Auto-Scaling Cloud Resources using LSTM and Reinforcement Learning to Guarantee Service-Level Agreements and Reduce Resource Costs

Jiang Zhong, Saisai Duan, Qing Li
Department of Computer Science and Technology, Chongqing University, Chongqing 400044, China
2692613726@qq.com

Abstract. Auto-Scaling cloud resources aim at responding to application demands by automatically scaling the compute resources at runtime to guarantee service-level agreements (SLAs) and reduce resource costs. Existing approaches often resort to predefined sets of rules to add/remove resources depending on the application usage. However, optimal adaptation rules are difficult to devise and generalize. A proactive approach is proposed to perform auto-scaling cloud resources in response to dynamic traffic changes. This paper applies Long Short-Term Memory (LSTM) to predicting the accurate number of requests in the next time and applies Reinforcement Learning (RL) to obtaining the optimal action to scale in or scale out virtual machines. To validate the proposal, experiments under two real-world workload traces are conducted, and the results show that the approach can ensure virtual machines to work steadily and can reduce SLA violations by up to 10%-30% compared with other approaches.

1. Introduction
Hundreds of thousands of users are enjoying various Internet services. The economic model of pay-per-use behind cloud computing allows companies to rent enough resources for a certain period and access them via Internet [1]. Cloud providers, Amazon EC2 and Google App Engine, offer cloud resources to the application providers in the form of Virtual Machines (VMs) with the scalability feature and pay-per-use charging model [2][7]. In particular, the application provider tends to be aware of the application environment and the required Quality of Service (QoS) [3][4]. On the one hand, the over-provisioning problem can occur when the cloud resources for a certain user exceed its demands, which consequently leads to zero Service Level Agreement (SLA) violations, but resources are wasted and as a result application provider costs arise. On the other hand, the under-provisioning problem can occur when the cloud resources are not suitable for the required QoS, which consequently results in interruption or delayed response to user requests, even worse, causes system crash. In order to find a trade-off between the SLA requirements of the application and the cost of renting resources, the resource schedule method is vital. A desirable solution would require an ability to predict the incoming workload and allocate resources ahead of time, which in turn will enable the application to be ready to handle resource provision problems when it actually occurs [5][6].

The contributions of this paper are:

1 https://github.com/Michael2397/Auto-Scaling
1) This paper proposes a model combining Long Short-Term Memory (LSTM) and Reinforcement Learning (RL) based on the concept of the autonomic computing.

2) This paper customizes LSTM to predict incoming workload using historical workload effectively.

3) This paper applies RL as a decision-maker that uses the historical resource utilization and the predicted results in order to obtain the optimal action to scale in or scale out VMs.

4) This paper conducts a series of experiments to evaluate the performance of proposed approach under real-world workload traces for different metrics compared with the other approaches.

The rest of this paper is structured as follows. Section 2 reviews prior work on auto-scaling. Section 3 provides details of the proposed approach. Section 4 presents performance evaluation and discusses the experimental results, and Section 5 concludes the paper and presents future works.

2. Related Works

There are a great number of studies have been conducted with regard to the dynamic resource provisioning, this section is an overview of existing research in the field of dynamic resource provisioning for cloud applications. IBM [8] has proposed a reference model for autonomic computing, which is called the control MAPE loop, including monitoring phase (M), analysis phase (A), planning phase (P), execution phase (E). In the monitoring phase, a monitoring component collects the information about the resources and the workload submitted by users, and this information is used to estimate future resource utilization in the analysis phase. In the planning phase, a suitable action (e.g., scale in or scale out) and how many scale VMs should be allocated is determined. Finally, the suitable action is performed in the execution phase. The four phases are executed regularly at specific time intervals. Huang et al. [9] focused on proactive analysis of scaling indicators, they made use of double exponential smoothing based resource prediction model, considering the current state of resources and history records. Mohamed et al. [10] paid attention to a purely reactive policy using the response time, due to their method is not able to cope with variations in demand without causing system instability, it faces a challenge that some delays may happen until the resource is ready for use. Messias et al. [11] proposed an adaptive prediction method combining time-series prediction models with genetic algorithms, their method did not need lots of historical data to train, because the method was able to adapt to various types of time series and adapt the extent to the incoming workload.

