A Density based Performance Prediction Model for Cloud Services

Lin Xu*, Song Zhang* and Jing Li*
*Dept. of Computer Science and Technology
University of Science and Technology of China
Hefei, Anhui, China, 230026
E-mail:linxu@mail.ustc.edu.cn

***Abstract*—Performance prediction for cloud services, as the fundamental of effective resource provision and throughput, is known to be very challenging. In practice, the lack of source code and fast response requirement invalid most existing approaches. A few studies targeting at cloud services indicate a rising trend of utilizing powerful machine learning techniques such as regression and neural networks into cloud service performance prediction. In this work, we propose a density based performance prediction model specially tailored for cloud services. This model makes full use of the advantage of machine learning techniques and provides a method to predict the performance of cloud services using historical information instead of source code. Experiments verify the feasibility of our proposed method in terms of both accuracy and efficiency.**

***Keywords*-performance prediction model; regression; neural network; machine learning**

I. Introduction

Cloud computing attracts extensive attention from both industry and academic because of its advantages in cost, stability, and flexibility. While Software-as-a-Service (SaaS) and Pay-as-You-Go scheme grant cloud computing vast flexibility, they also increase the complexity of task scheduling and quality assurance. Subsequently current *de facto* service models still stick to the hourly renting model, which still assigns the resources in unit of virtual machines, and restricts the system throughput of the cloud hosting system.

However, this situation would be dramatically improved by two methods. One is to adjust the executing order of tasks according to their predicted runtime, which aims at maximizing the resource utilization. The other is to change the traditional resource provisioning pattern. For instance, the resources such as CPU time and memory could be allocated in unit of tasks rather than virtual machines. That is, instead of requiring the users calculating how many resources the tasks need and renting virtual machines accordingly, the cloud computing system is expected to predict the performance and optimize the system throughput with a reasonable turn-around time automatically. Anyway, performance estimation is a challenging problem, on which researchers have spent decades, with different input from CPU level [20], compiler level [6] or system level [19].

To make a reliable estimation on the key metrics such as expected runtime, memory utilization and CPU utilization etc. a comprehensive model is necessary based on the descriptive factors or *features* of the task. But viewing from a cloud computing perspective, new challenges and opportunities are posed on the traditional performance prediction problem. First, CPU or compiler level approaches may not fit the cloud scenario because source code required by the compiler based approaches or special hardware required by the CPU based approaches are often unavailable in the cloud. Second, to support the further processing such as resource provisioning and scheduling, performance prediction of cloud services does not require perfect accuracy, but is expected to be an online and responsive algorithm. In addition, existing approaches have not provided a clear model between the extracted features (other than those from the source code) and performance metrics, thus restricting their applications in practice.

Motivated by these three differences of the performance prediction of cloud services, in this paper we explore a machine learning based approach, which treats each task as a black box, and aims to regress the performance metrics as a function of the domain-specific *features* of the task. More specifically we use the historical instances of the same type of service as training data and treat each instance as a point in an D-dimension feature space. And then a kernel density estimator is used to predict the performance measurement of the test task by combining the information from the k Nearest Neighbors (kNNs, where k for each test task depends on the specific circumstances) in the feature space. To the best of the authors' knowledge, it is the first time a machine learning based performance prediction algorithm specially tailored for cloud computing environment is proposed. And we summarize our contributions as below:

- We propose a resource management system framework which combines performance prediction with SLA requirements to conduct the resource provisioning and task scheduling in cloud.
- We introduce an online density based performance prediction model for cloud services based on a large number of historical instances, containing (categorical and continuous) parameter values and input information each. Noticeably, larger historical information indicates more accuracy. Meanwhile, with the limit historical information in practice, our model still performs better than other traditional models.

- We collect the service information on a high performance computing platform, which provides Vienna Abinitio Simulation Package (VASP) service, to build a small dataset containing 148 instances with 18 features. Experimental results indicate the capability and performance of our proposed model, and also verify that our model is more efficient than others. Besides, a concrete situation, which may often occur in the real world, is simulated on CloudSim[4], [5] to demonstrate the importance of performance prediction models.

The remaining of the paper is organized as follows. Section II reviews relevant researches related to performance prediction and highlights the relevance of our work for cloud computing. The proposed methodology is presented in section III. In section IV, we describe the practical dataset obtained from the high performance computing platform. In section V, experiments are performed and analyzed. Finally, conclusion and future work are summarized in section VI.

II. An Review of Related Work

The problem of service performance prediction can be divided into two categories. One is to predict the performance of interactive services such as web services. Pacheco-Sanchez et. al. investigated the Markovian Arrival Processes (MAP) and MAP/MAP/1 queueing model as a tool for performance prediction to provision cloud resources [14]. WebProphet [13] employed a novel technique based on timing perturbation to automate performance prediction for web services. And Reddy et. al. simulated the web services model to analyze and predict the performance metrics [15]. The other is to predict the performance of non-interactive services such as off-line bag-of-tasks etc. In this paper, we focus on the latter, to predict the performance of non-interactive services in cloud.

