Spot Virtual Machine Eviction Prediction in Microsoft Cloud

Fangkai Yang*, Bowen Pang*, Jue Zhang*, Bo Qiao*, Lu Wang*, Camille Couturier§, Chetan Bansal*, Soumya Ram†, Si Qin*, Zhen Ma§, Íñigo Goiri*, Eli Cortez†, Senthil Baladhandayutham§, Victor Rühle§, Saravan Rajmohan§, Qingwei Lin*, Dongmei Zhang*

Microsoft Research*, Microsoft 365§, Microsoft Azure†

ABSTRACT

Azure Spot Virtual Machines (Spot VMs) utilize unused compute capacity at significant cost savings. They can be evicted when Azure needs the capacity back, therefore suitable for workloads that can tolerate interruptions. A good prediction of Spot VM evictions is beneficial for Azure to optimize capacity utilization and offers users information to better plan Spot VM deployments by selecting clusters to reduce potential evictions. The current in-service cluster-level prediction method ignores the node heterogeneity by aggregating node information. In this paper, we propose a spatial-temporal node-level Spot VM eviction prediction model to capture the inter-node relations and time dependency. The experiments with Azure data show that our node-level eviction prediction model performs better than the node-level and cluster-level baselines.

CCS CONCEPTS

• **Computer systems organization → Cloud computing**;

KEYWORDS

Spot virtual machine, eviction prediction, unused capacity

ACM Reference Format:

Fangkai Yang*, Bowen Pang*, Jue Zhang*, Bo Qiao*, Lu Wang*, Camille Couturier§, Chetan Bansal*,, Soumya Ram†, Si Qin*, Zhen Ma§, Íñigo Goiri*, Eli Cortez†, Senthil Baladhandayutham§,, and Victor Rühle§, Saravan Rajmohan§, Qingwei Lin*, Dongmei Zhang* . 2022. Spot Virtual Machine Eviction Prediction in Microsoft Cloud. In *Companion Proceedings of the Web Conference 2022 (WWW '22 Companion), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3487553.3524229

1 INTRODUCTION

Mainstream cloud computing providers have designed and adopted Spot Virtual Machines (Spot VMs) or similar instances to utilize unused capacity in the cloud system [1, 2, 8, 17]. There is an amount of unused capacity available in Microsoft Azure due to service healing, recovery, and anticipated capacity growth [7]. Azure Spot Virtual Machines (Spot VMs) are designed to utilize the unused capacity at significant discounts of up to 90% compared with pay-as-you-go prices in exchange for very low SLOs (Service Level Objectives) [6]. When Azure needs the capacity back, *e.g.*, deploying on-demand VMs, the Azure infrastructure will evict Spot VMs with 30 seconds notice [8]. It is because on-demand VMs have higher priority than Spot VMs. Therefore, Spot VMs are suitable for workloads that can sustain or recover from interruptions and stateless applications or non-critical workloads that can run in the background at a flexible schedule, such as testing, advanced analytics, and machine learning [7, 8].

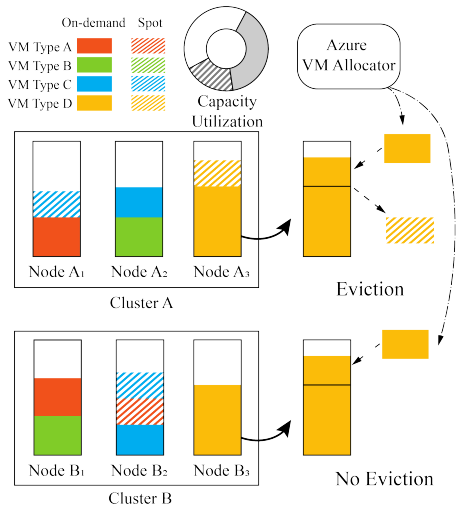

Figure 1: A motivation example for node-level prediction. Cluster *A* and *B* have the same cluster capacity utilization of on-demand VMs (solid-shaded) and Spot VMs (line-shaded) at the cluster level, but they have different VM distributions at the node level. In this example, there are four types of VMs *A*, *B*, *C*, and *D*. One rule in the VM allocator indicates that type *D* (yellow) on-demand VMs can only be allocated to nodes without other types of VMs. When allocating a type *D* on-demand VM to both Cluster *A* and *B*, this rule results in a Spot VM eviction on node A_3.

