1n} \\ \vdots & \ddots & \vdots \\ s_{m1} & \cdots & s_{mn} \end{pmatrix}. \quad (3)$$

$$TS = \begin{bmatrix} L_1 \\ L_2 \\ \cdots \\ L_m \end{bmatrix} \quad (4)$$

Actually, every row in TS is a group of feasible solutions and it is expressed as L_i and $L_i = \{(t,s)_1,(t,s)_2,...,(t,s)_n\}_i$.

Finally, two objective functions are given in (5) and (6). The purpose of the algorithm is to find the minimum value of them. Two constraints of the algorithm are given in (7) and (8), in which Sum_s represents the amount of the data input size generated by the Reduce phase.

$$T = \sum_{i=1}^{n} \left| \frac{\bar{t} - t_i}{\bar{t}} \right| \quad (5)$$

$$S = \sum_{i=1}^{n} \left| \frac{s_i - \bar{s}}{\bar{s}} \right| \quad (6)$$

$$Sum_s = \sum_{i=1}^{n} sr_i \quad (7)$$

$$t_i > 0, s_i > 0. \quad (8)$$

2) Design of DNSGA-II

Original NSGA-II can solve multi-object problems. However, it cannot fit into disperse variables. NSGA-II include six aspects. In this paper, the core part of NSGA-II, havs been altered to make it suitable for disperse variable, and this new algorithm is called DNSGA-II. The original NSGA-II algorithm uses Simulated Binary Crossover [23]. However, in the proposed scenario, the Crossover probability *pc* is used for a better grouping after being selected. The Crossover stage in this scheme consists of two steps:

Step 1: Randomly match a group of chromosomes;

Step 2: During matching chromosomes, randomly set intersections to make matched individual chromosomes exchange their information.

After randomly selecting paired chromosomes, two crossover positions are randomly generated; the cross section of elements on the other side of the parent is also removed. Then, the new cross section is added to the sequence of the parent that has cut out some of the elements.

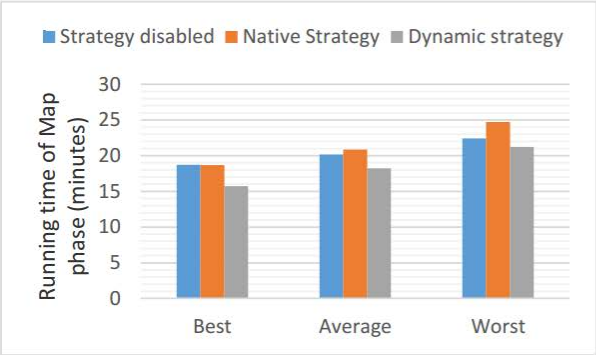
Taking two pairs of chromosomes as an example, where chromosome X= 143|875|62 and chromosome Y = 123|645|78. The cross section is divided by a vertical bar. First, the element corresponding to |875| of X is removed from Y, so Y' = 12378; then a gene fragment of A is added to Y', so offspring Y'' is 123|788|75. Similarly, the offspring X'' is 143|626|45. For newly produced offspring X'' and Y'', it needs to be decided whether the total data size is bigger than the storage quota. If it does not satisfies the above condition, and it contains any number from 1 to 8, it is regarded as applicable; otherwise, iteration will be operated. Under a special condition that there are not any feasible solutions after the final iteration, the repeated number is replaced by another number to format a sequence. Finally, the first group of the output results will be chosen to provide a sequence for reducers. Take Y as an example, it represents that reducer1, reducr2 and reducer are assigned to the Node1. Similarly, reducer6, reducer4 and reducer5 are assigned to the Node2.

IV. EXPERIMENT AND ANALYSIS

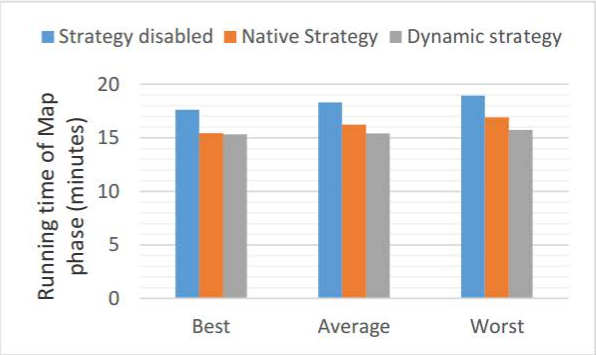
A real heterogeneous cloud environment has been set up in our laboratory to test the performance and benefits of the proposed scheme. The server is equipped with 288 GB of memory and a 10 TB hard driver. Eight virtual machines configured with different amounts of memory and processors are established on the server, and they are connected to a physical switch through the bridge mode. Table I has given the detailed configuration.

