2012 IEEE 4th International Conference on Cloud Computing Technology and Science

A Flexible Analysis and Prediction Framework on Resource Usage in Public Clouds

Chia-Yu Lin*, Yan-Ann Chen*, Yu-Chee Tseng*, Li-Chun Wang†
*Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
E-mail: sallylin0121@gmail.com, chenya@cs.nctu.edu.tw, yctseng@cs.nctu.edu.tw
†Department of Electrical and Computer Engineering, National Chiao Tung University, Hsinchu, Taiwan
E-mail: lichun@g2.nctu.edu.tw

***Abstract*—In cloud computing environments, users can rent virtual machines (VMs) from cloud providers to execute their programs or provide network services. While using this kind of cloud services, one of the biggest problems for the users is to determine the proper number of VMs to complete the jobs considering both budget and time. In this paper, we propose a resource prediction framework (RPF), which can help users choose the minimum number of virtual machines to complete their jobs within a user specified time constraint. In order to verify the feasibility of RPF, we have done three case studies, namely parallel frequent pattern growth (FP-Growth), parallel K-means, and Particle Swarm Optimization (PSO). FP-growth, K-means and PSO are data intensive algorithms. These algorithms are typically executed repeatedly with different execution parameters to find the optimal results. When evaluating RPF by these algorithms in cloud environments, we have to modify them to parallel versions. The evaluation results indicate that RPF can successfully obtain the minimum number of VMs with acceptable errors. According to our case studies, the proposed RPF can be adopted by data intensive jobs by providing flexibility to both end users and cloud system providers.**

***Keywords*-cloud computing, resource prediction, MapReduce, Parallel Frequent Pattern Growth, Parallel K-means, Particle Swarm Optimization**

I. Introduction

Cloud computing provides users flexible computing and storage resources. By renting resources from cloud providers, users could avoid expensive hardware investment and maintenance cost. The services of cloud computing is generally classified into three types, software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS). Amazon Elastic Compute Cloud (Amazon EC2), which is an IaaS provider, provides many types of VMs with different EC2 computing units and memory sizes. The billing method is based on the offered VM capabilities and the rented time. MapReduce is a parallel programming platform for users to develop parallel programs in cloud environments. Although the goal is to reduce execution time, users usually do not know how many VMs they really need to execute their jobs. There is a trade-off between the rented VM numbers and the resulting execution time. The goal of this paper is to propose a resource prediction framework (RPF) for predicting the minimum number of VMs required for a MapReduce job under a user specified time constraint.

The proposed RPF consists of training and prediction algorithms. In the training algorithm, users execute their parallel jobs with different execution parameters and the system records the execution parameters, the number of VMs used, and the corresponding execution time. After collecting enough data, the training algorithm computes a model for prediction. When the job is executed with a new set of execution parameters and an expected response time, the prediction algorithm computes a suggested minimum number of VMs based on the trained model to the user.

In this paper, we conduct three case studies, namely parallel frequent pattern growth (parallel FP-growth), parallel K-means and Particle Swarm Optimization (PSO), to verify the performance of RPF. Since these algorithms are data-intensive ones, their execution times may be reduced significantly while executing in parallel. We hence modify them to parallel versions by referring to [1][2][3] and execute them on RPF. The evaluation results show that RPF can predict the required number of VMs with small root mean square errors (RMSE).

The contribution of this paper are as follows:

- A novel VM-based and job-oriented resource prediction framework.
- The proposed training and prediction algorithms.
- The regression functions for prediction.

The rest of the paper is organized as follows: Section II details the related work of resource provisioning in cloud datacenters. Section III describes our system design. The training and prediction algorithms of RPF are discussed in Section IV. Section V, Section VI and Section VII describe our case studies to demonstrate some performance evaluations. Finally, Section VIII concludes the paper.

II. Related Work

Recent research on cloud computing has been focused on Service Level Agreement based (SLA-based) resource provisioning. Reference [4] proposes a model to manage the VMs in the datacenter dynamically to meet SLA. The model monitors the resource demand during the current time window in order to make decisions about the server allocations and job admissions during the next time window. Therefore, the model can adjust the number of VMs for high performance computing (HPC) jobs and web jobs

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

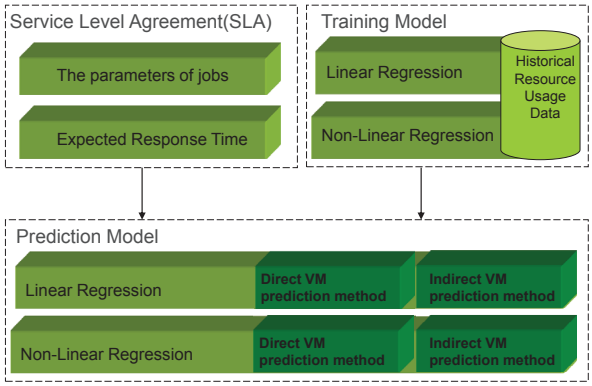


Figure 1. System architecture of proposed resource prediction framework.