A good estimation and prediction of Spot VM evictions, *i.e.*, the probability of Spot VMs getting evicted in the future, is crucial for the Azure cloud system and the users to make informed and optimized decisions. On the Azure side, Spot VM eviction predictions, available per region in the Azure Portal when deploying new VMs, are helpful to optimize capacity utilization planning and allocation management. On the user side, the eviction prediction informs users to optimize deployment plans to increase the survivability of Spot VMs and reduce the possibility of interruptions.

However, the rapid growth of Microsoft Azure in terms of the large-scale capacity and resource utilization makes it challenging to predict Spot VM evictions. Azure cloud system contains a large

WWW '22 Companion, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9130-6/22/04…$15.00
https://doi.org/10.1145/3487553.3524229

number of physical servers, *i.e.*, *Nodes*, that are allocated in different regions. Each region contains many *Clusters* and each cluster contains hundreds of nodes [14, 18]. In this paper, we investigate the challenges with the current cluster-level eviction prediction method and propose a new eviction prediction methodology in a finer granularity, *i.e.*, at the node level, to effectively capture the spatial and temporal information of nodes. As we will see, our method improves the eviction prediction accuracy.

2 CHALLENGES AND MOTIVATION

It is natural to use simulations for predicting future events in domains such as weather forecasting, and software prediction [20, 31], yet it is challenging to predict Spot VM evictions via simulating future allocations of VMs and capacity needs. Indeed, one of the most critical components of the Azure system is the VM allocator [18], which assigns VMs to nodes. The VM allocator is associated with numerous allocation rules and management constraints. It could allocate on-demand VMs to highly packed nodes with Spot VMs, which trigger Spot VM evictions. On the other hand, the large-scale nature and the complexity of Azure make it even harder to simulate evictions in the future. Thus, we adopt learning-based methods which do not require an allocation simulator by efficiently learning temporal patterns from historical deployment and eviction data.

The current eviction prediction method in Azure is at the cluster level. It can capture the cluster-level features, such as capacity utilization, VM deployments, and Spot VM evictions, through the historical data. As aforementioned, Azure clusters contain a large number of nodes that have heterogeneous characteristics in capacity and deployed VMs, and the Azure allocator assigns on-demand and Spot VMs to different nodes within the cluster. However, the cluster-level prediction method treats the nodes homogeneously by learning from the cluster-level features. It ignores the differences among nodes which could result in prediction errors. Taking Figure 1 as an example, Spot VM evictions can still happen in clusters with enough capacity when deploying on-demand VMs. The allocator has numerous heuristic allocation rules, and constraints [15, 26, 29]; one of the rules prevent collocating specific VM types (here type D, in yellow) with other types. It thus deploys on-demand VMs to nodes with Spot VMs, triggering their evictions even though there is enough capacity on other nodes.

This paper is motivated by resolving the challenges when predicting Spot VM evictions at the cluster level. We propose the node-level eviction prediction methods to increase the accuracy by capturing the node heterogeneity. Then, the predicted evictions on this cluster are aggregated from the nodes with Spot VMs deployed. Moreover, a spatial-temporal framework has been used to capture the node relations and historical patterns better. As a result, the eviction prediction can be improved.

3 METHODOLOGY

3.1 Preliminaries and Definition

In practice, Azure has large-scale clusters C and each cluster C_i consists of many nodes $\mathcal{N}$ that can be roughly classified as nodes with Spot VMs $\mathcal{N}_S = \{\mathcal{N}_{S_1}, \mathcal{N}_{S_2}, \cdots, \mathcal{N}_{S_P}\}$ and those without $\mathcal{N}_N = \{\mathcal{N}_{N_1}, \mathcal{N}_{N_2}, \cdots, \mathcal{N}_{N_K}\}$ in this paper. We aim to predict the Spot VM evictions $y_\tau^{C_i}$ for cluster C_i at the future timestep τ.

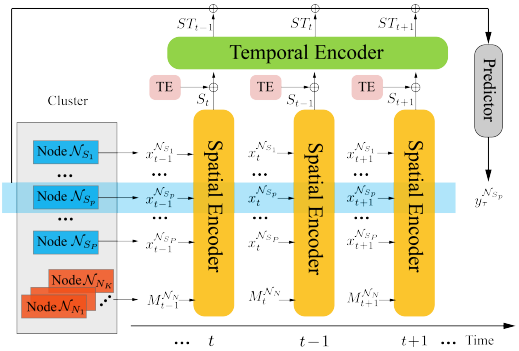

Figure 2: The overview of our node-level spatial-temporal prediction framework.