3. Proposed Approach

3.1. Algorithmic Framework

Algorithm I shows an overview of the algorithm’s operations: At the beginning, algorithmic initializes appropriate number of VMs based on historical experience. Then, the MAPE loop executes until there is no service running in the VM, especially, the algorithm carries out monitoring phase continuously every second, while executes analysis phase, planning phase and execution phase at specified intervals.

**Algorithm I: Algorithmic Framework**

```
1: Initialization: Initialize appropriate number of VMs based on historical experience
2: repeat every 1 second
3: Monitoring();
4: if Clock % Δt ==0 then
5: Analysis(history of request, the next time);
6: Planning(prediction value, current state);
7: Execution();
8: end if
9: end repeat
```
3.2. Monitoring phase
In monitoring phase, a monitor continuously collects information about the system, the information includes the number of requests and the CPU utilization during the ∆t interval. Finally, this information is stored in a knowledge respectively, and can be used by other phases (see algorithm II).

Algorithm II: Pseudo code for Monitoring phase
1: Begin
2: /* monitor the number of requests every second*/
3: MonitorRequest();
4: Store(workload of requests)
5: /*monitor the CPU utilization during the ∆t interval*/
6: MonitorCPU();
7: Store(workload of CPU utilization)
8: End

3.3. Analysis phase
Last phase has obtained the historical number of requests, in this phase, a workload analyzer is used to predict the number of requests in the next interval by utilizing forecasting techniques. In this paper, the long short-term memory (LSTM) is applied in order to deal with fluctuating requests. LSTM networks are a special kind of Recurrent Neural Networks (RNN), capable of learning long-term dependencies. The major feature of LSTM networks is their ability to retain and persist information for long sequences or periods of time, as a result, LSTM networks are well suited for time series forecasting.

Algorithm III: Pseudo code for Analysis phase
1: Begin
2: Input: Historical request sequence
3: Output: The number of requests in the next interval
4: Prediction value \( \leftarrow \) Predict the number of requests in the next time interval using LSTM
5: Return Prediction value.
6: End

3.4. Planning phase
This paper applies RL as a decision-maker to obtaining the optimal action. RL problems can generally be modelled using Markov Decision Processes (MDPs). MDPs are a particular mathematical framework suited to modelling decision making under uncertainty. As shown in Table 1, the system state is considered as under-utilization if the CPU utilization is less than the lower threshold, and the system state is considered as over-utilization if the CPU utilization is greater than the upper threshold. Otherwise, the system state is considered as normal-utilization. This paper uses Q-learning algorithmic for the implementation of the decision-maker. The Q-learning algorithmic is defined by the Q function, which is expressed by Eq. (1):

\[
Q(S_t, a_t) = r(s, a) + \gamma \max Q(S_{t+1}, a_{t+1})
\]

where \( S_t \) is the state of system at the time \( t \), \( a_t \) is the agent’s action at the time \( t \). \( Q(s, a) \) is the value function that the agent at the time \( t \) performing action at the current states, which plays the role of the agent’s brain, \( r(s, a) \) is the reward after the agent applies \( a_t \), and the state transits from \( S_t \) to \( S_{t+1} \). \( \gamma \) is a discount factor \( 0 \leq \gamma \leq 1 \), a value close to 1 for \( \gamma \) assigns a greater weight to future rewards, whereas a value close to 0 considers only the most recent rewards. When selecting actions, the agent has a probability to randomly select one rather than choose the best one on the basis of current knowledge, So reward function \( R(s, a) \) is defined as a probability value as shown in Table 2.
Algorithm IV: Pseudo code for Planning phase

1: Begin
2: Input: Reqpredict
3: Output: Selected action()
4: Initialization: The Q-values table of pairs(s, a) by zero;
5: while (t<schedulingInterval) begin
6: Calculate the CPU utilization of the VMs in the time t
7: if (U(t) > Uupper-threshold) then Current state = Over-Utilization
8: else if (U(t) < Ulower-threshold) then Current state = Under-Utilization
9: else Current state = Normal-Utilization
10: Choose one of the actions based on Current state using Table III
11: Perform action a, and receive the feedback reward r(s, a) to reach to the new state S_t.
12: Update the Q-value table using Eq. 10
13: end while
14: if (Reqpredict > Capacity*Uupper-threshold) then Next-state=Over-Utilization
15: else if (Reqpredict < Capacity*Ulower-threshold) then Next-state=Under-Utilization
16: else Next-state=Normal-Utilization
17: Selected action(S) = Lookup(Next-state, Q-values table)
18: Return Selected action(S)
19: End