To our knowledge, the oldest method to predict performence metrics of non-interactive algorithms and services is based on the analysis of source code, which gradually fades away for the reason that source code can hardly be acquired in practical situations. To solve this problem, researchers turn their attention to statistical and learning based techniques. In the middle of 1990s, Brewer in [3], [2] used linear regression models based on empirical measurements to predict the runtime of different implementations of portable, high-level libraries for multiprocessors, aiming to select the best one. Nevertheless, he only studied on the sub-quadratic-time algorithms while the linear regression models didn't work well in other cases. As what we can learn from Brewer's case, function based performance prediction models cannot be general due to the reason that different algorithms vary dramatically in terms of time complexity and we can't learn all the performance measurement through certain linear regression models. Frank Hutter [10] used both linear regression models and exact Gaussian processes to model the relevance between runtime and instance features. However, the methods only did a good job on small and continuous feature space and the computational expense was great. Ling Huang et. al. [8], [12], [17], [18] used ridge regression models to predict algorithm runtimes for its simplicity. However, the prediction performance was barely satisfactory. SVM method was proposed in [11] to predict runtime of algorithms, which was very time consuming and performed terrible. Regression Tree method introduced in [11], [1] stood out in performance prediction in comparison with other methods. Unfortunately, the performance of the regression tree method didn't hold firm and it depended on datasets to a large extent. Dodonov et. al. [16], [7] employed neural network techniques to predict performance of algorithms and services, while these methods needed a specific model to guide the learning process, which was not available for ones unfamiliar with the predicted services or algorithms.

As cloud computing appears on the horizon, researchers tend to rent machines in cloud doing their jobs instead of running them on specialized clusters. With numerous tasks of various services waiting to be executed in cloud, scheduling and provisioning policies are necessary, of which performance prediction is the fundamental. However, there are some new challenges to performance prediction in cloud computing. For the security consideration, the source code of tasks executing in cloud is not provided, which means we should estimate the performance measurements by empirical information from logs etc. Considering the consecutiveness and abundance of tasks, the predicting algorithm should be online and light-weighted. Generally speaking, the deadline is required by users in days. Consequently, wider criteria of accuracy of performance prediction can be accepted. The methods proposed by past works can not solve all the new challenges. Instead, we proposed a novel density based performance prediction model for cloud services with low time complexity.

III. Density Based Performance Prediction Model

A. The Framework

The majority of existing resource management systems in cloud are designed to schedule tasks directly while the minority take quality of service(QoS) into account. Few of them think highly of the performance prediction model. However, traditional resource management systems paying less attention to performance prediction of tasks may encounter inefficiency. Over provisioned resources may cause waste while inadequate resources may delay the tasks. In order to maximize both the task completion rate and system resource utilization, we precompute the resources needed by the task using performance prediction model, consider the user's SLA requirement and finally make an appropriate decision on resource provisioning. The framework of our resource management system in cloud is shown in Fig. 1.

- **Task Queue** Submitted tasks join in the task queue in order and wait to be processed.
- **Task Profiler** Analyzes the tasks and profiles them with significant features and configuration parameters.
- **Performance Predictor** Predicts the performance of tasks using approapriate performance prediction models in a performance prediction toolbox.
- **QoS Compiler** Transforms the user's QoS requirements from natural language to logical forms, such as XML.

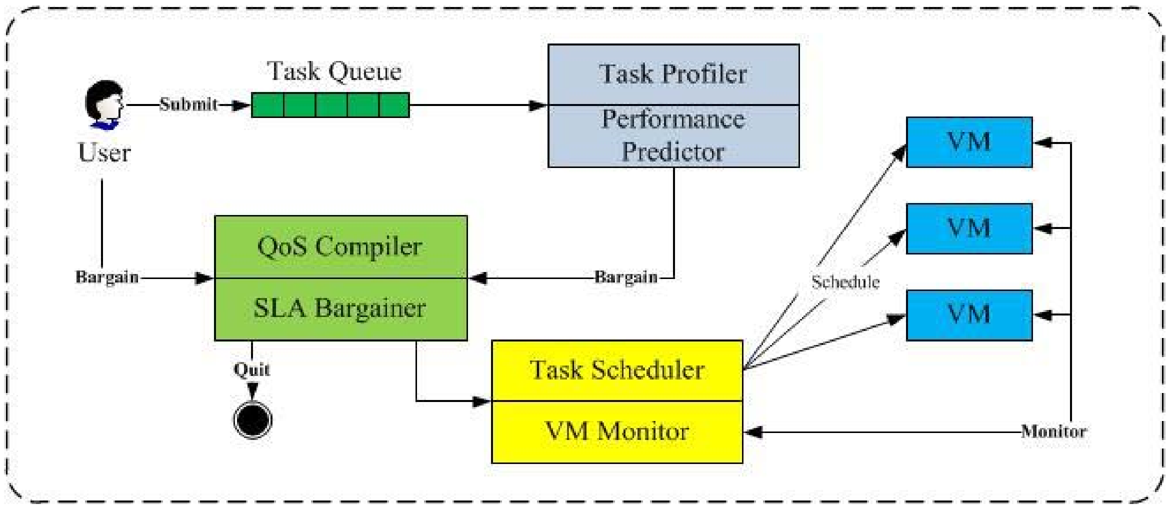

Fig. 1. Framework of resource management system in cloud. Before a task is assigned to some virtual machines, performance prediction should be made on it. The results will be compared with the user's original QoS requirements and available resources in cloud to help make SLA. If the agreements falls through, the task will be reject. Otherwise, the task will be assigned to the VMs by task scheduler. During the execution, the states of VMs and tasks are monitored to make appropriate migration policies in time.

