2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications

DEARS: A Deep Learning Based Elastic and Automatic Resource Scheduling Framework for Cloud Applications

Muhammad Hassan, Haopeng Chen
School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University
Shanghai, China
{hassankhan, chen-hp}@sjtu.edu.cn

Yutong Liu
Department of Computer Science and Engineering
Shanghai Jiao Tong University
Shanghai, China
isabelleliu@sjtu.edu.cn

***Abstract*—Cloud computing paradigm supports more enterprises to provide satisfactory web services to their clients. However, the bursty and fluctuation of requests challenge the traditional resource scheduling framework. Previous strategies manage the jobs in each virtual machines (VMs) according to the derived historical utilization patterns, where the misalignment on the utilization curves may cause the resource over-prediction and over-provisioning.**

To better reduce the service latency and the above mentioned problem, we propose DEARS, a Deep learning based Elastic and Automatic Resource Scheduling framework for cloud applications. It gives a proactive and reactive strategy, where the LSTM model is pro-applied to predict the future request demand based on historical workload. The corresponding VM allocation is separately managed by restriction assessment, VM provision, and dynamic consolidation modules. Then the SLAs feedback are iteratively applied to reactively improve the performance of resource allocation. Experiments based on real-life collected data shows the feasibility and efficiency of our framework. The high accuracy of prediction contributes to a more suitable allocation. And a better trade-off between QoS and SLAs in server side is achieved compared with the baselines.

***Index Terms*—Cloud computing, resource scheduling, deep learning, SLAs**

I. Introduction

With the rapid development of web services, the crowd of users is attracted to contribute hundreds and thousands of requests online. The examples of e-commerce and world cup websites prove the heavy workload in today's cloud computing paradigm. According to the statistics of Double Eleven Shopping Festival in 2017 of Alibaba, one of the biggest e-commerce provider in China, around 0.26 million transactions and 0.33 million orders are processed per second at peak [1]. As for the number of the requests, over 8 million requests are transferred on FIFA real-time website at the start of final match [2]. The bursty and fluctuation of requests demand an efficient resource allocation strategy in cloud computing, with the delay-minimizing and cost-saving goals.

The resources in cloud computing including CPU, memory, disk, bandwidth, and so on. are shared by allocating virtual machines (VMs) in an on-demand manner. To improve the resource utilization, a robust VM provisioning can be considered automatically and adaptively based on the time-variance of workloads. Considerable researches have been devoted to effective resource allocation and management. Some of them [3]–[6] adapt the derived pattern for resource provisioning within each VM, where the same jobs share similar resource utilization patterns according to their historical utilization curves. However, these curves can be easily misaligned in time, leading to the resource over-prediction and over-provisioning [7]. Compared with them, an automatically proactive approach by forecasting future resource demand values based on demand history can scale resources dynamically and reduce the corresponding overhead (i.e., time delay or power consumption) [8].

According to this forecast, the providers of cloud services will adjust the number of rented VMs while enforcing Service-Level Agreements (SLAs) [9]. Because of the bursty or sharp decrease on workloads in practice, this uncertainty will frequently cause the violation of SLAs or unnecessary expenses with inappropriate VM number decision. So besides of the proactive prediction, the reactive adjustment on the number of VMs should also be applied based on the feedback of the SLAs violation status.

To solve the above mentioned problems, in this paper, we propose DEARS, a **D**eep learning based **E**lastic and **A**utomatic **R**esource **S**cheduling framework for cloud applications. It pro-actively and reactively arranges VMs to support high resource utilization and enforce SLAs requirements. As this paper is enhanced from our former works [2], [10], [11], the contributions of this paper can be summarized as:

- *Prediction enhancement:* Since the workloads in cloud computing paradigm trace the human web browsing patterns, the deep learning based on time series sequence of real cloud workloads can reliably predict the future resource demand [8]. In this paper, we leverage the Long Short Term Memory (LSTM) model due to its efficiency in learning long-term dependencies [12]. Our experiments also show its achievement on higher prediction accuracy compared with the regularity based prediction in [10] and restriction added

978-1-7281-1141-4/18/$31.00 ©2018 IEEE
DOI 10.1109/BDCloud.2018.00086

prediction in [11].

- *SLAs feedback enhancement:* To avoid the uncertainty in real-life resource management and deal with the bursty or dramatic decrease on workloads, we analyze the violation of SLAs after resource scheduling, and regard it as the feedback for further adjustment in the following scheduling period.
- *Target enhancement:* Our former works only target on the number of requests as the periodic workload targeted in the whole framework. In this paper, we consider both the number of requests and the transfer bytes in responses as targets, which contribute to more accurate prediction and reliable resource scheduling.

Experiments based on FIFA world cup 1998 workload traces show the feasibility and efficiency of our proposed framework. This evolving workloads containing some sudden bursts are collected over 3 months with 1.35 billion requests, which is compatible with modern workload traces of servers. We extract the number of requests and the transfer bytes in responses and group them by seconds. The RMSEs of our prediction are satisfactory, which contributing to 99.3% prediction accuracy, higher than 95% for Dynamic strategy [2] and 97.5% for AERS [10] or EAERS [11]. The violations of SLAs is 0.58% , lower than the Dynamic strategy (20%), AERS (2.5%) and EAERS (1.6%).