dynamically. In [5] and [6], they meet SLA by resource prediction. The prediction results are the utilizations of CPU and memory. However, users cannot easily translate the results into the needed number of VM numbers from this estimation result. The framework we proposed output the minimum number of VMs, which is more intuitive for users. Reference [7] proposes an optimal resource provisioning for MapReduce programs. They analyze the execution time of mapper and reducer process and use regression methods to estimate the execution time of a MapReduce job with different numbers of VM. The analyzing process is only suitable for simple MapReduce jobs such as WordCount and PageRank. Our framework, however, can estimate the number of VMs and execution time for any kind of MapReduce jobs.

III. SYSTEM DESIGN

In this section, we introduce the architecture of RPF for predicting the minimum number of required VMs to execute a MapReduce job within the expected execution time. We assume that the MapReduce platform in the training process and in the prediction process is the same.

The input to this framework is a set of execution parameters of a MapReduce job and the users' expected execution time for the job. The RPF then recommends the number of VMs to run the MapReduce job. The system architecture is shown in Fig. 1.

In the SLA module, users have to input some part of execution parameters and the response time they expect for the MapReduce job. This information is sent to the prediction model.

In training model, the system collects the historical resource usage data including the number of VMs, execution parameters and response time of previous executions. In addition, the regression-based model is adopted to analyze the relationship among the response times, VM resources and the sets of execution parameters. The training model will output the regression model with the smallest RMSE to the prediction model.

The preferred regression-based model and the SLA requirements are the inputs of the prediction model. This prediction model could directly predict the minimum number of required VMs of a job, which is named "direct VM prediction method", by applying the regression model with the expected response time and the execution parameters. In some cases, however, we observed the direct VM prediction method is not applicable because the parallelism degree of the MapReduce job is low. An indirect VM prediction method is proposed to deal with this problem by predicting the execution time of a job first.

IV. RPF ALGORITHMS

Next, we present the training algorithm, prediction algorithm and an extension of prediction method in Section IV-A, Section IV-B and Section IV-C, respectively.

A. Training Algorithm

In order to construct an accurate prediction model, users have to execute their parallel jobs with different execution parameter sets. For instance, in a sorting job, the number of items to be sorted is one execution parameter and the response time of an execution is another. We define the following notations to formulate this algorithm. $s_{i1}, \ldots, s_{in}$ represent a set of n execution parameters in the ith execution. z_i is the number of VMs used in the ith execution. The input of the training algorithm is a set S, $\{(s_{11}, \ldots, s_{1n}, z_1), \ldots, (s_{i1}, \ldots, s_{in}, z_i), \ldots, (s_{k1}, \ldots, s_{kn}, z_k)\}$, which consists of the set of n execution parameters and the used VM numbers for each execution i. Since the regression-based method is utilized in this system, we adopt three common regression models for the training purpose. Eq. (1) is a linear regression model and Eq. (2) is a quadratic regression model and also a linear regression model since the coefficients are not in the exponential part. Eq. (3) is a non-linear regression model since the coefficients are exponential.

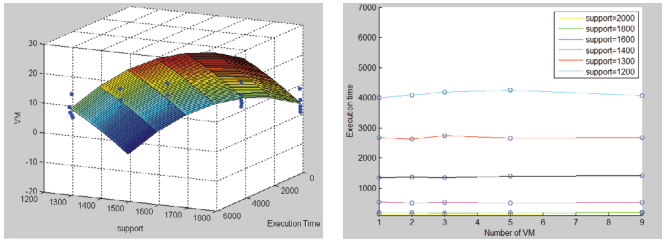
$$f(x_1, \ldots, x_n) = a_1 x_1 + \ldots + a_m x_n \quad (1)$$

$$f(x_1, \ldots, x_n) = a_1 x_1^2 + \ldots + a_n x_n^2 + a_{n+1} x_1 + \ldots + a_{2n} x_n + a_m \quad (2)$$

$$f(x_1, \ldots, x_n) = a_1 e^{a_2 x_1} + \ldots + a_{2n-1} e^{a_{2n} x_n} + a_m \quad (3)$$

In the regression-based method, a $(n+1)$-dimension space is constructed and the data points S_i are plotted at the coordinate $(s_{i1}, \ldots, s_{in}, z_i)$ for each execution i. The basic idea of regression-based method is to formulate a curve or surface which approximately fits all the data points by the function of the regression models. In other words, the training algorithm computes the coefficient set $A = (a_1, \ldots, a_m)$ of the regression models such that