TABLE I
THE DETAILED INFORMATION OF EACH VIRTUAL MACHINE

Node Id	Memory(GB)	Core Processors
Node1	10	8
Node2	8	4
Node3	8	1
Node4	8	8
Node5	4	8
Node6	4	4
Node7	18	4
Node8	12	8



(a)



(b)

Fig. 2. Running time comparison of Map phase under three speculative execution strategy. (a) Running time comparison for Sort algorithm under the best condition, the worst condition, and average value of most conditions; (b) Running time comparison for WordCount algorithm under the best condition, the worst condition, and average value of most conditions.

TABLE II
BACKUP TASK INFORMATION OF SORT ALGORITHM

Strategy	Sum of Backup	Success of Backup	Backup Success Rate (%)	Average Running Time of Per Map Task (Minutes)	Running Time of Map Phase (Minutes)
Strategy disabled	-	-	-	9.18	20.16
Native strategy	54	79	68.36	6.58	20.86
Dynamic strategy	19	22	86.36	7.58	18.25

TABLE III
BACKUP TASK INFORMATION OF WORDCOUNT ALGORITHM

Strategy	Sum of Backup	Success of Backup	Backup Success Rate (%)	Average Running Time of Per Map Task (Minutes)	Running Time of Map Phase (Minutes)
Strategy disabled	-	-	-	6.96	18.32
Native strategy	52	22	65.53	6.34	16.23
Dynamic strategy	43	35	81.40	6.12	15.32

In the experiment, dynamic speculative execution strategy work only in the Map stage, while in the Reduce stage, it is disabled to avoid adverse impacts on the backup task load balancing. In addition, only when all the Map tasks have been completed, reduce tasks will start.

Sort and WordCount algorithms were used in the experiments to evaluate the performance of the proposed load balancing scheme. Purdue MapReduce benchmark test suite [25] provided us with free data sets. For Sort algorithm, a data set with 30 GB data was provided while for WordCount algorithm, a cluster workload containing 50GB data was selected as the input. All of the test applications were based on Hadoop 2.6.0.

Overall, the testing process was divided into three steps.

Stage 1: Dataset Collecting. To get historical data on different input data, a Hadoop data collection tool was developed in the lab to collect the data.

Stage 2: Execution Time Prediction. PMK-ELM was then activated and used to predict execution time for current Reduce tasks.

Stage 3: Load balancing. Core resource allocation module in Hadoop allocate resources based on the results of DNSGA-II.

A. Evaluation of Dynamic Speculative Execution Strategy

In this part, the performance of the proposed strategy is compared with the native strategy of MapReduce.

Fig. 2(a) shows the job execution time of three strategies for Sort algorithm. On average, dynamic strategy finishes job 12.5% faster than the native policy and 9.5% faster than strategy disabled.

Fig. 2(b) shows the job execution time of three strategies for WordCount algorithm. On average, dynamic strategy finishes job 16.4% faster than the native policy and 5% faster than strategy disabled.

It is obvious that dynamic strategy can improve the performance for Sort algorithm while the native one cannot provide. To find the reasons, further analysis is given in Table II and Table III. The backup success rate of dynamic strategy is 18.01% higher than the native one for Sort algorithm while it has 19.81% improvement over the initial strategy for WordCount algorithm. Moreover, since Sort algorithm is an I/O-intensive application and unreasonable backup strategy could lead to more severe I/O bottlenecks, so the proposed strategy has achieved more evident results. However, WordCount algorithm is a CPU-intensive application, and usually, it would not reach the CPU bottleneck. In this case, though the accuracy of the native strategy is not high enough, a certain effect still can be obtained. Thus, a dynamic policy that has higher accuracy can further improve the performance.