The remaining part of the paper is organized as follows: Section II surveys the related work and some necessary background knowledge before go in-depth. Section III describes the problem statement and the system framework. Section IV and V introduce the details of the design including data processing and prediction model design, respectively. Section VI reports the evaluation results. Finally, Section VII concludes the paper.

II. Related Works

In cloud computing, resource allocation is the processing of assigning available resources to the on demand applications. Dynamic provisioning on virtualized resources are provided where the VMs are added or removed according to the workload requirements in a fine-grained, multiplexed manner. In this section, we will survey on some related work in resource allocation of cloud computing together with some researches on workload prediction. Some necessary background knowledge are also introduced to provide convenience to the next in-depth description of our design.

A. Resource Allocation in Cloud Computing

The main goal of resource allocation in cloud computing is to provide satisfactory web services to customers, where the SLAs should be enforced in resource sharing. Compared with the traditional Topology Aware Resource Allocation (TARA) and Linear scheduling strategy, the dynamic resource allocation model is considered more flexible and robust in practice [13]. And this model is also our main concern in this paper.

One of the commonly used dynamic resource allocation model is using VMs, including virtualization technologies and skewness to realize it. The virtualization technologies are used to allocate resources based on the application demands. And the skewness is used to measure the unevenness resource utilization of a server. This solution can avoid overload and save energy, where a load prediction algorithm is designed to capture the future resource usages of applications accurately [13].

Some of the resource utilization patterns are derived inside of the VMs using job scheduling. The paper [3] proposes a priority based job scheduling in VMs. It considers a set of evaluation criteria and customized assign different weight for each criteria. The final score and the corresponding consequent ranking are used to guide the job scheduling in VMs. Although this solution claims to be faster than the traditional First Come First Service (FCFS), Robin scheduling algorithm (RR) and Min-min or Max-min algorithm, it ignores the optimal finish time of jobs as its main concern is only priority. Further enhanced, Shen et al. [4] extract elastic features from job deadline and their reserved bandwidth for job rewards. Similarly, Liu et al. [14] take power consumption into the job scheduling consideration. As the power management algorithm is facilitated by the workload prediction, this post-active strategy can save more job latency, and obviously, energy consumption to some degree. No matter how many job features are considering in scheduling, this kind of methods are building on the observation that the same jobs (or applications) share similar resource utilization patterns among VMs. This pattern is derived according to the historical utilization curves, where the same job curves on different VMs may be misaligned in time. Using such pattern for resource allocation will lead resource over-prediction and thus over-provisioning [7].

To avoid this misaligned problem, another direction of research is to allocate the resources among VMs. Our former papers are developing in this kind of solution [2], [10], [11]. The paper [2] firstly proposes a basic framework contains three modules: performance monitor module, workload forecasting module, and VM scheduler module. After collecting and tracking the resource utilization data by performance monitor module, the workload forecasting module can generate the prediction of the future workload based on these collected history. The VM scheduler module then decide the number of VM needed to be increased or removed and the corresponding mappings from VMs to PMs. The paper [10] proposes an Autonomic and Elastic Resource Scheduling framework, called AERS. It adds two more modules, including QoS feedback module and SLAs management module, to better provide the satisfactory services for customers. To accelerate the decision-making stages, the paper [11] recently gives an enhanced version on AERS, where the workloads-VMs number mapping module in AERS is removed and the restriction model is applied in the workload forecasting module in order to eliminate the delay on modeling the specific cloud application. In this paper, we further develop our work on workload forecasting module. The deep learning model training by necessary features can contribute more accurate workload prediction result than previous work, which will be introduced next.

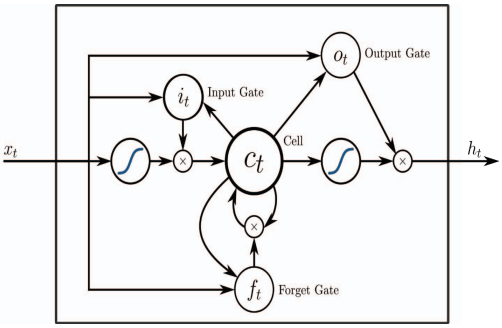


Fig. 1. A simple LSTM cell with three main gates.

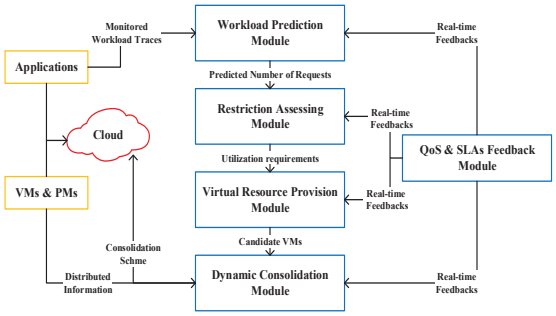


Fig. 2. The overview of the proposed framework

B. Workload Prediction in Cloud Computing

Workload traces are one of the examples of time distribution, representing the web browsing manner for each client. As mentioned at the beginning of section I, the workload will burstly increase on November 11 as the sudden increase of the Alibaba clients. Besides, tourist website requests also increase when summer vacation coming. All these examples show the time-variant feature of workload traces.