3.5. Execution phase
In this phase, the VM manager applies the optimal actions according to the result of the planning phase. The VM manager creates new VMs and dispatch the load balancer according to the best-fit strategy, or released latest initiated VM.

4. Performance Evaluation

4.1. Experimental Setup
To evaluate our approach, this paper uses the NASA traces [12] and the ClarkNet traces [13] represents user’s requests respectively. The NASA-HTTP contains two months’ worth of all HTTP requests to the NASA Kennedy Space Center WWW server in Florida. The ClarkNet-HTTP contains two week's worth of all HTTP requests to the ClarkNet WWW server. These workload traces represent realistic load variations over time that makes the results and conclusions more realistic and reliable to be used in real environments.
In the experiment, the proposed approach with other three strategies are compared. The first strategy is labeled “Worst”, which adds or releases VM when the system state is over-utilization or under-utilization. The second strategy is referred to as “Linear”, which is based on linear method to predict the workload and allocate resources ahead of time [14]. The third strategy is referred to as “LSTM”, which is focused on the Time Series forecasting of CPU usage of machines LSTM [6]. Finally, our approach is referred to as “LSTMQL”, which is combined LSTM to predict requests in the next time and RL to obtain the optimal action.

4.2. Performance Metrics
To evaluate the performance of the whole working system, following three performance metrics were used and compared with other algorithms, are described as: 1) SLA violation: SLA is used to monitor the quality of service, performance, response time from the service point of view. This paper is especially concerned about SLA violation and the delay time is used to approximate the SLA violation. 2) VMs allocated: This metric is defined as the number of VMs allocated to a cloud service Si at the ∆th interval. This is a secondary indicator. 3) The CPU utilization: The CPU utilization represents the utilization of resources.

4.3. Experimental results and Discussion
Table 3 shows the comparison of four approaches. Firstly, the “Worst” approach and the “Linear” approach get the worse results, resulting in bad user experience. The “Worst” approach rely solely on the current state, can not allocate VMs ahead of time. Although the “Linear” approach uses historical data to predict the requests, the accuracy is not as good as LSTM approach. Secondly, LSTM approach also has fewer scheduling times. Of course, LSTMQL approach needs more virtual machines, because it ensures enough machines to deal with workload spikes. The number of additional virtual no more than 100 machines compared with other approaches. Thirdly, the LSTM approach has better results than other approaches. However, when the requests suddenly appear peak, there will be a shake, and the CPU utilizations even reach 120%. Although LSTM can get more accurate prediction results, it does not use empirical data in planning phase. The Linear approach also has the same problem. While LSTMQL approach can get the stable results, this is because RL decides whether the system needs to schedule VMs, avoiding unnecessary resource scheduling.

In overall, the proposed approach enables the virtual machines to work steadily, decreases SLA violations by up to 20%-30% compared with “Worst” approach. The reason behind this performance can be summed up: First, LSTM network is more accurate in predicting future requests and the system can allocate appropriate resources ahead of time. Second, RL decides whether the system needs to schedule VMs, avoiding unnecessary resource scheduling, especially when requests suddenly appear peak, RL make the right decision based on historical experience and current system state.

|          | NASA/Worst | NASA/Linear | NASA/LSTM | NASA/Proposed | ClarkNet/Worst | ClarkNet/Linear | ClarkNet/LSTM | ClarkNet/Proposed |
|----------|------------|-------------|-----------|---------------|----------------|-----------------|---------------|-------------------|
| SLA violation(%) | 0.85% | 0.79% | 0.50% | 0.11% | 2.02% | 2.02% | 1.04% | 0.93% |
| Allocated Vms | 2306 | 2310 | 2345 | 2370 | 2812 | 2824 | 2818 | 3002 |
| Scheduling times | 38 | 36 | 32 | 30 | 16 | 14 | 14 | 11 |
| Cpu utilization (%) | 50.2% | 49.3% | 47.7% | 49.9% | 50.0% | 49.7% | 46.5% | 50.3% |