- **SLA Bargainer** Compares the performance prediction results with the user's QoS requirements, together with the available resources in cloud, to make service level agreement(SLA).
- **Task Scheduler** Schedules the tasks to the resources tailed by performance predictor and SLA bargainer.
- **VM Monitor** Monitors the state of virtual machines(VM) and tasks in order to make appropriate migration policies in time.

B. Density based Performance Prediction Model

The rationale of our proposed density based performance prediction model is that, based on the contiguity hypothesis, we expect an instance to have similar characteristics with ones in the neighborhood. In line with such a rationale, we first find several nearest neighbors of a predicted instance and the performance prediction will be made in association with performance measurements of the neighbors. In principle, our proposed density based performance prediction model can predict any type of performance measures, such as runtime, max memory usage, communication overhead, I/O consumption etc. For brevity, we use runtime as the performance measure to evaluate our proposed model. Details will be described in this section.

We describe a task of some service as an instance by a list of m features α = [α_1,α_2,...,α_m], extracted from a feature space A, and a set of t configuration parameters θ = [θ_1,θ_2,...,θ_t], drawn from a configuration space Θ. The features describe characteristics of the given task instance, such as the size of the given task and the number of loops etc. The configurations refer to the parameters needed in the task. Both the features and the configurations are the values which can significantly affect the runtime. We define the input space as a combination of feature and configuration spaces: $I = A \times \Theta$. The performance predictions will fall into the real number field $\mathbb{R}$.

Given task instances of the same service with a feature space A and a configuration space Θ, a density based performance prediction model is an approximate progress $f : I \to \mathbb{R}$. To build the prediction model on an instance set P of the same service, we search the logs to extract the configurations $\theta_i \in \Theta$ and features $\alpha_i \in A$. Also, runtime should be extracted as y_i. We combine the m-dimension feature vector α_i and t-dimension configuration vector θ_i as a $m + t$-dimension vector, referred as predictor variables $x_i = [\alpha_i, \theta_i]$. The training data then can be represented as $\{(x_i, y_i)\}$. We use $X = [x_1; x_2; ...; x_n]$ to denote n instances and Y for the vector of runtime.

When there comes a new task instance, we extract the features and configurations to generate x_{new}. Then we increase k by 1 from 2. We denote the k nearest neighbors as $x_{n_1}, x_{n_2}, ..., x_{n_k}$ and the distance between the k nearest neighobrs and the new task instance are $d_1, d_2, ..., d_k$. The k is finally determined when the $density$

$$f(k) = \frac{k+1}{\pi \times (\max \|x_{new} - x_{n_i}\|_2^2)^2}, \quad i \in [1,k] \qquad (1)$$

first reaches the maxima.

We use the performance of nearest neighbors for reference to estimate the performance of a new task instance. However, the contribution of each neighbor to predict the performance of the new instance is different and is closely related to the Euclidean distance between the new instance and the neighbor. Suppose the distance between a new instance and one of its neighbors is d_i and the contribution of the neighbor to help predict performance of the new instance is ω_i, then

$$\begin{cases} \sum_{i=1}^{k} \omega_i \ = \ 1 & \text{(2a)} \\ \omega_i \ = \ g(d_i) & \text{(2b)} \\ g\prime(d) < 0, \quad d \geq 0, \quad \omega > 0, \quad i \in [1, k] & \text{(2c)} \end{cases}$$

Considering the constraints above and to keep things simple, we choose

$$g(d_i) \ = \ \frac{\sum_{j=1}^{k} d_j + 1 - d_i}{k + (k-1)\sum_{j=1}^{k} d_j} \tag{3}$$

Finally, our kernel density esmitor is designed as

$$\hat{y} \ = < \omega, \mathbf{y} > \tag{4}$$

where $< x, y >$ computes the dot-product of vecotor $\mathbf{x}$ and $\mathbf{y}$.

Compared with typical kernel density algorithms like Gaussian kernel, our distance based weight kernel algorithm is simple and efficient, which is more appropriate to support online processing. Besides, our model can deal with the batch processing. If there are more than one task to be processed, we use $\hat{X}$ to represent the pending tasks and $\hat{Y}$ for predicted runtime value. Detail process is shown in Algorithm 1.

