Performance Modeling on the Basis of Application Type in Virtualized Environments

Fanxin Meng
School of Mechanical, Electrical and Information Engineering
Shandong University, Weihai.
Weihai, 264209, P.R. China.
E-mail: xinfanmeng@163.com

[1]Guangyu Du, [1,2]Hong He *, [1]Shengzhong Yuan
[1]School of Mechanical, Electrical and Information Engineering
Shandong University, Weihai.
Weihai, 264209, P.R. China.
[2]Institute of Software of Chinese Academy of Sciences,
Beijing. 100000, P.R. China.
E-mail: hehong@sdu.edu.cn

***Abstract*—Virtualization technology plays an essential role in resource in modern large data centers while it also causes interference among virtual machines which co-located in common physical machine contending for the shared physical resources. In this paper, we study the performance prediction models in virtualized environment. Unclassified model developed from all types of applications is quite inaccuracy to predict performance of test applications as it is too general. We respectively develop models for each type of applications classified by the resources that they use. See5/C5 technology is used to determine the type of test application before executing its corresponding performance prediction model and linear regression technique is adopt to develop performance prediction models. Finally, comparing classified models with the unclassified one, the former get 0.038 average prediction errors for test applications while unclassified model arrives 0.609 average prediction errors.**

***Index Terms*—performance modeling, performance prediction, interference, virtualization, Liner regression**

I. INTRODUCTION

As the rapid development of computer hardware, virtualization technology would become the focusing research of the next-generation data center. With the ability of resources sharing, isolation and live migration, virtualization technology [1] not only maximizes the performance of the hardware, but also makes a contribution to saving energy, improving operation efficiency and reducing economic cost when deploying enterprises to data centers. In virtualized environments, multiple virtual machines (VMs) are created on a physical host and run at the same time. Ideally, we hope each virtual machine (VM) Co-located on one host can achieve the best performance as the VM owns the host resources alone. However, the performance of the VM is affected by the behavior of other VMs in its neighbors due to the shared usage of resources in the system and is sensitive to the creation of new VM. Ku et al. [2] analyze the creation overhead of VM by implementing an automated measurement method and indicate that the creation of VM generates significant overhead for initiating virtual hardware resources.

In this work, we use Xen [1] [3] which create multiple VMs and process them in parallel on a single physical machine as our virtual platform. Two VMs (VM1 and VM2) are created in a physical host using Xen hypervisor [4]. Each VM domain runs one of benchmark applications. The benchmark application running in VM1 is the application that we want to predict performance. We aim to get system-level characteristics values which can be collected when VMs run in a host and apply them to build interference performance models. Using Xen hypervisor, we assign applications to each VM and measure the resource utilization via the existing system tools, for example, xentop and iosta. However, in reality, more than two VMs will be created on a host and even more applications will be running in one VM. In order to understand the interference that one type of applications suffered from other type of applications better, we only study the interference produced by two VMs which are created on a host. Four kinds of experiments are done for the four types of benchmark applications respectively. In order to obtain sufficient data to train models, we do multiple tests for each kind of experiment by means of changing applications. For each experiment, different types of applications are assigned to VMs to capture the variance of interference. The type of benchmark applications running in both VM1 and VM2 need to be known when we collect data for training models, but there is no need to know the type of applications running

*Corresponding author: Hong He (hehong@sdu.edu.cn)

doi:10.4304/jsw.8.11.2847-2854

in VM2 when applied models while to know the characteristics of application running in VM1 is necessary. All applications used in our experiment are derived from standard benchmark suites. We will describe these benchmark applications used in our experiment in detail in section II. In experiments, we find that applications belonging to the same type would show similar characteristics under different interference. However, different types of applications show diverse characteristics under same interference.

