

PUVAR: Minimize Idle Resource SLO Violations by Uncertainty-aware Scheduling in Cloud Platforms

Jiawei Li

INSC, Tsinghua University
Beijing, China

li-jw19@mails.tsinghua.edu.cn

Han Zhang

INSC, Tsinghua University
Zhongguancun Laboratory

Beijing, China
zhhan@tsinghua.edu.cn

Jilong Wang

INSC, Tsinghua University
Zhongguancun Laboratory

Beijing, China
wjl@cernet.edu.cn

Abstract—Nowadays, idle resource makes up a non-negligible fraction of datacenter capacity in mainstream cloud platforms. Cloud platforms offer idle resource with low service level objectives (SLOs) at low prices to attract cost-sensitive users. Despite their fault tolerance, these users still want some SLO guarantees for idle resource. Cloud platforms have started to provide statistical SLO for idle resource, but the violations of statistical SLOs have not been modeled and minimized. In this paper, we propose *PUVAR*, a scheduling policy based on prediction+optimization, to explicitly model and minimize the statistical SLO violation for idle resource in cloud platforms. The design of *PUVAR* possesses two major innovations: (1) explicitly quantify prediction uncertainty of idle resource capacity and iteratively reduce its impact on scheduling decisions; (2) treat the SLO of idle resources as a soft constraint and minimize its violation by two-stage scheduling of regular requests and idle requests. We provide a theoretical convergence rate for the parameter optimization of *PUVAR*. Promising results of ablation analysis by comparison with popular baseline algorithms on the trace of a real cloud platform indicate that *PUVAR* can significantly reduce the SLO violation of idle resource with little additional cost on the current scheduling of regular requests.

Index Terms—idle cloud resource, SLO violation, prediction+optimization, uncertainty-aware scheduling

I. INTRODUCTION

Due to fluctuations in user demand over time, cloud platforms usually over-provision resource at peak times while leaving a lot of resource unused at non-peak times (idle resource). To monetize these idle resource, mainstream cloud platforms launch different compute instances in succession, e.g. Amazon EC2 Spot Instance, Azure Harvesting Virtual Machine (VM), and Spot Block VM, and Google Preemptible VM. The idle resource is popular among a large number of fault-tolerant but cost-sensitive users. Despite their tolerance of low SLOs, users still want some degree of service guarantees. Recently, Microsoft has provided statistical SLO for requests running on Harvesting VMs [1], e.g. 60% probability for surviving more than 1 hour, 40% probability for surviving more than 1 day, etc. **The SLO of idle resource is defined as its eviction frequency.** However, there is no optimization

for the scheduling of requests running on idle resource, which we call idle requests.

Previous resource scheduling and optimization problems in cloud platforms mainly focused on regular resource. However, idle resource brings unique challenges. We design *PUVAR*, a new scheduling policy based on an uncertainty-aware prediction + optimization framework, which contains the following innovations:

- 1) We model idle resource SLO by eviction frequency in each request's life duration and relate it to the prediction uncertainty of idle resource capacity.
- 2) We allow users to specify SLO requirements for idle requests and formulate a combinatorial optimization problem to explicitly minimize the SLO violation and number of used machines at the same time.
- 3) We build a prediction model for idle resource capacity, design a two-stage online scheduling policy, and iteratively optimize the prediction model and scheduling policy in a combined way.
- 4) We provide a theoretical convergence rate of parameter optimization and do ablation analysis based on trace-driven experiments to demonstrate the superiority of *PUVAR*.

II. VIOLATION MINIMIZATION PROBLEM

We explicitly model an idle request's statistical SLO (eviction frequency) by the time fraction of its life duration that the cloud platform does not allocate resource for it. The offline version of the statistical SLO violation minimization problem could be formulated as follows.

$$\text{Min} \quad \sum_{i=1}^N (v_i - p_i)^+ \quad (1)$$

$$\text{S.t.} \quad X, Y = \underset{X, Y}{\operatorname{argmin}} \sum_{j=1}^M \mathcal{I} \left[\sum_{i=1}^N \sum_{t=1}^T X_{ijt} + \sum_{l=1}^L \sum_{t=1}^T Y_{ljt} > 0 \right] \quad (2)$$

$$\sum_{j=1}^M Y_{ljt} = 1, \quad \forall 1 \leq l \leq L, e_l \leq t \leq e_l + d_l \quad (3)$$

$$Y_{l j e_l} = \dots = Y_{l j, e_l + d_l}, \quad \forall 1 \leq l \leq L, 1 \leq j \leq M \quad (4)$$