3.2 Cluster-level Prediction

The cluster-level prediction method is an end-to-end solution with cluster-level features. These features are collected and simplified into time series features $\mathcal{X}^{C_i} = \{x_1^{C_i}, x_2^{C_i}, \cdots, x_T^{C_i}\}$, *i.e.*, the time-series features of cluster C_i in the previous T hours. At each timestep t, $x_t^{C_i} = \{x_t^1, x_t^2, \cdots, x_t^K\}$ where K is the feature dimension. These features include the information of cluster capacity, core and memory utilization, and evicted Spot VMs (see details in Section 4.1). Then time series prediction methods (refer to Section 4.2) are used to predict Spot VM eviction rate $y_\tau^{C_i}$ in the future time τ by taking those time series features. The objective is to minimize the mean squared error (MSE) loss $\mathcal{L}_\tau^C$ between the predicted eviction rate $y_\tau^{C_i}$ and the target eviction rate y_τ^{tgt,C_i}:

$$\mathcal{L}_\tau^C = \frac{1}{|C|} \sum_{i=1}^{|\tilde{\mathbf{O}}|} (y_\tau^{C_i} - y_\tau^{tgt,C_i})^2. \tag{1}$$

3.3 Node-level Prediction

In contrast to the cluster-level prediction, we adopt spatial-temporal transformer networks to capture both the inter-nodes and temporal relations in our node-level prediction method. The spatial-temporal transformer framework is proven to efficiently capture features in the domains of context generation [25], behavior generation and recognition [13, 28, 36] and traffic flow prediction [35]. However, the structure of this framework varies among these works depending on related scenarios, and we modified it to fit our eviction prediction scenario (see Figure 2). The framework maintains the original Transformer [33] architecture. The difference is that each Transformer encoder is dedicated to a specific task.

The *Spatial Encoder* focuses on the inter-node relations at a single timestep t with the input $X_t^{\mathcal{N}} = X_t^{\mathcal{N}_S} \cup X_t^{\mathcal{N}_N}$, presenting the information of P nodes with Spot VMs $X_t^{\mathcal{N}_S} = \{x_t^{\mathcal{N}_{S_1}}, x_t^{\mathcal{N}_{S_2}}, \cdots, x_t^{\mathcal{N}_{S_P}}\}$ and K nodes without Spot VMs $X_t^{\mathcal{N}_N} = \{x_t^{\mathcal{N}_{N_1}}, x_t^{\mathcal{N}_{N_2}}, \cdots, x_t^{\mathcal{N}_{N_K}}\}$. Considering the fact that $|\mathcal{N}|$ varies among clusters and $\mathcal{N}_N$ contains redundant information, we merge the representations $X_t^{\mathcal{N}_N}$ through a merger (we use aggregation in our paper) as $M_t^{\mathcal{N}_N}$ to reduce the amount of calculation and memory consumption, and also lower the disturbance from redundant information. Then the input is $\mathbf{X}_t = \{x_t^{\mathcal{N}_{S_1}}, x_t^{\mathcal{N}_{S_2}}, \cdots, x_t^{\mathcal{N}_{S_P}}, M_t^{\mathcal{N}_N}\}$. The spatial encoder

keeps the architecture of Transformer encoder [33] which contains a multi-head self-attention layer and a feed-forward layer, and each of them has a residual connection. The self-attention layer allows for each node to attend to other nodes by calculating the attention distribution, *i.e.*, $Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{QK^T}}{\sqrt{d}}\mathbf{V})$, where $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ represent query, key and value vectors that are linearly transformed from $\mathbf{X}_t$. The output is then fed into a feed-forward layer to have the inter-node representations S_t, which acts as inputs to the temporal encoder as follows.

The *Temporal Encoder* captures the temporal dependencies between timesteps from historical features. A large number of deployments and the large-scale size of Azure clusters make the related information easily covered by irrelevant and outdated representations. In this paper, we use a sliding window with a horizon of T to avoid disturbances. It takes the sequence of processed inter-node representations $[S_1, S_2, \cdots, S_T]$ from the spatial encoder as inputs. In order to indicate the temporal position of the spatial representations, we introduce the Temporal Encoding (TE) here, similar to the positional encoding [33]. The structure of the temporal encoder is similar to the spatial encoder with a multi-head self-attention layer and a feed-forward layer, but the self-attention attends to the time dependency. The output $ST = \{ST_1, ST_2, \cdots, ST_T\}$ acts as the global spatial-temporal information.

