[APPENDIX] PROTORAIL: A RISK-COGNIZANT IMITATION AGENT FOR ADAPTIVE VCPU OVERSUBSCRIPTION IN THE CLOUD

Lu Wang [*1] **Mayukh Das** [*1] **Fangkai Yang** [1] **Bo Qiao** [1] **Hang Dong** [1] **Si Qin** [1] **Victor Ruehle** [1] **Chetan Bansal** [1] **Eli Cortez** [1] **Inigo Goiri** [1] **Saravan Rajmohan** [1] **Qingwei Lin** [1] **Dongmei Zhang** [1]

(Code at `https://tinyurl.com/protorail`)

A ADDITIONAL STUDIES

A.1 Hyper-parameter Tuning:

The number of prototypes is the key hyper-parameters for our model. We tune the number of prototypes in $\{2, 3, 4, 5, 6\}$ and set 3 for vCPUs and 4 for airplane tickets respectively with the optimal search value. The detailed results are shown in Table 1.

Table 1. Hyper-parameter tuning.

# options	*vCPU Oversub.*		*Flight Tickets*	
	Hot Node	Core	Cost	Profit
2	0.26%	8067	0.32M	12.59M
3	0%	8161	0.27M	13.10M
4	0%	8092	0.14M	13.65M
5	0.05%	8154	0.22M	13.15M
6	0.07%	8150	0.16M	12.64M

There are 2 primary hyperparameters in the KITL module; the uncertainty thresholds $\mathbb{U}_p, \mathbb{U}_a$. While they do not directly affect the loss surface, they control query generation. Extremely conservative (high) uncertainty thresholds lead to extremely focused queries and reduce the number of queries. However, it often leads to missing out on prototypes that really do need feedback, since the final query is an intersection with top-N prototype embeddings. On the other hand, the relaxed (lower) thresholds lead to too many queries. Another minor hyperparameter is 'N', the cardinality of top-N choices of in the case of p_d. All results are reported with $N = 5$. However, we did perform a grid search over all 3 hyperparameters and observed that for thresholds a range of $(0.5, 0.75)$ was mostly appropriate, with 0.55 being the value used for reported results. For N higher values make no sense since p_d intersects with p_μ anyway. Values of N lower than 3 resulted in $p_\mu \cap p_d = \emptyset$.

A.2 Pressure Test:

Our experiments are based on real-world datasets. It is impractical to conduct pressure tests from two aspects: a) we cannot do endless vCPU oversubscription if there is no stranded memory available to place additional VMs. b) the real-world CPU utilization distribution would rarely make many hot nodes as their CPU utilization will not peak simultaneously. Nevertheless, we also experimented on the high-pressure test by manually reducing the hot ratio (which is much smaller than the real-world value), the results are shown as follows, where PROTORAIL also achieves the best performance on both metrics compared with the best two baselines.

Table 2. Pressure Test

Method	Hot Node	Remain Core
Behavior Cloning	92.61%	28
Dagger	90.74%	23
ProtoHAIL	86.25%	34

B APPLICATION IN PRACTICE

The vCPU oversubscription policy has been successfully implemented in several scenarios within cloud services. This policy determines the appropriate oversubscription rate for each vCPU request made by users. By effectively managing oversubscription, we can reduce resource wastage and ensure that user requirements are met. Our proposed adaptive oversubscription policy improves vCPU utilization by 9.4% and maintains a 0% hot node rate in the cloud system. We also provide a case study to demonstrate the effectiveness of our approach in real-world environments.

B.1 More details on Case Study

In this section, we compare the performance of our proposed method with the strongest baseline, behavior cloning. The data was collected from one of Cloud services over a two-week period. This service has been deployed in approximately 300 clusters.

[1]Microsoft. Correspondence to: Mayukh Das <mayukhdas@microsoft.com>, Lu Wang <wlu@microsoft.com>.

Proceedings of the 8^{th} MLSys Conference, Santa Clara, CA, USA, 2025.