This work is conducted with the support of the National Key Research and Development Program of China under Grant 2020YFE0200500. Jilong Wang is also affiliated with the Quan Cheng Laboratory, Jinan 250103, Shandong Province, China. Han Zhang is the corresponding author.

$$v_i = \sum_{t=e_i}^{e_i+d_i} \sum_{j=1}^M (1 - X_{ijt})/d_i, \quad \forall 1 \leq i \leq N \quad (5)$$

$$\hat{a}_{jt} = 1 - \sum_{l=1}^L c_l Y_{ljt}, \quad \forall 1 \leq j \leq M, 1 \leq t \leq T \quad (6)$$

$$\sum_{i=1}^N c_i X_{ijt} \leq \hat{a}_{jt}, \quad \forall 1 \leq j \leq M, 1 \leq t \leq T \quad (7)$$

$$X_{ijt}, Y_{ljt} \in \{0, 1\} \quad (8)$$

$$\forall 1 \leq i \leq N, 1 \leq l \leq L, 1 \leq j \leq M, 1 \leq t \leq T$$

$$Y_{ljt} = 0, \quad \forall l = 1 \dots L, t = 1 \dots T, j \notin \mathcal{M}_l \quad (9)$$

$$X_{ijt} = 0, \quad \forall i = 1 \dots N, t = 1 \dots T, j \notin \mathcal{M}_i \quad (10)$$

The objective is to minimize the violation of statistical SLO, where p_i is the statistical SLO of idle request i specified by user, i.e. eviction frequency no more than p_i . v_i is the actual eviction frequency and the SLO violation is quantified by the difference between v_i and p_i once v_i exceeds p_i . $X_{ijt} \in \{0, 1\}$ indicates whether to schedule idle job i , and $Y_{ljt} \in \{0, 1\}$ indicates whether to schedule regular job l on machine j at time t . Constraint (3) and (4) together ensure that the regular jobs must be scheduled and only scheduled on one machine, where constraint (3) means that Y_{ljt} for request l on machine j must be the same during its entire duration $e_l, \dots, e_l + d_l$. Constraint (4) limits that one job i can be scheduled on at most one machine j only once at its arrival time e_i . Equation (5) defines the violation frequency of idle request i . Equation (6) means for any machine j , at any time t , the total fraction of idle resource equals 1 minus the total fraction of utilized regular resource. Inequality (7) means for any machine j , at any time t , the total demand of all active idle jobs allocated on it cannot exceed the idle resource on that machine at time t . Constraint (9) and (10) limit that requests can only be scheduled on machines in its own candidate set.

III. UNCERTAINTY-AWARE PREDICTION

A. Prediction Model.

Regular user-facing workloads in large cloud platforms usually exhibit predictable trends and weekly patterns. According to our observations from trace and previous work, the total demand of regular requests in cloud platforms could be decomposed into three components: trend, weekly pattern, and irregular noise. We assume that the time series of regular demand on each machine j could be approximated by a stochastic process $R_j(t)$ and adopt the commonly used additive time series forecasting model for the stochastic process,

$$\sum_{l=1}^L c_{lj} \cdot \mathcal{I}[e_l < t < e_l + d_l] \approx R_j(t) = T_j(t) + S_j(t) + \varepsilon_{jt} \quad (11)$$

where we slightly abuse the notation and denote c_{lj} as the resource demand of regular request l on machine j and let it equal 0 if j is not in l 's candidate machine set. The value of $R_j(t)$ denotes the total amount of CPU demand at time t , $T_j(t)$ is the deterministic trend, $S_j(t)$ is the deterministic cycled weekly pattern and ε_t is the irregular noise at time t , which is a random variable. A set of parameters $\beta_{jj'}$ are fitted in a linear model to describe the relation from regular demand to regular utilization on each machine.