Algorithm 1 Density-based Predcition Model($X, \hat{X}, Y$)

```
 1: Standardizing the instances
 2: for each instance $T$ in $\hat{X}$ do
 3:     $k = 2$;
 4:     Find $k$ nearest neighbors in $X$ of $T$
 5:     if the $density$ defined by Equation 1 begins to decrease then
 6:         Break;
 7:     else
 8:         $k++$;
 9:         Go to 4
10:     end if
11:     Find the related $Y_T$ of the $k$ nearest neighbors
12:     Compute the distance based weight vector $W_T$ using Equation 3.
13: end for
14: Combine all the row vector $W_T$ as $W$ and all the column vector $Y_T$ as $Y$
    return $\hat{Y} = W \times Y$
```

At the end of this section, we will discuss the computational and storage complexity of proposed density based performance prediction model. The computational complexity of KD-tree based nearest neighbor and range searching is $O(\sqrt{n} + p)$, where n is the number of instances and p is the number of reported instances. The computational complexity of *pdist(X)*, which is a function in Matlab for distance computing, is $O(kn^2 - kn)$, where k is a constant and n is the number of instances in X. The computational complexity of *pdist2(X,Y)*, which is also a function in Matlab for distance computing, is $O(mn)$, where m is the number of instances in X and n is the number of instances in Y. Our algorithm is comprised of two for-loops and the outer one can be parallelized. Consequently, the computational complexity of our proposed model is $O(an^2 + bn + c\sqrt{n} + d)$, where a, b, c and d are constants. For the storage complexity, the model should store all the features of history instances. Assuming n instances with d features each and each data consisting of b bits, the storage complexity is $O(ndb)$.

Our model is appropriate in cloud computing, where mass data is processed and large historical information is ready for prediction reference. Even at the very beginning of a cloud service, without abundant historical information, our prediction model still performs better than other typical models. Along with the ongoing service, larger historical information can be stored to achieve more accurate predictions.

IV. Construct a practical dataset

Before introducing the experiments made for evaluating our model, we first construct a practical dataset from our high performance computing(HPC) platform. Details are described in the following section.

A. High Performance Computing Platform

The HPC platform in the supercomputing center of USTC is comprised of Inspur NF5260M3 Rack Servers, Inspur NX5440 Blade Servers, Sugon TC4600 Blade Servers and Lenovo B710 Blade Servers, containing a login node, a storage node and 44 computing nodes totally with 704 CPU cores. The HPC platform is used to provide VASP(a package for performing ab-initio quantum-mechanical molecular dynamics simulations using pseudopotentials or the projector-augmented wave method and a plane wave basis set) and Gaussian(a computer program for computational chemistry) services. All the jobs are provisioned and scheduled by IBM Platform LSF Express 8.3.

B. Construction Process

We built the practical dataset from historical data of VASP service. First, we randomly chose 4 users from whom the valid tasks(uncompleted tasks were not included) were selected. Second, we used the command in LSF to collect the information about runtime, memory and swap usage, CPU utilization and provisioned computing nodes etc. Third, we searched the certain files for input parameters of relevant tasks, such as the number and the species of atoms, the main computing algorithm and so on. Fourth, we jointed the information obtained from step 2 and 3 to get a dataset ***vasp*** containing 148 instances with 18 dimensions. Finally, we checked integrity. The missing values were filled by defaults. What's more, strings and categorical values were encoded as real number.

V. Experiments

In this section, we designed experiments to compare typical performance prediction models introduced in section II.

Meanwhile, we also assessed the confidence of our proposed model. To evaluate these models with different datasets(i.e., varying in size, dimensionality etc.), we totally used 6 datasets ranging from the real-world one described in section IV and others obtained from [9]. Details of each dataset are presented in Table I. Finally, a concrete situation was simulated on CloudSim to confirm the significance of our perforamnce prediction model. We performed our experiments on a Lenovo B710 blade server, which has eight AMD 800MHz cores, 32GB RAM and a 100GB hard disk. All experiments are repeated with 10 random trials.

TABLE I
DESCRIPTION OF DATASETS

Dataset	vasp	Concorde	SWV	Minisat	RCW	CORLAT
Size	148	111	1245	765	1980	5840
Feature	18	65	136	135	72	106

A. Evaluation Metrics

Given a set of observed runtime values $y_1, y_2, ..., y_n$, we get predictions $\hat{y_1}, \hat{y_2}, ..., \hat{y_n}$ and variances $\sigma_1^2, \sigma_2^2, ..., \sigma_n^2$ through a prediction model. The prediction of a certain distribution allows us to assess the model's confidence. To ensure the fairness of evaluation, we use the following four metrics to synthetically study the models' confidence.

1) **Root Mean Square Error(RMSE)** a frequently used measure of the differences between values predicted by an estimator and values actually observed. RMSE is defined as $\sqrt{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2/n}$ and lower values are better.
2) **Pearson's Correlation Coefficient(PCC)** a measure of the linear correlation between two variables, giving a value between $+1$ and -1 inclusive. It is defined as $\sum_{i=1}^{n}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})/(\sqrt{\sum_{i=1}^{n}(y_i - \bar{y_i})^2}\sqrt{\sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})^2})$. Higher values are better.
3) **Log Likelihood(LL)** a measure to describe a function of a parameter given an outcome. It is defined as $\sum_{i=1}^{n} log\phi((y_i - \hat{y}_i)/\sigma)$, where ϕ denotes the probability density function of a standard normal distribution. Higher values are better.
4) **Theil's Inequality Coefficient(TIC)** a measure of how well a set of predicted values compares to a corresponding set of observed values. It is defined as $\sqrt{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}/(\sqrt{y_i^2} + \sqrt{\hat{y_i}^2})$. The closer the value is to zero, the better the prediction model.