Former researchers have gotten some achievements in the respect of performance prediction modeling in virtualized environments. Kraft et al. [5] refer to conclusion of [3] and propose the method based on queuing network model and disk scheduling algorithm to predict disk IO response time when multiple guest VMs coexist. Govindan et al. [6] apply the models based on historical data to predict the number of I/O event of virtual machine instance and select the virtual machine with the most onerous I/O events in priority when scheduling. Gupta et al. [7] use XenMon tools to monitor the CPU utilization of the gust domain and device driver domain in the Xen platform and use the data to improve virtual scheduling algorithm to make it more equitable. [8] use RUBiS [9] as experimental basis and compare difference of two virtualization technologies Xen and OpenVZ by analyzing application performance, resource usage, the underlying system data and point out that applications have more performance loss and high LLC (Last Level Cache) loss rate in Xen. Kudu et al. [10] used artificial neural networks to develop models to predict the performance of applications. Wood et al. [11] attempted to predict the application resource overhead in virtualized environments basing on the application resource usage characteristics in traditional platform by regression-based technology. Pu et al. [12] studied the performance status of interactive network application in virtualized platform and pointed out that the combination of multiple NET-intensive applications in VMs co-located in the same host has serious interference, while the combination of CPU intensive application and NET-intensive application shows better performance score. Paper [13] has developed the unclassified regression model for different types of applications under different interference to predict the performance drop, but it didn't consider the variance of application's resources usage patterns enough. For most of test applications, this model is too general to predict performance very accuracy.

The main works in this paper include three parts: Firstly, we generate decision tree by analyzing the relationship between application type and application characteristics to determine the type of applications. Secondly, classified models for each of application types and unclassified model for all the application types are developed. At last compare the classified models with unclassified model, we find that the former has higher prediction accuracy than the latter. The rest of the paper is organized as follows: Section II introduces the types of applications used in our experiments and presents the classification method of predicting type of applications. Then we develop the performance prediction models using linear regression technology in Section III, and conduct the models training in section IV. We summarize the paper and works in Section V.

II. CLASSIFY APPLICATIONS

We need to predict the type of one application basing on the variance of resource consumption before choosing corresponding performance model built in Section IV when the application is going to conduct new start or live migration. In this section, we develop the classification decision tree about application's type by system-level characteristics.

A. Applications and System-level Characteristics

We have known that the resource usage patterns of different types of applications are different. According to the differences of application resource usage, we classify benchmark applications into CPU intensive application, IO intensive application, Memory intensive application and Mixed application. CPU intensive application is the application which mainly consumes CPU resource. CPU resource is relatively easy to control because they are sliced into multiple virtual CPUs and assigned to each VM exactly. Most of the computation-intensive applications belong to CPU intensive application. Compared to CPU intensive applications, I/O intensive applications perform large numbers of IO operations at running time. However, the I/O operations from all the virtual guest domains are firstly sent to domain0 and then communicate to I/O devices through domain0. Therefore, the privilege domain (domain0) becomes bottleneck area. Most of the applications based on Web Service and applications based on database are I/O intensive applications. As the name suggests, Memory intensive applications require lots of memory. Due to the memory size limits on a host, they perform frequent memory switches to meet their demand. For Xen virtualization, the memory allocation mechanism for each VM is shadow page table technology. Multiple Memory intensive applications used in our experiment are derived from SPCE CPU2000 benchmark. Mixed application demands for all types of resources and do not show crazy demand to special resources. Applications such as 3D graphics rendering tools Povray [9], Ccrypt [15] and decrypt applications are the mixed applications. In order to describe the behavior of applications better, we collect their system-level characteristic when they are running under interference. In order to describe the behavior of applications and modeling the performance of applications, we collect their system-level characteristic when they are running under interference.

As applications running, we collect the system characteristic parameters at some time points by xentop and iostat tools installed in the VMs using to predict the behavior of application. These parameters collected by researchers reflect the resources usage behavior of the applications. They are sensitive to the behavior of other VMs co-located on a same host which means they are a group of vital important parameters to predict application performance under interference. 10 kinds of different characteristic parameters collected in our experiment are as follows: cpuutil (CPU average utilization), cachehits and cachemisses per second, vmswitches (Virtual machine switches per second) and novmswitches (Virtual machine no switches per second), reads_issued and writes_issued (reads and writes issued per second),

time_reading and time_writting (Disk reading and writing time per VM), blocks (I/O blocks per second).