For the estimation of trend $T_j(t)$, we adopt a simple and effective *Moving Average Smoothing*. Denote the last d historical samples as $\hat{R}_j(t-d), \dots, \hat{R}_j(t-1)$, the moving average (MA) is

$$T_j(t) = \sum_{\tau=t-d}^{t-1} \hat{R}_j(\tau)/d \quad (12)$$

where d is the parameter of time period in MA model. For the weekly pattern, we extract a weekly template from historical data and repeatedly apply the template as new week's pattern prediction. For the irregular noise, we fit its distribution by the *Gaussian Mixture Model* (GMM).

B. Prediction Uncertainty Quantification.

The prediction uncertainty is introduced by the irregular noise, which is generated from a GMM whose probability density function (pdf) is

$$f(\varepsilon_{j't}|\Theta) = \sum_{k=1}^K w_k f_k(\varepsilon_{j't}|z_k, \theta_k) \quad (13)$$

where each $f_k(\varepsilon_{j't}|z_k, \theta_k)$ is the Gaussian probability density of a mixture component with parameters $\theta_k = (\mu_k, \sigma_k^2)$. $z = (z_1, \dots, z_K)$ is a vector of K binary indicator variables that are mutually exclusive and exhaustive, i.e. one and only one $z_k, k = 1 \dots K$ equals 1 and others equal 0. $w_k = P(z_k = 1)$ is the mixture weight, representing the probability that a sample ε_t is generated by component k , and $\sum_{k=1}^K w_k = 1$. The complete set of parameters of the GMM is

$$\Theta = \{w_1, \dots, w_K, \theta_1, \dots, \theta_K\} \quad (14)$$

and the cdf is

$$F_{j'}(p) = P(\varepsilon_{j't} \leq p) = \int_{-\infty}^p f(\varepsilon_{j't}|\Theta) d\varepsilon_{j't} \quad (15)$$

We use the probability that the actual idle capacity is less than the predicted capacity to approximate the resource eviction probability, i.e.

$$\tilde{a}_{jt} = \sup\{\gamma | P(a_{jt} < \gamma) < p_i\} \quad (16)$$

Integrate constraint 16 in the violation minimization problem and the problem becomes

$$\begin{aligned} P(\hat{a}_{jt} \geq \gamma) &\geq 1 - p_i \\ \Rightarrow P(1 - \sum_{j'=1}^M \beta_{jj'} \cdot (T_{j'}(t) + S_{j'}(t) + \varepsilon_{j't}) \geq \gamma) &\geq 1 - p_i \end{aligned}$$

$$\begin{aligned}
&\Rightarrow P\left(\sum_{j'=1}^M \beta_{jj'} \varepsilon_{j't} \leq 1 - \sum_{j'=1}^M \beta_j T_{j'}(t) - \sum_{j'=1}^M \beta_j S_{j'}(t) - \gamma\right) \\
&\geq 1 - p_i \\
&\Rightarrow \tilde{a}_{jt} = \sup\{\gamma | P(\hat{a}_{jt} \geq \gamma) \geq 1 - p_i\} \\
&= 1 - \sum_{j'=1}^M \beta_{jj'} T_{j'}(t) - \sum_{j'=1}^M \beta_{jj'} S_{j'}(t) \\
&\quad - \sum_{j'=1}^M F_j^{-1}((1 - p_i)/\beta_{jj'}) \quad (17)
\end{aligned}$$

Since $\tilde{a}_{jt} = \sup\{\gamma | P(\hat{a}_{jt} \geq \gamma) \geq 1 - p_i\}$ holds for all requests i allocated on machine j , it holds that $\tilde{a}_{jt} = 1 - \sum_{j'=1}^M \beta_{jj'} T_{j'}(t) - \sum_{j'=1}^M \beta_{jj'} S_{j'}(t) - \min_{p_i} F_j^{-1}((1 - p_i)/\beta_{jj'})$ where i is any request that is allocated on machine j and F^{-1} is the inverse cdf of GMM. This suggests that the more strict the SLO requirement (the smaller the eviction probability p_i), the more conservative the estimation of idle capacity.

IV. ONLINE SCHEDULING FRAMEWORK

In the online setting, regular requests and idle requests arrive on the fly. We integrate the prediction into the optimization problem by transforming the eviction probability to capacity constraint using the cumulative distribution function (CDF) of the distribution. The framework contains a prediction module and a scheduling module. The scheduling module is divided into two stages: scheduling regular requests (in priority) and scheduling idle requests. The prediction module and scheduling module work together in an iterative way to reduce SLO violation for the idle resource.