B. Experiments on Various Models

In this section, we compare our density based performance prediction model with typical prediction models described in section II by five datasets. The evaluation measures and goodness-of-fit conditions are shown in Table II and Fig. 2.

As shown in Table II, Ridge model always executes fastest while the accuracy of the model is unacceptable. Gaussian Process(GP) and SVM methods always take a lot more time doing mediocre predictions. The neural network method, radial basis function(RBF), performs poorly in terms of accuracy and time consumption. Regression Tree(RTree) method holds unconstant performance on various datasets. Compared with other prediction models, our proposed density based performance prediction model(DWDist) is particularly noteworthy since it does a good job on accuracy and time consumption. Besides, the performance of our model is not as sensitive as other prediction models to datasets.

Fig. 2 visually reveals the performance of different prediction models. Some subplots have fewer points since negative values are ignored. It can be perceived that GP, Ridge and SVM methods sometimes predict negative values which are less consultative for subsequent provisioning or scheduling. RBF and RTree methods perform barely satisfactory but sometimes are excessively sensitive to datasets. Generally speaking, our DWDist model performs best compared with the typical prediction models despite some flaws in some cases. Taking the Concorde as an example, the performance of our model is obviously better than others; however, it is worse when compared to other datasets using our model for runtime prediction. The probable reason for the relatively terrible performance is that the instances in the dataset are sparse. According to our rale mentioned above, predicted instances with dense instances around can achieve high prediction accuracy. Otherwise, the accuracy will be influenced negatively.

C. Further Experiments on ***vasp***

The preceding indicates that our density based performance prediction model is better than typical performance models. To strengthen the thesis, we further experiment each model on ***vasp***, which is derived from the real scenario and has practical significance. The size of this dataset is relatively small. However, compared with other prediction models, our model still performs noticeably well.

TABLE III
QUANTITATIVE COMPARISION OF MODELS FOR RUNTIME PREDICTION ON ***vasp***

	RMSE	PCC	LL	TIC	TIME(s)
DWDist	**1.5784E6**	**0.8752**	**-282.7081**	0.5678	0.0313
GP	1.5890E6	0.5568	-282.7866	**0.5669**	7.78
RBF	3.0038E10	0.0427	-459.8410	1	0.41
Ridge	2.6440E6	0.2707	-462.9585	0.6959	**0.02**
SVM	8.7291E6	0.3953	-313.2204	0.8821	1092.7
RTree	2.5495E6	0.5066	-291.1613	0.6797	0.0625

The ***vasp*** dataset contains 148 instances with 18 features each. We randomly choose 18 instances for prediction and remaining as training data. Table III, Fig. 3 and Fig. 4(Predictions using RBF model aren't shown on the figure since there are too many negative values.) evaluate the prediction in terms of quantity and visual sense respectively. Intuitively, Ridge regression and SVM methods perform terrible on runtime prediction. Other methods have obvious deficiencies when predicting some instances. Fig. 5 describes the performance of remaining prediction models by bar diagram. The upper subplot shows the prediction distribution

TABLE II
QUANTITATIVE COMPARISION OF MODELS FOR RUNTIME PREDICTIONS

Dataset	Metric	DWDist	GP	RBF	Ridge	SVM	RTree
Concorde	RMSE	919.0338	1421.2	**713.5047**	4388.6	1464.9	739.222
	PCC	0.7776	0	**0.8648**	0.4159	0.3770	0.8541
	LL	-914.9111	-1.0484E34	**-910.6980**	-1.2257E3	-1.0244E3	-911.1728
	TIC	0.3338	0.5941	**0.2323**	0.8193	0.5965	0.2417
	TIME(s)	0.1094	49.06	0.47	**0.02**	147.57	0.07
SWV	RMSE	**0.3369**	89.6452	8.0396E3	2.8063E3	1.3220E3	22.9107
	PCC	**0.9556**	0.1546	-0.1656	0.2572	0.3322	0.2110
	LL	**-214.9126**	-1.183E3	-2.0822E3	-1.9126E3	-1.7222E3	-910.1124
	TIC	**0.1110**	0.9798	0.9999	0.9991	0.9983	0.9247
	TIME(s)	0.24	4.4133E3	14.85	**0.18**	1.2959E3	0.56
Minisat	RMSE	530.7906	551.3504	541.7434	1.7126E5	2.0849E3	**249.6304**
	PCC	0.9357	0.9325	0.94	-0.0793	0.4474	**0.9863**
	LL	-1.3622E3	-1.3685E3	-1.3709E3	-4.1810E3	-1.4992E3	**-1.3603E3**
	TIC	0.1567	0.1593	0.1532	0.9854	0.4739	**0.0727**
	TIME(s)	0.1719	1.1176E3	4.33	**0.13**	621.22	0.4063
RCW	RMSE	**14.1267**	47.9286	58.0378	1.2749E4	1.1313E3	29.6479
	PCC	**0.9996**	0.9985	0.9975	0.4721	0.7865	0.9994
	LL	**-1.4156E3**	-2.1321E3	-2.1393E3	-1.5806E3	-2.3883E3	-2.1381E3
	TIC	**0.0141**	0.0257	0.0307	0.9646	0.4125	0.0157
	TIME(s)	0.1719	5.9694E3	38.18	**0.09**	2.3677E4	1.7656
CORLAT	RMSE	**9.6653**	142.76974	90.3440	1.18141E6	512.7107	11.2422
	PCC	**0.9998**	0.9532	0.9826	0.0482	0.7752	0.9998
	LL	-1.4162E3	**-1.4140E3**	-1.4223E3	-1.7018E9	-1.5595E3	-1.4175E3
	TIC	**0.0096**	0.1457	0.0890	0.9998	0.3926	0.0112
	TIME(s)	0.66	4.1365E4	702.23	**0.58**	2.8658E5	5.8