There are two main types of forecasting methods for workload prediction: prediction based on classical time series theory and prediction based on artificial intelligence. The prediction in paper [2] belongs to the first kind. It considers the workloads with periodicity but some sudden rises. The authors in this paper leverage a simple moving average method [15] and Gompertiz Curve [16] for fittings. The papers [10], [11] are all applying regularity based prediction, while the paper [11] adds the restriction model in prediction. The historical workloads of one specified hour are recorded into queues. The corresponding probability distribution is acquired from each queue and lead to the final prediction. The restriction model in paper [11] is the further processing on predicted results. After the integration of this model and regularity prediction, the required CPU cycles can be directly obtained form the module.

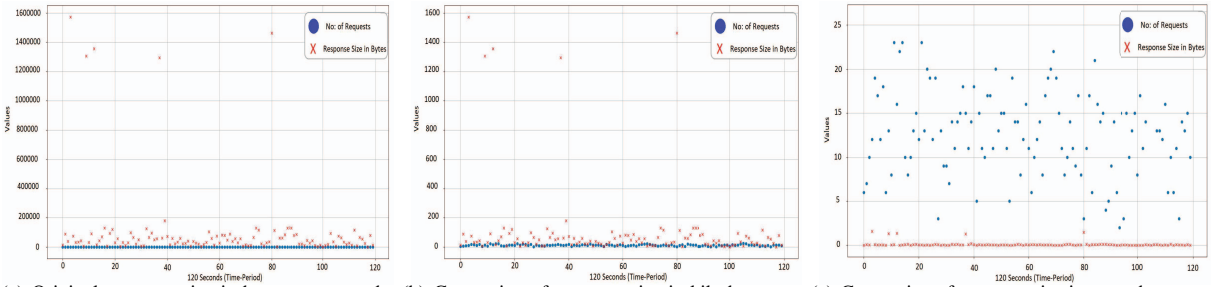
For the second kind, Qiu et al. [17] leverage a deep belief network composed of multiple-layered restricted Boltzmann machines and a regression layer. It intends to extract high level features and predict with unsupervised manner. From our observation, the prediction on the workload can benefit from the time dependency. For instance, the prediction of the next year double eleven day is not only depending on one year performance, but also the previous several years. It makes Long-Short Term Memory (LSTM) model sufficient to give more accurate prediction on workload and solve its bursty problem in cloud computing services compare with [17]. As shown in Fig. 1, the sigmoid layer running on the forget gate makes LSTM useful in sequence related problems. And its memory ability on any unimportant information gives the solution for the problem of long-term dependencies. That is the reason to choose this model in our workload prediction module.

III. Problem Statement and Framework

We denote the target server cluster as C, with P physical servers and D types of resources needed to be allocated. Applications deployed in this cluster provide services through RESTful or WebService APIs. Both the number of requests and the size of required resources contribute to the workloads for each application. So in our framework, the number of requests N_t and the response size R_t at timestamp t are the mainly concerned indexes. Actually, these two indexes represent the *state* space for a high-dimensional resource allocation framework, where the current resource utilization level of each server can be indicated [14]. As the workloads in state space fits the time-series sequence. The evaluation on state space is based on the prediction of time-series sequence, where we choose LSTM model for analysis. And meanwhile, the *action* space for this framework is the execution of resource allocation. The operations applied in action space are decided both by the predicted workload requirements, but also the SLAs feedback, with a proper weighted combination.

As shown in Fig. 2, the overview of the whole framework is consisted by five modules: workload prediction module, restriction assessing module, virtual resource provision module, dynamic consolidation module, and SLAs feedback module. The function of each module is summarized as:

- *Workload Prediction Module:* This module takes the real-time workload traces as input and predict the corresponding number of requests at the next time stamp.
- *Restriction Assessing Module:* This module takes the predicted number of requests to calculate the required CPU cycles and further assess the average utilization at VM and PM level.
- *Virtual Resource Provision Module:* This module considers the change of VM numbers by the comparison between the workload requirements and the present utilization status of current VMs.
- *Dynamic Consolidation Module:* The functions of dynamic consolidation module are directly inherited from our previous work [11], including logical clustering, PM screening, VM selection and mapping.
- *SLAs Feedback Module:* This module will monitor the real-time QoS and SLAs values after the appliance of our re-



(a) Original response size in bytes per second. (b) Conversion of response size in kilo-bytes per second. (c) Conversion of response size in mega-bytes per second.

Fig. 3. The comparison between the different down-scaling on response size.

source allocation strategy. Further adjustments of parameters in our strategy are made based on the feedbacks.