The global information is then added to the node time series feature resulting in $\mathbf{X}^{\mathcal{N}_{S_p}} = \{x_1^{\mathcal{N}_{S_p}} + ST_1, x_2^{\mathcal{N}_{S_p}} + ST_2, \cdots, x_T^{\mathcal{N}_{S_p}} + ST_T\}$, where the node with Spot VMs, *i.e.*, $\mathcal{N}_{S_p}$, is the target node we aim to predict Spot VM eviction rate $y_\tau^{\mathcal{N}_{S_p}}$ in the future timestep τ. After obtaining predictions for all the nodes with Spot VMs $\mathcal{N}_S$, we aggregate the results at the cluster-level to get cluster-level evictions for comparisons. The aggregated cluster-level eviction rate y_τ^{agg,C_i} is defined as:

$$y_\tau^{agg,C_i} = \frac{\sum_{p=1}^{P} y_\tau^{\mathcal{N}_{S_p}} Z_{S_p}}{\sum_{p=1}^{P} Z_{S_p}}, \tag{2}$$

where Z_{S_p} is the number of Spot VMs in node $\mathcal{N}_{S_p}$ at the last timestep T in the sliding window. Hence, our objective loss is similar to the cluster-level loss $\mathcal{L}_\tau^C$ by minimizing the MSE loss between the aggregated and target eviction rate at the cluster level.

4 EXPERIMENTAL EVALUATION

4.1 Experimental Setup

The data we used for the experiments are collected from the Azure cloud system. The training dataset consists of recent two weeks' data from 20 clusters containing around 12000 nodes, and the testing dataset collects the data in the following one week from the same clusters. The data are collected per hour. At each timestep, the node features include the number of on-demand and Spot VMs, node capacity (core and memory), node capacity utilization (core and memory utilization), evicted core/memory/number of Spot VMs from the previous 3 hours till the current timestep, and evicted core/memory/number of Spot VMs from current timestep to the next 3 hours. Note that the eviction in this paper is defined as the number of Spot VMs at the current timestep that gets evicted in the future, and we do not consider the evictions of future deployed Spot VMs. The cluster features are aggregated by the node features within this cluster. In order to reduce computation and memory consumption, we use a sliding window (Section 3.3) with 48 hours to make one data sample. Note that for the last three timesteps in every training sample, we remove the features of evicted core/memory/number of Spot VMs to the next hours, preventing the future formation leak in training, *e.g.*, X_T does not have eviction information from timestep $T+1, T+2, T+3$. Similarly, X_{T-1} can not see the eviction features at timestep $T+2, T+3$, and X_{T-2} can not see $T+3$. We experiment our model and baselines (see Section 4.2) with the above collected data. All experiments are performed on a machine with Intel(R) Xeon(R) CPU E5-2690 @ 2.6GHz and 112GB memory. The implementation is available[1].

4.2 Baselines

Four commonly used time series prediction approaches in industries, *i.e.*, Linear Regression (LR) [30], Random Forest (RF) [10], Gradient Boosting Decision Tree (GBDT) [16], Support Vector Regression (SVR) [5], and Long Short-term Memory (LSTM) [19] are adopted to have cluster-level and node-level eviction predictions. For the node-level prediction methods, after all the nodes with Spot VMs at the current timestep have predicted the eviction rate in the next three hours, they are aggregated at the cluster level to get the eviction rate in this cluster. This node-level aggregated eviction rate is then compared with the cluster-level predictions. We evaluate these methods with RMSE on the next one, two and three hours.

4.3 Experimental Results

In this paper, we aim to address three research questions:

- RQ1: Does node-level prediction contribute to improving eviction rate prediction on cluster-level?

As shown in Table 1, our node-level spatial-temporal model performs better than other node-level and cluster-level baselines. It takes the spatial-temporal information from other nodes to improve prediction. Among the baselines, node-level prediction methods are generally not better than the cluster-level ones with the same algorithm. In particular, regression methods (RF, GBDT, SVR) have better performance at the cluster level. It might be that training on nodes alone from all clusters causes unnecessary noise but fails to capture a general temporal node pattern. LSTM-based methods have a poor performance than the regression methods (RF, GBDT, SVR), and there are a few works discussing the performance issue of LSTM learning on sequential data [12, 33]. It is notable that GBDT-Cluster shows a close performance with our model.