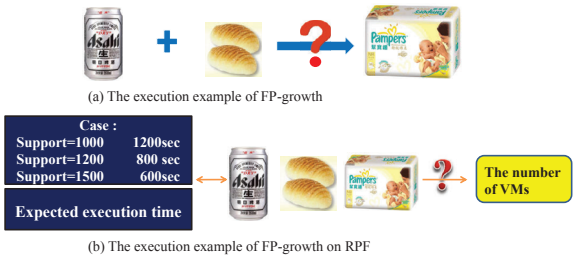
978-1-4673-4510-1/12/$31.00 ©2012 IEEE



(a) The relationship of VM, execution time and execution parameter

(b) The relationship of VM and execution time

Figure 2. The relationship among the number of VMs, execution time and support in parallel FP-Growth case study.



(a) The execution example of FP-growth

(b) The execution example of FP-growth on RPF

Figure 3. The application scenario of FP-growth.

$\sum_{i=1}^{k}[z_i - f(s_{i1}, \ldots, s_{in})]^2$ is minimized. The detail information of the regression method is in [8]. Among these three regression models, the model with the minimum RMSE, i.e. $\sqrt[2]{\sum_{i=1}^{k}[z_i - f(s_{i1}, \ldots, s_{in})]^2}$, is chosen as the model for the prediction algorithm.

B. Prediction Algorithm

The training procedure chooses the regression model with RMSE the smallest and outputs the computed coefficient set, $A = (a_1^*, \ldots, a_m^*)$, of the regression model. After the model is trained, the prediction algorithm could predict the required number of VMs of a job when it is a new execution parameter set. At first, the user specifies a new set of execution parameters, $\{s'_1, \ldots, s'_n\}$, for a job in this execution. In the training phase, the response time of an execution is one of the execution parameters. However, the response time is unknown in the prediction phase. The execution time parameter should be the expected execution time, T_{ept}, specified by the user for this new job. Substituting the variables of the chosen model among Eq. (1), Eq. (2) and Eq. (3) with $\{s'_1, \ldots, s'_{n-1}, T_{ept}\}$ will obtain the predicted number of VMs required for this execution. For instance, we assume that Eq. (2) is the preferred model and the expected execution time is at the last parameter of the model. Therefore, the prediction model shown in Eq. (4) is Eq. (2) with the computed coefficient set A^* from training algorithm.

$$f(x_1, \ldots, x_n) = a_1^* x_1^2 + \ldots + a_n^* x_n^2 + a_{n+1}^* x_1 + \ldots + a_{2n}^* x_n + a_m^* \quad (4)$$

Finally, the predicted number of VMs is calculated by $f(s'_1, \ldots, s'_{n-1}, T_{ept}) = a_1^* {s'_1}^2 + \ldots + a_{n-1}^* {s'_{n-1}}^2 + a_n^* {T_{ept}}^2 + a_{n+1}^* s'_1 + \ldots + a_{2n-1}^* s'_{n-1} + a_{2n}^* T_{ept} + a_m^*$.

C. Extension: Indirect VM Prediction

As we observed in the experiment, the required number of VMs might not closely related to all of the execution parameters if the parallel program is not fully parallelized. Fig. 2 is the experiment result of parallel FP-growth. Fig. 2(a) and (b) illustrate that the number of VMs is not related to the execution time and the execution time is only related to "support", which is an execution parameter of FP-growth. Therefore, the prediction algorithm in Section IV-B might not be applicable in this situation. We propose an indirect VM prediction algorithm for solving this problem. First, it needs some modifications to the training algorithm in Section IV-A. z_i in the indirect VM prediction is the execution time of the ith execution and the used number of VMs is one of the execution parameters for the ith execution. Except for the modifications, the training algorithm is the same as that in Section IV-A. The basic idea of indirect VM prediction is to bound the expected execution time by predicting the execution time with all of the feasible number of VMs, $V = (v_1, \ldots, v_r)$. When there is a new job with a new execution parameter set, the execution times with different number of VMs parameters are predicted by the chosen regression model and the execution times are recorded by a set $T = (t_1, \ldots, t_r)$. t_r represents the predicted execution time by r VMs execution parameter. By comparing the execution time predicted with the T_{ept} and we could obtain a time called T_{prt} which is is smaller than T_{ept} and is the largest one in T. After finding T_{prt}, we can output the number of VMs in V which is corresponding to T_{prt}. Then, this predicted number of VMs is the minimum number of VMs, which can complete the new job within the T_{ept}.