Proposed by Amazon EC2 [1], the topology of clouds is combined by region and availability zone. The region zone is partitioned by the geographical positions and the availability zone represents different data center for requests processing. The communication overhead across availability zones is caused by the increasing number of separated VMs, which is inevitable to be avoided in design. It is intuitive that to decrease this overhead, the VMs can be put into a same zone, but it will dramatically hurt the useability of the cloud system in practice.

The SLAs is another important evaluation. It indicates the reliability and effectiveness of our proposed framework. The violation of SLAs is considered in performance evaluations, which specified as the percentage of requests whose processing delay exceeded the latency requirements in client side.

To fulfill the above mentioned requirements, the design goal of our framework is to provide availability-aware, communication overhead optimized, and SLAs guaranteed resource allocation.

IV. Data Analysis and Processing

In this paper, we choose the FIFA worldcup 1998 [18] workload traces as our target dataset. This data is collected over 3 months with around 1.35 billion requests. The detail information of the raw trace data is shown in Table I. As this dataset has evolving workloads including sudden burst, the resource utilization pattern of this dataset is suitable for our concerned problem. The sudden burst workloads inside represent the sudden increasing in requests online which are unresponsive for most servers because of the limited amount of available resources, and this is also our main concern in solution design.

This workload traces are header responses by the servers for specific requests from clients. The whole trace includes 7 information blocks: IP address, login name of user, authentication name, data as timestamp, request line, status code, and response size in bytes. As this header response format is compatible with modern workload traces where the website

TABLE I
Detail information of raw data

Duration	88 days
Number of total requests	1,352,084,107
Average number of requests per minute	10,796
Total transferring bytes	4,991GB
Average transferring bytes per minute	40.8GB

services (e.g., facebook, google) are set on Apache server, the design and implementation on this target dataset can be also readily extended to other data or servers. However, not all information blocks are involving in prediction utilization, we are interested in the *timestamp* and *response size* blocks to build a time-series workload data used for deep learning prediction, and the *status code* for the efficiency checking in evaluation section.

Two necessary features need to be extracted from dataset: number of requests and the size of responses in seconds. We firstly group the raw data by seconds and then count the numbers together with sum the sizes in one second. Both of these two features will be combined as the input, and the output is these two corresponding feature values at the next timestamp.

However, the response size by default is in bytes and the unit of number of requests is 1. According to the raw data, the value of the number of request ranges from 0 to 10^3 but the value of response size ranges from 0 to 10^6. As shown in Fig. 3(a), the number of requests is at lower scales compared with the response size in bytes. Proved by the experiments, this huge scale difference will make two features (number of requests and size of response) irrelevant in prediction. To solve this problem, we standardize the response size attribute by down-scaling. We experimentally figure the mega-bytes scale can keep the closest relational similarity between two features as shown in Fig. 3(c), compared with bytes in Fig. 3(a) and kilo-bytes in Fig. 3(b). After standardizing, these two features should also be transformed between 0 to 1 to further reduce variance. The normalization is done by scikit-learn object MinMaxScaler [2]. The detailed information of the transformed

[1]https://aws.amazon.com/ec2/

[2]https://goo.gl/H3qHJU

TABLE II
DETAIL INFORMATION OF TRANSFORMED DATASET

Total number of the transformed dataset	$7.405\times10e^6$/second
Number of requests	
Minimum number	0.00/second
Maximum number	$3.488\times10e^3$/second
Transferring bytes in response	
Minimum response bytes	0.00/second
Maximum response bytes	$6.67\times10e^4$/second

TABLE III
DETAILED INFORMATION OF OUR LSTM MODEL

Name	Type	# of hidden units	# of Params.
lstm_1	Hidden layer	50	10600
dropout_1	Dropout layer	50	0
lstm_2	Hidden layer	250	30100
dropout_2	Dropout layer	250	0
flatten_1	Flatten layer	250	0
dense_1	Dense layer	2	502
activation_1	Output layer	2	0

data after processing on the original 88 days' dataset is shown in Table. II.

V. DESIGN OF PROPOSED FRAMEWORK

In this section, we will introduce the detailed design of our proposed framework.

A. Prediction Model Design

In this section, we will introduce the input transformation and specific model design for prediction.

1) Input Transformation: As the workload prediction model in deep learning is a supervised learning problem, we need to transform the input as the format of variables X and its corresponding label Y. For prediction on the number of request and response size at the next timestamp, we arrange the number of the request N_t and size of response R_t at the next timestamp t as the label of the values of N_{t-1} and R_{t-1} at the former timestamp $t-1$. For example, the time-series data is represented as $\{\mathrm{N}_1, R_1, 1\}, \{N_2, R_2, 2\}, ..., \{N_{t-1}, R_{t-1}, t-1\}, \{N_t, R_t, t\}$. The $\{N_2, R_2\}$ at time 2 will become the label of $\{\mathrm{N}_1, R_1\}$ at time 1, and combined as one sample. The rest can be done in the same manner, like the sample at time $t-1$ is the variable of $\{N_{t-1}, R_{t-1}\}$ with the label of $\{N_t, R_t\}$.