The data appears in our paper are all normalized. Interference among applications can be perceived as application performance changes, which means we can predict application's performance to indicate the performance interference among applications. A lot of metrics can be used to measure the performance of an application, such as the complete runtime and I/O throughput which is determined by the type of application. Because workloads always have different size and different operations, we combine the relative complete time and IO throughput to measure the performance of the application. Then we get the performance score of application.

B. Classification for Applications

In this section, we construct the decision tree by benchmark applications system-level characteristics to predict the type of test applications which are running in interference. For the virtual application which is asked to predict its performance, we would choose uncorrelated model about its type if we have known nothing about its type, thus leading to the bad performance prediction results. Using system-level characteristics to the decision tree, we can predict the type of application accurately and don't influence the performance of system significantly at the same time. We adopt decision tree classification method which is usually used in data mining to develop our classification rules. Other popular classification methods in data mining such as neural network and Bayesian [20] are not appropriate to our work. Neural networks require multiple iterations to get the final model, which would result in high cost. Bayesian method requires the knowledge of related fields and distribution information of classes.

The classic classification and prediction software See5/C5 developed by Rulequest Research Company [16] is chosen to train our classification decision tree. It is called See5 and C5 when it runs in Windows and UNIX operating system respectively. In this work, we use it in Windows platform. See5/C5 algorithm is developed from decision-tree classification algorithm C4.5 which can generate both decision-tree and the rule sets. The advantage of See5 is having a high speed and taking up less memory capacity to generate a rule set for large data sets, especially for the data with continuous variables. See5/C5 algorithm also has a strong advantage in accuracy, computational speed, robustness, scalability and decidability.

According to the characteristic values of benchmark applications, we construct the decision tree using See5/C5 method. The parameters of each type of benchmark applications under interference are all used to train the decision tree. Finally, the decision three are created as Figure.1. Benchmark applications are classified four types basing on their characteristics. According to the decision tree, we can transform them to rules for each application types. We notice that only four parameters are selected to construct the decision tree. They are chosen from all of parameters basing on their contribution to the classification. Zhang et al. [17] propose a modeling parameters optimization method MPO-SMD with the basic modeling idea of machine learning, which is applied to search for the optimal modeling parameters that are used to develop models. But here See5 method use information entropy to search for the optimal modeling parameters. It makes an automatic selection around all of parameters which collected for benchmark applications and four parameters are chosen by the decision tree model at last. Nevertheless, each of these four characteristics also has different effect on the decision tree. The blocks utilization is 100% to the classification, the properties of cachehits, read_issued and cachemisses utilization are 74%, 53% and 30% respectively. Based on the decision tree, we can classify the benchmark applications exactly. Table1 shows the characteristic values of several test applications and the last column in the table gives the predicted results (possibilities that the application belongs to each type). For example, the possibility that the test application of CPU_1 belongs to CPU intensive type is 0.98, while to the Mixed and I/O intensive type are 0.02. We define the type which possesses the maximum prediction possibility as the application's type.

```
Decision tree:
    blocks >= 103.2 (59.645): IO-intensive (17)
    blocks <= 16.09 (59.645):
    :...cachehits <= 312.47 (1765.755): CPU-intensive
(14)
        cachehits >= 3219.04 (1765.755):
        :...reads_issued >= 2.49 (1.975): Mixed (15)
            reads_issued <= 1.46 (1.975):
            :...cachemisses(k ) <= 5.88 (423.56): Mixed (4)
                cachemisses(k ) >= 841.24 (423.56): Memory-
                                                intensive (16)
```

Figure 1. Prediction Decision Tree about Application Type

III. PERFORMANCE PREDICTION MODELS

A. Linear Regression Models

In this article, we use the linear regression techniques to build performance prediction model. Researchers in [10] [18] point out that linear regression model can't predict performance score of applications accurately. In this work, we mainly want to show the advantage of classified models and don't care much about the modeling method. In the next part, we still choose the less costly linear regression technology to construct our models.

TABLE I.