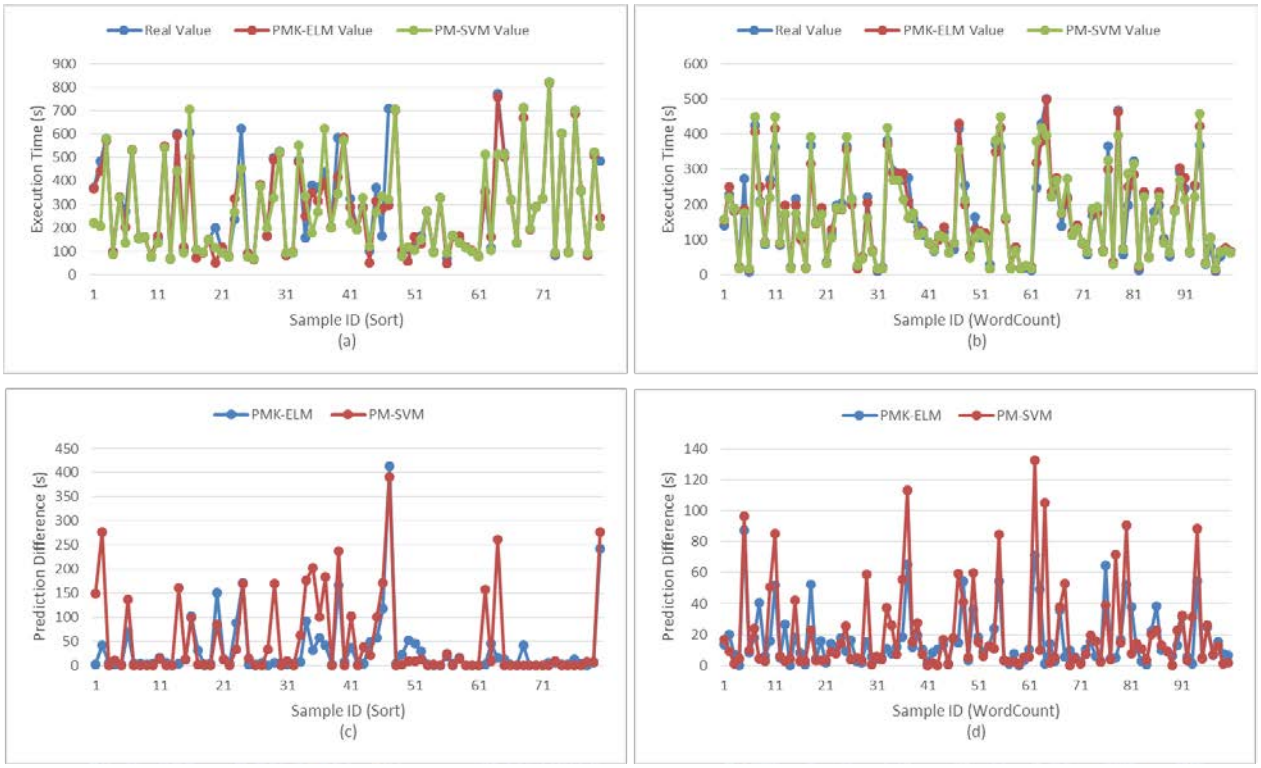


Fig. 3. Performance comparison between PMK-ELM and PM-SVM. (a) Evaluation on data fitting of PMK-ELM and PM-SVM for Sort algorithm; (b) Evaluation on data fitting of PMK-ELM and PM-SVM for WordCount algorithm; (c) Evaluation on estimation difference for Sort algorithm; (d) Evaluation on estimation difference for WordCount algorithm.

TABLE IV
DIFFERENT EXPERIMENT PARAMETERS

	Memory(GB)	Training Datasets Size(Pieces)	Test Dataset Size(Pieces)
Sort	640	560	80
WordCount	800	700	100

TABLE V
THE BEST PARAMETERS FOR SORT ALGORITHM GENERATED BY GA

	Sort		WordCount	
	PMK-ELM	PM-SVM	PMK-ELM	PM-SVM
C	88.529	-	20.521	-
b	-	27.453	-	6.914
σ	0.052	5.115	0.867	16.583
MAPE	14.641%	14.930%	12.647%	13.420%
Training Time(s)	0.024	19.965	0.043	3.2314
Test Time(s)	0.002	0.148	0.003	0.307
Time of finding best parameters(s)	57.643	14429.849	58.476	16861.249

B. Evaluation of PMK-ELM

To evaluate the performance of PMK-ELM, during the experiment, different input sizes and different numbers of input Reducer were also tested, as illustrated in Table IV. For the comparison purposes, the proposed prediction model based on support vector machine (PM-SVM) was replicated in the test environment. A log analysis tool was developed to collect the training and test sets.