- RQ2: How does prediction accuracy change with the prediction horizon?

Table 1 shows a trend of increased RMSE with increasing predicted time horizon, *i.e.*, a longer-time prediction leads to a higher RMSE. However, we do see that for some of the baseline methods, the error doesn't increase monotonically (*e.g.*, larger error for the 2-hour prediction). This could be due to noise in the data leading to statistical fluctuations, or redundancy in the nodes, that these baseline methods fail to tackle with the amount of data provided.

[1] aka.ms/EvictionPrediction

Table 1: Spot VM eviction prediction performance (RMSE, lower is better)

Method	1-hour (%)	2-hour (%)	3-hour (%)
LR-Cluster [30]	48.45	61.24	57.00
LR-Node [30]	43.30	53.49	46.20
RF-Cluster [10]	17.79	19.82	22.17
RF-Node [10]	23.56	13.68	23.51
GBDT-Cluster [16]	15.61	18.25	19.13
GBDT-Node [16]	23.51	13.68	22.77
SVR-Cluster [5]	23.31	24.53	24.85
SVR-Node [5]	23.30	**12.74**	24.40
LSTM-Cluster [19]	31.31	32.01	30.72
LSTM-Node [19]	26.19	49.09	37.99
Our model	**13.77**	16.82	**18.50**

Our method shows a monotonic RMSE increase with an increasing prediction horizon when considering all clusters. This does not hold for all cluster buckets (see RQ3). The two weeks of training data and the length of sliding window were constrained by the current implementation of the self-attention architecture, which has quadratic time/space complexity. As follow-up work, we plan to modify the self-attention layer as in Reformer [24] and Longformer [9] to allow training on longer sequences and periods while reducing the training and inference time.

- RQ3: Does the capacity utilization of the clusters have any relation to the prediction accuracy?

We bucketize all the clusters from our dataset into four buckets based on the core utilization, *i.e.*, bucket A with core utilization range from 80% to 100%, bucket B with 60% to 80%, etc. As shown in Table 2, clusters with high core utilization generally have a lower Spot/On-demand ratio, which indicates Spot VMs are much fewer in the highly packed nodes. On the other hand, cluster bucket D has more Spot VMs than on-demand VMs, indicating that clusters with lower core utilization are preferable and safer for Spot VMs. Our node-level eviction prediction model has higher accuracy in cluster buckets with higher and lower core utilization, but it performs poorly in clusters with around half core utilization. It might be because the evictions in the clusters with high and low core utilization are more predictable, *e.g.*, more evictions in highly core-utilized clusters and fewer evictions in low core-utilized ones. The bottleneck of the prediction method lies in those bucket C clusters that are more dynamic and unpredictable. One of our further works is to include future deployment information to improve our prediction model since we can obtain the deployment notification ahead of the deployment actions.

5 APPLICATION IN PRACTICE

The prediction of the eviction rate of Spot VMs has been applied to several scenarios in Azure. During the Spot VM allocation, we leverage the predicted eviction rates to choose suitable clusters to match with the service's SLO. When evictions occur, Azure has the mechanism of retrying spot VMs across different clusters in the same region, and the newly-chosen clusters are often required to have less predicted eviction rates than the existing ones. Therefore, choosing which new clusters for spot VM restoration is also guided by the eviction rate prediction results. With the improved prediction method described in this work, the decisions on both allocation and restoration are further optimized, reducing unnecessary service interruptions arising from imprecise eviction rate prediction.

Table 2: Spot VM eviction prediction performance with our model in bucketized clusters.