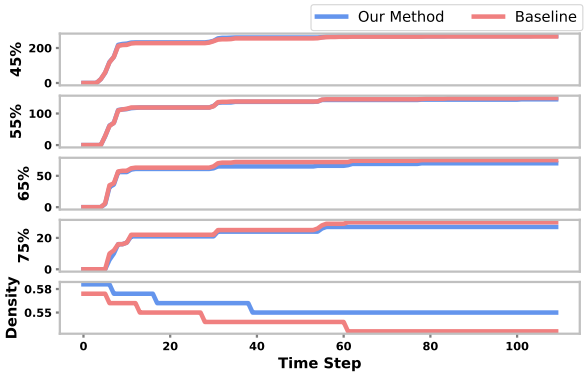

Figure 1. A/B test Results

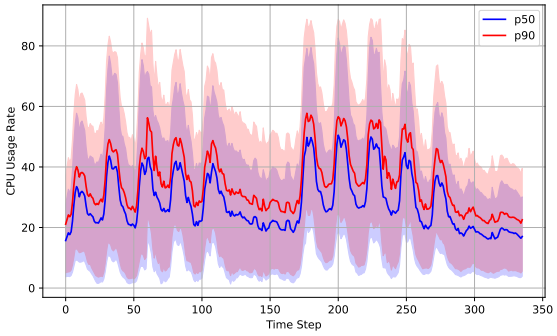

Figure 2. Cpu Usage Rate

Figure 2 shows the CPU usage rate for this service, with each time step representing one hour. The p50 and p90 values indicate the 50th and 90th quantiles of the CPU usage rates among all VMs. The shaded area represents the standard deviation, and the solid line indicates the mean value. We can see that the mean of p90 is around 60%, indicating that there is significant potential capacity that can be leveraged to serve more users. This motivated us to implement an oversubscription policy to save computational resources and serve more requests.

The results are shown in Figure 1. The y-axis shows the cumulative number of hot nodes during the test time at different hot thresholds (45%, 55%, 65%, and 75%). Density is the average node density, calculated as $\frac{usedCPU}{totalCPU}$, which corresponds to the remaining core capacity. A smaller number of hot nodes and a larger density indicate a more effective oversubscription policy.

We observe that (1) our method results in fewer hot nodes and a larger node density. This is because we use human-in-the-loop imitation learning to accurately predict long-term user usage rates, allowing us to optimize the current oversubscription rate for more efficient results. Additionally, the risk-aware mechanism helps to quickly adjust the policy when hot nodes are detected. (2) we see that a larger node density does not necessarily mean more hot nodes. With the right oversubscription rate and risk-aware mechanism, we can effectively coordinate VMs over the long term. This indicates significant potential for optimizing the oversubscription policy. In fact, there are no hot nodes when the threshold is set to 85%, and less than 10% of clusters have a usage rate above 75%.

a transparent and better understanding of the learned policy.

B.2 Domains (Datasets)

Virtual CPU oversubscription. A Virtual Machine (VM) is a virtualized instance of a computer that runs on a physical server and accesses computing resources to perform functions (Li et al., 2010). Users purchase VMs from cloud platform providers to host applications and services. The cloud platform contains many of physical servers, *i.e.*, *Nodes*, and each node hosts a certain number of VMs.

CPU bottlenecks are more severe and common than memory and network in cloud (Mahapatra & Venkatrao, 1999). As shown in Figure 1 in main paper, three VMs are placed in the same node, and the CPU dimension is fully packed with unused memory, *i.e.*, stranded memory displayed as the grey box. If oversubscribing actual physical CPU size, additional VMs can be allocated on this node reducing stranded memory. Also, virtual CPU (vCPU) represents a portion or share of the underlying physical CPU that is assigned to a VM. Thus vCPU is the sellable billing unit in cloud platforms. So, we focus on vCPU oversubscription.

We collect real data from cloud platform for internal users, *i.e.*, responsible owners of M company services and applications. Such internal users host their services on VMs whose vCPU usage demonstrates patterns. For example, services like emails and work-related software demonstrate diurnal and weekly patterns in regions with similar time zones.

corresponding to work and social activities, resulting in the most requests and traffic income on daytime and weekdays. Thus the vCPU usage is high during these time periods while low at nighttime and weekends. On the other hand, services providing social media and gaming applications show different vCPU usage patterns when peak/hot usage happens in spare time. Moreover, other non-user-facing services running regularly, like monitoring and maintenance services, do not show daily or weekly patterns but display patterns caused by underlying configurations from service teams. The diverse vCPU usage patterns of services motivate us to adaptively oversubscribe vCPUs of VMs from each service. For example, oversubscribe services with low vCPU usage, given the context.