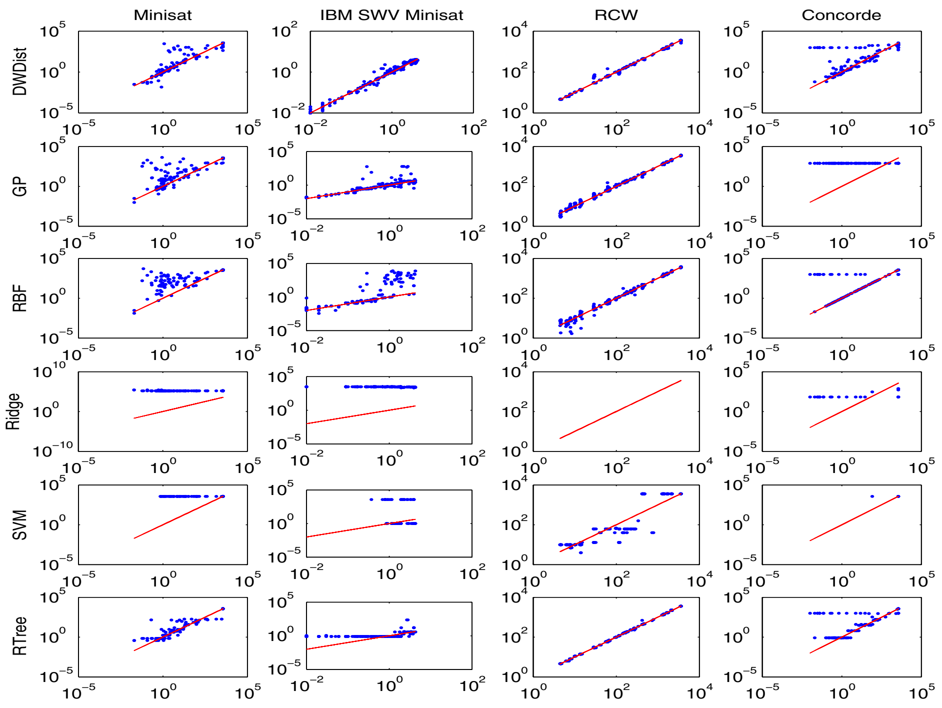

Fig. 2. Visual comparison of prediction models on previously unseen test instances. We generally choose 200 instances for prediction. For those whose size is less than 200, we use the whole dataset for prediction. The data sets used in each column are shown at the top. The prediction models used in each row are shown at the left. The x-axis of each scatter plot denotes the observed runtime and the y-axis of each scatter plot denotes the predicted runtime. Both x and y axes are logarithmic. The red line in each subplot denotes the best prediction.

comprehensively while the under subplot shows the part. The red circle surrounds nothing, which means the GP model makes a meaningless negative prediction there. Compared with regression tree model, our model does a good job in the majority. From the perspective of quantity, our density based performance prediction model wins without doubt.

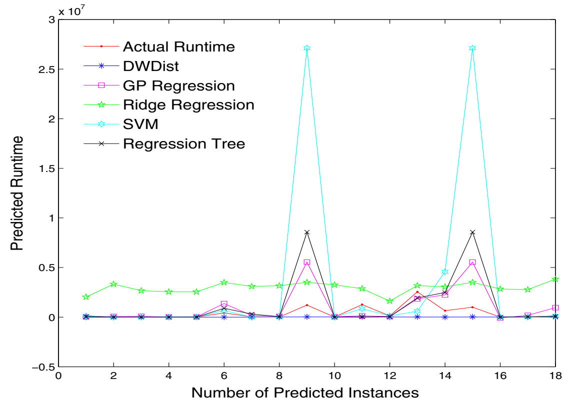

Fig. 3. Visual comparison of 5 models for runtime prediction on 18 instances.