V. Case Study of FP-growth

Fig. 3(a) shows the example of FP-growth. FP-growth can find out that whether they buy diapers if customers have already bought beers and bread. This information is useful for some retailers such as Wal-Mart. Fig. 3(b) shows the execution process of FP-growth on RPF. While we are executing parallel FP-growth to find out the association rules, RPF records the execution time and execution status such as the number of VMs and support. This historical resource usage data can be used to predict the execution number of VMs.

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

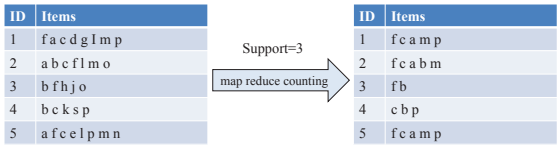


Figure 4. MapReduce counting in Parallel FP-growth.

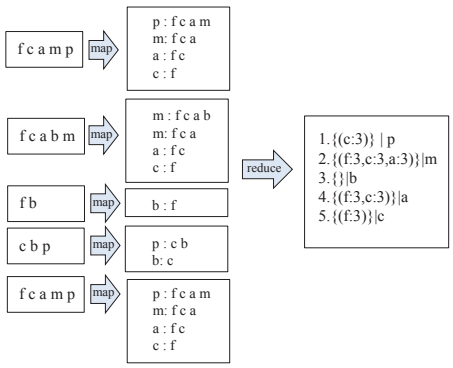


Figure 5. The example of parallel FP-growth.

A. Parallel FP-growth

FP-growth uses the divide and conquer strategy to find the frequent items. There are two phases in FP-Growth algorithm. First phase is constructing FP-tree. In this phase, we find frequent items whose appearance frequency is larger than minimum support and sort items in frequency descending order. And then we construct a root node which is marked by null and add the branch of each traction. The second phase is FP-Growth. In this phase, we find out conditional pattern bases which are associated with every frequent item in FP-tree. The conditional bases whose frequency are larger than minimum support are put in conditional FP-trees. After constructing conditional FP-tree of every frequent item, we can output the frequent patterns. The detail of FP-growth is in [9][10].

We execute MapReduce counting to find frequent items whose appearance frequency is larger than minimum support and sort items in frequency descending orderFig. 4. After MapReduce counting, frequent items are separated into many parts for every mapper to execute. Every mapper output the conditional patterns according to input files and send the result to the reducers. The reducers have two step to execute. First, the reducers collect the conditional patterns from every mapper and combine these patterns which are associated with same frequent item. Second, the reducers construct the conditional FP-tree of every frequent item and output the frequent patterns Fig. 5. The execution time of parallel FP-growth is much shorter than the execution time of FP-growth.

B. Parallel FP-growth on RPF

We execute parallel FP-growth on MapReduce platform with different support. Therefore, we can collect different execution parameters, such as support and execution time to be the input of training algorithm. In Fig. 6, we input support 2000 to 11000 and the historical execution time in the training algorithm. After executing training algorithm, we can obtain an regression model whose root mean square error (RMSE) is the smallest. We use this regression model to predict the number of execution VMs for the new jobs. The new job's support which is 5000 and expected response time 750 second which is set by users is the input of prediction algorithm. We put the support, the expected response time and the regression coefficient set A which is the output of training algorithm into the prediction equation. The equation outputs the minimum number of VMs. The example of prediction algorithm is shown in Fig. 7. If the prediction result is not accurate enough, we can try to use prediction algorithm in Section IV-C to estimate the number of execution VMs. We put the support, the number of VMs and the regression coefficient set A which is the output of training algorithm into the prediction equation. The equation outputs the execution time of every situation. We can use the expected execution time and the result of prediction algorithm to decide the minimum number of VMs of the new job. Fig. 8 shows the example of prediction execution time methods.

C. Experiment Setup

We construct two physical machines as the platform. The CPU of physical machines is Intel(R) Core(TM) i7-2600 CPU 3.40GHz and 4G ram. There are four virtual machines in each physical machine. There are 1 core and 1G memory in each virtual machine. Hadoop is installed in virtual machines. We refer to [1] and Apache Mahout project [11] to modify the FP-growth in parallel. The dataset is from [12]. We choose the dataset which consists of 10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users.

D. Evaluation

The models in Eq. (1), Eq. (2) and Eq. (3) are adopted in this case study. The trained models of direct and indirect VM prediction method are shown in Fig. 9 and Fig. 10, respectively. Fig. 11 shows the RMSEs of these two prediction methods. The prediction result of exponential function is the most accurate and the result of quadratic function has the maximum error. Overall, the regression methods we propose can output the minimum number of execution VMs to users accurately.