2) Model Architecture: Assume that the length of the workload traces is N, we set the size of sliding window as a fixed W $(0 < W < N)$. As shown in Fig. 4, the W_i $(0 < i < N - W)$ is the input for the prediction of W_{i+1} and so on.

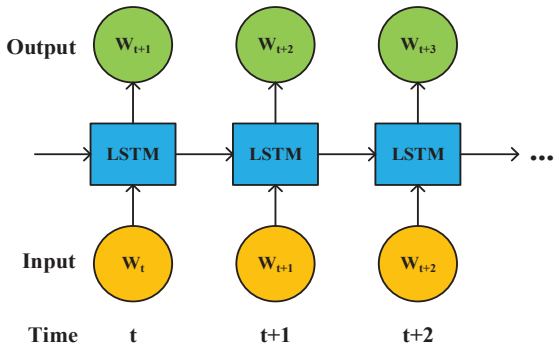


Fig. 4. The structure of LSTM model

Our model consists of one input layer of 50 neurons, one hidden layer with 250 units, and one fully connected hidden layer as an output layer with 2 units. A 0.2 dropout mask with solid probability percentage of cell units is applied to the output of every LSTM layer. The goal of this dropout mask is to remove the potential strong dependency on one dimension so as to prevent overfitting. The optimizer of our model is Adam in Tensorflow. The detailed information about the layers are summarized in Table. III including the name and type of the layer, corresponding input and output shape and the number of related numbers. In this table, the flatten layer converts the 3 dimensional input to 2 dimensional output. Dense layer is the fully connected layer with flat number of hidden units. And activation layer applies non-linear activation on the output of the dense layer.

B. Resource Allocation Algorithm Design

Before applying the predicted workload to resource allocation algorithm, the capacity of each VM should be evaluated by the CPU cycles, which is different according to various VM configurations. Taking 1.0 GHz single-core processor in VM as an example, the total 6×10^{10} CPU cycles can be handled per minute. Measured from our experiments, each request take 3×10^7 CPU cycles, which means a single-core processor can deal with at most 10^3 requests as 50% CPU utilization. Compared with the predicted number of requests, the *partition* of VM can be considered as either addition (i.e., capable requests are lower than required requests) or deduction (i.e., capable requests are higher than required requests).

Enhanced from our former work [11], we propose an improved resource allocation strategy in order to reduce the processing latency and increase the requirement satisfaction. For the reduction of the processing latency, we combine the *Virtual Resource Placement Module* into the *Dynamic Consolidation Module*. As the *Virtual Resource Placement Module* has the same function with the *Micro-management Unit* in *Dynamic Consolidation Module*, this combination can reduce the redundancy in previous framework. For the feasibility of our framework, we separate the restriction assessing unit in previous dynamic consolidation module as a single module. As mentioned before, the capability of VMs is measured by CPU cycles which are partly depending on the utilization degree. So we put this separated unit before virtual resource provision module.

For requirement satisfaction, we improve the *SLAs Feedback Module*, where the monitored SLAs measurements will be put into the consideration of utilization. In the restriction assessing module, two supply and demand relationships should be considered:

(1) *VM level relationship:* In this relationship, the system demands and the supply of VMs are analyzed. Our former work [11] only considers the CPU cycles required as demands. In this paper, we consider both the CPU cycles required and the SLAs feedback. The estimation of the average utilization on VM level can be expressed as:

$$Utilization_{avg} = \frac{CPUCyclesRequired + \alpha \cdot SLAs}{t \cdot \sum NumberOfCores_i \cdot f_i}.$$

The numerator in the function requires both the CPU cycles required calculated from the predicted workload and the SLAs feedback represented by the violation level. The corresponding weight α can be learned from multiple attempts. The dominator represents the total available VMs capability, where the $NumberOfCores_i$ and f_i are the configuration of each VM and the current frequency, respectively, together with the timestamp t. The feedback of SLAs can support the flexible adjustment on VM numbers to deal with the uncertainty in real-life. For instance, if the violation of SLAs is high when the CPU cycles required is A and the current utilization is B. Then the number of VMs should be increased in the following timestamps when the required CPU cycles and current utilization are also A and B.

(2) *PM level relationship:* This relationship is between the VMs' demands and PM's supply. The SLAs will also be applied into this level, where the utilization of PM will be estimated as:

$$Util_{PM} = \frac{t \cdot \sum NumberOfCores_i \cdot f_i \cdot Util_i + \beta \cdot SLAs}{NumberOfCores_{PM} \cdot f_{PM}}$$

The numerator represents the utilization in VMs placed on corresponding PM, while the dominator calculates the available capability of this PM. $Util_i$ is the CPU utilization measurement for each VM i. Same with α, the weight β can be learned after multiple attempts. Here, the violation of SLAs will increase/decrease the VMs' demands to fulfill the reality.