We collect two-week data of VM features, including the usage of vCPU, memory, and network, that belong to 30 randomly sampled services. As the vCPU usage has a lot

of fine-grained variances, we take the peak usage in the one-hour bucket as the representative data point, making oversubscription conservative. Note that in Figure 1, VMs are allocated onto nodes, and VMs from different services can be collocated in the same node. Then, we propose a simplified allocation simulator that allocates VMs via Best-Fit allocation policy (Hadary et al., 2020). VMs are allocated after vCPU oversubscription.

Other domains - Airline ticket overbooking: As noted earlier to highlight how our approach can seamlessly work on any domain to optimize adaptive oversubscription, based on utilization patterns we present Airline ticket overbooking scenario. Flight overbooking, *i.e.*, selling more tickets than the available seats, is a common practice that allows airline companies to improve their load factors and increase revenues (Nazifi et al., 2021). Yet, the difficulty in estimating ticket demands and no-shows results in inappropriate overbooking strategies, such that users with tickets cannot onboard, *i.e.*, offloaded. In general, flight ticket demands show quarter patterns that peak tourist seasons have higher flight demands (Suryani et al., 2010; Banerjee et al., 2020). This motivates adaptive oversubscription of flight tickets.

We collect airline passengers' data from the overbooking reports of the U.S. Department of Transportation (DOT). The dataset covers overbooking information of 32 airline companies in the U.S. from 1998 to 2021 [1], reported quarterly. Each quarter's data includes offloaded number of passengers (voluntary/involuntary) and the onboard number of passengers. As the overbooking rate of each airline company and the actual demands are not reported, we generate synthetic overbooking rate and flight demands to create a semi-synthetic dataset. We sample the overbooking rate within a range on 3%-5%, as per common industry practice. With the popularity of electronic tickets, no-shows are less compared to the paper-ticket period, we assume no-show rate to decay with years. Then the demand is the aggregation of bumped, no-shows, and onboard passengers. We developed a simulator trained with GBDT on the semi-synthetic data that outputs the overbooking rate of the next quarter given current quarter features.

B.3 Experiment Configurations

All experiments are performed on Ubuntu 20.04 LTS system with Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz CPU, 112 Gigabyte memory and single NVIDIA Tesla P100 accelerator. Detailed settings are listed below.

B.3.1 Base Learner:

We implemented our method by extending on top of Behavior Cloning as the base imitation learner in our experimental version. As explained later, we also use Behavior Cloning as one of the baselines. We set the learning rate as $1e-2$, MLP unites is 64, batch size is 128, optimizer is Adam. We tune the number of prototypes in $\{2, 3, 4, 5, 6\}$ and set 3 for vCPUs and 4 for airplane tickets respectively with the optimal search value. We tune the weights of different loss components within $[0.1, 0.2, 0.3, ..., 1.0]$ via grid-search. Finally, we obtained the best value on $w_1 = 0.8$, $w_2 = 0.1$, $w_3 = 0.1$, $w_4 = 1.0$.

[1] https://www.bts.gov/denied-confirmed-space

REFERENCES

Banerjee, N., Morton, A., and Akartunalı, K. Passenger demand forecasting in scheduled transportation. *European Journal of Operational Research*, 286(3):797–810, 2020.

Hadary, O., Marshall, L., Menache, I., Pan, A., Greeff, E. E., Dion, D., Dorminey, S., Joshi, S., Chen, Y., Russinovich, M., et al. Protean:VM allocation service at scale. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pp. 845–861, 2020.

Li, Y., Li, W., and Jiang, C. A survey of virtual machine system: Current technology and future trends. In *2010 Third International Symposium on Electronic Commerce and Security*, pp. 332–336. IEEE, 2010.

Mahapatra, N. R. and Venkatrao, B. The processor-memory bottleneck: problems and solutions. *Crossroads*, 5(3es): 2, 1999.

Nazifi, A., Gelbrich, K., Grégoire, Y., Koch, S., El-Manstrly, D., and Wirtz, J. Proactive handling of flight overbooking: how to reduce negative ewom and the costs of bumping customers. *Journal of Service Research*, 24(2):206–225, 2021.

Suryani, E., Chou, S.-Y., and Chen, C.-H. Air passenger demand forecasting and passenger terminal capacity expansion: A system dynamics framework. *Expert Systems with Applications*, 37(3):2324–2339, 2010.