VI. Case Study of K-means

K-means is the simplest cluster algorithm to find clusters from a lot of data. We modify K-means algorithm to

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

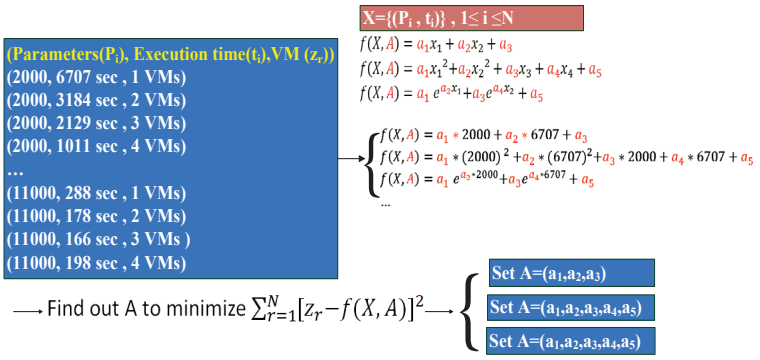


Figure 6. The example of training algorithm by FP-growth.

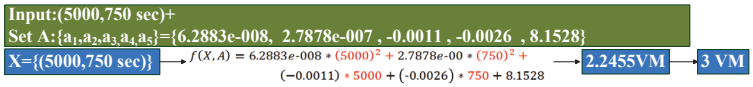


Figure 7. The example of direct prediction method by FP-growth.

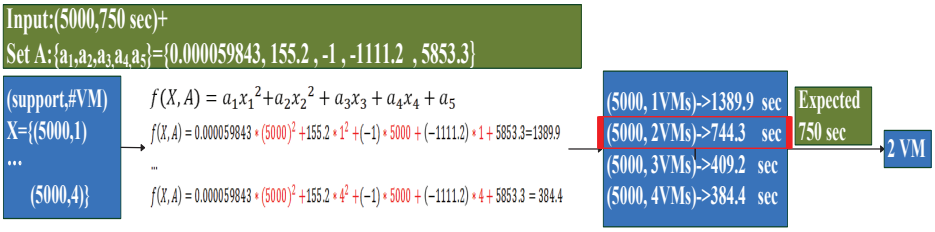
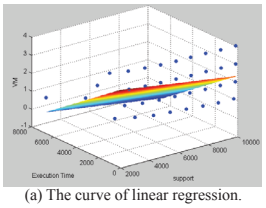
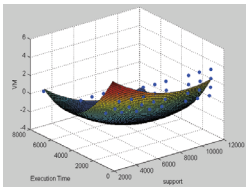


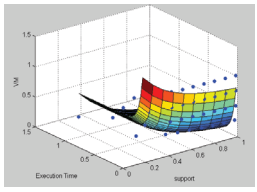
Figure 8. The example of indirect prediction method by FP-growth.



(a) The curve of linear regression.

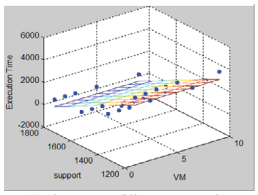


(b) The curve of quadratic regression.

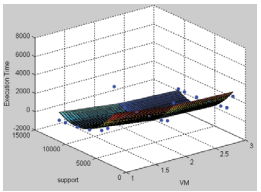


(c) The curve of exponential regression.

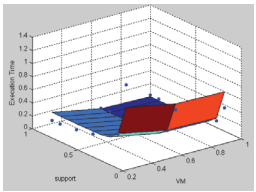
Figure 9. The trained models of parallel FP-growth experiment by direct prediction method.



(a) The curve of linear regression.



(b) The curve of quadratic regression.



(c) The curve of exponential regression.

Figure 10. The trained models of parallel FP-growth experiment by indirect prediction method.

parallel K-means, execute it on Hadoop platforms to collect the historical job usage data and estimate the minimum VM numbers of a new K-means jobs. The scenario and introduction of K-means is in Section VI-A.

A. *Parallel K-means*

K-means clustering algorithm is a well-known unsupervised clustering algorithm and it partitions objects into groups by analyzing the relationship or similarity of ob-

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

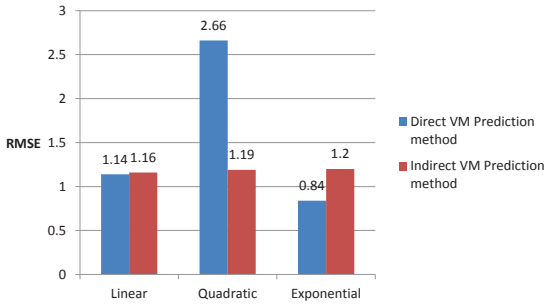


Figure 11. Performance evaluation of parallel FP-growth experiment.