CHARACTERISTIC VALUES AND PREDICTION RESULT

Application	cpuutil	cachehits	cache-misses	vmswitches	covm-switches	Pre-result
CPU_1	0.62	57.32	2.97	21.31	28.76	CPU
CPU_2	0.98	36.27	6.64	42.37	33.31	CPU
IO_1	0.02	1757.52	47.37	403.74	0.00	IO
IO-2	0.07	5456.10	346.17	1402.08	0.40	IO
Memory_1	0.49	6347.31	9028.15	576.35	29.64	Memory
Memory_2	0.90	72364.29	893.62	51.36	26.31	Memory
Mixed_1	1.11	42659.34	89.47	48.58	12.75	Mixed
Mixed_2	0.70	99632.10	114.59	158.39	26.49	Mixed
	read_issue	time_ reading	writes_ issued	time_ writing	blocks	Pre-accuracy
CPU_1	0.13	0.03	0.03	0.05	1.01	0.95
CPU_2	0.03	2.31	0.36	3.51	2.32	0.95
IO_1	292.67	978.97	3.23	57.51	401.60	0.96
IO-2	347.89	1402.51	603.99	8690.10	1506.56	0.96
Memory_1	0.33	2.05	0.39	0.00	3.26	0.96
Memory_2	0.16	0.47	0.09	0.08	0.01	0.96
Mixed_1	96.67	146.78	0.37	0.19	0.72	0.96
Mixed_2	86.17	96.69	0.27	0.03	1.32	0.96

Liner regression is one kind of random models widely used in many fields, especially prediction. Two kinds of variables exist in liner regression: dependent variables and independent variables. Here, characteristics collected at one time point during the VM runtime are independent variables and the corresponding application's performance score is the dependent variables. To find the relationship between these two kinds of variables, we form a set of equations: (1) is a liner combination of the different characteristic parameters which calculates the performance score of one application

$$Y^j=\beta_0+\beta_1*M_1^j+\cdots+\beta_{10}*M_{10}^j \quad (1)$$

Where, M_i^j is the value of parameter M_i collected at the time point j. Y^j is the performance score of the application. We calculate the coefficients of the egression equations $\hat{\beta}_0,\hat{\beta}_1,\cdots,\hat{\beta}_{10}$ basing on the values collected from benchmark applications using Least Square Regression Method. The model uses the set of coefficients $\hat{\beta}_0,\hat{\beta}_1,\cdots,\hat{\beta}_{10}$ to describe the relationship between the application performance score and resource usage. Then we can predict the performance score of the test applications at the time point j using equation (2),

$$\hat{Y}^j=\hat{\beta}_0+\hat{\beta}_1*M_1^j+\cdots+\hat{\beta}_{10}*M_{10}^j \quad (2)$$

The characteristics collected at one time point would vulnerable to be influenced by accidental factors, leading to the prediction models inaccuracy. To minimize this influence, we collect characteristics several times during the running time of one application. Applying these characteristics to equation (2), we get several prediction results of the application and take the average. The actual performance score and predicted performance score of benchmarks satisfy the Least Square Regression Method [19] to minimize the error.

To create one model that only uses significance property and has the ability to avoid overfitting data, we use Stepwise Linear Regression Techniques to determine the best set of attributes to predict the application performance. Stepwise linear regression techniques do a significance test to regression coefficient of the independent variables and remove the variables which present no significant effect from the remaining variables until all variables in the regression coefficient are essential. We will introduce its application when training models in section IV.

B. Applying Models

Once the models are built, they will be applied to predict the performance of test applications. We firstly collect characteristic when a test application running under interference generated from other VMs co-located on a host. Then we apply classification decision tree to these parameters to predict application type. When the type of the application is determined, we check whether its performance prediction model has existed in the system. If the corresponding model exist, we will transfer to it, else we will start training corresponding performance prediction model through benchmark applications. After getting the performance prediction models, we adopt collected characteristics as model input and the corresponding performance score is the output.