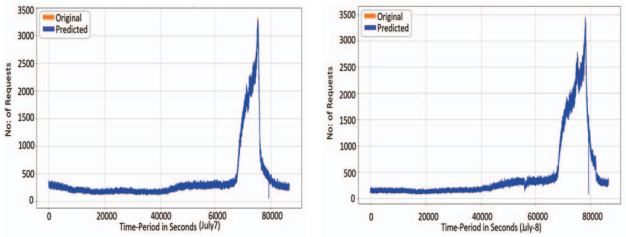
VI. Evaluation

In this section, we evaluate two parts of the performances:

- The performance of the workload prediction, especially the accuracy of the prediction model.
- The performance of the resource scheduling, including the violation of SLAs between the predicted value and the original value, to show the efficiency and robustness of our framework.

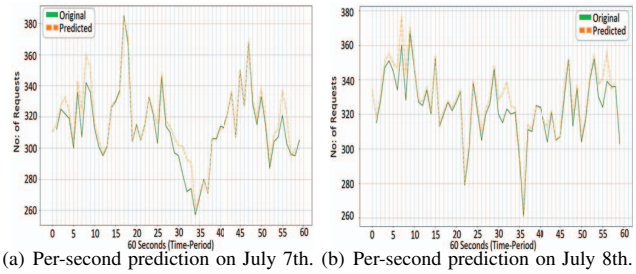
Taking as comparison, we adapt our three former works as benchmarks:

- Dynamic Virtual Resource Management with Traffic Burst [2] in 2014: This is the fundamental dynamic resource management framework, with the Gompertz Curve prediction and simply comparison for VM number decision.
- AERS [10] in 2016: It applies regularity based prediction and enhanced VM mapping framework by consolidation.
- EAERS [11] in 2017: It is the enhanced version of AERS, where the prediction is added by restriction model and framework is shrink by the parallelled processing on dynamic consolidation.



(a) Per-second prediction on July 7th. (b) Per-second prediction on July 8th.

Fig. 5. The evaluation on the number of the request per-second predictions.



(a) Per-second prediction on July 7th. (b) Per-second prediction on July 8th.

Fig. 6. The evaluation on the smaller samples of the number of the request.

A. Performances About Workload Prediction

The experiments on workload prediction are carried out using Deep Learning Framework Tensorflow version 1.8, and the LSTM model is implemented on the Python Deep learning Library Keras. The training and testing on the model are performed on a NVIDIA GTX 1080 ti GPU Computer with Ubuntu 16.04.5 LTS.

The 88 days' dataset can be divided to 2057.14 hours' data. We split the whole dataset into 80% training set and 20% testing set, with the batch size of 500,000 and epochs of 400. We take Root Mean Square Error (RMSE) [8] as the important criteria for prediction evaluation. The RMSE is calculated by the predicted values $\hat{X}$, the original value X and the length n:

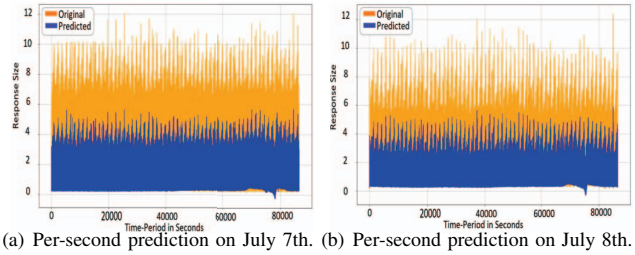
$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}\left(\hat{X}_t - X_t\right)^2}$$

1) Per-second prediction: According to the observation of our dataset, the number of the requests is highest on July 7th and July 8th, which are only included in testing set. The prediction accuracy is mainly compared on these two days.

Fig. 5 (a) and (b) show the prediction on the number of the requests. In both of the figures, we take 86,400 samples including the whole-day workload traces in July 7th and 8th. The prediction results showed in the figures is overlapping with the original values. The RMSE in Fig. 5 (a) is 30.700 and another in (b) is 34.080, which is quite satisfactory.

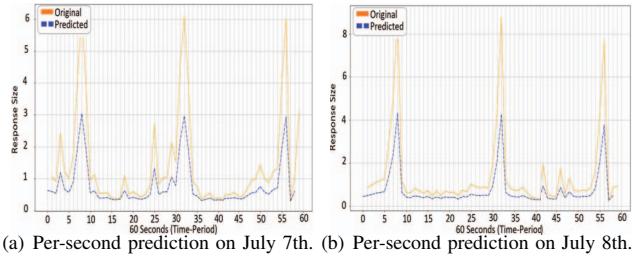
For better visualization we take 60 samples from July 7th and 8th to see the result, which are shown in Fig. 6 (a) and (b). The test RMSE on this smaller samples still performs well, indicating 23.510 in July 7th and 21.629 in July 8th.

Similarly, the evaluations on the transferring response size are also depending on whole-day workload and smaller samples on July 7th and 8th. The corresponding results are shown in Fig. 7 and 8. The Test RMSE respectively is 1.568 for Fig. 7 (a), 1.396 for Fig. 7 (b), 1.424 for Fig. 8 (a) and 1.871 for Fig. 8 (b), which are even better than the prediction of number of requests.