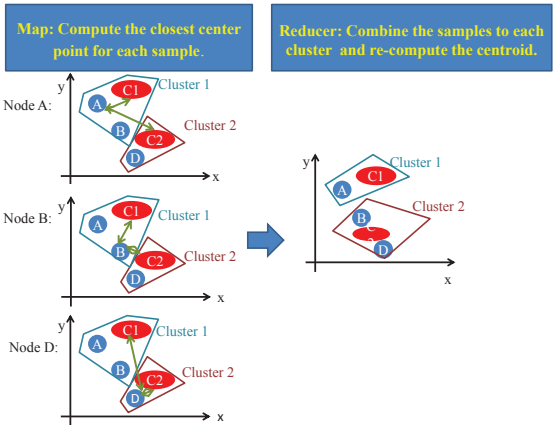


Figure 12. The execution example of parallel K-means.

jects. Usually, the input data of K-means called dataset is intensive. For example, the BigCross dataset [13] which is 11,620,300 points in 57-dimensional space and the Census1990 dataset [14] which is 2,458,285 points in 68 dimensions are used in [15]. Furthermore, analyzing the relationship between objects could be processed in parallel. The sequential K-means algorithm could be modified to a parallel K-means algorithm. Then, we use the parallel K-means to be a case study of RPF.

K-means is used to partition the data into K clusters. The input of K-means are the cluster number K and the data which is used to cluster. The output is K clusters. There are four steps of K-means. First step is choosing K data to be the central node of every cluster. Every node compute the distance between it and the central node is second step. The third step is assign every node to the cluster according to the distance result in second step. The fourth step is computing the central node of every cluster according to the average value of the data in the clusters. The second, third and fourth steps are repeated until the clusters are not changed anymore.

Fig. 12 shows the execution process of parallel K-means. The input data node A,B and D is allotted into different mapper to compute the distance between it and every central node C1, C2. After mappers output the distance result of A,B and D, the reducers have two step to execute. First, the reducers find the closest cluster for A,B and D according to the distance result. Therefore, B is changed to cluster 2. Second, the reducers compute the central node of every cluster according to the average value of the data node in the clusters. The central node of cluster 2 is changed to D. The mapper and reducer process are iterative executed until the clusters are not changed anymore.

B. Parallel K-means on RPF

The steps of resource prediction of parallel PSO is same with parallel FP-growth. We send different execution parameters K and the corresponding execution time to training algorithm. The training algorithm outputs an regression model whose root mean square error (RMSE) is the smallest. We use this regression model to predict the number of execution VMs for the new jobs with different K. The prediction algorithm and indirect VM prediction algorithm output the minimum number of VMs of the new jobs. We choose the result whose RMSE is smaller to be the output of RPF.

C. Parallel K-menas on RPF Experiment Setup

The experiment environment is same with the FP-growth experiment. We refer to [2] and Apache Mahout project [11] to modify the K-means in parallel.The dataset is also same with FP-growth dataset which is rating data sets from the MovieLens web site [12]. We use K-means to cluster the users who have the same interest of movies.

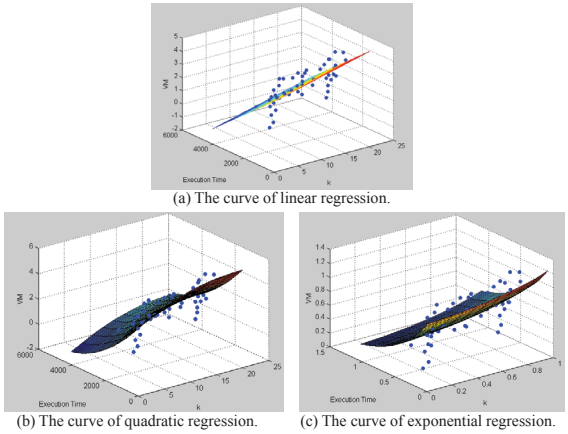
D. Evaluation

The models in Eq. (1), Eq. (2) and Eq. (3) are adopted in this case study. The trained models of direct and indirect VM prediction method are shown in Fig. 13 and Fig. 14, respectively. Fig. 15 shows the RMSEs of these two prediction methods. The prediction result of quadratic function is the most accurate and the result of linear function has the maximum error. We also can find that the RMSE of RPF is no more than 1.5. In brief, RPF can predict the minimum number of execution VMs to users accurately.