The interference is mutual among VMs co-located on a host. Applications in other VMs are also interfered by the other applications. In order to predict the performance of all applications in a host, we wake up corresponding models to predict the performance of all applications. Based on the performance of whole VMs, we can judge whether the migration of virtual instance is successful or not. If the new application produces obvious interference on the original applications after migration, we must reconsider the migration destination host of the new application, or consider whether move out some of original applications to another VM. In this way, we could not only ensure the performance of new instances, but also ensure the performance of other virtual instances that have already assigned to the host.

IV. MODELS TRAINING AND IMPLEMENTATION

We observe that the performance score among identical type applications is not much different in our experiments.

However, applications with different types running under identical interference show completely different characteristic and performance score. According to the different resource usage patterns among different type of applications, we construct performance prediction model for each application type respectively. Before developing our classified performance models, we first develop the unclassified model using all types of benchmark applications.

A. Linear Regression Model with Unclassified Method

We firstly use linear regression technology to construct unclassified model. The regression coefficients of unclassified model and significance test of parameters before applying step regression operations are shown in Table II. In the last column of the table, Flag '***' implies this parameter extremely significant to model equation and significance gradually reduces with the reduction number of '*'. Flag '.' implies slight significance while parameters without any flags presents that it has no significance to regression equations. According to Table II, we find that most of characteristics have little contribution to the model equations. Compare the prediction results with measured performance score, we find that this model equation has low prediction accuracy to test applications. Then we further perform step regression operations to linear regression equation, deleting parameters which have little significance to equations or even affect the accuracy of model. Finally we get the step regression model in Table III which only includes five significant characteristics. Comparing prediction results of two models with measured performance score on different type applications, we find that the step regression model has been greatly improved for most applications except I/O application. This is because the parameters which affect the performance of I/O intensive applications are deleted when we perform step regression operations. Table IV shows the prediction errors of several test applications using step aggression models. Comparing performance score with the measured score, we can find that the unclassified method have high prediction errors to Memory intensive application as well as quite good performance to CPU intensive application. Considering the variance of prediction accuracy that unclassified model has shown to different types of applications, we build models for each application type respectively in the following part.

B. Linear Regression Models with Classified Method

In this part, we use liner regression technology to develop models for each application type. We have classified applications into four types based on application resource usage patterns in section II and give the quite accurate application classification decision tree. Four models are developed based on the four types of applications: CPU model, Memory model, I/O model and mixed model.

We firstly develop the model about CPU intensive application. Other models will use the same method to develop. When training the CPU model, we only use the characteristics of CPU intensive benchmark applications and use step regression method to optimize model equations. Then we choose the parameters which mainly affect the performance of CPU intensive applications through step regression technology. Table V shows the regression coefficients after being optimized for CPU intensive applications and we find that model about CPU intensive application is controlled by six parameters from collected ten characteristics. Table VI ~ Table VIII show the regression models for other types of application respectively. We can find that each model selects different parameters from ten kinds of parameters. Table IX shows the prediction errors of the same test applications in Table IV using classified models and we find that classified model has good performance over unclassified model.

TABLE II.

LINEAR REGRESSION COFFICIENTS

Parameters	Estimate	Flag
(Intercept)	4.17e-01	
cpuutil	9.59e-01	
cachehits	3.51e-06	
cachemisses	-1.08e-04	**
vmswitches	-5.47e-06	
novmswitches	6.36e-02	**
reads_issued	1.81e-02	**
time_reading	-1.01e-03	
writes_issued	7.23e-03	
time_writting	-7.37e-04	
blocks	-3.83e-04	

TABLE III.

STEP LINEAR REGRESSION COFFICIENTS

Parameters	Estimate	Flag
(Intercept)	5.13e-01	
Cachemisess	-9.10e-05	**
novmswitches	7.94e-02	***
reads_issued	1.59e-02	***
Writes_issued	9.40e-03	
time_writting	-9.59e-04	*

TABLE IV

PREDICTION ACCURACY OF UNCLASSIFIED METHOD

Test Applications	Measured score	Predicted score	Prediction Error
cpu_1	2.70	2.85	0.021
cpu_2	2.90	3.04	0.047
memory_1	0.96	2.15	1.240
memory_2	1.58	4.20	1.660
iO_1	2.31	3.71	0.607
iO_2	2.40	3.85	0.602
mixed_2	12.82	5.05	0.606
mixed_1	3.76	3.43	0.088
Average error			0.609

TABLE V.