(a) Per-second prediction on July 7th. (b) Per-second prediction on July 8th.

Fig. 7. The evaluation on the transferring response size per-second prediction.



(a) Per-second prediction on July 7th. (b) Per-second prediction on July 8th.

Fig. 8. The evaluation on the smaller samples of the transferring response size.

2) Per-minute prediction: As our VM scheduler algorithm takes at least next 2 minutes' input on the predicted number of requests, we evaluate on the minute's prediction ability of our model. We take 60 seconds' values as input and predicts next 60 seconds two times to generate two minutes' values, where the sliding window is set to 60. The actual and predicted values are compared to calculate the RMSE on 2-mins prediction. We randomly select another two days besides July 7th and 8th for evaluation. The detailed comparisons on May 15th, June 15th, July 7th and July 8th are shown in Table IV. The RMSE is small in four evaluations, which shows the efficiency of our model that even smaller input can accurately generate more results.

3) Prediction Accuracy Comparison: The prediction method in EAERS is actually same with the one in AERS, where the restriction model only works on the predicted values rather than the predicted processing. So, the comparisons between four different workload prediction methods are shown in the first column of Fig. 9. Our deep learning methods perform better than another two predictions, as it flexibly consider the real workload status and deal with the sudden burst and dramatic decrease. It also shows a hopeful direction in cloud computing research area that the deep learning applied framework can be considered.

TABLE IV
THE EVALUATION ON PER-MINUTE PREDICTION

Date		May 15^{th}	June 15^{th}	July 7^{th}	July 8^{th}
# of requests	Original	1122	36756	17216	35575
	Predicted	1147	38517	16927	35737
Mean	Original	9.429	308.874	144.672	298.950
	Predicted	9.639	323.672	142.244	300.311
Min	Original	2.000	253.000	108.000	252.000
	Predicted	0.000	0.000	0.000	0.000
Max	Original	22.000	360.000	186.000	369.000
	Predicted	19.000	370.000	190.000	373.000
RMSE		2.085	14.798	2.428	1.361

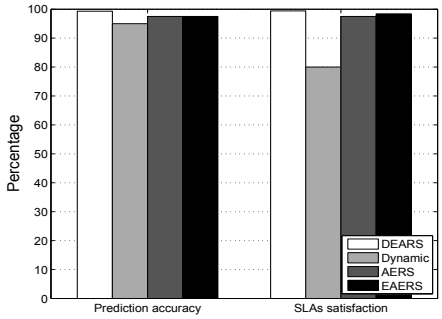


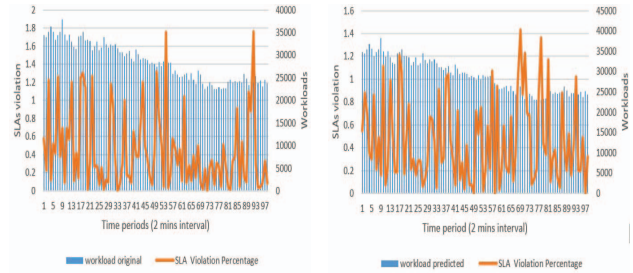
Fig. 9. The comparisons on prediction accuracy and SLAs satisfaction.

B. Performances About Resource Scheduling

To evaluate the performances about resource scheduling framework, we have created a prototype of the proposed framework in Java. The prototype is carried on cloudsim simulator [19], which requests one data center, a dynamic number of host and the required number of VM migrations as inputs. The data center is set as endless number of physical machines, and the VMs have the configurations of 2GHz CPU, 4GB memory and 10G I/O bandwidth. We adapt the July 8th's log traces as the original workload and the corresponding predicted workloads. The size of the input dataset if 720, where each value represents the total number of requests during 2 minutes' time period.

The SLAs violations are evaluated on predicted and original workloads, as shown in Fig. 10. The SLAs violations increase with the workloads, but the performance become better and better depending on the SLAs feedback contribution. To sum up, the percentage of SLAs violation for predicted value is 0.58%, while the percentage of SLA violation for original value is 0.50%. We observe that the SLA violation on original values is less than it on predicted ones. As the accuracy ratio on prediction of our model is 99.3%, we can assume that the resource scheduling on predicted workload can fulfill the real-life configurations.

The comparisons between four different resource scheduling methods are shown in the second column of Fig. 9. Our proposed DEARS framework can achieve higher SLAs satisfaction than other three benchmarks. As the Dynamic one is



(a) SLAs violation with predicted workloads (b) SLAs violation with original workloads

Fig. 10. The SLAs violation performance with two types of workloads.

the fundamental one and AERS and EAERS are all enhanced version based on this fundamental strategy, the Dynamic strategy achieve lowest performance. Mostly, the deep learning accuracy increasing contributes to the better performance on resource scheduling, where the SLAs feedback contributes on another part of performance improvement.