VII. Case Study of PSO

Particle Swarm Optimization (PSO), an optimization algorithm that was inspired by bird and fish foraging social behavior [16]. In the beginning, the birds have no idea about the food location. They guess and fly to better location by their experience and intuition. When a bird finds the food, it broadcasts the location to other birds. Other birds fly to the food location. Bird foraging behavior is the concept of mutual influence in society which can lead all individual bird toward the location of the optimal solution. PSO has become popular because it is simple, requires little tuning, and has been found to be effective for a wide range of problems. Since every node in PSO can compute the local best value

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

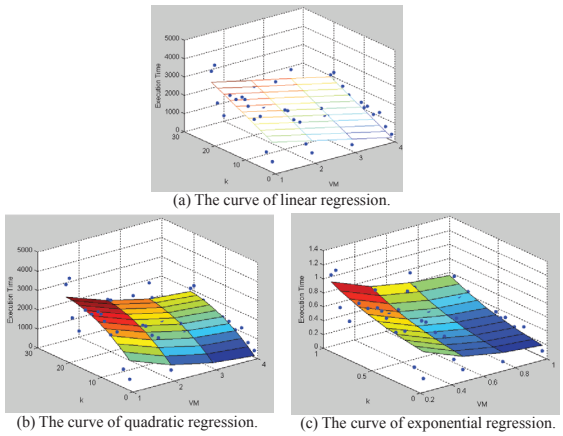


(a) The curve of linear regression.

(b) The curve of quadratic regression.

(c) The curve of exponential regression.

Figure 13. The trained models of parallel K-means experiment by direct prediction method.



(a) The curve of linear regression.

(b) The curve of quadratic regression.

(c) The curve of exponential regression.

Figure 14. The trained models of parallel K-means experiment by indirect prediction method.

by itself, PSO is suitable for MapReduce framework. In this paper, we refer to [3] to modify PSO in parallel and demonstrate parallel PSO on RPF.

A. Parallel PSO on RPF

The steps of resource prediction of parallel PSO is same with parallel FP-growth and parallel K-means. We execute parallel PSO with Rosenbrocks test function by different dimension d. We collect the different parameter d and execution time to be the input of training algorithm. After the process of training, we predict the minimum number of VMs of a new job with different d by the prediction algorithm and indirect VM prediction algorithm. The performance of parallel on RPF is shown below.

B. Parallel PSO on RPF Experiment Setup

The experiment environment is same with the FP-growth and K-means experiment. The input dataset is generated by latin hypercube sampling [17] with different domain. The

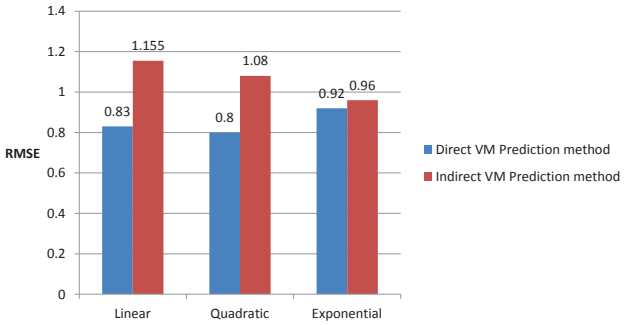
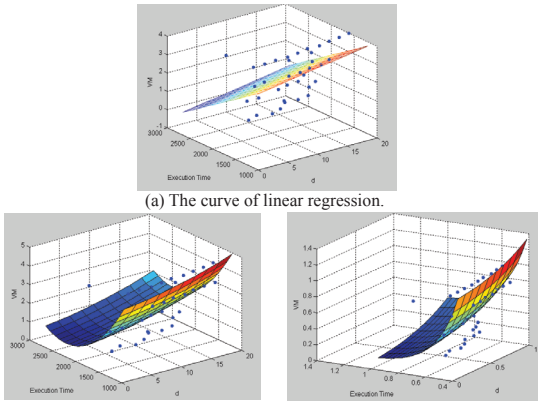


Figure 15. Performance evaluation of parallel K-means experiment.



(a) The curve of linear regression.

(b) The curve of quadratic regression.

(c) The curve of exponential regression.

Figure 16. The trained models of PSO experiment by direct prediction method.

optimization function of PSO is Rosenbrocks test function and $x_i \in [-100, 100]$. We use PSO to find optimal solution of Rosenbrocks function.

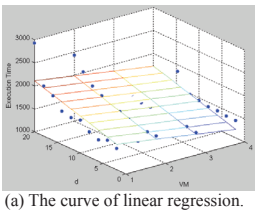
C. Evaluation

The models in Eq. (1), Eq. (2) and Eq. (3) are adopted in this case study. The trained models of direct and indirect VM prediction method are shown in Fig. 16 and Fig. 17, respectively. The RMSEs of these two prediction methods are shown in Fig. 18. The prediction result of indirect VM prediction is more accurate than direct VM prediction. The reason is that parallel PSO program is not thorough parallel. That is, while users' program is not fully parallel, the indirect VM prediction algorithm can provide the better results.