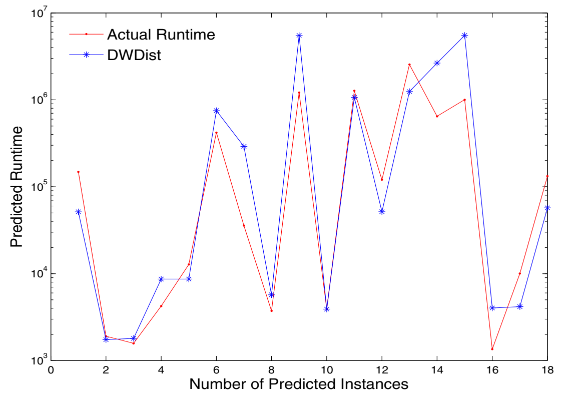

Fig. 4. Part of Fig. 3. The y-axis is logarithmic.

D. Simulation on CloudSim

CloudSim is an open source simulation framework which enables seamless modeling, simulation and experimentation of cloud computing infrastructures and application sevices. More specially, CloudSim supports for modeling cloud computing infrastructures, including data centers, hosts and virtual machines. Meanwhile, important concepts in cloud services like data centers, service brokers, scheduling and allocation policies are fine defined in CloudSim. Last but not the least, CloudSim has the ability to control virualization engine and flexibly switch between space-shared and time-shared allocation of process cores to virtualized services[5]. Researchers are keen to study virtual machine migrations and scheduling

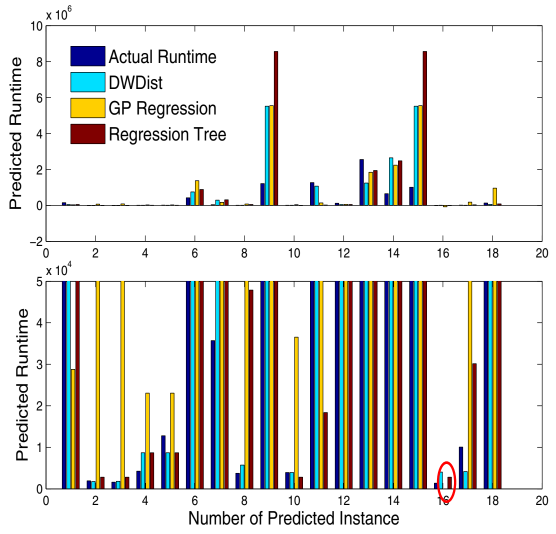

Fig. 5. Visual comparision of 3 models for runtime prediction on 18 instances.

policies using CloudSim, which can significantly cut down the cost and is easy to implement.

In this paper, we use CloudSim to simulate a realistic scenario possibly occuring in cloud computing environments. Suppose that a private cloud with limited computing resources is excepted to process abundant tasks. All the tasks should wait in line for processing. Unfortunately, the first task to be processed is so time-consuming that lots of tasks may delay if we use the policy of first come first served. Nevertheless, if we estimate the runtime of each task using the performance prediction model and reject or adjust the order of time-consuming tasks, the users' satisfaction will be dramatically improved. Concretely, we constructed a data center with one host. A virtual machine with 16 processors was created on the host. We chose 30 vasp tasks in our HPC platform to build workload trace. First, we used the policy of first come first served to run the workload, and various deadlines were defined to observed the completion rate of all tasks at the same time. Second, we predicted the runtime of each task before running the workload and positioned the time-consuming tasks to the end of the task queue. The completion rate of tasks under these two methods is shown on Fig. 6.

As is exhibited in the figure, there is a rapid increase on completion rate of tasks under PR scheduling policy when the deadlines are relatively short. This is mainly because we resort the task queue and process the shorter time-consuming tasks first. The curve of PR policy rises slightly after 50000s and sometimes remains stable, which means time-consuming tasks start to run after 50000s. As the curve of FCFS, it rises considerably from 130000s to 160000s, which means shorter time-consuming tasks are processed during this period. Overall, the completion rate of tasks shown by PR curve is higher than that shown by FCFS curve. However, the two

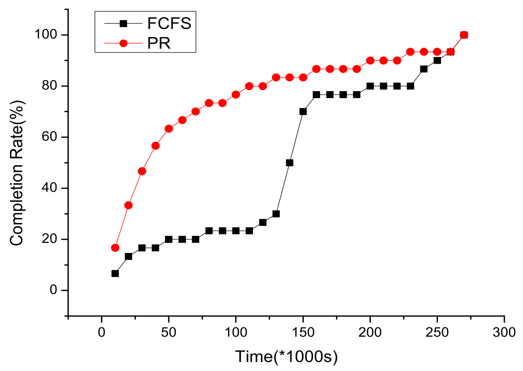

Fig. 6. Completion rate of tasks under FCFS and PR scheduling policies. FCFS means first come first served and PR means resorting the tasks queue according to the performance prediction results of tasks.

curves join together finally, since the completion rate is 100% when the deadline is long enough. In the practical situation, the deadlines cannot be defined that long. As a consequence, predicting the performance of tasks and rearranging the tasks executing order are very meaningful.