MODEL COEFFICIENTS FOR CPU INTENSIVE APPLICATIONS

Parameters	Estimate	Flag
(Intercept)	4.729008	***
cachemisses	-0.008798	**
vmswitches	-0.033011	.
novmswitches	-0.055242	*
reads_issued	2.599196	*
writes_issued	-0.410311	.
blocks	0.602664	*

TABLE VI.

MODEL COEFFICIENTS FOR MEMORY INTENSIVE APPLICATIONS

Parameters	Estimate	Flag
(Intercept)	2.309e+00	**
cpuutil	-1.168e+00	*
cachehits	-1.811e-06	*
cachemisses	3.524e-05	*
vmswiches	-1.372e-03	*
novmswitches	-2.251e-02	.
Read_issued	4.210e-01	

TABLE VII.

MODEL COEFFICIENTS FOR MIXED INTENSIVE APPLICATIONS

Parameters	Estimate	Flag
(Intercept)	-2.657e-01	
cachehits	4.576e-05	***
vmswitches	1.709e-02	***
novmswitches	4.526e-02	***
reads_issued	1.787e-02	**
time_reading	-7.476e-03	**
writes_issued	-6.752e-02	.
blocks	3.363e-01	***

TABLE VIII.

MODEL COEFFICIENTS FOR I/O INTENSIVE APPLICATIONS

parameters	Estimate	Flag
(Intercept)	9.024e-01	***
cpuutil	-1.356e+00	**
cachehits	9.341e-05	.
read_issued	7.127e-03	*
write_issued	-1.836e-03	*

TABLE IX.

PREDICTION ACCURACY OF CLASSIFIED METHOD

Test applications	Measured score	Predicted score	Prediction Error
cpu_1	2.7	2.87	0.062
cpu_3	2.9	3.07	0.060
memory_1	0.96	0.92	0.041
memory_2	1.58	1.57	0.005
io_1	2.31	2.32	0.004
io_2	2.40	2.36	0.017
mixed_2	12.82	11.96	0.067
mixed_1	3.76	3.94	0.048
Average error		2.87	0.038

C. Comparison of Classified Models and Unclassified Model

When the classified models complete training, we apply them to test applications to verify the models' accuracy. In order to compare with unclassified method, we predict the performance of test applications which same as in Table IV using corresponding classified models. As shown in Table IX, classified model has good performance over unclassified model with 3.8162% average prediction error, which is better than 60.8942% of unclassified model. Figure.2, 3, 4, 5 compare the performance score predicted by unclassified method and classified method with measured score. As shown in Figure.2 and Figure.3, the accuracy of CPU model and I/O model is not very obvious to individual test applications but still better than unclassified model. This may be caused by external interference. However, in Figure.4 and Figure.5, Memory intensive application models and mixed application models show much better prediction accuracy than unclassified models. When we apply them to predict the performance of one application, we just need to automatic predict its type using the classification method in Section II.