PMK-ELM and PM-SVM use RBF function as their kernel function, and its description is shown in Eq. (9). Moreover, PMK-ELM needs another parameter *C* and its definition have been given in [23]. A parameter *b* should be obtained for PM-SVM, whose definition has shown in [12].

$$K\left(x_i, x_j\right) = exp(-\sigma\left\|x_i - x_j\right\|^2), \sigma > 0 \qquad (9)$$

A Genetic Algorithm (GA) is applied to generate the best parameters of PM-SVM and PMK-ELM. In the experiment, max_gen was set 200, *C*, *b* and σ varied from 0 to 100 and the size of the population was 50. Experiment results have been presented in Table V. MAPE, the same metric as [12], is then used to evaluate the results.

The values shown in Table V are the average results of running the applications for 50 times. PMK-ELM trained more than 100 times faster than PM-SVM for Sort algorithm and about 80 times faster than PM-SVM for WordCount algorithm. Although the test time of each group and each application is very short, PMK-ELM still needs shorter time than the PM-SVM. In addition, the accuracy of PMK-ELM is also higher than the PM-SVM.

Fig. 3(a) and Fig. 3(b) depict the prediction values of PMK-ELM and PM-SVM results and their comparison. In Fig. 3(a), values generated by PMK-ELM fit closer to the real values compared with those produced by PM-SVM. In Fig. 3(b), a similar trend can be discovered. If the training time and testing time is taken into account, PMK-ELM is a better choice.

C. The performance of the proposed load balancing scheme (Hadoop-LB)

In this section, the Sort experiment is firstly run once with

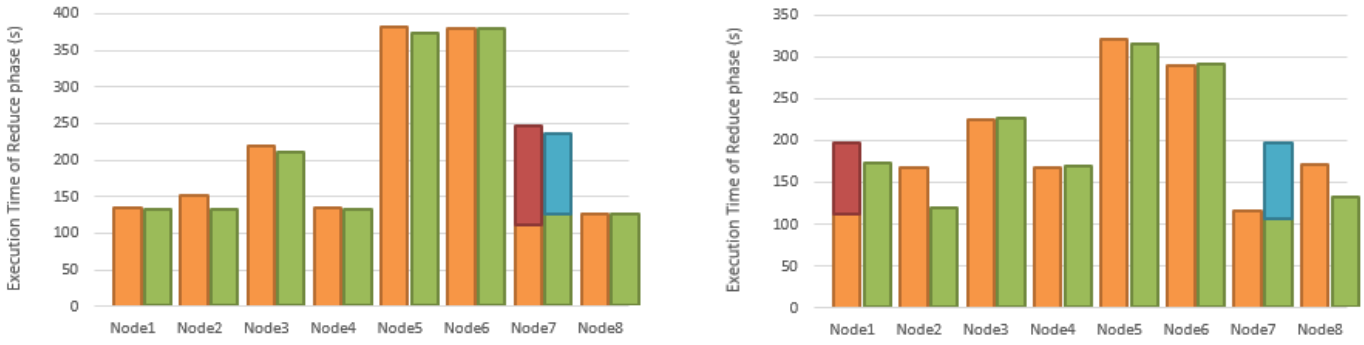


Fig. 4. Comparison between Hadoop-Original and Hadoop-LB in execution time. (a) Final data placement condition under Hadoop-Original; (b) Final data placement condition under Hadoop-LB.

its execution time and hard disk space recorded. Input data volume in the part is 16GB. Corresponding results are shown in Table VI. From Table VI, it can be seen that Node5 and Node6 consumed the most time when executing more tasks. However, the overall job execution time is decided by them. In Table VI, more tasks were assigned to the Node7, which led to its heaviest load. It obvious that data volume is not the only factor that affects the execution time, and node performance is also significant. Then, the results generated by the application were deleted not to affect the performance evaluation while PMK-ELM and DNSGA-II are applied.