VI. Conclusion & Future Work

Performance prediction is important for many applications. Making effective resource provisioning and scheduling policies in cloud depends on a good prediction of performance of tasks. Nevertheless, the requirements of performance prediction models for cloud services are slightly different from the traditional models. Cloud services are usually from the third party and not open source. The relationship between the input(containing the system configuration parameters) of cloud services and performance measurements is ambiguous. What's more, compared with accuracy, time consumption of the prediction model is the critical consideration. Most of the previous researches on performance prediction models for cloud services have flaws either in accuracy or time consumption. In this paper, we design a density based performance prediction model, which can predict performance of cloud services well in terms of accuracy and time consumption. Furthermore, we also build a practical dataset and experiment various performance prediction models on this dataset. Experimental results demonstrate that our proposed performance prediction model can achieve satisfactory results with a few time consumption in practical situations. Thus, our density based performance prediction is preferable for cloud services.

Although our model performs better than other typical performance prediction models, it's still far from perfect in accuracy. In our future work, we are going to improve the density kernel algorithm to achieve higher accuracy. Furthermore, we are going to apply the density based prediction model to resource provision and scheduling on LSF to help increase system resource utilization.

References

[1] T. Bartz-Beielstein and S. Markon. Tuning search algorithms for real-world applications: a regression tree based approach. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 1, pages 1111–1118 Vol.1, 2004.

[2] Eric A. Brewer. High-level optimization via automated statistical modeling. *SIGPLAN Not.*, 30(8):80–91, August 1995.

[3] Eric Allen Brewer. Portable high-performance supercomputing: high-level platform-dependent optimization. Cambridge, MA, USA, 1995. Massachusetts Institute of Technology. Not available from Univ. Microfilms Int.

[4] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, C&#x00e9;sar A. F. De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper.*, 41(1):23–50, January 2011.

[5] Rodrigo N. Calheiros, Rajiv Ranjan, Cesar A. F. De Rose, and Rajkumar Buyya. Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services. *CoRR*, abs/0903.2525, 2009.

[6] Calin Cascaval, Luiz De Rose, David A. Padua, and Daniel A. Reed. Compile-time based performance prediction. In *Proceedings of the 12th International Workshop on Languages and Compilers for Parallel Computing*, LCPC '99, pages 365–379, 2000.

[7] Evgueni Dodonov and Rodrigo Fernandes de Mello. A novel approach for distributed application scheduling based on prediction of communication events. *Future Gener. Comput. Syst.*, 26(5):740–752, May 2010.

[8] Ling Huang, Jinzhu Jia, Bin Yu, Byung-Gon Chun, Petros Maniatis, and Mayur Naik. Predicting execution time of computer programs using sparse polynomial regression. In *NIPS*, pages 883–891, 2010.

[9] Frank Hutter. SAT/MIP/TSP download page. http://www.cs.ubc.ca/labs/beta/Projects/EPMs/dataset.tar.gz, November 2012.

[10] Frank Hutter, Youssef Hamadi, Holger H. Hoos, and Kevin Leyton-Brown. Performance prediction and automated tuning of randomized and parametric algorithms. In *CP*, pages 213–228, 2006.

[11] Frank Hutter, Lin Xu, Holger H. Hoos, and Kevin Leyton-Brown. Algorithm runtime prediction: The state of the art. *CoRR*, abs/1211.0906, 2012.

[12] Kevin Leyton-Brown, Eugene Nudelman, and Yoav Shoham. Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, CP '02, pages 556–572, 2002.

[13] Zhichun Li, Ming Zhang, Zhaosheng Zhu, Yan Chen, Albert Greenberg, and Yi-Min Wang. Webprophet: automating performance prediction for web services. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI'10, pages 10–10, 2010.

[14] Sergio Pacheco-Sanchez, Giuliano Casale, Bryan W. Scotney, Sally I. McClean, Gerard P. Parr, and Stephen Dawson. Markovian workload characterization for qos prediction in the cloud. In *IEEE CLOUD'11*, pages 147–154, 2011.

[15] Ch. Ram Mohan Reddy, D. Evangelin Geetha, K. G. Srinivasa, T. V. Suresh Kumar, and K. Rajani Kanth. Early performance prediction of web services. *CoRR*, abs/1201.2034, 2012.

[16] Kate Smith-Miles and Leo Lopes. Generalising algorithm performance in instance space: a timetabling case study. In *Proceedings of the 5th international conference on Learning and Intelligent Optimization*, LION'05, pages 524–538, 2011.

[17] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Satzilla-07: the design and analysis of an algorithm portfolio for sat. In *Proceedings of the 13th international conference on Principles and practice of constraint programming*, CP'07, pages 712–727, 2007.

[18] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *J. Artif. Int. Res.*, 32(1):565–606, June 2008.

[19] Yuanyuan Zhang, Wei Sun, and Yasushi Inoguchi. Predict task running time in grid environments based on cpu load predictions. *Future Generation Computer Systems*, 24(6):489 – 497, 2008.

[20] Gengbin Zheng, Gunavardhan Kakulapati, and L.V. Kale. Bigsim: a parallel simulator for performance prediction of extremely large parallel machines. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, page 78, 2004.