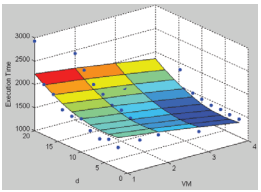
VIII. CONCLUSION

In this paper, we proposed a VM-based and job-oriented resource prediction framework (RPF) to predict the minimum number of VMs which can complete the user's MapReduce job within the user specified response time. We not only proposed RPF but also demonstrated parallel FP-growth,

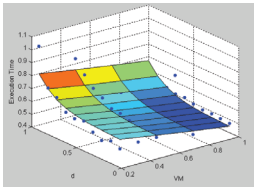
978-1-4673-4510-1/12/$31.00 ©2012 IEEE



(a) The curve of linear regression.



(b) The curve of quadratic regression.



(c) The curve of exponential regression.

Figure 17. The trained models of PSO experiment by indirect prediction method.

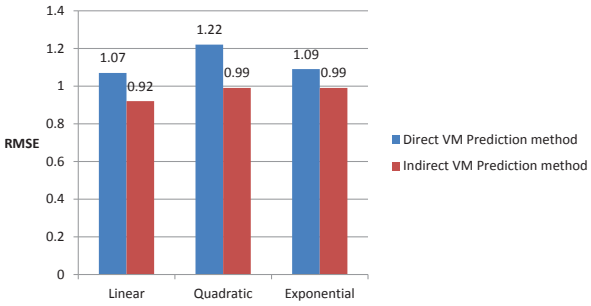


Figure 18. Performance evaluation of PSO experiment

parallel K-means and parallel PSO on RPF for evaluating the performance of our proposed framework. The evaluation results showed that RPF can predict the number of VM accurately and can be adopted by data intensive algorithms. In our future work, we will propose a VM allocation model to cooperate with RPF which could efficiently utilize the resources of a data center.

Acknowledgment

Y.-C. Tseng's research is co-sponsored by MoE ATU Plan, by NSC grants 98-2219-E-009-019, 98-2219-E-009-005, and 99-2218-E-009-005, by AS-102-TP-A06 of Academia Sinica, by ITRI, Taiwan, and by D-Link. This work was also conducted under the "AMI Enhancement Project" of III, which is subsidized by MoEA, ROC.

References

[1] H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Chang, "PFP: Parallel FP-growth for query recommendation," in *Proc. on ACM Conference on Recommender Systems (RecSys)*, 2008.

[2] W. Zhao, H. Ma, and Q. He, "Parallel K-Means Clustering Based on MapReduce," in *Proc. of International Conference on Cloud Computing Technology and Science (CloudCom)*, 2009.

[3] A. W. McNabb, C. K. Monson, and K. D. Seppi, "Parallel PSO using mapreduce," in *Proc. of IEEE Congress on Evolutionary Computation (CEC)*, 2007, pp. 7–14.

[4] S. K. Garg, S. K. Gopalaiyengar, and R. Buyya, "SLA-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter," in *Proc. of International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP)*, 2011.

[5] P. Xiong, Y. Chi, S. Zhu, H. J. Moon, C. Pu, and H. Hacigumus, "Intelligent management of virtualized resources for database systems in cloud environment," in *Proc. of IEEE International Conference on Data Engineering (ICDE)*, 2011.

[6] G. Reig, J. Alonso, and J. Guitart, "Prediction of job resource requirements for deadline schedulers to manage high-level SLAs on the cloud," in *Proc. of IEEE International Symposium on Network Computing and Applications (NCA)*, 2010.

[7] F. Tian and K. Chen, "Towards optimal resource provisioning for running mapreduce programs in public clouds," in *Proc. of IEEE Conference on Cloud Computing (CLOUD)*, 2011.

[8] G. A. F. Seber and C. J. Wild, *Nonlinear Regression*. John Wiley & Sons, 2003.

[9] J. Luan, "Data mining and its applications in higher education," *New Directions for Institutional Research*, vol. 2002, no. 113, pp. 17–36, 2002.

[10] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *Proc. of ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2000, pp. 1–12.

[11] S. Owen, R. Anil, T. Dunning, and E. Friedman, *Mahout in Action*. Manning Publications Co., 2011.

[12] "Movielens data sets," http://www.grouplens.org/node/7.

[13] "The BigCross datasets," http://www.cs.uni-paderborn.de/en/fachgebiete/ag-bloemer/research/clustering/streamkmpp.

[14] "UC Irvine Machine Learning Repository," http://archive.ics.uci.edu/ml/databases/census1990/.

[15] M. Shindler, A. Wong, and A. Meyerson, "Fast and Accurate k-means For Large Datasets," in *Proc. of Twenty-Fifth Annual Conference on Neural Information Processing Systems (NIPS)*, 2011.

[16] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. of IEEE International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.

[17] M. McKay, R. Beckman, and W. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

978-1-4673-4510-1/12/$31.00 ©2012 IEEE