A. Schedule Requests.

Currently one of the primary concerns of scheduling in cloud platforms is to improve the machine utilization, i.e. minimize the total number of used machines and leave as many empty machines as possible, which is usually achieved by the Best-Fit packing algorithm. However, for the consideration of minimizing idle resource SLO violation, we want to even out the regular utilization to reserve more idle resource capacity. The well-known algorithm to achieve this goal is the Worst-Fit packing algorithm. We reduce SLO violation of idle resource into account when scheduling regular requests, without much increase in the number of used machines, by a combination of using the Best-Fit and the Worst-Fit algorithm. With a small probability α , the scheduling of regular requests uses Worst-Fit, and with probability $1 - \alpha$ the scheduling uses Best-Fit.

B. Iterative Optimization.

After each round of scheduling, PUVAR optimizes parameter α in the scheduling policy. We adopt Bayesian optimization to optimize the relaxation probability α . The optimization objective is not traditional prediction accuracy,

but the eventual performance of the online scheduling algorithm: SLO violation, $f(\alpha) = \sum_{i=1}^N (v_i - p_i)^+$, which is black-box and expensive to evaluate (need actual scheduling). The key implementation of Bayesian optimization relies on a *surrogate function* to approximate the objective function, and an *acquisition function* to direct the next sampling point. We choose the widely used *Gaussian Process Regression* (GPR) equipped with *Radial Basis Function* (RBF) kernel as surrogate function and the most popular acquisition function *Expected Improvement* (EI) as the acquisition function,

$$EI(\alpha) = \begin{cases} (f(\alpha^+) + \xi - \mu(\alpha))\Phi(Z) - \sigma(\alpha)\phi(Z), & \sigma(\alpha) > 0 \\ 0, & \sigma(\alpha) = 0 \end{cases} \quad (18)$$

where

$$Z = \begin{cases} \frac{\mu(\alpha) - f(\alpha^+) - \xi}{\sigma(\alpha)}, & \sigma(\alpha) > 0 \\ 0, & \sigma(\alpha) = 0 \end{cases} \quad (19)$$

Here μ and σ are the mean and standard deviation of posterior GPR at point α using available samples. α^+ is the best point in historical samples. Φ is CDF of standard Gaussian distribution. $\xi = 0.01$ is the trade-off weight of exploration and exploitation. The next sampling point of α is chosen to maximize the EI, i.e. $\arg \max_{\alpha} EI(\alpha)$, at which point the objective value is small (exploitation) or the uncertainty is high (exploration).

Proposition 1: The convergence rate of PUVAR's optimization of α is $O(1/n)$, where n is the number of iterations.

V. EXPERIMENT

We evaluate the performance of our proposed PUVAR scheduling policy on **AzureTracesForPacking2020** [2]. The fluctuation of an error of our prediction model on test data (the second week) falls below 2%, and the 95th percentile of the fluctuation is less than 1%. Compared with purely Best-Fit, prediction without noise, and scheduling without iterative optimization, PUVAR does not affect the number of used machines much, even compared with only using Best-Fit, but significantly reduces the SLO violations.

VI. CONCLUSION

In this paper, we present PUVAR, a scheduling policy based on an uncertainty-aware prediction + optimization framework. PUVAR achieves a significant reduction of SLO violation of idle resource without much increase in the number of used machines, suggesting that cloud platforms can potentially improve idle resource user experience significantly at little additional cost.

REFERENCES

- [1] P. Ambati, I. Goiri, F. Frujeri, A. Gun, K. Wang, B. Dolan, B. Corell, S. Pasupuleti, T. Moscibroda, S. Elnikety, M. Fontoura, and R. Bianchini. Providing SLOs for Resource-Harvesting VMs in cloud platforms. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 735–751. USENIX Association, Nov. 2020.
- [2] O. Hadary, L. Marshall, I. Menache, A. Pan, E. E. Greeff, D. Dion, S. Dorminey, S. Joshi, Y. Chen, M. Russinovich, and T. Moscibroda. Protean: VM allocation service at scale. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 845–861. USENIX Association, Nov. 2020.