TABLE VI
EXECUTION TIME OF DIFFERENT NODES

Node Id	Reducer ID	Execution Time(s)
Node1	1,9	134,132
Node2	5,12	131,152
Node3	2,10	210,219
Node4	0,8	134,127
Node5	6,15	382,374
Node6	3,11	379,279
Node7	7,13,6,17	125,111,136,110
Node8	4,14	130,120

TABLE VII
LEFT HARD DISK SPACE CHANGE WITH PMK-ELM AND DNSGA-II

Node Id	Before Execution(GB)	Hadoop-Original	Hadoop-LB
Node1	271.16	269.36	268.37
Node2	276.03	274.22	274.15
Node3	288.13	286.34	286.26
Node4	289.85	288.08	287.96
Node5	268.98	267.19	267.09
Node6	204.33	202.44	202.37
Node7	286.73	283.05	283.90
Node8	301.04	299.22	299.15

As a result, better performance was obtained. DNSGA-II randomly chooses a group of solutions to form each group, one is group A={{1,0,17},{1,9},{10,2},{11,3},{4,12},{13,5},…}. {1,0,17} represents that reducer0, reducer1 and reducer 17 were assigned to the Node1 while {1,9} represents reducer1 and reducer9 were processed on the Node2, and so on. The benefits are shown in Fig. 4, Table VI, Table VII and Table VIII.

TABLE VIII
COMPARISON BETWEEN ORIGINAL AND OPTIMIZED SCHEME IN DIFFERENCE OF DISTRIBUTION

Node Id	Hadoop-Original	Hadoop-LB
Difference of Distribution (%)	1.160	1.075
Job Execution Time(s)	744	663

As shown in Fig. 4, the maximum reducer execution time of Fig. 4(b) is shorter than the original Group in Fig. 4(a), which determines the Hadoop-LB finish the reduce stage faster than the original. The results shown in Table VIII also prove it. Not only does the load balancing scheme make the application run faster, but also help the hard disk occupation more reasonable. Table VII shows the hard disk occupation when PMK-ELM and DNSGA-II are applied. The difference in distribution calculated by Eq. (6) in Section IV has been given in Table VIII, in which the proposed scheme also shows a better performance in job execution time.

V. CONCLUSION

In this paper, a new dynamic speculative execution strategy is proposed to improve the performance of the Map phase. A prediction model called PMK-ELM is presented to predict the execution time of each reducer. Combined with the DNSGA-II, which is designed to facilitate the selection of a suitable sequence for disperse variables, better load balancing is achieved. According to the experiment, 10.9% of the time is saved while difference of distribution is also decreased.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp.50-58, 2010.

[2] Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu, "Achieving Efficient Cloud Search Services: Multi-keyword Ranked Search over Encrypted Cloud Data Supporting Parallel Computing," *IEICE Trans. Commun.*, vol. E98-B, no. 1, pp.190-200, 2015.

[3] Z. Fu, X. Sun, N. Linge, and L. Zhou, "Achieving effective cloud search services: multi-keyword ranked search over encrypted cloud data supporting synonym query," *IEEE Trans. Consumer Electron.*, vol. 60, no. 1, pp. 164-172, 2014.

[4] R. Sánchez, F. Almenares, P. Arias, D. Díaz-Sánchez, and A. Marín, "Enhancing privacy and dynamic federation in IdM for consumer cloud computing," *IEEE Trans. Consumer Electron.*, vol. 58, no. 1, pp. 95-103, 2012

[5] S. Abolfazli, Z. Sanaei, M. Alizadeh, A. Gani, and F. Xia, "An experimental analysis on cloud-based mobile augmentation in mobile cloud computing," *IEEE Trans. Consumer Electron.*, vol. 58, no. 1, pp. 146-154, 2014.

[6] P. A. Cabarcos, F. A. Mendoza, R. S., Guerrero, A. M. Lopez, and D. Diaz-Sanchez, "SuSSo: seamless and ubiquitous single sign-on for cloud service continuity across devices," *IEEE Trans. Consumer Electron.*, vol. 58, no. 4, pp. 1425-1433, 2012.

[7] S. Lee and D. Lee, and S. Lee, "Personalized DTV program recommendation system under a cloud computing environment," *IEEE Trans. Consumer Electron.*, vol. 56, no. 2, pp. 1034-1042, 2010.

[8] Y. Lee, "An integrated cloud-based smart home management system with community hierarchy," *IEEE Trans. Consumer Electron.*, vol. 62, no. 1, pp.1-9, 2016.

[9] B. Eom, C. Lee, H. Lee, and W. Ryu, "An adaptive remote display scheme to deliver mobile cloud services, *IEEE Trans. Consumer Electron.*, vol. 60, no. 3, pp. 540-547, 2014.

[10] S. Grzonkowski, and P. M. Corcoran, "Sharing cloud services: user authentication for social enhancement of home networking," *IEEE Trans. Consumer Electron.*, vol. 57, no. 3, pp. 1424-1432, 2011.