5. Conclusion and Future Work
This paper proposes an efficient autonomic resource provisioning approach based on the control MAPE loop. The approach applies LSTM to predicting the requests in the analysis phase and uses RL to obtain the optimal action to scale in or scale out virtual machines in the planning phase. This paper evaluates the performance of the proposed approach under real world workload traces and compare the results with other approaches. Experimental results demonstrated that the approach ensures virtual machines to
work steadily in unstable environment and reduces SLA violations by up to 20%-30% compared with other approaches. The work only considers homogenous tasks schedule. In future work, effective resource schedule approaches for heterogeneous tasks and fork-join tasks in cloud system will be explore.

Acknowledgments
This work was financially supported by the National Key Research and Development Program of China (Grant: No.2017YFB 1402400), Social Undertakings and Livelihood Security Science and Technology Innovation Funds of CQ CSTC (Grant: csc2017shmsA0641), the National Nature Science Foundation of China (Grant: 61762025), the Fundamental Research Funds for the Central Universities (Grant: NO.2018CDYYSY0055), the Graduate Research and Innovation Foundation of Chongqing, China (Grant: No.CYB18058).

References
[1] Jamshidi P , Ahmad A , Pahl C . Cloud Migration Research: A Systematic Review[J]. IEEE Transactions on Cloud Computing, 2013, 1(2):142-157.
[2] Mao Y , Oak J , Pompili A , et al. DRAPS: Dynamic and Resource-Aware Placement Scheme for Docker Containers in a Heterogeneous Cluster[J]. 2018.
[3] Kumar S , Pandey M K , Nath A , et al. Missing QoS-values predictions using neural networks for cloud computing environments[C]// 2015 International Conference on Computing and Network Communications (CoCoNet). IEEE, 2015.
[4] Alzahrani E J , Tari Z , Zeephongsekul P , et al. SLA-Aware Resource Scaling for Energy Efficiency[C]// 2016 IEEE 18th International Conference on High-Performance Computing and Communications, IEEE 14th International Conference on Smart City, and IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS). IEEE Computer Society, 2016.
[5] Aydin O , Guldamasioglu S . [IEEE 2017 4th International Conference on Electrical and Electronic Engineering (ICEEE) - Ankara, Turkey (2017.4.8-2017.4.10)] 2017 4th International Conference on Electrical and Electronic Engineering (ICEEE) - Using LSTM networks to predict engine condition on large scale data processing framework[C]// International Conference on Electrical & Electronic Engineering. IEEE, 2017:281-285.
[6] Janardhanan D , Barrett E . CPU workload forecasting of machines in data centers using LSTM recurrent neural networks and ARIMA models[C]// Internet Technology & Secured Transactions. IEEE, 2018.
[7] Tania Lorido-Botrán, Miguel-Alonso J , Lozano J A . A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments[J]. Journal of Grid Computing, 2014, 12(4):559-592.
[8] Computing, A. 2006. An architectural blueprint for autonomic computing. IBM White Paper, 31, 1-6.
[9] Huang J , Li C , Yu J . Resource prediction based on double exponential smoothing in cloud computing[C]// International Conference on Consumer Electronics. IEEE, 2012.
[10] An autonomic approach to manage elasticity of business processes in the Cloud[J]. Future Generation Computer Systems, 2015, 50:49-61.
[11] Messias, Valter Rogério, Estrella J C , Ehlers R , et al. Combining time series prediction models using genetic algorithm to autoscaling Web applications hosted in the cloud infrastructure[J]. Neural Computing and Applications, 2016, 27(8):2383-2406.
[12] Nasa-http-two months of http logs from the ksc.nasa www server. http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html (Accessed 19.01.17).
[13] Clarknet-http-two weeks of http logs from the clarknet www server. http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html (Accessed 19.01.17).
[14] Yang J , Liu C , Shang Y , et al. A cost-aware auto-scaling approach using the workload prediction in service clouds[J]. Information Systems Frontiers, 2014, 16(1):7-18.