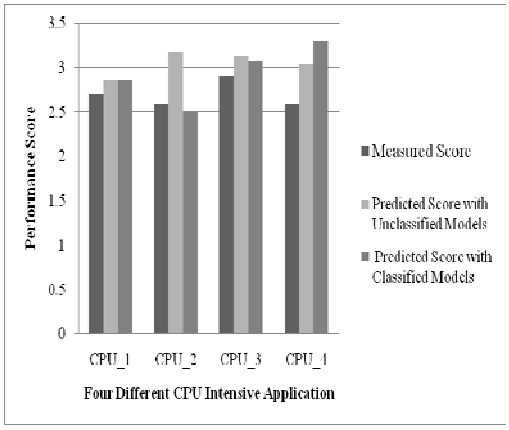

Figure 2. Performance Sore of CPU Intensive Applications with Different Models

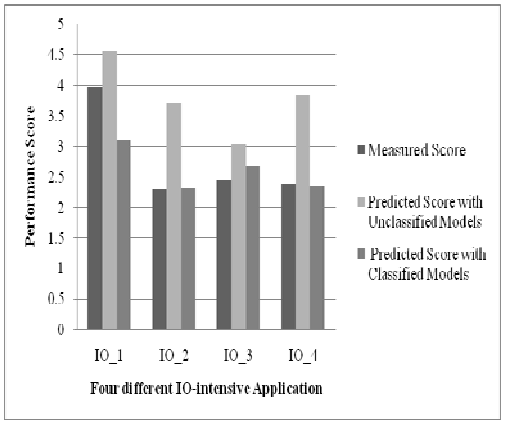

Figure 3. Performance Sore of I/O Intensive Applications with Different Models

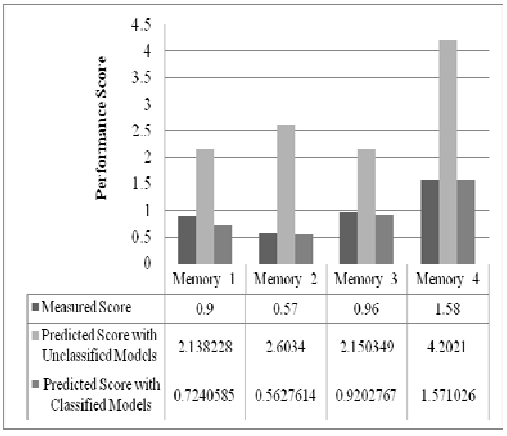

Figure 4. Performance Sore of Memory Intensive Applications with Different Models

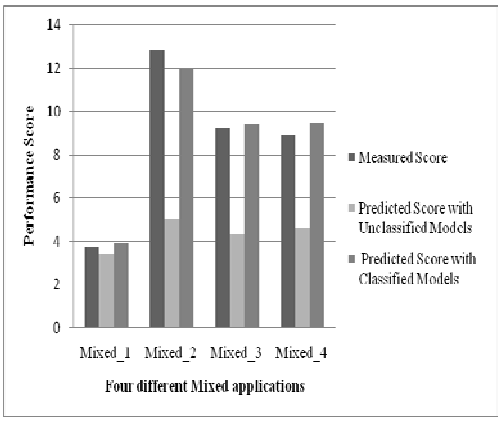

Figure 5. Performance Sore of Mixed Intensive Applications with Different Models

V. CONCLUSION

With virtualization technology widely used in data centers, research in interference among VMs co-located on a same physical machine becomes more crucial. Efficient performance prediction models are essential for the task scheduling and resource allocation architecture to achieve efficient utilization of resources and meet the maximum needs of users. In this work, we propose the classification model that is used to predict application type and the classified performance prediction models that are used to predict the performance of application basing on the application type. The classified models could conduct performance prediction quite accurate and don't care of the behavior of neighborhood applications. Compared with unclassified model, it can get 0.038 average prediction errors for test applications while unclassified model arrives 0.609 average prediction errors in the same conditions. In the future, we intend to expand our models to adapt the situation that multiple VMs created on a single physical machine and take the number of VM as a parameter of the performance prediction model. In the future, we are going to propose resource management methodology to optimize the performance in virtual computing systems like [21].

ACKNOWLEDGMENT

This work is financially supported by Graduate Innovation Foundation of Shandong University, Weihai (GIFSDUWH).

REFERENCES

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield and R.Neugebauer. Xen and the Art of Virtualization. In Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, 2003, 37(5), pp. 164~177.

[2] Mino Ku, Eunmi Choi, Dugki Min, Impact of Virtual Machine Creation Overhead on Virtualized Computing Environment, IJACT: International Journal of Advancements in Computing Technology, 2011, 3(7), pp. 326 ~ 333.

[3] The Xen™ virtual machine monitors: http://www.cl.cam.ac.uk/research/srg/netos/xen/

[4] Xen hypervisor: http://www.xen.org/products/xenhyp.html ,

[5] S. Kraft, G. Casale, D. Krishnamurthy, D. Greer, P. Kilpatrick, IO Performance Prediction in Consolidated Virtualized Environments, In Proceedings of the Second Joint WOSP/SIPEW International Conference on Performance Engineering, 2011, pp. 295~306.