[11] B. Palanisamy, A. Singh, and L. Liu, "Cost-effective resource provisioning for mapreduce in a cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 5, pp. 1265-1279, 2015.

[12] Y. Fan, W. Wu, Y. Xu, and H. Chen, "Improving MapReduce Performance by Balancing Skewed Loads," *China Communications*, vol. 11, no. 8, pp. 85-108, 2014.

[13] A. Matsunaga, and J. A. B. Fortes, "On the use of machine learning to predict the time and resources consumed by applications," *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pp. 495-504, 2010.

[14] T. P. Jing, and J. Yan, "Computing resource prediction for mapreduce applications using decision tree," Web Technologies and Applications, pp. 570-577, 2012.

[15] T. Wood, L. Cherkasova, K. Ozonat, and P. Shenoy, "Profiling and modeling resource usage of virtualized applications," *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, pp. 366-387, 2008.

[16] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1 pp. 155-162, 2012.

[17] M. Zaharia, A. Konwinski, A.D. Joseph, R. Katz, and I. Stoica, "Improving Mapreduce Performance in Heterogeneous Environments," *OSDI*, pp. 29-42, 2008.

[18] Q. Chen, L. Cheng, and X. Zhen, "Improving MapReduce performance using smart speculative execution strategy," *IEEE Trans. Comput.*, vol. 63, no. 4, pp. 954-967, 2014.

[19] Q. Liu, W. Cai, J. Shen, D. Jin, and N. Linge, "A load-balancing approach based on modified K-ELM and NSGA-II in a heterogeneous cloud environment," *Proceedings of 2016 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 411-412, 2016.

[20] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489-501, 2006.

[21] A. Samat, P. Du, S. Liu, J. Li, and L. Cheng, "Ensemble Extreme Learning Machines for Hyperspectral Image Classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 4, pp. 1060-1069, 2014.

[22] G. B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 42, no. 2, pp. 513-529, 2012.

[23] G. B. Huang, and C. K. Siew, "Extreme learning machine with randomly assigned RBF kernels," *International Journal of Information Technology*, vol. 11, no. 1, pp. 16-24, 2005.

[24] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans on Evol. Comput.*, vol. 6, no. 2, pp. 182-197, 2002.

[25] Farz Ahmad, Srimat T. Chakradhar, Anand Raghunathan, and T. N. Vijaykumar, "Tarazu: optimizing MapReduce on heterogeneous clusters," *Proceeding of ACM SIGARCH Computer Architecture News*, pp. 61-74, 2012.

BIOGRAPHIES

Qi Liu (M'11) received his BSc degree in Computer Science and Technology from Zhuzhou Institute of Technology, China in 2003, and his MSc and Ph.D. in Data Telecommunications and Networks from the University of Salford, UK in 2006 and 2010. His research interests include context awareness, data communication in MANET and WSN, and smart grid. His recent research work focuses on intelligent agriculture and meteorological observation systems based on WSN.

Weidong Cai received his bachelor's degree in Software Engineering from Nanjing University of Information Science and Technology in 2014, and he is pursuing a master's degree in software engineering at the Nanjing University of Information Science and Technology. His research interests include Cloud Computing, Distributed Computing and Data Mining.

Jian Shen received his bachelor's degree in Electronic Science and Technology Specialty from Nanjing University of Information, Science and Technology in 2007, and he received his masters and PhD in Information and communication from CHOSUN University, South Korean in 2009 and 2012. His research interests includes Computer network security, information security, mobile computing and network, wireless ad hoc network.

Xiaodong Liu received his PhD in Computer Science from De Montfort University and joined Napier in 1999. He is a Reader and is currently leading the Software Systems research group in the IIDI, Edinburgh Napier University. He was the director of Centre for Information & Software Systems. He is an active researcher in software engineering with internationally excellent reputation and leading expertise in context-aware adaptive services, service evolution, mobile clouds, pervasive computing, software reuse, and green software engineering. He has meanwhile a successful track record of teaching in a number of software engineering courses which are widely informed by his research activities.

Nigel Linge received his BSc degree in Electronics from the University of Salford, UK in 1983, and his PhD in Computer Networks from the University of Salford, UK, in 1987. He was promoted to Professor of Telecommunications at the University of Salford, UK in 1997. His research interests include location based and context aware information systems, protocols, mobile systems and applications of networking technology in areas such as energy and building monitoring.