Cluster buckets	A (100-80)	B (80-60)	C (60-40)	D (40-20)
Core util (%)	86.10	73.09	54.27	26.70
Spot/On-demand (%)	8.71	27.29	23.00	195.49
1-hour RMSE (%)	10.11	10.58	28.11	14.80
2-hour RMSE (%)	11.08	11.76	26.42	13.73
3-hour RMSE (%)	12.36	12.31	24.04	20.58

6 RELATED WORK

Spot VMs or similar preemptible instances are used in cloud computing providers, including Microsoft Azure [8], Amazon [2], Google [17], and Alibaba [1]. In all of these providers, the spot VMs or instances are evictable based on the supply and demand relations. In addition, Harvest VMs [4] offer a more flexible way to utilize unused capacity. Although cloud computing providers conduct reports on analyzing the eviction scenarios [3, 8], very few works can be found that research and explore on the Spot VM eviction prediction and its applications. There are methods in the literature working on time series prediction on the behaviors of a cloud computing system using multiple signals such as workload and performance [23, 27, 32, 34]. Researchers also integrated the time series prediction model with the information summarized from historical data [4, 14, 21, 22]. Research on the hierarchical spatial-temporal structure for the time-series prediction problem can also be found in recent literature to improve prediction accuracy [11, 13, 35, 37]. However, most of these works are strongly scenario-related and not directly applicable to the Spot VM eviction prediction problem.

7 CONCLUSION

In this paper, we investigate Spot VM eviction prediction methods at the node level and the cluster level. We present our node-level spatial-temporal prediction method and its application in Microsoft Azure. Our experiments on industrial datasets collected by the Azure cloud system show that our model outperforms the node-level and cluster-level baselines with higher prediction accuracy. By answering three proposed research questions, we found that: 1) cluster-level prediction baselines generally perform better than their node-level implementations, 2) a longer-time prediction has a lower prediction accuracy in cluster-level baselines and our model, which is not observed at the node-level, 3) clusters with medium core utilization is the bottleneck case of our prediction model. We also demonstrate the application of the proposed eviction prediction in Spot VM retry and restoration in Azure.

8 ACKNOWLEDGEMENT

We thank Chuan Luo, Mahmoud Sayed, Yandan Wang, Gurpreet Virdi for their extensive knowledge and insightful suggestions.

REFERENCES

[1] Alibaba. [n. d.]. Alibaba Preemptible Instances. https://www.alibabacloud.com/help/doc-detail/52088.htm. Accessed: 2021-10-25.

[2] Amazon. [n. d.]. Amazon EC2 Spot Instances. https://www.amazonaws.cn/en/ec2/spot-instances/. Accessed: 2021-10-25.

[3] Amazon. [n. d.]. Spot Instance Interruptions. https://docs.amazonaws.cn/en_us/AWSEC2/latest/UserGuide/spot-interruptions.html. Accessed: 2021-10-25.

[4] Pradeep Ambati, Íñigo Goiri, Felipe Frujeri, Alper Gun, Ke Wang, Brian Dolan, Brian Corell, Sekhar Pasupuleti, Thomas Moscibroda, Sameh Elnikety, et al. 2020. Providing slos for resource-harvesting vms in cloud platforms. In *14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20)*. 735–751.

[5] Mariette Awad and Rahul Khanna. 2015. Support vector regression. In *Efficient learning machines*. Springer, 67–80.

[6] Microsoft Azure. [n. d.]. Announcing the preview of Azure Spot Virtual Machines. https://azure.microsoft.com/en-us/blog/announcing-the-preview-of-azure-spot-virtual-machines/. Accessed: 2021-10-25.

[7] Microsoft Azure. [n. d.]. Azure Spot Virtual Machines. https://azure.microsoft.com/en-us/services/virtual-machines/spot/#overview. Accessed: 2021-10-25.

[8] Microsoft Azure. [n. d.]. Use Azure Spot Virtual Machines. https://docs.microsoft.com/en-us/azure/virtual-machines/spot-vms. Accessed: 2021-10-25.

[9] Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150* (2020).

[10] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[11] Jian Cao, Jiwen Fu, Minglu Li, and Jinjun Chen. 2014. CPU load prediction for cloud environment based on a dynamic ensemble model. *Software: Practice and Experience* 44, 7 (2014), 793–804.

[12] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[13] Yuren Cong, Wentong Liao, Hanno Ackermann, Bodo Rosenhahn, and Michael Ying Yang. 2021. Spatial-Temporal Transformer for Dynamic Scene Graph Generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 16372–16382.

[14] Hang Dong, Si Qin, Yong Xu, Bo Qiao, Shandan Zhou, Xian Yang, Chuan Luo, Pu Zhao, Qingwei Lin, Hongyu Zhang, et al. 2021. Effective low capacity status prediction for cloud systems. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1236–1241.