VII. Conclusion & Future Work

Resource allocation is an attractive topic in cloud computing. Although considerable researches have been made in this area, some of them may suffer the misalignment problem on analysis, which leads to the over-provisioning of virtual resources. To reduce the processing latency on resource scheduling and enforce the SLAs requirements, a proactive and reactive resource allocation strategy should be considered. In this paper, we propose DEARS, a deep learning based elastic and automatic resource scheduling framework. It contains five modules including workload prediction module, restriction assessing module, virtual resource provision module, dynamic consolidation module, and SLAs feedback module. Enhanced from our previous work, we improve the accuracy of workload prediction by deep learning methods based on LSTM model. And we consider both the number of requests and the transfer bytes in responses as workloads in prediction. The violation of SLAs is regarded as a feedback to adjust the number of VMs both on VM level and PM level. Experiments based on FIFA 1998 workloads prove the feasibility of our framework, containing higher prediction accuracy and better SLAs fulfilling level.

To better improve our work, we consider two aspects as future work:

- As the dataset we used is a server log files workload Traces of FIFA 1998, which is from a static website, we would like to apply deep learning technique on dynamic websites which are popular nowadays.
- In this paper, we just analyze the number of requests and the transfer bytes in responses. We would like to further classify the request types and distinguish the group of requests by their types for more specific resource scheduling.

Acknowledgement

This paper is supported by Joint Research Laboratory of Financial Information Security, Bank of China.

References

[1] BrandChannel, Alibabas 2017 11.11. global shopping festival passes $25b, https://www.brandchannel.com/2017/11/12/2017-alibaba-11-11\ \-double-11-results-111217/, november 12, 2017.
[2] H. Lu, H. Chen, S. Ma, W. Dai, P. Xing, Dynamic virtual resource management in clouds coping with traffic burst, in: IEEE SCC, Anchorage, AK, USA, 2014, pp. 590–596.
[3] S. Ghanbari, M. Othman, A priority based job scheduling algorithm in cloud computing, Procedia Engineering 50 (9) (2012) 778–785.
[4] H. Shen, L. Yu, L. Chen, Z. Li, Goodbye to fixed bandwidth reservation: Job scheduling with elastic bandwidth reservation in clouds, in: IEEE CloudCom, Hong Kong, China, 2017, pp. 1–8.
[5] L. Yu, L. Chen, Z. Cai, H. Shen, Y. Liang, Y. Pan, Stochastic load balancing for virtual resource management in datacenters, IEEE Transactions on Cloud Computing PP (99) (2016) 1–1.
[6] D. Xie, N. Ding, Y. C. Hu, R. Kompella, The only constant is change: Incorporating time-varying network reservations in data centers, in: ACM SIGCOMM, Helsinki, Finland, 2012, pp. 199–210.
[7] L. Chen, H. Shen, Considering resource demand misalignments to reduce resource over-provisioning in cloud datacenters, in: IEEE INFOCOM, Atlanta, GA, USA, 2017, pp. 1–9.
[8] C. Vazquez, R. Krishnan, E. John, Time series forecasting of cloud data center workloads for dynamic resource provisioning, Dissertations & Theses - Gradworks.
[9] L. Blasi, G. Brataas, M. Boniface, J. Butler, F. D'Andria, M. Drescher, R. Jimenez, K. Krogmann, G. Kousiouris, B. Koller, Cloud computing service level agreements – exploitation of research results.
[10] J. Sun, H. Chen, Z. Yin, Aers: An autonomic and elastic resource scheduling framework for cloud applications, in: IEEE SCC, San Francisco, CA, USA, 2016, pp. 66–73.
[11] Z. Yin, H. Chen, J. Sun, F. Hu, Eaers: An enhanced version of autonomic and elastic resource scheduling framework for cloud applications, in: IEEE CLOUD, Honolulu, HI, USA, 2017, pp. 512–519.
[12] M. Akram, C. El, Sequence to sequence weather forecasting with long short-term memory recurrent neural networks, International Journal of Computer Applications 143 (11) (2016) 7–11.
[13] N. Krishnaveni, Survey on dynamic resource allocation strategy in cloud computing environment 2 (6) (2013) 731–732.
[14] N. Liu, Z. Li, J. Xu, Z. Xu, S. Lin, Q. Qiu, J. Tang, Y. Wang, A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning, in: IEEE ICDCS, Atlanta, GA, USA, 2017, pp. 372–382.
[15] A. Ranabahu, P. Patel, A. Sheth, Service level agreement in cloud computing, Cloud Sla.
[16] C. P. Winsor, The gompertz curve as a growth curve, Proceedings of the National Academy of Sciences of the United States of America 18 (1) (1932) 1–8.
[17] F. Qiu, B. Zhang, J. Guo, A deep learning approach for vm workload prediction in the cloud, in: IEEE SNPD, Shanghai, China, 2016, pp. 319–324.
[18] M. Arlitt, T. Jin, A workload characterization study of the 1998 world cup web site, IEEE Netw 14 (3) (2000) 30–37.
[19] T. Goyal, A. Singh, A. Agrawal, Cloudsim: simulator for cloud computing infrastructure and modeling, Procedia Engineering 38 (4) (2012) 3566–3572.