[6] S. Govindan, A. R. Nath, A. Das, B. Urgaonkar, A. Sivasubramaniam, Xen and co.: communication-aware CPU scheduling for consolidated xen-based hosting platforms, In Proceedings of the 3rd international conference on Virtual execution environments, 2007, pp. 126~136.

[7] D. Gupta, L. Cherkasova, R. Gardner, A. Vahdat, Enforcing Performance Isolation Across Virtual Machines in Xen, In Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware, 2006, pp. 342~362.

[8] P. Padala, X. Zhu, Z. Wang, S. Singhal, and K. Shin, Performance evaluation of virtualization technologies for server consolidation, Tech. Rep. HPL-2007- 59.

[9] C. Amza, E. Cecchet, A. Chanda, A. Cox, S. Elnikety, R. Gil, J. Marguerite, K. Raja mani, W. Zwaenepoel, Specification and implementation of dynamic Web site benchmark, In Proceedings of the 5th IEEE Workshop Conference on Workload Characterization, 2002, pp.3~13.

[10] S. Kundu, R. Rangaswami, K. Dutta, M. Zhao, Application performance modeling in a virtualized environment, In Proceedings of 2010 IEEE 16th International Symposium on High Performance Computer Architecture, 2010, pp.1~10.

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

[12] X.Pu, L. Liu, Y. Mei, S. Sivathanu, Y. Koh, C. Pu, Understanding Performance Interference of I/O Workload in Virtualized Cloud Environment , In Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing, 2010, pp. 51~58.

[13] Y. Koh, K. Rob, B. Paul, B. Mic, Zhihua. Wen, P. Calton, An analysis of performance interference effects in virtual environments, In proceedings of ISPASS 2007 International Symposium on Performance Analysis of System & Software, 2007, pp.200~209.

[14] IOzone Filesystem Benchmark: http://www.iozone.org

[15] Ccrypt: http://ccrypt.sourceforge.net/.

[16] Rulequest: http://www.rulequest.org/.

[17] Zhang Xiaoping, Wang Yang, Liu Guixiong, Modeling Parameters Optimization for Target Localization in WSN Based on LSSVR by Sampling Measurement Data, IJACT: International Journal of Advancements in Computing Technology, 2012, 4(6) , pp. 346 ~ 357.

[18] R.C. Chiang, H.H. Huang, TRACON: Interference-Aware Scheduling for Data-Intensive Applications in Virtualized Environments, In proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, 2011, pp. 1~12.

[19] M.Arioli, S.Gratto, Linear regression models, least-squares problems, normal equations, and stopping criteria for the conjugate gradient method, Computer Physic communications, 2012, 183(11), pp.2322~2336.

[20] Zhili Zhang, Hongwei Song, Yan Li, Hao Yang, Prediction Algorithm for state Prediction Model, JOURNAL OF COMPUTERS, 2012, 7(2), pp.507~513.

[21] XiaoJun Chen, Jing Zhang, JunHuai Li, Xiang Li, A Resource Management Methodology for Collaborative Computing System over Multiple Virtual Machines, 2011, 6(11), pp.2282~2291.

Fanxin Meng received the bachelor degree in Computer science and technology from Shandong University in 2011. She is currently a postgraduate in Shandong University at Weihai. Her research interests are virtualization and applied mathematics.

Guangyu Du received the bachelor degree in Computer science and technology from Shandong University in 2011. He is currently a postgraduate in Shandong University at Weihai. His research interests are virtualization and high performance computing.

Hong He received the doctor degree in Computer software and theory from Shandong University. She is currently an associate professor in Shandong University at Weihai. Her research interests are grid computing, cloud computing and high performance computing.

Shengzhong Yuan is currently a senior engineer of Network and Information Management Center in Shandong University at Weihai. His research interests are in the areas of virtual computing and cloud computing, software engineering, college information.