[15] Ramkumar Eswaraprasad and Linesh Raja. 2017. A review of virtual machine (VM) resource scheduling algorithms in cloud computing environment. *Journal of Statistics and Management Systems* 20, 4 (2017), 703–711.

[16] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[17] Google. [n. d.]. Google Spot VMs. https://cloud.google.com/compute/docs/instances/spot. Accessed: 2021-10-25.

[18] Ori Hadary, Luke Marshall, Ishai Menache, Abhisek Pan, Esaias E Greeff, David Dion, Star Dorminey, Shailesh Joshi, Yang Chen, Mark Russinovich, et al. 2020. Protean:{VM} Allocation Service at Scale. In *14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20)*. 845–861.

[19] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[20] Peter L Houtekamer, Louis Lefaivre, Jacques Derome, Harold Ritchie, and Herschel L Mitchell. 1996. A system simulation approach to ensemble prediction. *Monthly Weather Review* 124, 6 (1996), 1225–1242.

[21] Yexi Jiang, Chang-Shing Perng, Tao Li, and Rong Chang. 2012. Intelligent cloud capacity management. In *2012 IEEE Network Operations and Management Symposium*. IEEE, 502–505.

[22] Yexi Jiang, Chang-shing Perng, Tao Li, and Rong Chang. 2012. Self-adaptive cloud capacity planning. In *2012 IEEE Ninth International Conference on Services Computing*. IEEE, 73–80.

[23] Yexi Jiang, Chang-Shing Perng, Tao Li, and Rong N Chang. 2013. Cloud analytics for capacity planning and instant VM provisioning. *IEEE Transactions on Network and Service Management* 10, 3 (2013), 312–325.

[24] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rkgNKkHtvB

[25] Yang Liu and Mirella Lapata. 2019. Hierarchical transformers for multi-document summarization. *arXiv preprint arXiv:1905.13164* (2019).

[26] Zoltán Ádám Mann. 2015. Allocation of virtual machines in cloud data centers—a survey of problem models and optimization algorithms. *Acm Computing Surveys (CSUR)* 48, 1 (2015), 1–34.

[27] Sunilkumar S Manvi and Gopal Krishna Shyam. 2014. Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey. *Journal of network and computer applications* 41 (2014), 424–440.

[28] Chiara Plizzari, Marco Cannici, and Matteo Matteucci. 2020. Spatial temporal transformer network for skeleton-based action recognition. *arXiv preprint arXiv:2012.06399* (2020).

[29] Bo Qiao, Fangkai Yang, Chuan Luo, Yanan Wang, Johnny Li, Qingwei Lin, Hongyu Zhang, Mohit Datta, Andrew Zhou, Thomas Moscibroda, et al. 2021. Intelligent container reallocation at Microsoft 365. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1438–1443.

[30] George AF Seber and Alan J Lee. 2012. *Linear regression analysis*. Vol. 329. John Wiley & Sons.

[31] Martin Shepperd and Gada Kadoda. 2001. Comparing software prediction techniques using simulation. *IEEE Transactions on Software Engineering* 27, 11 (2001), 1014–1022.

[32] Weijia Song, Zhen Xiao, Qi Chen, and Haipeng Luo. 2013. Adaptive resource provisioning for the cloud using online bin packing. *IEEE Trans. Comput.* 63, 11 (2013), 2647–2660.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[34] Bin Xia, Tao Li, Qi-Feng Zhou, Qianmu Li, and Hong Zhang. 2018. An effective classification-based framework for predicting cloud capacity demand in cloud services. *IEEE Transactions on Services Computing* (2018).

[35] Mingxing Xu, Wenrui Dai, Chunmiao Liu, Xing Gao, Weiyao Lin, Guo-Jun Qi, and Hongkai Xiong. 2020. Spatial-temporal transformer networks for traffic flow forecasting. *arXiv preprint arXiv:2001.02908* (2020).

[36] Fangkai Yang, Wenjie Yin, Tetsunari Inamura, Mårten Björkman, and Christopher Peters. 2020. Group behavior recognition using attention-and graph-based neural networks. In *ECAI 2020*. IOS Press, 1626–1633.

[37] Linyi Yang, Tin Lok James Ng, Barry Smyth, and Riuhai Dong. 2020. Html: Hierarchical transformer-based multi-task learning for volatility prediction. In *Proceedings of The Web Conference 2020*. 441–451.